White Paper | October 2005

McAfee®
Proven Security™

A Brief History of Malware

An Educational Note for Service Providers

Table of Contents

An Educational Note for Service Providers

Viruses and the Rise of the Internet

In 1969, there were four hosts on the Internet. In 2005, that number has exceeded 300 million.

It is not surprising that the evolution of computer viruses is directly related to the success and evolution of the Internet, and the comparison between the Internet and a living body that is continuously fighting viral infection and disease is both easy to understand and picture. As the Internet has assumed a life of its own, connecting computers, servers, laptops, and mobile phones around the world into a single, evolving web of interconnectivity, so, too, has malicious code quickly evolved and mutated to become a myriad of increasingly more complex malicious software programs.

Simply put, anti-virus is the antidote to this infection.

As the Internet has evolved, so has the nature of the threat. Viruses have spawned new forms of malicious life that thrive upon the computational technology of Internet connectivity, data, and voice communications. These new threats can rapidly recreate themselves (worms) to attack their hosts, and then spread rapidly from one host to another. Recently, independent threats have combined in the form of *blended threats* that conspire to identify, disable, or destroy any vulnerable carrier hosts.

So where did this all start?

Brain (1986) was one of the earliest viruses. It infected the boot sector of floppy disks, which were the principal method of transmitting files of data from one computer to another. This virus was written in machine code, the basic computing language for personal computers (PCs). Virus propagation was slow and depended upon users physically carrying the infection from one machine to another, and then transmitting the infection via the floppy disk when the PC booted up. These viruses became know as boot sector viruses because the upload executed the virus process. By the early 1990s, well-known viruses like Stoned, Jerusalem, and Cascade began to circulate.

The first major mutation of viruses took place in July 1995. This was when the first macro virus was developed. It was notably different from boot sector viruses because it was written in a readable format. The use of such macro programming within common office applications resulted in the Concept virus. Viruses written in readable format, combined with the existence of macro programming manuals and the enhanced capabilities of macro viruses relative to boot sector and contemporary file viruses, allowed new macro viruses and variants of existing viruses to be rapidly developed and distributed. Furthermore, with computers now being connected to local area networks (LANs) that were slowly being interconnected to each other, the increased importance and feasibility of file sharing provided an efficient distribution mechanism for viruses, which further attracted more writers to this new breed of malicious code.

The next major mutation of viruses took place in 1999 when a macro-virus author turned his attention to the use of e-mail as a distribution mechanism. Melissa, the first infamous global virus, was born. After Melissa, viruses were no longer solely reliant on file sharing by floppy disk, network shared files, or e-mail attachments. Viruses had the capability to propagate through e-mail clients such as Outlook and Outlook Express. As of a result of this and new developments in the capabilities of the Windows® Scripting Host, a devastating virus known as Love Letter was spawned on May 4, 2000. The world has never been the same since.

Evolving, mutating, and growing in intelligence and its ability to survive and spread its infection, the virus has jumped from the humble floppy disk to distributing itself quickly around the internal network. The virus is presently capable of spreading seemingly unseen, effortlessly and unstoppably across the global Internet, infecting anything and everything it touches.

As antidotes to viruses were developed and immunization programs created and deployed to counteract their effect, some viruses were able to adapt and learn to circumnavigate the efforts made to stop them, and new malicious organisms rapidly came into existence. Today we not only have to cope with viruses, but also with worms, Trojan horses, backdoors, rootkits, HTTP exploits, privilege escalation exploits, and buffer overflow exploits. These new threats identify and prey upon vulnerabilities in applications and software programs to transmit and spread attacks.

In 2002, these threats began to combine, and the blended threat was born. By utilizing multiple techniques, blended threats can spread far quicker than conventional threats.

And the devastation they can wreak can be far more widespread and destructive.

The New Malicious Bacteria

Today, PC users and network operators have to fight off and immunize themselves against an ever increasing variety of methods of attack which can infect randomly or target specific networks or machines in a coordinated attack.

- Worms
- Trojan horses and potentially unwanted programs/ spyware/bots
- Buffer overflow exploits
- Privilege escalation exploits
- Backdoors
- Rootkits
- HTTP exploits

Worms

Worms are malicious programs that spread themselves automatically. Viruses are malicious programs that spread by human intervention such as inserting a floppy disk into a computer or double-clicking on an e-mail attachment. Most viruses are spread by convincing the system user to open a malicious attachment. Worms are able to propagate autonomously. Worms spread by exploiting vulnerabilities in a computer system, then using network connectivity to find and attack other vulnerable systems. The lack of user intervention allows worms to spread far faster than viruses.

The term *worm* was first applied to self-replicating computer programs by John Brunner in the 1975 sci-fi novel, *The Shockwave Rider*. In the book, a malicious program spreads itself throughout the government's massive computer system, exposing private data and government secrets. Researchers at Xerox's famous PARC laboratory first applied the term to real-world, self-spreading programs.

Worms are equally if not more damaging than viruses. Recent worms such as Code Red and Nimda have caused billions of dollars of damage,[1] clean-up costs, and loss of business revenue. Attackers are using worms more frequently, since they can do so much damage so quickly.

The first widespread Internet worm appeared in 1988. A graduate student at Cornell University, Robert Morris, created a worm program that exploited several vulnerabilities and released it to the then-growing Internet. Although Morris claimed that there was no malicious intent behind the worm, and that it was just an "experiment" that went terribly wrong, the Morris worm damaged over 6,000 Internet-connected computers and caused hundreds of thousands of dollars in clean-up costs. As an example of the speed with which worm technology has evolved, consider that Code Red spread over 300,000 computers in just 14 hours.

The anatomy of a worm

Worms have three main parts:

- Attack mechanism
- Payload
- New target selection

Worms exploit one or more specific vulnerabilities in a computer system.[2] A worm's attack mechanism exploits the vulnerability in the target system and uses that vulnerability to copy itself onto the target system.

The payload is the part of the worm code that performs malicious actions against the compromised host. Some worms have no payload; they simply spread themselves and drain system resources. Similarly, a worm can be any type of program. If a worm targets a vulnerability that allows the worm to run its payload at the root or administrator privilege level, the payload will be able to reformat the hard drive, or install rootkits and backdoor programs. The payload may search the computer for data to send to a central server where the worm's author can collect it. Any number of other malicious activities can be part of a worm payload.

Once the worm code is executing on the attacked system, it attempts to spread again. To do this, a worm must locate target computers which are vulnerable to its attack mechanism. The mechanisms used vary in sophistication.

The dissection of a worm attack

Code Red exploited vulnerability in the Microsoft® IIS Web server. It had three different versions, each one improving the distribution mechanism, which resulted in dramatic acceleration and spreading. The first version of the Code Red worm simply sent Hypertext Transfer Protocol (HTTP—the underlying protocol used by the World Wide Web) requests containing its exploit code to random Internet Protocol (IP—an address that identifies a computer on a network) addresses. Such a simple target selection technique requires a large number of attempted attacks for each successful one. In order for Code Red to successfully infect the randomly selected IP address, the following criteria have to be met:

1 According to *Computer Economics* (see *http://www.computereconomics.com/article.cfm?id=936*), the associated damage from the Code Red and Nimda Worms came to $3.25 billion.
2 Buffer overflow vulnerabilities comprise a majority of the vulnerabilities that worms exploit.

- There is a computer at that IP address
- The computer at that IP address must be running an Internet Information Server (IIS—the Microsoft Web server that runs on Windows NT platforms)
- The IIS Web server application must be vulnerable to exploitation (the server is not yet patched or protected)

A later version of Code Red improved on the target selection mechanism by selecting target IP addresses that were numerically close to the IP address of the infected machine. For example, if the infected machine's IP address was 192.168.1.134, Code Red II randomly selected IP addresses that begin with 192.168.1 before targeting other IP addresses. Since computers on a given network often have similar addresses, this greatly improves the chances that the IP address chosen will actually have a computer at that IP address.

More intelligent, targeted worms may use predefined lists of known servers, telecommunication providers, well-known companies, government departments, domain name system (DNS) data, or other techniques to more efficiently select potential targets.

Trojan Horses and Bots

In the often repeated story of the Trojan horse used to get inside and attack the fortified and well-defended city of Troy from within, invaders used something that was seemingly benign as a vehicle for the attack: a large wooden horse. Similarly, Trojan horses of the information security world are seemingly benign programs that attack computer systems from within.

Once inside the computer, a Trojan program commonly replaces key system files and/or programs with malicious versions of the same. When these programs are executed, they perform their predetermined destructive activities, and users are powerless to stop them.

For example, an attacker may replace one of the Windows operating system dynamically linked libraries (DLLs) with a malicious version. DLLs are program files that Windows calls on to perform various tasks. An attacker may replace one of these DLLs with a Trojan horse version that does everything the normal DLL did, and a little more. That little more may be any number of things, from reformatting the hard drive to stealing credit card numbers.

In recent months, spyware and potentially unwanted programs have begun to wreak more havoc than worms or viruses. Although often unseen at first, PC users, particularly users connecting to the Internet from home, have increasingly noticed that their PCs are becoming slower and programs are crashing more often. In many cases, their PCs have become almost useless, because the memory and the processing power of the machines is taken up trying to send their private information from their PCs to the Internet. Or they are fighting off a myriad of unwanted spam advertisements that pop up onto their computer screens and advertise goods from all over the world. Most annoying of all, is the advertisement that continuously announces to and reminds the frustrated user that their PC now has a potentially unwanted program and they should purchase a particular software solution to clean it up.

Recently, hackers have utilized the distributed resources of thousands of Internet-connected PCs to launch Denial of Service (DoS) attacks against unlucky targeted organizations or servers. This is done by the master hackers who deposit their software code by Trojan horse onto the PCs, which then register with their host and await further instructions as to when and how to launch an attack. At a time chosen by the master hacker, the PC robots (*bots*), under external control, will launch their code and attack the designated target from the unwitting residential owner's PC. This way, the master attacker remains anonymous, but thanks to the unprotected home user, can utilize the resources of thousands of computers around the world to achieve his goal.

Potentially Unwanted Programs

Potentially unwanted program (PUPs) is the generic term given to programs that find their way onto a machine, and that the user may or may not want to have. Although often deposited by way of Trojan horse, PUPs can be made by a legitimate corporate entity for some beneficial purpose. A PUP is defined as ANY piece of code which a reasonably security- or privacy-minded computer user may want to be informed of, and, in some cases, remove. Even though they may be intended for legitimate corporate use, PUPs can alter the security state of the computer on which they are installed, and therefore most users will want to be aware of them. The main types of PUPs are:

Adware: software whose primary function is to make revenue through advertising targeted at the person using the computer on which it is installed. This revenue can be made by the vendor or partners of the vendor. This does not imply that any personal information is captured or transmitted as part of the software's functioning, though that is often the case.

Spyware: software whose function includes the transmission of personal information to a third party without the user's knowledge and explicit consent. This usage is distinct from the common usage of spyware to represent commercial software that has security or privacy implications.

Dialer: a piece of code that redirects Internet connections

to a party other than the user's default ISP for the purpose of incurring connection charges for a content provider, vendor, or other third party.

Remote administration tool: a tool designed to allow remote control of a system by a knowledgeable administrator. However, when controlled by a party other than the legitimate owner or administrator, remote administration tools are a large security threat.

Password cracker: a piece of code designed to allow a legitimate user or administrator to recover lost or forgotten passwords from accounts or data files. When in the hands of an attacker, these same tools allow access to confidential information and represent a security and privacy threat.

Joke: a piece of code that has no malicious payload or use and does not impact security or privacy states, but that may alarm or annoy a user.

Buffer Overflow Exploits

Buffer overflow exploits are one of the largest problems in computer security today. In all application programs, there are buffers that hold data. These buffers have a fixed size. If an attacker sends too much data into one of these buffers, the buffer overflows. The server then executes the data that "overflowed" as a program. Depending on the nature of the data sent by the attacker to the buffer, this program may do any number of things, from sending passwords to unknown destinations and hackers, to altering system files, or installing backdoors.

Programs can prevent buffer overflows by checking the length of the data submitted to the buffer before storing it in the buffer. If the data is too large, it returns an error. Unfortunately, in developing software programs or applications, many programmers forget to check the length of the data before saving it to a buffer. Therefore, applications that contain a large number of unchecked buffers are vulnerable to attack. When a vendor releases a patch to stop these potential buffer overflows, the patch simply adds code that checks the length of the data before it saves it to the buffer. If a patch is available, it will prevent a buffer from being overflowed.

Buffer overflow exploits are such a large problem for several reasons:

- Buffer overflow exploits are very common. There are hundreds of known unchecked buffers that can be overflowed by hackers, with more being discovered all the time. Over 50 percent of the CERT advisories deal with buffer overflow exploits
- Buffer overflow exploits are easy to use. Anyone (10-year-olds and script kiddies included) can download buffer overflow attack code and follow a simple "recipe" to execute it. No advanced technical knowledge is necessary to run pre-written buffer overflow exploit programs
- Buffer overflow exploits are very powerful. In many cases, the malicious code that executes as a result of a buffer overflow will run with administrator-level privileges, and therefore can do anything it wants to the server

Privilege Escalation Exploits

Privilege escalation exploits grant administrator or root-level access to users who previously did not have such access. For example, an account exists on all Windows NT and 2000 servers called "Guest." This account, by default, has no password. Anyone can log on to the server using this "Guest" account and then use a common privilege escalation exploit call "GetAdmin" to gain administrator-level access to the system. Many other privilege escalation exploits exist, such as HackDLL and others. These exploits are very useful to an attacker, since they allow anyone who has any level of access to a system, to easily elevate their privilege level and perform any activities they desire.

Backdoors

When attackers obtain root-level access to a server, such as using a buffer overflow exploit or a privilege escalation exploit, they will want to do two things:

1. Install a backdoor
2. Cover their tracks

Backdoors allow attackers to remotely access a system again in the future. For example, the attacker may have used a particular security hole to get root-level access to a computer. However, over time, that particular security hole may be closed, preventing the attacker from accessing the system again. In order to avoid being shut out in the future, attackers install backdoors. These backdoors take different forms, but all allow an attacker to access the server again without going through the standard login procedures and without having to repeat the attack that gave them access in the first place.

Many worms install backdoors as a part of their malicious payload. Code Red II, for example, installed a backdoor that provided access to the C and D drives of the compromised Web server from anywhere on the Internet. Other common backdoor programs are Netbus and BackOrifice, which allow attackers to remotely control a compromised server.

Rootkits

Rootkits are used to cover an attacker's tracks. If an attacker installs a backdoor or other malicious program, the system administrator may notice the new program and remove it, ending the hacker's ability to access the system in the future. The goal of a rootkit is to disguise the existence of malicious programs on a system.

By replacing certain system programs with modified versions of those same programs, rootkits mask the presence of backdoors or other malicious programs. For example, the UNIX program "Ls" prints a directory listing of the file system. This would normally allow a system administrator to see files left by an attacker. The rootkit installs a modified version of "Ls" that displays all the fields and programs in the directory except the backdoor program and any other fields left by the attacker. This effectively masks the evidence of the system compromise. Rootkits generally replace "Ls" as well as many other operating system programs to cover their tracks.

HTTP Exploits

HTTP exploits involve using the Web server application to perform malicious activities. These attacks are very common and are growing in popularity because firewalls typically block most traffic from the Internet to keep it away from corporate servers. However, HTTP traffic, used for Web browsing, is almost always allowed to pass through firewalls unhindered. Thus, attackers have a direct line to the Web server. If they can coerce the Web server into performing malicious activities, they can access resources that would otherwise be unavailable.

New HTPP exploits appear quite frequently. Some recent exploits include the Unicode directory traversal exploit and the double hex encoding exploit. Directory traversal exploits use strings like "./../" to access directories outside the normal webroot directory where Web content is stored. Since most Web servers will block URLs that contain "./", attackers circumvent this protection by using the special Unicode/hexadecimal encodings to represent the "./" pattern. By typing a properly crafted attack string into a Web browser, attackers can access other directories on the Web server. These other directories may contain confidential information, passwords, or other sensitive files. By using an HTTP exploit, attackers can access these files easily through a standard Web browser. Other HTPP exploits allow attackers to execute programs, alter system information, access registry keys, and perform other malicious activities.

Summary

The solution to the much publicized threat from malicious software viruses has materialized in the form of anti-virus software and anti-spyware that counters viruses and spyware programs as they are identified within infected files or as they transit thru a network.

However, the simple name "anti-virus" (commonly known as AV) does not properly convey the total protection that most anti-virus solutions now provide.

Varying levels of anti-virus protection can now be provided, the extent of the protection being offered being dependant upon the size and importance of the network equipment being protected.

At the edge of the network, enterprise desktop and commercial/retail anti-virus solutions targeted at residential end users now seek to provide a more general level of support to counter the effects of most of the malicious virus software, including worms and Trojan horses (which carry PUPs/spyware).

In the core of network, dedicated security appliances may be implemented to protect network assets from individual specific threats; for example, a Web server may be protected by a dedicated Web-server protection appliance, or company e-mails may be scanned for viruses and other forms of malware at the point of ingress by a dedicated, managed-mail security appliance.

For more information on the McAfee® range of anti-virus and anti-spyware solutions, please contact your local McAfee sales office or visit *www.mcafee.com*.

McAfee, Inc. 3965 Freedom Circle, Santa Clara, CA 95054, 888.847.8766, www.mcafee.com

6-sps-malware-001-1005