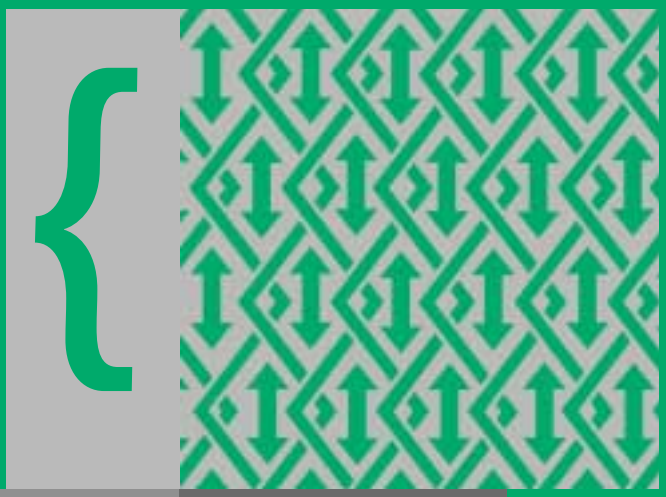
;login:

THE MAGAZINE OF USENIX & SAGE
October 2000 • volume 25 • number 6



inside:

SECURITY
The Network Police Blotter

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

the network police blotter

by Marcus J. Ranum

Marcus J. Ranum is CTO of Network Flight Recorder, Inc. He's the author of several security products and of a book on computer security (with Dan Geer and Avi Rubin) and is a part-time sysadmin.

<mjr@nfr.net>

Some Dirty Tricks

Two issues ago, I asked people to email me dirty burglar-alarm tricks from their networks. I've gotten a couple of really interesting suggestions, and I'm going to share a few of them with you. I'm editing the original emails down to a nubbin, in order to fit them in this space, so if any inaccuracies are induced, it's probably my error.

The inimitable Gene Spafford sent me a ton of good suggestions, including creating a bogus user and stuffing its /var/spool/mail file with a few fake messages from root, including discussion of various machines' root passwords. The mail files were watched by a separate process that detected when they were accessed – in those days intruders would frequently try a "grep password /var/spool/mail/*" upon entering a new machine. He also left a specialized program in the user's bin directory that looked like a setuid access program, but which, in fact, would generate an alarm and freeze a copy of the process table, etc. Another delicious dirty trick from Gene was to doctor the crypto routines in a decompiled version of the Morris Internet Worm, so that they would simply waste the bad guy's CPU cycles instead of cracking passwords. Sure enough, someone stole the doctored copy, and, presumably, enjoyed a private moment of frustration when his ill-gotten software didn't work as expected!

Daniel Wesermann described how he once rigged the routing table on a bastion host to contain a number of attractive-looking routes to nonexistent subnets, which, in fact, pointed to a "black hole" Linux machine that did nothing but suck up the traffic directed to it. He tinkered with the source to netstat to make it look like there were even active connections from within the bogus subnets, to further confuse the attackers. When a friend managed to seize control of the bastion host, and began to explore, the "black hole" machine set off alarms and logged packets.

I'm sending these guys cool "Network Police" windbreakers so they can show everyone which side they're on.

The Slaughter of the Innocents

One of the cherished ideologies of computer security in the last few years has been the notion of "full disclosure." In the full-disclosure universe, when one finds a security hole in someone else's software, one gets them to fix it by announcing the bug in a public forum (e.g., bugtraq) and including details of how the bug can be induced or exploited. Sometimes, one posts a tool that exploits the bug – as a "demonstration" – so that there can be no doubt of the bug's seriousness. The logic goes as follows:

- By disclosing bug details, we all learn how to avoid similar bugs in the future.
- By disclosing the bug's existence, we force the vendor to issue a patch.
- By releasing an exploit tool, we force the end user to install the patch.
- The bad guys may already know about the bug so we will tell the good guys.
- You/We need these tools so we can test and secure our systems.

In principle, this seems sensible, but it ignores the third parties (usually average users) who are harmed in the process of all the "improvement" that is taking place. Indeed, it's somewhat reminiscent of the old "we had to destroy the village in order to save it" kind of logic that went out of vogue in the late 1970s. Under the guise of improving security, a lot of people are being made miserable at the hands of armies of script-kiddies[1] armed with the latest hacking tools.

Let's examine the claims of the full disclosure ideologues in detail:

BY DISCLOSING BUG DETAILS WE LEARN HOW TO AVOID SIMILAR BUGS IN THE FUTURE
Unfortunately, most of the security holes that are being exploited today are categories of holes that are already well known. For example, a huge number of today's exploits are buffer-overrun/stack-smashing attacks. The Morris Internet Worm of 1988 exploited buffer overruns; we've known about that category of problem for a very long time, in Internet years. The fact that programmers keep making mistakes of that sort isn't news to anyone, either: anyone who is writing security-critical software should already be aware of array boundaries. There are, periodically, new paradigms of security flaws, but these are discovered relatively rarely.

BY DISCLOSING THE BUG'S EXISTENCE, WE FORCE THE VENDOR TO ISSUE A PATCH
Unfortunately, there are vendors that care about security, and there are vendors that do not. Vendors that care will issue patches in a timely manner, and "do the right thing" about vulnerabilities. Those that do not care will probably continue not to care. The evidence is certainly contradictory on this issue. Microsoft, for example, has pushed out patches quite quickly on some occasions, but meanwhile has allowed Windows' authentication scheme to remain so badly flawed that l0phtcrack consistently makes mincemeat of it. Email attachment execution and scripting have been responsible for several egregious security holes – resulting in band-aid patches but no fixes to address the fundamental problem. Apparently disclosure's effectiveness is mixed in this regard. I suspect that "security through public humiliation" simply annoys everyone: it insults the diligent vendor's efforts and is a flea-bite to the uncaring vendor.

BY RELEASING AN EXPLOIT TOOL, WE FORCE THE END USER TO INSTALL THE PATCH
Unfortunately, this only happens in the rare case of relatively sophisticated users who care about securing their systems. A few years ago, Dan Farmer[2] did a terrifying study that indicated a majority (60+%) of Internet sites were running Web servers with documented security holes – the end users simply had not taken the time to install the necessary patches. More important, when an exploit tool is released, it spreads into use extremely quickly. Even the sophisticated end user will not be able to monitor bugtraq 24 hours a day every day; they are still left vulnerable for an undetermined amount of time. If the proponents of full disclosure really cared to help, they would announce two or three weeks ahead of time to warn users, "on X/Y/Z we are releasing a vulnerability in PDQ that provides remote access – turn off that software immediately." Some disclosures have happened in that manner, but the majority simply appear as an announcement and an exploit tool. Hours or minutes later, innocent people around the Internet are suffering and do not know why. Hint to exploit releasers: they aren't grateful for your assistance, either.

THE BAD GUYS ALREADY KNOW ABOUT THE BUG, SO WE WILL TELL THE GOOD GUYS
Unfortunately, many of the vulnerabilities that are exposed are disclosed by their discoverers. So, perhaps the bad guys already know about the bug, but the number of bad guys knowing it is usually fairly small. Indeed, the good guys have fairly good intelligence networks of their own; usually they find out about a vulnerability fairly quickly once it leaks into the hacker[3] community-at-large. In other words, the vulnerabilities are not being exposed to the good guys much faster than they would in the course of normal events. One thing for certain, however, the vulnerabilities are being shared much more quickly among the hackers, under the guise of "helpfulness."

YOU/WE NEED THESE TOOLS SO WE CAN TEST AND SECURE OUR SYSTEMS
Unfortunately, though few like to admit it, this simply isn't the case. In virtually every case, it is possible to test for the presence of a vulnerability without having to exploit it.

Many of the vulnerabilities that are exposed are disclosed by their discoverers.

In order to make progress, we need to raise accountability for computer-security failures.

For example, rather than executing code that penetrates the system, the tool could simply notify the user that a particular piece of software needs to be upgraded. In many of the tools used by "legitimate security professionals" there are options that have no necessary purpose for *legitimate* security purposes. Take the popular `nmap` utility, for example – a tool that is useful for scanning networks to look for open ports on servers, or to identify and categorize systems – the scanner has considerable functionality that is intended to defeat firewalls and to make the scans harder to detect. Why would a legitimate security professional, acting on his own or a client's behalf, feel the need to hide an authorized scan of a network? The answer is simple: no legitimate security professional needs to act like a hacker.

I've run into security experts who claim they need penetration tools so they can convince their customers to take action. Apparently CERT alerts, vendor-issued patches and release notes, and bugtraq postings from legitimate security practitioners are not enough. I suppose it's possible someone could be so obtuse (many appear to believe the damaging effects of tobacco smoke don't apply to them, either . . .) but this is not a technical problem, it's an exercise in management and interpersonal skills. Of course, it's possible that the security expert has a vested interest in having the customer scared and feeling vulnerable – it helps sell their services.

Blaming the Victim

I've seen a distressing trend toward blaming the victim of a hacking attack: "they should have installed their vendor patches" or "it served them right for being so lame." In a few cases, when I've debated this topic over beers at a conference, I've heard even seasoned professionals display an amazing lack of sympathy for innocent victims. I think what happens is that we security practitioners are so focused on our little niche of the universe that we forget sometimes that "real users" don't want to care about this stuff – they just want to lead their lives and get useful work done on the Internet. Any of you who've been on the receiving end of a hacking attack may remember what an unpleasant violation it is, how helpless, puzzled, and frustrated it makes you feel. Not to mention how much of your time and money it eats up.

The Internet as a whole spent over *half a billion* dollars on computer security last year. This is basically money spent to accomplish no purpose other than to avoid damage. The ILOVEYOU virus and distributed denial of service attacks such as were launched at Amazon.com and eBay.com cost hundreds of millions of dollars in lost time and productivity, to say nothing of aggravation and unhappiness. A lot of the blame for this slaughter of the innocents can be laid at the feet of "legitimate security professionals" who choose to play on both sides of the fence. They want to have the secret thrill of punching weaknesses in innocent users' defenses, but they want to have their hands clean of actually pulling the trigger in person. If you read between the lines of the cases where alleged perpetrators of denial of service attacks were caught, you'll quickly see that *many of them didn't even understand how their tools worked*. They just downloaded them and made complete strangers miserable.

Accountability for All

In order to make progress, we need to raise accountability for computer-security failures. In this case, vendors that produce buggy insecure products must be held accountable if they knowingly release software that places their customers at risk. So it's not just the attack-tool writers and distributors that need to clean up their act. One of the primary intellectual underpinnings of full disclosure is the notion that vendors won't act responsibly unless they are forced – probably an accurate belief in some cases. I

don't think ad hoc force is the way to go about it and neither does the software industry. It's fairly clear that UCITA is an attempt to establish advance protection against liability for vendors producing shoddy software. If UCITA goes forward in its current incarnation then I believe it will badly hurt the cause of the end user and computer security. In moments of idle daydreaming I imagine how fun it would be to ask a judge to issue a restraining order blocking the release of some new version of a major product that was known to contain fundamental security flaws. Of course that's not going to happen, but we've got to ask ourselves what would happen to a car manufacturer that sold a car knowing it tended to catch fire and explode every so often. Eventually, if you put people at risk long enough, it will come back to haunt you. We need to make sure that happens, if we want to see change for the better.

A Plea

I beg you, please, if you find a vulnerability in someone's software, work with them to get it fixed. Don't make other people suffer for your ego by distributing something to show everyone how smart you are. If you want to show people how smart you are, write a better firewall, or a secure browser, or mailer, or something – anything – that has a useful, helpful, and legitimate purpose. Don't write and distribute denial of service tools. Don't write and distribute rootkits and trojan horses. Don't arm, aid, and abet script kiddies. If this situation does not stop, we can expect knee-jerk legislation from our elected leaders – we security practitioners need to clean house ourselves before someone else feels obligated to step in. Don't market yourself by hurting other people; market yourself by helping.

The Bottom Line

The sad reality of the full-disclosure process, as it is practiced today, is that it holds the entire Internet hostage to the views of a vocal minority that have chosen to pursue their egotistic agenda while ignoring the damage it does to innocent users. These innocent users are not technically sophisticated, do not care about security, and simply want to be left alone to lead their lives without unwarranted intrusion from a hacker. They don't upgrade their systems, they don't install firewalls, they simply do not expect that some immature script kiddie is going to amuse himself by molesting them. The immature and unskilled script kiddies are, in fact, being armed by smart but "ethically challenged" individuals who want to have their cake and eat it, too – they want the thrill of being a "black hat" hacker while commanding the salary, stock options, and peer respect accorded to true professionals. What boggles my mind is how they've managed to convince themselves they're doing us all a big favor.

Notes

1. A "script kiddie" is a hacker who doesn't really know anything about security but who knows how to download the latest attack tools from the Net and run them over and over until they find a machine they can break into and ravish. It's a term I first coined back in the early 1990s that has apparently found its way into common usage.

2. See <http://www.fish.com/survey>.

3. People have complained about my politically incorrect usage of the term "hacker" to broadly refer to cybercriminals and vandals. The original meaning of "hacker" has been usurped by the mass media, and, in the interest of being widely comprehended, I avoid arguing about vocabulary. You may feel free to mentally substitute "cracker" or "attacker" or whatever suits your fancy.