November 15, 2001

Crypto-Gram Newsletter

by Bruce Schneier
Founder and CTO
Counterpane Internet Security, Inc.
schneier@schneier.com
<http://www.counterpane.com>

A free monthly newsletter providing summaries, analyses, insights, and commentaries on computer security and cryptography.

Back issues are available at <http://www.schneier.com/crypto-gram.html>. To subscribe, visit <http://www.schneier.com/crypto-gram.html> or send a blank message to crypto-gram-subscribe@chaparraltree.com.

In this issue:

Full Disclosure

Microsoft is leading the charge to restrict the free flow of computer security vulnerabilities. Last month Scott Culp, manager of the security response center at Microsoft, published an essay describing the current practice of publishing security vulnerabilities to be "information anarchy." He claimed that we'd all be a lot safer if researchers would keep details about vulnerabilities to themselves, and stop arming hackers with offensive tools. Last week, at Microsoft's Trusted Computing Forum, Culp announced a new coalition to put these ideas into practice.

This is the classic "bug secrecy vs. full disclosure" debate. I've written about it previously in Crypto-Gram; others have written about it as well. It's a complicated issue with subtle implications all over computer security, and it's one worth discussing again.

The Window of Exposure

I coined a term called the "Window of Exposure" to explain the evolution of a security vulnerability over time. A vulnerability is a bug; it's a programming mistake made by a programmer during the product's development and not caught during testing. It's an opening that someone can abuse to break into the computer or do something normally prohibited.

Assume there's a vulnerability in a product and no one knows about it. There is little danger, because no one knows to exploit the vulnerability. This vulnerability can lie undiscovered for a short time -- Windows XP vulnerabilities were discovered before the product was released -- or for years. Eventually, someone discovers the vulnerability. Maybe it's a good guy who tells the developer. Maybe it's a bad guy who exploits the vulnerability to break into systems. Maybe it's a guy who tells no one, and then someone else discovers it a few months later. In any case, once someone knows about the vulnerability, the danger increases.

Eventually, news of the vulnerability spreads. Maybe it spreads amongst

the security community. Maybe it spreads amongst the hacker underground. The danger increases as more people learn about the vulnerability. At some point, the vulnerability is announced. Maybe it's announced on Bugtraq or another vulnerability Web site. Maybe it's announced by the security researcher in a press release, or by CERT, or by the software developer. Maybe it's announced on a hacker bulletin board. But once it's announced, the danger increases even more because more people know about it.

Then, someone writes an exploit: an automatic tool that exercises the vulnerability. This is an inflection point, and one that doesn't have a real-world analog for two reasons. One, software has the ability to separate skill from ability. Once a tool is written, anyone can exploit the vulnerability, regardless of his skill or understanding. And two, this tool can be distributed widely for zero cost, thereby giving everybody who wants it the ability. This is where "script kiddies" come into play: people who use automatic attack tools to break into systems. Once a tool is written, the danger increases by orders of magnitude.

Then, the software developer issues a patch. The danger decreases, but not as much as we'd like to think. A great many computers on the Internet don't have their patches up to date; there are many examples of systems being broken into using vulnerabilities that should have been patched. I don't fault the sysadmins for this; there are just too many patches, and many of them are sloppily written and poorly tested. So while the danger decreases, it never gets back down to zero.

You can think of this as a graph of danger versus time, and the Window of Exposure as the area under the graph. The goal is to make this area as small as possible. In other words, we want there to be as little danger as possible over the life cycle of the software and the particular vulnerability. Proponents of bug secrecy and proponents of full disclosure simply have different ideas for achieving that.

History of Full Disclosure

During the early years of computers and networks, bug secrecy was the norm. When users and researchers found vulnerabilities in a software product, they would quietly alert the vendor. In theory, the vendor would then fix the vulnerability. After CERT was founded in 1988, it became a clearinghouse for vulnerabilities. People would send newly discovered vulnerabilities to CERT. CERT would then verify them, alert the vendors, and publish the details (and the fix) once the fix was available.

The problem with this system is that the vendors didn't have any motivation to fix vulnerabilities. CERT wouldn't publish until there was a fix, so there was no urgency. It was easier to keep the vulnerabilities secret. There were incidents of vendors threatening researchers if they made their findings public, and smear campaigns against researchers who announced the existence of vulnerabilities (even if they omitted details). And so many vulnerabilities remained unfixed for years.

The full disclosure movement was born out of frustration with this process. Once a vulnerability is published, public pressures give vendors a strong incentive to fix the problem quickly. For the most part, this has worked. Today, many researchers publish vulnerabilities they discover on mailing lists such as Bugtraq. The press writes about the vulnerabilities in the computer magazines. The vendors scramble to patch these vulnerabilities as soon as they are publicized, so they can write their own press releases about how quickly and thoroughly they fixed things. The full disclosure movement is improving Internet security.

At the same time, hackers use these mailing lists to learn about vulnerabilities and write exploits. Sometimes the researchers themselves write demonstration exploits. Sometimes others do. These exploits are used to break into vulnerable computers and networks, and greatly decrease Internet security. In his essay, Culp points to Code Red, Li0n, Sadmind, Ramen, and Nimda as examples of malicious code written after

researchers demonstrated how particular vulnerabilities worked.

Those against the full-disclosure movement argue that publishing vulnerability details does more harm than good by arming the criminal hackers with tools they can use to break into systems. Security is much better served, they counter, by keeping the exact details of vulnerabilities secret.

Full-disclosure proponents counter that this assumes that the researcher who publicizes the vulnerability is always the first one to discover it, which simply isn't true. Sometimes vulnerabilities have been known by attackers (sometimes passed about quietly in the hacker underground) for months or years before the vendor ever found out. The sooner a vulnerability is publicized and fixed, the better it is for everyone, they say. And returning to bug secrecy would only bring back vendor denial and inaction.

That's the debate in a nutshell: Is the benefit of publicizing an attack worth the increased threat of the enemy learning about it? Should we reduce the Window of Exposure by trying to limit knowledge of the vulnerability, or by publishing the vulnerability to force vendors to fix it as quickly as possible?

What we've learned during the past eight or so years is that full disclosure helps much more than it hurts. Since full disclosure has become the norm, the computer industry has transformed itself from a group of companies that ignores security and belittles vulnerabilities into one that fixes vulnerabilities as quickly as possible. A few companies are even going further, and taking security seriously enough to attempt to build quality software from the beginning: to fix vulnerabilities before the product is released. And far fewer problems are showing up first in the hacker underground, attacking people with absolutely no warning. It used to be that vulnerability information was only available to a select few: security researchers and hackers who were connected enough in their respective communities. Now it is available to everyone.

This democratization is important. If a known vulnerability exists and you don't know about it, then you're making security decisions with substandard data. Word will eventually get out -- the Window of Exposure will grow -- but you have no control, or knowledge, of when or how. All you can do is hope that the bad guys don't find out before the good guys fix the problem. Full disclosure means that everyone gets the information at the same time, and everyone can act on it.

And detailed information is required. If a researcher just publishes vague statements about the vulnerability, then the vendor can claim that it's not real. If the researcher publishes scientific details without example code, then the vendor can claim that it's just theoretical. The only way to make vendors sit up and take notice is to publish details: both in human- and computer-readable form. (Microsoft is guilty of both of these practices, using their PR machine to deny and belittle vulnerabilities until they are demonstrated with actual code.) And demonstration code is the only way to verify that a vendor's vulnerability patch actually patched the vulnerability.

This free information flow, of both description and proof-of-concept code, is also vital for security research. Research and development in computer security has blossomed in the past decade, and much of that can be attributed to the full-disclosure movement. The ability to publish research findings -- both good and bad -- leads to better security for everyone. Without publication, the security community can't learn from each other's mistakes. Everyone must operate with blinders on, making the same mistakes over and over. Full disclosure is essential if we are to continue to improve the security of our computers and networks.

Bug Secrecy Example

You can see the problems with bug secrecy in the digital-rights-management industry. The DMCA has enshrined the bug secrecy paradigm into law; in most cases it is illegal to publish vulnerabilities or

automatic hacking tools against copy-protection schemes. Researchers are harassed, and pressured against distributing their work. Security vulnerabilities are kept secret. And the result is a plethora of insecure systems, their owners blustering behind the law hoping that no one finds out how bad they really are.

The result is that users can't make intelligent decisions on security. Here's one example: A few months ago, security researcher Niels Ferguson found a security flaw in Intel's HDCP Digital Video Encryption System, but withheld publication out of fear of being prosecuted under the DMCA. Intel's reaction was reminiscent of the pre-full-disclosure days: they dismissed the break as "theoretical" and maintained that the system was still secure. Imagine you're thinking about buying Intel's system. What do you do? You have no real information, so you have to trust either Ferguson or Intel.

Here's another: A few weeks ago, a release of the Linux kernel came without the customary detailed information about the OS's security. The developers cited fear of the DMCA as a reason why those details were withheld. Imagine you're evaluating operating systems: Do you feel more or less confident about the security the Linux kernel version 2.2, now that you have no details?

Full Disclosure and Responsibility

Culp has a point when he talks about responsibility. (Of course, Scott is avoiding "mea Culpa.") The goal here is to improve security, not to arm people who break into computers and networks. Automatic hacking tools with easy point-and-click interfaces, ready made for script kiddies, cause a lot of damage to organizations and their networks. There are such things as responsible and irresponsible disclosure. It's not always easy to tell the difference, but I have some guidelines.

First, I am opposed to attacks that primarily sow fear. Publishing

vulnerabilities that there's no real evidence for is bad. Publishing vulnerabilities that are more smoke than fire is bad. Publishing vulnerabilities in critical systems that cannot be easily fixed and whose exploitation will cause serious harm (e.g., the air traffic control system) is bad.

Second, I believe in giving the vendor advance notice. CERT took this to an extreme, sometimes giving the vendor years to fix the problem. I'd like to see the researcher tell the vendor that he will publish the vulnerability in a few weeks, and then stick to that promise. Currently CERT gives vendors 45 days, but will disclose vulnerability information immediately for paid subscribers. Microsoft proposes a 30-day secrecy period. While this is a good idea in theory, creating a special insider group of people "in the know" has its own set of problems.

Third, I agree with Culp that it is irresponsible, and possibly criminal, to distribute easy-to-use exploits. Reverse engineering security systems, discovering vulnerabilities, writing research papers about them, and even writing demonstration code, benefits research; it makes us smarter at designing secure systems. Distributing exploits just make us more vulnerable. I'd like to get my hands on the people who write virus creation kits, for example. They've got a lot to answer for.

This is not clear-cut: there are tools that do both good and bad, and sometimes the difference is merely marketing. Dan Farmer was vilified for writing SATAN; today, vulnerability assessment tools are viable security administration products. Remote administration tools look a lot like Back Orifice (although less feature-rich). L0phtCrack is a hacker tool to break weak passwords as a prelude to an attack, but LC 3.0 is sold as a network administration tool to test for weak passwords. And the program that Dmitry Sklyarov was arrested for writing has legitimate uses. In fact, most tools have both good and bad uses, and when in doubt I believe it is better to get the information in the hands of people who need it, even if it means

that the bad guys get it too.

One thing to pay attention to is the agenda of the researcher. Publishing a security vulnerability is often a publicity play; the researcher is looking to get his own name in the newspaper by successfully bagging his prey. The publicizer often has his own agenda: he's a security consultant, or an employee of a company that offers security products or services. I am a little tired of companies that publish vulnerabilities in order to push their own product or service. Although, of course, a non-altruistic motive does not mean that the information is bad.

I like the "be part of the solution, not part of the problem" metric. Researching security is part of the solution. Convincing vendors to fix problems is part of the solution. Sowing fear is part of the problem. Handing attack tools to clueless teenagers is part of the problem.

The Inevitability of Security Vulnerabilities

None of this would be an issue if software were engineered properly in the first place. A security vulnerability is a programming mistake: either an out-and-out mistake like a buffer overflow, which should have been caught and prevented, or an opening introduced by a lack of understanding the interactions in a complex piece of code. If there were no security vulnerabilities, there would be no problem. It's poor software quality that causes this mess in the first place.

While this is true -- software vendors uniformly produce shoddy software -- the sheer complexity of modern software and networks means that vulnerabilities, lots of vulnerabilities, are inevitable. They're in every major software package. Each time Microsoft releases an operating system it crows about how extensive the testing was and how secure it is, and every time it contains more security vulnerabilities than the previous operating system. I don't believe this trend will reverse itself anytime soon.

Vendors don't take security seriously because there is no market incentive for them to, and no adverse effects when they don't. I have long argued that software vendors should not be exempt from the product liability laws that govern the rest of commerce. When this happens, vendors will do more than pay lip service to security vulnerabilities: they will fix them as quickly as possible. But until then, full disclosure is the only way we have to motivate vendors to act responsibly.

Microsoft's motives in promoting bug secrecy are obvious: it's a whole lot easier to squelch security information than it is to fix problems, or design products securely in the first place. Microsoft's steady stream of public security vulnerabilities has led many people to question the security of their future products. And with analysts like Gartner advising people to abandon Microsoft IIS because of all its insecurities, giving customers less security information about their products would be good for business.

Bug secrecy is a viable solution only if software vendors are followers of W. Edwards Deming's quality management principles. The longer a bug remains unfixed, the bigger a problem it is. And because the number of systems on the Internet is constantly growing, the longer a security vulnerability remains unfixed, the larger the window of exposure. If companies believe this and then act accordingly, then there is a powerful argument for secrecy.

However, history shows this isn't the case. Read Scott Culp's essay; he did not say: "Hey guys, if you have a bug, send it to me and I'll make sure it gets fixed pronto." What he did was to rail against the publication of vulnerabilities, and ask researchers to keep details under their hats. Otherwise, he threatened, "vendors will have no choice but to find other ways to protect their customers," whatever that means. That's the attitude that makes full disclosure the only viable way to reduce the window of vulnerability.

In his essay, Culp compares the practice of publishing vulnerabilities to

shouting "Fire" in a crowded movie theater. What he forgets is that there actually is a fire; the vulnerabilities exist regardless. Blaming the person who disclosed the vulnerability is like imprisoning the person who first saw the flames. Disclosure does not create security vulnerabilities; programmers create them, and they remain until other programmers find and remove them. Everyone makes mistakes; they are natural events in the sense that they inevitably happen. But that's no excuse for pretending that they are caused by forces out of our control, and mitigated when we get around to it.

Scott Culp's essay:
<http://www.microsoft.com/technet/columns/security/...>

Q&A with Culp:
<http://news.cnet.com/news/0-1014-201-7819204-0.html>

News articles on Culp:
<http://www.theregister.co.uk/content/55/22332.html>
<http://news.cnet.com/news/0-1003-200-7560391.html?...>
<http://cgi.zdnet.com/slink?153618:8469234>

Microsoft's push for secrecy:
<http://www.securityfocus.com/news/281>
<http://213.40.196.62/media/670.ppt>
<http://www.theregister.co.uk/content/4/22614.html>
<http://www.theregister.co.uk/content/4/22740.html>

My original essay on the Window of Exposure:
<http://www.schneier.com/crypto-gram-0009.html#1>

My earlier essays on full disclosure:
<http://www.schneier.com/...>
<http://www.schneier.com/...>
Note that the nCipher anecdote is untrue. Details are here:

<http://www.schneier.com/crypto-gram-0104.html#2>

Other commentary:
<http://www.securityfocus.com/news/270>
<http://web.ranum.com/usenix/ranum_5_temp.pdf>
<http://www.osopinion.com/perl/story/13871.html>
<http://www.synthesis.net/tech/fulldisclosure/>
<http://www.osopinion.com/perl/story/14401.html>
<http://www.net-security.org/text/articles/...>

Thanks to Tina Bird, Jon Callas, Scott Culp, Greg Guerin, Elias Levy, Jeff Moss, Eric Raymond, and Elizabeth Zwicky for reading and commenting on this essay.

Crypto-Gram Reprints

Why Digital Signatures are Not Signatures
<http://www.schneier.com/crypto-gram-0011.html#1>

Programming Satan's Computer: Why Computers are Insecure
<http://www.schneier.com/...>

Elliptic-Curve Public-Key Cryptography
<http://www.schneier.com/...>

The Future of Fraud: Three reasons why electronic commerce is different
<http://www.schneier.com/...>

Software Copy Protection: Why copy protection does not work:
<http://www.schneier.com/crypto-gram-9811.html#copy>

News

After all the posturing, Sen. Gregg is not going to introduce a bill mandating government access to cryptography:
<http://www.wired.com/news/conflict/0,2100,47635,00.html>

FBI is expanding its Internet wiretapping efforts:
<http://www.interactiveweek.com/article/...>

Just in case anyone is doubting my near hysteria about the entertainment industry and their willingness to destroy the computer industry to achieve their goals.... Last month the RIAA tried to sneak an amendment into the anti-terrorism legislation giving them the right to hack into people's computers, looking for unauthorized copyrighted material. I am disgusted at this attempt to equate people who make unauthorized copies of music with people who fly passenger jets into skyscrapers.
<http://www.wired.com/news/conflict/...>
<http://www.zdnet.com/zdnn/stories/comment/...>
And they're also considering using denial-of-service attacks to shut down file sharing servers:
<http://www.theregister.co.uk/content/55/22327.html>
And the SSSCA saga continues:
<http://www.newsforge.com/article.pl?sid=01/10/19/...>
Not much is known about the legislation, because Sen. Hollings refuses to release much information about it and is blocking hearings:
<http://www.newsforge.com/article.pl?sid=01/09/20/...>
These guys have to be stopped before they destroy computer security for everyone.
The SSSCA draft:
<http://cryptome.org/sssca.htm>

Seems like "terrorist" is the current thing to call someone you don't like. Michael Lane Thomas, Microsoft's senior .Net developer evangelist, called virus writers "industrial terrorists." As I wrote in the last issue, this is wrongheaded and damaging. Terrorists cause terror, and should not be

confused with normal criminals. Malware writers are annoying -- they cause damage, cost money, and destroy data -- but they're not terrorists.
<http://www.theregister.co.uk/content/56/22423.html>
<http://www.zdnet.co.uk/itweek/columns/2001/40/...>
Particularly disturbing is the way Thomas tried to invoke patriotism and counterterrorism in an effort to get people to use Microsoft products. He said that if people stop using Microsoft's IIS because of security concerns -- as Gartner advocated recently -- that this "would only accomplish what the industrial terrorists want."

CERT predicts computer attacks this year will be double what they were last year.
<http://www.securityfocus.com/news/266>
<http://www.zdnet.com/zdnn/stories/news/...>

There's a bunch of new stuff at NIST. A report of the Second Modes of Operation Workshop:
<http://csrc.nist.gov/encryption/modes/workshop2/...>
Changes in the Digital Signature Standard, FIPS 186-2. Among other things, the change notice specifies recommended key sizes and modifications to the RNG:
<http://csrc.nist.gov/encryption/tkdigsigs.html>
The Key Management Schemes document for the Key Management Workshop, scheduled for 1-2 November:
<http://csrc.nist.gov/encryption/kms/workshop2-page.html>

A new book claims that a woman cryptanalyst broke part of the German Enigma encoding machine before World War II, but her supervisors ignored her theories.
<http://www.wired.com/news/women/0,1540,47560,00.html>

A new version of Linux is being released without security information, out of fear of the DMCA. Honestly, I don't see how the DMCA applies here, but this is a good indication of the level of fear in the community.

<http://www.securityfocus.com/news/274>
<http://www.securityfocus.com/columnists/35>

Worry about insider attacks:
<http://www.computerworld.com/storyba/...>

Good article on the need for software security standards:
<http://www.computerworld.com/storyba/...>

Hacking wireless networks:
<http://news.bbc.co.uk/hi/english/sci/tech/...>

Microsoft's DRM2, their new digital-rights management security software, has been broken. A hacking tool developed by someone with the pseudonym Beale Screamer can strip the copy protection off audio files.
<http://www.theregister.co.uk/content/55/22354.html>
<http://news.cnet.com/news/0-1005-200-7590303.html>
<http://cgi.zdnet.com/slink?154661:8469234>
<http://cryptome.org/ms-drm.htm>
Once again, we learn that any digital copy protection system can be broken. This break isn't even interesting.

And Hong Kong hackers are making a business out of breaking Microsoft copy protection:
<http://www.zdnet.com/zdnn/stories/news/...>

Nice four-part article on security policies.
<http://www.securityfocus.com/cgi-bin/infocus.pl?id=1193>
<http://www.securityfocus.com/cgi-bin/infocus.pl?id=1473>
<http://www.securityfocus.com/cgi-bin/infocus.pl?id=1487>
<http://www.securityfocus.com/cgi-bin/infocus.pl?id=1497>

Identity theft is on the rise:
<http://cgi.zdnet.com/slink?154713:8469234>

Good article on information warfare:
<http://www.techreview.com/magazine/nov01/...>

IDSs and their complexity:
<http://cgi.zdnet.com/slink?154665:8469234>

More news on the Enigma machine stolen from Bletchley Park last year. The rotors have been recovered:
<http://news.bbc.co.uk/hi/english/uk/newsid_1609000/...>

Two-part article on honeypots, by the Honeynet Project:
<http://www.securityfocus.com/cgi-bin/infocus.pl?id=1492>
<http://www.securityfocus.com/cgi-bin/infocus.pl?id=1498>

Essay by Whitfield Diffie and Susan Landau on the security/privacy implications of Microsoft's .NET initiative:
<http://www.kingpublishing.com/fc/new_technology/...>

Security flaw in software that automatically updates anti-virus software:
<http://www.zdnet.com/zdnn/stories/news/...>

Some months ago I warned about the security problems that Unicode will bring. Here's an example:
<http://www.securityfocus.com/bid/3461>

Network Associates is trying to sell PGP:
<http://www.wired.com/news/privacy/0,1848,47551,00.html>
<http://www.securityfocus.com/news/264>

Good example of real-world risks from networked systems. An Australian man was found guilty of hacking into a Queensland computerized waste-management system and caused millions of liters of raw sewage to spill out into local parks and rivers.
<http://www.theregister.co.uk/content/4/22579.html>

More problems with Microsoft Passport:
<http://www.wired.com/news/technology/...>
<http://news.cnet.com/news/0-1003-200-7764433.html>
<http://www.washingtonpost.com/wp-dyn/articles/...>
It's an enormous risk to put all this information in a single repository. And the Microsoft PR spin completely missed the point. They said: "Ultimately, the big takeaway from this is that there is no evidence that anyone has ever taken advantage of this." One, would there be any evidence? Two, the real problem is future risk, not current injury. And three, on what basis should I continue trusting Microsoft's vacuous security promises?

Passport vulnerabilities Web site:
<http://alive.znep.com/~marcs/passport/>

DeCSS has been ruled "speech" by a California State Appeals Court, overturning the lower court ruling. Good news!
<http://www.wired.com/news/print/0,1294,48075,00.html>
<http://www.courtinfo.ca.gov/courts/courtsofappeal/...>
<http://slashdot.org/yro/01/11/01/1953236.shtml>
<http://www.theregister.co.uk/content/55/22613.html>

A GSM phone with end-to-end encryption:
<http://www.pcworld.com/news/article/0,aid,51368,00.asp>

Essay on the problems of "security by obscurity." Makes the point that Microsoft is likely to break its record for security patches this year: over 100. (That's two a week.)
<http://www.vnunet.com/Analysis/1126488>

Within hours of Windows XP being released, pirates broke the copy protection schemes and started distributing stolen copies.
<http://www.newsbytes.com/news/01/171651.html>
Microsoft's spin here is correct. There are certainly security risks involved in using pirated software.

More security data: A survey says that one in nine IIS servers can be taken over by hackers. Is it any wonder that Gartner is advising people to switch?
<http://www.infoworld.com/articles/hn/xml/01/11/02/...>

Another survey says that two thirds of all wireless networks in London are completely open:
<http://news.bbc.co.uk/hi/english/sci/tech/...>

Hacking tools are getting worse:
<http://www.computing.vnunet.com/News/1126643>
<http://www.securityfocus.com/news/280>

Counterpane Internet Security News

It's been a great few months at Counterpane. We're signing up monitoring customers at an unprecedented rate. We've got the best security VARs in the country--major national resellers like VeriSign and NEC, as well as excellent local resellers like Accudata, FishNet, Espiria, and Cadre Systems--selling Counterpane monitoring. We've just launched the "Counterpane Protected" program. Industry analysts are touting Counterpane's capabilities and opportunities.

Counterpane Protected:
<http://www.counterpane.com/protected.html>
<http://www.counterpane.com/pr-protected.html>

Analyst commentaries:
<http://www.counterpane.com/analyst.html>

Schneier is speaking in Dallas (11/28), Baltimore (12/3), and New York (12/5):
<http://www.techmecca.net>

<http://www.medrecinst.com/conferences/security/...>
<http://www.infosecurityevent.com>

GOVNET

The U.S. government wants its own private Internet. The idea is to create a secure GOVNET, physically separate from the public Internet. I think this is a good idea, although it will be very expensive and difficult to do and will almost certainly have insecurities. But even a mediocre implementation would be more secure than what they have now.

Limiting access to a network goes a long way towards improving its security. Hackers can't attempt to break in from half a planet away. Well-meaning friends can't pass along viruses. Trojans can't alert their owners of successful infections. Users can't access questionable Web sites and release their passwords, configurations, and private information. Outsiders can't sniff passwords. The software would be just as buggy -- applications and operating systems would have the same vulnerabilities -- but accessing those vulnerabilities would be much harder.

The effectiveness of this is directly tied to how strong the physical separation is. The networks have to be physically different. GOVNET can't run on the Internet over a VPN. GOVNET can't have firewall-protected gateways to the Internet. GOVNET can't be separated from the Internet by one of those silly "air-gap" products. GOVNET has to use its own routers, its own servers, and its own clients. If a GOVNET user wants to use the Internet, he needs two computers on his desk. He can use the same programs on both, but they have to be different copies. And he can't share files between them, not even by floppy disk.

Breaking any of these rules hurts the security. Pass a MS Word floppy between the two networks, and you risk infection by a macro virus. Attach a computer to both networks, and you risk all sorts of malware jumping

over. Add public dial-up access points, and then the public can try to break in.

GOVNET isn't a new idea. There are already several separate internets in the U.S. government -- INTELINK, SIPRNET, NIPRNET, etc. -- some of these classified networks. The classified networks are completely encrypted, and all access points are in secured rooms and buildings. They're a whole lot more secure than the Internet, but it took the Melissa virus 24 hours to jump over from the Internet to one of these networks. And the LoveLetter virus infected several of these computers.

I can imagine what happened. Some senior executive checked his e-mail on the Internet. Then he plugged the same laptop into one of the private, secure, classified, separate networks. And the viruses crossed over.

But even that is worlds better than what we have today. And a GOVNET designed from scratch can include other security features. There can be mandated strong authentication (inasmuch as commercial products allow it). All the links can be encrypted. Anonymity can be banned. There can be better accountability. There can be an approved list of permitted software. GOVNET could not prevent insider attacks, but it could make them a lot harder to get away with.

On the other hand, physically separating a network from the Internet makes it a whole lot less useful. And usefulness is why companies connected their corporate networks to the Internet in the first place. In a lot of ways, this is a huge step backwards. The Internet got its name because it was a network of networks. In the old days, there was Arpanet, Milnet, BITnet, Usenet, JANET, and a host of other disjoint networks. Connecting them to the Internet made them all more useful.

Inasmuch as GOVNET (and the others) disconnect themselves from the Internet, they become less useful. Networks like INTELINK have well-defined missions; that's why they work. GOVNET doesn't, and that's its

biggest weakness. Users will need to access pieces of the Internet, and the temptation will always be there to link to the Internet through some kind of firewall. And then the separation is gone. Unfortunately, the security of something like GOVNET is likely to be inversely proportional to its utility.

Press Release:
<http://w3.gsa.gov/web/x/publicaffairs.nsf/...>

News and Commentary:
<http://news.bbc.co.uk/low/english/sci/tech/...>
<http://www.theregister.co.uk/content/archive/22156.html>
<http://www.zdnet.com/zdnn/stories/news/...>
<http://www.zdnet.com/zdnn/stories/news/...>
<http://www.zdnet.com/sp/stories/news/...>
<http://www.zdnet.com/zdnn/stories/news/...>
<http://www3.gartner.com/DisplayDocument?doc_cd=101741>

Password Safe Vulnerability

Password Safe is a free Windows utility that securely stores passwords. I designed it when I first realized that I had too many passwords to remember. I could either choose poor passwords -- or reuse passwords for different purposes -- or create a program that encrypts passwords on a hard drive. This is the basic idea: choose one strong password (or passphrase), and encrypt all your other passwords using that password and Password Safe. Password Safe is small and simple; it is designed to do one thing well, and is not laden with features and options.

Recently, a small vulnerability was discovered in Password Safe. The issue is that in some circumstances, when you minimize Password Safe with the "clear clipboard when minimized" and "lock password database on minimize" options turned on, Windows memory management will leave a

username or password in memory. It's not the master safe combination that can be left in memory, but the most recently used stored password. This leak was found in Windows 95, and we don't yet know if it happens in other versions of Windows. (I would assume it does until proven otherwise.)

Living with the vulnerability is easy. Close the program completely between uses. Don't rely on PS 1.7.x's password-on-restore feature as 100% protection against an attacker who, say, steals your laptop. And if you don't use password-on-restore anyway, then this doesn't affect you.

Is this a real vulnerability? Yes. Is Password Safe still secure? Yes, and I haven't stopped using it because of this. Is this something that should be fixed? Yes.

Password Safe should be updated in any case. I have a list of about a dozen minor tweaks and improvements, and the program needs to be verified on Windows XP. And Password Safe needs to be made open source. But I don't have the time to deal with it.

If anyone would like to take over programming responsibilities for Password Safe, I would like to hear from them. I am looking for someone who is experienced in Windows programming, especially in the sorts of memory management issues that this vulnerability brings to light, and someone who is willing to do this work gratis (for the fame and glory) on a free piece of software. I want to retain design control over the project, but am happy to make the source code freely available. I also would like to port Password Safe to the Macintosh and to the Palm OS.

Interested parties should send an e-mail to me.

Password Safe:
<http://www.schneier.com/passsafe.html>

Vulnerability Description:
<http://cert.uni-stuttgart.de/archive/bugtraq/2001/...>

Microsoft on Windows XP

eWeek carried an interview with Microsoft's Jim Allchin, the VP of Microsoft's Platform Group. I want to quote and comment on two bits of it.

"Windows XP is dramatically more secure than Windows 2000 or any of the prior systems. Buffer overflow has been one of the attacks frequently used on the Internet. We have gone through all code and, in an automated way, found places where there could be buffer overflow, and those have been removed in Windows XP.

"We have also turned off by default a whole set of things so that users are configured in a minimalist kind of way, making them less vulnerable. We also put a Win XP machine naked on the Internet and let people try and crack it. There have been no entrances and no issues so far."

I like saving these quotes. Every time Microsoft releases an operating system, they claim it is dramatically more secure than the previous operating system. They said this with Windows NT. They said this with Windows 2000. Every time they are wrong. We'll get back to the above quotes in a year or so.

"We test all our security fixes. With all the security fixes we've published over the last 10 years, we go through a regression for every one, we put it on the Microsoft Web site before we publish it to people, we run it in production here, and we feel very confident about the quality of that.... The security fixes that we produce don't include other functionality; they are specifically designed to remove a potential intrusion."

We did't even have to wait a year for this one. On the same day this

interview ran, Microsoft had to pull a security patch because it broke people's networks when they installed it. Oops. So much for regression testing, so much for testing patches on Microsoft's own network.

And another Microsoft patch, the one to fix the nasty vulnerability described in MS01-50, was difficult to install, partly because of "Microsoft's decision to roll in a number of fixes that had nothing to do with security, such as correcting an error in how Excel sorts Czech-language lists" (according to Business Week).

I have long maintained that Microsoft treats security problems as PR problems, and this is just more evidence for that. They consistently lie to the press when talking about security, as evinced again and again by their actions.

The Allchin interview in eWeek:
<http://www.eweek.com/article/...>

Microsoft's faulty patch:
<http://www.computerworld.com/storyba/...>
<http://www.pcmag.com/article/...>
<http://www.internetnews.com/dev-news/article/...>
<http://www.theregister.co.uk/content/4/22382.html>

Microsoft stuffing other updates into a security patch:
<http://www.businessweek.com/magazine/content/01_46/...>

Comments from Readers

> **From: Alistair McDonald <alistair@ bacchusconsultancy.com>Subject: SSSCA: International Aspects**

> Moving on the SSSCA, have you given any thought to the international implications of acts such as this? We have seen with the DMCA that if

> reverse-engineering work is done in another country, that the U.S. will attempt to prosecute as soon as the individual is within U.S. borders. I do not know if they would try and extradite for such a crime, but it's possible that they might.

> With the SSSCA, it's computer hardware (plus, I guess, supporting software) that has to conform to the law. But there is no reason for other countries, (I guess Taiwan comes to mind as producer of motherboards, hard drives, controllers and other components) to produce hardware that conforms. There's no reason for other countries to introduce similar legislation either. So in other countries, copying is indeed possible without the SSSCA. The U.S. (and maybe Canada) would possibly be an island where copy protection was, indeed, protected, but the rest of the world would be free to copy. There's no feasible way for the U.S. to force all hardware worldwide to be SSSCA-compliant.

> Digressing slightly, what is a computer? A modern washing machine has one. I recall that Microsoft marketed embedded Windows at the appliance market a few years ago, I have no idea how successful they are or were in this market, but computers are becoming more common on cars and domestic appliances. Imagine when people find out that a particular model of refrigerator (or car), coupled with a serial terminal, a bit of VBscript, and maybe a breakout box or spaghetti of wiring, can be used to copy music freely!

> Back onto the SSSCA: I haven't looked in detail at it, but copies made abroad, on non-SSSCA hardware, could easily be transmitted to U.S. citizens. It's possible that the SSSCA requires copy protection for every application and file format, but even so, bootleg applications would appear, perhaps with slightly modified file formats without signatures, or whatever.

> It would be impossible to regulate this: even if all incoming broadband

were analyzed (and that itself would be impossible), files could be compressed, encrypted, renamed, and so on, to enable them to slip through. The data (movies, music, etc) would get into the U.S.A. Without limiting net access within and without the USA, this is inevitable. Would it be a crime to receive an e-mail with a non-SSSCA attachment, even if the e-mail were unsolicited? The details are fascinating! Imagine if you could get a foreign friend to e-mail a rival a dodgy file, then blow the whistle to the authorities!

What about the effects on commerce? What about American multinationals, with offices in Europe? Will they even be able to share word-processed documents or e-mail? What about using SSSCA-compliant hardware in their European offices; will it be possible, or, on the other hand, mandatory? Would there be legislation that a U.S. corporation can only share SSSCA-compliant files? The implications could be far-ranging: a corporation would no longer be buying in a free market, but in an SSSCA-compliant market, presumably with increased costs. There goes profitability.

In one way, America might be committing economic suicide. In another, it might just be a great inconvenience for everyone buying a next-generation computer. Imagine that pre-SSSCA computers become so popular that people snap them up at garage sales!

In reality, I think that the past has shown that any copy-protection scheme can be beaten. At worst, this will be an inconvenience to U.S. citizens and businesses. At best, people will see sense and this bill will never become law.

From: Anonymous
Subject: CD and DVD copy protection

> I have long argued that the entertainment industry
> doesn't want people to have computers. The entertainment

> industry wants users to sit back and consume things.
> They are trying to turn a computer into an
> Internet Entertainment Platform, along the lines of
> a television and VCR.

BULLSEYE!

I work for a company that develops CD and DVD hardware. And I am always amazed by the ideas that have been bandied about here at work that would turn CDs and DVDs into Pay Per Play systems.

My employer considers these ideas because they fear a possible lawsuit from the entertainment industry for "letting the cat out of the bag"; i.e., vicarious copyright infringement.

We don't pirate the music or movies, but the pirates use our hardware on a PC, so we could somehow be liable. That's the fear, anyway, so every once in a while, a new format is proposed here at work, DIVX-style audio discs, pay per play movie discs, etc, etc.

If you want to see the Entertainment Industry's wet-dream version of what a CD/DVD would really look like, and how it would be used, check out:

<http://www.dataplay.com>

Basically, "fair use" as we know it today is completely circumvented by hardware.

You have to pay to play the data, even if the data is on a disc, in your hand, paid for with your own money.

From: Martin Rex <martin.rex@sap-ag.de>Subject: Liability and Software

> Bruce's statement on the liability for software is absolutely correct, as well as the analogy to Firestone tires.

> Code Red uses the IIS well *WITHIN* spec, since it performs requests that are valid under the HTTP/1.0 and 1.1 specs. It is very clearly a problem in the webserver if it cannot cope with a valid request properly.

> Similar with Outlook, where the (security zone) settings claim to provide protection from executable or other unsafe content, however the mechanisms implemented by Microsoft have never reliably worked so far. Nimda is just characters -- it is Outlook that turns it into a virus. My Unix-based pure-ASCII mailer would show Nimda to me as what it actually is: characters.

> If a Firestone tire blows up under very specific steering patterns "left, right, left, left, left, right, left," the manufacturer can not get away with stating that this particular usage pattern is "improper use" for which he is not liable.

> If a manufacturer labels his lawnmower explicitly as safe for children at all ages including toddlers, then he can certainly be held liable when children play with it and get hurt.

> Microsoft is selling software which they claim is "safe" for the Internet although it evidently is not safe for any practical meaning of the word safe.

> Internet-safe means that their IIS does not only handle Microsoft-approved URLs correctly, but that it handles all valid HTTP/1.0 and HTTP/1.1 protocol correctly, including the patterns used by Code Red and Nimda.

> Same goes for Outlook: E-mail programs *MUST* be safe for all uses

> allowed by the underlying protocol specs, not just the few uses that Microsoft happens to try in their end-user tests.

> **From: Edward Welbourne <eddy@vortigen.demon.co.uk>Subject: Re: Liability and Software**

> In the last Crypto-Gram, Buck Hodges <ewhodges@yahoo.com> wrote
> > The abuse of defects is what differentiates this from traditional
> > product liability. We should instead prosecute the criminals that
> > intentionally cause mayhem and destruction.

> So, roughly, Buck argues that worms, viruses, etc. should be compared with someone emptying a box of caltrops onto the road: the tire manufacturer isn't responsible for any ensuing carnage, the hooligan is.

> Which is fair enough, but if a locksmith changed the locks on the door to your apartment for you but used locks that are stupidly easy to pick, there should and probably would be a product liability case to answer, at least if you were subsequently robbed and the locksmith had chosen the locks for you without making the risk clear. The robber abused defects to commit a crime: but the lock-smith still had some responsibility to ensure that's not easy for the robber to succeed.

> > Many of the defects ... exploited by worms and viruses would not
> > have any effect on the software product if the product were used
> > properly.

> Here's the rub: and it more or less leaves the question wide open. A major difference between software and car tires is what's meant by "using the product properly."

> With cars, it's easy enough: if you're driving it on a well-maintained

> public road at a speed safe for the road conditions in the absence of ice, oil spills, caltrops, shards of glass or other anomalous circumstances, you're using the product properly. There are a few other circumstances when you can claim you were using it properly, but the above covers pretty much all the use that's ever made of car tires; and the exceptions have to do with minor degradation in some of the constraints listed.

> With locks it's not as clear cut, though I dare say there's at least rather more case law than with software; and I would argue that security holes in software should be considered alongside locks rather than car tires; though the situation with software is somewhat weirder.

> To illustrate.... My webserver recently (on 2001 August 5th) received a request:

> GET /default.ida?...%u00=a HTTP/1.0

> where the ... comprised a sequence of 224 'N' characters, followed by a sequence of 22 tokens of form %uxxxx, with each x replaced by a hex digit, 0-9 or a-f. In fact, the 22 tokens in question were: %u9090 %u6858 %ucbd3 %u7801 %u9090 %u6858 %ucbd3 %u7801 %u9090 %u6858 %ucbd3 %u7801 %u9090 %u9090 %u8190 %u00c3 %u0003 %u8b00 %u531b %u53ff %u0078 %u0000.

> Now, as I understand it, this is a well-formed HTTP request. Not a very pretty one, I'll grant -- but take a look at some of the links coming off the latest Crypto-Gram and notice that quite a lot of them devolve into little more than a sequence of punctuators and gibberish after the initial protocol://site.domain, possibly with a :port and maybe some of the path. It's not all that weird after all. Albeit, I grant, it's by no means innocent.

> I have a webserver installed on my computer and I use it properly. That is, my computer is configured to delegate, to it, the handling of incoming connection requests on port 80, the default port for HTTP. Since the above was a valid HTTP request, addressed to that port, I would claim that all parties (including the requestor) were "using it properly."

> I am pleased to say that I don't run IIS, so my webserver duly logged the request, along with its time, date, originating IP address (I sent abuse@ this a polite e-mail suggesting they check for a virus or worm), the HTTP response code (404) returned and the number of bytes transmitted in that response. My webserver behaved properly.

> Yet a default IIS installation receiving the same request (as I understand it, this was the Code Red worm's opening gambit) would be hijacked by a program the installation's owner probably didn't consent to have running on their computer.

> Under the circumstances, I cannot see how IIS was "not being used properly" in this case. I grant that some other software -- something invoked as a result of accessing /default.ida on an IIS box -- was, indeed, not being used properly. However, since Microsoft's installation process for IIS automagically made arrangements for IIS to delegate to that other software in such cases, without warning the user that this might involve extra risks, the party who wasn't using something properly was Microsoft (albeit via its software); it seems to me that this really does resemble the Firestone tire situation.

> I won't deny there's scope for further argument here: I'll only claim that the water's a lot muddier than in the case of car tires. I suspect that similar arguments can be applied to Outlook and many other favorites of the builders of worms and viruses.

> On the other hand, I'd have to agree with Buck about the perils of

> extending product liability to software; it's sufficiently complex we'd be unable to release any software for fear of product liability suits. I guess that's why everyone and his dog has a license which disclaims all responsibility -- a practice for which I have more patience when the product's source code is openly available than when the source code is kept private -- to avoid such suits.

> > Lawyers would absolutely love having the ability to bring massive
> > class action lawsuits against Microsoft, IBM, Adobe, Apple, and
> > others (even Red Hat) to hold them liable for the damages caused
> > by flaws exploited by worms, viruses, and other malicious code.
> > They would make a fortune, and we the consumers would be paying
> > the legal bills through higher software prices.

> In the aftermath of each worm attack, we get to hear reports in the press of how billions of dollars were lost as a result of the attack. Now, I'm somewhat inclined to suspect such claims are inflated, but in any case doubtless there are some decidedly non-trivial losses being made; and a good lawyer could make them sound about as big as gets claimed.

> Much of the loss is borne by corporations. The loss is suffered because they were using Microsoft's products, which have a consistent history of security failings. There are less risk-rich software solutions available, especially when it comes to IIS's competitors, so much of that loss was "avoidable." The corporations' executives (and employees) have a duty of care, to the shareholders, to avoid expensive risks, where possible. The fact that a crime was committed against the corporation doesn't protect them, any more than if a robbery had only been possible because the staff had all consistently been sloppy about locking doors, windows, safes, etc. when leaving their offices.

> So I guess it's time some shareholders sued some corporations, their

> boards or their IT managers over a failure of "due diligence" causing actual corporate losses (as reported in the press). Or some lawyers could start a class action suit on behalf of some shareholders.

> Since Microsoft does use some of its own software and does suffer lossage sometimes through worms, viruses, etc., it would be particularly poignant to see its shareholders suing its executives over the culpable irresponsibility of using its products.

CRYPTO-GRAM is a free monthly newsletter providing summaries, analyses, insights, and commentaries on computer security and cryptography. Back issues are available on <http://www.schneier.com/crypto-gram.html>.

To subscribe, visit <http://www.schneier.com/crypto-gram.html> or send a blank message to crypto-gram-subscribe@chaparraltree.com. To unsubscribe, visit <http://www.schneier.com/crypto-gram-faq.html>.

Please feel free to forward CRYPTO-GRAM to colleagues and friends who will find it valuable. Permission is granted to reprint CRYPTO-GRAM, as long as it is reprinted in its entirety.

CRYPTO-GRAM is written by Bruce Schneier. Schneier is founder and CTO of Counterpane Internet Security Inc., the author of "Secrets and Lies" and "Applied Cryptography," and an inventor of the Blowfish, Twofish, and Yarrow algorithms. He is a member of the Advisory Board of the Electronic Privacy Information Center (EPIC). He is a frequent writer and lecturer on computer security and cryptography.

Counterpane Internet Security, Inc. is the world leader in Managed Security Monitoring. Counterpane's expert security analysts protect networks for Fortune 1000 companies world-wide.

<http://www.counterpane.com/>

next issue
previous issue
back to Crypto-Gram index

Schneier.com is a personal website. Opinions expressed are not necessarily those of BT.