Features Editor: Rebecca L. Deuel ■ rdeuel@computer.org

Whose Bug Is It Anyway? The Battle over Handling Software Flaws

Ashton Applewhite

In January 2003, the Slammer virus compromised nearly all vulnerable Microsoft SQL servers in about 30 minutes, slowing or halting Internet systems worldwide. That summer, the Blaster, or Lovesan, virus infected millions of Windows-based PCs; a variant worm suspended online operations at the US Department of State and brought down the Navy Marine Corps Intranet. ATM networks became targets when banks began migrating to Windows-based machines. And, in January 2004, worms started wriggling worldwide through Microsoft's MSN Messenger instant messaging application.

"In 2003, the CERT reported around 250,000 attacks on the Internet, and there are probably lots that are not reported," says Larry Clinton, chief operating officer of Internet Security Alliance, an affiliate of Carnegie Mellon's CERT Coordination Center.

Attacks exploit vulnerabilities in software code. They come in many forms: logic attacks, Trojan horses, worms and viruses, and variants of each. They serve a host of purposes: corporate espionage, white-collar crime, social "hacktivism," terrorism, and notoriety. Greater connectivity, more complex software, and the persistence of older protocols ensure growing vulnerability.

End users lose time and money when networks go down. Software vendors lose face and market share. Security researchers struggle to keep pace with the bugs to keep businesses operating safely.

The only people with no complaints are the hackers, who reverse-engineer patches released by vendors to exploit the holes. It's enough to make you nostalgic for the old days of the Nimba and Code Red viruses, when attacks came six months after vendors released patches. Blaster attacks began three weeks after release. Security experts anticipate so-called "zero day" vulnerabilities, in which attacks precede patches. Although marathon patching sessions have become the norm for harried IT administrators, even top-of-the-line patch management can't keep up with malicious code's growing sophistication.

Industry makes a stand

What happens when a software vulnerability is discovered? It depends on who you ask. To establish agreed-on "best practices" to guide the process, several companies came together in September 2002 to form the Organization for Internet Safety (www.oisafety.org). Members include software companies and security consultants such as Oracle, Microsoft, @stake, BindView, SCO Group, Foundstone, Guardent, Internet Security Systems, SGI, and Symantec.

In August 2003, OIS released guidelines for reporting and managing vulnerabilities. Key recommendations include the following:

- Researchers, or finders, should contact vendors upon discovering a software vulnerability and "participate in identifying the vulnerability to the greatest extent possi-

0740-7459/04/$20.00 © 2004 IEEE

ble," says Scott Blake, vice president of Information Security of BindView and chair of OIS's Communications Committee.

- Vendors should establish a predictable way of being contacted and respond to reports within one week. They should take all reports seriously, work to fix them in a timely fashion, and keep finders apprised of their progress.
- Vendors should have a minimum of 30 days to fix vulnerabilities (a process that can take between one week and several months). During this period, finders shouldn't disclose any information about the vulnerability and should wait an additional 30 days after the patch is issued before releasing any details.
- Vendors should credit the finders but aren't obligated to do so unless the finders follow the complex reporting process detailed in the guidelines.

"The essence can be boiled down to one sentence: The vendor and the researcher should work together in collegial fashion to resolve the issue," Blake says. "Finders want a sense that the vendor is working diligently on the problem, and vendors want to know that they'll have a certain period in which to do reasonable fixing and testing."

A matter of motives

Blake describes the OIS guidelines as "less a matter of trying to improve what was happening than to formalize it."

John Pescatore, research director for Internet Security at Gartner, agrees. "[There's] a kind of informal industry norm that the OIS initiative sort of codifies," Pescatore says. "If you find a vulnerability, you notify the vendor. If the vendor doesn't reply within two weeks, you notify them again and give them another two weeks. After a month, it's OK to go public unless the vendor's said, 'Hey, we're working on it, we need more time.'"

The OIS guidelines do a good job of standardizing informal protocol that the vast majority of researchers are already following, and they make no difference to the renegade 10 percent, says Pescatore.

Not that this informal process always works. Few organizations (large companies in particular) have followed Gartner's suggestion to establish a standard "company-name/security" URL with instructions for reporting bugs. Researchers often don't know whether vendors have received their reports.

These guidelines underwhelmed some independent researchers who expose security vulnerabilities for a living. "The OIS would like you to think there's a de facto way [in which flaws are reported]," says Dave Aitel, a respected ex-hacker and founder of Immunity, an Internet security firm. "But it's typically been the case that whoever finds the vulnerability makes a judgment call on what they want to do with it."

Thor Larholm, senior security researcher at security consulting firm PivX Solutions, points to a set of informal guidelines called RFPolicy, the open source equivalent of the OIS recommendations. "They work well, and they work for any kind of software," he says, but he describes the current state of disclosure policies as "semi-anarchy."

The independent researchers' major complaint is that the OIS guidelines are too pro-vendor. "The thing about the OIS specs is that they are a set of guides written by Windows for Windows. There are literally hundreds of steps you have to go through if you disagree with any of the protocol, and that's too many for most researchers," says Larholm. "They place too much of the burden on the researcher, which is one of the reasons they're not much implemented."

The only people with no complaints are the hackers, who reverse-engineer patches released by vendors to exploit the holes.

Many researchers who report vulnerabilities on a voluntary, noncontractual basis find it ironic to be saddled with an unwieldy set of procedures. Another objection is both economic and philosophical. As Blake acknowledges, "Vulnerability research itself is valuable because it's good QA that someone else is doing."

"If I'm going to do the work for free, I'd like to get the software for free," Aitel counters. "If Microsoft is so interested in disclosure, why not offer a reward for every bug we give them? They could pay up to $20,000, and they'd have the disclosures they wanted."

But why should vendors pay if they can get the work done for free, keeping costs and prices low? Independent researchers build their businesses on their expertise in finding bugs. And they see the guidelines as a threat.

"Microsoft is trying to solve a PR problem by trying to make what I do illegal," Aitel says.

Full disclosure debate

Mutual mistrust doesn't help matters. Vendors resent researchers who go public with bugs without informing them in advance. Researchers say that loopholes in the guidelines make it too easy for vendors to downplay the severity of vulnerabilities and that the guidelines offer too little incentive to come up with the best possible fix—or any fix at all.

The researchers' concern is legitimate, says attorney Jennifer Granick of Stanford University's Center for Internet and Society. "A company's interest is not necessarily in fixing products but in putting out products that their customers think are secure. Disclosing information promotes security but undermines the perception of security,"

Granick says.

The practice of full disclosure, under which vulnerability information is publicly released before vendors have a chance to respond to it, has long been a sticking point between vendors and researchers.

"As soon as you tell one person, you've tipped the balance, and people will begin producing exploits," says Blake of OIS.

Granick sees it another way. "One of the fallacies of limited disclosure is that the vendors think that nobody knows about vulnerabilities, and that is not true. We just don't know who knows," she says.

Consider the difference in opinion over handling exploit code—"proof-of-concept" code a security researcher publishes to demonstrate a flaw. The OIS Code of Conduct expressly prohibits this practice, arguing that releasing the code gives malicious hackers a head start in exploiting the vulnerability. Security researchers, on the other hand, say that often the only way to highlight the seriousness of a flaw—or to get a response from the vendor—is to demonstrate it. Furthermore, the exploit code is often a high-level description of the bug that only highly skilled computer scientists can work with.

"If you don't release the proof-of-concept code, somebody else will," Larholm says. "When a responsible researcher does so, it's exactly that: proof of concept. It's not a hacking tool but a tool used to prove the existence of a vulnerability and to verify whether a system is patched or not."

The best timing for disclosure depends on several factors, including the software's purpose and the nature of the flaw. If the vulnerability will let hackers into bank or government databases, those customers need the information and the patch as soon as possible.

"So a policy that sets a certain time period is always problematic. There's no fix for that," says Granick. "Information can be used for good or evil, and if you want the good side, the information has to be disclosed."

She believes it's better to get the information out there fast. "You want a free flow of information, just like in other scientific fields, ideally without betraying important interests."

The process of public disclosure has forced vendors to react much more quickly than a decade ago, when people reported flaws to an agency like CERT and kept mum until a patch materialized six months later.

Suggestions or regulations?

The OIS emphasizes that their guidelines are just suggestions. "The only mechanism that exists for any kind of enforcement is the court of public opinion," says Blake. "We envision that people will share both good and bad experiences with both vendors and researchers, and that this will affect people's behavior."

Citing energetic lobbying efforts on the part of Microsoft and other vendors, security researchers find this disingenuous. They have little doubt that OIS members would like to see Congress enact laws that make researchers liable if security information goes public.

But in fact, no insiders describe regulation as imminent, or sensible. Describing the Internet as "an entirely different modality than we are used to—inherently interoperable, and not owned by anyone," Clinton of the Internet Security Alliance says FCC-style regulation would be ineffective and potentially dangerous. Others objections are that regulatory systems inherently impede open sources, it would be difficult to agree on policies, and regulatory mechanisms couldn't begin to keep up with technical innovation.

Gartner's Pescatore would like to see the Consumer Product Safety Commission start defining safety levels for consumer software, just as they do for car tires and baby gear. "Saying that software vendors should never bear any liability because software is so complex is like saying a 747 can never be perfect; it's' pretty complex too," he says.

But although a pending class-action suit against Microsoft on the basis of identity theft might well start a trend, the consensus is that judgments would be hard to enforce for a number of reasons. For starters, customers surrender the right to sue via "shrink-wrap" or "click-wrap" licensing agreements, and many strip inconvenient security features out of the software they buy. Prosecuting researchers is possible but problematic under the Digital Millennium Copyright Act, which imposes criminal penalties for subverting copyright intentions. There's anecdotal evidence of flaws going unpublished because of finders' fears of prosecution—a chilling effect with few proponents.

Security researchers say that often the only way to highlight the seriousness of a flaw is to demonstrate it.

Let the market rule

The market is an extremely adaptable and effective regulator. When the Nimba and Code Red viruses attacked Microsoft Web server products in 2001, consumers started looking at Apache, Unix, and Linux-based alternatives. Consequently, Microsoft changed its business practices. The company says that its next desktop operating system, code-named Longhorn and due in mid 2005, will be more secure, and many vendors have instituted more rigorous security testing procedures.

Pescatore advises Gartner clients to ask for proof from vendors that vulnerabilities have been removed and

suggests that they ask, "By the way, what's the warranty?"

Common ground

Both vendors and researchers want the Internet to be a safe place to conduct business. The argument isn't really about what researchers do; it's about the conditions under which they agree to do it and who's calling the shots.

The open source community's RFPolicy clearly places the burden on the "maintainer," who must demonstrate progress to the "originator" (the finder of the flaw) to be given enough time to fix the hole. It's equally clear, however, that the objective is cooperation between all members of the security community: users, researchers, and vendors. While fixing a flaw might take months, entering into a dialogue doesn't, and good-faith updates that keep the researcher in the loop are the cornerstone of a mutually acceptable solution.

All parties have legitimate interests, and establishing a standard that respects those interests makes sense. "What we're missing is some sort of framework to function in that's equally recognized by vendors and researchers—and a process with six or 10 steps, not hundreds," Larholm says. Vendors need set guidelines that will be used and enforced, and researchers need help in reporting vulnerabilities and avoiding liability.

The OIS is listening. It requested public input on the August document, posted and attributed the comments it received, and is incorporating a substantial percentage of them in updated versions of the guidelines.

One obstacle to collaboration is that vendors must contact researchers individually because they have no collective representation. To remedy that situation, Larholm is establishing an international trade organization of security researchers, which has received enthusiastic backing from the security industry and community. "If both sides could work together responsibly and more closely, it would definitely be great, and it would benefit both parties—particularly the vendors," he says.

A Trip to South Africa

Alan Davis

When I was editor in chief of *IEEE Software* in the mid 1990s, I met Andy Bytheway (yes, it's pronounced "by the way"), a British scholar who had immigrated to South Africa around 1994. Every year, Andy tried to convince the rest of the Editorial and Industrial Advisory Board members to have the next annual meeting in Cape Town. Every year we voted it down. Personally, I voted against it because of my strong stand against apartheid, even though I knew it had officially ended. Finally, in 2001, we relented, and made our way to South Africa in June 2002.

My wife Ginny and I were overwhelmed by Cape Town's physical beauty—it's truly spectacular. The city is surrounded on three sides by ocean, and by towering bluffs (rising around 1,000 meters) on the fourth. We spent three weeks in South Africa, including 10 days at Kruger National Park and Swaziland.

As we departed by plane from South Africa, Ginny remarked, "You know, that was a fantastic place. I wouldn't mind coming back for an extended stay sometime." And so the seed was sown.

Planning the trip

As soon as we got home, I started planning a return trip for August–December 2003. In preparation, I

- Applied for a sabbatical from my university
- Wrote proposals to two book publishers
- Asked Andy for information on housing and temporary work opportunities at the University of the Western Cape (UWC), where he was teaching
- Applied for a Fulbright (to pay the costs of travel to Cape Town)
- Started researching chef schools in Cape Town for our son
- Started studying what to do in Southern Africa as tourists

The sabbatical process took six months to get approved. I was glad I started early!

My correspondence with Andy was not as productive as I'd hoped. (As you'll see later, however, the final arrangements were perfect!) Although UWC said it would love to host me, it had no way to compensate me other than with an office and an Internet connection. I learned that UWC was founded during the apartheid era as a nonwhite university and, as such, was given few resources. Although officially the country has abolished apartheid and dissolved most of the systematic policies of oppression, many remnants still exist. For example, by law, the funding per capita is the same at all government universities, but those like UWC use most of the funds they receive to pay their large debt caused by past students' failure to pay their tuition bills. The more I learned about UWC, the more I knew I needed to teach there. I decided to visit UWC for three months and do the work gratis.