Network Worms

Thomas M. Chen
Southern Methodist University, USA

Greg W. Tally
SPARTA Inc., USA

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, & 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. Because 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, for example, 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, for example, 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 multisegmented "tapeworm" monster in John Brunner's novel, *The Shockwave Rider* (Shoch & 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. Because 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 preprogrammed newsgroup. 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, & 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.