Interested in learning more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Examination of PC security: How we got where we are and how to fix it

This essay explores the reasons for the poor state of PC security that currently exists. This essay focuses on the end users rather than the administrators. Threats and solutions are examined form an end-user's perspective.

Copyright SANS Institute
Author Retains Full Rights

AD

Examination of PC security: How we got where we are and how to fix it

Thomas Sprinkmeier

September 23, 2004

Contents

1 Abstract **2**

2 Mainframe days **2**

3 Personal Computers **2**
3.1 Standalone . . . 2
3.2 Networking, BBS, intra-nets and the Internet . . . 2
3.2.1 Security through "user unfriendly"-ness . . . 3
3.3 Over-eager servants . . . 3

4 Problems **4**
4.1 Abuse of trust . . . 4
4.2 Configuration . . . 4
4.3 Complexity . . . 4
4.4 Bugs and Vulnerabilities . . . 6
4.5 Security through obscurity . . . 7
4.6 Hostile Environment: the WWW . . . 7
4.7 Insecure by design . . . 8
4.8 The inevitable car analogy . . . 9

5 Solutions **9**
5.1 Onwards to the Past . . . 9
5.1.1 Back even further . . . 10
5.2 Regulation/Accountability . . . 10
5.3 User Education . . . 11
5.3.1 Impartiality . . . 11
5.3.2 Profitability through gullibility . . . 11
5.3.3 Context of knowledge . . . 11
5.3.4 Apathy . . . 12

6 Conclusion **13**

A Bibliography **14**

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

1 Abstract

This essay explores the reasons for the poor state of PC security that currently exists. This essay focuses on the end users rather than the administrators. Threats and solutions are examined form an end-user's perspective.

2 Mainframe days

A 'Computer' used to be a Mainframe, sitting in a dedicated room with acolytes tending to its every need. The Mainframe was a central resource, surrounded by 'dumb' terminals.

Users were passive consumers of the resources available — unable to install or configure the system. Interactions between users was limited.

The computer was a valuable resource, and it was treated as such.

The terminals available to the users had very limited capabilities. Who would want an Anna Kournikova screen-saver on a 80x25 green-screen-terminal? Who could afford to store it with hard-disk costs at hundreds of dollars per Megabyte?

Computing was about centralisation: Central processing, central administration, central control, central monitoring. End users were little more than passive consumers of the resources provided. They could run programs already installed and query databases managed by others.

At the end of the month, every CPU cycle, byte of RAM and inch of backup tape was accounted for and billed. Security was partially achieved as a byproduct of accountability.

The system was secure because the users had no control. Passengers [1] on a bus, train, or plane cannot affect the safety of the vehicle. In mainframes and mass transport vehicles, security is achieved by keeping the consumers away from the controls.

3 Personal Computers

Along came the "Personal Computer". The idea of a 'personal' computer was as alien as the idea of a 'personal' plane is today, surely this is just a play-thing for the rich and eccentric hobbyists?

3.1 Standalone

From a user perspective, the first encounters with PC's were likely to be of the standalone type.

Except for a few PC's actually used as 'dumb terminals', most would have been isolated from each other.

These PC's became personal little islands of creativity.

Disconnected from each other, PC's were relatively safe. Viruses tried to spread through infected floppy disks, but these were easy to scan. Virus propagation and evolution was so slow that having a virus scanner was enough, there was little need to constantly update the virus definitions.

Most people shared files with only a few others. This community quickly identified untrustworthy or careless members, further limiting virus propagation.

3.2 Networking, BBS, intra-nets and the Internet

PC's were eventually networked, mostly to allow sharing expensive resources like printers. File sharing, via dedicated servers, Bulletin boards and this new thing called the Internet started to become more and more commonplace.

With the promise of lower administration costs, greater personal freedom, and being able to tell the megalomaniac administrators to take a hike, PC's started proliferating in the workplace and, eventually, even at home.

A hobbyist might sporadically connect his computer to Bulletin Board Services (BBSes). A corporate user might be in a corporate LAN, sharing access to a printer. A university user might have a connection to the Internet.

Theoretically, viruses could now spread much faster, but they didn't: Computers were still passive, requiring user intervention to run malware.

[1] Discounting hijackers and other miscreants

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- It was easy enough to scan programs you received: it was such a lengthy manual process to get them in the first place, using WAIS [2], gopher, ftp, and the like, that virus scanning was just another step in that process.
- Viruses were still spreading and evolving so slowly that an outdated virus scanner was usually sufficient.
- Resources like disk space, memory, and bandwidth were so scarce that all but the tiniest increase in the size of a file would be noticed.
- PC's were unique, offering few services and all in a different way. Imagine a virus that tried to activate the printer. Imagine trying to cope with network printers, parallel port printers, serial port printers and who knows what else, all without a common printing interface?

Viruses therefore had to be carefully and expertly crafted and were very limited in what they could do.

3.2.1 Security through "user unfriendly"-ness

At university I marveled at the patience and persistence of a friend who used to download images from newsgroups. These images were 'uuencoded' (converted to ASCII, newsgroups didn't support binary attachments), split into dozens of parts (message sizes were severely limited) and posted over a matter of days to a newsgroup. My friend would individually download (at 1200bps!), re-assemble and decode these images obtaining, after some hours of effort, a picture of the Brandenburg gate, or a mountain scene, or an exotic flower in bloom. Whenever his hard disk overflowed he would have to archive to innumerable floppy disks.

The point of this reminiscing is that obtaining binary data from other computers used to be difficult. By the time your learned enough about computers to be able to exchange files, you also learned enough, by osmosis if nothing else, to be careful while doing so. No one would spend hours downloading a 'picture' with a *.exe* extension and blindly execute it, much the same a model builder would not accidentally build a model plane instead of a boat, and then put it in the water.

Things used to be more difficult. You had to know which file extension went with which viewing program. Today these details are deliberately hidden from users in an effort to make computers "user friendly".

Unfortunately, computers aren't smart enough to differentiate between *users* and *abusers*, so features to make the system easier to *use* usually make it easier to *abuse* also.

As an example, hiding file extensions means less confusion, but also greater risk of inadvertently executing malware [3]. If you're trying to trick a user into running your malware, "click here to see a picture" works much better than "download these 5 parts, concatenate them, decode them, ignore the fact that this picture is actually a .exe file and then run it to see a picture".

3.3 Over-eager servants

Then there's the fact that Computers are no longer passive. Computers constantly do things on your behalf in order to assist you and be more "user friendly".

They automatically open attachments, they download and run scripts, they fill in fields on web pages, they automatically decompress file archives to make them easier to browse. Most people are only vaguely aware of all the things that a typical computer does on their behalf. Computers try to predict your instructions and execute them on your behalf.

As if that wasn't enough, computers now actively listen for instructions from others as well by running file and print sharing services, listening for incoming messages and offering services to whoever asks.

PC's were mostly passive, waiting for instructions from a user before doing anything. The only way to run malicious code on a PC was to trick the user into running it for you. Now-days computers are hyperactive, jumping at the chance to help anyone do anything. 'Computer Security' used to be summed up as "If you're silly enough to run a program off a floppy disk, you deserved it". Today computer security is about curbing your PC's enthusiasm.

[2] Wide Area Information Service, a text-pased precursor to the HTTP
[3] 'W32.Sircam.Worm@mm' is one of the many viruses that does this.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

4 Problems

4.1 Abuse of trust

Misplaced trust is a dangerous thing. Most users are unaware of the blind trust their computer places in others.

This is a dangerous combination. Take the simple example of email.

- The Directory Name Service translates the human-friendly hostname (*host.example.com*) to a computer friendly IP address (*10.12.42.197*). This service is vulnerable to DNS spoofing [1].
- Routing is used to deliver the packets to their destination. Various attacks can lead packets astray or be used impersonate others [3].
- IP headers in packets contain the source and destination address. These addresses can be faked [19].
- The Simple Mail Transfer Protocol [31] trusts all the information supplied about senders, recipients, reply addresses. Imagine how easy it would be to eradicate spammers if reliable return addresses were provided!

The end-points of the SMTP transaction are not verified, opening the door to man-in-the-middle attacks [2]. There may be other attacks [4] as well, depending on your network topology and where the 'bad-guy' is.

Given all this, it's clear that an email is about as trustworthy as a note scribbled on a postcard dropped near your letterbox. *So why do so many people blindly trust emails?*. Virus writers and phishers [5] depend on users being unaware of the limitations of the technology they're using.

4.2 Configuration

Security, or lack thereof, is often more a matter of how something is configured than how well it was made in the first place. "Factory Defaults" are often left unchanged, so they have a huge impact.

The default installation of many UNIX-es left certain services and user accounts in an unsafe state[7]. Microsoft products have a long history of unsafe defaults.

The default settings are often a trade-off between usability and security. At one extreme, turning on all services will make a system easy to use, but too insecure. The other extreme is a system that's so tightly secured as to be almost unusable. Neither of these is likely to result in a happy customer.

Trying to achieve both is doomed to failure due to the different environments that the system will be used in. A PC on a corporate LAN might reasonable be configured with minimum security, while the same PC on a university LAN (or worse yet, the Internet) should obviously be a lot less trusting.

The best compromise seems to be a machine that is secure by default but easy to reconfigure (using Dialog boxes, 'Wizards' etc.). One danger of this approach is that if the reconfiguration tool is itself subject to the security vs usability trade-off: too secure and it will be too hard to use, too friendly and it may lead to unsafe configurations.

Take as an example a hypothetical Wizard to enable file sharing. If it asks which subnets or networks interface to enable for file sharing it fails the usability test as users will have to go out and learn what those things are [6]. If it doesn't ask and enables file sharing on all interfaces to all hosts it fails the security test by leaving an insecure configuration. A third option, using phrases like "locally connected computers" is going to confuse some [7] while annoying others [8].

4.3 Complexity

> *Dijkstra's principle:* Programs are either are so simple that they obviously contain no bugs, or so complex that they contain no obvious bugs.

[4] ARP poisoning,

[5] (fishing) (n.) The act of sending an e-mail to a user falsely claiming to be an established legitimate enterprise in an attempt to scam the user into surrendering private information that will be used for identity theft.[11]

[6] Most users will accept the defaults, reviving the problem (picking defaults) that the Wizard was meant to solve.

[7] "Is a computer 'local' if it's in the next room?"

[8] "Does that mean 'local subnet'? How do I allow WAN access?"

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

People often bemoan the complexity and inherent unfairness in our laws. It seems that if you have enough money, you need not pay taxes [26], that apparently you can even get away with murder [4].

While many people cry for more and stricter laws, cooler heads realize that more laws only mean more loopholes, and stricter punishments only lead to more appeals by those who can afford them. In the end, both of these measures only result in less equality.

The same is true of security in computing. Faced with a multitude of threats, some are convinced that the answer is more protection, more code, more options and more control.

Microsoft Internet Explorer [9] has dozens of security related options:

Allow javascript? Allow Java? If so, which virtual machine? Allow this kind of cookie or that kind of attachment? Request signed applets?

All of these options apply to a number of security classes — the idea being that the trustworthiness of a given site determines which class it belongs to and, by extension, what privileges it has on your computer.

Even if you figure all this out, along comes an unexpected attack vector [28] [8] that has no corresponding security options because it exploits an unexpected or obscure vulnerability. Who would have thought displaying a picture was dangerous?

Even assuming that all this works, *is it useful?*

Some sites may require javascript but abuse cookies. Some sites may make appropriate use of cookies but have questionable ActiveX applets. Some may require ActiveX to navigate efficiently while abusing javascript.

There are too many combinations to allow a simple set of blanket rules about what kind of behavior is desirable. The result is that the complexity of these rules leads to the requirement of even more complexity to adequately apply them.

Ultimately there are so many rules, options and settings that no-one bothers to even look at them however, paradoxically, knowing they exist bestows a false sense of security. *In the end, security is degraded by the illusion of having it.*

But what causes this complexity?

The fusion of Data and Code: One of the worst offenders for increasing complexity is "the fusion of data and code".

Data used to be static — something to be loaded into a program, manipulated, utilised, modified and saved.

Code used to be equally static — loaded from disk, executed but never modified, especially by the user.

In an effort to "enhance the user experience" the line between these two concepts has been blurred. Previously static documents can now contain executable code, usually in some macro language. As the Melissa [5] virus (among others [16]) demonstrated, this kind of feature (usually used by a tiny fraction of users) is a dangerous tool in the arsenal of the malware writers.

The same is true of web-browsing with the addition of Java-script, Java, ActiveX, Flash,, ASP, PHP and the like.

Interactions: The problem is often not any single feature, but the interactions between them and the way these can be abused by others to their own ends.

- Being able to automatically launch the correct application to handle an attachment is a nice idea.
- Being able to execute some macros when opening a document is a nice idea.
- Being able to send a document as email straight from the word processor is a nice idea.

Three "user friendly" ideas that, when combined result in an "abuser friendly" system. It's not the individual features that pose the risk, rather the interaction between them. Modern computers are full of such sets of interactions, and the number of interactions increases geometrically as the number of subsystems increases.

[9] MS IE is certainly not the only product with this problem, but an obvious example.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Convergence Increased complexity is often the result of adding new features. This applies to individual programs as well as to complete systems.

There is a trend of 'converging' many single-use devices into multi-use ones:

- Mobile phones taking on PDA-like features, image and sound recorders
- PCs extending into the home entertainment arena
- Internet enabled refrigerators [10].

Even when not combined into a single unit, separate devices are often interconnected. This leads to:

- More interactions: As can be seen above, interactions can be difficult to predict and impossible to secure. Once your humble VCR has turned into a PC/HiFi/DVD-player, what's to stop someone including a 'bonus track' on their rental movies that uses the Internet to report your viewing habits?
- More attack vectors: An analogue mobile phone was simple and robust. Some modern mobile phones are vulnerable to malicious SMS messages [18] and BlueTooth attack [11].
- Unexpected capabilities: Who would think to install and update a firewall and virus checker in a refrigerator? The Internet access capability is likely to leave a lot of people out in the cold (pun intended) when it comes to security simply because they don't realise the precautions necessary to protect their icebox-with-delusions-of-grandeur.
- Unwanted capabilities: It is now commonplace to ban mobile phones from change rooms and other privacy sensitive areas. Why? Because some of them may have cameras installed. I end up inconvenienced for a feature I don't even have! How long before I get frisked on entering a cinema [12] lest I have a video-capable mobile phone?
- Single point of failure: Having one device that does everything means nothing is available if that one device fails.
- Greater consequences: Had a bad day today. I forgot to enable the firewall on my palmtop-PC-mobile-phone when I used it to open the garage door. It caught a virus via BlueTooth off the neighbours lawnmower. Now my garage door won't stop opening and closing, the fridge has turned into a sauna and won't stop ordering orange juice, the washing machine is shredding everything, my PC ate my tax records and the TV is stuck on "I love Lucy" reruns! Oh, and the car is still sulking about the scheduled service being late. It keeps reporting itself as stolen.

  Impossible? Imagine your Digital camera downloading straight to your home-entertainment-PC. All your treasured family snaps conveniently on disk, rather than securely in a photo album. Now imagine a virus wiping out the lot.

4.4 Bugs and Vulnerabilities

All non-trivial programs have bugs [9].

Thanks to increasing complexity and interactions within your computer, almost any bug has a chance of becoming a vulnerability.

Even if there were no bugs at all in software, it's dangerous to let the computer decide for you what to do. Take web-bugs as an example: the computer, trying to be helpful and user-friendly will automatically load images contained in emails. A web-bug is a special, usually invisible picture that, on downloading, confirms not only that you've opened your emails, but can provide others with all sorts of information about you. Then end result is that previewing SPAM confirms your email address as being active and invites more SPAM!

[10] personally I'd prefer an Internet enabled microwave, allowing to you watch TV white heating a TV dinner!

[11] Undoubtedly there will be another attack vector by the time you read this!

[12] Banning mobile phones from Cinemas isn't altogether unappealing, but with two small children a vibrate-only phone is very reassuring!

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

4.5 Security through obscurity

In cryptography an algorithm and a secret (known as a key) are combined to turn plain-text into cipher-text [13]. One of the fundamentals of cryptography is that if the algorithm needs to be kept secret in order to protect the cipher-text then the algorithm is really part of the key. The implication is that by supplying someone the necessary algorithms, you are also supplying them part of the key required to break all the cipher-text encoded with that algorithm.

In cryptography security is achieved by dispersing the algorithm widely and subjecting it to peer review. Claims of 'super secret secure algorithms' are met with suspicion at best. Usually these claims aren't even taken seriously enough to ridicule.

In software development there are two trains of thought:

- Opening the source code to peer review makes the product stronger by allowing bugs to be found.
- Keeping the source secret makes the product stronger by denying the 'bad-guy' information about the bugs in the code

These arguments lead to a fierce and at times fanatical debate over the merits of open vs. closed source. Deciding who's right is left as an exercise for the reader.

The relevance to security of these arguments is that the closed-source way aligns itself with security-by-obscurity. The argument boils down to this: if a bug is discovered it's best to keep the details hidden lest some evil-doer finds out how to exploit the bug. This argument is not without merit — there are fears that patches, even binary patches, are being reverse-engineered [20] to discover the original weakness, allowing un-patched systems to be attacked. Here lies support not only for the argument (information in the wrongs hands is dangerous), but also the counter-argument (information *will* find it's way into the wrongs hands).

Withholding information about vulnerabilities benefits the vendor by allowing their systems to appear more secure. Every Linux Kernel bug is available for all to see, evaluate and count. Microsoft often releases 'cumulative patches'. Without knowing details about which bugs were cumulatively fixed it's impossible to judge, not only how secure a system is, but whether or not security is improving or degrading as time goes on.

Critical bugs can be hushed up until a fix is available. Apparently fixing critical bugs quickly makes the vendor look good. Withholding information about bugs from customers denies them the opportunity to take mitigating steps — if some product has a critical, unfixed bug I'd like to know about it so I can decide whether or not to keep using it until the bug is fixed!

The problem with this thinking is that it places all the power in the hands of the company. Companies make decisions based on their own best interests, not necessarily yours! Security is too important for you to allow someone else to make these decisions for you.

Obscurity isn't a bad defense, but it shouldn't be your only defence[12]. Imagine a law requiring everyone to carry a flashing light at night. It would surely improve security, as muggers would have a hard time sneaking up on people. Pretending that the bad-guys are hamstrung by 'closed source security' is like pretending that the muggers won't disable the flashing lights. The law-abiding suffer (through lack of information required to protect themselves, or by having to carry a mugger-attracting flashing lights) and the the law-breakers are, at best, unaffected.

4.6 Hostile Environment: the WWW

The Internet is a wonderful place. It allows all people, everywhere on the globe to reach out and interact, freed from the limitations of geography, boundaries and distance.

Pity some of the people seeking to 'interact' aren't all that friendly!

At the time of writing this article, a fresh Microsoft Windows XP install has a survival time of less than 20 minutes [34]. Imagine a world were your new car could be expected to be involved in an accident within 20 minutes of leaving the dealers lot. Now THAT's a hostile environment!

The Internet has brought us many things, and most of them have been abused to launch attacks on us. Email, web-browsing, on-line chatting, file sharing — if your computer is connected to the Internet, it's at risk. Even the humble PING is a tool for evil [23]!

[13] Assuming symmetric cryptography

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

A reasonably prudent person would face such an environment asking "How can I protect myself?". Unfortunately, a Marketing department when faced with such an environment asks "What bells and whistles can I add to my products to attract customers?".

How about a mechanism to download and execute code automatically?

4.7 Insecure by design

Security must be part of the initial design of a system. Retrofitting security to a bad (from a security standpoint) design is usually futile. Security considerations compete with other requirements during the design phase, so as usual it's a tradeoff.

Let's examine these tradeoffs in the design of ActiveX [14] Controls and Java [10] Applets. Both of these technologies provide 'mobile code' via a web-browser interface — you visit a web-page and an executable is downloaded to your system and run locally.

Both systems provide a mechanism for allowing or denying code execution based on the origin of the code. Both allow verification of digital signatures. In real-world terms, it's like a guard at the door who will check your papers, and verify your passport, before letting you through the door. Both technologies can revert to asking the user whether or not to allow code to be executed if the question cannot be resolved by code origin and digital certificate validation.

At this point, ActiveX security grinds to a halt. If the code is allowed to run, as a result of preconfigured access rules or user permission, it executes on the local machine with all the permissions and access rights of the user. Any action the user can make, the ActiveX control can make as well — deleting files, opening network sockets, changing permissions on files, the list is quite extensive as most users log onto Windows as *Power Users* or *Administrators*[14].

In the guarding-the-house analogy, you have a guard at the door with a guestlist and, possible, instructions to check passports. If unsure about whether or not to admit someone, the guard will call you on the intercom. Once inside, the guest has complete and unrestricted access to all parts of the house that you have access to. The guest can even open a back door and invite friends in [15].

Java security goes further. Once authorised to execute (by the guestlist of user-intervention), the downloaded code is run in a virtual machine under close scrutiny of a security manager. All actions are checked against access policies before being allowed or denied. The default access policy is very restrictive.

In the house analogy, having decided that a guest may enter the house, the guard now follows the guest everywhere, observing everything he does. If anything the guest does looks like it may break the permissions assigned, the guard takes immediate, preemptive and painful action [16]. The default rules allow the guest access to the foyer, where there's a phone programmed to allow calls only to the guest's home and a one-way mirror through which the owner of the house can watch.

ActiveX security is *weak by design*. Even if implemented perfectly, even if digital certificates couldn't be faked [29], the *design* has two serious flaws:

1. It relies on the end user to decide whether to run code of not, and
2. It allows no control over the behaviour of the code once it's allowed to execute

Effectively there is only one layer of defence — the user. Most users are gradually, sometimes painfully, learning about computer security. There will, however, always be those who answer SPAM emails, download free Buffy screensavers, execute emailed securty patches and authorize ActiveX controls.

The Java security model has the same weakness of handing the decision about running the code over to the user. However, the risk is mitigated by having a second line of defence — the behaviour of the program is then monitored and, if necessary, curtailed. It doesn't matter so much if you fool the guard at the door, he's still watching you once you get in.

The problem with the Jave model is that, to get a Java applet ro do anything 'useful' (like opening local files), the user has to customize the default restrictions to allow these actions. Most people don't do this, so Java applets are seen as having very limited functionality.

The ActiveX model results in quick easy functionality at the cost of security. Clearly the design tradeoff was increased user-friendliness at the cost of security.

[14] These are access groups that control permissions in Windows

[15] This is how alot of spyware is installed

[16] Imagine a psycpathic clippy: "Hi, it looks like you're trying to open that cupboard. Allow me to flatten the hair on the back of your head with my club!"

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

4.8 The inevitable car analogy

Discussions on computers seem to inevitably end in analogies involving cars.

What mileage would we get if cars had kept pace with PCs? What would the road toll be if car stability was on par with PC stability?

Most of these analogies break down because they fail to take into account some fundamental differences:

- Threat to life: Accidents involving cars can, have, and will continue to kill. The assumption is that security defects in PCs can't [17], haven't [18] and hopefully won't kill people. But how much longer will this be the case?

- The Economics of cars and software are vastly different. The incremental cost of software is negligible [19], unlike the cost of producing a car. While both have significant developmental cost, making copies of a CD and printing a cardboard box is a lot cheaper than building a car. The field is even more skewed when it comes to fixing mistakes. Developing a patch for faulty software and the associated regression testing may be expensive, but the unit-cost to apply the patch is negligible. A defect in a car has to be expensively rectified in every car.

- Profitability requires scarcity. Having bought a car, no-one is likely to deprive themselves of it again by just giving it away. On the other hand, software can be copied without loosing the original. Scarcity in the software market is artificially created using abstract concepts like 'copyright', 'intellectual property' and 'anti-piracy' measures like serial numbers and product activation.

- Personal computing is dominated by one player holding over 90% of the desktop operating system market. Companies exist only to make profit, where *profit = number of customers * profit per customer*. Customer satisfaction and profit per customer are usually traded off against each other: keeping customers happy is expensive, and high prices make for unhappy customers. Customer satisfaction and market share are usually directly related.

  In a competitive market there is a balancing act. Customers must be satisfied, otherwise they leave, but satisfying customers is expensive so it's best not to overdo it. Generally, a successful company is one with happy customers.

  In a monopoly market customer satisfaction is much less important. The way to increase profits is to increase the market (or encroach on other markets) and to judge just how much you can charge before your customers revolt.

  The market forces in the software world are different from the car world. Commercial organisations make decisions based on market forces, so understanding these forces, and putting them in the right context, is vital. Even if the car analogy were a closer match in other respects, the distortion brought on by the dominance of the software market by one player is enough to render most comparisons meaningless.

5 Solutions

5.1 Onwards to the Past

We could revert back to Central Control. A PC used to be a company resource, used in an approved manner by approved users.

'Trusted computing' [25] seeks to remove user control from the PC. Actions can be centrally checked and approved, or denied, based on company policies, just like in the Mainframe days. Your 'personal' Computer will become a glorified dumb terminal.

Security would be improved, but at what cost? This is the equivalent of mandating trained chauffeurs for every 'personal' car in an effort to lower the road toll. Of course, these trained drivers would only take you to destinations with approved parking lots. For your own safety.

> *Lord Acton:* Power tends to corrupt; absolute power corrupts absolutely [33].

[17] At least not directly
[18] At least not many [21]
[19] Especially now that printed manuals are a faint memory

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

This technology would place immense power in the hands of a few entities. A company exists only to make money, which is usually achieved by having many happy customers. But what if there was a better way to make money than pleasing pesky customers? Do you trust big companies to be altruistic enough to forgo immense profit because it 'would be the wrong thing to do'?

I believe that the real reason for this technology is to allow increased control over PCs. The corresponding decrease in control by the 'owners' may result in better security, but only as a side-effect.

A more subtle form of centralised control is the automation of software updates [20]. Changes can be brought in piece by piece, and not all changes need to be in software. Windows 2000 Service pack 3 had a significant change to the End User License Agreement (EULA) giving Microsoft *more* control over the PC than owner!. Public outcry reverted the changes, but they reappeared in Windows XP Service Pack 1 [30]. Security updates to Microsoft Media Player are bundled with Digital Rights Management (DRM) [22].

This is definitely the direction 'big business' wants to go, and unless consumers are careful we'll get it whether we like it or not.

5.1.1 Back even further

Back in the Wild Wild West bounties were offered to capture criminals. Some seem to think that what's good for one WWW is good for another and have offered bounties on the heads of World Wide Web criminals [13] [24].

Justice should not be a 'you get what you pay for' commodity. The amount of protection you get from wrongdoers shouldn't depend on the size of the bounty you can put on their heads

5.2 Regulation/Accountability

Most other industries have some form of regulation or accountability. You'll never find as many bugs in a box of cereal as you will in a box of software. When you buy a car you don't get a 20-page License Agreement full of dense legalese absolving the car company from any and all responsibility for their product [36].

Laws and regulations are passed to protect us from ourselves, each other and from vendors. Seat-belts are mandatory, roads have speed limits, laws are passed to minimize the harmful effects of passive smoking. So why not a few laws to protect us from buggy software?

The reason given for the appalling lack of consumer protection in the software market is that 'software isn't dangerous'. But that's not entirely true. Software defects have undeniably caused death [21], albeit in special circumstances. The question is, with the all-pervasive nature of computers today, what other dangers are there?

- The WebTV [32] attack overloaded emergency switchboards. How many critical calls were delayed in situations were minutes count?
- While not directly causing the August 2003 blackout in the US, the Sasser worm was blamed by some for impeding efforts to restore power. If this is true, is it unreasonable to assume that leaving 50 million people without power for an extra hour or so might cause fatalities? How many unfit couch-potatoes were done in by the unaccustomed efforts imposed by having to entertain themselves when the TV went dead? How many calls for help were prevented by a cordless-phone base-station without power? How many broken bones were caused by trips on unlit stairs?
- Many people are foregoing conventional telephones in favour of VoIP technology. How will they fare when the next-big-worm brings the Internet to its knees?

It's not a single flaw, or a single instance of software that's dangerous. The danger lies in the enormous effect such a single flaw can have, and the side-effects leaking from the virtual world into the real one. A faulty TV console won't kill you, but a million TV consoles jamming emergency switchboards can if you happen to be in need of an ambulance.

Even if safety critical software systems, like your cars anti-lock braking system, can be adequately secured, what about your in-car entertainment system? Imagine a million in-car stereos suddenly blaring "look out behind you!" and the resulting chaos caused by a million distracted drivers.

[20] The ultimate form of this would be Software Leasing[27]

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Industry hates government regulation. They'd much rather have no regulation or, failing that, self-regulation (also known as 'foxes guarding the hen-house' regulation). Imagine, however, the outcry if the next Melissa virus or Sasser worm managed to infect up-market SUV's and kill hundreds of rich people. In typical knee-jerk fashion the government might just clamp regulation on the industry.

5.3 User Education

I believe that user education is the best alternative to pursue, though not without its problems.

Education has to be impartial, else it becomes marketing. Education for a fast-moving field like this must be constant, or else it quickly becomes irrelevant. Lastly, companies don't actually like educated consumers.

5.3.1 Impartiality

Computer security is full of sponsored 'awareness campaigns' and 'impartial studies'. These end up as little more than marketing campaigns. Even when not sponsored, a lot of arguments are tainted by the fierce ideological battle between the open [6] and closed [17] software camps.

It's nearly impossible for a novice in the field to get unbiased information. All one can do is immerse oneself in the arguments and hope to get enlightenment rather than conversion, to gather enough information before absorbing too much theology.

The old saying about statistics *figures can't lie, but lies can figure* applies in full measure. Even seemingly solid statistics are perverted to champion one side of the argument over the other. Which system has more 'critical bugs' depends entirely on how you define a 'critical' bug and where you draw the boundaries of a system. Not that the number of these bugs is even important — it's the impact that matters.

5.3.2 Profitability through gullibility

Corporations don't like smart, educated consumers. It's harder to market [21] to them. Why should a company spend money educating end-users only to dilute the effects of it's marketing? Companies exist, after all, to make a profit. Your breakfast cereal box is plastered with flashy logos and icons with catchy slogans printed in full colour. Squirreled away in a corner somewhere is some mandated information telling you, in an unbiased way, what's actually in the box. If your cereal company wanted to educate you, wouldn't this information be more prominent? Wouldn't the '35% sugar' be written just as boldly as the '95% fat free'? Can you think of *any* industry with that kind of honesty and dedication to customer education?

I thought not.

5.3.3 Context of knowledge

In the 'real world', complex rules eventually become 'common sense'. Don't stick a knife in the toaster, don't walk down dark alleys, don't trust strangers, and, if it sounds too good to be true, it probably is. The problem with computer security is that these rules are based on abstract as well as real concepts, and that they change too quickly.

Security in the real world is mostly concerned with physical security (don't walk down dark alleys). In the PC world, physical security is usually already taken care of (no-one leaves an expensive PC lying about to be stolen). What's left is 'virtual security', and it can be difficult to convince someone that, even though they're safe and comfortable in their lounge-room, visiting certain kinds of sites can be dangerous.

Threats in the physical world are based on concepts that most people are at least vaguely familiar with. Don't stick a knife in the toaster — knives are metal, metal conducts electricity, toasters use electricity to heat, and electricity kills. This chain of logic is based on concepts that are absorbed either unconsciously, or at such an early age that we are not aware of any effort.

Virtual threats are based on a whole new set of concepts that most people are unfamiliar with:

- Don't accept self-signed certificates — even though the reassuring padlock appears, self-signed certificates can be created by malicious users, fooling the browser into placing too much trust

[21] pronounced "L-I-E"

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

in the site, allowing it to download malicious content which exploits a vulnerability to take over your system.

- Don't CC joke emails to dozens of people — mail clients harvest such addresses and add them to their address book [22], putting those recipients in the line of fire next time an email-virus runs rampant. Worse yet, someone may post the joke, including email addresses, on some web page or newsgroup to be harvested by spammers.

These chains of events and concepts are alien enough to casual users to prevent 'common sense' from highlighting the danger. Most novice computer users learn rote behaviour (do this, don't do that) without understanding WHY such behaviour is desirable or dangerous.

Computer security changes too quickly to rely on a given set of meta-rules. It used to be 'common sense' that it was safe to open an email, as long as you don't click on any attachments. Now it's 'common sense' that (depending on your email client) even previewing an email can be dangerous. Without understanding why these rules keep changing it's very difficult to apply them correctly — they get forgotten, mis-remembered or ignored.

5.3.4 Apathy

Most users see a computer as a beige box that mysteriously does things. People in the computer industry constantly underestimate just how little average user cares about security as long as it does not have a direct impact on them.

PC's are so alien that it's difficult to convey in meaningful, real-live concepts, the consequences of their actions. Cars need to be serviced at regular intervals. This is done despite the cost because it's obvious that if you don't things wear out, brake, and cost even more to fix. There is a simple relationship: if *I* don't service *my* car regularly it ends up costing *me* more *money*.

There is no such simple correlation between failing to secure one's PC and personal hardships. An infected PC may crash more often (So what? Users have been trained to expect computer to crash [23]), run slower (PC's are so massively overpowered that the slowdown has to be extreme before it becomes inconvenient) or be used to attack others (key word being 'others'). Why spend money on virus checkers and firewalls, and time learning to use them, and effort applying patches when there is no personal gain?

Invariably something bad does happen, but the natural reaction is to blame hackers, shoddy software, hackers, flimsy PC's or hackers. If all else fails, blame hackers. No-one would drive a car with squealing brakes for weeks, and then blame the tire company when the brakes fail. PC users will put up with oddly behaving PC's for weeks, then blame others when the system fails.

[22] User-friendliness trumps security again!

[23] A bug in Windows 95 and 98 caused a crash after 49.7 days[15]. It took 4 years to be discovered, so rare are up-times of that order!

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

6 Conclusion

How did we get into this state of computer insecurity? Computers are misunderstood. The average computer has more options, settings, buttons and widgets, in short, more moving parts than the Titanic (and we all know what happened to that!). Companies selling PC's and software for them have understood that the only way that such a hideously complex device will ever be welcome is if it looks simple to use.

While software companies have been busy on the one hand adding features, interactions and piling on complexity, on the other hand they've been telling us how simple, safe and user-friendly these machines are. Don't be scared, it's just an appliance like a toaster or a fridge. Buy one, turn it on and your life will be better!

Computer security is a victim of the rush to achieve market share through user-friendliness. Public demand has shifted somewhat to security, and lots of noises are being made about focusing on security now. The problem is that security is not being sought for it's own sake, but rather as a means to maintain market share. No one knows what the market will ask for tomorrow, but vendors will rush to supply it, leaving security a poor second, again.

The only way out of this morass is for users to be educated so they can make better decisions and demand better products. The emphasis is on the *users* making the decisions, because they are the ones affected by them! Don't let someone else tell you what's best for you, don't let others make decisions for you.

There is no technical solution to PC security. There is no magic bullet — no firewall, virus checker, patch or proxy server that can provide it. Security cannot be retrofitted to an insecure system, so it has to be designed into the system. Security must be multi-layered because any one layer can be breached. The only way that will happen is if customers demand it, because writing secure software is expensive.

I'd like to finish with a relevant quote from one of my favorite [24] articles:

> What would the engineer say, after you had explained your problem, and enumerated all of the dissatisfactions in your life? He would probably tell you that life is a very hard and complicated thing; that no interface can change that; that anyone who believes otherwise is a sucker; and that if you don't like having choices made for you, you should start making your own. [35]

[24] The part about the Hole Hawg *still* cracks me up!

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

A Bibliography

[1] DNS Spoofing. *Men&Mice*. http://www.menandmice.com/9000/9211_dns_spoofing.html.

[2] Man-in-the-middle attack. *ITsecurity.com*. http://www.itsecurity.com/dictionary/middle.htm.

[3] What is source routed traffic and why is it a threat? *Vesaria Network Security Specialists*. http://www.vesaria.com/Firewall/FAQ/sec30.php.

[4] The verdict. *CNN*, Oct 1993. http://www.cnn.com/US/OJ/verdict/index.html.

[5] Advisory CA199904 Melissa Macro Virus. *CERT*, Mar 1999. http://www.cert.org/advisories/CA-1999-04.html.

[6] Advocacy, Frequently Asked Questions. *Open Source Initiative*, May 2001. http://www.opensource.ac.uk/mirrors/www.opensource.org/advocacy/faq.html.

[7] The twenty most critical internet security vulnerabilities (updated) t̃he experts consensus. *SANS*, Oct 2003. http://www.sans.org/top20/#u4.

[8] Auscert. *Updated gdkpixbuf packages fix denial of service vulnerability*, Mar 2004. http://www.auscert.org.au/render.html?it=3941\&cid=1.

[9] Is verification necessary. Apr 2004. http://c2.com/cgi/wiki?IsVerificationNecessary.

[10] Java technology. *Sun Microsystems, Inc.*, 2004. http://java.sun.com.

[11] phishing. *Jupitermedia Corporation*, 2004. http://isp.webopedia.com/TERM/P/phishing.html.

[12] Jay Beale. "security through obscurity" ain't what they think it is. *Bastille Linux*, 2000.

[13] Ted Bridis. Federal bounty may nab e-mail spammers. *Associated Press*, Sep 2004. http://www.cnn.com/2004/TECH/internet/09/17/ftc.spambounty.ap/index.html.

[14] Microsoft Corporation. Activex controls. http://www.microsoft.com/com/tech/ActiveX.asp.

[15] Microsoft Corporation. 216641: Computer Hangs After 49.7 Days. *Microsoft Knowledge Base*, Jun 1999. http://support.microsoft.com:80/support/kb/articles/q216/6/41.asp.

[16] Mark Joseph Edwards. Proof of concept virus first to infect macromedia flash files. Jan 2002. http://www.winnetmag.com/Article/ArticleID/23724/23724.html.

[17] Rob Enderle. Innovation loses if open source wins. *E-Commerce Times*, Nov 2003. http://www.ecommercetimes.com/story/32154.html.

[18] Peter Ferrie et al. SymbOS.Cabir. Jul 2004. http://securityresponse.symantec.com/avcenter/venc/data/epoc.cabir.html.

[19] Rik Farrow. Spoofing source addresses. http://www.spirit.com/Network/net0300.html.

[20] Rik Farrow. Reverse-engineering new exploits. *CMP Media LLC*, Mar 2004. http://www.networkmagazine.com/shared/article/showArticle.jhtml?articleId=18201800.

[21] Debbie Gage and John McCormick. Can software kill? *Baseline*, Mar 2004. http://www.eweek.com/article2/0,1759,1544225,00.asp.

[22] Thomas C Greene. Ms security patch eula gives billg admin privileges on your box. *The Register*, Jun 2002. http://www.theregister.co.uk/2002/06/30/ms_security_patch_eula_gives/.

[23] Malachi Kenney. Ping of death. *insecure.org*, Oct 1996. http://www.insecure.org/sploits/ping-o-death.html.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

[24] Robert Lemos. Microsoft to offer bounty on hackers. *CNET News.com*, Nov 2003. http://news.com.com/2100-7355_3-5102110.html.

[25] Steven Levy. A net of control. *Newsweek International*, 2004.

[26] Stephen Long. Packer tax win. *The World Today*, Mar 2004. http://www.abc.net.au/worldtoday/content/2004/s1078192.htm.

[27] Erich Luening. Microsoft details office rental plans. *CNET News.com*, Nov 1999. http://news.com.com/2100-1001-232659.html.

[28] John MCCormick. The threat of browser helper objects. *TechRepublic*, Jul 2004. http://www.zdnet.com.au/insight/security/0,39023764,39153405,00.htm.

[29] Microsoft. Erroneous verisign-issued digital certificates pose spoofing hazard. *Microsoft Security Bulletin MS01-017*, Jun 2003.

[30] Andrew Orlowski. Microsoft eula asks for root rights — again. *The Register*, Aug 2002. http://www.theregister.co.uk/2002/08/02/microsoft_eula_asks_for_root/.

[31] Jonathan B. Postel. Simple mail transfer protocol. *Internet RFC/STD/FYI/BCP Archives*, Aug 1982. http://www.faqs.org/rfcs/rfc821.html.

[32] Kevin Poulsen. Alleged webtv 911 hacker charged with cyberterrorism. *SecurityFocus*, Feb 2004. http://www.securityfocus.com/news/8136.

[33] Jim Powell. Great thinkers on liberty. *About the Acton Institute*, 2001. http://www.acton.org/about/.

[34] SANS. Survival time history. 2004.

[35] Neal Stephenson. In the beginning was the command line. 1999. http://www.spack.org/wiki/InTheBeginningWasTheCommandLine.

[36] Con Zymaris. A comparison of the gpl and the microsoft eula. *Cybersource*, Apr 2003. www.cybersource.com.au/cyber/about/comparing_the_gpl_to_eula.pdf.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Last Updated: September 3rd, 2015

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Crystal City 2015	Crystal City, VAUS	Sep 08, 2015 - Sep 13, 2015	Live Event
SANS Network Security 2015	Las Vegas, NVUS	Sep 12, 2015 - Sep 21, 2015	Live Event
SANS Seoul 2015	Seoul, KR	Sep 14, 2015 - Sep 19, 2015	Live Event
Data Breach Investigation Summit & Training	Dallas, TXUS	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Baltimore 2015	Baltimore, MDUS	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Perth 2015	Perth, AU	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Tallinn 2015	Tallinn, EE	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS ICS Amsterdam 2015	Amsterdam, NL	Sep 22, 2015 - Sep 28, 2015	Live Event
SANS Bangalore 2015	Bangalore, IN	Sep 28, 2015 - Oct 17, 2015	Live Event
SANS DFIR Prague 2015	Prague, CZ	Oct 05, 2015 - Oct 17, 2015	Live Event
SANS Seattle 2015	Seattle, WAUS	Oct 05, 2015 - Oct 10, 2015	Live Event
SANS Tysons Corner 2015	Tysons Corner, VAUS	Oct 12, 2015 - Oct 17, 2015	Live Event
SOS: SANS October Singapore 2015	Singapore, SG	Oct 12, 2015 - Oct 24, 2015	Live Event
SANS Gulf Region 2015	Dubai, AE	Oct 17, 2015 - Oct 29, 2015	Live Event
SANS Cyber Defense San Diego 2015	San Diego, CAUS	Oct 19, 2015 - Oct 24, 2015	Live Event
SANS Tokyo Autumn 2015	Tokyo, JP	Oct 19, 2015 - Oct 31, 2015	Live Event
SANS South Florida 2015	Fort Lauderdale, FLUS	Nov 09, 2015 - Nov 14, 2015	Live Event
SANS Sydney 2015	Sydney, AU	Nov 09, 2015 - Nov 21, 2015	Live Event
SANS London 2015	London, GB	Nov 14, 2015 - Nov 23, 2015	Live Event
Pen Test Hackfest Summit & Training	Alexandria, VAUS	Nov 16, 2015 - Nov 23, 2015	Live Event
SANS Hyderabad 2015	Hyderabad, IN	Nov 24, 2015 - Dec 04, 2015	Live Event
SANS Cape Town 2015	Cape Town, ZA	Nov 30, 2015 - Dec 05, 2015	Live Event
SANS San Francisco 2015	San Francisco, CAUS	Nov 30, 2015 - Dec 05, 2015	Live Event
SANS Milan 2015	OnlineIT	Sep 07, 2015 - Sep 12, 2015	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced