Computer Parasitology

Carey Nachenberg
Symantec AntiVirus Research Center
Cnachenberg@symantec.com

Table of Contents

Introduction

Computer viruses have progressed from urban myth to annoyance to major threat; yet, even with all the damage that computer viruses have done, they pale in comparison to what we have seen and have yet to see from the computer worm.

Viruses are computer programs that are designed to spread themselves from one file to another on a *single* computer. A virus might rapidly infect every application file on an individual computer, or slowly infect the documents on that computer, but it does not intentionally try to spread itself from that computer to other computers. In most cases, that's where humans come in. We send e-mail document attachments, trade programs on diskettes, or copy files to file servers. When the next unsuspecting user receives the infected file or disk, they spread the virus to their computer, and so on.

So how do humans spread viruses? Most people exchange information in time intervals on the order of minutes, hours or days. Furthermore, information is sent to a relatively small group of people. Looking in my own e-mail out-box, I send messages with attachments (usually documents) to an average of three people roughly every 33 minutes during business hours. While these figures may not be typical of most users, they're certainly plausible and are corroborated by the (relatively) slow spread of most computer viruses.

Worms, on the other hand, are insidious because they rely less (or not at all) upon human behavior in order to spread themselves from one computer to others. The *computer worm* is a program that is designed to copy itself from one computer to another, leveraging some network medium: e-mail, TCP/IP, etc. The worm is more interested in infecting as many machines as possible on the network, and less interested in spreading many copies of itself on a single computer (like a computer virus). The prototypical worm infects (or causes its code to run on) a target system only once; after the initial infection, the worm attempts to spread to other machines on the network.

The infamous Melissa only required a user to open a single infected document to spread itself to hundreds of thousands of users! IRC worms only require a simple user login to the IRC online chat system to spread themselves, and the recent Explore.Zip worm could gain hold of thousands of machines with the launch of a single program. While humans exchange information at large time intervals to small groups of people, computer worms have no such restrictions.

Computer worms will pose the greatest threat to both consumer and corporate computer systems over the next decade. As we will see, they will change the nature of anti-virus (anti-malware?) software and require paradigm shifts in enterprise security and infrastructure[i]. This paper will cover the past, present and future of computer worms, attempt to provide IT professionals with an understanding of how and why these threats spread, and offer some insight into the nature of future anti-worm solutions and worm-resistant corporate infrastructures that can be leveraged to protect the enterprise from these threats.

Worm Classifications

This section will propose some general classifications for computer worms. Computer worms can be classified based on two characteristics:

1. The transport mechanism used by the worm to send itself.
2. How the worm is actually launched on a computer system.

Worm Transport Classifications

The following are known or potential worm transport schemes.

E-mail Worms

The *e-mail worm* is one that uses e-mail as its primary means of transport. Under this top-level classification, we can create two subcategories:

1. *Native e-mail worms*
2. *Parasitic e-mail worms*

A *native e-mail worm* is one which is built in the native scripting language of the host e-mail system. Such a worm is carried in a proprietary form along with its associated e-mail message, as opposed to being carried as a file attachment. The native e-mail worm can only exist within the e-mail platform, and is not viable outside of the host e-mail system. To date, we have seen no native e-mail worms.

A *parasitic e-mail worm* is one which leverages the transport capabilities of an e-mail system to spread itself. The parasitic e-mail worm, for example, may use the e-mail program to send itself as an attachment to an e-mail. The parasitic e-mail worm can exist outside of the e-mail platform, and may actually use other techniques to spread itself. Of the recent worms, Melissa, ExploreZip and Happy99 would be considered parasitic e-mail worms.

Arbitrary Protocol Worms: IRC Worms, TCP/IP Worms, etc.

The a*rbitrary protocol worms* spread themselves using one or more non e-mail based protocols, such as IRC's DCC protocol, the FTP protocol, or using simple TCP/IP sockets. The Internet Worm could be considered an arbitrary protocol worm since it used standard TCP/IP connections (from one UNIX box to another) to spread itself.

The *peer-to-peer worm* is another example of an arbitrary protocol worm that spreads itself over peer-to-peer networks.

Worm Launch Classifications

These classifications are used to describe how the worm actually gains control of a computer system: does it require user interaction (and how much?), or can it spread unaided.

Self-launching Worms

Worms that are capable of spreading to a new system and actively running on that system can be called *self-launching worms*. These worms do not require user interaction in order to gain control of a system; instead, they exploit some aspect of the host (operating system, application system, e-mail system) to cause their code to automatically execute upon introduction to a new system. The Internet Worm and the IRC worms (described in later sections) are examples of self-launching worms. A subset of this category is the *back door worm*. The back door worm is one that exploits a back door in a target system to gain entry *and* to ensure that it is launched, again without human intervention.

User-launched Worms

This category of worms requires user intervention in order to execute on a new system. For instance, the Melissa worm/virus is just such a worm. In order for Melissa to infect a system, an infected attachment must be manually opened/viewed by a user. The worm cannot cause itself to launch on a system without user intervention.

Hybrid-launch Worms

These worms are capable of spreading using both mechanisms. An example of a hybrid-launch worm is ExploreZip. This worm, when sent in e-mail, required a user to launch the infected attachment to gain control of the system. On the other hand, once running on a computer system, ExploreZip would automatically spread itself to other computers over the peer-to-peer network. These targeted machines would then become infected on the next reboot (without any known user intervention).

A Brief History of Worms

The Xerox Worms: The First Computer Worms

The first computer worms were created in the labs of John Shock and Jon Hepps of Xerox's Palo Alto Research Center in the early 1980s. The team developed five different worms, each of which were designed to perform useful tasks on the network. These researchers quickly learned the danger of worms; one of the self-replicating programs had a bug that caused it to malfunction while running during non-business hours. The next morning, users found their computers crashed and found that resetting them did not help; the worm would continually re-infect the systems and quickly cause them to crash. After this event, the research group produced a "vaccine" (perhaps the first ever anti-virus) to prevent the worm from crashing systems and subsequent worm research died out.[ii]

The CHRISTMA EXEC Worm: The First Widespread E-mail, User-launched Worm

Years later, in December of 1987, the first widespread computer worm was released into IBM VM/CMS systems. This worm, dubbed the CHRISTMA EXEC Worm, was a simplified predecessor to the Melissa virus/worm seen this year. The worm was written in the REXX programming language, a common batch programming language on UNIX systems. When a user received this file as an e-mail attachment, they had to detach the file to the hard drive and manually run the batch file to cause the worm to spread. Once the worm was launched by the user, it displayed a character-based picture of a Christmas tree and then attempted to spread itself further over the IBM e-mail system. The worm used entries in the audit trail file (that contained logs of all received and sent e-mail), as well as the user's personal address book to determine its next round of recipients. [iii]

Even though CHRISTMA required significant manual user intervention to spread, it clogged thousands of mailboxes and the underlying network with millions of copies of itself[iv].

The Internet Worm: The First Arbitrary Protocol, Back Door Worm

Roughly one year later, on November 2nd, 1988, the infamous Internet Worm (created by Robert Tappan Morris) wreaked havoc on the fledgling Internet. Unlike the CHRISTMA EXEC Worm, this worm was designed to spread itself without any human intervention. The Morris worm would spread to DEC VAX and Sun 3 machines running BSD UNIX[v] by connecting to the target UNIX machine in one of three ways (to the e-mail service, the *finger* service, or by hacking passwords and performing a remote execution on a target machine). Once it had penetrated the target computer, it would send over a small C program and cause the target host to compile and run this program. Next, the compiled program would pull over the remaining worm components and launch them; at this point the worm had a foothold of the targeted computer and could spread to new computers on the Internet. [vi] It is estimated that the Morris worm affected more than 6,000 computers on the Internet using these techniques.

The IRC Worms: The First Consumer-oriented Arbitrary Protocol, Self Launching Worms

In 1997, standard end-users users encountered the first mainstream computer worms: IRC worms. IRC, or Internet Relay Chat, is a chatting system available on the Internet. Users can connect to this system using an IRC client (e.g. a Windows program) and chat with thousands of other users about thousands of different topics. One of the most popular IRC client programs, mIRC, was equipped with a fairly powerful scripting language that was targeted by the first IRC worms. These scripts were programmed to send themselves to new users as they joined the IRC chat "channels." Once the recipient was infected, their mIRC program would execute the worm logic and also participate in the reproduction of the worm. While many were benign, some of these worms were capable of other malicious actions. For instance, users who spread the worms could send commands to infected systems and cause these systems to perform any number of malicious actions.[vii] In essence, these were the first "remote control" worms ever created, since they allowed an attacker to remotely control and/or damage infected systems.

The Happy99 Worm: The First Mainstream Consumer-oriented Worm

The Happy99 worm, released in early 1999, has arguably affected more home Internet users than any other worm in history. As of July 27th, 1999, the Symantec AntiVirus Research Center has received over 4,200 submissions of Happy99. Once the worm is received in e-mail and launched by the user, it displays a window with fireworks and, in the background, installs itself onto the system. Subsequently, any time the user sends an e-mail (with the popular Internet SMTP protocol used by Eudora, Netscape mail, etc.), or posts a message to a USENET newsgroup, the worm will send an additional message to the same recipient(s), adding the worm as an attachment. While this worm undoubtedly affected both corporate and consumer users, this worm is considered a consumer-oriented worm because it spreads using the SMTP protocol, which is the predominant e-mail protocol used by end-users.

The Melissa Virus+Worm: The First Mainstream Corporate Macro Hybrid (both a Virus and Worm)

The "Melissa Virus" is actually both a computer virus and a computer worm, or hybrid threat. Melissa is spread inside of Word 97 document files. When a user receives and subsequently views a Melissa-infected document, the Melissa macros will run inside of Word for Windows 97 and use the Outlook e-mail program (if present) to send a copy of the infected document to the first fifty users in the Outlook address book. Melissa will also infect the Word for Windows environment, allowing it to spread to other documents on the user's machine. These second-generation infections are also fully infectious, and if shared with other users, will continue the infection process.

Based on the number of e-mail boxes that were flooded with copies of Melissa, it is without a doubt, the most prolific worm of all time [viii]. Melissa spread itself prolifically during its short reign, filling up potentially hundreds of thousands of *e-mail boxes*. However, due to the quick handiwork of system and mail administrators, the actual number of computer users directly affected by Melissa was probably far less than the number of affected mailboxes. In other words, most users probably did not actually launch Melissa, but might have found several copies of it in their in-box had they been able to access e-mail.

Melissa achieved such widespread distribution because when it spreads, it sends itself to the first fifty entries in the Outlook address book. While some of these entries might refer to typical end-user e-mail addresses, many corporations have entries such as "All Whammydyne Employees" among the top fifty entries. An e-mail sent to the "All Whammydyne Employees" address would literally reach each of the 3,000 employees at Whammydyne. Consequently, a small number of seed infections in a corporation could quickly clog hundreds of thousands of e-mail boxes with Melissa infections.

The ExploreZip Worm: The First Widespread Hybrid-launch, Arbitrary Protocol Worm

Unlike Melissa or Happy99, the ExploreZip Worm spread itself to other computers using two distinct mechanisms. First, like Melissa, ExploreZip was capable of leveraging Outlook, Outlook Express and Exchange e-mail programs to send itself over e-mail. Instead of sending itself to the first fifty users like Melissa, this worm sends itself to users that have recently sent e-mail to the infected user.

In addition to spreading itself via e-mail, ExploreZip will also iterate through all machines that are visible on a peer-to-peer Microsoft network. The worm will copy itself to accessible machines and update a configuration file on the target machine to cause the machine to launch the ExploreZip worm during the next boot-up. The worm continually searches for other peer machines to infect, and consequently, was difficult to eradicate from corporate networks. The moment an administrator would remove the worm from a machine, another copy would re-infect it.

In addition to infecting peer-to-peer networked machines, ExploreZip also deleted the contents of a variety of files from both local hard drives and the hard drives of networked peers. The peer-to-peer capabilities of this worm clearly underscore the vulnerability of peer-to-peer networks in the enterprise.

Evolution of Enabling Technology

Three technological trends have had a huge impact on the viability and simplicity of computer worms: infrastructural homogeneity, ubiquitous programmability, and increasing connectedness via a homogeneous communications mechanism.

Infrastructural Homogeneity

The homogeneity of computers, operating systems and communications platforms has been the single largest enabler for computer worms.

Today, more than 90% of the world's desktop computers are running the Wintel platform. An equally high percentage of end-users use standard SMTP e-mail, and many large corporations (and even some end-users) are standardizing on e-mail clients like Microsoft Outlook. In the word processing area, Microsoft Office enjoys a virtual monopoly for home and business users alike.

In agriculture, such a homogeneous environment is called a monoculture, and is generally known to be a *bad thing*. If a farmer plants a single variety of crop on their land (for instance, to optimize their yield of that crop), they increase the susceptibility of their crop to disease – and once the disease affects one plant, it can easily spread to genetically-similar neighbor plants. In essence, the standardization of all of the above computing technologies has created a computing monoculture that has increased our computers' susceptibility to computer-borne disease.

Already, we have seen thousands of macro viruses attacking the Microsoft Office platform. Several high-profile worms have leveraged the widespread Outlook and SMTP e-mail platforms. Finally, more than 99% of all computer viruses are designed for the Wintel platform. However, it is likely that virus writers have targeted these platforms for more than just their sheer ubiquitousness. These platforms are also readily available to young male adolescents (the predominant creators of malware), well documented, and most importantly, easily programmable.

Ubiquitous Programmability

Ubiquitous programmability of Windows components (Word/Excel macros, COM, etc.) has made it possible for worms to spread without complex programming.

Who would have thought that the word processor or spreadsheet would be the single most successful platform for computer viruses and worms? E-mail is easy to imagine – worms could send themselves over e-mail - but common Office applications? Unfortunately, by adding robust programming capabilities to the Microsoft Office product line, Microsoft has made the Office products the platform of choice for virus and Trojan creation. By allowing macros to copy themselves from one document to another, the Office platform has been a huge enabler for computer viruses.

Unfortunately, Microsoft Office macros not only have access to the features and components of the Office suite, but also to other components of the host computer system. If Microsoft had limited the Office macro environment so that macros could only interact with other components of the Office suite, this would have limited malware creations to viruses and Trojan horses that could spread themselves only within the limited Office environment. Unfortunately, a second technology – the Component Object Model (COM) – has had a huge impact on today's malware.

In a nutshell, when a programmer designs a new application, s/he can choose to make the functionality of the application available to the rest of the system (and not just the user) via COM. Programmers can then design their programs to make use of the functionality provided by the COM-enabled program. Using COM, Outlook provides the means for other programs to log-in to the user's mailbox, examine messages, extract attachments, enumerate the entries in the address book and send e-mail. For instance, one could write an expense reporting application that would make use of the Outlook e-mail program. The designer of the expense reporting application could program it to use e-mail functionality of Outlook to send copies of an expense report to the finance department, without having to know a single thing about how to program an e-mail system. Clearly, COM technology has been a huge enabling technology for programmers.

In a logical move, and one which provides great value for legitimate programmers, Microsoft made it possible for Office macros to leverage the same powerful features of COM. Furthermore, given the simplicity of the macro programming language supported by Office, virtually anyone could pick up a book and develop powerful macro programs that have the ability to do far more than summing tables in a spreadsheet (a typical task that could be performed by a macro)! These COM-enabled macros can affect the entire host computer system, and more worrisome, the worldwide network of machines based on the same technological monoculture.

Increased Connectedness via a Homogeneous Communications Mechanism

The increasing connectedness of the Internet permits worms to spread faster, and to more machines, than ever before.

Until recently, the propagation rate of computer viruses was limited to how fast computer users sent infected files in e-mail (or on file servers, floppy diskettes, etc.). Computer viruses can quickly infect many files on a single computer system but spread much more slowly from one computer system to another because of their reliance on user behavior. Given that users share information more than they share programs (at least in the corporate environment), user-initiated e-mail has allowed macro viruses to spread far more quickly than their binary cousins (DOS and Windows viruses). Yet as quickly as macro viruses can be spread by user e-mail, they pose a limited threat to the computing community at large. By the time a new macro virus can infect a handful of users, anti-virus companies can respond with a fingerprint update and prevent any further spread.

With more users on the Internet than ever before as potential targets, worms can now spread more quickly than ever. And as mentioned, the homogeneous, ubiquitous, COM-accessible communications mechanisms makes writing such a worm a snap. Why should a malware threat wait to be sent by the user when it can send itself? The computer worm doesn't wait for the user to send its malicious logic in an e-mail. Instead, it takes matters into its own hands. The worm exploits the communications infrastructure to send itself from one computer to another; consequently, it can potentially spread itself thousands of times faster than a traditional virus.

While e-mail is an ideal transport mechanism for computer worms, it is far from the only viable communications mechanism. Malicious code has only begun to exploit peer-to-peer networking, and this trend it likely be change in the coming years. Windows 95, 98, NT (and soon Windows 2000) support peer-to-peer networks. Users can configure Windows to allow other users on the Windows network to access their files without restriction. By exploiting this facility, computer worms can quickly find other machines on the Windows network and copy itself to these machines. The most recent Explore.Zip worm uses exactly such a mechanism to spread itself over corporate networks, and was extremely successful.

Once such a threat sends itself to a second computer, given the high probability that that computer will support the same homogeneous technologies as the first host, it has a high chance of continuing its spread.

Other Factors

There are a number of other factors which may enable worms to spread much more quickly, in both the home and corporations.

Corporate/Consumer Bridge Technologies

"Malware authors go with what malware authors know." In other words, virus writers will design their threats to exploit those technologies that they have on their own computer so they can test their creations. Consequently, those corporations that employ worm-enabling technologies that are common to both the corporation and the home are much more susceptible to these attacks. For example, to date we have seen a number of viruses and worms leverage Microsoft Outlook and Outlook Express to send themselves; these programs are widely used corporate and consumer e-mail programs (and they share the same COM programmable interface). Conversely, we have seen no worm-based attacks that leverage Notes to spread themselves. Notes, unlike the Outlook products, is used exclusively in corporations, and is a less available technology for virus/worm authors.

Home Networking

As more homes begin to setup home networks, we expect the growth in number and prevalence of computer worms will grow dramatically. Given that many of today's virus writers can't test a worm before deploying it due to lack of resources, access to a home network for testing purposes may accelerate the growth and release of new worm programs.

The Future of Worms

While the majority of computer malware consists merely of knockoffs of older malware strains, there are a core group of virus writers that consider themselves "trail blazers." This group has brought us threats like Boza, the first 32-bit Windows virus, Strange Brew, the first Java virus, and countless others. It is inevitable that over the next few years we will see viruses and worms attacking and leveraging many heretofore untargeted platforms, as varied as Palm Pilots, Lotus Notes, and personal web servers.

Cable/DSL Brings Worms To The Home

Continuous static connection + Connected desktop apps + Scripting Capabilities = Worm heaven

As more users adopt broadband technologies in the home, we expect that the incidence of computer worms targeted at home users (and small businesses) will grow rapidly. Today, home Internet users are assigned dynamic IP addresses each time they login to the Internet. Given that users log in and out frequently, it becomes very difficult for a worm to find such a target machine and spread to it.

As users migrate to broadband technology such as cable modems or Digital Subscriber Line (DSL), they will have constant, reasonably static connections to the Internet, making their computer a “sitting goose.” Computer hackers or roving worms will be able to easily cull Internet addresses and use them to attack these machines. Furthermore, we expect that as more users adopt broadband technologies in the home, that consumer-driven connected applications will grow in popularity.

Today, products like PointCast sprinkle corporate desktops, but are less appealing for the disconnected home user. However, as broadband technologies become more ubiquitous in the home, these connected applications will grow in popularity; real-time stock tickers, search agents, personal web servers, music players, and “instant message”/chat programs will be running on every desktop. Furthermore, as we have seen in the Office application category, vendors will start adding macro/scripting support to these applications to extend their capabilities for power users.

While each of these connected applications will increase the quality of the home computing experience, each also contributes security risks for the home user. We expect that computer worms will exploit these connected (and often not security-conscious) applications as back doors into home systems. In such an environment, a worm like Melissa will easily infect huge numbers of home users. Unfortunately, since threats that affect the home inevitably find their way into the corporation, these threats will have an impact on the enterprise too. We expect that as corporate users bring their connected applications from home into the workplace, this will be a ripe platform for worm propagation, and the spread-rate of these worms over this new medium will rival that of the peer-to-peer and MAPI-based worms described below.

The personal firewall (in conjunction with anti-virus software) will become a must-have application and help to stem at least some of the worms and viruses that will plague the growing number of connected desktops.

MAPI Worms

MAPI is an acronym that stands for Messaging Application Programming Interface. Many common e-mail client applications, such as Outlook, Exchange, etc. support the MAPI system (which is provided via the COM mechanisms discussed in previous sections). This means that other applications, including worms and viruses, can leverage e-mail functionality (sending e-mail, receiving e-mail, examining attachments, etc.) without having to understand the details of e-mail systems or protocols. In fact, Office 97 macro viruses can access the MAPI system using the simple Visual Basic language, making it possible to construct a computer worm in less than 30 lines of programming statements! The Melissa and ExploreZip viruses used this facility to spread themselves, and we expect that this mechanism will be employed more than any other worm replication mechanism in the foreseeable future.

Some corporations have found good uses for Office 97, macro-based MAPI support: at least some departments create self-mailing documents and spreadsheets to facilitate submission of expense reports and other forms. However, given the huge threat this functionality poses to corporations, it is questionable whether this functionality should be supported at all by Office applications. Furthermore, merely making macro-based MAPI support an configurable option of the Office platform is an inadequate solution.

Even today, many macro viruses can and do change Office settings – for example, disabling macro protection – without the users knowledge. A properly designed worm could easily enable disabled MAPI support and then use this facility to spread itself. The only way to address MAPI-based worms is to deliver Office applications with no MAPI capabilities, or more generally no COM capabilities, to the enterprise.

MAPI or e-mail based worms have two interesting properties that are worth further discussion. First, the sheer volume of e-mail generated by a worm can have unanticipated collective effects on the computer network or e-mail infrastructure. As we saw with Melissa, a number of companies actually had their e-mail servers crash due to the sheer volume and velocity of e-mails sent by this virus/worm. These crashes occurred despite the fact that Melissa was engineered to limit its spread to fifty e-mail addresses and only e-mail itself once from any infected machine.

Second, based on anecdotal reports from Symantec customers [ix], e-mail based worms may achieve a high rate of penetration in corporate e-mailboxes, but these worms actually have a much lower penetration rate on the corporate desktop. These corporate customers indicated that only a small percentage of users actually opened Melissa-infected documents and infected their desktops. While even a small number of infected users might spread a Melissa-like threat to hundreds of thousands of corporate users, unless these attachments are opened by those users, they pose little threat to corporate informational assets; the infections contained in e-mail boxes are not actively running and cannot steal information from desktop files, the server, etc. Based on this experience with Melissa, it might be interesting to study the average rate that users check e-mail, and how often they bother to open e-mail attachments at all. Another interesting question: how long does it take IT departments to detect such a spreading threat.

Of course, whether or not mail-based threats achieve high desktop penetration is of little help to administrators; corporate users and IT staff will still experience huge amounts of lost time due to clean-up efforts, downed e-mail systems, and help-desk calls. However, it does indicate that e-mail-only worms may pose more of a threat as a denial of service attack than as a desktop infiltrator/information asset thief.

Information Stealers and Remote Control Worms

Over the last few years, we've seen a rash of new virus, worm and Trojan threats that are capable of exporting information from an infiltrated machine or allowing a remote attacker to take control of such a machine. While older malicious code threats would delete files or format hard drives, the new *payloads* of choice are information stealing and remote control since these payloads leverage the power of the Internet.

A good example of this is the Prettypark worm. If this worm finds its way onto a computer and can acquire a connection to the Internet, it will connect to the IRC chatting service and wait for commands from an attacker. A malicious person can send Prettypark commands and perform any number of malicious actions on the compromised computer including stealing information, deleting files, etc. Personal and corporate firewalls and encryption software can help to limit the risk from these threats.

We expect that the number and complexity of information-stealing and remote control worms will dramatically increase over the next few years. It is even believable that a digital underclass will develop that will use such malicious tools to extort, black mail and steal valuable corporate information.

Peer-to-peer Worms

The Melissa virus/worm spread itself using MAPI e-mail commands and achieved a huge penetration of company e-mail systems, flooding virtually every e-mailbox in some corporations. However, it is unclear how many actual corporate desktop computers it was able to infiltrate. How many of the thousands of corporate users who received Melissa in e-mail actually viewed the document attachment and launched Melissa? According to word-of-mouth accounts, few.

On the other hand, consider the ExploreZip worm. This worm spreads both using e-mail and by exploiting peer-to-peer networks. Specifically, ExploreZip will seek out other networked Windows 95/98/NT computers with unsecured, mapped hard drives and spread to these machines and run without the need for user intervention. While a Melissa-like threat could fill e-mail systems with millions of copies of itself, an peer-to-peer threat like ExploreZip has the potential to achieve a much higher rate of actual desktop (and server) penetration in organizations with lax security. Based on anecdotal evidence, those Symantec customers that were hit by ExploreZip had a much higher number of active infections (i.e. actively running copies of ExploreZip on Windows 95/98/NT machines) than those hit by Melissa.

Worms can spread to other peer-networked machines by using simple Windows programming techniques, so this is likely to be exploited far more in the future. This should serve as a wakeup call to administrators; peer-to-peer networks are large holes waiting to be exploited by computer worms.

E-mail Scripting Worms

We often tell end-users that they can't get a virus by simply opening an e-mail. While this is true for most consumer e-mail packages, there is a risk with corporate, group-ware e-mail systems. Most of the group-ware e-mail programs allow the user to embed programmable scripts in messages. These scripts allow the user to create simple mail-based user interfaces, forms, etc.; unfortunately, on some platforms, they may also be leveraged to produce malicious or self replicating code.

If such a script-based worm were to be deployed in a corporation, it could quickly flood the e-mail system with copies of itself. It could also spread to partner corporations that share the same e-mail system, assuming that the e-mail is transported using the native protocols of the group-ware system. However, such a worm would probably not be able to spread to other external organizations. Why? Since most *inter-company* mail is sent over the standard SMTP protocol, the script information associated with messages is stripped from the e-mail as it passes through the Internet, rendering it harmless.

Do we expect to see these worms? Yes and no. Given that the Internet acts as a natural “antibiotic” for these types of malicious code (stripping it out of non-internal e-mail), the likelihood is that such a malicious message would have to originate inside a corporation (or one of its partners) to be successful. The risk of creating such a worm would therefore be high, since the worm author could be easily caught. On the other hand, the Word Concept virus was created inside a corporation as a proof of concept. In any case, it would be a good idea to check with e-mail vendors for information in this area.

ActiveX and Java Worms

While worm authors may choose to build ActiveX-based worms, the likelihood of Java worms is extremely small in the short to medium term.

ActiveX programs are basically fully functional Windows programs that are capable of performing any number of malicious actions (just like ExploreZip), and therefore should be considered a potential threat. ExploreZip, Happy99 and even Windows viruses could easily be built and deployed as ActiveX components instead of standard Windows programs. That said, given that ActiveX components must be digitally signed before they can be used in most browser environments, we believe that this will limit the number of ActiveX worms actually deployed in the wild. Any time people can be linked to their creations, the associated liability appears to be a strong deterrent. (See the Ubiquitous Authentication section for more details.)

Java worms are unlikely for two reasons. First, Java security prevents unsigned Java applets from spreading themselves or accessing the local computer resources. Second, while signed Java applets can access the host computer system and potentially spread themselves, the liability issues associated with digitally signed applets will likely limit the number of wild threats.

Second Generation Worms

Computer viruses and anti-virus technologies have experienced a co-evolution over the years. For example, in the DOS virus space, simple self-replicating viruses prompted simple string scanning engines. These string scanning engines prompted encrypted viruses, which prompted "algorithmic," script-based wild-card string scanning technologies. These updated engines prompted polymorphic or self-mutating viruses, and so on.

As of the writing of this paper, we have seen only the first generation of computer worms. These worms don't attempt to hide themselves, always propagate identical copies of themselves, and are fairly easy to remove from computer systems. Over the next few years, we are likely to see a similar evolution of worm technology that parallels the evolution seen in the computer virus world. The following sections describe some of the anticipated advancements we expect to see in this space.

Polymorphic Worms

Worms like Melissa and ExploreZip spread themselves through e-mail by sending identical copies of themselves from one computer to another over e-mail or over the peer-to-peer network. For example, every time ExploreZip sends an e-mail to another user, it attaches the exact same 210,432 bytes of executable code. Furthermore, ExploreZip always sends a virtually identical text message in each e-mail:

Hi *Recipient Name*!

I received your email and I shall send you a reply ASAP.

Till then, take a look at the attached zipped docs.

bye/sincerely *Recipient Name*

In a similar fashion, each time Melissa sends itself in e-mail, it uses the same subject line and same message text.

Early on, a number of resourceful system administrators used these static attributes to help them weed out these worms from their e-mail servers. By writing simple scripts or programs that looked for these static text messages, many administrators were able to purge both Melissa and ExploreZip from their e-mail databases. What is the likelihood that the next batch of worms will be this simple? Small.

These worms could easily change or randomly generate their attachment names, the e-mail text, the subject lines and other attributes to make these worms a nightmare to detect. Just as with the early compute viruses, administrators are using homegrown tools and techniques to root out these menaces; however, in the near future, we expect that such homegrown solutions will quickly hit their limits.

Furthermore, the next iteration of worms could easily polymorph themselves, as hundreds of traditional DOS, Windows and macro viruses do today. Or even worse, these worms could be equipped with a "worm generator" that would generate entirely new worm strains with similar or different propagation mechanisms, different sequences of instructions, etc. and flood corporate e-mail systems and networks with hundreds of functionally different worm and virus strains! While anti-virus companies have strong technologies for detecting polymorphic viruses, signature scanning techniques and even simple heuristics will be useless against randomly generated worm/virus threats.

Retro Worms

Retroviruses, a misnomer from biology, are computer viruses that actively attack anti-virus software to prevent themselves from being detected. Many retroviruses will delete anti-virus definition files, disable memory resident scanners and generally wreak havoc with anti-virus software. We expect that at least some of the next generation computer worms will also possess these characteristics.

Consider what kind of trouble ExploreZip would have created had it deleted the anti-virus software of every computer it spread to? Not only would the administrator have to distribute virus definitions to thousands of computers, but he or she would also have to redistribute and re-install the entire anti-virus application. While a retro-virus might have reach a small percentage of the corporate desktops, a worm like ExploreZip could quickly decimate a majority of the desktops' anti-virus protection.

Memory resident computer viruses have been around for years, and anti-virus companies have always instructed its customers to boot from a clean, write-protected floppy and then scan their system with the latest virus definitions to clean them. This high-cost solution works well on DOS and Windows 9X-based systems, but will become less effective as corporations migrate to Windows NT and Windows 2000 (potentially encrypted) NTFS file systems. If desktops are configured to use NTFS partitions, traditional anti-virus software will be unable to access protected files outside of Windows NT, and the problem only gets worse with Windows 2000's encrypted file system.

The next generation of anti-virus software may contain Windows memory scanning and memory repair technologies [x] to deal with memory-resident Windows viruses and worms, but it is unclear how effective these memory-oriented engines will work. For example, consider what would happen if an NTFS-based system became infected by a new memory resident retro worm (or virus). If this retro worm prevented anti-virus software from loading while the worm was resident, how could the administrator repair such a system? They could not launch the anti-virus software from within Windows, since the worm would prevent such an action. Furthermore, if the administrator attempted to clean-boot the machine and run the anti-virus software from a DOS floppy diskette, they will also fail due to anti-virus software's reluctance to scan NTFS file systems.

Perhaps one of the only solutions to this problem is for anti-virus vendors to produce one-shot tools that are specifically tailored for each new resident retro worm (virus). These tools must be constructed from scratch, so they cannot be recognized by retro worms; otherwise they will be just as susceptible to the retro worm as existing anti-virus software. (And now consider something really scary – a worm or virus that prevents any new executable files from being copied to the system or launched from any drive except the local computer drives. It's time to get the disk formatting tools out!)

Stubborn Worms

Stubborn worms (viruses) are those worms that are designed to prevent themselves from being unloaded from an infected system. There are a number of ways of accomplishing this and many are described in Peter Szor's paper found in these proceedings [xi]. There are several possible attacks that anti-virus software might be able to use to remove such a worm, but in general, such a worm would be difficult if not impossible to remove from a system.

Wireless Worms

While wireless devices are in their infancy in the United States, they are gaining widespread acceptance in Japan (where the wireless infrastructure is actually more advanced than the wired infrastructure) and in Europe. Today's cellular phones have most of their software content embedded in firmware, and do not have the ability to run or download arbitrary software applications, but this is likely to change in the next two to three years. As these devices begin to support arbitrary software applications, there is no reason why they cannot also host malicious mobile code.

Today, in the small hand held (SHH) space, Palm Pilots and Windows CE devices have the ability to run arbitrary software applications, including computer viruses. Furthermore, the newer Palm Pilot computers and other hand-held devices also support infrared and wireless communication (Palm VII). Using this technology, users of the Palm III and later models can actually exchange applications "through the air!" Unfortunately, it would also be simple to construct a computer worm that would send itself using the same mechanism. In the future, we expect that other small handheld devices will also support such wireless application exchange, and we will see computer viruses and worms attack these devices. In fact, we already know of several malicious programs that purported to spread using the infrared communications mechanism of handheld calculators!

How would such a wireless worm work? If we assume that wireless SHH devices can easily locate and talk with other devices using either standard data-oriented cellular phone numbers or via an IP addresses, a wireless worm could do the following:

1. Examine the "recently called" list or "received calls from" list of the device.
2. Send a message to the phone numbers/IP addresses of all devices in the lists above. This message would contain the computer worm. The message subject might say "try this cool game, it's fun!"
3. Upon receipt at the other end, the user would read the message and run the worm program, causing it to spread to other devices.

Such a wireless threat could quickly infiltrate thousands of wireless devices, delete or modify phone entries, application data, etc. Hopefully, since all transmissions between devices are likely to be logged (and billed!), authorities can quickly pinpoint the origin of such attacks and take appropriate measures. However, given the dependence that businesses have on these phones, a wireless worm attack could have a devastating affect on businesses, government and consumers alike and might be a target for terrorist cyber-attacks.

Examination of Worm Epidemics

Arguably, since the dawn of modern computing, we've seen a total of four worm-driven epidemics: CHRISTMA EXEC, the Internet Worm, Melissa and ExploreZip. This section will examine each of these threats and attempt to identify why they were successful and what underlying holes they exploited to spread as prolifically as they did.

Case Study: Mass E-mail Worms, CHRISTMA EXEC and Melissa

Let's take the easy ones first. CHRISTMA EXEC and Melissa both spread exceptionally well due to their exploitation of *corporate* e-mail. Even though each of these threats required user intervention to spread themselves (the user had to manually detach and launch CHRISTMA EXEC to spread it), they spread to thousands of machines in a matter of days. The following list details why these worms spread so well and more generally why corporate e-mail is such an effective vector for worms:

It's easy to obtain "addresses" of other targets

Since most e-mail programs maintain address lists to facilitate sending e-mail in the enterprise (or even from the home machine), the worm has an easy list of targets to send itself to.

Homogeneous e-mail makes spreading easy

Since most desktops in a corporation use the same mail solution, a worm can exploit this homogeneity and spread rapidly.

Humans are the biggest security risk; there's no need to find a back door into the system

Why would a worm try to hack into a UNIX system or find a back-door past the firewall when it can just send itself to a person that is likely to open the attachment and spread the worm throughout their corporation? Without human involvement, these e-mail worms would not spread beyond the first recipient.

Corporate e-mail systems offer "one degree of separation"

Corporate e-mail systems allow anyone in the corporation to send e-mail to anyone else in the corporation. The e-mail address book provides all the addresses in one easy-to-find place. Furthermore, modern operating systems like Windows make it easy for a worm to get to this information (with COM, etc.). Consequently, a e-mail-based worm can quickly reach everyone in an organization. This is in contrast to consumer e-mail networks, where a user might have several dozen e-mail addresses in his/her address book, but not thousands. This is *one reason* why threats like Melissa spread so rapidly to so many machines, while threats like Happy99 (an SMTP-based, largely consumer oriented worm) have spread widely, but much more slowly.

While corporate e-mail systems allow one user to easily send mail to any other user in the corporation, these address books typically do not contain consumer e-mail addresses. This is one reason why Melissa had such a low penetration of home users outside of corporations. The other reason is that most home users don't use exchange, so even if a home user did get hit, the worm would not be able to spread to other computers as a worm, but only in its viral capacity. Had Melissa spread using either Internet (SMTP) e-mail systems or Exchange, we probably would have seen a high infection rate of home users as well.

Why infect one other computer when you can infect 50… or 50,000

Both Melissa and CHRISTMA EXEC spread so rapidly because during each iteration, they sent themselves to a large number of machines. Even worse, Melissa spread itself to potentially thousands of other machines with each infection because of its inadvertent usage of "all company" e-mail addresses near the top of the Exchange address book.

Spread to other computers as soon as you can

These worms spread themselves to other computers via e-mail as soon as they were opened/run on a computer. This allowed them to spread rapidly and not at human speeds. Contrast this behavior with the Happy99 worm, which would only send itself out to other users when the infected user sent e-mail to those users. Since most home users don't send extensive amounts of e-mail, this limited the spread rate of the Happy99 virus.

Mailbox penetration or computer penetration?

While these e-mail based worms were prolific at spreading themselves, they mostly wound up languishing in e-mailboxes rather than actually running on people machines. In the case of Melissa, anecdotal evidence from Symantec customers indicated that the virus/worm got spread to thousands of e-mailboxes but was actually launched by only a small subset of the corporate population. This can be attributed to several factors:

1. People check their e-mail/detach attachments at a slow rate compared to the spread rate of the worm.
2. System administrators intervened and brought down e-mail systems before more people could become infected and spread the threat.
3. The worm spread so prolifically that it brought down e-mail systems, preventing most users from accessing it.
4. Given that all Melissa e-mails were identical, this allowed IT staffs to warn their users, also limiting the actual machine penetration.

One or all of the above effects may have had an impact on the success of Melissa.

Case Study: Back Door Worms and The Internet Worm

A back door worm is one that gains access to a system via a “back door” or system vulnerability, rather than being delivered via front-Office means (e-mail, for example). This class of worms has the potential to achieve rapid system penetration (as opposed to just e-mailbox penetration, like Melissa) since it gains entry into and runs on a system without user intervention.

There are a number of methods that can be used to penetrate supposedly secure systems [xii], including:

1. Exploiting default administrator passwords, that have not been reset, to gain access to a system.
2. Using dictionary-based password attacks to break into user accounts and remotely login to a system.
3. Using buffer-overflow attacks on common or obscure network services to cause arbitrary machine code to be executed on a target system.
4. Exploitation of debug facilities that are built into standard network services.
5. Attack of non-secured shared drives and peer-to-peer devices.

The Internet Worm actually used the second, third and fourth items from the list above to penetrate computer systems on the fledgling Internet. The following factors may be responsible for the success of the Internet Worm

It’s easy to obtain “addresses” of other targets

Just like Melissa, the Internet Worm found it easy to find other targets. Without this facility, it would have had a much harder time spreading. According to [x]:

“The [Internet] worm's role in life is to reproduce - nothing more. To do that it needs to find other hosts. It does a 'netstat -r -n' to find local routes to other hosts & networks, looks in /etc/hosts, and uses the yellow pages distributed hosts file if it's available. Any time it finds a host, it tries to infect it through one of the three methods, see above. Once it finds a local network (like 129.63.nn.nn for ulowell) it sequentially tries every address in that range.”

With the advent of corporate directories (such as Novell’s NDS or Microsoft’s ActiveDirectory), worms will find it even easier to obtain a list of targets to attack. While the Internet Worm used a roundabout method to locate targets, its technique was evidently just as efficient as any modern worm we’ve seen to date.

Homogeneous environments makes spreading easy

The Internet Worm was successful (supposedly hitting up to 1/10th of all computers on the fledgling Internet) because it targeted the dominant platforms and exploited common holes. Given corporate standardization on the Windows platform (and a small number of other platforms: Novell, Linux, Solaris, HP/UX, AIX), corporations may be overly susceptible to future back door-based worm attacks.

Back door worms spread best unhindered

Had Internet administrators installed operating system security fixes/patches or used firewalls with updated security rules, they may have largely prevented the Internet Worm fiasco. The back doors that the Internet Worm used to infiltrate systems were well known at the time of infection and fixes were available well before the Internet Worm made its debut!

Today, some modern firewalls with application-level proxies can block externally-based back door worms from entering the corporate network. These firewalls monitor all transactions from the outside world into the corporate network and can detect buffer overflow exploits or the other attacks mentioned above and alert the administrator to the threat. Like anti-virus software, these firewalls must be updated on a regular basis or they may leave the corporate network unprotected from recently discovered attacks.

If corporations maintain an up-to-date firewall, most back door-only worms will be successfully repelled from entering the corporate network via an external network connection. However, these back door worms may enter the corporation via other means: users can download them from USENET news groups, receive them in e-mail, etc.

If a back door worm does infiltrate the corporate network, via any number of avenues (such as e-mail, download off the Internet, etc.), it can wreak havoc on the enterprise. Once a worm finds its way into the corporation, it has free reign of the corporate network and can attack virtually any machine in the corporation without hindrance from a firewall. *To protect against such back door worms, corporations should definitely consider employing internal firewalls around systems that contain critical data and/or applications.*

Spread to other computers without user intervention

The Internet Worm spread itself to other computers as soon as it gained control of a new computer and did not require human intervention to spread. This allowed it to spread extremely rapidly to susceptible machines.

Computer penetration

The Internet Worm achieved widespread computer penetration since it actively infiltrated computer systems and did not rely upon user behavior to gain control of these systems. A user did not have to detach the Internet Worm from an e-mail to spread it; this worm spread itself without any user intervention or action. Similarly, by employing a back door scheme, ExploreZip was also able to penetrate a larger number of actual computer systems (vs. e-mailboxes) than Melissa.

Case Study: Hybrid Worms and ExploreZip

The Hybrid Worm is one that uses a number of different mechanisms to spread itself over computer networks. To date, the ExploreZip Worm is the best example of a successful hybrid worm: it spreads itself using both e-mail and back doors. This type of worm has arguably has the best odds of both infiltrating many machines in a single company, as well as spreading well across different companies, due to its combination of infection vectors. This combination of vectors allows ExploreZip to:

1. Quickly penetrate corporate firewalls, via the e-mail vector.
2. Quickly spread itself across the peer-to-peer corporate network, achieving active running infections on desktops, using its back-door vector.
3. Quickly spread itself to other corporate sites and end-users, via the e-mail vector.

It's easy to obtain "addresses" of other targets

As with the other worms, ExploreZip spreads quickly because it has easy access to lists of target computers. Specifically, ExploreZip uses two mechanisms to spread. First, it scrutinizes the Outlook e-mail in-box to determine which users have recently sent e-mail to the infected user. The worm then replies to these users, sending a copy of itself as an attachment. As these target users open the attachment, they will spread the worm as well.

In addition, the worm uses facilities built into the Windows operating system and Windows networks to locate other peer-to-peer networked hard drives. If lax security is in place, the worm will find one or more vulnerable computers and copy itself to these machines. Finally, it updates an .INI file on the target drive so the host computer will launch and spread the worm upon the next reboot. It is clear that peer-to-peer networks are extremely vulnerable to worm-based attack and should be eliminated from the corporate network if at all possible.

Homogeneous computers makes spreading easy

As with the other worm threats, ExploreZip requires a homogeneous environment to spread. The worm relies upon an Outlook (Outlook Express) e-mail client and peer-to-peer Windows networking to spread.

The human is the biggest security risk; there's no need to find a back door into the system

Given that the Hybrid Worm exploits both e-mail as well as back doors, it can easily gain access to a firewall-protected corporate network via e-mail. Once inside the poorly secured corporate network, the worm can use either mechanism to spread itself.

Spread slowly or spread quickly?

Unlike Melissa, ExploreZip did not exploit the "1 degree of separation" aspect of corporate e-mail systems. Rather than sending itself to the first fifty users in the address book, ExploreZip only sent itself to those users that sent e-mail to the infected machine. Presumably, this would cause ExploreZip to spread at a much slower rate than other worms: the rate at which users receive new e-mail. This technique arguably limited the spread of ExploreZip, at least via e-mail, but may have also contributed to its success! While Melissa took down corporate e-mail servers due to the sheer volume of e-mail, ExploreZip metered its spread, and the lack of e-mail volume may have prevented administrators from being alerted to ExploreZip's presence.

ExploreZip may have also had success because it continuously tries to spread to other peer-to-peer networked computers once it has infected a system. While rapid replication in e-mail may be a giveaway for a computer worm, rapid replication via other back-door mechanisms appears to be a very effective vector assuming it does not noticeably bog down the corporate network. Using this peer-to-peer mechanism, several of Symantec's customers had thousands of computers hit by ExploreZip!

Finally, ExploreZip was also unlike Melissa in that it continues to send itself (to anyone that sends e-mail to an infected system) well after the initial infection of a system. Melissa, on the other hand, only sends e-mail out once during the initial infection. Because of this behavior, even a single isolated computer infected by ExploreZip can continue to actively "push" the infection for days or weeks. This behavior, coupled with the e-mail reply behavior described above, may have balanced ExploreZip's spread rate.

Mailbox penetration or computer penetration?

Unlike Melissa or CHRISTMA EXEC, ExploreZip was able to achieve a high penetration of e-mailboxes *and* actual computer systems. The high penetration of computer systems can be attributed to its peer-to-peer spreading mechanism. Users on other peer-to-peer networked computers could become infected with a simple reboot; there is no need to open e-mail or launch an attachment!

Payload and trigger conditions affect the worms' viability

Of the major computer worms, only ExploreZip has a malicious payload. This worm will destroy the contents of several types of files, including documents, spreadsheets and programming source code, on both local and networked volumes. ExploreZip will perform this attack the moment it infiltrates a new system and will continue to attack the local and networked computers until it is manually terminated or the infected machine is quarantined from the network.

By using such a payload mechanism, ExploreZip has doomed itself to be discovered very rapidly. On the other hand, it also has the opportunity to damage a large amount of critical data before it can be eradicated from the corporate network. If ExploreZip had a delayed payload (e.g. deleted files only after a month of infection), ExploreZip would have probably never been able to deploy its malicious payload in an actual computing environment. This is in contrast to slower-spreading computer viruses like CIH. Since these viruses spread themselves more slowly and attract less attention, they have the opportunity to penetrate a large number of systems over a longer period of time. By intentionally dealing their payload, they can achieve widespread distribution and then perform more widespread damage.

Since ExploreZip will delete files from networked volumes without actually residing on the computer that hosts those volumes, ExploreZip can destroy data on machines whether or not they are protected by anti-virus software! This is yet another reason why peer-to-peer networks are so dangerous in the corporation. A single infection of ExploreZip on an unprotected machine can destroy files on hundreds of anti-virus-protected networked computers, without generating a single anti-virus alert.

Containment

How can corporations protect themselves against computer worms? The following sections describe some of the defenses that corporations and governments can use to stem the threat of computer worms.

Proactive Steps

Run Anti-virus Software on Servers, Gateways, and Desktops

Enough said.

Remove "all company" Addresses From Your Lists

Computer users rarely have the need to send e-mails to the entire company and such a facility is extremely vulnerable to e-mail-based computer worms. E-mail administrators should limit public e-mail lists to small functional groups and eliminate all company-wide lists. Should users need to send such an e-mail (this should be rare), they can forward the e-mail to an administrator for company-wide posting.

Lock Down All Peer-to-peer Networking

Peer-to-peer networks are a huge security risk for network-aware worms and viruses. We recommend that administrators lock down peer-to-peer networked drives on all computers where this is not absolutely required. Administrators may also want to establish an official policy against peer-to-peer volumes and distribute this to users.

At the very least, the administrator should maintain a special computer grouping or domain in the network management software (or anti-virus console) for all peer-to-peer networked computers. This will enable quick deployment of anti-virus definitions to these particularly vulnerable machines.

Deploy Internal Firewalls

Corporate firewalls are fairly effective at preventing both hacker and malware attacks from outside sources; however, they provide no benefit once a worm has entered the corporate network. As we have seen with ExploreZip, the vast majority of computers that were actually penetrated by ExploreZip were attacked from within the corporation, from other peer-to-peer networked computers. Deploying internal firewalls could prevent such intra-network infections.

Administrators should consider deploying internal firewalls around corporate servers, such as:

1. File servers
2. E-mail servers
3. Corporate databases/SQL servers.

In addition, personal firewalls are effective at preventing attacks on desktop PCs running Windows 9X or Windows NT. While this may be a more expensive option, it could seriously neuter many back door worms and Trojan horses.

Disable E-mail Script Capabilities

If your group-ware product supports e-mail scripting, this should be disabled for all but a few users (most likely those in the IT department). By disabling these facilities, you can protect your corporation from Native e-mail threats.

Strip Executable Content From Incoming E-mail

Some group-ware and gateway-based anti-virus products have options to allow the administrator to strip executable content from either incoming or outgoing e-mail messages. Administrators should take advantage of these facilities if they are available. For example, while some employees use macros in the corporation, it is rarely the case that users need to exchange macros between companies. By configuring your anti-virus solution to automatically strip all macros from document attachments entering or leaving the enterprise, you can protect your users and business partners from inadvertent worm or virus infection. If such a measure is too obtrusive for your work environment, the same facility can be used only in the event of a worm outbreak (or impending outbreak).

Use Heuristics and If Possible, Digital Immune System Technology

Anti-virus heuristics are very effective at detecting new and unknown viruses; depending on the type of virus, heuristics can detect up to 90+% of all new and unknown strains. By coupling strong heuristics on the desktop, the server and the gateway with a digital immune system, the anti-virus offering can be made more even powerful.

A digital immune system (such as the one being developed by Symantec and IBM) is a *closed loop* system that is comprised of both client and server components. The system generally works in the following way:

1. The anti-virus software on the desktop, file server, gateway, or group-ware server detects a potential new or unknown virus with heuristics.
2. The sample is quarantined from the host computer and sent to the administrators console for review.
3. The administrator can examine the sample, automatically strip any proprietary content, and forward the sample to the vendor-operated automated servers for analysis.
4. Upon receiving the sample(s), the automated servers determine if the sample is already detected by the latest anti-virus definitions (its possible that another user submitted the same sample only hours earlier). If so, the system immediately returns an appropriate fix for the virus, for deployment in the enterprise.
5. If the sample is not known, it is fed into an automated replication system. The sample is replicated to other files (documents, executable files, etc.)
6. The system correlates all replicated copies of the virus/worm and derives a fingerprint and a cure.
7. The system tests the fingerprint and the cure to make sure it works on all replicated samples.
8. The system officially integrates the fingerprint and the cure into the virus database.
9. The system sends the solution back to the customer and immediately scans all other pending issues just in case they contain the same virus/worm (this speeds up the resolution time for subsequently submitted issues). The user can then deploy these definitions to their servers, gateways and desktops.

10. As new updates are produced, they can be publicly posted to web servers or proactively pushed to the community of digital immune users.

The first version of the Symantec/IBM digital immune system, released in July, 1999, is capable of automatically analyzing and deploying a fingerprint and cure for most Word and Excel macro viruses in about 2 hours time (at the time of this writing, about a week after deployment, three viruses have been automatically analyzed). Such a closed loop system can be very effective in reducing the window of vulnerability for corporate users, and as the software systems and hardware imp rove in speed, we hope to reduce the cycle time to fifteen minutes or less. The ultimate goal is to deploy the cure faster than the virus/worm can spread itself.

Before Impending Infection

To Prepare For E-mail Worms

If you receive notification of a possible new e-mail-based worm threat, but do not believe that this worm has already infiltrated your network, you have the following options:

1. If your vendor has provided you with virus definitions, immediately deploy these virus definitions to the gateway anti-virus scanners and the group-ware anti-virus scanners. Next, deploy the virus definitions to the file servers and finally to the desktops.
2. If your e-mail gateway (or anti-virus protection) can be configured to strip attachments, configure this protection to automatically strip all file attachments of the appropriate type (e.g. prevent document attachments if you're facing a Melissa-like threat, etc.). This rather draconian measure can be undone later after virus definitions are available.

To Prepare For Arbitrary-protocol Worms

If you receive notification of a possible new arbitrary-protocol worm threat, but do not believe that this worm has already infiltrated your network, you have the following options:

1. If the worm may use e-mail facilities, follow the steps above.
2. If the worm uses other TCP/IP-based mechanisms to spread, update your firewall rules to temporarily block potential points of entrance into your corporation.

Active Infection

If you believe that the worm has infiltrated your network, the following recommendations may help:

If Hit By A Destructive Worm: Update File Server Permissions

If you suspect that a worm may be modifying or destroying data on shared network volumes, immediately change the privileges on all files to READ ONLY. This will prevent the worm from destroying important corporate data, while allowing users to access critical business data.

If Hit By a Data Export Worm: Limit Access To Data and Networks

If you suspect that a worm may attempt to access sensitive files and export this information out of the enterprise, immediately do the following:

1. If you believe the worm is using standard network protocols to export information, lock down Internet access, stopping all outgoing transmissions to the Internet. If you know specific communications ports that the worm uses to communicate with the outside world, you can

configure the firewall to specifically block these transmissions and allow other legitimate transmissions to continue.

2. If you believe the worm is file server-aware and may be examining your file servers, take down all file servers until a reasonably full virus definition rollout (to desktops and file servers) and containment process is complete.
3. Disable all outgoing e-mail (or down e-mail servers) if the worm exports information via e-mail.

If Hit By an E-mail or Arbitrary-protocol Worm Infection: Distribute Virus Definitions to Gateways, E-mail Servers and File Servers First

Ideally, when receiving a new virus definition set from an anti-virus vendor, administrators should roll out the latest definitions to the entire corporation. In practice, this is next to impossible. There is a huge list of reasons why network management software (NSM) fails to properly distribute software updates: remote computers aren't available to be updated, computers have been turned off, hard drives are too full, users reboot during updates, etc. That said, administrators should roll out their virus definitions in a tiered fashion to reduce the spread of e-mail-based or hybrid computer worms. By rolling the definitions out to the computer systems that e-mail worms use to spread themselves, one can quickly lock down the corporate network from further infections.

First, roll out the anti-virus definition update to the e-mail gateway(s). This will prevent the threat from re-entering and re-infecting the corporate network from the Internet.

Second, take down all group-ware e-mail servers, roll out the anti-virus definition update to these servers, and immediately initiate a scan of all messages. Some anti-virus products allow you to scan only those messages that arrived after a certain time. If you know the approximate date and time of the initial infection, you can use this facility to limit the scanning time and get users back on e-mail as soon as possible. These steps will prevent any new copies of the worm from filling users' mailboxes and remove any existing copies that have already flooded the e-mail system. While it would be ideal for users to continue using e-mail during the cleaning process, this leaves them open to infection (with most anti-virus products for the common group-ware systems).

Third, roll out the virus definitions to all file servers and initiate an immediate on-demand scan of all shared volumes. After rolling out the new virus definitions, there is no need to "down" the server; the anti-virus software will monitor all file transfers to and from the server in real time.

Finally, begin rollout of the virus definitions to all desktops and initiate an on-demand scan of all desktop systems.

If Hit By a File Server-aware Worm Infection: Distribute Virus Definitions to File Servers First

If your company is infected with a worm that uses file servers to spread itself, you should update your server anti-virus definitions first and immediately initiate an on-demand scan of all susceptible volumes.

Next, roll out the new virus definitions to the gateway and group-ware anti-virus products, and finally to the desktops. Make sure to initiate an on-demand scan of all systems as they are updated.

If Hit By a Back Door Worm Infection: Down All Affected Networks

If you have reason to believe that your company has been infected with a back door worm, you should update your gateway, server and group-ware anti-virus definitions first and immediately initiate an on-demand scan.

Next, you have two choices:

1. Shut down all critical networks and/or target machines that may act as a vector or a host for the worm and then initiate manual cleanup of all potentially infected machines. This extreme response should be applied in the case where the worm is known to re-infect connected systems or if the worm is damaging.
2. Roll out virus definitions to potentially infected machines and initiate an on-demand scan as soon as possible.

Future Anti-worm Technologies

The following sections describe current and future technologies that will help to protect against computer worms.

Windows Memory Scanning and Repair

Many of the new computer worms, such as ExploreZip, Happy99 and Remote Explorer (both a virus and a worm) are native 32-bit Windows programs. While this current generation of worms can easily be removed from memory with standard Windows tools, the next generation of worms may not be so simple. Native 32-bit Windows r*etro worms* and *stubborn worms* may be difficult, if not impossible, to remove from computer systems without specialized memory scanning and repair tools. There are a number of technological hurdles that must be overcome before these technologies are effective at removing new/unknown worms, but even with limited functionality they may still provide real value for corporate customers and consumers alike (See Retro Worms section).

Behavior Blockers

Instead of detecting computer worms after they have a chance to infiltrate and gain control of a computer system, why not stop them from infiltrating the system in the first place! This is what behavior blockers attempt to do. A worm-oriented behavior blocker might do the following:

1. Require user, administrator or policy-driven authorization to update critical areas of the Windows computer (such as the Windows registry, INI files, batch files, system services, device drivers, etc.)
2. Block all attempts to send e-mail by all non-authorized programs, macros or scripts.
3. Block all attempts to send e-mail to more than N users (where N might be 50, 100 or 500) by non-authorized programs.
4. Block all attempts to use non-standard Internet services (this is what a personal firewall does; see below for details).

As with all behavior blockers, stopping the behaviors listed above will occasionally prevent useful activities from occurring. However, the high cost of computer worm outbreaks may quickly justify the inclusion of worm-oriented behavior blocking on the corporate and consumer desktops.

Personal Firewalls

Like corporate firewalls, personal firewalls monitor all network communications entering and leaving the protected computer. These lightweight firewalls are designed to block all transmissions that aren't specifically allowed by the user (or administrator) and may help to prevent arbitrary protocol worms from gaining a foothold in a corporation.

Worm Heuristics

Anti-virus companies are already updating their heuristics to detect the next wave of macro-based computer worms. Companies are also looking into methods for heuristically detecting 32-bit Windows-based worms; this is actually a much more difficult task than simple virus heuristics. While viruses are typically small programs (on the order of several kilobytes), 32-bit Windows threats (including computer worms) are often hundreds of kilobytes or more, and are much more difficult to detect heuristically. Case in point, ExploreZip, a native Windows worm built using Inprise's Delphi development environment, was not detected by a single anti-virus heuristic (known to the author).

Automated Worm Replication and Analysis

It is likely that in the future, anti-virus vendors that are pursuing digital immune system strategies will update their immune systems to automatically generate cures for computer worms, in addition to viruses. Already, researchers at the IBM T.J. Watson Research Center are working on automated replication for computer worms, for inclusion in the Symantec/IBM Digital Immune System.

Unlike computer viruses, which can be automatically replicated and analyzed on a single, contained computer, worms require entire virtual test networks to be constructed to detect the worm's replication across multiple machines. Furthermore, given the wide variety of protocols, applications, and systems that may be leveraged by the worm to help it spread, such worm replication and auto-analysis systems will undoubtedly be complex beasts (Melissa leverages Outlook, Happy99 leverages SMTP e-mail, IRC worms require the mIRC client, etc.).

Future Containment Approaches

Given the destructive potential of the computer worms we have seen thus far, how might corporations and vendors deal with computer worms in the future? The following sections provide some insight into potential solutions to alleviate the computer worm problem.

Ubiquitous Authentication

When Microsoft announced that it would use digital signatures to protect against malicious mobile code, the entire anti-virus community nearly had a fit. Many researchers complained "Digital signatures/ authentication systems don't actually stop malicious mobile code." This is true. But while these systems can't tell you whether or not a piece of code is malicious or not, they can tell you where it came from, and this can be an equally useful function.

At the last ICSA conference, I asked the question: "How many corporations have been hit by a malicious ActiveX or Java program?" The answer was zero, and this concurred with all of the research I had done over the previous months [xiii]. Why is this the case? Based on a decent amount of research, my deduction is that authentication had a great deal to do with the lack of in-the-wild attacks. Internet Explorer, by default, will only accept authenticated ActiveX controls from the Internet and it will only allow authenticated Java access the local computer system. Therefore, any potentially harmful ActiveX or Java content downloaded off the web will be digitally signed, assigning liability to the software producer. While such a technique will not prevent a computer virus from infecting an ActiveX object before it is digitally signed, it will prevent the majority of intentional attacks. (And based on experience, the former case is very rare as well.)

In a similar fashion, robust, authenticated e-mail systems may also help to dramatically reduce the number of computer worms infecting corporations. For instance, consider the following solution: At the corporate e-mail gateway, a proxy authentication server verifies the authenticity of all incoming e-mail via digital signatures. Any e-mail that can't be authenticated will automatically have all attachments that may contain executable content stripped from the message. All authenticated e-mails will pass through unhindered. Such a system could also be configured to be even more stringent, stripping all non-signed executable files, ActiveX controls or user macros from all messages. A similar solution could be used at the firewall to verify the content of all executable code pulled from the World Wide Web.

Such a system would not strictly protect against e-mail-based or computer worms, but it would:

1. Provide a huge disincentive for anyone within the corporation to intentionally spread a user-launched worm, since they could now be tracked and easily prosecuted.
2. Provide a huge disincentive for anyone outside the corporation to intentionally introduce a user-launched worm into the corporation, since they could also be tracked and easily prosecuted.
3. Prevent any user-launched worms of external origin from entering the corporation accidentally.

Unfortunately, given the lack of a ubiquitous public key infrastructure (PKI) for the Internet, such solutions are at least several years away. While it may seem overkill to require authentication for each and every e-mail or web transmission into the corporation, based on all available information, this is one of the most effective ways to discourage/prevent malicious code attacks.

Policy-driven File/Macro-level Access Control

In most corporations, any user can run any program (or macro) on their PC without intervention. Such an open environment is extremely susceptible to computer viruses, worms and Trojan horses. If computer worms continue to increase in prevalence, we may see corporations adopting and deploying "access control" technologies on the desktop. An access control solution would only allow specifically approved programs and macros to run on the typical user's computer, without exception. This scheme, while extreme, can largely prevent infestation by malicious code.

Macro-free Products

If worms continue to grow in prevalence, corporations may start requesting (and getting!) Office and group-ware products that are macro-free or macro-light. While Microsoft has, up to this point, resisted producing Office products without macro capabilities, the time may soon come when both macro-enabled and macro-free(light) versions of the Office applications are available. This simple solution should save millions of dollars in cleanup costs, and corporations will soon have to make a concerted effort to determine if the usefulness of the macro justifies its cost due to its security risk.

Conclusion

Computer worms have grown to become the fastest spreading and most costly malicious code threats of this decade. These worms leverage our infrastructural homogeneity, the ubiquitous programmability of Wintel machines, and increasing connectedness of the Internet to spread rapidly through the enterprise. Relatively speaking, computer viruses spread slowly when compared to the computer worm. While a virus might slowly spread from one corporate department to another, the computer worm can often blitzkrieg through an organization in hours or even minutes. This makes worms, especially with destructive payloads or data export capabilities, extremely ruthless attackers.

Today's anti-virus solutions are only marginally effective at protecting against the fast-spreading worm and it is clear that these solutions will need to evolve and be supplemented by other systems to provide sufficient protection for customers. Solutions like digital immune systems, e-mail authentication, heuristics, and behavior blocking will help to control the worm problem in the future; however, corporations need to seriously consider containment plans and emergency response to deal with emergencies and special cases. Without major changes to applications, communications facilities and the operating systems – all of which are unlikely to be accepted in the corporation or the home - computer worms are an inevitability and anti-virus/anti-worm software is, at best, a partial solution to the problem.

Good luck and may you never need a parasitologist.

---

[i] Private communications with Steve Trilling, SARC
[ii] http://www.ifs.univie.ac.at/~c9225414/security/worm.html
[iii] http://www.eu.veg.org/Risks/20.32.html
[iv] http://www.byte.com/art/9611/sec11/art4.htm
[v] http://www.ifs.univie.ac.at/~c9225414/security/worm.html
[vi] http://www.ee.ryerson.ca:8080/~elf/hack/iworm.html
[vii]http://www.avpve.com/viruses/classification/4_mirc.html
[viii] Private communications with Steve Trilling, SARC
[ix] Private communication with Sherrallee Buzzell, SARC Coordinator
[x] ‘Memory Scanning Under Windows NT’, Peter Szor. Proceedings of the 1999 Virus Bulletin Conference.
[xi] ‘Memory Scanning Under Windows NT’, Peter Szor. Proceedings of the 1999 Virus Bulletin Conference.
[xii] http://www.ee.ryerson.ca:8080/~elf/hack/iworm.html
[xiii] ‘Mobile Code Threats, Fact or Fiction’, Carey Nachenberg. Proceedings of the 1999 ICSA IVPC Conference.