The Lifecycle of a Vulnerability

INTERNET | SECURITY | SYSTEMS®

Ahead of the threat.™

INTRODUCTION

Consumers have become quite familiar with security vendors updating their antivirus signatures to keep pace with known exploits. Yet how many of us have ever questioned whether signatures provide the most preemptive protection against all exploits, known and unknown? Are these new signatures addressing the underlying problem or are they really only addressing symptoms of the problem? How are the exploits able to succeed when both antivirus and firewall technology are in place and up to date? This white paper discusses these questions first by examining the definitions, relationship and distinctions between software vulnerabilities and exploits. Next, it explains how vulnerabilities are created, discovered, communicated or exploited and then resolved. Finally, this paper establishes the superiority of vulnerability-driven research over exploit-driven research in providing protection "ahead of the threat."

ANTIVIRUS AND FIREWALL TECHNOLOGY ARE NOT SUFFICIENT

By now, most companies recognize that firewall and antivirus technology are not enough to protect their enterprise. Firewalls reduce the attack surface by closing off access to ports that do not require public exposure. Antivirus programs, on the other hand, block known exploits for which they have signatures. However, if an unknown exploit appears on an open port, firewalls and antivirus technologies aren't much help. In fact, according to the 2004 CSI/FBI Computer Crime and Security Survey, 98 percent of respondents had firewall technology in place and 99 percent had antivirus. However, 78 percent of these same respondents had virus attacks, 39 percent experienced system penetration and 37 percent experienced unauthorized access to information[1]. Clearly more protection than just antivirus and firewall is needed.

According to AV-Test.org, antivirus vendors require between 7 to 30 hours[2] after seeing the exploit to reverse-engineer it, write a signature to stop the attack and send out the update to their customers. However, exploits such as Slammer have been known to propagate worldwide in just 15 minutes. The damage had already been done before an antivirus signature could be constructed.

Before proceeding, it's important to define some key terms to better understand the basics of exploits, worms, Trojans, threat vectors and software vulnerabilities.

EXPLOITS

The term "exploit" refers to a piece of software that is capable of granting or extending privileges on a computer system contrary to that system's design. Exploits find a way to gain unauthorized access to systems that, theoretically, should not allow this to happen.

There are several classes of exploits. A **remote** exploit refers to an exploit that can take advantage of a software vulnerability remotely, over a network. A **local** exploit can only escalate privileges on a system where some kind of local access is already permitted. There's also a **remote authenticated** exploit that takes advantage of a software vulnerability remotely, but can become a local exploit once the "user" has been authenticated.

In order to work, the exploit requires the existence of a software vulnerability. Knowing about the software vulnerability enables the hacker to write exploit code that takes advantage of this vulnerability. A good example of just such a software vulnerability is the Microsoft Remote Procedure Call (RPC) DCOM vulnerability.

1 Source: 2004 CSI/FBI Computer Crime and Security Survey, Computer Security Institute; www.gocsi.com.
2 http://www.avtest.org

Two common types of exploits are worms and standalone exploits.

A worm is a destructive software program containing code capable of gaining access to a computer or network and then causing that computer or network harm by deleting, modifying, distributing, or otherwise manipulating the data. MS Blaster is a network worm that spreads by exploiting the RPC DCOM vulnerability[3] in Windows. This exploit sends massive amounts of packets to the windowsupdate.com Web site in a distributed denial of service (DDoS) attack[4], overwhelming it so that the Web site response is either slowed or stopped altogether.

A Trojan (a type of standalone exploit), or Trojan horse, is a malicious program disguised as legitimate software. Trojans are known to create a backdoor on a computer that grants malicious users access to the system, possibly allowing confidential or personal information to be compromised. Klogger is a password-stealing Trojan which allows a hacker to monitor a user's activities on an infected computer. This type of Trojan drops a keylogging component, which starts keylogging when a user is asked to input a login and password. Then it stores the recorded keystrokes data for later submission or sends this data to a hacker immediately[5]. The hacker can then sign onto the same site as if he were the "owner" of the ID and password.

In addition to different types of exploits, there are different directions, or vectors, the exploits use to attack. The two major threat vectors are the network and the application vectors.

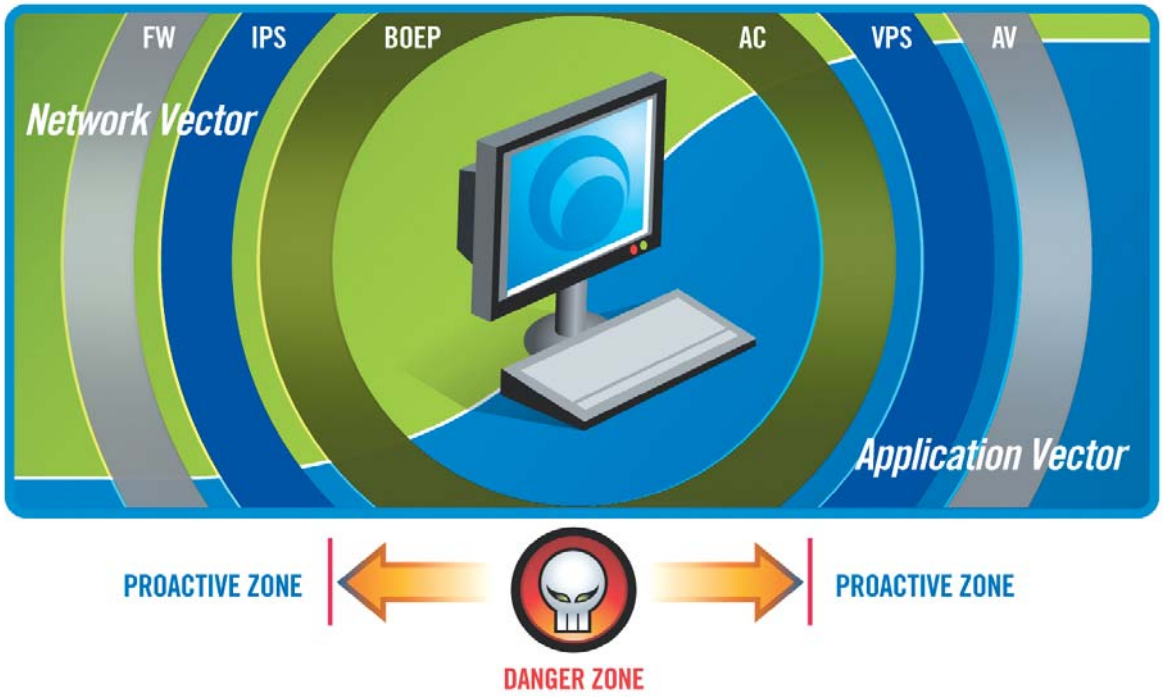


Figure 1

Network Threat Vector - Network-based attacks utilize malicious network traffic to remotely compromise their target systems. Unlike other threats, a network-based attack can penetrate the target system, launch the attack's malicious payload and propagate or replicate itself without human intervention. Network-based attacks on the host predominantly exploit vulnerabilities in protocols and network-aware processes. These vulnerabilities are typically the result of programming errors that provide opportunities for a buffer overflow. A worm such as MS Blaster is a good example of a network-based attack[6].

3 Refer to Microsoft Security Bulletin MS03-026 for more information.

4 http://www.f-secure.com/v-descs/msblast.shtml.

5 http://www.f-secure.com/v-descs/pswsteal.shtml.

6 Defining the Rules for Preemptive Host Protection: Internet Security Systems' Multi-Layered Strategy; http://www.iss.net/support/documentation/whitepapers.

Executable File-based Threat (Application Threat Vector) - Application-based attacks utilize executable files to attack and compromise target systems. Unlike network-based attacks, executable files typically require some form of user involvement to launch an attack. The MyDoom worm variants are sent to users via e-mail, and contain a semi-randomly named executable attachment. Upon user execution of the attachment, the worm launches, installs itself onto the system and begins propagating via e-mail to other e-mail addresses found on the compromised host. Some variants of the MyDoom worm also launch DDoS attacks against specific targets[7].

In summary, the exploit, whether it be a worm or a standalone exploit, can gain unauthorized access to a system either by exploiting a software vulnerability via the network vector (which requires no user interaction) or by inducing the user to launch the malicious executable file via the application vector.

This white paper focuses on exploits that gain access via the network vector and therefore target software vulnerabilities. One might ask, "If the exploits require a software vulnerability, why not eliminate or shield the vulnerability? Won't that stop the exploit as well?" This suggests that the real problem is not the exploit but the software vulnerability. If a particular security company views the exploit as the root problem, then it will develop a solution to address the exploit, giving birth to the phrase "exploit-driven research." However, if a security company thinks that the software vulnerability is the root problem, it will develop a solution to address the software vulnerability, leading to "vulnerability-driven research." By addressing the vulnerability, it would seem logical that the result would be higher-quality protection.

To clarify this point, let's take another look at the Microsoft Remote Procedure Call software vulnerability and the MS Blaster exploit. There are 13 different publicly known ways to exploit this vulnerability. The MS Blaster exploit is only one of these ways. With exploit-driven research, building a signature against MS Blaster protects the target system against MS Blaster exploits, but the target system is still vulnerable to the other 12 ways of exploitation.

- Is the RPC vulnerability protected against all attacks? No.
- Is the RPC vulnerability protected against one of thirteen different types of attacks? Yes.
- Is the RPC vulnerability still vulnerable to 12 other types of attacks? Yes.

In this case, exploit-driven research provided protection against one exploit (MS Blaster) but did not fully address the software vulnerability. The Agobot worm also attacks the RPC vulnerability, but not in the same way as MS Blaster. Exploit protection against MS Blaster would allow infection from Agobot, because the exploit protection against MS Blaster only specifically thwarts MS Blaster, and no other exploits. However, if one could shield the vulnerability so that all 13 ways of exploitation fail, wouldn't this be preferable and a far superior level of protection? Superior protection must be backed by vulnerability-driven research and not exploit-driven research.

SOFTWARE VULNERABILITIES

A software vulnerability is an issue in a software product that enables a malicious individual to undermine the product's security and potentially cause harm. It's easiest to think of a software vulnerability as an open door. This open door is a point of entry through which anything and everything could enter. There are three different ways to provide protection for this open door: closing the door, confronting anyone or anything that attempts to enter through this door or shielding the door from all potential entrants.

Closing the Door - The only way to truly "close the door" or eliminate the software vulnerability is for the software vendor (that introduced the vulnerability to begin with) to issue an effective patch which addresses the vulnerability. However, even when this is done, end users of this software (in this case, corporations) may be so swamped with patch management and testing that this patch isn't applied until a year after the patch was originally issued. For example, the Nimda exploit first appeared and spread worldwide in 30 minutes on September 18, 2001. This caused a great deal of damage, even though a software patch which directly addressed the underlying software vulnerability had been issued on October 17, 2000, a full 336 days earlier. Because applying these patches is so time-consuming, other options become more effective.

7 Defining the Rules for Preemptive Host Protection: Internet Security Systems' Multi-Layered Strategy; http://www.iss.net/support/documentation/whitepapers.

Reacting to the Exploit - The second option is to use exploit-based research to identify the exploit and write a signature to counter the exploit. It's important to realize two things about this approach: (1) this approach will only work on an exploit-by-exploit basis and (2) it is only effective from the time the signature is first added to the antivirus signature database until the patch (to "close the door") can be applied. This approach does not provide protection for the full window of vulnerability, from the time the vulnerability was first discovered until the patch is applied. Though a signature exists to counter a specific exploit, this does not mean that the vulnerability is fully protected.

To clarify this, let's revisit the open door analogy. If a mosquito enters through the open door, the mosquito is now in the house and was initially successful getting in. Since the mosquito has appeared, the exploit-driven research vendor now attempts to solve this problem (which, in its mind, is the mosquito). Perhaps the vendor might react by installing a mosquito fogger at the door. From this point forward, the vendor has addressed the mosquito problem and need not worry about further mosquito attacks through this open door.

- But is the door still vulnerable? Yes.
- Are those inside still vulnerable? Yes.

Let's say it rains next and some of the rain comes in through the door. Again, this new "exploit" was initially successful. Since the rain has appeared, the exploit-driven research vendor attempts to solve this problem (which is now the rain). Perhaps the vendor might react by placing some sort of shelter over the door to prevent further raindrops from entering. From this point forward, the vendor has addressed the rain and need not worry about more rain getting in.

- But is the door still vulnerable? Yes.
- Are those inside still vulnerable? Yes.

Having thwarted both mosquitoes and raindrops, is there now full protection for the "open door" vulnerability? Hardly. There is only exploit protection against mosquitoes and raindrops. But what about thieves? If a thief now comes in by the very same open door through which the mosquito and raindrop entered, he is not stopped and has access to whatever lies within. The exploit-driven security vendor may react by posting a security guard at the door to prevent further thieves from entering. But hasn't the damage already been done? What good is the guard posted at the door if the thief has already stolen personal and confidential data?

In this analogy, reactive, exploit-driven security resulted in three successful attacks. It was only after the fact that further exploits of these three types would be thwarted. Even so, after each exploit was dealt with, the door remained open and vulnerable. Reactive security fights off each individual attack on a case-by-case basis (one exploit is countered by one signature, one signature thwarts one exploit) after each initial, different type of attack has already succeeded. However, nothing was done to address the underlying vulnerability.

Shielding the Vulnerability - The third option is to shield the vulnerability so that the vulnerability is directly addressed, not the exploit. Knowing that the vulnerability exists, one does not need to experience an exploit before providing protection. The truly preemptive security vendor will proactively shield the "open door," or software vulnerability, long before the exploits are written. The vendor can thwart all attacks that attempt to exploit that vulnerability without knowing anything about each specific attack. This replaces the reactive nature of exploit-driven research with a proactive approach: vulnerability-driven research. When shielding the vulnerability, there's no need to react to the exploits, because exploits are not viewed as the true problem. They are simply a symptom of a greater problem. Eliminate the greater problem and the symptoms go away as well.

Back to the analogy: to protect the open door from attacks, covering the opening with a temporary shield or force field effectively protects those inside from all intruders/attacks from the outside. It seems clear that attacks have a greater chance (if not a certainty) of succeeding with exploit-driven research than with vulnerability-driven research. With vulnerability-driven research, the vendor gets a jump on the exploits by providing protection "ahead of the threat."

How can the vulnerability-driven security vendor provide such comprehensive protection for the vulnerability? It really comes down to the quality of the signatures. Pattern-match signatures counter attacks on a one-for-one basis, while algorithm signatures counter attacks at a much higher ratio. Multiple, different attacks can be detected and thwarted by one algorithm alone because that one algorithm is specifically designed to address the software vulnerability rather than individual exploits.

Let's discuss how vulnerabilities are created, discovered, communicated or exploited, and then resolved. Vulnerabilities are typically the result of programming errors that provide opportunities for a buffer overflow. Since buffer overflows are the most predominant and potentially dangerous software vulnerabilities, this white paper will limit its focus to this particular vulnerability.

> *Buffer overflow vulnerabilities exist in every type of software, from the core of the Windows operating system to large enterprise software applications like SAP, Exchange or even applications that run on every single desktop. Buffer overflow vulnerabilities were around as far back as 10 to 15 years ago. Other attacks are becoming more popular, but the real hard-hitting vulnerabilities that are exploited by worms and by hackers on a day-to-day basis are usually buffer overflows. These hard-hitting worms have the ability to "take down" corporations.*
>
> - Neel Mehta, ISS X-Force Research

LIFECYCLE OF A VULNERABILITY

Vulnerability Creation and Progression - Software vulnerabilities can be exploited and cause damage, but how do they get into the software to begin with? Buffer overflow vulnerabilities are the result of a coding mistake involving the mismanagement of memory. For performance reasons, "unmanaged" code is often used to write the operating system or the services that run on computers. The term "unmanaged" refers to the fact that the programmer must perform the application's memory management from within the code. In other words, the programmer is responsible for allocating and deleting memory as it is used. Though there are performance advantages to writing code using unmanaged programming languages, there are also disadvantages.

> *[Writing unmanaged code] gives the programmer the ability to do a lot of things, including making very big mistakes that could allow the software to do something unexpected. With buffer overflow vulnerabilities, the memory region allocated for a specific variable is too small to then hold the data that's written to it, which leads to corruption of memory. Depending on where that memory corruption occurs, it can be very easy to take control of the system and cause it to do something that an attacker might want it to do. This generally leads to complete compromise of the system.*
>
> - Neel Mehta, X-Force Research

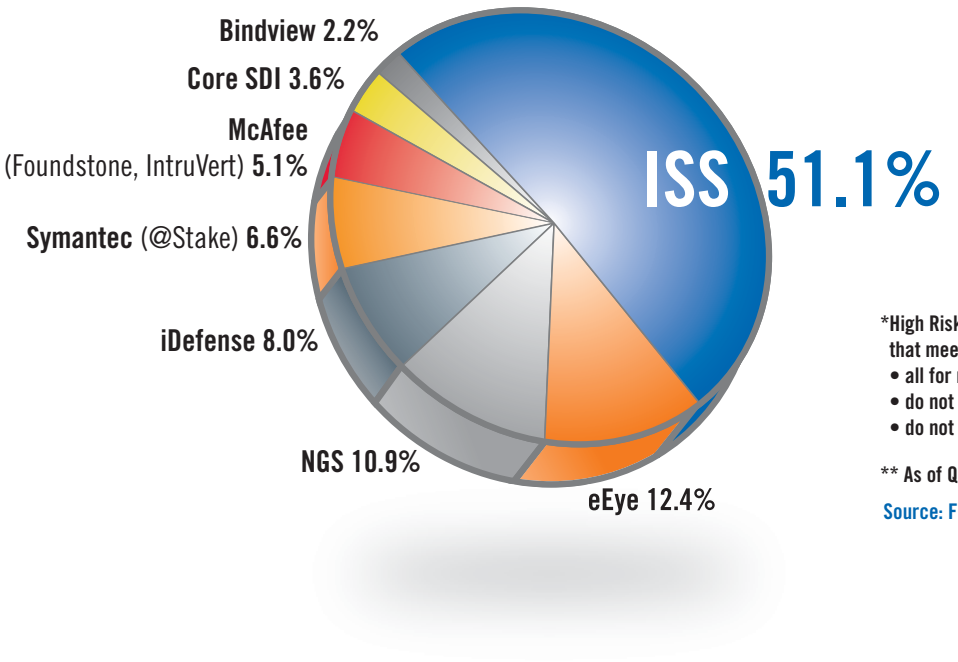
If this software vulnerability remains undetected in the code throughout the development and testing of code, chances are, it'll make it into generally available code that is then sold to the public. Once in the public domain, there are various groups of people looking for these software vulnerabilities: research groups (the "good guys") and hackers (the "bad guys").

FINDING THE SOFTWARE VULNERABILITIES

Hackers (The Bad Guys) - These are the people trying to break into systems for various reasons ranging from bragging rights to financial motivation. Early on, the primary reason for hacking was simply to say that they could do it and to demonstrate their skill. Now, however, there are a growing number of cases where credit cards have been stolen via computer break-ins. A recent trend involves organized crime providing financial incentives for hackers to obtain credit card numbers, and profiting from these break-ins.

Research Groups (The Good Guys) - These are groups of people or research organizations that look for software vulnerabilities. The purpose of such research is to find the vulnerability before the hackers do in order to provide their customers the most preemptive protection possible. When they find a software vulnerability, these research groups work with the software vendor that created the vulnerability to develop a patch solution. These research groups are well positioned to provide proactive protection as well, giving them a "leg up" on the competition. One such research group is the X-Force research team at Internet Security Systems (ISS). Historically, ISS' X-Force has identified more high risk vulnerabilities than all other companies combined.

X-Force® High Risk Advisories* 1998-2005**



*High Risk Advisories are comprised of vulnerabilities that meet the following combination of conditions:
- all for remote compromises (compromise over a network), not local
- do not require user interaction to exploit
- do not require authentication to exploit

** As of Q1 2005

Source: Frost & Sullivan, April 2005

Figure 2

Microsoft has acknowledged and thanked ISS' X-Force on numerous occasions for discovering these vulnerabilities. One such Microsoft bulletin, MS05-025, issued on June 14, 2005 thanked Mark Dowd of ISS X-Force for reporting the PNG Image Rendering Memory Corruption Vulnerability — (CAN-2005-1211).

Software vulnerabilities can be found most quickly when the source code is available. Reading the source code enables the researcher to understand what the application is doing. When the researcher understands the application behavior, even the subtlest vulnerabilities can be discovered.

Another method of vulnerability research involves stress testing or fuzz testing. This occurs when the application is bombarded with white noise, or malformed data, in every way until it crashes. It's kind of a black box approach. If you don't really know what's going on "under the hood," just try to hit the application with everything.

> *Going back 5 or 6 years, the vulnerabilities found were fairly simple. They were stack-based overflows that are the result of very simple mistakes that aren't commonplace in code anymore. Now, the vulnerabilities are becoming much harder to find. This is not to say that there aren't hackers out there with a lot of time on their hands that are doing it full-time. Exploiting vulnerabilities is not as challenging as finding them nowadays, especially in major software such as the core of the Windows operating system. A lot of the easy-to-find vulnerabilities have been eliminated and the ones that are left are a lot more subtle, so it's hard for Microsoft to find them. It's hard for us to find them. It's hard for hackers to find them. It really comes down to who has the most time and the most skill.*
>
> - Neel Mehta, X-Force Research

ISS' X-Force Develops an Internal Exploit and Analyzes Potential Attack Vectors - Once the X Force researchers find a vulnerability, they develop an internal exploit to verify that the vulnerability can be exploited. They then make further attempts to exploit the vulnerability from many different "attack vectors." Attack vectors are different ways of introducing the exploit to the vulnerability, such as through Web page retrieval, opening an attachment, etc. Researchers perform all of this testing in a "safe" environment and take measures to ensure that exploits remain internal and are not released into the wild. This internal exploit is never released to anyone, not even to the software vendor whose vulnerability was discovered. Enough information is provided to the software vendor so that it can produce an effective patch against the vulnerability.

ISS' X-Force Communicates with the Software Vendor - Once the vulnerability has been identified, the "discovering company" contacts the software vendor in a responsible and confidential manner. This allows software vendors and security professionals to resolve the security issue while simultaneously minimizing (if not eliminating) the chance that the information could fall into the wrong hands. More specifically, this type of communication accomplishes three things:

1. It prevents public release of this vulnerability information. If the vulnerability information were made public prior to patch availability, this would tip off hackers to the vulnerability.
2. It allows the software vendor to develop a software patch to properly address this vulnerability. In addition to publicly announcing the vulnerability, the software vendor must also provide a patch to address this vulnerability.
3. It allows the discovering security company to ensure that its customers are protected against the new vulnerability with either existing protection or newly issued protection. In the case of ISS' X-Force, it's not uncommon for ISS products to already be protecting against new-found vulnerabilities. An existing algorithm or piece of logic within its protection engine may already detect traffic targeting the new-found vulnerability. One example of this is protocol anomaly signatures, which catch traffic patterns that should never happen in legitimate traffic. If an unknown exploit appears and is not considered legitimate traffic, these protocol anomaly signatures will catch the exploit.

When the software vendor issues its press release concerning these vulnerabilities, it has patches ready for its customers, and the discovering company is also ready with updates as appropriate. Since the discovering company and the software vendor are the only two parties aware of this vulnerability, it's important to note here that the discovering company has the advantage of providing the most proactive protection for its customer base…truly "ahead of the threat" protection.

Hackers Develop Exploits - There's always the chance that the good guys don't find the software vulnerability first. But even if they do and have not yet publicly announced the vulnerability, software remains unpatched and vulnerable. Moreover, as noted earlier, many companies do not apply patches for up to a year after the patch is released, so they also remain vulnerable. Therefore, any time between when the new code was released to the public and when the customers apply the patch is a vulnerable window of time. The hacker can craft an exploit gaining unauthorized access to many systems and causing damage worldwide in a matter of minutes, much as Nimda did in 2001 and SQL Slammer did in 2003.

How do these exploits work and how easy are they to build? To explain how an exploit works, let's look at something called the "stack," a memory store on the computer that contains key pieces of information. One of those key pieces is the return address for a function. When a function makes a call, it has to remember where to go back to when it returns. This return address is stored on the stack, so if the stack is corrupted, the return address may also be corrupted and may make the function return to a different place in memory, a different part of code. If this different place in memory contains the code a hacker has supplied, then the hacker is basically running whatever code he wants. That's the simplest case of buffer overflow exploitation.

With other memory regions such as the heap, or static data, exploiting becomes a lot easier as attackers understand how to exploit these vulnerabilities better. These vulnerabilities come down to the same thing: there's operating system data stored in band with user data. When the user data is corrupted or overflow occurs out of the user data, operating system data is encountered which is then interpreted in a certain way by the operating system. Using an exploit, an attacker can cause specific memory corruption that leads to his gaining control of the system.

To understand buffer overflows, one must see that corruption of user data leads to corruption of operating system data. Though it may sound complicated, it's been done for about 15 years now. In fact, at this point, exploitation of buffer overflows in every memory region of every type is well documented. Unfortunately, if someone didn't know how to do [a buffer overflow exploit], they could go to the Web right now, read a paper and probably be doing it by the end of the day. It's really that simple. It's been talked about so much. There are so many examples of how to do it that anyone can just pick up and modify. There isn't a big learning curve anymore.

- Neel Mehta, X-Force Research

Unfortunately, it's usually only when an unknown exploit appears on the Internet that most people realize which type of research their security vendor conducts: exploit-driven or vulnerability-driven. Your enterprise is currently being protected by a security vendor that either reacts to the exploit or shields the vulnerability, providing protection "ahead of the threat." Which level of protection would you prefer?

INTERNET SECURITY SYSTEMS' VIRTUAL PATCH™ TECHNOLOGY

Security solutions powered by proactive research provide a viable alternative to the current patching crisis - offering what ISS calls virtual patch technology. As software vulnerabilities continue to increase, so will the number of patches to install. Today, many organizations remain in an ongoing "triage" mode trying to determine which critical patches to apply first. If a security solution has the benefit of proactive security content updates focused on vulnerabilities, the system will protect those vulnerabilities during the window of exposure between vulnerability announcement and patch application — hence the term, virtual patch.

For example: ISS discovered the SSL PCT1 vulnerability, which was publicly disclosed on April 13, 2004 in Microsoft Security Bulletin MS04-011. Before Microsoft issued a patch and before the vulnerability was announced publicly, ISS silently incorporated protection for the vulnerability into its protection products in September 2003. ISS' focus on primary, vulnerability-based research resulted in products that provided a virtual patch, or buffer of protection, before Microsoft officially announced the vulnerability and issued a patch. This level of protection is extremely difficult to provide without a dedicated group of in-house security experts.

ISS' X-Force security research team conducts original, primary research on vulnerabilities and threats, which is applied to ISS' protection engine in the form of security content updates. As noted above, the X Force is credited with discovering and mitigating more major software vulnerabilities since 1998 than all other commercial security research organizations combined.

THE PROVENTIA® FAMILY OF SECURITY PRODUCTS

All of the products in ISS' Proventia family are driven by a common engine, greatly simplifying deployment and management. This one product family delivers effective security at all levels of the enterprise: network, server and desktop.

Proventia Intrusion Prevention Appliance - Preemptively blocks known and unknown threats, preserving network uptime, reducing emergency patching and preventing damaging security breaches. These appliances sit inline on the network or gateway and are available in a range of operating speeds.

Proventia Integrated Security Appliance - Industrial-strength intrusion prevention technology combined with firewall, antivirus, Web filtering and antispam technology in a convenient, easy-to-manage format. These appliances provide comprehensive security previously unavailable for remote offices and medium-sized businesses.

Proventia Web Filter - Accurately blocks unwanted Web content from more than 20 million catalogued Web sites, more than its nearest competitors, helping businesses improve productivity, enforce Internet usage policies and reduce legal liability.

Proventia Mail Filter - Analyzes inbound and outbound e-mail to improve productivity and free up network resources. Its 10-step analysis process accurately detects spam and allows legitimate e-mail through.

Proventia Desktop - Combines a personal firewall, intrusion detection, intrusion prevention and system compliance enforcement in a single agent, ensuring that desktops are protected and adhere to corporate standards. It identifies and blocks known and unknown threats without user intervention for better protection at a lower cost.

Proventia Server - Provides real-time intrusion detection and prevention by analyzing events, host logs and inbound and outbound network activity to block malicious attacks from damaging critical assets.

Internet Scanner® Vulnerability Assessment Application - Conducts automated vulnerability detection and analysis of networked systems including servers, desktops and infrastructure devices. This helps organizations proactively protect their information assets by discovering vulnerabilities that expose them to compromise.

SiteProtector™ Centralized Management System - Provides scalable, centralized management for the Proventia product family, significantly reducing demands on staff and other operational resources. SiteProtector's unified command, control and monitoring system scales easily to support deployments of any size.

INTERNET SECURITY SYSTEMS' SERVICES OFFERINGS

Support Center Practices (SCP) Certified Customer Support Organization - SCP Certification ensures that a support organization maintains the very highest levels of service and support through continuous improvement. All efforts are guided by the customer's feedback and satisfaction with support. Currently, Internet Security Systems is the only enterprise security vendor to be SCP certified at all levels of support.

Professional Security Services - Delivers comprehensive, enterprise-wide security assessment, design and deployment services to help build effective network security solutions. Experts demonstrate how to implement network security best practices to reduce online threats to critical business assets.

Managed Security Services - Complements the Proventia product family and actually guarantees protection for business operations. The industry's most comprehensive solution for 24/7 security management, ISS' Managed Security Services constantly monitors attacks on the Internet and customer networks from six Security Operations Centers around the world.

ABOUT INTERNET SECURITY SYSTEMS, INC.

Internet Security Systems is the trusted expert to global enterprises and world governments, providing products and services that protect against Internet threats. An established world leader in security since 1994, ISS delivers proven cost efficiencies and reduces regulatory and business risk across the enterprise. ISS products and services are based on the proactive security intelligence conducted by ISS' X-Force research and development team - the unequivocal world authority in vulnerability and threat research. With headquarters in Atlanta, Internet Security Systems has additional operations throughout the Americas, Asia, Australia, Europe and the Middle East. For more information, visit the Internet Security Systems Web site at www.iss.net or call 800-776-2362.

GLOBAL HEADQUARTERS

6303 Barfield Road
Atlanta, GA 30328
United States
Phone: (404) 236-2600
e-mail: sales@iss.net

REGIONAL HEADQUARTERS

Australia and New Zealand
Internet Security Systems Pty Ltd.
Level 6, 15 Astor Terrace
Spring Hill Queensland 4000
Australia
Phone: +61 (0)7 3838 1555
Fax: +61 (0)7 3832 4756
e-mail: aus-info@iss.net

Asia Pacific
Internet Security Systems K. K.
JR Tokyu Meguro Bldg. 3-1-1
Kami-Osaki, Shinagawa-ku
Tokyo 141-0021
Japan
Phone: +81 (3) 5740-4050
Fax: +81 (3) 5487-0711
e-mail: jp-sales@iss.net

Europe, Middle East and Africa
Ringlaan 39 bus 5
1853 Strombeek-Bever
Belgium
Phone: +32 (2) 479 67 97
Fax: +32 (2) 479 75 18
e-mail: isseur@iss.net

Latin America
6303 Barfield Road
Atlanta, GA 30328
United States
Phone: (404) 236-2709
Fax: (509) 756-5406
e-mail: isslatam@iss.net

Distribution: General
PM-LCVWP-0805

INTERNET | SECURITY | SYSTEMS®

Ahead of the threat.™