Network Worms

Thomas M. Chen*
Dept. of Electrical Engineering
Southern Methodist University
PO Box 750338, Dallas, Texas 75275
Tel: +1 214-768-8541
Email: tchen@engr.smu.edu

Gregg W. Tally
SPARTA, Inc.
7110 Samuel Morse Drive
Columbia, Maryland 21046
Gregg.tally@sparta.com

Introduction

Internet users are currently plagued by an assortment of malicious software (malware). The Internet provides not only connectivity for network services such as e-mail and Web browsing, but also an environment for the spread of malware between computers. Users can be affected even if their computers are not vulnerable to malware. For example, fast-spreading worms can cause widespread congestion that will bring down network services.

Worms and viruses are both common types of self-replicating malware but differ in their method of replication (Grimes, 2001; Harley, Slade, and Gattiker, 2001; Szor, 2005). A computer virus depends on hijacking control of another (host) program to attach a copy of its virus code to more files or programs. When the newly infected program is executed, the virus code is also executed. In contrast, a worm is a standalone program that does not depend on other programs (Nazario, 2004). It replicates by searching for vulnerable targets through the network, and attempts to transfer a copy of itself. Worms are dependent on the network environment to spread. Over the years, the Internet has become a fertile environment for worms to thrive.

The constant exposure of computer users to worm threats from the Internet is a major concern. Another concern is the possible rate of infection. Since worms are automated programs, they can spread without any human action. The fastest time needed to infect a majority of Internet users is a matter of speculation, but some worry that a new worm outbreak could spread through the Internet much faster than defenses could detect and block it. The most reliable defenses are based on attack signatures. If a new worm does not have an existing signature, it could have some time to spread unhindered and complete its damage before a signature can be devised for it.

Perhaps a greater concern about worms is their role as vehicles for delivery of other malware in their payload. Once a worm has compromised a host victim, it can execute any payload. Historical examples of worms have included:

- Trojan horses: software with a hidden malicious function, e.g., to steal confidential data or open a backdoor;
- droppers: designed to facilitate downloading of other malware;

- bots: software to listen covertly for and execute remote commands, e.g., to send spam or carry out a distributed denial of service (DDoS) attack.

These types of malware are not able to spread by themselves, and therefore take advantage of the self-replication characteristic of worms to spread.

This article presents a review of the historical development of worms, and an overview of worm anatomy from a functional perspective.

Background

The term "worm" was created by John Shoch and Jon Hupp at Xerox PARC in 1979, inspired by the network-based multi-segmented "tapeworm" monster in John Brunner's novel, *The Shockwave Rider* (Shoch and Hupp, 1982). They were aware of an earlier self-replicating program, Creeper, written by Bob Thomas at BBN, which propelled itself between nodes of the ARPANET. They invented a worm to traverse their internal Ethernet LAN seeking idle processors after normal working hours for the purpose of distributed computing. Since the worms were intended for beneficial uses among cooperative users, there was no attempt at stealth or malicious payload. Their worms were designed with limited lifetimes, and responsive to a special "kill" packet. Despite these safeguards, one of the worm programs believed to have been accidentally corrupted ran out of control and crashed several computers overnight.

The most famous worm incident was the Morris worm in November 1988 that disabled 6,000 computers in a few hours (Spafford, 1989). Robert Morris Jr. was a student at Cornell University at the time. The damage was caused by the worm re-infecting computers that were already infected, until the computers slowed down and crashed. It was probably the first worm to use a combination of methods to spread quickly. First, it attempted to crack password files on Unix systems. The password file was encrypted but publicly readable. The worm could encrypt password guesses and compare them to the contents of the password file. Second, it exploited the debug option in the Unix sendmail program. Third, it carried out a buffer overflow exploit taking advantage of a vulnerability in the Unix finger daemon program.

Worm development was relatively slow until 1999 when e-mail became a popular infection vector. In March 1999, Melissa spread to 100,000 computers in 3 days, setting a new record and shutting down e-mail for many companies using Microsoft Exchange Server (CERT advisory CA-1999-04, 1999). It was a Microsoft Word macro that used the functions of Word and Outlook e-mail to propagate. When the macro is executed in Word, it launched Outlook and sent itself to 50 recipients found in the address book. Additionally, it infected the Word normal.dot template, so that any Word document created from the template would carry the infection.

In the summer of 1999, the PrettyPark worm propagated as an e-mail attachment called "Pretty Park.exe" with the icon of a character from the television show, "South Park." If executed, it installed itself into the system folder and modified the registry to ensure that it ran whenever any .exe program was executed. It e-mailed itself to addresses found in the address book. Another worm, ExploreZip, appeared to be a WinZip file attachment in e-mail but was not really a zipped file. When executed, it displayed an error message but the worm secretly copied itself into the systems folder and loaded itself into the registry. It e-mailed itself using Outlook or Exchange to recipients found in unread messages in the inbox. It monitored all incoming messages and replied to senders with a copy of itself.

The summer of 2000 saw more mass mailing worms. In May 2000, the Love Letter worm appeared with the subject line "I love you" and encouraged the recipient to read the attachment which was a Visual Basic script (CERT advisory CA-2000-04, 2000). When executed, the worm installed copies of itself into the windows and system directories and modified the registry to ensure that it would be run during bootup. It infected various types of files (.vbs, .jpg, .mp3, etc.) on local drives and networked shared directories. If Outlook is installed, the worm e-mailed copies of itself to anyone found in the address book. In addition, the worm sent copies of itself via IRC channels.

Appearing around the same time, NewLove was a Visual Basic script worm. It was interesting as a polymorphic worm that tried to change its appearance in every copy. The worm forwarded itself with a file name chosen randomly from "recent documents" to all addresses in the Outlook address book. The e-mail has no text but has a subject line including the new file name.

In October 2000, the Hybris worm spread as an e-mail attachment (CERT incident note IN-2001-02, 2001). If executed, it modified the "wsock32.dll" file in order to track all Internet traffic at the infected host. For every e-mail sent, it subsequently sent a copy of itself to the same recipient. It had the interesting capability to receive plug-ins dynamically by connecting to a pre-programmed newgroup. The plug-ins were encrypted and updated the worm code. This capability is potentially dangerous because the worm functionality can be changed at any time by the worm author.

A new wave of more sophisticated worms began in early 2001. In March 2001, the Lion worm spread among Linux computers using the "pscan" application, a freely distributed network port scanner written in Perl. The worm used this port scanner in combination with the "randb" program to scan class B hosts listening on TCP port 53 that were vulnerable to the BIND buffer overflow vulnerability. It then attacked these hosts using an exploit called "name." After a system was compromised, the worm stole password files and other sensitive information (IP address, accounts) and sent these by e-mail. It also installed several things: the t0rn rootkit to evade detection, the DDoS agent TFN2K, a Trojanized version of SSH to listen on port 33568, and backdoor root shells on TCP ports 60008 and 33567.

In May 2001, the Sadmind worm first exploited a buffer overflow vulnerability in Sun Solaris systems. These compromised systems were then used to carry out an attack to compromise Microsoft IIS (Internet Information Services) Web servers.

In July 2001, the Code Red worm caused major damages by exploiting a buffer overflow vulnerability discovered in Microsoft IIS Web servers about a month earlier (Berghel, 2001; Moore, Shannon, and Brown, 2001). Specifically, the Index Server ISAPI vulnerability allowed a remote attacker to gain full system level access (Microsoft Security Bulletin MS01-033, 2001). The first version of the Code Red worm appeared on July 12. On infected systems, it set up 100 parallel threads, each an exact replica of the worm, in order to spread faster. It attempted to generate pseudorandom IP addresses but used a static seed which (apparently unintentionally) resulted in identical lists of IP addresses. Although 200,000 hosts were infected in 6 days, the worm was slowed down by the fact that the same targets were getting hit repeatedly. A second version of Code Red appeared on July 19. This version spread much faster because the static seed had been changed to a random seed, ensuring that each copy of the worm generated different IP addresses. More than 359,000 computers were reportedly infected by Code Red version 2 within 14 hours. By design, the worm stopped by itself on July 20. On August 4, a new worm self-named Code Red II used the same buffer overflow exploit but a different payload. It

generated random IP addresses but they are not completely random; about 1 out of 8 are completely random; 4 out of 8 addresses are within the same class A range of the infected host's address; and 3 out of 8 addresses are within the same class B range of the infected host's address. On infected systems, it activated 300 parallel threads to spread faster. The enormous number of parallel threads created a flood of scans, resulting in serious network congestion.

In September 2001, the Nimda worm used a combination of five methods to spread quickly: e-mail to addresses from the host's Web cache and default MAPI mailbox, with random subject lines and an attachment named "readme.exe"; attacked random Microsoft IIS Web servers though a buffer overflow vulnerability published a year earlier; copied itself across open nework shares; added Javascript to Web pages to infect Web browsers; and looked for backdoors left by previous Code Red II and Sadmind worms. It was able to spread to 450,000 hosts within 12 hours. Although none of the methods was new, the combination of so many methods in one worm was unusually complex.

In November 2002, the Winevar worm was an example showing the capability to protect itself by disabling antivirus software. It used a list of keywords to scan memory to stop recognized antivirus processes and scan the hard drive to delete associated files.

In January 2003, the SQL Slammer (or Sapphire) worm spread among Microsoft SQL servers (Moore, et al., 2003). Interestingly, the worm consists simply of a 376-byte payload in a single 404-byte UDP packet. This is advantageous for fast spreading because infected hosts can generate these short UDP packets quickly. Unlike TCP, UDP is connectionless and does not require a host to wait for a connection set up. Infected hosts were put into a simple loop to send UDP packets to randomly generated IP addresses as fast as possible. The packets carried an exploit for a buffer overflow vulnerability in Microsoft SQL Server discovered six months earlier (Microsoft Security Bulletin MS02-039, 2002).

The week of August 11, 2003, has been called one of the worst weeks in worm history. First, the Blaster (or Lovsan) worm exploited a DCOM RPC (distributed component object model remote procedure call) vulnerability in Windows 2000 and Windows XP systems. On vulnerable systems, the worm opened a remote shell process that transfers a file "msblast.exe" from an infected host. Seven days later, the Welchi (or Nachi) worm used the same exploit along with an exploit for a WebDAV vulnerability in Microsoft IIS 5.0 servers. Interestingly, Welchi attempted to remove Blaster from infected systems and applied the Microsoft patch for the RPC vulnerability. It was programmed to self terminate on January 1, 2004. One day after Welchi, the Sobig.F worm, the fifth variant of the original Sobig.A worm discovered in January 2003, spread by mass mailing. It spoofed the "from" address in e-mails with a randomly chosen address found on the victim's computer. It had the capability to download arbitrary files to an infected computer. It was used to set up spam relay servers and steal confidential system information. At pre-programmed times, it contacted a number of master servers to get download instructions. Around the same day, the Dumaru worm pretended to be a Microsoft patch "patch.exe" attachment in e-mail. If opened, the worm copies itself into the system directory and installs a Trojan horse that listens to an IRC channel for commands from the worm author.

2004 was notable for a conflict between the authors of MyDoom, Netsky, and Bagle, evidenced by messages embedded in the worm codes. The MyDoom.A worm appeared in January. It e-mailed itself to addresses harvested from various types of files on the infected host, along with various subject lines and attachment names. The payload contains a DDoS agent and a backdoor to download arbitrary files. Soon afterward, the Bagle worm spread similarly by e-mail and installed a Trojan horse that opened a backdoor to allow remote control. The Netsky

family of worms, also mass mailers, appeared shortly afterward with comments embedded in its code directed at the authors of MyDoom and Bagle, and some variants contained code to remove them from infected hosts.

Although worms have continued to evolve since 2004, there have not been "big" worm outbreaks on the scale of Slammer or Code Red. Worm writers have seemed to be spending more efforts towards exploring new infection vectors, such as instant messaging, Internet relay chat (IRC), peer-to-peer file sharing, or SMS/MMS (short message service/multimedia messaging service). It could be said that worms are still perceived as a major threat but fading in importance compared to other emerging malware threats. Since 2005, concern has been gradually shifting away from worms towards other types of malware, namely bots, spyware, and rootkits.

Worm anatomy

Worms must have certain functions in their code for self replication:

- target identification: to locate new targets
- infection mechanism: to compromise a new target
- replication: to transfer a worm copy to a target.

Optionally, worms might contain timing control and a payload. Timing might be controlled for self termination; downloading plug-ins or worm code updates; downloading new malware to infected systems; or activation of the payload.

Worms do not always carry a payload, and payloads can be virtually anything. Payloads such as a DDoS agent might be activated by the timing control (e.g., to start flooding at the same time) (Mirkovic, Dietrich, Dittrich, and Reiher, 2004).

Target identification

The simplest method to find new targets is randomly chosen IP addresses (essentially 32-bit numbers). However, this approach is not efficient. As more hosts become infected, the spreading rate slows down due to infected hosts hitting targets that are already infected. This inefficiency creates excessive traffic in the network, which slows down the spreading rate further.

Worms such as Blaster and Code Red II have used more complicated algorithms for IP addresses. Blaster chose a random IP address only 60% of the time; at other times, it attempted to find an address in the same local network as the infected victim. Code Red II chose random IP addresses 1 out of 8 times; 4 out of 8 addresses were within the same class A range; and 3 out of 8 addresses were within the same class B range as the infected host.

Another popular method is to harvest e-mail addresses from the victim host. Early mass mailing worms starting with Melissa found addresses from the address book. More sophisticated worms such as MyDoom can harvest e-mail addresses from many types of files located on a victim. The rationale for targeting addresses found from a victim is that recipients are more likely to read e-mail if it was apparently sent from an acquaintance.

Infection vectors

It is apparent from the historical review that worms can spread by any number of ways. Since 1999, e-mail has been one of the most popular infection vectors because: worms often

carry their own SMTP engine; e-mail can take advantage of social engineering; messages can be easily forged and mutated; e-mail can take advantage of social connections which may be more effective than random contacts.

Worms can also exploit vulnerabilities. The most common type of exploit is a buffer overflow because: it can usually be done remotely; it can give complete control over a target; and buffer overflow vulnerabilities are found in many operating systems and applications (Foster, Osipov, and Bhalla, 2005). Even the early Morris worm used a buffer overflow exploit.

Worms such as Lirva and Fizzer were able to spread by file sharing, namely the KaZaa peer-to-peer network. The worm resides in a shared folder, usually with a harmless name.

Worms can spread by messaging via instant messaging or IRC (Internet relay chat). The 2003 Lirva worm was able to spread by IRC. An infected file or URL is sent to a chat channel. In March 2005, the Kelvir family of worms began to spread by instant messaging via MSN Messenger. Random looking messages contained a link to a Web site which attempted to download files. When downloaded and executed, the worm continues to spread by sending instant messages to all found MSN messenger contacts.

Additional infection vectors include: password cracking (e.g., Morris worm); copy to open network shares; modification of Web sites for drive-by downloading; taking advantage of backdoors left by previous worms or Trojan horses (e.g., Nimda worm); and spreading by Bluetooth, SMS/MMS (short message service/multimedia messaging service), or other wireless connections (Hypponen, 2006).

Payloads

The optional payload of a worm is executed after a new victim has been compromised. Some worms have no payloads, and the reason is not known for certain. A payload could be virtually anything. In past cases, common payloads have included: bots to control a group of infected hosts as a bot net (Schiller and Binkley, 2007); spam relay servers to generate spam; backdoors to allow covert remote access; DDoS agents such as TFN2K; spyware, key loggers, and other Trojan horses; and rootkits to evade detection. While destructive payloads are entirely possible, it might be counterproductive, resulting in slower spreading and more attention from security experts.

The payload is often considered to be a clue to the worm author's motivations. When a payload is absent, the worm might be a proof of concept (e.g., to see how fast it could spread). Payloads for spamming or stealing personal information suggests a profit motive.

Future Trends

Although widespread outbreaks of fast-spreading worms have been less common since 2003, worms are still a serious threat according to most surveys of organizations. The nature of the threat has simply continued to evolve.

First, worms continue to expand to new infection vectors. For example, the Cabir worm in June 2004 was the first to spread by Bluetooth between Symbian smartphones (Hypponen, 2006). This was followed by the ComWar worm in March 2005 using MMS as the infection vector. ComWar was followed shortly by the Mabir worm which was able to spread by both MMS and Bluetooth.

Second, there has been a growing prevalence of payloads oriented towards control (bots, backdoors, rootkits) and financial profit (spyware, keyloggers). Anti-spyware and rootkit detection programs are quickly becoming essential protection for computer users. These other types of malware do not have to be delivered by worms. For example, malware can spread by drive-by-downloading at a malicious Web site. But worms continue to be a popular vehicle to deliver a variety of malware.

Third, social engineering continues to be common. Malware writers have been quick to take advantage of interest in current events to entice e-mail recipients to read spam. Another example is one of the most prevalent worms in 2006 was the Nyxem (or Blackmal or "Kama Sutra") worm which offered sexually provocative subject lines and body texts.

Conclusion

Worm evolution has progressed from early experimentation to sophisticate vehicles for other types of malware. Worms are commonplace on the Internet and threaten to expand to other networking environments such as wireless.

Unfortunately, the nature of the worm threat is essentially similar to other criminal activities. Worms are created by criminals, and it is impossible to predict how new worms will be invented. Thus, defenses are always catching up to new attacks. There are natural questions that may always be somewhat uncertain. For instance, when will another major worm outbreak happen? How fast could a worm spread and what damage will be caused? Continued research is needed to address these questions.

References

Berghel, H. (2001). The Code Red worm. Communications of the ACM, 44(12), 15-19.

CERT advisory CA-1999-04. (1999). Melissa macro virus. Retrieved February 2, 2007 from http://www.cert.org/advisories/CA-1999-04.html.

CERT advisory CA-2000-04. (2000). Love letter worm. Retrieved February 2, 2007 from http://www.cert.org/advisories/CA-2000-04.html.

CERT incident note IN-2001-02. (2001). Open mail relays used to deliver Hybris worm. Retrieved February 2, 2007 from http://www.cert.org/incident_notes/IN-2001-02.html.

Foster, J., Osipov, V., and Bhalla, N. (2005). Buffer Overflow Attacks. Rockland, MA: Syngress Publishing.

Grimes, R. (2001). Malicious Mobile Code: Virus Protection for Windows. Sebastopol, CA: O'Reilly & Associates.

Harley, D., Slade, R., and Gattiker, R. (2001). Viruses Revealed. New York, NY: Osborne/McGraw-Hill.

Hypponen, M. (2006). Malware goes mobile. Scientific American, 295(5), 70-77.

Microsoft Security Bulletin MS02-039. (2002). Buffer overruns in SQL Server 2000 resolution service could enable code execution. Retrieved February 2, 2007 from http://www.microsoft.com/technet/security/bulletin/MS02-039.asp.

Microsoft Security Bulletin MS01-033. (2001). Unchecked buffer in Index Server ISAPI extension could enable web server compromise. Retrieved February 2, 2007 from http://www.microsoft.com/technet/security/bulletin/MS01-033.asp.

Mirkovic, J., Dietrich, S., Dittrich, D., and Reiher, P. (2004). Internet Denial of Service: Attack and Defense Mechanisms. Upper Saddle River, NJ: Prentice Hall.

Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., and Weaver, N. (2003). Inside the Slammer worm. IEEE Security & Privacy, 1(4), 33-39.

Moore, D., Shannon, C., Brown, J. (2002). Code-Red: a case study on the spread and victims of an Internet worm. Proc. of ACM Internet Measurement Workshop 2002, Marseille, 273-284.

Nazario, J. (2004). Defense and Detection Strategies Against Internet Worms. Norwood, MA: Artech House.

Schiller, C., and Binkley, J. (2007). Botnets: the Killer Web App. Rockland, MA: Syngress Publishing.

Shoch, J., and Hupp, J. (1982). The 'worm' programs - early experience with a distributed computation. Communications of the ACM, 25(3), 172-180.

Spafford, E. (1989). The Internet worm program: an analysis. ACM Computer Communications Review, 19(1), 17-57.

Szor, P. (2005). The Art of Computer Virus Research and Defense. Upper Saddle River, NJ: Addison-Wesley.

Key Terms and Definitions

Computer virus: a set of program instructions capable of self replication by attaching to a normal host file or program.

Exploit: code written to take advantage of a specific vulnerability.

Infection vector: the transmission channel for spreading an infection.

Malicious software (malware): the broad variety of software containing a harmful function, such as viruses, worms, and Trojan horses.

Payload: the part of a virus or worm that is executed after a target host has been successfully compromised and infected.

Social engineering: a type of attack taking advantage of human gullibility.

Vulnerability: a weakness or bug in software programs that could lead to a security compromise if exploited.

Worm: an automated standalone program capable of self replication by copying itself to vulnerable hosts through a network.