Malicious Codes in Depth

by: Mohammad Heidari, 11/29/2004

http://www.securitydocs.com/library/2742

Dedicated to my Grand Master - Hemmatabadi – The fine man Who left me too soon, He is truly missed.

The art of war teaches us to rely not on the likelihood of the enemy's not coming but on our own readiness to receive him, not on the chance of he is not attacking, but rather on the fact that we have made our position unassailable.

- "The Art of War" – Sun Tzu

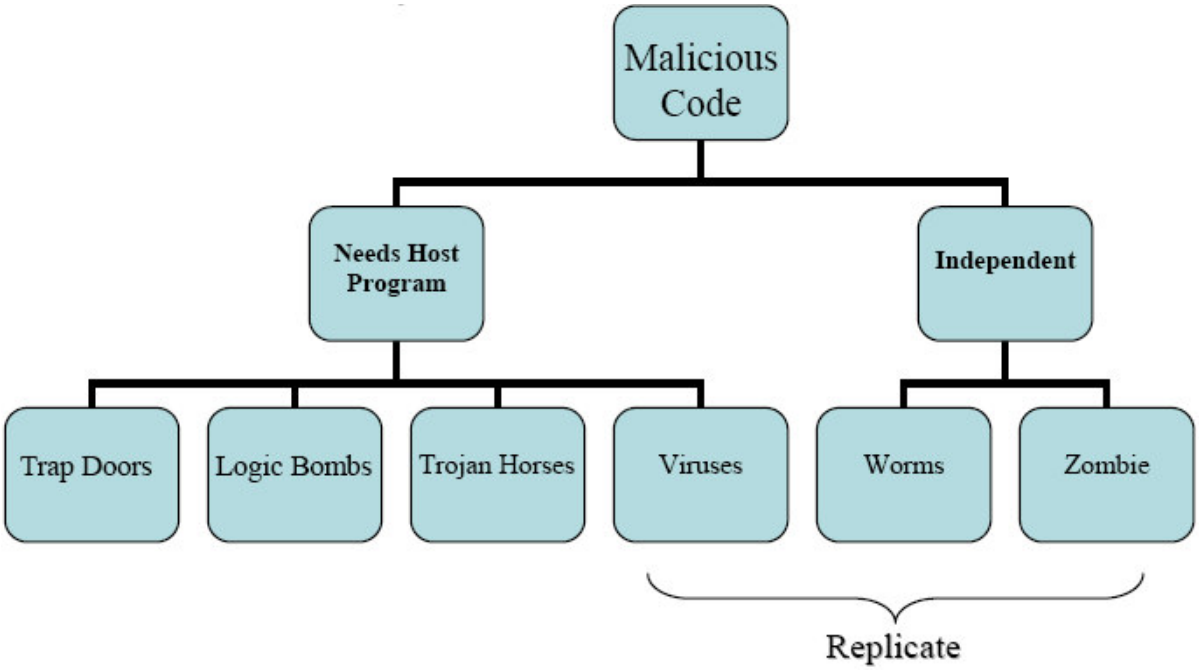
Abstract

Malicious code refers to a broad category of software threats to your network and systems. Perhaps the most sophisticated types of threats to computer systems are presented by malicious codes that exploit vulnerabilities in computer systems. Any code which modifies or destroys data, steals data , allows unauthorized access Exploits or damage a system, and does something that user did not intend to do, is called malicious code. This paper will briefly introduce you to the various types of malicious code you will encounter, including Viruses, Trojan horses, Logic bombs and Worms.

Taxonomy of malicious Code

A computer program is a sequence of symbols that are caucused to achieve a desired functionality; the program is termed malicious when their sequences of instructions are used to intentionally cause adverse affects to the system. In the other words we can't call any "bug" as a Malicious Code. Malicious codes are also called programmed threats. The following figure provides an overall taxonomy of Malicious Code.

Figure 1 Malicious Code Taxonomy



Taxonomy is a system of classification allowing one to uniquely identify something. As presented in the above figure, threats can be divided into two categories:

- Independents: are self contained program that can be scheduled and ran by the operating system.
- Needs host program: are essentially fragments of programs that can not exist independently of some actual application program, utility or system program.

You must also differentiate between these software threats that do not replicate and these that do. (Replication is a process that a code reproduces or duplicates itself.)The former are fragments of programs that are to be activated when the host program is invoked to perform a specific function , the latter consist of either a program fragment or an independent program (worm , zombie) that when executed may produce one or more copies of itself to be activated later on the same system or some other system . In the following, I briefly survey each at these parts of malicious software.

Trap doors

defined - 1.syn.Back doors a bad thing. 2. A Trap door function is one which is easy to compute but very difficult to compute the inverse of [Jargon Dictionary]
A trap door is a secret entry point into a program that allows someone that is aware at the trap door to gain access without going through the usual security access procedure. In many cases attacks using trap doors can give a great degree of access to the application, important data, or given the hosting system. Trap doors have been used legitimately by programmers to debug and test programs, some of the legitimate reasons for trap doors are:

1. Intentionally leaves them for testing, and make testing easier.
2. Intentionally leaves them for covert means of access. In the other words, allows access in event of errors.
3. Intentionally leaves them for fixing bugs.

But they may use illegitimately, to provide future, illegal access. Trap doors become threats when they are used by unscrupulous programmers to gain unauthorized access.

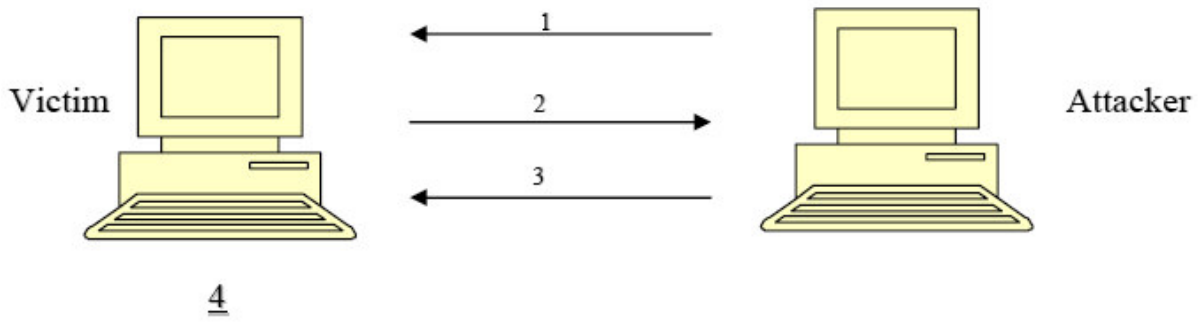
Back door is another name for a trap door, back doors provide immediate access to a system by passing employed authentication and security protocols, Attackers can use back doors to bypass security control and gain control at a system without time consuming hacking.

Logic Bombs

defined - The logic bomb is code embedded in some legitimate program that execute when a certain predefined events occurs, these codes surreptitiously inserted into an application or operating system that causes it to perform some destructive or security – compromising activity whenever specified conditions are met [Jargon Dictionary]

A bomb may sent a note to an attacker when a user is logged on to the internet and is using an specific program such as a word processor, this message informs the attacker that the user is ready for an attack, figure 2 shows a logic bomb in operation .Notice that this bomb dose not actually begin the attack but tells the attacker that the victim has met needed state for an attack to begin.

Figure 2 Logic Bombs



1. Attacker implants logic bomb
2. Victim reports installation
3. Attacker sends attack message
4. Victim dose as logic bomb installation

Examples of conditions that can be used as triggers for a logic bomb are the presence or absence at certain files, a particular day of the week or date, or a particular user running the application. One triggered a bomb may alter or delete data or entire files, cause a machine half or do some other damage.

Trojan Horses

defined - A malicious, security –breaking program that is disguised as something benign, such as directory lister, archiver, game, or (in one notorious 1990 case on Mac) a program to find and destroy viruses!" [Jargon Dictionary]

A Trojan horse is a useful, or apparently useful program or command procedure containing hidden code that when invoked performs some unwanted or harmful function. Trojan Horses can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly. For example, to gain access to the files of another user on a shared system, a user could create a Trojan Horse program that when executed, changed the invoking user's file permissions so that the file are readable by any user, the another example of Trojan horse program is a compiler that has been modified to insert additional code into certain programs as they are compiled such as a system login program, the code creates a trap door in the login program that permits the author to log on to the system using a special password. Another common motivation for the Trojan horse is data destruction.
The program appears to be performing a useful function but it may also be quietly deleting the victim's files.

Zombie

A zombie is a program that secretly takes over another internet-attached computer and then uses that computer to launch attacks that are difficult to trace to the zombie's creator. Zombies are used in Denial of service attacks, typically against targeted web sites. The zombie is planted on hundreds of computers belonging to unsuspecting third parties and then used to overwhelm the target website by launching on overwhelming onslaught of internet traffic.

Viruses

defined - [From the obvious analogy with biological viruses]. A cracker program that searches out other programs and 'infects' them by embedding a copy of itself in them so that they become Trojan horses. When these programs are executed, the embedded virus is executed too, thus propagating the ' infection ' this normally happens invisibly to the user. Unlike a worm, a virus can not infect other computers without assistance. It is propagated by vectors such as humans trading programs with their friends the virus may do nothing but propagate itself and then allow the program to run normally. Usually, however, after propagating silently for a while, it starts doing things like writing cute messages on the terminal or playing strange tricks with

the display. Many nasty viruses, written by particularly perversely minded crackers, do irreversible. Damage, like nuking the entire user's files… [Jargon Dictionary]

A virus is a program that can ' infect ' other programs by modifying them , the modification include a copy of the virus program , which can then go on to infect other programs . Therefore the key characteristic of virus is the ability to self replicate by modifying a normal program file with a copy of itself. On Nov, 1983 Fred Cohen ("father of computer virus") thought of the idea of computer viruses as a graduate student at USC. Cohen wrote the first documented virus and demonstrated on the USC campus network. "Virus" named after biological virus the following table shows details:

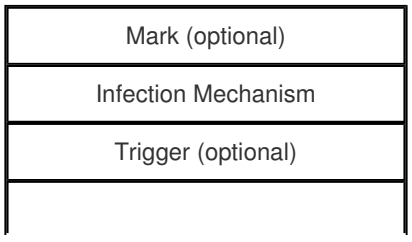
Biological Virus	**Computer Virus**
Consist of DNA or RNA strand surrounded by protein shell to bond to host cell	Consist of set of instructions stored in host program
No life outside of host cell	Active only when host program is executed
Replicates by taking over host's metabolic machinery with it's own DNA/RNA	Replicates when host program is executed or host file is opened
Copies infect other cells	Copies infect (attach to) other host program

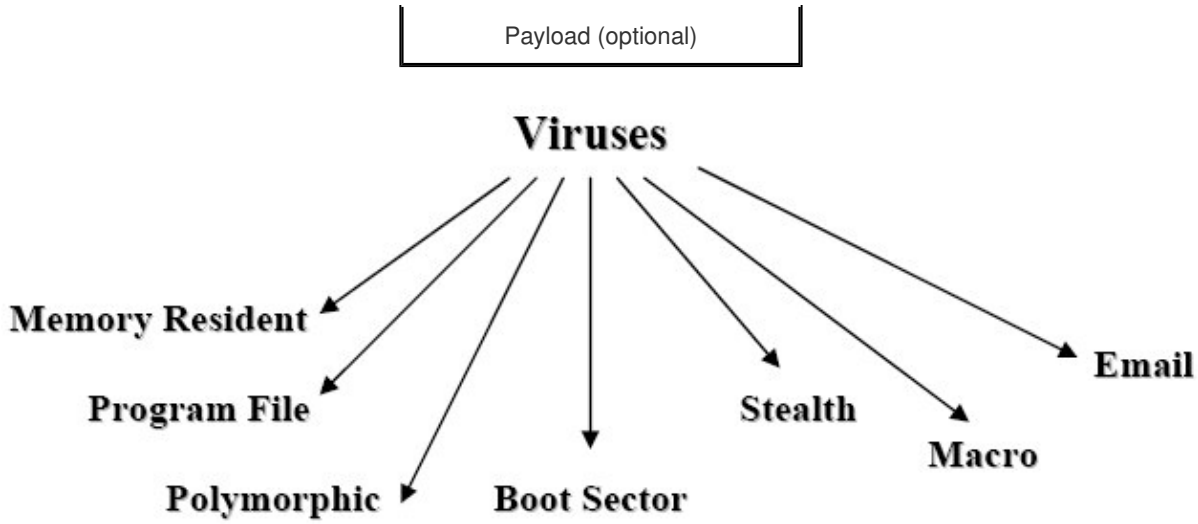
A virus can do anything that other programs do. The only difference is that it attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function such as erasing files and programs. During its lifetime a typical virus goes through the following four phases:

- Dormant phase: The virus is idle the virus will eventually be activated by some event, such as a date. The presence of another program or file, or the capacity of the disk exceeding some limit, not all viruses have this stage.
- Propagation phase: The virus places an identical copy of itself into other programs or into certain system areas on the disk. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.
- Triggering phase: The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.
- Execution phase: The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.

Virus Anatomy
Virus Structure has four ports
Mark can prevent re-infection attempts
Infection Mechanism causes spread to other files
Triggers are conditions for delivering payload
Payload is the possible damage to infected computer

Figure 3 Anatomy of Virus

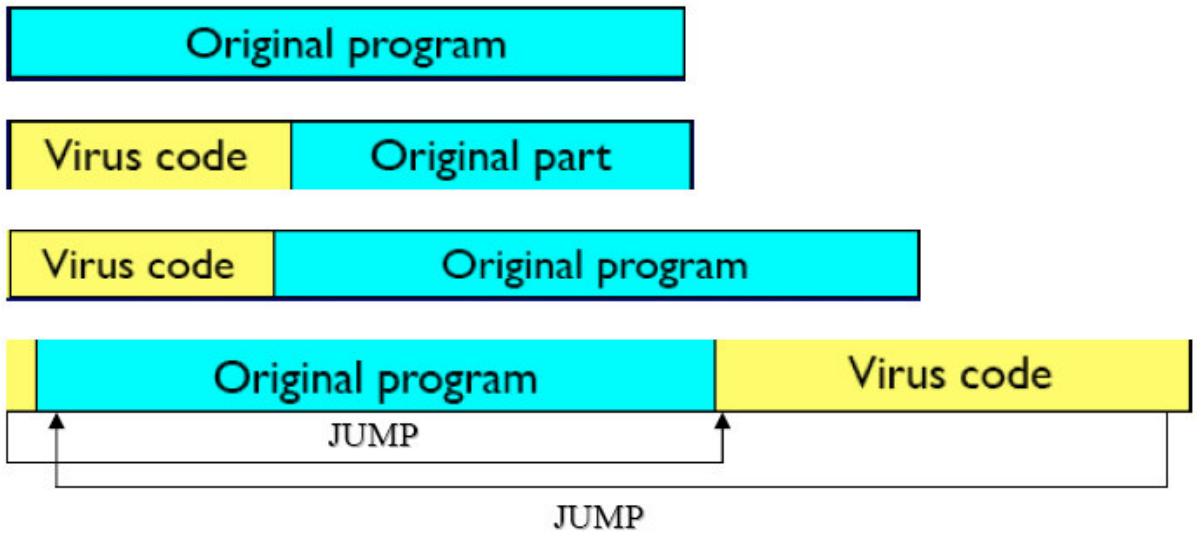




Memory – resident virus: lodges in main memory as part of a resident system program. From that point on, virus infects every program that executes.

Program file virus: Infects programs such as Exe/Com/Sys – files. The following figures show details:

Figure 5 Program File Viruses



Polymorphic virus: creates copies during replication that are functionally equivalents but have distinctly different bit patterns. In this case the “signature “of the virus will vary with each copy. To achieve this variation, the virus may randomly insert superfluous instructions or interchange the order of independent in-generally called a mutation engine, creates a random encryption key to encrypt the reminder of the virus. The key is stored with the virus, and the mutation engine itself is altered. When an infected program is invoked, the virus uses the stored random key to decrypt the virus, when the virus replicates, a different random key is selected.

Boot Sector Virus: Boot sector viruses infect the system area of the disk that is read when the disk is initially accessed or booted. This area can include the master boot record the operation system’s boot sector or both. A virus infecting these areas typically takes the system instructions it finds and moves them to some other area on the disk. The virus is then free to place its own code in the boot record. When the system initializes, the virus loads into memory and simply points to the new location for the system instructions. The system then boots in a normal fashion except the virus is now resident in memory. A boot sector virus can replicate without your executing any programs from an infected disk. Simply accessing the disk is sufficient.

For example, most PCs do a systems check on boot up that verifies the operation of the floppy drive even this verification process is sufficient to activate a boot sector virus if one exist on a floppy left in the machine and the hard drive can also become infected.

Stealth Virus: A format virus explicitly designed to hide itself from detection by antivirus software. When the virus is loaded into memory, it monitors system calls to files and disk sectors, when a call is trapped the, virus modifies the information returned to the process making the call so that it sees the original uninfected information. This aids the virus in avoiding detection. For example many boot sector viruses contain stealth ability. If the infected disk is booted, programs such as FDISK report a normal boot record. The virus is intercepting sector calls from FDISK and returning the original boot sector information. If you boot the system from a clean floppy disk however, the drive is inaccessible. If you run FDISK again, the program reports a corrupted boot sector on the drive. To use stealth, however, the virus must be actively running in memory, which means that the stealth portion of the virus is vulnerable to detect by antivirus.

Macro Virus: it is set of macro commands, specific to an application, which automatically executes in an unsolicited manner and spread to that application's documents. According to the national computer security agency (www.ncsa.com), macro viruses now make up two – thirds of all computer viruses. Macro viruses are particularly threatening for a number of reasons:

1. A macro virus is platform independent. Virtually all of the macro viruses infect Microsoft word documents. Any hardware platform and operating system that supports word can be infected.
2. Macro viruses infect documents, not executable portions of code. Most of the information introduced on to a computer system is in the form of a document rather than a program.
3. Macro viruses are easily spread. A very common method is by electronic mail.

Macro viruses take advantage of a feature found in word and other office applications such as Microsoft Excel, namely the macro. In essence, a macro is an executable program embedded in a word processing document or other type of file. What makes it possible to create a macro virus is the auto executing macro this is a macro that is automatically invoked, without explicit user input. Common auto execute events are opening a file, closing a file and starting an application. Once a macro is running, it can copy itself to other documents, delete files and cause other sorts of damage to the users In Microsoft word. There are three types of auto executing macros:

1. Auto execute: If a macro named Auto exec is in the "Normal. Dot" template or in a global template stored in word's start up directory, it is executed whenever word is started
2. Auto macro: An auto macro executes when a defined event occurs, such as opening or closing a document
3. Command macro: If a macro in a global macro file or a macro attached to a document has the name of an existing word command, it is executed whenever the user invoked that command.

A common technique for spreading a macro virus is as follows:
An auto macro or command macro is attached to a word document that is introduced into a system by e-mail or disk transfer. After the document is opened, the macro executes. The macro copies itself to the global macro file. When the next session of word opens, the infected global macro is active. When this macro executes, it can replicates itself and cause damage.

Email Virus: A more recent development in malicious software is the e-mail virus. The first rapidly spreading e-mail viruses, such as Melissa, made use of a Microsoft word macro embedded in an attachment. If the recipient opens the e-mail attachment, the word macro is activated then:

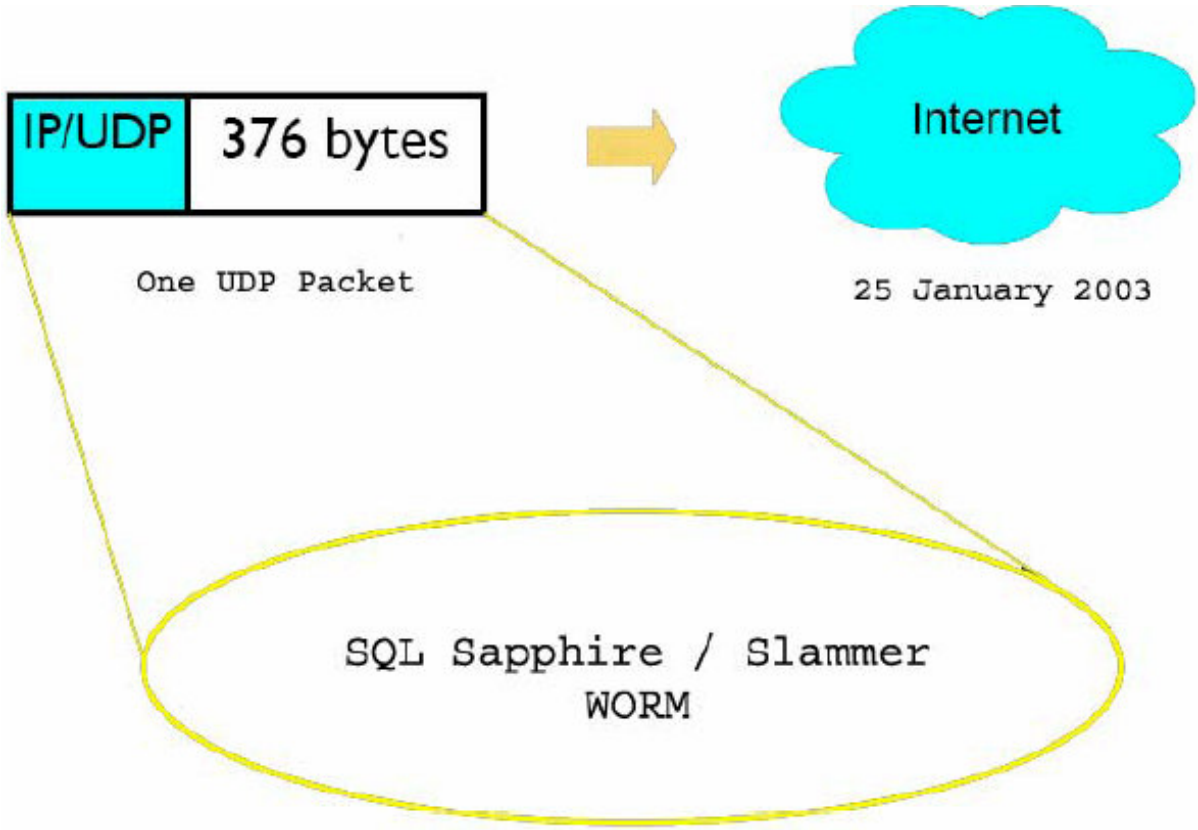
1. The e-mail virus sends itself to everyone on the mailing list in the user's e-mail package

2. The virus does local damage

Worms

Can one IP packet cripple the Internet within 10 minutes? On January 25Th 2003 “SQL Sapphire Slammer “worm causes more than 1.2 billion US dollars damage, 70% South Korea’s network paralyzed, 300,000 ISP subscribers in Portugal knocked offline, 13,000 Bank of America machines shut down, Continental Airline’s ticketing system crippled.

Figure 6 SQL Sapphire / Slammer Worm

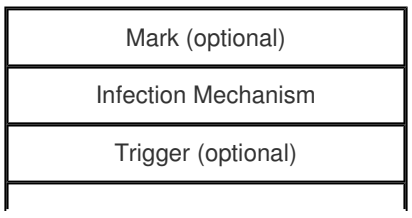


Worm (n)
[From ‘tape worm’ in John Brunner’s novel “The Shockwave Rider “…], A program that propagates itself over a network, reproducing itself as it goes … [Jargon Dictionary]

Worm is also self-replicating but a stand-alone program that exploits security holes to compromise other computers and spread copies of itself through the network. Unlike viruses, worms do not need to parasitically attach to other programs. Because of the recursive structure of this propagation, the spread rate of worms is very fast and poses a big threat on the Internet infrastructure as a whole.

Worm Anatomy
Mark: structurally similar to viruses, except a stand-alone program instead of program fragment
Infection Mechanism: searches for weakly protected computers through a network (i.e., worms are network based)
Triggers: are Conditions for delivering payload
Payload: might drop a Trojan horse or parasitically infect files, so worms can have Trojan horse or virus characteristics

Figure 7 Worms Anatomy



Payload (optional)

Propagation Carriers and Distribution Mechanism

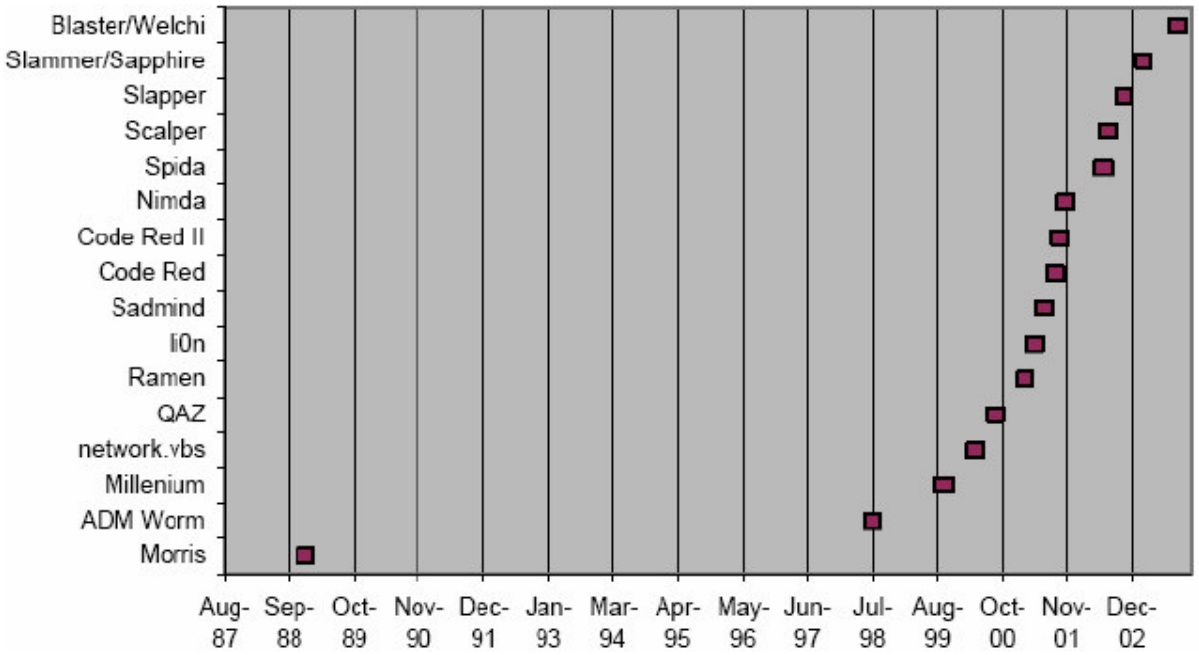
The means by which propagation occurs can also affect the speed and stealth of a worm. A worm can either actively spread itself from machine to machine, or it can be carried along as part of normal communication.

Self-Carried: A self-carried worm actively transmits itself as part of the infection process. This mechanism is commonly employed in self-activating scanning or topological worms, as the act of transmitting the worm is part of the infection process. Some passive worms, such as CRClean, also use self-carried propagation.

Second Channel: Some worms, such as Blaster, require a secondary communication channel to complete the infection. Although the exploit uses RPC, the victim machine connects back to the infecting machine using TFTP to download the worm body, completing the infection process.

Embedded: An embedded worm sends itself along as part of a normal communication channel, either appending to or replacing normal messages. As a result, the propagation does not appear as anomalous when viewed as a pattern of communication. The contagion strategy is an example of a passive worm that uses embedded propagation. An embedded strategy, although relatively stealthy, only makes sense when the target selection strategy is also stealthy. Otherwise, the worm will give itself away by its target selection traffic, and reaps little benefit from the stealth that embedded propagation provides. Thus a scanning worm is unlikely to use an embedded distribution strategy, while passive worms can benefit considerably by ensuring that distribution is as stealthy as target selection.
The speed at which embedded worms spread is highly dependent on how the application is used, as is how far from the natural patterns of communication such a worm could deviate in order to hasten it's propagation without compromising its stealth ness.

Figure 8 Worm Highlights



In the rest of this section I try to introduce you with the popular worms.

The Internet Worm (November 3rd, 1988)
The Internet Worm is believed to be the first computer worm ever created. It was released in November 1988 when World Wide Web even did not exist. The worm had taken advantage of lapses in security on systems that were running 4.2 or 4.3 BSD UNIX or derivatives like SunOS. Since the IP address space was too sparse to be scanned at that time the worm retrieved addresses of possible victims from files like /etc/hosts.equiv, /.rhosts, .forward, and .rhosts. Worm could penetrate a remote system by any of three ways: exploit a bug in the finger server that allowed it to download code in place of a finger request and trick the server into executing it, the worm could use a sendmail SMTP mail service, exercising a bug in the debugging code that allowed it to execute a command interpreter and download code across a mail connection. The worm could guess a password for user accounts. In any case the worm arranged to run a remote shell that it could use to copy, compile, and execute the 99-line bootstrap. The bootstrap set up its own network connection with the local worm and copied over the rest of the files it needed. The worm built itself from these files and the infection procedure started over again. The worm itself has a bug that made it create many copies of itself on machines it infected, which quickly used up all available processor time on those systems. Nevertheless, the Internet worm was sophistically designed. For example its crypt function was executed 9 times faster than the native UNIX crypt routine, it removed all its files from the disk and prevented core dumps to complicate its disassembling. The worm quickly distributed itself to over 6000 computers and has started the era of Internet worms. Since several different propagation methods were used the first worm was the first multi-vector worm at the same time.

Code Red I (July 13th, 2001)
Initial version of the Code Red worm was first seen in the wild on July 13th, 2001 the worm spread by compromising Microsoft IIS web servers using the .ida Vulnerability discovered almost a month before that. Once it infected a host, Code-Red spread by launching 99 threads that generated random IP addresses, and then tried to compromise those IP addresses using the same vulnerability. However, the first version of the worm had an apparent bug. The random number generator was initialized with a fixed seed, so that all copies of the worm in a particular thread, on all hosts, generated and attempted to compromise exactly the same sequence of IP addresses. Thus, first version of Code Red worm had a linear spread and never compromised many machines. On July 19th, 2001, a second version of the worm began to spread. This version is known as Code Red I. Code Red I. has the same code base as its first version in almost all respects. The only differences were fixing the bug in the random number generator and a DDoS payload targeting the IP address of www.whitehouse.gov. Theoretical analysis of the spread based on the random IP range scanning is presented later in the text.

Code Red II (August 4th, 2001)
The Code Red II worm was released on Saturday August 4th, 2001 and spread rapidly. Despite a comment string “Code Red II,” found in the worm body it is a differently coded worm. It uses the same vulnerability, however. When successful, the payload installed a root backdoor allowing unrestricted remote access to the infected host. The worm was also a randomly scanning worm that chose IP addresses and tried to infect the corresponding host. The innovation was that it used a localized scanning strategy. In particular the worm attempted to infect hosts with addresses closer to the scanning host with higher probability. It chose randomly from its own class A (/8 network) with probability 1/2, with probability 3/8 it chose from the class B network (/16 network), and with probability 1/8 it would choose a random address from the whole Internet. This strategy appeared quite successful because the infection often proceeds faster since hosts with similar IP addresses are often close in the network topology and thus have better connectivity. In addition, this strategy allows the worm effectively infect hosts behind a firewall once it manages to get into the local network.

Nimda (September 18th, 2001)
Nimda is another example of a multi-vector worm. Nimda started to spread On September 18th, 2001 and remained active for months after it started. Nimda spread extensively behind firewalls because it is believed to have used at least five different methods to spread itself:

1. by infecting Web servers from infected client machines via active probing for a Microsoft IIS vulnerability
2. by bulk emailing of itself as an attachment based on email addresses determined from the infected machine
3. by copying itself across open network shares
4. by scanning for the backdoors left behind by Code Red II and also the “sadmind” worm,
5. by adding malicious code to the web pages on compromised machines.

Onset of Nimda was quite rapid, rising in just half an hour from essentially no probing to a sustained rate of nearly 100

probes/sec. The worm's multi-vector nature helped it to effectively propagate behind firewalls. For example, most firewalls allow mail to pass inside, relying on the mail servers to remove pathogens. Yet since many mail servers remove pathogens based on signatures, they aren't effective during the first few minutes to hours of an outbreak.

Benjamin (May 18th, 2002)
Benjamin is a typical P2P worm that offers itself for download with different file names, file types, and file lengths through the KaZaA network. In particular, the worm had more than 2000 different file names to use and padded the files with garbage bytes. In a departure from many other viruses and worms, 'Benjamin' may have had a commercial motivation. The worm opens a Web page named "benjamin.xww.de" which contained advertisements.

Slapper (September 13th, 2002)
Slapper spread on Linux machines by using a flaw discovered in OpenSSL libraries in August 2002. The worm was found in Eastern Europe late on Friday September 13th 2002. The worm scanned for potentially vulnerable systems on 80/tcp using an invalid HTTP GET request. When a potentially vulnerable Apache system was detected, the worm attempted to connect to the SSL service in order to install the exploit code. Once infected, the victim server started scanning for additional hosts to continue the worm's propagation. The worm constructed a distributed P2P network. In particular, newly infected hosts were instructed to maintain connection via a set of UDP ports. The network could act as a platform for Distributed Denial of Service (DDoS) attacks against other sites. Infected hosts shared information on other infected systems as well as attack instructions. Thus, an attacker could control a distributed network of subverted hosts by connecting to any of the participating nodes.

SQLslammer/Sapphire (January 25th, 2003)
The SQL slammer (a.k.a. Sapphire) worm was the fastest computer worm ever. The number of infected hosts doubled every 8.5 seconds. The worm infected more than 90 percent of vulnerable hosts within 10 minutes. The worm exploited buffer overflow vulnerability in computers running Microsoft's SQL Server or MSDE 2000. The weakness in an underlying indexing service was discovered almost a year before the worm outbreak and Microsoft released patch for the vulnerability. The worm infected at least 75,000 hosts, perhaps considerably more, and caused network outages worldwide. Sapphire spread was nearly two orders of magnitude faster than the spread of Code Red. Both worms used the same basic strategy of random scanning to find vulnerable machines and then transferring the exploitive payload; they differed in their scanning constraints. The Code Red was latency limited; SQLslammer was bandwidth-limited and was sending infection probes at the maximum speed possible with the available network connectivity.
SQLslammer's size was 376 bytes - so small, that even with all the packet headers, the payload was only a single 404-byte UDP packet. This can be contrasted with the 4kb size of Code Red, or the 60kb size of Nimda. Previous scanning worms spread via many threads, each invoking connect () to probe random addresses. Therefore, each thread's scanning rate was limited by network latency. In particular, the time required transmitting a TCP SYN packet and waiting for a response or timeout. Worms can compensate this latency by invoking many threads. However, context switch overhead is significant and there are insufficient resources to create enough threads to counteract the network delays. As a result the worm becomes latency dependent again. In contrast, SQlslammer's scanner was limited by each compromised machine's bandwidth to the Internet. Since the SQL Server vulnerability was exploitable using a single packet sent to UDP port 1434; the worm was able to send these scans without requiring a response from the potential victim. Fortunately, SQLslammer worm had a bug in its random number generator that left considerable portion of the Internet hosts not scanned.

Blaster (a.k.a. Lovesan) worm (August 11th, 2003)
The worm exploits the buffer overflow vulnerability in the Distributed Component Object Model (DCOM) Remote Procedure Calls (RPC) interface that allows arbitrary code to be executed on most of the Windows NT, Windows 2000, and Windows XP platforms. Fortunately, the worm is designed very poorly. Firstly, its scanning rate is very small and the worm itself is latency-limited. Therefore, every machine was probed only once in a half an hour on average during the peak of the epidemics. Secondly, the worm has a bug that forced many of the machines to endlessly reboot thus reducing the number of the scanning hosts. The worm has a payload to create a SYN DDoS attack against Windows update sites. The Blaster worm showed that the auto update functionality provided by Microsoft is quite successful. Despite the fact that at the time the bug was discovered almost all of the PCs were vulnerable just three weeks later when the worm started to spread only half a million of the machines were subverted.
Another important lesson from the Blaster worm epidemics is that even the machines that are not patched by the worm outbreak time are soon patched and only one or at most two worms that share the same vulnerability have a chance for

widespread.

Welchia worm (August 19th, 2003)
Welchia worm raise a question whether an Internet worm can be good. The worm exploits the same Microsoft DCOM RPC vulnerability as the Blaster worm in addition to the MS03-007 vulnerability in the Microsoft IIS by randomly scanning the IP address space. Surprisingly, the payload of the worm cleans the system from the Blaster worm, downloads the patch for the DCOM RPC bug and patches the system. The worm contains a code to remove itself from the host PC in January 2004. Despite the fact that the worm itself is believed not to contain any malicious code and on contrary cures the infected systems security experts worldwide still treat it as a malicious worm because it installs itself without permission form the user and resides in memory until 2004 in addition to overloading the networks with the probing and patch downloading traffic. Nevertheless, it seems likely that Welchia will cure all the systems infected with the Blaster worm within several days.

Top 2004 Worms

MyDoom
spreads by e-mail to Windows PCs, searches for e-mail addresses in various files, opens backdoor for remote access.

Netsky
spreads by e-mail, exploits Internet Explorer to automatically execute e-mail attachments, and removes MyDoom and Bagle from PCs.

Bagle
spreads by e-mail, tries to remove Netsky from PCs, and opens backdoor for remote access, downloads code updates from Web, disables antivirus and firewall software.

Spywares

A definition of Spyware provided by Steve Gibson states "Spyware is ANY SOFTWARE which employs a user's Internet connection in the background (the so-called "back channel" connection MUST BE PRECEDED by a complete and truthful disclosure of proposed back channel usage, followed by the receipt of explicit, informed, consent for such use.

Any Software communicating across the Internet absent these elements is guilty of information theft and is properly and rightfully termed: "Spyware". The term Spyware, in most cases, is synonymous with Adware, and is potentially a Trojan horse program. Spywares can collect the sensitive data (such as the version of Operating System you are running, Browser type, is scripting enabled, what version of Java you are running, Screen size, Available plug-ins, DNS information from your current domain, Run a trace route back to you to find out where you live on the net.) by two varieties of ways:

- By cookies
- By install itself and then execute

By Cookies
Cookies are small files that are placed in your system by web servers when you visit, and can track and record your internet usage. Each time you visit a site, the site checks to see if you have a cookie for that site, if you do then they retrieve your personal settings for the site, if not they deliver a cookie to your machine. Cookies come in a couple different flavors. **Persistent cookies**, which are configured to stay on your system for many years using an expiration date, or **"session cookies"** that are removed when the session is closed.

By install itself and then execute
Spyware typically is an independent program that runs in the background. Programmers working for Spyware distributing companies can write a routine that can run with system privileges and retrieve information from your computer. If they want to retrieve Word documents from their targets, then they write code that looks for word documents and sends them back to the proper place on the internet.

Conclusion

Malicious Code "Study in Depth" provides a layered approach to securing information and resources, as well as maintaining confidentiality, integrity, and availability of these resources. Viruses / Worms are consistently among most common attacks. In this paper I explain Malicious Codes, including Trap doors, Trojan horses, and Logic bombs, Zombie, Viruses, Worms and Spywares.

Acknowledgement

I am grateful to my brother “Vahid” for his efforts in editing this paper.

References

D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N.Weaver. Inside the Slammer Worm. *IEEE Security and Privacy*, July 2003.

C.C. Zou, L. GAO, W. Gong, and D. Towsley. Monitoring and Early Warning for Internet Worms. In *10th ACM Symposium on Computer and Communication Security*, Washington DC, 2003.

CERT, “Code Red: Worm Exploiting buffer Overflow in IIS Indexing Service DLL,” Incident Note IN-2001-8, July 19, 2001

CERT, “Code Red II: Another Worm Exploiting buffer Overflow In IIS Indexing Service DLL,” Incident Note IN-2001-9, August 6, 2001

Siliconvalley, “'Benjamin' Worm Plagues KaZaA,” http://siliconvalley.internet.com/news/article.php/3531_1141841

CERT, “Apache/mod_ssl Worm”, Advisory CA-2002-27, Sep 14, 2002

US Department of Homeland Security, “Potential for Significant Impact on Internet Operations Due To Vulnerability in Microsoft Operating Systems,” Advisory, July 30, 2003, http://all.net

Symantec. W32.Benjamin.Worm http://securityresponse.symantec.com/avcenter/venc/data/w32.benjamin.worm.htm

Arce, Ivan and Elias Levy. “An Analysis of the Slapper Worm.” http://www.coresecurity.com/files/files/12/AttackTrends.pdf