Malicious Software

Thomas M. Chen
Southern Methodist University, USA

Gregg W. Tally
SPARTA, Inc., USA

INTRODUCTION

Malicious software (malware) allows an intruder to take over or damage a target host without the owner's consent and often without his or her knowledge. Over the past thirty years, malware has become a more serious worldwide problem as Internet-connected computers have proliferated and operating systems have become more complex. Today, the average PC user must be more cognizant of computer security than ever before due to the constant threat of possible infection. Although exact costs are difficult to determine, there is little doubt that malware has widespread impact on equipment damages, loss of data, and loss of productivity. According to surveys, malware is one of the most common and costly types of attack on organizations (CERT, CSO, & ECTF, 2005).

In the early days of computing, malware was predominantly viruses and Trojan horses that spread among computers mainly by floppy disks and shared files (Grimes, 2001). The typical virus writer was a young male experimenting by himself and looking for notoriety. Today, malware is largely worms, viruses, spyware, bots, and Trojans proliferating through computer networks. Worms are a particular concern due to their ability to spread by themselves through computer networks. They can exploit weaknesses in operating systems or common applications such as Web and e-mail clients. They are often used as vehicles to install other types of malware onto hosts. Many thousands of worms and viruses are constantly tracked by the WildList (Wildlist Organization International, 2006) and antivirus companies.

Naturally, host-based and network-based defenses have also evolved in sophistication in response to growing threats. Surveys have found that organizations almost universally use antivirus software, firewalls, intrusion detection systems, and other means of protection (Gordon, Loeb, Lucyshyn, & Richardson, 2005). These defenses certainly block a tremendous amount of malware and prevent global disasters. However, their effectiveness is widely known to be limited by their ability to accurately detect malware. Detection accuracy is critical because malware must be blocked without interfering with legitimate computer activities or network traffic. This difficulty is compounded by the creativity of attackers continually attempting to invent new methods to avoid detection.

BACKGROUND

Self-Replicating Malware

Malware can be classified into self-replicating or non-self-replicating. Self-replicating malware consists of viruses and worms. Fred Cohen originated the term virus after biological viruses for their manner of parasitically injecting their RNA into a normal cell, which then hijack the cell's reproductive process to produce copies of the virus (Cohen, 1994). Analogously, computer viruses attach their code to a normal program or file, which takes over control of execution of the infected program to copy the virus code to another program.

Polymorphism was a major development in virus evolution around 1990. Polymorphic viruses are able

to scramble their form to have at most a few bytes in common between copies to avoid detection by virus scanners. In 1991, the dark avenger's mutation engine was an easy to use program for adding polymorphism to any virus. A number of other "mutation engines" were subsequently created by other virus writers.

A new wave of mass-mailing viruses began with Melissa in 1999. It was a macro virus infecting Microsoft Word normal templates. On infected computers, it launched Microsoft Outlook and e-mailed copies of itself to 50 recipients in the address book. It demonstrated the effectiveness of e-mail as a propagation vector, infecting 100,000 computers in 3 days. Since then, e-mail has continued to be a popular vector for viruses and worms because e-mail is used by everyone across different operating systems (Harley, Slade, & Gattiker, 2001). Mass-mailing worms today often carry their own SMTP engines to mail themselves and circumvent security features in e-mail programs.

Whereas viruses are program fragments dependent on execution of a host program, worms are standalone programs capable of spreading by themselves (Nazario, 2004; Skoudis, 2004). A worm searches for potential targets through a computer network and sends a copy of itself if the target is successfully compromised. Worms take advantage of networks and have proliferated as Internet connectivity has become ubiquitous.

One of the earliest and most famous worms was written by Robert Morris Jr. in 1988. Perhaps released accidentally, it disabled 6,000 hosts, which was 10% of the ARPANET (the predecessor to the Internet). A number of fast worms, notably Code Red I, Code Red II, and Nimda appeared in 2001. Two years later, another wave of fast worms included SQL Slammer/Sapphire, Blaster, and Sobig.F. The following year was dominated by MyDoom, Netsky, and Bagle worms (Turner et al., 2006).

Nonself-replicating malware classification of nonself-replicating malware into disjoint subcategories is difficult because many types of nonself-replicating malware share similar characteristics. Perhaps the largest category is Trojan horses defined as programs with hidden malicious functions. A Trojan horse may be disguised as a legitimate program to avoid detection. For example, a Trojan horse could be installed on a host with the name of a legitimate system file (displacing that file). Alternatively, the intention of the disguise could be to deceive users into executing it. For example, a Trojan horse could appear to be a graphic attachment in an e-mail message but in actuality be a malicious program. Trojans do not replicate by themselves but could spread by file sharing or downloading.

Remote administration or access trojans (RATs) are a well-known type of trojan horse giving covert remote control to attackers. One of the first was Netbus written in 1998. It works in a client-server fashion with the server component installed on the target machine responding to the attacker's client. Another well-known RAT was Back Orifice released by Cult of the Dead Cow in 1998, which was later released as an open source version Back Orifice 2000.

A backdoor is software giving access to a system bypassing normal authentication mechanisms (Skoudis, 2004). Programmers have written backdoors sometimes to allow convenient access for legitimate testing or administrative purposes, but backdoors can be installed and exploited by attackers to maintain covert remote control after a target has been compromised. For example, the Nimda worm dropped a backdoor on infected hosts.

Relatively recently, bots such as Spybot and Gaobot have become a major problem (Turner et al., 2006). Bots installed on a group of hosts act as a large bot net to carry out a remote attacker's instructions which are typically communicated via Internet relay chat (IRC). Bot net sizes in the thousands to hundreds of thousands have been observed. Bot nets have been rented or sold as platforms for spamming, distributed denial of service, and other criminal activities (Lewis, 2005).

A rootkit is low-level software, possibly at the kernel level, designed to conceal certain files and processes. Rootkits are sometimes bundled as part of malware such as worms (Hoglund & Butler, 2006) because the concealment allows attackers to maintain longer control over their targets.

Spyware is software that collects and sends personal information through the network to a remote attacker

(Evans, 2005). Spyware may be bundled with a legitimate program, and its presence may be mentioned in a end user license agreement (EULA). Commonly, a type of spyware called adware is bundled for the purpose of collecting information about user behavior to customize delivery of advertising. Accepting the EULA is considered explicit agreement to installation of the spyware, but many people neglect to read EULAs carefully. More pernicious types of spyware deliberately hide their presence and attempt to steal personal data by recording data to a file which is transmitted to or retrieved by a remote attacker.

MALICIOUS SOFTWARE

Malware involves an ongoing conflict between attackers and defenders. Worms are a prime example of a malware attack. Computers are typically protected by a combination of host-based and network-based defenses.

Self replication basics worms actively select and attack their targets through a network automatically. The capability for self replication is enabled by certain functions in the worm code (Skoudis, 2004). First, a function for target location chooses the next host for attack. The simplest algorithm chooses random IP address as pseudorandomly generated 32-bit numbers. Random target selection is not completely effective because the B and C class address spaces are more populated. Hence, some worms target B and C class addresses more often. Also, some worms favor targets on the same local area network as the victim because they are easier to reach. Another common way to identify targets is to harvest e-mail addresses from the victim host.

Second, a function in the worm code must contain the infection mechanism to compromise a selected target. The most common method is an exploit of a vulnerability. Most operating systems and applications software have vulnerabilities or weaknesses discovered over time. The most common type of vulnerability is a buffer overflow, which can lead to running arbitrary malicious code on a target host if attacked successfully (Foster, Osipov, Bhalla, & Heinen, 2005). When a vulnerability is discovered, the software developer is usually notified privately and given a chance to develop a patch or update. The vulnerability may be publicly disclosed later along with the patch. Vulnerabilities are regularly published in Microsoft security bulletins, CERT advisories, Bugtraq, MITRE CVEs, and other places. This process allows users to update their systems before attackers can write the exploit code that takes advantage of the vulnerability. Other vulnerabilities may be discovered by attackers but not disclosed, in hopes of catching targets unprotected against so-called zero-day exploits.

Exploits are not the only way for worms to spread. Social engineering takes advantage of human gullibility to trick users into taking an action to help the worm (e.g., opening an e-mail attachment). Password attacks attempt to compromise a target by trying default passwords, easily guessed passwords, or cracking the password file. Another way to spread is to look for backdoors left by other worms.

Worms can easily include multiple exploits to compromise more targets faster. The Morris worm was an example using a combination of different exploits to attack targets: a buffer overflow exploit of the Unix finger daemon; an exploit of the debug mode of the sendmail program; and cracking the password file by a dictionary attack. Another prominent example of a blended threat was Nimda in 2001, using five different vectors.

A third function in the worm code enables replication of the worm to a compromised target. Replication might be combined with the exploit. For example, SQL Slammer/Sapphire carried a buffer overflow exploit and a copy of the worm within a single 404-byte UDP packet.

Finally, worm code may optionally contain a payload. The payload is executed on the target and might be virtually anything such as data theft, data deletion, or installation of other malware.

Host-Based Defenses

The most common suite of host-based defenses includes antivirus software, spyware detection software, and a personal firewall. Antivirus and antispyware software

aim to identify specific malware, disinfect, or remove infected files, and prevent new infections if possible. Antivirus and antispyware programs largely work by signatures, which are sets of characteristics that will identify a specific malware (Szor, 2005). Signatures are preferred for their accuracy in identifying known malware, but new malware without a matching signature can escape detection. Antivirus software typically include heuristic rules to detect suspicious new malware based on their behavior or construction. For example, behavior blocking looks at the behavior of programs and raises a warning if the behavior appears suspicious. The disadvantage of heuristics is a possibly high rate of false positives (false alarms).

Another defense against malware is software patching. Software developers often publicize new vulnerabilities along with patches for them. This works for known vulnerabilities but not all vulnerabilities are known by the developers. Also, it can be inconvenient for users to keep up with regular patching.

Host-based intrusion detection systems are processes that observe system activities and raise alarms for suspicious activities. For example, if someone fails several consecutive login attempts, that would be a suspicious activity suggesting that the person does not know the correct password.

Lastly, computers typically include personal firewalls, implemented as software at the network interface. Incoming and outgoing traffic is blocked according to the firewall policies. There might be firewalls on the perimeter of a user's network, but a personal firewall allows packet filtering to be customized to individual preferences.

Network-Based Defenses

Compared to host-based defenses, network-based defenses have the advantage of providing broad protection to groups of users without any special requirements on hosts (Nazario, 2004). Firewalls are perhaps the best known network defense (Northcutt, Zeltser, Winters, Fredrick, & Ritchey, 2002). Firewalls apply filtering rules to block malicious traffic including malware. Rules are often based on fields in packet header fields such as source and destination addresses, source and destination ports, and protocol.

Routers with access control lists (ACLs) can block traffic similarly to firewalls. Routers must process packet headers for the purpose of forwarding packets along the correct routes. ACLs are simply additional rules to specify which packets are dropped.

Network-based intrusion detection systems (IDS) are specialized equipment to observe and classify traffic as normal, suspicious, or malicious. IDS raise alarms for suspicious traffic but do not take active actions (intrusion prevention systems have that additional capability to block malicious traffic). Like antivirus software, IDS typically work by a combination of signature-based and behavior-based detection (also called misuse and anomaly detection). Signatures are traffic characteristics that unique identify malware traffic and are preferred for accurate detection. However, not all malware traffic is known, and therefore malware might escape signature-based detection (Riordan, Wespi, & Zamboni, 2005). Behavior-based or anomaly detection aims to identify all suspicious traffic that deviates in some sense from normal traffic.

Honeypots are decoy computers intentionally set up to look vulnerable to attackers (Spitzner, 2003). They are not used for legitimate services so all traffic received by a honeypot is unsolicited and inherently suspicious. Their general purpose is to learn about attacker behavior but can be configured to collect malware, particularly worms that choose their targets automatically and randomly. The risks associated with malware impose the necessity for special precautions to limit possibly compromised honeypots from spreading malware to other computers.

CHALLENGES

New vulnerabilities are constantly being discovered in operating systems and applications software, giving rise to new exploits for malware. Turner et al. (2006) reported an average of 10 new vulnerabilities discovered per day. Accurate detection of new exploits requires signatures, but signatures usually takes a few hours

to days to develop. In the absence of a signature, the effectiveness of defenses will depend on the accuracy of anomaly (or behavior-based) detection. Anomaly detection based on unique behavioral traits of worms is an active area of research (Al-Hammadi & Leckie, 2005; Gu, Sharif, Qin, Dagon, Lee, & Riley, 2004; Kawaguchi, Azuma, Ueda, Shigeno, & Okada, 2006). For example, random worms might be inferred by the observation of a large number of failed connection messages (Berk, Bakos, & Morris, 2003). Another active research problem is automated defenses after detection such as automatic generation of worm signatures (Newsome, Karp, & Song, 2005; Simkhada, Tsunoda, Waizumi, & Nemoto, 2005) or dynamic quarantine (Moore, Shannon, Voelker, & Savage, 2003).

The situation is complicated by the many means of self-preservation that malware today often use. First, malware attempts to be stealthy through polymorphism or rootkit techniques. Second, malware can actively attack defenses. It is not uncommon for viruses and worms to disable antivirus software on targets by stopping antivirus processes and disabling registry keys. Third, malware has the capability to dynamically download new code or plug-ins, changing its functionality.

FUTURE TRENDS

Malware is always seeking new propagation vectors in addition to the Internet. Recently, malware has begun to spread via wireless networks to mobile devices such as cell phones and PDAs and is increasingly targeting instant messaging (Turner et al., 2006). E-mail and social engineering will continue to be popular propagation vectors.

The changing nature of payloads, increasingly towards remote control and data theft, suggests that malware is become more used for cybercrimes. Malware for profit has been called crimeware. This trend is also suggested by increasing use of stealth techniques.

Finally, worm outbreaks have become faster than humans can respond. For example, SQL Slammer/ Sapphire is reported to have infected 90 percent of the vulnerable hosts within 10 minutes. This trend means more dependence on automated defenses in the future. However, the effectiveness of automated defenses will depend on a solution to the problem of accurate detection.

CONCLUSION

Current defenses based on signatures and anomaly detection are imperfect. Signatures are preferred for accuracy but take time to develop and distribute. On the other hand, anomaly detection has the difficult challenge of differentiating normal from malicious behavior. In the future, malware attacks will be carried out faster, and we will depend more on automated defenses. These defenses will need solutions to automating signature development and making anomaly detection more accurate.

Finally, users are an important part of security. Since malware often use social engineering, user education and awareness of secure practices (such as patching and antivirus updating) are essential. Just as with anything valuable, users must be constantly vigilant to protect their computers and data.

REFERENCES

Al-Hammadi, Y., & Leckie, C. (2005). Anomaly detection for Internet worms. In *Proceedings of IEEE IM 2005* (pp. 133-146).

Berk, V., Bakos, G., & Morris, R. (2003). Designing a framework for active worm detection on global networks. In *Proceedings of the 1st IEEE International Workshop on Info. Assurance* (pp. 13-23).

CERT, CSO, and ECTF. (2005). *2005 e-crime watch survey*. Retrieved April 24, 2006, from http://www.cert.org/archive/pdf/ecrimesummary05.pdf

Cohen, F. (1994). *A short course on computer viruses*. New York: Wiley & Sons.

Evans, G. (2005). *Spyware study and reference guide*. Marina Del Rey, CA: Ligatt Publishing.

Foster, J., Osipov, V., Bhalla, N., & Heinen, N. (2005). *Buffer overflow attacks: Detect, exploit, prevent*. Rockland, MA: Syngress Publishing.

Gordon, L., Loeb, M., Lucyshyn, W., & Richardson, R. (2005). *CSI/FBI computer crime and security survey*. Retrieved April 24, 2006, from http://www.gocsi.com

Grimes, R. (2001). *Malicious mobile code*. Sebastopol, CA: O'Reilly & Associates.

Gu, G., Sharif, M., Qin, X., Dagon, D., Lee, W., & Riley, G. (2004). Worm detection, early warning, and response based on local victim information. In *Proceedings of the 20th IEEE Annual Computer Sec. Applic. Conf*. (pp. 136-145).

Harley, D., Slade, R., & Gattiker, R. (2001). *Viruses revealed*. New York: McGraw-Hill.

Hoglund, G., & Butler, J. (2006). *Rootkits: Subverting the windows kernel*. Upper Saddle River, NJ: Addison-Wesley.

Kawaguchi, N., Azuma, Y., Ueda, S., Shigeno, H., & Okada, K. (2006). ACTM: Anomaly connection tree method to detect silent worms. In *Proceedings of the 20th IEEE International Conference on Advanced Information Networking and Application* (pp. 901-908).

Lewis, J. (2005). *McAfee virtual criminology report: North American study into organized crime and the Internet*. Retrieved April 24, 2006, from http://www.mcafeesecurity.com/us/local_content/misc/mcafee_na_virtual_criminology_report.pdf

Moore, D., Shannon, C., Voelker, G., & Savage, S. (2003). Internet quarantine: Requirements for containing self-propagating code. In *Proceedings of IEEE INFOCOM 2003*.

Nazario, J. (2004). *Defense and detection strategies against Internet worms*. Norwood, MA: Artech House.

Newsome, J., Karp, B., & Song, D. (2005). Polygraph: Automatically generating signatures for polymorphic worms. In *Proceedings of the 2005 IEEE Symp. on Security and Privacy* (pp. 226-241).

Northcutt, S., Zeltser, L., Winters, S., Fredrick, K., & Ritchey, R. (2002). *Inside network perimeter security: The definitive guide to firewalls, vpns, routers, and intrusion detection systems*. Indianapolis, IN: New Riders.

Riordan, J., Wespi, A., & Zamboni, D. (2005). How to hook worms. *IEEE Spectrum*, *42*(5), 32-36.

Simkhada, K., Tsunoda, H., Waizumi, Y., & Nemoto, Y. (2005). Differencing worm flows and normal flows for automatic generation of worm signatures. In *Proceedings of IEEE International Symp. on Multimedia*.

Skoudis, E. (2004). *Malware: Fighting malicious code*. Upper Saddle River, NJ: Prentice-Hall PTR.

Spitzner, L. (2003). *Honeypots: Tracking hackers*. Boston, MA: Pearson Education.

Szor, P. (2005). *The art of computer virus research and defense*. Upper Saddle River, NJ: Addison-Wesley.

Turner, D., Entwisle, S., Friedrichs, O., Ahmad, D., Blackbird, J., & Fossi, M. (2006). *Symantec Internet security threat report: Trends for July 2005-December 2005*. Retrieved April 24, 2006, from http://www.symantec.com.

Wildlist Organization International. (2006). Retrieved April 24, 2006 from http://www.wildlist.org/WildList/.

KEY TERMS

Antivirus: Software to detect viruses and worms, clean infected files, and prevent new infections.

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

Firewall: A device or software to selectively filter packets.

Intrusion detection system: A device or software to detect suspicious or malicious activities.

Malware: Software intended to perform a malicious action.

Rootkit: Low-level software designed to avoid detection on a compromised host.

Spyware: A type of malware that collects personal user information and transmits to a remote attacker.

Trojan horse: A type of malware with a hidden malicious function.

Virus: A type of self-replicating malware that infects other files or programs.

Vulnerability: A security weakness in operating system or application software.

Worm: A standalone program capable of automated replicating itself through a computer network.