March 15, 2002

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:

SNMP Vulnerabilities

SNMP is the Simple Network Management Protocol, the most popular protocol to manage network devices. Hundreds, possibly thousands, of products use it. Last fall, a group of Finnish researchers discovered multiple vulnerabilities in SNMP. By exploiting the vulnerabilities, an attacker could cause a denial-of-service attack, and in some cases take over control of the system.

The vulnerabilities concerns SNMP's trap-handling and request-handling functions, and stem from problems in the reference code (probably) used inside the Abstract Syntax Notation (ASN.1) and Basic Encoding Rules (BER). The SNMP vulnerabilities affect hundreds of different devices: operating systems, network equipment, software packages, even things like digital cameras. It's a BIG deal.

It's actually a bigger deal than has been reported. ASN.1 is used inside a lot of other applications, such as OpenSSL. These vulnerabilities aren't limited to SNMPv1; that's just the only thing that's been well-publicized at this point. (The recently reported problems in mod_ssl and Apache are apparently related to this, too.)

The history of the vulnerability's discovery and publication is an interesting story, and illustrates the tension between bug secrecy and full disclosure. A research group from the Oulu University Secure Programming Group in Oulu, Finland, first discovered this problem in October 2001, and decided not to publish because it was such a large problem. CERT took on the task of coordinating the fix with the major software vendors, and has said that the reason publication was delayed so long is that there were so many vendors to contact. CERT even had problems with vendors not taking the problem seriously, and had to spend considerable effort to get the right people to pay attention. Lesson #1: If bugs are secret, many vendors won't bother patching their systems.

The vulnerability was published on 12 February. Supposedly, this was two weeks earlier than planned, and because the story was leaking too much. CERT felt that early publication was better than widespread rumors. Some companies were caught off-guard. Even though they had months to patch their systems, they weren't ready and needed those two extra weeks. Some companies didn't bother to start worrying about the problem until publication was imminent. Lesson #2: It is only the threat of publication that makes many vendors patch their systems. (To be fair, many companies did a great job proactively patching their systems. And in many cases, the patches were not trivial. Some vendors were swamped by the sheer number of different products and releases they had to patch and test. And I stress "test", because patching mature code carries a strong probability of either not fixing the problem or of introducing new problems.)

When CERT finally published and the Oulu Web site went live, there were all sorts of reactions. Some tried to capitalize on the announcement to sell their products; others tried to minimize it. Many vendors had no idea if they were vulnerable or not But because publication included demonstration code -- the PROTOS tool -- vendors and security companies were able to test networks and equipment. Lesson #3: Publication must include enough information to reproduce the vulnerability; otherwise, there's no way for anyone to determine how serious the threat is. And Lesson #4: If there is no way to independently verify the vulnerability, then organizations are forced to rely on information from potentially biased sources.

As of this writing, there have been no credible reports of this vulnerability being exploited in the wild. Counterpane's monitoring has not detected any of our customers being attacked via this vulnerability. This is not to say that no one has -- writing an attack tool is a straightforward programming task -- but no one has published such a tool and put it in the hands of the script kiddies. Lesson #5: Publication does not automatically mean the vulnerability will be exploited.

So far we've been lucky. But a tool could show up at any time, so relying on that luck would not be smart. And even though everyone has been urged to patch their systems and products, some will not. Even if it takes months before someone writes an attack tool, it will work against a surprisingly large subset of systems. Lesson #6: Publication increases the likelihood that a vulnerability will be exploited.

And there are a lot of systems for which patches will never be available. Many router vendors have gone out of business in the last few years, and not every mom-and-pop software company out there has the money or clue to replace their hardware because their code has a problem. Lesson #7: Since many, many systems will remain unpatched, this vulnerability will pose a risk for years to come.

At Counterpane, we were able to make use of the public demonstration code to quickly write filters for our Sentry and push them down to our customers' networks. We did this within hours, so even if they didn't patch their systems we could monitor them for evidence of exploitation. We patched our own Sentry. This wasn't perfect -- in some systems the attack didn't show up in their audit logs -- but it let us know which systems would benefit from other security tools, like IDS signatures tuned to detect the PROTOS tool. We collected and maintained a list of intrusion detection signatures for Snort, RealSecure, CiscoIDS, Network Flight Recorder, etc., that were specifically designed to collect the PROTOS' tool's test packets.

We also sent out an advisory to our customers -- a voice of reason among the slightly hysterical news articles -- and made our Network Intelligence group available in a conference call to reassure them. We made our research available to the FBI and other security organizations. Lesson #8: Vigilant monitoring provides another layer of security, if and when products and patches fail.

While these vulnerabilities are serious, the fact that SNMP is vulnerable should not come as a surprise to anyone. Vulnerability "U7" in the SANS Top 20 talks about SNMP. SANS's recommendation: "If you do not absolutely require SNMP, disable it." This was good advice when the list was released, and it's good advice now.

CERT advisory:
<http://www.cert.org/advisories/CA-2002-03.html>
<http://www.cert.org/tech_tips/snmp_faq.html>

Counterpane advisory:
<http://www.counterpane.com/alert-snmp.html>

Oulu's analysis and PROTOS test suite:
<http://www.ee.oulu.fi/research/ouspg/protos/testing/...>

Analyses and articles:
<http://techupdate.zdnet.com/techupdate/stories/main/...>
<http://www.theregister.co.uk/content/4/24167.html>
<http://www.infoworld.com/articles/op/xml/02/03/04/...>

"Responsible Disclosure" IETF Document

The so-called "Responsible Disclosure" IETF document has been released as a draft. Despite some of the writings to the contrary, it is not a blueprint for companies to keep their vulnerabilities secret. It's one view of how a vulnerability should be released: first to the vendor and then, after a reasonable amount of time has passed, to the general public.

In general, I agree with the philosophy of the document. I want vendors to have time to prepare patches before vulnerabilities are made public. At the same time, I don't want publication to be limited in any way. This document attempts to strike a balance, and I think it does a good job.

My complaints are more about procedure than content. I don't think this makes any sense as an IETF document. The IETF is a standards body; they're good at specifications about which bits go where, but have no business making political proposals.

I don't like the way the document is written: there are so many suggestions and so few requirements that someone could do almost anything and then claim to be following the document. And I believe the document could be used by companies to justify withholding vulnerability information and ignoring security problems. In my account of the SNMP vulnerability above, you can see how the threat of full disclosure is what forced companies to patch their systems. To the extent that following the IETF document reduces that threat, it also reduces our security.

And I think the title is wrong. It's "Limited Disclosure," or maybe "Slow Disclosure." Whether or not it is responsible is still a matter of debate.

I know there are several alternative proposals being drafted. I would like to see some kind of single consensus document at the end of this process, but I don't think the IETF is the place to reach that consensus.

Draft Document
<http://www.ietf.org/internet-drafts/...>

<http://zdnet.com.com/2100-1105-842656.html>
<http://www.computerworld.com/storyba/...>
<http://www.eweek.com/article/...>

Crypto-Gram Reprints

Security patch treadmill:
<http://www.schneier.com/crypto-gram-0103.html#1>

Insurance and the future of network security:

<http://www.schneier.com/crypto-gram-0103.html#3>

The "death" of IDSs:
<http://www.schneier.com/crypto-gram-0103.html#9>

802.11 security:
<http://www.schneier.com/crypto-gram-0103.html#10>

Software complexity and security:
<http://www.schneier.com/...>

Why the worst cryptography is in systems that pass initial cryptanalysis:
<http://www.schneier.com/...>

Terrorists, Cryptography, and Export Laws

For years, we cryptographers have been saying that limiting export of cryptography to 40 bits is ineffectual, because terrorists aren't stupid enough to use broken 40-bit keys. So, imagine my surprise when we learn that a terrorist is stupid enough to use 40-bit keys.

Turns out that the Richard Reid, the inept shoe-bomber terrorist, encrypted his PC files using 40-bit keys. And they were brute forced.

It's easy to misinterpret this. The lesson here is not that all terrorists are stupid, only that there exists at least one stupid terrorist. Despite this deliciously ironic anecdote, the benefits to society of ubiquitous strong encryption outweigh the disadvantages of not being able to decrypt the hard drives of captured terrorists.

<http://www.newscientist.com/news/news.jsp?id=ns99991804>
<http://news.independent.co.uk/world/americas/...>
<http://slashdot.org/articles/02/01/18/146229.shtml>

News

A DoD study shows that face scanning only works about 50% of the time under the best conditions:
<http://www.wired.com/news/politics/0,1283,50470,00.html>

EPIC's National ID Card resource page:
<http://www.epic.org/privacy/id_cards/>

Internet eavesdropping on a massive scale. Turns out that Comcast has been secretly recording the surfing habits -- every Web page visited -- of its million high-speed customers. They're doing this for performance purposes, but this data would be available to police and the FBI with a court order, and to lawyers in civil lawsuits.
<http://www.siliconvalley.com/mld/siliconvalley/...>
At least one Congressman, Rep Markey, wants to make this practice illegal:
<http://www.usatoday.com/life/cyber/tech/2002/02/13/...>
A couple of days after this news story broke, Comcast said they would no longer do this:
<http://www.nytimes.com/2002/02/14/technology/...>

Good story on the security risks of the SOAP protocol:
<http://www.prescod.net/rest/security.html>

Security holes in the anonymous Web-surfing program SafeWeb:
<http://www.wired.com/news/politics/0,1283,50371,00.html>
<http://www.theregister.co.uk/content/6/24105.html>
<http://www.cs.bu.edu/techreports/pdf/...>

Interesting story about airplane security. Air marshals took control of a plane bound for Salt Lake City, and one skeptical passenger didn't believe them. How do you verify credentials in a situation like this?
<http://www.salon.com/people/feature/2002/02/20/...>

Outsourcing security: dos, don'ts, and advice
<http://www.networknews.co.uk/Analysis/1129412>

Nice essay on the full disclosure debate:
<http://www.infosecnews.com/opinion/2002/02/27_02.htm>

Scarfo pleads guilty, so there'll be no appellate court consideration of the FBI's key logger and its warrant requirements.
<http://story.news.yahoo.com/news?tmpl=story&u=/...>

Interesting Macintosh vulnerability. These are cascades of fairly innocuous features that combine to cause problems.
<http://homepage.mac.com/vm_converter/...>

Good article on investigative techniques, and problems, for solving cybercrimes.
<http://www.osopinion.com/perl/story/16502.html>

An Oracle study concludes that insiders are worse than hackers. This should be no surprise.
<http://www.networknews.co.uk/News/1129574>
<http://www.theregus.com/content/55/24212.html>
<http://oraclepressoffice.bulletonline.com/...>

Latest bogus security challenge. Too stupid for even the Doghouse:
<http://biz.yahoo.com/bw/020301/12094_1.html>

Excellent (and long) paper on a risk-management approach to computer security:
<http://cisac.stanford.edu/docs/soohoo.pdf>

Microsoft is delaying .NET, citing security concerns as one of the reasons. Assuming this is a real reason, and not an invented reason for the press, this is good news. Possibly it's even a harbinger of a new way of doing business in Redmond.

<http://www.pcmag.com/article/...>
I'm still skeptical, but hopeful.
<http://news.zdnet.co.uk/story/0,,t269-s2105503,00.html>

Two different papers on optical information leakage, one from CRT displays and the other from LEDs. Information can be read at a distance, without physical contact to the target display. This kind of thing is analogous to electrical emanations, aka TEMPEST.
<http://www.cl.cam.ac.uk/~mgk25/ieee02-optical.pdf>
<http://applied-math.org/optical_tempest.pdf>
Several reporters have asked me how big a deal I thought this is. Short answer: not very. I have no way of defending myself against attackers this well motivated and this well funded. They can already park a van outside my house and eavesdrop on my computer. They can already break into my house when I'm away and install dozens of listening devices and keyboard monitors and who-knows-what-have-you. So, now they have another way they can eavesdrop on me. I still can't do anything about any of them. At least, not without spending a WHOLE lot of money.

Network Associates has killed PGP:
<http://www.zdnet.com/anchordesk/stories/story/...>
<http://www.computerworld.com/storyba/...>
This doesn't mean that PGP is dead, mind you. The OpenPGP standard, and GPG, are still going strong.
<http://www.theregister.co.uk/content/54/24336.html>
<http://www.openpgp.org/>

A really clever social-engineering hack. Nothing to do with computers, but well worth reading.
<http://seattletimes.nwsource.com/html/localnews/...>

Excellent paper on software liabilities, written by a couple of attorneys and someone from CERT. Required reading.
<http://www.cert.org/archive/pdf/...>

Counterpane News

Schneier will be presenting Counterpane's Managed Security Monitoring service in the following cities: Austin, Boston, Dallas, Chicago, Minneapolis, New York, and Tallahassee. Visit this page to see dates, and to sign up:
<http://www.counterpane.com/seminars.html>

There are several new case studies from Counterpane customers here:
<http://www.counterpane.com/experiences.html>

Assorted Counterpane partner/customer press releases:
<http://www.counterpane.com/pr-neccisco.html>
<http://www.counterpane.com/pr-dyntek.html>

Bernstein's Factoring Breakthrough?

Last fall, mathematician Dan Bernstein circulated a paper discussing improvements in integer factorization, using specialized parallel hardware. The paper didn't get much attention until recently, when discussions sprang up on SlashDot and other Internet forums about the results. A naive read of the paper implies that factoring is now significantly easier using the machine described in the paper, and that keys as long as 2048 bits can now be broken.

This is not the case. The improvements described in Bernstein's paper are unlikely to produce the claimed speed improvements for practically useful numbers.

Currently the fastest factoring algorithm is the Number Field Sieve (NFS), which supplanted the Quadratic Sieve several years ago. Basically, the NFS has two phases. The first is a search for equations that satisfy certain mathematical properties. This step is highly parallelizable, and today is routinely done with thousands of computers. The second step is a large matrix calculation, which eventually produces the prime factors of the target number.

Bernstein attempts to improve the efficiency of both steps. There are some good observations here that will result in some minor speedups in factoring, but the enormous improvements claimed are more a result of redefining efficiency than anything else. Bernstein positions his results as an effect of massive parallization. To me, this is misleading. You can always simulate a parallel machine on a single computer by using a time-sliced architecture. In his model, the "cost" of factoring is a product of time and space, and he claims that he can reduce the cost of parallel sorting from a factor of m^4 to m^3. Bernstein justifies his assumptions by claiming that a single processor needs m^2 memory, whereas an array of m^2 processors only needs constant memory. This may be true, but neglects to factor in the cost associated with connecting those processors: tying a million simple processors together is much more expensive than using a single processor of the same design with a million bits of memory. Again, it is not clear that this technique will buy you anything for practical sized numbers.

To be sure, Bernstein does not say anything different. (In fact, I commend him for not being part of the hyperbole.) His result is asymptotic. This means that it is eventually true, as the size of the number factored approaches infinity. This says nothing about how much more efficient Bernstein's algorithm is, or even whether or not it is more efficient than current techniques. Bernstein himself says this in one of his posts: "Protecting against [these techniques] means switching from n-bit keys to f(n)-bit keys. I'd like to emphasize that, at this point, very little is known about the function f. It's clear that f(n) is approximately (3.009...)n for *very* large sizes n, but I don't know whether f(n) is larger than n for *useful* sizes n." What he means is: at some bit length these techniques will be useful, but we have no idea what that bit length is.

I don't believe in the factor of n - 3n length improvement. Any practical implementation of these techniques depends heavily on complicated technological assumptions and tradeoffs. Parallel computing is much easier to say than it is to do, and there are always hidden complexities. I think when all the math is said and done, these other complexities will even out his enhancements.

This is not to belittle Bernstein's work. This is good research. I like his novel way of using sorting techniques to carry out the linear algebra part. This might be useful in a variety of other contexts, and is likely to open up new research directions in the design of more efficient sorting networks and sparse matrix algorithms. There are other speed improvements to the NFS in this paper, and they will most definitely be researched further.

Over the past several decades factoring has steadily gotten easier, and it's gotten easier faster than anyone would have believed. Speed improvements have come from four sources. One, processors have gotten faster. Two, processors have gotten cheaper and easier to network in parallel computations. Three, there have been steady flows of minor improvements to the factoring algorithms. And four, there have been fundamental advances in the mathematics of factoring.

I believe that Bernstein's work falls under the third category, and takes advantage of ancillary improvements in the second category. And if history is any guide, it will be years before anyone knows exactly whether, and how, this work will affect the actual factoring of practical numbers.

Bernstein Paper:
<http://cr.yp.to/papers/nfscircuit.ps>

Richard Clarke on 9/11's Lessons

At this year's RSA Conference, Richard Clarke gave the keynote address on cybersecurity in the aftermath of September 11th. The beginning of his talk was excellent: he listed six lessons of the terrorist attacks. It started falling apart when he states that businesses should, out of the goodness of their hearts and concern for their way of life, produce and use more secure products. And it collapsed further when he advocated a new exemption to the Freedom of Information Act that everyone who studies the issue claims is unnecessary and harmful. But his opening was good.

Here are Clarke's six lessons. They don't just apply to combating cyber-terrorism, and I will explain them in terms of everyday network security.

1. "We have enemies." Everyone does. Companies have competitors. People have others who don't like them. Some enemies target us by name, others simply want to rob someone and don't care whom. Too many organizations justify their inattention to security by saying: "Who would want to attack us?" That just doesn't make sense.

2. "Don't underestimate them." Don't. Whether it is a DVD pirate living in a country with no copyright laws, or a hacker kid who spends days trying to break into a network, cyberspace attackers have proven to be better funded, smarter, and more tenacious than anyone has estimated. If you assume that your enemies won't be able to figure out your defenses and bypass them, you're not paying attention.

3. "They will use our technology against us." This is especially true in cyberspace. Almost all attacks involve using the very network being attacked. Maybe it's a vulnerability in the software; maybe it's a feature that should never have been created. Hacking is judo: using network software to do things it was never intended to do.

4. "They will attack the seams of our technology." As bad as most cryptography is out there, it's almost always easier to break a system by some other method. Attacks on the seams -- the places where different

technologies come together -- are more fruitful. Think of the FBI reading PGP-encrypted mail by installing a keyboard sniffer, or people who bypass copy-protection controls by mimicking them rather than breaking them. This lesson is obvious to anyone who has broken security software.

5. "Our technology is surprisingly interdependent." That's certainly clear. We've seen vulnerabilities in IIS affect all sorts of systems. We've seen malicious code use features of Microsoft Word and Outlook to spread. A single SNMP vulnerability affects hundreds of products. Interdependence is how the Internet works. It's also how it fails.

6. "The only way to solve this problem is for government and industry to work together." This is more subjective, but I agree with it. I don't think that industry can do it alone, mostly because they have no incentive to do it. I don't think that government can do it alone, because they don't have the capability. Clarke seems to think that it's government's job to provide some funding, high-level coordination, and general cheerleading. I think it's government's job to provide a financial incentive to business. If you want to fix network security, hack the business model. Remove the liability exemptions from software. Demand regular reporting similar to what was required for Y2K. Make the CEO care.

Clarke spends a lot of time visiting companies and talking to them about security. I'm sure the CEOs give him a warm reception, and tell him that they'll make their stuff more secure. But what happens a month later, when budgets are tight and a release date looms? Will the CEO remember his promise to Clarke, or will he listen to the demands of the market? If the government really wants the CEO to care, it's going to have to make security a market force.

I don't see any other way.

Comments from Readers

> **From: "Douglas St.Clair" <dwstclair@tellink.net>Subject: RE: CRYPTO-GRAM, February 15, 2002**

> I taught the course in Systems Safety Analysis at Picatinny Arsenal to nuclear, biological, chemical, and conventional weapons designers in the late 1960s. There are two points I want to make based on this experience.

> First, the distinction between safety and reliability is based on time and place, not the design. For example, consider a faulty door on a manned NASA vehicle. If the door falls off while the vehicle is being moved to the launch site, it is unreliable. If the door fails in space and men die, then it is an unsafe door.

> Now consider how things get designed. Let's consider weapons systems because, for one reason, it gets us away from the love/hate Microsoft issues. Somebody wants to put a missile on a Humvee. The safety guy wants at least two switches be thrown before the missile fires and he wants them placed so it takes two men to do it at the same time. The reliability guy wants switches all over the vehicle so that if one of them is thrown the little rascal is underway. Both guys can argue reasonably for their position. In the end, since they can't agree, some manager who may not understand the technical issues will make a decision.

> I think that the issue also revolves around the genetic makeup of the players. The ideal safety guy measures risk in terms of the magnitude of regret. The ideal reliability guy considers magnitude of risk. If the risk is less than some limit he will accept it. Managers are encouraged to make decisions, in the absence of good data, no matter what the risk. I think these traits are coded in the genes of the principals.

> Before any substantial change can be made in Microsoft, the company's culture needs to change and that means the players will

need to change too. I don't see Microsoft firing and hiring the substantial number of people required to make a change in culture possible. In the end, Barne's Law, "If nothing changes, everything will remain the same," will win out.

From: "Mike Mathison" <MikeMathison@btinternet.com>Subject: RE: Judging Microsoft

Forgive me, I'm not a security expert, but I can't help summarizing some of your advice on how we are to gauge Microsoft's progress towards security salvation as follows:

* The less useful Microsoft makes its products and the more features it removes from them the more virtuous it becomes.

* The more difficult Microsoft makes its products to use and configure (lots of switch-able options and add-ins) the more virtuous it becomes.

* The more Microsoft abandons its new key strategic technology direction (universal Web services over ubiquitous HTTP with smart universal clients) the more virtuous it becomes.

If I were Microsoft, I would conclude that if you are correct and they really have to do all of this, then they are doomed, and so I guess I would instead be hoping that you are wrong (or at least a little overzealous in your triumph at their admission of sin).

Incidentally, I do sometimes wonder if some of the computer security industry _really_ wants Microsoft to sort out its problems or is just paying lip service and having a good time whining. Let's not forget there was a time when Microsoft neither knew nor cared very much about the Internet. Those were the days when Netscape was verging on a browser monopoly and you needed to know or pay someone with a degree in electronic engineering to set up a dialup connection on a PC. No one argues that Bill Gates was wrong to change his company's direction then, and although it was late in the day he did succeed, despite widespread skepticism. He's neither stupid nor lazy (there, I said it).

From: Mark Reynolds <mark@aoainc.com>Subject: CRYPTO-GRAM Feb 15, 2002: Comment

In the most recent issue of CRYPTO-GRAM, in the article "Judging Microsoft," it is stated that "Originally, e-mail was text only, and e-mail viruses were impossible," as part of the discussion on recommended security improvements. While the first part of this statement is definitely accurate, I would disagree with the second part.

Even very early Unix(tm) systems had the ability to define a "virtual" user that was actually an alias to the standard input of a program. Thus, an entry such as

print "| /usr/bin/lpr"

in the aliases file would cause all e-mail sent to "print" to be sent to the stdin of /usr/bin/lpr. It was not uncommon for people to attempt, and occasionally succeed, in creating a specially formatted text-only e-mail that would exploit a vulnerability in the destination program. The fact that e-mail at that time was not really 8-bit clean only added to the challenge.

While this was not strictly speaking a virus, it was certainly a security vulnerability, and given the ability to exploit buffer overflows in lpr, for example, it is not a huge leap to imagine a self-propagating e-mail message that would be able to infect any site that defined such a well-known alias, and that also had a network connection. I do not know if this was ever actually done back then, but it was certainly made

> possible as soon as the ability to have piped (|) output in an e-mail alias was added.

> **From: "Andrew van der Stock" <ajv@greebo.net>Subject: Factual Mistakes in Microsoft Analysis**

> The central thesis of the paper is correct, and the suggestions for improvement worthy, but some of the facts are simply wrong, and need to be rectified.

> "...too many Microsoft server components run as Administrator" -- most services as shipped from MS use LOCALSYSTEM, not "Administrator." LOCALSYSTEM is far more privileged than Administrator, except that it cannot use SMB-based networking -- unless it implements its own SMB stack. The risk of using LOCALSYSTEM is somewhat worse than running as Administrator as it can act as part of the operating system. The .NET CLR runs as MACHINE, an analogous account to LOCALSYSTEM, but with few privileges.

> "Microsoft should drop the code-signing security paradigm in favor of the sandbox paradigm." No, this is called defense in depth. Both code signing and sandbox paradigm (such as Java or .NET's CLR model) should be implemented and used. ActiveX as it stands today is un-securable, and should be disabled by default unless a user explicitly requires its services.

> "Too much of it functions without leaving audit records of what happened." This is wrong, and demonstrates beyond reasonable doubt the authors' lack of experience with the platform they are slagging off. Use Event Viewer. You'd be surprised how much stuff is in there, even by default. Everything that the Object Manager grants access to (which is everything) is auditable, you just need the right tool to turn on auditing if it doesn't already have a GUI to do so. The MS BackOffice

> products produce large quantities of very visible and useful logging. Just because it isn't syslog doesn't mean it doesn't exist.

> Here's a picture of my machine giving adequate information about how it is running. I could have chosen any one of the other 418 events in this log, or the 1910 events in the System event log, or the 893 events in the security event log (which are pure auditing in the TCB sense).

> <http://www.evilsecurity.com/eventvwr.jpg>

> "Microsoft should drop all plans for automatic software updates via the Internet until they can be done securely and reliably." The risk of all those machines out there on the Internet without any updates at all, far outweighs the risks of individual machine death. And for the vast, vast majority of all users, the updates work as planned. Sure, when things go wrong, it's painful for the individual, but the pain suffered by the whole is much less. The authors obviously have a libertarian bias, and should acknowledge that the Internet is a commons to be shared. The only way to share a commons is to minimize the risk from bad elements. If this means individuals assuming greater risk for a resource they own, so be it. I'd rather be on a network where there aren't thousands of zombies waiting to attack me because of insecurities caused by a vendor. Let them fix the mess in the best way they can. Most end users (think of your mum) can't manage this process (through learned help
> lessness more than anything else), and in some ways from a human-computer interface and fair trading laws point of view, they shouldn't have to. Like a tire recall -- it's your responsibility to show up at the dealer, and then it's the dealer's responsibility to swap the faulty parts out. End of story for *consumers*. Corporates already block Windows Update through group policy, or simply by not letting their users log on as Administrator-level accounts.

> "We would also like to see Microsoft abandon the Registry in favor of a

> less opaque and more user-friendly system." No, I don't think so. The authors offer no replacement, but there's only one other mainstream alternative: /etc. Don't make me laugh! The fundamentals of a single remotely administratable place for settings are vital for large scale deployments. There are increased risks for remote registry attacks, but the other issues like undocumented settings, etc. (which are valid complaints) can be addressed by the authors of programs -- if they choose to make the registry settings (or file format) known, then great. The registry is not end-user-friendly for a very simple reason: there's no valid reason for people to go there.

> It's like the ECU in your car, sure there's some fun to be had if you know exactly what you're doing with someone else's software, but there's much damage to be done if you have no clue. And most of MS's users are completely unaware that the registry exists. The trick is for security professionals like us to manage the risk of managing settings database or files. And they aren't the only vendor to suffer from poor QA in this area: fire up a stock RH 7.2 box and use vipw -- /etc/shadow becomes world readable. Permissions are problematic the world over.

> **From: Adam Shostack <adam@zeroknowledge.com>Subject: Re: Rebuttal to our Microsoft Article**

> Andrew van der Stock makes a number of good points in the above comment. One of the challenges in writing an article as we did is the difficulty of balancing a high level approach with never talking about details. For example, his comments about localsystem are correct, but how important is this? The reality is that too many services run with too much privilege. Our argument for less privilege is not weakened by the details of localsystem vs. administrator. These services should run as a user explicitly tagged as "daemon subject to attack," and appropriately restricted.

> Regarding code-signing of all ActiveX code, we simply don't agree. Code signing from arbitrary providers of code requires a lot of PKI effort for little return. Bruce's previous comments in Crypto-Gram about SSL and its PKI come to mind; who cares if the code you have is signed by someone you've never heard of? (Signed updates from a vendor whose code you already run is another matter.)

> Regarding logging, once again we have a level problem. It is challenging to make a Windows system log to a remote device. (There are now excellent programs available that do this, and there is an issue of how much Microsoft should provide vs. letting the market step in.) And while syslog is insecure and unreliable, it at least makes it easy to send the information to another system, where it can't be altered by malicious code. We could have talked about this, and Andrew's provision of details is certainly useful in reminding us the risks of such an essay. However, each paragraph was cut fairly mercilessly of technical details, and that led to some statements that are easy to misinterpret.

> Andrew doesn't mention that we also made some mistakes in our discussion of the SOAP protocol, which uses HTTP as one of its many transports. Again, this was a matter of trying both to keep our comments at a high level and to touch on some details, and we messed up.

> **From: "John.Deters" <John.Deters@target.com>Subject: Microsoft and Security**

> You freely offer Microsoft a laundry list of top-level fixes that could help restore security. The following link illustrates Microsoft's current lack of commitment to security better than any other example I've seen.

> <http://support.microsoft.com/default.aspx?...>

The problem includes a list of 28 games that may hang when you try to play them. The relevant message is found in this "solution":

>To resolve this issue, do one of the following:
>
>If you are using Windows NT, Windows 2000, or Windows XP please
>log off the computer, and then log on to the computer as a user
>with Administrator rights.

So according to Microsoft, BY DESIGN I must be an admin to play Train Simulator (or any of the 28 games listed at the above URL.) All these games have associated Web sites, and indeed have URLs in the code that encourage you to "click here to visit the Microsoft Web site" for various reasons. So even if you tried HARD to be safe, you are still likely to fire up IE and hit the Web as an administrator.

Why should a game require a player to be an administrator? Who could possibly have manufactured a larger security hole?

From: Scott and Karleen James <smjkkj@attbi.com>Subject: Liability

As I read the pros and cons for software liability, I have to stop and think about how other industries have to deal with product production and reliability issues. I am not specifically trained in the security area but I have been in the real business world long enough to make what I hope are some valid observations.

I worked in the software industry. We sold newspaper publishing software. Effects of coding errors ranged from minor to catastrophic. Minor issues like text not being drawn in exactly the correct location often had simple work-arounds or were handled in the next upgrade of the software system. Catastrophic issues usually involved putting someone on-site. These were situations where the database contained

> garbage, or no usable film was being produced. Newspaper people get really anxious when their software can't reliably produce output so that they can publish a paper everyday. They can't just change software like one would trade in a used car for a newer or different model. In any case, there seems to be relatively little danger to me personally for careless coding except that I might possibly lose my job if I screw up too often. Often I was at the mercy of code written by someone else, patched or improved by yet another person and now suspect or broken as a result of a change made in a module elsewhere in the several hundred thousand lines of code that made up our server/client system. A major problem could easily mean the loss of thousands of dollars of revenue, so ignoring the problem was never a choice.

> I also worked at the Johnson Space Center. I am quite aware of what Mission Critical software implies. In the world of NASA software, a major problem can put millions of dollars and people's lives at risk.

> Now for a different perspective, my wife works for a large petroleum refining company. She works in the health, safety, security and environment areas. I'd like to draw a comparison. If you have 10 programmers and nine of them are religiously following good design and coding practices for creating secure, reliable software but one of them isn't, what do you get? Clearly, the effects change depending on what areas the shoddy coder touches. Overall, though, I would expect a pretty good product. In the refining industry, if one guy in the facility doesn't follow procedures, he puts EVERYONE at risk. I realize my analogy is not a perfect one, but it gets one thinking in the right direction.

> The thing that will really improve software is when someone figures out how to establish a [more] direct link between the risks of using a product and the creation of the product. The software industry seems to think the tobacco industry business model is a good one. It's okay to

> kill your customers, there will always be a new one to replace the one you just lost. When an individual programmer associates the risks in using a product with the choices made during design and coding, then quality, security, and reliability will jump up dramatically. In my view, it seems like it is the corporate entity that takes the blame for the substandard work, not Frank or Bob or George or whoever.

> If more programmers treated the software like a skyscraper in which they lived on the top most floor, I'll bet they would be pretty keen on having a good foundation (design) and they would choose quality materials and skilled workers for construction (coding and testing). Having the building collapse with them in it is not a risk they want to take. Why should software be so different? When I wrote or modified code, I left my signature in the file because I was proud of my work. If there was something wrong with my work, I wanted to fix it.

> A final thought: ever wonder how much "programmer burnout" contributes to faulty software? Consider how long it takes to design, code, test, and release a software title. If there is high turnover in the skilled labor pool, what effect might that have on the product?

> **From: Marc Mutz <mutz@kde.org>Subject: Software Liability**

> Your last issue of Crypto-Gram was full of reader's comments as well as essays from you and others on why software companies should be held liable for bugs in their products.

> Despite all that hype for making companies liable for bugs and security vulnerabilities in their products, there's one thing I'd like to remind everyone of:

> If software companies were liable when their products broke, Open Source programmers would be much more hesitant to release their work to the public. The GPL explicitly denies all such claims against the

> authors of software released under that license.
>
> I'd be more that hesitant to work on OpenPGP-related code in KDE if I could be held liable for bugs, since a large-scale lawsuit against me would ruin my life, thank you.
>
> So, in the end we'd probably see a decrease in Open Source software deployment (since nobody's there to get money back from) and an increase in proprietary software deployment (since large companies have much money that can be pressed from them for buggy software, much like the tobacco industry is drained currently).

> **From: Mary Ann Davidson <Mary.Ann.Davidson@ oracle.com>Subject: Oracle's 14 Security Evaluations**

> In the February 15 Crypto-Gram that you say Oracle does not actually have 14 independent evaluations.
>
> The ads do say 14 independent evaluations. It is true that we have evaluated different versions of server products against the same criteria, because prior to the Common Criteria we had to do country-specific evaluations (US TCSEC or Orange Book, UK TCSEC and the occasional Russian Criteria evaluation). The UK would not accept US evaluations, for example. Happily, the Common Criteria (ISO-15408) is now recognized by multiple countries as the de facto evaluation standard, which means we only do Common Criteria evaluations plus the occasional FIPS-140 to validate cryptographic modules.
>
> However, we have indeed completed 14 of these independent evaluations. Each evaluation is separate and distinct. It is not as if one pays the fee and gets a certificate in the mail.
>
> We remain committed to evaluations: Oracle Label Security is in evaluation at EAL4, and we will be evaluating Oracle9i Database

> release 2 and Oracle9i Label Security release 2 at EAL4, as well as submitting Oracle9i Application Server for a FIPS-140 evaluation.

> I would also note that the US Federal government requires independent measures of assurance. National Security Telecommunications Information Systems Security Policy (NSTISSP) #11, which goes into full effect in July 2002, requires any system involved in national security to have independent measures of assurance (i.e., a security evaluation). In a post 9/11 world, there will be few waivers granted for this, and rightly so. I would hope that you would see the merit of someone other than a vendor attesting to their security claims. Apparently, the U.S. government and other governments do.

> For our server products, security evaluations represent about $1,000,000 apiece in additional security QA by someone other than Oracle. The larger benefit is that one has to change development processes to get through an evaluation, as the security of the development environment is part of what is assessed. Unlike other vendors, we did not feel compelled to have a new-found "security initiative," because our development processes were altered over 10 years ago as part of the evaluation process. We are, as it happens, extending our security development process across the rest of the stack. When the marketing campaign is done, we will still be committed to assurance, and a secure product lifecycle for every product in our stack. I can also assure you that we use security flaws as an opportunity for process improvement -- we are continually revising our coding standards and release checklists based on new information and threats (and that includes flaws that we or other research
> ers find in our products).

> Security people cannot have it both ways. The criticism is that "no

product is truly unbreakable" but perhaps the criticism ought to be "why don't more vendors draw a line in the sand and commit to bulletproof software, and commit to comply with relevant standards on 'what you means when you say you are secure?'" Is the glass half-empty or half-full? I am also aware that your firm focuses on assuming that there is no unbreakable software and monitoring for attacks accordingly. Surely there can be room for both in order to raise the security bar. Would you advocate no independent measures of assurance because even evaluations are not perfect? If I tell customers that unbreakable software is too hard, and we give up, it may indeed give your firm and others like it an increasing market, but it will be abdicating the responsibility that vendors MUST assume for their own product security. I agree with you that the hard stuff in security is very hard, but giving up because securi
ty is too hard is inexcusable. Too many have given up already.

From: Anonymous
Subject: Cloud Nine ISP

> Here's an example of a hack causing a company to go out of
> business. Cloud Nine, a UK ISP, ceased operations this week
> after being the victim of a DOS attack. The network needed
> to be rebuilt as a result of the attack, and the company's
> insurance wouldn't cover the repairs.

I must admit to being disappointed to read this in your newsletter. Go back to those two URLs at <www.ispreview.co.uk>. I guarantee your Snake Oil alarms will go off.

"We tried overnight to bring our Web servers back online but were seeing denial of service attacks against all our key servers, including email and DNS. These were of an extremely widespread nature.

"We felt we had a moral duty not to expose our customers to possible

> attacks as well."

> Since when was taking your customers' Internet connectivity away better for them than a denial-of-service attack?

> Searching <www.theregister.co.uk> comes up with a snippet of their initial statement:

> "First a firewall password brute force attack resulting in successful hash and destruction of the firewall"

> Hash? If I remember correctly this attack was meant to have been bounced from a Web server, but it seems those comments have had to be taken offline:

> <http://web.archive.org/web/20020215145613/http://...>

> To bathe in Snake Oil see:

> <http://www.ispreview.co.uk/cgi-bin/ubb2/...>

> **From: Nathan Neulinger <nneul@umr.edu>Subject: noticed something in Applied Cryptography**

> Happened to be re-reading Applied Cryptography, 2nd Edition, and noticed something creepy on p.99 in "The Politics of Key Escrow," 3rd paragraph.

> "Imagine a major terrorist attack in New York; what sorts of limits on the police would be thrown aside in the aftermath?"

> Just thought it was rather scary how right you can be without even realizing it.

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