Testimony for the Subcommittee on Technology, Information Policy, Intergovernmental Relations and the Census

Hearing on "Worm and Virus Defense: How Can We Protect the Nation's Computers from These Threats?"

Christopher Wysopal

Director of Research and Development

@stake, Inc.

Introduction

Chairman Putnam and members of the Committee, thank you for inviting me to testify today on the subject of protecting the nation's computers from viruses and worms. This is a great honor for me. My company, @stake, consults for the Fortune 1000, primarily financial and telecom companies as well as independent software vendors. We enable them to build more secure software and secure their infrastructures. We also build products that automate the process of finding flaws in software. With these products and with manual methods we provide software security testing, also known as vulnerability research, for our customers. I am a founding member of the Organization for Internet Safety (OIS). OIS is a group of software vendors and security companies joined together with the goal of producing a process for reporting and responding to vulnerability information safely.

The problem of worms and viruses has plagued personal computers since their inception. The source of the problem is twofold: software that is written with time to market concerns and features as more important than safety, and computer users who don't understand that they need to take proper precautions given an increasingly risky computing environment.

A public network such as the Internet is an environment with hostile actors. The software that runs inside of critical infrastructure components such as network routers and servers, as well as desktop applications such as email and Web browsers, needs to be designed in a defensive manner. It must be built with the security quality processes of secure coding and security testing. The computer industry is slowly making progress in this direction, but the economics of software development leads to the reuse of old insecure code even in new products. Computer users are also loath to "upgrade" to new, more secure versions of software due to the cost and the resources necessary to make the change.

The current flawed computing infrastructure is not going to change for the better overnight. It will take many years of hard work. This situation leads to the need to manage newly discovered vulnerability information carefully and to apply secondary lines of defense to protect vulnerable computers until software can be created and deployed that is significantly more secure. There is no "silver bullet" secondary line of

defense. There needs to be a combination of technologies such as automated patching, antivirus products, firewalls and user education.

Vulnerabilities not detected before a software product is released are often found in the field by customers (and their security contractors), independent vulnerability researchers, and the vendors themselves. It is critical that the vulnerability information be handled properly so that a fix can be created and installed by the software user before malicious individuals take advantage of the flaw.

The Organization for Internet Safety has created a vulnerability handling process where flaw finders can easily report issues to vendors; vendors can diagnose and remedy the problem, and then release a fix. The process is a compromise between the need to produce a fix as quickly as possible yet still adequately test that the fix works and does not cause additional problems. It is also a compromise between the needs of users and security professionals to have adequate information available to protect systems, and the need to keep the details required to exploit the vulnerability away from those who would write worms or manual exploit tools.

The Root Cause of the Problem is Software Flaws

Every virus or worm takes advantage of a security flaw in the design or the implementation of a software program, whether that program is the operating system of a personal computer or network router, is running a Web or database server, or is a desktop application such as an email program or word processor. The flaws that are exploited by viruses and worms are not limited to flaws in the security *features* of these programs. The exploited flaw can exist almost anywhere inside a program that processes data directly from a network or from a file delivered by an email attachment.

Practically every software program in this modern age of the Internet falls into the category of requiring security quality processes during its development. If these processes are not in place and followed rigorously by the manufacturer, flaws will inevitably creep into the software. These unknown, latent flaws will then be shipped to the software customer. Many of these flaws will eventually be discovered during the software's lifetime.

When the details of these flaws get into the hands of the malicious individuals who write and distribute viruses and worms, many computer users suffer. The computer users affected are not only the users of the software with the flaws, but other users who share the common resources of the Internet. This is due to the fact that worms and viruses tend to clog up networks and mail servers.

Until recently, perhaps within the last 3 years, building software that was highly resistant to attack was not a top priority of software vendors. Fortunately many are now on the path to educating their software developers to build their products with a secure development process.

Securely built software has security processes added to the design, implementation, and testing phases of the software development lifecycle. Software designs need to be analyzed using threat modeling techniques to assure that the design protects against known threats. Implementation is the phase where the source code is actually written by software developers. These developers need to follow secure coding practices to avoid generating flaws such as buffer overruns. A buffer overrun is precisely the flaw that the Blaster worm exploited. Finally, security testing needs to be performed to catch any errors that the developers made that lead to security flaws.

Most software vendors have sophisticated quality testing processes where all discovered flaws ("bugs") are ranked according to severity and the likelihood a user will be affected by the flaw. The process is designed to eliminate the most serious problems while leaving many minor flaws unfixed due to time to market concerns. Security flaws can and do fit into this process. The important factor is that security flaws can have much more severe impact than a run of the mill software "bug". When security flaws are found they need to be given a high enough priority so they don't go unfixed by the time the software ships to customers.

Patches Are Not a Complete Solution

When a serious flaw is discovered in the field a software fix in the form of a patch is a necessity. If a serious flaw remains for many weeks the data shows that it will eventually be exploited. Research by the HoneyNet Project[1] showed that an unpatched Linux 6.2 system connected to the Internet would be compromised in less than 72 hours. Other operating systems had similar results. Many users think that no one is directing an attack at them so they don't need to bother with security patches. Worms and automated exploit tools don't discriminate. They make every system a target of chance.

Some argue that making patching easier and even automated is the solution. But there are problems with patching which I will outline. The only real long-term solution is to eliminate or at least drastically reduce the number of necessary patches by developing software with a secure development process.

Patches Are Often Not Applied

Automatic patching is a great solution for some computers, but many environments have requirements that don't allow patches to be applied in an automatic or even timely manual manner. Critical computers need to have acceptance testing performed on the new patches before any changes are made. Even when vendors do extensive regression testing they cannot test for all configurations. Patches have been known to cause computers to become unstable. Computer downtime and rebooting often accompany patches. When the patched computer is a critical system this requires planning and computer redundancy. If a patch fails and crashes the system it may take hours to fix.

In industrial and telecom environments, many special purpose computers are treated as appliances even though they have general purpose operating systems running inside them

[1] http://project.honeynet.org/papers/stats/

such as Windows, Solaris, or Linux. The purchaser of this equipment often does not know what software is running inside. @stake has performed audits of telecom and utility companies and found systems such as these that are years out of date with patches.

Internet Patch Distribution

Another problem with patching is *the Internet is the distribution system*. This means that a computer needs to be connected to the Internet to be easily patched. The irony is the Internet is the attack vector that puts the computer at risk. There are two timely examples that illustrate the seriousness of this problem.

The Blaster worm discovered on August 11, 2003 affected the majority of Microsoft Windows computers. It was designed to attack the Microsoft "Windows Update" Web site, which is where computer users were supposed to go to patch their systems. Fortunately the worm writer made mistakes that caused the attack to be easily disabled. Increasingly attacks are becoming more and more sophisticated. If the attack had been successful it would have been nearly impossible for most users to patch their systems.

The Cisco denial of service flaw made public on July 18, 2003 had the potential to cause parts of the Internet to fail if it was exploited. This meant that Cisco had to get the patch out to their "Tier One" customers that run critical portions of the Internet before releasing vulnerability information to all customers. If they hadn't released the patch this way there was the potential that the Internet would cease to function properly and no one could patch. Fortunately, Cisco found this problem internally, which is the best case for flaws that exist in deployed products. They were able to carefully manage the vulnerability information and patch release. If this problem had been first discovered by someone with malicious intent they could have essentially disabled the Internet and downloading a patch from Cisco's Web site would not be possible. Cisco's processes helped mitigate this fragile state of affairs.

Widespread Problems Strain Patching Resources

Most organizations have the people resources to handle patching critical infrastructure such as firewalls and routers or servers because there are a limited number of these computers. When a flaw is found that is so widespread that it affects almost every desktop or laptop, it usually takes many hours or days to patch them all. This is why for some worms you see large sophisticated organizations with computer downtime that can last days.

Reactive Solutions Not Keeping Up

The fast moving Slammer worm, which infected Microsoft SQL Servers last year, was able to compromise nearly all vulnerable systems in about 30 minutes. System administrators didn't know what hit them until it was too late. Reactive solutions such as patching computers after news of a new worm or waiting for antivirus signature updates are not keeping up with the growing sophistication of malicious code.

Preventing the Next Blaster or SoBig

Some simple design changes could have prevented the Blaster worm and the SoBig.F email virus. These changes do make using Windows computers for file sharing or email a tiny bit more difficult, but the result would be eliminating whole classes of worms or viruses. The net result would be making the Internet more reliable and eliminating the need for many users to have computer downtime due to a future Blaster-like worm or SoBig-like virus.

Blaster took advantage of a service that all Windows computers expose to the network by default. This service allows Windows computers to perform file sharing and run programs remotely on each other. The consensus of security professionals is that a service like this should never be exposed to the Internet unless necessary. The default Windows configuration should be that no services are exposed to the network by default. This is how some other current operating systems are configured out of the box. Many security savvy users configure their Windows systems either by using the Windows built-in firewall or another software firewall. By making sure services are not exposed to the network by default the Blaster worm would have been a fraction of the problem it was.

The SoBig email virus takes advantage of the fact that many users still open attachments without understanding what kind of file it is. Email viruses often try to disguise that an attachment is an executable file that will take control of their system when opened. All email programs need to be designed to not allow executable content to be sent or received. It is just too dangerous. Some newer email programs do this. Older Email programs that allow this should be considered unfit for use on the Internet and eliminated. Eradicating executable attachments from the Internet will eliminate most email viruses.

Vulnerability Researchers

Many of the vulnerabilities found in software after it is shipped to customers are not found by the vendor. Some vulnerabilities are stumbled upon by customers. Others are found through directed research by vulnerability researchers. These are individuals who investigate the security of software for academic reasons, profit, or merely curiosity. In all of these categories there are vulnerability researchers that uphold high ethical standards and those that don't.

A primary motivation of vulnerability researchers is altruistic. There currently is no independent or government watchdog group looking out for the safety needs of normal computer users the way the National Highway Traffic Safety Administration looks after the safety of car owners. Given this vacuum, vulnerability researchers feel that someone has to test and find vulnerabilities. They feel that every flaw they find and report to the vendor is another flaw that will be fixed before a malicious person finds and exploits it. In this way vulnerability researchers make all computer users safer.

At @stake, our customers rely on us to find vulnerabilities in the software they are using and report those issues to the appropriate software vendor. When the issue is fixed, all users of that software benefit.

One of the motivations of vulnerability researchers are to make a name for themselves the way an academic does publishing a paper. "Publish or perish" is certainly a way for vulnerability researchers to maintain a name for themselves in the security community and perhaps the larger information technology world. Many individuals are after credibility or fame amongst their peers. The profit motivation is there for vulnerability researchers who sell security products or services. Publishing their research is a way of demonstrating their expertise.

Whatever the motivation of researchers it is important that they handle the fruits of their labors carefully. Vulnerability information in the hands of a software vendor can allow them to fix their product. In the hands of a worm writer, some information has the potential to shut down the Internet. Most vulnerability researchers understand this power and behave ethically, but of course some don't.

There is a group of researchers that use publishing vulnerability information as a way of embarrassing vendors into cleaning up their security processes? . They want the vendor to look bad and to have the vendor's customers harmed. Five to ten years ago there was some legitimacy to this position. At that time most vendors tried to ignore researchers reporting flaws. They would not fix flaws and hoped the researcher would go away. Over time vendors have learned that their customers expect them to fix flaws in a timely way and that in many cases the stability of their customers' computing environments is at risk. Most vendors today are responsive, but mistrust remains.

There are also vulnerability researchers who fall into the malicious hacker category. They do not share their information with the vendor; they share it only with their friends. They write exploit tools that allow them and their friends to break into computers. This is a very dangerous situation because if there is no patch for the flaw the person with the exploit tool can compromise computers with impunity. It is next to impossible to detect and catch these individuals. The only solution to this is to produce software with fewer defects using a secure development process.

Why Vulnerability Researchers Succeed

There are two main reasons why vulnerability researchers succeed in finding flaws that the vendor should have found. Current development processes create an inordinate number of flaws and have limited capabilities for finding them. Much of the software developed today is not built with a secure development process. Security design flaws and insecure coding techniques are endemic in the industry. This is how the flaws get there in the first place. There are signs that many software vendors are improving. Yet even when a vendor switches to a secure development process for new code they still often include old, insecurely developed code in their new products.

Compounding this insecure development problem is inadequate security testing to find the flaws. Security testing is challenging because the testing team cannot be sure they have found all the security flaws within a fixed about of time. The complexity of modern software creates a situation where some flaws are relatively easy to find and some are more difficult and take more time to find. Even if a vendor is able to eliminate all basic flaws the chance remains someone with excess time and energy can still find a security vulnerability.

The industry needs to learn how to design and build software more securely. It also needs to learn from the techniques of vulnerability researchers how to change their quality assurance processes to include security testing.

Vulnerability researchers are essentially performing a security testing function that should have been done as part of the software quality assurance testing process by the vendor. Vulnerability researchers think differently than traditional testers. Testing applications for security flaws takes a true paradigm shift on the part of the tester; they need to begin to think of themselves not as a verifier, but as an attacker. They perform *negative testing*. Negative testing is forcing a program to perform actions on invalid or malicious data in order to reveal what the program could allow an attacker to do.

Positive testing is testing to see if a feature of a software program works. If a program is supposed to save a file to disk when save is selected from the menu, the tester will test to see if that feature works. To contrast negative testing with functional or positive testing imagine a simple financial software application that receives an account number as input and displays an account balance as output. Negative testing inputs invalid data in for the account number to ascertain how the program responds.

The simplest negative testing is to input an invalid account number and check to see that the program returns an error message. This is about the limit of negative testing that is commonly seen in a quality assurance test plan. In order to adequately test the security of an application much more extensive negative testing is required.

The goal of the security tester is to get the program to fault; to get the program to do processing it wasn't designed to do. The security tester designs tests which input data that is particularly problematic for an application to deal with. If the program produces erroneous results, the security tester hones the data in an effort to control the way the program is failing. If the tester is able to exert control over the program or is able to get the program to become unresponsive, it is a vulnerability.

Vulnerability Information Handling

The fact that flaws, or vulnerabilities, exist at all in software shipped to customers is a problem that needs to be solved. But this is a very difficult problem that will not be fixed overnight or even in a few years. The fact that there is a large amount of software already deployed with latent undiscovered flaws mean that we will be dealing with newly

discovered vulnerabilities for the foreseeable future. A process for handling vulnerability information in a timely and safe way is required.

There is some debate in the vulnerability research community as to the best way to handle vulnerability information. However, most agree that it makes sense to inform the vendor of the vulnerable product and to give them time to create a patch. If this wasn't the case there would be much more chaos on the Internet. 4,200 vulnerabilities were tracked by the CERT Coordination center last year. Almost all of them had patches available for public information release due to vulnerability researchers informing vendors.

The area where there is much debate is how much detail should be published about the vulnerability. Detailed information usually allows someone to craft a tool that exploits the flaw. Once an exploit tool is written and released, it allows a much wider audience of unskilled computer users to attack vulnerable systems. On the other hand more details can help security professionals and system administrators defend computers in ways that the vendor may not have envisioned. Many don't like the fact that they need to rely on the vendor to provide the best solution for them because there are times when this is not the case.

The problem of detail is most visible with open source software. Often the patch is distributed as a source code "diff" (for difference) file that is the ultimate detail. It shows exactly which lines of code were vulnerable and how they were changed. This sometimes leads to open source projects slipping out security fixes as part of regular releases. They don't want to notify users of the fix lest malicious individuals get the information too.

Another point of disagreement in the vulnerability research community is how to work with vendors. Many are suspicious of vendors dragging their feet and not wanting to actually fix problems. In the past many vendors did not work well with researchers. This is changing but the sense of mistrust continues.

Organization for Internet Safety Process

Introduction

The Organization for Internet Safety (OIS) was formed by a group of vendors and security companies to come up with best practices of vulnerability information handling. OIS has published a process that flaw finders can use to report flaws to vendors and for vendors to use to respond to these reports. The process, "Guidelines for Security Vulnerability Reporting and Response" (http://www.oisafety.org/reference/process.pdf) was published on July 28, 2003 after a public review period. The security companies and vendors involved intend to adopt the process themselves and promote the process to their peers.

The goal of the process was to protect the computer user community as a whole. There were times when tradeoffs needed to be made between timeliness and reliability or

between helping sophisticated users who could better protect themselves, and the majority of users who are unable to help themselves.

Participants[2]

Although additional participants may be involved in this process, the primary participants are:

- **The Finder**. The security researcher, customer, or other interested person or organization who identifies the vulnerability.
- **The Vendor**. The person, organization, or company that developed the product, or is responsible for maintaining it.
- **Coordinator**. An optional participant that serves as a proxy for the Finder and/or Vendor, assists with technical evaluations, or performs other functions to promote the effectiveness of the security response process.
- **Arbitrator**. An optional participant that adjudicates disputes between the Finder and Vendor.

Phases[3]

The basic steps of the OIS Security Vulnerability Reporting and Response Process are:

1. **Discovery**. The Finder discovers what it considers to be a security vulnerability (the Potential Flaw).
2. **Notification**. The Finder notifies the Vendor and advises it of the Potential Flaw. The Vendor confirms that it has received the notification.
3. **Investigation**. The Vendor investigates the Finder's report in an attempt to verify and validate the Finder's claims, and works collaboratively with the Finder as it does so.
4. **Resolution**. If the Potential Flaw is confirmed, the Vendor develops a remedy (typically a software change or procedure) that reduces or eliminates the vulnerability.
5. **Release**. In a coordinated fashion, the Vendor and the Finder publicly release information about the vulnerability and its remedy.

Timeline

There is no single universal timeframe for which all vulnerabilities can be investigated and remedied. Some flaws can be fixed in one line of source code. Others may require weeks of redesign and coding. Some vendors support only one version of a product that is affected and others may support dozens, compounding the problem of creating patches in a timely matter. In practice vulnerabilities take between a week and several months to remedy. The OIS process suggests 30 days as a starting point.

---

[2] Excerpted from "Guidelines for Security Vulnerability Reporting and Response" (http://www.oisafety.org/reference/process.pdf), published on July 28, 2003

[3] Excerpted from "Guidelines for Security Vulnerability Reporting and Response" (http://www.oisafety.org/reference/process.pdf), published on July 28, 2003

The guidelines prohibit publishing details that could be used to create exploits for the vulnerability until 30 days after the patch is released by the vendor. This is to allow enough time for customers to install the patch, but still allow long term researchers the information needed to better understand how vulnerabilities occur and how they can be prevented.

Conclusion

As a society we are already dependant on computers working properly for many important functions ranging from the financial system to the power grid. The types of software vulnerabilities that lead to worms and viruses such as Blaster and SoBig are well understood. Researchers now know how to build software with significantly less vulnerabilities. Instead of focusing solution efforts on lines of defense deployed by every customer, such as patching solutions and antivirus software, we should focus our nation's limited security expertise on the source of problem: the flaws in software. Software needs to be developed with a secure development process and old insecure software should be eliminated.

Viruses and worms are moving from an annoyance to shutting down government offices and businesses for days. Their impact grows each year. When a technology contains dangerous unseen risks we should have assurances that it is built properly. We need the "electrical code" for building software and we need a way to assure that the code is followed. This will reduce the risk of insecure software at its source and strengthen the computer infrastructure for us all.