

How Viruses, Worms and Trojans Work

In order to deal with viruses, worms and trojans, you have to understand them. This chapter gives you the background you need. First, I define viruses and other types of malevolent software such as worms and trojans. Then I provide a brief history of their evolution in order to give you some perspective. In doing so, I describe how they work. Finally, I discuss where they are likely headed in the near future. *

Spam gets all the attention, but it's not the only bad guy on the information superhighway. There is a category of software (popularly referred to as viruses) that can cause a wide variety of problems, from nuisance style interferences to severe damage and theft. A computer virus, however, is actually just one type of malicious software; other types include worms and trojans, and so I'll use the more correct and generic term of 'malware'. In the past, malware was transmitted through files downloaded from bulletin boards or via floppy disks exchanged between individual users, and its existence prompted the creation of 'anti-virus' software.

Over the past few years, however, email has become the favored transmission method due to its pervasive nature and its flexibility. Thus, all email users need to be knowledgeable about how malware works. In Chapter 9, I'll discuss how to deal with malware on the user level, and in Chapter 14, I'll address the same issue from the administrator's point of view.

Viruses, worms and trojans: definitions & how they work

Before launching into a discussion of how viruses work, it's appropriate to define terms. There are actually three general types of malware, of which viruses are one. Let's take a look at each type of malware and discuss how they work in turn.

Viruses

A computer virus is a small computer program that is embedded in a larger, legitimate program. The virus is designed so that when a user executes the legitimate program, the virus executes first, and when it's finished doing whatever it was put together to do, the original program runs, without the user ever being aware that a virus was put into play.

A virus's primary task is to make a copy of itself by infecting another program, but viruses often perform other tasks, from displaying obnoxious messages to damaging or destroying information on the computer's storage media.

These types of programs were named 'viruses' in 1983 when USC doctoral candidate Fred Cohen noticed that, like biological viruses, they needed a host to live on, and infected other hosts by making copies of themselves (the technical term is "self-replicating"). Like biological viruses, they modify the host program that they reside in. He used the term "computer virus" to describe a computer program that 'can affect other computer programs by modifying them in such a way as to include a (possibly evolved) copy of itself.'

A virus can be either a file infecter or a system (boot-record) infecter.

File infecters

File infecters attach themselves to programs that users would use during their daily work. Some file infecters select one or more programs to infect each time the original infected program is run (these are called 'direct-action' while other file infecters (called 'resident') install themselves into random access memory when the original infected program is run, and then wait to infect other programs when they are executed.

System infecters

System infecters don't attack ordinary programs; rather, they target specific files and portions of computer disks that are used when a computer is turned on or rebooted. Some system infecters attach themselves to the Master Boot Record (MBR) of a hard disk while others only infect the DOS boot sector of hard disks and floppy disks.

Some viruses do both - infect both ordinary files and system files. These are called "multipartite".

File system infecters

Another category of viruses are File System (or Cluster) viruses that modify the directory table entries. Thus, when a user runs a specific program, the modified directory entry actually points to a different location, where the virus is - and the virus executes, and then runs the program that the user originally desired. This is sort of like when someone asks you to pass the salt at the dinner table - and you stop use it yourself first before passing it on.

Virus attributes

Each of these viruses has one or more attributes.

A stealth virus is one that hides the modifications it has made. For example, one common modification is to change the size of a file, since a file infector adds code to an existing program. This hiding of the change of file size is usually done by monitoring system calls to functions that would report such changes, and substituting false information. For example, suppose a virus changed the size of MANAGER.EXE from 26,578 bytes to 28,200 bytes. If that virus was also a stealth virus, it would intercept commands like "DIR" so that a file listing would still report a length of 26,578 bytes for MANAGER.EXE.

A polymorphic virus is one that creates different versions of itself. The purpose of this functionality is to create many versions in the hopes that at least some of the versions will evade identification by anti-virus scanners that are keyed to look for specific versions of viruses.

Another analogy to biological viruses is worth noting: just as genetic diversity in a population decreases the chance of a single disease wiping out a population, the diversity of software systems on a network similarly limits the destructive potential of viruses. The popularity of Windows on the desktop has caused an increase in the possibility of a single virus causing significant harm to the computer using population. If there was a wide diversity in operating systems used on the desktop, it would be nearly impossible for a single virus to cause widespread damage.

Armoring is a technique used in writing viruses so that anti-virus programmers can't take the virus apart in order to determine how it works and what its intended functionality is.

A companion virus is one that creates a new program with the same name as an existing program. When a user attempts to run the original program, the new, virus-laden program is run instead. When the virus-laden program has finished executing, it will typically then run the original program so as to not alert the user that anything untoward has occurred. On IBM compatible PCs, programs with .COM extensions are run before identical named programs with .EXE extensions. Thus, a virus could create a program named WORD.COM (or name itself WORD.COM), and it would then be run before WORD.EXE would.

A cavity virus is one that fills empty spaces in the host program's program code with itself so as to avoid detection.

An encrypted virus is written so that it can't be changed or removed from the computer it has infected.

A tunneling virus is one that calls system functions (interrupt handlers) in the operating system directly, thus bypassing anti-virus programs that may be loaded and are set up to provide the same functionality.

A dropper is a program that had been modified to install another program - the one containing the virus - on to the target computer. The dropper contains code that is designed so that it won't be detected by virus scanners that normally detect virus signatures. In other words, the dropper isn't infected with the virus. Droppers are rather uncommon, and are sometimes known as trojans (see the section on Trojans later in this chapter) whose payload is the action of installing a virus. Droppers which install a virus only in memory (without infecting files on the disk) are sometimes called "injectors."

Worms

A worm is similar to a virus in that it makes copies of itself, but it does not need to attach itself to a host (another computer program) in order to survive. Instead, when executed, it can simply replicate itself from machine to machine via a network. Like viruses, however, a worm can perform secondary actions ranging from merely nuisance level to highly destructive.

I've explained how a virus became named - what about worms? How were they named? When a computer is running more than one program, it is critical that each program stay out of the way of the other. This is done by allocating a certain area of the computer's memory to each program, and no other program was allowed to intrude in that area. Humans being fallible, some programmers inevitably created programs that didn't behave properly, and intruded on other programs' areas. When a programmer attempted to debug such a program, they would trace the affected memory locations on a piece of paper; the resulting diagrams looked like holes in a piece of wood infected by worms - irregular traces that wandered randomly throughout the wood.

As a result, these programs that infiltrated other programs' memory spaces became known as worms. In an early network of computers, one of these programs infiltrated not only the memory space of other programs on the same computer, but also other computers on the network. As a result, programs that can replicate themselves from one computer to another without the need of a host to harbor them are called worms.

There are two types of worms. Computer worms are programs that are entirely contained on one computer. When a computer worm replicates, the entire program is copied to another computer. Network worms, on the other hand, are programs that are made up of many parts,

and each of these parts can run on a different computer. Each of these parts (technically called segments) performs a single, discrete function, and communicates with the other segments via the network in order to provide the complete functionality of the worm.

A worm can replicate extremely quickly because of the multiplier effect of a network. Suppose a worm only infects two other machines (in real life, a worm on one machine would typically infect hundreds or thousands of machines.) The first infected machine infects two other machines, each of which can again infect two more machines. After only 16 rounds of infections, over 65,000 machines would be infected.

The recent spread of worms has caused considerable alarm in the computer community, as sophisticated techniques have caused worms to spread around the world and achieve significant penetration in under a day. The so-called "Warhol worm", a theoretical construct at this writing, has been projected to be able to cause nearly 100% infection of its target systems in 8 to 15 minutes.

A worm that can spread with this type of speed will bypass any type of human interaction, since it will be done before humans will be able to react.

Trojans

Trojan Horses (known as a 'trojan' for short) differs from viruses and worms in that they don't replicate themselves, relying on a separate mechanism for distribution. Instead, their primary feature is that they masquerade as a legitimate program or offer something desirable (such as a link for something free or interesting), but harbor a malevolent purpose.

Trojans, as you might suspect, are named after the famed "Trojan Horse" of ancient Greek history. In Homer's Iliad, the Greeks presented their opponents in the Trojan War, the citizens of Troy, a large wooden horse as a gesture of peace. The Trojans accepted the gift and brought it inside their city gates. At night, Greek soldiers who had hidden inside the horse emerged, opened the city gates from within, and let their fellow soldiers into the city in order to take it over. The ruse was successful and the Greeks captured Troy.

There are two types of trojans.

The first is a program or function that is contained in another program. The second is a program that completely replaces a legitimate program. In both cases, the program or function performs an operation that the programmer intended but that is not documented, and that the user would probably not approve of if they knew about it.

Because trojans don't replicate by themselves, they must be installed on a computer by an outside action, either manually, by a user (such as clicking on a link in an email) or by a separate program, such as an installer or a virus whose purpose is to install and/or execute other malware (a 'dropper').

Once a trojan is resident, it typically lies dormant until it is called or executed by an external event or internal timer. An external event might be a user running the program, or being called by a second program. For example, a trojan hidden in a spell checker could be activated either by a user explicitly running the spell checker, or by a separate text editing program calling that external spell checking program upon loading a new document.

An internal timer might look for any type of event, such as the system clock passing a specified date or time, or an external action occurring a certain number of times. For example, the AIDS Information trojan activated once the user had rebooted their machine 90 times.

Trojans are generally known for harboring different types of rotten deeds than viruses and worms. Sure, they can display nasty messages, delete files on your system, or send your passwords and documents to another party. But they can also open up your computer and give someone elsewhere on the Internet full access to your computer and your network, giving them access to your programs and data, allowing them to use your machine for nefarious purposes such as sending spam or executing DOS attacks on other computers - or even allow them to take control of your machine from you.

Hiding Strategies

A piece of malware that can be easily found is much less effective than one that can stay hidden. As a result, well written malware uses a variety of techniques to avoid detection. Common means of detection include looking for changes in file attributes, and looking for virus signatures.

The first virus detection technique to come into play was to compare the attributes of existing files to a database that contains the actual, well-known attributes. Viruses would alter the size and date attributes of a file during the process of infecting it. Those modifications were easily spotted as the files attacked were commonly used files whose attributes were widely known. (I still remember listening to warnings at PC User Group meetings in the mid 80' s about "If your COMMAND.COM file isn' t dated <something>, it's been infected!")

This technique for detecting malware has been around for a long time, so even older viruses that simply infected a file would often alter the information contained in the File Allocation Table about the file, such as the date last updated and the file size, setting those values back to the original, correct ones.

Another technique of evasion was to infect the boot sector of a computer instead of the data or program files on the hard disk. When the computer was turned on, the boot sector virus was loaded into memory, which would then load the original boot sector code as well, and transfer control to it. Thus, the operating system wasn' t aware of the presence of the virus since the virus was part of the environment for as long as the operating system had been running.

Another common means of detection is to look for a ' virus signature' - a pattern of code indicative of a virus - in a file or in memory. Once a virus has been found, it's examined for a pattern of code that's unique to itself. Then, files on a computer can be scanned by anti-virus software, using the signature as a key. If found, an appropriate action can be taken; appropriate being chosen by the user. Some users may want the anti-virus software to take care of the virus-laden file automatically, while other users may want to be warned but have the final control themselves. Additionally, some files can' t be deleted or repaired automatically without harming the computer, and anti-virus software will take those circumstances into effect and simply warn the user of the situation.

Evading the virus signature detection was more difficult than simply changing file attributes, but not impossible for reasonably skilled programmers. Modern malware (called "polymorphic") often modifies itself 'on the fly' so that it doesn' t contain an easily-identifiable

pattern that could be classified as a signature. This modification is done in one or both of two methods - encryption and self-modifying code.

Polymorphic viruses usually contain two pieces - the part that does the infection (the virus itself) and the engine that provides the algorithms that encrypt and modify the code in the infector. Each time the engine generates the infector, the algorithm inside the engine customizes the end-result so that there isn't a single standard signature in the virus file. The infector also has to include the engine as part of its payload, and so the engine needs to modify its own code as well, without affecting the core generation algorithms. Tricky stuff, but not impossible, and certainly fun for someone who enjoys a good technical challenge.

A (not so) brief history of malware

It's hard to imagine a time without viruses, but only about 20% of the time since computers were invented has malware been a problem. Theoretically, viruses and their brethren have been around since the late 1960s, but they only broke into the public consciousness in the past ten years or so.

As you read through this chronology, you'll see that after the first couple, every piece of malware listed is Microsoft DOS or Windows-related. Why is that? Well, partly, it's because of the large number of Windows machines out there. As Willie Sutton said, when asked why he robbed banks, "It's where they keep the money." Similarly, there are more Windows machines than any other kind, so for someone who wants to cause havoc, it makes sense to target a platform that is in widespread use.

However, there's more to it than just that. Symantec was quoted in the Wall Street Journal on September 18, 2003 as saying that more than 4,000 separate instances of Windows-oriented malware have been found in the wild, compared to 11 for Unix/Linux and none for the Macintosh. One would think that given the 40+% market share of Linux servers and the 10% share of Macintosh desktop machines, there would be a similar proportion of viruses targeting those platforms. The fact of the matter is that those platforms are more secure than Windows, and thus are more resilient to attacks than their Windows counterparts.

As a result, this history is nearly all about DOS and Windows. If you're a Windows user, be prepared to have the pants scared off of you. If you're a Linux or Macintosh user, you'll probably be more amused than anything else. With that caveat, let's proceed.

The first viruses

The very first viruses targeted mainframes in the late 1960s and early 1970s - the Pervading Animal virus attacked the Univac 1108, for instance. However, these were secluded instances that were generally more a theoretical curiosity than a widely-spread physical threat, since access to computers (at the time there were only mainframes and minicomputers) was strictly controlled.

The Elk Cloner is often credited with being the first virus that appeared 'in the wild' - in other words, outside of the computer where it was created. Created in 1982 by Rich Skrenta, it spread via Apple DOS 3.3 floppies. Skrenta has archived some trivia, including the source, at <http://www.skrenta.com/cloner/>.

The first popular viruses showed up in the mid 1980s. These were attached to programs that were distributed via bulletin boards that people accessed via their personal computers.

A virus writer would take a copy of a popular program or game, infect it with their virus, and upload it to bulletin boards that were accessed by computer enthusiasts. Someone would download the program or game and run it on their computer, unaware it had been infected. Some people would take copies of the infected program and give it to friends and associates, and even provide copies for giveaway or sale at user groups. The phenomena of shareware, where a computer programmer would write a program for free, distribute it throughout a network of bulletin boards, and ask people for voluntary payment if they found it useful, made sharing programs a popular pastime among computer hobbyists worldwide.

Indeed, for a number of years in the late 1980s, one of the chief responsibilities of a user group program librarian was to make sure that the files in the group's library were free of viruses.

The Brain

The "Brain", a virus written by two Pakistani brothers in the software business in 1986, was created in order to track piracy. Unfortunately for them, the virus spread beyond the Pakistan borders and the world was introduced to the dark side of computer programming. The first anti-virus program was written in 1988 - actually a virus itself - to find and remove the Brain virus, and to configure disks to prevent them from being infected a second time.

The Lehigh virus

Since the first viruses were simply attached to programs and games, it was relatively easy for anti-virus software to find and rid those programs of their infections, or for users to simply go back to their original, uninfected versions. Once the program was cleaned or replaced, the virus was gone.

As a result, virus writers became increasingly clever. One technique employed was to place a copy of the virus in memory so that it could infect program after program - as long as the computer was turned on. The next technique was to infect not just programs and games, but the system files and boot sectors on floppy and hard disks. These files and sectors were accessed each time the computer was turned on, thus ensuring that the virus would be executed and capable of doing its dirty work for as long as the computer was on.

The Lehigh virus (discovered at Lehigh University in November on 1987) attacked the core startup program COMMAND.COM on IBM PC compatible computers, and then made itself resident in memory. A month later, the Jerusalem virus was discovered at Hebrew University in Israel, and contained a bug that caused it to reinfect programs that already contained the virus.

Getting rid of these viruses was no longer simply a matter of replacing a program with the original; in order to get rid of boot sector viruses, the hard disk would sometimes need to be reformatted and all system software and programs would have to be installed again from scratch - hopefully from copies that were not infected!

In 1988, the first encrypted virus, Cascade, showed up in Germany. A year later, the Dark Avenger virus introduced a new feature - it would lay resident on a system and slowly damage files on the computer, in the hopes that infected files would be backed up over clean ones.

Then, once the virus was discovered, the user would replace the files on their computer with supposedly clean ones from backup, but to no avail.

Frodo, the first full-stealth file infector, showed up in October of 1989, and was designed to lay in wait, and then to damage the hard drive if the program it infected was run on or after September 22 of any year.

AIDS Information Disk trojan

The first trojan that hit the big time was probably the AIDS Information Disk trojan of late 1989. About ten thousand copies of a floppy together with supporting literature were mailed primarily to medical institutions throughout Europe. Purportedly sample or 'review' copies, the packages also included a license agreement that said, in part,

"In case of breach of license, PC Cyborg Corporation reserves the right to use program mechanisms to ensure termination of the use of these programs. These program mechanisms will adversely affect other program applications on microcomputers. You are hereby advised of the most serious consequences of your failure to abide by the terms of this license agreement."

and

"Warning: Do not use these programs unless you are prepared to pay for them".

The program contained a program that was essentially a rudimentary electronic brochure discussing AIDS and providing some simple AIDS risk assessment processes. The accompanying installation program, however, did much more than just install the program on the hard disk of the user. First, it replaced the existing AUTOEXEC.BAT file with its own version and renamed the original. Next, it installed an additional hidden program. The new AUTOEXEC.BAT first called the hidden program and then the original AUTOEXEC.BAT.

The hidden program kept track of the number of times the computer was restarted, and after 90 reboots, encrypted all of the data on the hard disk, presented the user with an invoice, and demanded payment in order to get the key to unencrypt the data.

You would think that no one would be so dumb as to implement such a scam and attach real contact information to the demand, but it turned out otherwise. This was 1989, when people were still feeling out the computer industry's boundaries, after all. The perpetrators were four individuals in Panama who had formed a company called PC Cyborg, and the evidence uncovered indicated that an additional 200,000 copies were planned to be shipped to American addresses if the European trial was successful.

This was a trojan because the installation program had a second, secret purpose specifically designed by the creators, but was unacceptable, and was not described to the users.

LAN-enabled Worms

In the late 1980s, the popularity of the local area network gave birth to a new generation of malware - worms that propagated from machine to machine automatically. Up to this point, malware was only transmitted via communication between two computers - an intermittent

link, or via “sneakernet” - one person giving a virus-infected floppy to another person. LANs provided an always-on connection between groups of machines, and thus once one machine on a network was infected, all others on the network could be easily infected. Indeed, many a network administrator tore their hair out once their server was infected.

The opening of the 1990 decade also saw a number of anti-virus programs show up in response, as virus-battling suddenly became a significant priority in business. In response, virus writers became more adept, creating increasingly complex virus sporting polymorphisms, armoring, and multipartite characteristics. Norton Anti-Virus was released by Symantec in 1991.

Internet-enabled malware

While not in the public eye, there was another network available to computer programmers in the late 1980s, and it, too, was the vehicle for a worm. The predecessor to the Internet, the ARPANET, was attacked by a worm written by 23 year old programmer Robert Morris. The program disabled thousands of computers on the network by filling their storage space with multiple copies of itself. Morris wrote the worm as an experiment but without sufficient rigor, the experiment went out of control.

The popular growth of the Internet in the early 1990s laid the foundations for yet another generation of malware - that which is transmitted via email. With the explosion of email usage in the late 1990s (do you know anyone without an email account anymore?), email quickly became the easiest way to transmit heinous programs and infect large numbers of machines.

Stoned virus

“Stoned”, making its debut in 1987, got its name from a message it would display on the user's computer: “Y our PC is now Stoned!” Subsequent variants displayed different messages or no message at all.

Propagated via floppy disks, Stoned would occupy the boot sector. If the computer was started via the floppy disk, the virus was read into memory, took control of the system, and infected the hard disk when the booting process was finished. The partition and master boot record were moved to another part of the hard disk, and was referred to by the virus.

Ordinarily, data was not lost from the hard disk, because DOS (this was 1987, remember) didn't use the boot sector that the virus uses. However, if that sector was used by third-party software to store data, during formatting, or for password access, or by drivers to access large partitions, problems would result.

In its original form, Stoned was 440 bytes long, would not infect floppy disks in B> drive, and moved Interrupt 12's return, denying use of memory between 638K and 640K to DOS, where the virus resides (remember, 640K ought to be enough for anybody!), using 1795 bytes of that space. As a result, CHKDSK would show 653312 total bytes memory instead of 655360 on most PCs.

It monitored Interrupt 13, and any attempt by DOS to read or write using Interrupt 13 triggered the virus to infect a disk that was in the floppy disk drive. It would also move the floppy disk's original Boot record code to the area used by the Directory, and if the disk had files listed in the now-overwritten sector, access to those files was now prevented since the directory information was gone.

Clever users used utility programs to recover the files themselves, since they weren't deleted themselves, as long as other files weren't written in the same space.

A teenager in Wellington, New Zealand wrote the virus in 1987 as a joke program to spread to other PCs via copying to floppy disks. When the floppy booted the PC, Your PC is now Stoned! would appear briefly on the monitor, and the speaker would emit a beep if the clock time was zero, or on every 8th boot.

Laroux virus

In June of 1996, Microsoft Excel became the first target of viruses designed to infect data files. Laroux, a macro virus, infected XLS workbooks (version 5.0 or later), and is still found in the wild, over eight years later.

Laroux consists of two macros: auto_open and check_files. The first macro ran when a document was opened (duh!) and forced the execution of the check_files macro. A file was created in the XLSTART directory (variously named PERSONAL.XLS or PLDT.XLS) that also contained the viral macros; these files were opened automatically upon subsequent executions of Excel so that any other spreadsheet file opened would also get infected.

CAP virus

The CAP virus upped the ante in Microsoft macro exploits in February of 1997. Hailing from Venezuela, this Word macro virus was specifically designed to fool users through a variety of techniques in order to stay hidden while propagating, and worked really well in that endeavor.

It consisted of a number of encrypted macros with names of common functions like AutoOpen, FileOpen, FileClose and so on. It deleted existing macros in the document and then removed the Tools/Macro and Tools/Customize menus in Word in order to stay hidden by not allowing users to access functions that would expose it. It also saved all files in DOC format, regardless of what format the user chose. This tricked many users, since they would try to save documents in RTF format (which can't contain macros and thus can't spread macro viruses), but the files, despite having an RTF extension, were still DOC files and still contained the virus.

CIH

CIH, known by various monikers, including PE_CIH, W95.CIH, and Win32/CIH, was distributed via uploads of infected files to Usenet in June of 1998, and was the first widely destructive virus.

Depending on the variant, CIH was designed to execute its destructive actions on either April 26th or the 26th of any month. The actions included overwriting the first megabyte of hard disk space (which contains system data such the File Allocation Table, the Root directory, and the Master Boot Record, among others) and overwriting the BIOS code, preventing boot up of the computer, if the computer used a flash RAM BIOS chip..

CIH was the first known virus to copy pieces of its code to unused areas inside Windows 9x EXE files in order to hide its existence by not increasing the actual size of the infected file.

StealthBoot virus

The StealthBoot originated in Tucson, Arizona in November of 1998, and was actually written in order to generate revenue, although not in a nefarious method. The author made it available on floppy disk and published it in a book.

As the name indicates, the virus attacked the boot record much like Stoned. It differed in that it also attacked the B floppy disk drive, and that it moved the original boot sector information and part of itself to the end of the disk, and marked those sectors as 'bad' so that the operating system wouldn't try to write other files on top.

This virus used the technique of directing operating system reads (such as the DIR command) away from the disk sectors it resides in, in order to hide from anti-virus programs from looking at those sectors and finding the virus code.

Happy99.exe worm

January 1999 marked the release of HAPPY99.EXE, a Windows 95/98 executable file that was spread over the Internet. Technically a computer worm, this one created a nuisance effect, displaying a fireworks show which was sleight of hand to distract the user from the real purpose, changing the Windows system file, winsock32.dll, that controls access to the Internet. Once modified, the new winsock32.dll caused a copy of the worm to be sent along with email or posts to Usenet.

Ethan virus

The Ethan Word 97 macro virus, found in January of 1999, infected the normal.dot template and executed when a document was closed. One function it performed was to turn off Word's Virus Protection option, so that users wouldn't be warned of its existence.

Melissa

Things got interesting a short later when Melissa showed up. Another Word 97 macro, Melissa (I still keep hearing the Allman Brothers in the background) was embedded in a document that was attached to an email message with an alluring subject line like 'Important message from <your name>' and a body that said "Here is the document you asked for - don't show anyone else!"

Now, who could resist this? In those innocent times (April of 1999, to be specific), not many people. When the user opened the document, the macro virus automatically executed and performed a number of actions, some of which were rather novel at the time.

First, the security settings inside Word were changed so that all macros would run when subsequent documents were opened, without the user knowing it was happening. Normal.dot, the default Word template upon which new documents were based, was infected with the virus, and the computer's registry was modified so that the virus would only run once per session.

Next, the macro looked up the first 50 email addresses in every available Microsoft MAPI address book on the machine, and automatically sent a copy of the message (and the infected document, of course) to each of those addresses. This feature was particularly clever, in that future recipients would receive a copy of this message from someone who had their email

address in their address book, and could thus be ostensibly trusted. Melissa spread like wildfire because of this technique.

This was particularly clever because Outlook puts their mailing lists first in the address book architecture. Thus, an email to the first 50 entries could actually end up going to hundreds or thousands of people, depending on how many addresses lists an individual user had, and how many people were on those lists.

Melissa also used a clever mechanics that took advantage of the poorly designed Word object model. Melissa was written in a language called Visual Basic for Applications (known as VBA). A pseudo-programming language, VBA can automate many tasks that Microsoft Office users would commonly perform, such as creating or copying documents and sending email messages.

“Hmmm...”, you say to yourself. “Automatically? This sounds like it could be abused.” And yes, it could. Like regular Office macros, VBA programs can be written to be run when a document or spreadsheet (even a presentation) is opened, without the user knowing about it.

Furthermore, unlike regular macros, Melissa wasn't visible if you opened the document and investigated the macros contained in it because the macro was contained in the document.open method of the Word object. Not many users had the technical knowledge to understand how to use the Visual Basic editor and manually look at the VBA code by hand.

Melissa overwhelmed many email systems, to the extent that a number of large companies shut their email infrastructure down completely while the virus could be removed, their computers patched, and their users trained not to open attachments. Not that this training really worked, as the next few virus attacks will show. For details, check out www.cert.org/advisories/CA-1999-04.html.

PrettyPark worm

The PrettyPark worm (June, 1999) was notable in that it operated on three different levels: as a worm, as a password-stealing Trojan horse, and as a backdoor to the victim's PC. It spread by e-mail when an unsuspecting user received a message with a file attachment called PrettyPark.exe. Once the victim ran the attached file, as they were encouraged to in a manner similar to Melissa, the worm infects the system by copying itself to the \Windows\System\ directory as a file named files32.vxd. Finally, it modified the Registry, so that it ran whenever Windows ran an EXE program file. It was particularly difficult to get rid of because of these multiple levels of infection.

Back Orifice 2000

The BackOrifice.Trojan showed up in 1999. An update of one of the most well-known trojans, this malware consists of two pieces, a client component and a server component. Together, they allow others to gain complete control over a computer via a network connection, and operated under Windows NT. The port number through which the trojan operates can be configured by the attacker.

Bubbleboy in the wild

November 1999 saw the release of the the Bubbleboy worm to the wild. It was the first malware found that could spread to a user's PC without the need to explicitly open an attachment, because the script containing the worm was included in the email message itself, not an attachment.

Bubbleboy depended on two ActiveX controls to be available on the system and marked as "safe" and employed holes in Microsoft's implementation of HTML rendering. Once marked, Internet Explorer 4.0 or later could use them improperly. Specifically, Outlook Express executes the worm if an email carrying the worm is viewed through the "Preview Pane", and Outlook will execute the worm if an email carrying the worm is opened. Like Melissa, Bubbleboy will send itself to every contact in the available email address books of the user.

Chode / 911 worm

A year later, the 911 Worm (also called Chode), became the first malware to be subject of an advisory from the FBI. It was the first code that would travel by itself - no email, sneakernet or other manual transmission was necessary. It used Windows sharing to travel from one Windows PC to another on the Internet.

ILOVEYOU

As I mentioned, people didn't learn. The ILOVEYOU virus overwhelmed the Internet in May of 2000. A piece of Visual Basic script (known as VBScript), it was attached to an email message like Melissa, but also employed other means of transmission as well. Usenet postings, IRC communications and Windows file sharing were all also used.

The ILOVEYOU virus was a piece of VBScript (Visual Basic script) that infected machines through a variety of means. It was attached to an email message and USENET news postings, transmitted through Internet Relay Chat (IRC) and even via Windows file sharing.

Like Melissa, it emailed itself to every email address in every email address book available to the user. The subject line was "ILOVEYOU" and the body of the message said, "Kindly check the attached LOVELETTER coming from me." When the user opened the attachment, the VBA code executed and started all over again. And, like Melissa, many people did so without thinking because the message they received was from someone who they knew, since their name was in the sender's address book.

Unlike Melissa, however, ILOVEYOU carried a rather destructive payload. Files with certain extensions anywhere on the network accessible by the user were replaced by a copy of the virus file. The registry was modified each time the virus was executed, which could cause problems if the virus was executed a large number of times in a short period of time. And confidential data, including user names and passwords on the local machine, were emailed back to the program's author, a teenage student in the Philippines.

This virus, like Melissa, relied on two things. First, it used the insecure default settings of a variety of Microsoft products, including Outlook, Internet Explorer, and the Windows Scripting Host to take advantage of security holes. Second, it used social engineering techniques, like sending mail from addresses known by the recipient, to trick users into performing an ill-advised action that the security holes allowed.

DDOS Attacks

The new millennium also featured the debut of the first widespread Distributed Denial of Service (“DDOS”) attack. Dozens of high profile web sites are shut down for hours as a series of DDOS attacks were orchestrated through UC - Santa Barbara computers that were hijacked by miscreants who planted malware on them.

Anna Kournikova

In February of 2001, knockout tennis player Anna Kournikova was made even more famous as digital pictures of her were offered as bait in another email attachment virus. Like Melissa and ILOVEYOU, executing the attachment forwarded the message and virus attachment to every address in the recipient's address book. The key difference between this and previous viruses was that experts believe it was created by a toolkit instead of being coded from scratch. The availability of such a toolkit meant that now amateur crackers could create and deploy viruses with a minimum of knowledge.

Code Red

Four months later, the Code Red worm attacked and infected tens of thousands of Windows NT and 2000 servers. The traffic that Code Red generated made a noticeable impact on the Internet, and was intended to attack the White House website. Fortunately, a group of defenders was able to analyze the code and develop a defense.

Code Red attacks Internet Information Server 4.0 or greater, the Web server software used on Windows NT 4.0 and Windows 2000, using a security hole that has been patched long ago. The worm scans the Internet for servers that are at risk, and when it finds one, copies itself to that server. The new copy then continues the search for other unpatched servers. Code Red also performs three other tasks - replicating itself for the first 20 days of each month, defacing servers' home pages, and attacking www.whitehouse.gov.

It's been a couple of years, but I still see Code Red attack attempts on our servers. It's amazing that there are still unpatched servers out there. Doesn't anyone read the news anymore?

Nimda

Late in the week after the 9/11 terrorist attacks on the World Trade Center and the Pentagon, the Nimda virus flooded the Internet and attacked millions of computers worldwide. The virus used yet more sophisticated methods, including multiple approaches, to infect and replicate, and was tremendously successful despite the increased wariness of users and administrators due to previous high-profile infections.

The first approach took advantage of a flaw in IIS that allowed an attacker to navigate to any desired folder on the drive that contained the Web site, and then access files in it.

The next approach used a mass mailer program that executes every 10 days. Like many other routines, it searches for email addresses on the system and replicates itself through email sent to those addresses.

The third approach is to infect EXE files in a number of fashions.

And the last was a series of system modifications that opened holes for other attacks to take place.

All in all, Nimda (that's "admin"s pelled backwards) caused untold millions of dollars in damage and downtime.

Klez

A month later, Klez struck. It, like other mass mailing worms, copied itself to all email addresses in the recipient's Microsoft Outlook address book. This worm differed from previous worms of the same style in that it contained a great deal of code that attempted to disable popular anti-virus programs so as to avoid detection while it was infecting and spreading.

It also carried a damaging payload - on the 13th of odd-numbered months, it causes files on local and network shares to become zero bytes in size, potentially destroying data on those shares.

SQL Slammer

January 2003 saw the SQL Slammer worm take the record as the fastest spreading piece of malware. Infecting machines running Microsoft's SQL Server software, it took advantage of a huge number of machines that hadn't been patched with an update that had been available months earlier - including over 40% of the machines inside Microsoft Corporation, giving additional credence to naysayers who defended user's lack of updating habits, as even Microsoft couldn't keep their own computers patched properly. Because the SQL Server software was used in many key business applications, businesses were particularly hard hit, shutting down automated teller machines and delaying airline flights.

W32.SoBig

The SoBig mass mailer worm took over the 'fastest spreading' title from SQL Slammer just 7 months later, infecting millions of machines in August of 2003. It also was worthy of note because it used a new technique where the From address is spoofed.

The subject line varied, but was one of the following: Re: Details, Re: Approved, Re: Thank you!, Re: Wicked screensaver, and Your details. The body contained a message directing the recipient to the attachment, saying "Please see the attached file for details". The attachment was a .PIF file with one of several names, including your_details.pif, details.pif, wicked_scr.scr, application.pif or movie0045.pif.

Not only did SoBig send itself to every email address it could find on the infected machine, but it also forged the "From" address in the email, using a different email address it found on the infected machine. In other words, suppose Al has Barb, Carl and Donna in his address book. If Al got infected with SoBig, the worm would send emails to Barb, Carl, and Donna. However, Barb, Carl and Donna wouldn't get emails with a "From" address of Al. Rather, Barb might get an email "from" Carl, and Carl would get an email "from" Donna, and so on. As a result, Barb would contact Carl, complaining to him that he sent an infected email to her. Carl, on the other hand, would be mystified, because (assuming he didn't know Barb,

and didn't know that he and Barb had Al as a mutual acquaintance) he would have no way of telling how his address was used in an email to someone he didn't know.

This scenario repeated itself around the world millions of times as people accused innocent people of propagating viruses. Ugly. And the reason this is important is that this same mechanism is now being used by spammers to send spam - so that spam arrives at a recipient's in box with a From address that is simply a random address taken from an infected user's machine.

Additionally, a substantial amount of spurious email traffic was generated as a result of virus notifications being sent to invalid or inappropriate email addresses. For example, Carl's email server might get a virus warning from Barb's email server upon detection of the SoBig virus in the email being sent to Barb. However, since Carl's computer isn't infected, that email warning is incorrect. Yet, it was still sent. Multiplying this scenario millions of times caused a significant amount of extraneous and unnecessary traffic.

MS Blaster/LOVSAN 8/29

The hits just kept on coming in the late summer and fall of 2003. The next one to hit was MS Blaster (also known as LOVSAN.) A worm that exploited the DCOM Remote Procedure Call vulnerability on Windows 2000 and Windows XP machines. The big new feature of MS Blaster was that it was programmed to perform a Denial Of Service attack on the Windows Update Web server in order to prevent infected users from downloading a patch to protect against the vulnerability.

Fortunately, the worm had a significant flaw, in that it targeted an alias of Windows Update, and upon discovery, Microsoft simply disabled the alias by removing the DNS record for windowsupdate.com, and attacks on the alias went nowhere. They were lucky, this time.

The worm's code also had a text message inside, which was never displayed to the user: I just want to say LOVE YOU SAN!! billy gates why do you make this possible ? Stop making money and fix your software!!

W32.Swen

People had barely recovered from MS Blaster when Swen struck on September 18. Swen was notable for two reasons. First, it used multiple means of propagation, each means carrying a different 'style' of interaction with an intended victim. Second, one of the means was an extremely convincing forgery of a what claimed to be Microsoft security bulletin (except Microsoft never sends out patches via email.)

A mass mailing worm that uses its own SMTP engine to spread itself, Swen also used file sharing networks such as KaZaA and IRC for the propagation vehicle as well as through mapped drives on networks, and via email attachments. Like Klez, attempts to disable anti virus and firewall programs running on the target computer.

If the worm is sent through email, the message is formatted to appear as if it's a critical security message from Microsoft, and either urges the recipient to run the attached file, or to

Swen exploited a vulnerability in Microsoft Outlook and Outlook Express whereby messages that are opened or previewed can execute malicious code. It distinguished itself through a new display of social engineering during execution. If the worm was installed on the

computer already, it displayed a notice that “This update does not need to be installed on this system.”

If the worm was not present, or if the filename of the executable began with the letter i, p, q, or u, the worm presents the user with a dialog that asks “This will install Microsoft Security Update. Do you wish to continue?” However, the installation will proceed regardless of the choice selected. If the user chooses “Yes”, a progress dialog will be displayed. If the user chooses “No”, the worm is installed without any messages being displayed to the user.

The worm then performs a variety of system configuration changes, and, in another fit of unique behavior, occasionally presents the user with a fake MAPI32 Except error that requests name, email, username and password information from the user.

An alternate form of Swen is an email message that purports to be a mail delivery failure notice, with the ' attached message' actually being an executable containing the worm.

As of this writing in late September of 2003, a week after initial attacks, one of my mails servers was still getting hit with nearly a hundred copies of the fraudulent email message every day.

What to do about malware

As you' ve seen in this chapter, malware is becoming inextricably linked to the spread of spam. As a result, you have to deal with both spam and malware at the same time, and many of the strategies and techniques are similar, or apply to both. At the same time, the way an end-user deals with malware (and spam) is different than the way an email administrator does. Thus, I' ve put this information into two different chapters - one for users dealing with spam and malware (see Chapter 9), and another for administrators (see Chapter 14.)

Trends for the future

The future does not look bright when viewed through the eyes of someone trying to defend against malware. As you' ve seen in the section on malware history, attacks are coming faster and faster, and are becoming more dangerous.

Let's face it, we've been lucky so far.

The malware generators - those folks who write viruses, worms, trojans, and other electronic nasties that have been plaguing our lives for the last few years - have produced a relatively benign crop of attacks - up to this point.

I say ' benign' for two reasons.

Malware has attacked known security holes

First of all, they' e targeted known security holes in the Windows environment, so there was a patch waiting to be applied after an attack began.

But what if a sophisticated hacker exploits a flaw that a software maker doesn' t know about, or knows about but hasn' addressed? It's amazing that this hasn' happened yet.

Such an attack, called a "Day Zero" event because there are zero days of warning about the vulnerable software, is becoming an increasingly larger worry on the desks of tech workers worldwide.

How real is this worry? Microsoft issues warnings about 'critical' security holes on a regular basis - there are clearly many more that exist in their software but just haven't been discovered yet. Jim Allchin, group VP for MSFT platforms, said in eWeek that some Microsoft code is so flawed that disclosing it would present a national security risk. And Microsoft has a track record of ignoring private warnings about existing vulnerabilities for months (many examples can be found at www.woodywatch.com) - often addressing the issue only after being threatened with public disclosure of the problem.

The Wall Street Journal carried an article entitled "'Day Zero' Fear Could Boost Windows Rivals" on page one of section B on Thursday, September 18, 2003. In it, Mark Dickelman, CIO of an electronic payments company in Chicago, is looking at distributing the risk they're currently exposed to by moving some of their computers to a non-Windows platform. That way, those machines are not subject to the same wide range of risks that Windows machines are, and thus a successful attack on one platform won't completely cripple the company.

As mentioned earlier, Symantec is quoted in the article as saying that more than 4,000 separate instances of Windows-oriented malware have been found in the wild, compared to 11 for Unix/Linux and none for the Macintosh.

Malware's effects haven't been real damaging

The second reason we've been lucky is that most of malware released to the wild hasn't been designed to do much more than flood networks and disrupt service. Rather, they'll attempt to replicate themselves and display a greeting or taunt a public figure. But there haven't been any reports of widespread damage to data or machines. The few that had true evil intent were so badly written that they didn't work at all, or had internal flaws that prevented them from becoming a true problem.

But imagine if just one of the recent viruses to have successfully swept the Internet had been set up to cause real damage - corrupt data, destroy hard disks, or perform DOS attacks on mission-critical infrastructure like DNS servers, hospitals, or airports?

Combine this possibility with an exploit based on a security hole that Microsoft hasn't produced a patch for yet, and much of the technological infrastructure that relies on Microsoft Windows could be shut down for days or weeks while the exploit is verified, a patch is created and tested, and then deployed.

And deployment won't be easy - SoBig specifically targeted the Windows Update site, although in a clumsy, easily side-stepped manner. What if microsoft.com itself was targeted? There'd be no easy - and reliable - way to deliver a patch once it developed. Outages could last weeks while the mess was sorted out.

When they figure it all out

So what's going to happen when a clever malware author puts it all together? I'm no pundit, so my sermonizing about the structure of a successful attack would be sort of pointless. It's logical to conclude, however, that the results would fall into several categories. First, individual machines and/or the data on them would be disabled or destroyed. If you don't have backups of both your software applications and your data, you're going to lose stuff. Next, networks (both internal and the Internet) will be overwhelmed with spurious traffic. As a

result, legitimate access will be impaired or unavailable. Finally, business functions that rely on computers to be available will be disrupted.

The malware attacks of 2003 already had a variety of crippling effects in isolated locations. SQL Slammer flooded the Internet with traffic, disabled the Internet in Korea, shut down airline reservation systems, locked down many ATM machines, and the safety monitoring system for a nuclear reactor. A few months later, MS-Blaster forced the largest railroad in the Eastern US, CTX, to close for several hours. Air Canada's check-in system was shut down, as was Maryland's DMV.

The so-called “Warhol Worm”, named because it can take over the majority of machines on the Internet in under 15 minutes, has already proven to be technically feasible. Indeed, SQL Slammer infected over 170,000 machines in about 10 minutes. With that kind of infection rate, human response will be ineffective.

Put these two together, and a properly constructed attack could, with a bit of luck, shut down the Internet and most personal computer operations for days or weeks.