

The Evolving Virus Threat

Mr. Carey Nachenberg
Chief Researcher, Symantec Corporation

Over the past twenty years, computer viruses have evolved from simple computer programs capable of spreading on a single PC to complex software worms which can ravage entire computer networks. Much of the evolution of new computer virus threats has come from two major sources. First, as popular new computing platforms become available virus authors seek to exploit these platforms. Second, the co-evolution of anti-virus technology has spurred the development increasingly powerful, more complex computer virus threats.

Today, our computing infrastructure is more vulnerable than ever before due to a convergence of four key factors. First, the world population of computers is more homogeneous than ever before, creating a computing “monoculture;” if one machine is susceptible to a bug, the rest are too. Second, our computing systems are more connected than ever before, enabling a fast spreading virus to rapidly infect millions of machines. Third, our computers are more programmable than ever before; with simple script programming languages, even novice users can create malicious code that can control virtually every aspect of the computer system. Finally, the hardware and software platforms used by big business and the home user are converging, lowering the bar to build and test malicious software.

This talk will study the evolution of computer viruses and consider the future virus threats we are likely to encounter. In addition, it will take an in-depth look at anti-virus technologies, present and future, to better understand how our computer countermeasures will have to change to provide adequate protection against tomorrow’s viruses. Finally, we will discuss how written and software-enforced policy changes can help to dramatically reduce the computer virus problem.

The Evolving Virus Threat

Over the past twenty years, computer viruses have evolved from simple computer programs capable of spreading on a single PC to complex software worms which can ravage entire computer networks. The vast majority of this evolution has come from two major sources. First, as popular new computing platforms become available virus authors seek to exploit these platforms and their weaknesses. Second, the co-evolution of anti-virus technology has spurred the development increasingly powerful, more complex computer virus threats. This paper will examine how both of these factors have contributed to today's virus and malicious code program and consider how both viruses and anti-virus technology will change in the future.

Computer Virus-Anti-Virus Co-evolution

In the late 1980s and early '90s, computer viruses were considered an urban myth.ⁱ At the time, only a small number of computer viruses had been written and infection was fairly uncommon. Today, the situation is much different: over 56,000 distinct virus, worm and Trojan horse strains are known! In addition to the increase in the number of viruses, these digital invaders have also become more complex and difficult to detect. These improvements can be at least partially attributed to the efforts of anti-virus products. As anti-virus technology improves and gives vendors to detect the latest virus threats, the virus writers evolve their next generation threats to exploit weaknesses in the anti-virus products to avoid detection.

This co-evolution has led to a number of remarkable improvements over the years in both virus and anti-virus technologies.

The first computer viruses were simple machine language programs that had the ability to attach and spread identical copies of themselves from program to program or disk to disk. Such viruses proved easy for anti-virus products to detect, since they always made exact copies of themselves. An anti-virus researcher could merely extract a virus signature – a small sequence of bytes found in the virus but not likely to be found in other, non-infected programs – and add this to a virus database. The anti-virus software would then use this database of signatures when scanning a computer system. If any of the files on the system contained signatures matching those in the database, the file could be quarantined and repaired.

This detection method worked well for a while, but as virus writers realized how easily their creations were being detected by anti-virus products, they quickly began to evolve. Virus authors realized that they could make their viruses more difficult to detect if the viruses did not spread exact copies of themselves. If the same virus signature could not be found in all virally-infected files, anti-virus products would have difficulty detecting the virus. This led to the creation of the *encrypted virus*.

The encrypted virus conceals its presence by encrypting the bulk of its viral logic. When an infected program is launched, the virus usurps control of the system and then uses a simple decryption subroutine to decrypt the rest of the encrypted viral logic, thus the virus' logic is only visible when it is actively running, but never when it is stored on the disk in an infected file. The virus then launches this decrypted logic to infect other program files. As new program files are infected, the virus can change the encryption slightly (by varying the encryption key) to produce new infections which bare little resemblance to the parent infection.

The encrypted virus posed new problems for anti-virus products. Clearly, since the bulk of the virus was encrypted, anti-virus researchers could not easily choose virus signatures and expect to find them in all infected files. In actuality, these simple encrypted viruses still did have a static component which does not change from infection to infection – their decryption routine logic – and this small bit of code was enough to allow anti-virus products to extract a fingerprint and detect this new generation of viruses.

Again, virus authors quickly recognized the ability of anti-virus producers to detect their newest encrypted progeny, so they move forward again and invented the polymorphic virus. The polymorphic virus is a computer virus which mutates (or generates from scratch) all non-encrypted viral logic in each new infection (polymorphic viruses still have unchanging code, but it is entirely encrypted; the virus mutates any non-encrypted code in each infection to avoid presenting an anti-virus scanner with a fixed fingerprint). This results in a computer virus which has no consistent virus fingerprint from infection to infection. Anti-virus researchers could no longer choose a static fingerprint of bytes extracted from the virus, since none of the infections used the same set of instructions.

Once again, the anti-virus establishment had to evolve or perish, and they came up with a paradigm known as *generic decryption*. The basic concept was to use CPU emulation to trick a polymorphic virus into revealing itself. By running a potentially infected file within a computer simulation, the anti-virus software could watch the program run and search for telltale signs of the virus. The technique worked, and once again, the virus writers and the anti-virus community were at parity.

Other such advances in the virus space, such as *stealth viruses*, have also been spurred by the development of more robust anti-virus offerings. A stealth virus is one which installs itself in the computer's memory and intercepts requests to scan or access infected files. When such a request is made – by an anti-virus scanner, for instance – the virus can immediately disinfect the file and provide the requesting application with the file's original contents. Once the requesting program finishes with the file, the virus can re-infect the disinfected file and save it back to the disk. This technique made it more difficult to detect viruses of any type – normal, encrypted or polymorphic – since the virus could completely conceal its presence any time an infected file was scanned.

So where have the years of leapfrogging taken us? Today, the anti-virus community has developed a robust set of tools for detecting viruses in files, in system memory, and in network packets. We have largely automated many of the repetitive analysis tasks to reduce our turnaround time. And we are beginning to deploy the next generation of automated, closed-loop anti-virus systems (this will be the focus of the last part of this paper).

In contrast, the virus writers have achieved a state of the art where their most complex creations take days or weeks to analyze and develop cures for; however, the average computer virus can be processed in minutes.

While these technological advances have materialized because of virus-anti-virus co-evolution, many of the advances in the malicious code space have been driven by new platforms and new computing infrastructures. With each new operating system, Windows 3.1, Windows 95, Windows NT, Linux, etc. have come new virus threats. Similarly, applications that work within these operating systems have also begun to offer programmable functionality and we have seen a similar evolution of threats in this space. Virus writers have developed macro viruses targeting Word for Windows 6.0, Office 95, Office 97, and now the Office 2000 applications.

While new platform emergence has had a significant impact on the quantity and variety of malicious code threats, we believe that a set of convergent factors have materialized over the last few years that will have an even larger impact on malicious code and on business, government and on laypeople in computer-dependent societies. The second part of this paper will discuss how our current computing environment and communications infrastructure has impacted and will continue to impact the progression, distribution and impact of malicious code.

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

Today's Influencing Factors

Computing Infrastructure Homogeneity

The homogeneity of computing hardware, operating systems, application software and communications platforms will be the single largest enabler for computer viruses, worms and Trojan horses.

Today, more than ninety percent of the world's computers are running the Windows operating system on Intel-based hardware. ⁱⁱAn equally high percentage of end-users use standard SMTP Internet e-mail, and many large corporations are standardizing on e-mail systems like Microsoft Outlook and Lotus Notes. In the word processing area, the Microsoft Office suite enjoys a virtual monopoly for home users and business users alike. ⁱⁱⁱEssentially, each of our desktops have a *genetically* similar software and hardware makeup.

In agriculture, such a homogeneous environment is called a monoculture, and is generally known to have negative consequences. If a farmer sows a single variety of crop on their land (for instance, to increase their yield of that crop), they subsequently increase the susceptibility of their entire crop to disease. If the disease affects one plant, it can quickly and easily spread to genetically-similar neighboring 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. With a growing, homogeneous set of hosts, a virus doesn't have far to travel before it finds fertile ground to launch another infection.

Already, we have seen thousands of macro viruses attacking the Microsoft Office platform. Several high-profile worms (network-aware viruses) have used the Outlook and SMTP e-mail programs to spread themselves. ^{iv} Furthermore, more than 99% of all computer viruses are designed to spread on the DOS/Windows/Intel platform.

Clearly, hardware standardization has given us a huge benefit; it allows corporations, government and end-users to standardize their software and hardware systems, reduce troubleshooting and technical support costs, and reduce replacement costs. For software developers, having a single monolithic platform decreases the costs of software development; we only have to develop for a single platform instead of ten different platforms. It also improves the stability of software, as much as this is counterintuitive to those of us that use computers every day.

Unfortunately, these benefits have transformed our society into one which is highly dependent on a single computing environment. And while PCs started out as corporate tools, they now pervade the most secure government systems, financial institutions, nuclear power plants, our intelligence community, as well as the home. Long the users of proprietary systems, the government and financial institutions now uses hardware and software that is the same as is found in the home, in Iraq and in other rogue states. Given that the vast majority of government, businesses and home users use the same platform, a digital Armageddon is far from being a fairytale.

Ubiquitous Programmability

The ubiquitous programmability of the Windows operating system has made it possible to write viruses and worms without complex programming.

Who would have thought that the word processor or spreadsheet would be the single most successful host 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 current office product lines, software vendors have made the Office products the platform of choice for virus, worm and Trojan horses.

Users can write simple programs, called macros, and attach them to their documents and spreadsheets. These programs, written in Visual Basic, an easy-to-use, BASIC-like programming language, can perform useful function such as spell checking your document, summing tables in a spreadsheet, or auto-e-mail completed expense reports to the finance department. In addition, macros can be copied or can copy themselves to other documents. This feature allows users to easily share or install useful macros across the corporation. However, by allowing macros to copy themselves from one document to another, the Office platform also becomes extremely susceptible to computer virus threats. Today, more than 80% of all computer virus incidents (actual infections found by users or corporations) are due to macro viruses in Word and Excel.

Unfortunately, these 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 vendors had limited the Office macro systems so that macros could only interact with other components of the Office suite, this would have limited malware creations to viruses that could spread themselves only within the confines of Office documents like Word and Excel.

Unfortunately, the marriage of the Office macro programming languages, and 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 software application, he or she can make the functionality of the application available to the rest of the software applications running on the system (and not just to the user) via the COM system. Other programmers can subsequently design their software logic to leverage the functionality provided by the first COM-enabled program.

For example, using COM, Microsoft Outlook enables 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. Using this facility, a user could write an expense reporting application in BASIC or C that would make use of the Outlook e-mail program via software APIs. The programmer could program their application 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, knowing e-mail protocols, etc.

Clearly, COM technology has been a huge enabling technology for programmers. The typical programmer can now make extremely rich software applications by leveraging other components on the system. If you don't know how to send e-mail, no problem! Just call upon the e-mail program to do it for you. Not sure how to use the printer? Use software API calls to talk to the COM printer object and it will take care of the printing details for you.

In a logical move which yielded tremendous value for programmers, vendors made it possible for Office macros to leverage the same powerful features of COM. With this newly added functionality and the simplicity of the BASIC-like macro programming language supported by Office products, almost any competent user could pick up a book and develop powerful macro programs that have the ability to do far more than summing tables in a spreadsheet! These COM-enabled macros can examine and modify the entire host computer system, and more worrisome, they can leverage the built-in communications facilities of the computer to spread over the worldwide network of homogeneous machines.

Complete Connectedness

The increasing connectivity and enumerability of the today's communication systems permit worms to spread faster, and to more machines than ever before.

Until recently, the spread rate of computer viruses was limited to how fast computer users exchanged infected files in e-mail, through file servers, on floppy diskettes, etc. Traditional computer viruses (that

don't intentionally spread over networks) 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.

Since users share information more than they share programs (at least in corporate and government settings), user-initiated e-mail has enabled macro viruses to spread far more quickly than binary viruses (such as DOS and Windows viruses).^v However, the typical user sends only a handful of documents to a small set of co-workers during the average week. So, while a macro virus might quickly be transmitted half way across the world from the United States to Europe, it will only spread to a small number of users over a period of days or weeks. Then those target users must open the infected document, edit some other documents, and send them out to their co-workers. This whole human-centric process is cumbersome and restricts how rapidly these viruses can be spread. Fortunately, by the time a new macro virus can infect even a handful of users, anti-virus companies can respond with a fingerprint update and prevent any further spread.

Unfortunately, with more computers on e-mail and the Internet than ever before, worms can now spread more quickly than any traditional virus. And as mentioned, the homogeneous, ubiquitous, COM-accessible communications mechanisms makes writing such a worm a snap. Why should a virus wait to be sent by the user as an e-mail attachment when it can send itself? Why only send itself to a handful of computers when it could send itself to an entire organization? 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 components of the computer system – whether the network or e-mail – 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 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 Explore.Zip worm used exactly such a mechanism to spread itself over corporate networks, and was extremely successful.^{vi}

As mentioned, today's computer networks are more connected than ever before; a user or program can send an e-mail from any computer directly to any other computer in the entire network in seconds. However, a second facet of our communications systems make them even more susceptible to attack: modern software directories permit the enumeration of each and every node connected to the network. For example, corporate groupware products like Lotus Notes and Microsoft Exchange allow users to view every single e-mail user in the entire corporation – and if they like, the user can send e-mail to each and every one of these addresses.

Besides groupware directories, a number of other directory sources exist which would permit a malicious attacker *or software agent* to obtain a list of targets. For example, corporate LDAP^{vii} directories, Internet search engines, and public mailing lists (so-called listservs) all provide the means for enumerating and targeting potentially millions of users!

The ability of users or software programs to enumerate and target specific computer systems makes computer worms considerably more troubling. First, a malicious attacker could use these publicly available directories to select an initial distribution list: all CIOs at fortune 500 companies, all CFOs at financial institutions, etc. Second, once within a corporation or government entity, the computer worm can use the same directory mechanisms to enumerate targets and spread itself. While the corporate e-mail directory may not be available outside of the firewall, once inside a corporation, a worm can easily access this information and use it to spread. This is exactly how worms like Melissa propagated so rapidly. In many cases, Melissa exploited the corporate directory to spread to hundreds of thousands of mailboxes in hours!^{viii}

Technology Migration to the Home

The migration of the PC from the corporation to the home, and the further adoption of home networking lowers the bar for virus development... and testing.

“Virus authors go with what virus 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, at the time of this writing, we have seen a number of viruses and worms leverage Microsoft Outlook and Outlook Express to send themselves. These e-mail programs are widely used in both the corporation and at home (and they share the same COM programming interface). Conversely, we have seen no worm-based attacks that leverage Lotus Notes to spread themselves. Notes, unlike Microsoft Outlook, is used exclusively in corporations, and is a less available technology for virus/worm authors to tinker with at home. While we fully expect to see Notes worms in the future, the lack of a consumer-oriented Notes client has undoubtedly retarded the development of Notes-centric threats.

As more companies adopt products like Outlook, Eudora and other e-mail programs, virus writers (who are predominantly 14-24 year-old males^{ix}) will have all of the components necessary to build and test their viral creations in their own home. Today, it is not uncommon to find home networks of several machines and all the components can be bought cheaply at the local computer superstore. Regrettably, these local networks provide the virus writer with everything they need to develop and then test their computer worms. More than ever before, the authors of these worms have access to the hardware and software platforms employed by businesses and the government. ^x

The popular Linux platform will also likely become an increasingly attractive platform for virus development. Since Linux is offered free of charge, source code and all, virus writers will have easy access to documentation, operating system source code, and everything they need to build and test their viruses. Linux runs on the same computers that millions of users already own, and it is well regarded in the programming and hacking communities. Contrast this with Solaris; while Solaris is a very popular UNIX platform, it is much less available to the common home user, and consequently, we expect to have fewer threats targeting it. ^{xi}

Now that we have examined the reasons behind our recent rash of computer worm epidemics, let's examine what the near-term future holds for computer viruses and worms:

Predictions For Tomorrow

Broadband in the Home

As more users adopt broadband communications technologies in the home, we expect that the incidence of computer worms targeted at home users and small businesses will grow swiftly. *This will also seriously affect telecommuters in both government and industry.* Today, home users are assigned dynamic network addresses each time they login to the Internet. Since users log on and off frequently, this makes it very difficult for a worm to target such a machine and spread to it.

However, as users migrate to broadband technologies such as cable modems or Digital Subscriber Line (DSL), they will increasingly have constant, static connections to the Internet, making their computer an easy target. Computer hackers or roving worms will be able to easily enumerate home user Internet addresses and use them to attack these machines. Once they have a foothold on the home computer, they can quickly spread through the VPN and onto the corporate or government network. For example, if a home user becomes infected by a Word for Windows macro virus, they may easily infect their work

documents and then transfer these to their corporate PC over the VPN. Similarly, a worm like ExploreZip could potentially spread through the VPN onto other visible machines on the corporate network.

Additionally, we expect that as more users adopt broadband technologies in the home, consumer-oriented connected applications will grow in popularity. Today, products like PointCast™ dot corporate desktops, but are less appealing for the home user who has a lower-speed connection to the Internet. However, as broadband technologies become more popular, these connected applications will grow in popularity; real-time stock tickers, personal web servers, search agents, and “instant message”/chat programs will be running on every desktop. And, as we have seen in the Office application space, vendors will start adding macro/programming support to these applications to extend their capabilities for power users.

While these connected applications will improve the quality of the computing experience, each also contributes security risks for the home user. We expect that the next generation of computer worms will exploit these connected (and often not security-conscious) applications and use them as back doors into home systems and then into the enterprise. 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 popular worms of 1999.

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.

General Sophistication Will Increase

While the vast majority of the older viruses were written in Assembly language, a low-level programming language that is arcane and difficult to use, an increasing number of the latest worms and Trojan horses are built using more modern (high level) programming languages and tools. These high level worms and Trojan horses are more difficult to analyze (because optimizing compilers often obscure the code logic to improve efficiency), utilize more complex techniques to spread and better leverage the operating system and all available exploits of the target platform. While creating a detection and cure for these threats is still relatively straightforward, their increased complexity has dramatically increased the amount of time it takes to fully disassemble and analyze these pathogens. This has, and will continue to put a strain on security experts who already have a full plate.

Information Stealers

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.^{xii} While older malicious code threats would delete files or format hard drives, the new *payloads* of choice steal information and send it to an attacker, or allow an attacker to remotely control a compromised computer.^{xiii} ^{xiv} Imagine a password-stealing worm which recorded all keystrokes and sent them to an attacker. It could watch users log into their government or business computers and obtain passwords and other critical information.

A good example of this is the PrettyPark worm. If this worm finds its way onto a computer and then acquires a connection to the Internet. The worm will connect to the IRC^{xv} chatting service and wait for commands from an attacker. An attacker can then send commands to the PrettyPark worm and perform any number of malicious actions on the compromised computer including stealing information and deleting

files. Luckily, 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 and government information.

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 and therefore should be considered a potential threat. ExploreZip, Happy99 and many Windows viruses could be built and deployed as ActiveX components instead of standard Windows programs (there is little distinction).

However, there is one aspect of ActiveX which limits its viability as a virus, worm or Trojan horse platform. ActiveX components must be digitally signed before they can be used in most web browsers. We believe that this will limit the number of ActiveX worms actually deployed in the wild. Why? Because when a user signs his or her ActiveX component, he or she identifies themselves as the creator and becomes liable and traceable as the creator of the malicious code. The explicit liability afforded by the digital signature has arguably acted as a strong deterrent – at least for the casual virus writer. Sadly, the liability afforded by digital signing will not likely deter a motivated criminal or terrorist from a rogue nation.

Java worms are unlikely for two reasons. First, Java's strong security prevents Java applets from spreading themselves or accessing the local computer resources. Second, while digitally signed Java applets can access the host computer system and potentially spread themselves, the liability issues described above will likely limit the number of wild threats.^{xvi}

How will the next generation of worms spread unhindered?

As we have seen, computer viruses and anti-virus technologies have experienced a co-evolution over the years. 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. However, over the next few years we expect to see a similar evolution of worm technology that parallels the evolution seen in the computer virus world.

Polymorphic Worms

Worms like Melissa and ExploreZip spread themselves through e-mail by sending identical copies of themselves from one computer to another over the network. For example, every time ExploreZip sends an e-mail to another user, it attaches the exact same 210,432 bytes of executable code. Moreover, 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.

Since these worms don't change themselves as they spread, resourceful system administrators can use these attributes to help them weed out these worms from their e-mail servers. By writing simple scripts or e-mail rules that looked for these fixed text messages, many administrators were able to purge Melissa and ExploreZip infections from their e-mail systems.

What is the likelihood that the next batch of worms will be this simple? Small. As we have seen in the past, virus writers have tried to outmode anti-virus vendors at every turn. Once they recognize how easily Melissa and ExploreZip can be thwarted by today's anti-virus software, they will undoubtedly make their next creations more difficult to detect.

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

Furthermore, the next iteration of worms could easily mutate themselves, as hundreds of traditional viruses do today. 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 third-generation technologies for detecting polymorphic viruses, many of our existing technologies will be helpless against large numbers of randomly generated worm/virus threats.

Retro Worms, Stealth Worms and Stubborn Worms

Retroviruses, a misnomer from biology, are computer viruses that attack anti-virus software to prevent themselves from being detected. Retroviruses delete anti-virus definition files, disable memory resident anti-virus protection and attempt to disable anti-virus software in any number of ways. We expect that some or all of next generation computer worms will also use these exploits.

Consider what would have happened if ExploreZip deleted the anti-virus software from every computer it infected? In addition to distributing new virus definitions to thousands of machines, the administrator would also have to redistribute and re-install the entire anti-virus application. While a traditional retro-virus might reach only a small percentage of corporate desktops, a worm like ExploreZip could quickly devastate a majority of the desktops' anti-virus protection.

While Stealth worms could hide themselves from anti-virus products, so-called stubborn worms could prevent themselves from being unloaded from an infected system. There are a number of ways for a virus or worm to accomplish this and several are described in Peter Szor's paper found in the Virus Bulletin 1999 proceedings^{xvii}.

Wireless Worms

While handheld, wireless devices are in their infancy in the United States, they are gaining widespread acceptance in Europe and Japan. Today's cellular phones have most of their software content embedded in read-only chips, and do not have the ability to run or download arbitrary software applications; however,

this is likely to change in the next two to three years. As these devices are given the capacity to download and run arbitrary software applications, there is no reason why they cannot also host malicious mobile code.

How would such a wireless worm work? If we assume that wireless devices can easily locate and talk with other devices using either standard data-oriented cellular phone numbers or via network 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/network 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 might read the message and run the worm, 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. 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.

So what is the next evolutionary step for anti-virus software

Given that today’s computer viruses and worms can spread more rapidly than ever before, it follows that anti-virus software vendors must develop systems that can automatically analyze new threats, develop a cure and distribute that cure faster than ever before. If such a system can spread the cure faster than the virus or worm can spread itself, it can help to slow these fast-spreading threats. In addition, in the future, we expect that anti-virus technologies will no longer be deployed as stand-alone applications. Rather, they will be deployed with other componentry, such as intrusion detection software, personal firewalls, etc. By melding these disparate technologies, we can achieve better protection for the enterprise and more effectively protect against new and unknown threats.

One implementation of such a system could work in the following way:

1. Anti-virus, intrusion detection or firewall software on the desktop, file server, gateway, or group-ware server detects a potential new or unknown virus with *heuristic technology*. Heuristics are behavior-based technologies that can detect the suspicious behavior of new and unknown threats.
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.) and to isolated networked computers.
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 integrates the fingerprint and the cure into the official 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.

Conclusions

Network aware viruses, or computer worms have grown to become the fastest spreading and most costly malicious code threats of this decade. These worms leverage our newly created infrastructural homogeneity, the total programmability and lack of security on Windows/Intel machines, and the increasing connectivity of the Internet to spread rapidly through the enterprise and government. Furthermore, the availability of powerful PCs in virtually every household has lowered the bar for development *and testing* of these threats. As more users obtain broadband access to the Internet and work from home, we expect that whole new classes of worms will emerge and target these users. Thus, the home desktop PC or laptop becomes yet another corporate asset that needs to be secured.

Today's paradigms are only marginally effective at protecting against the fast-spreading worm and it is clear that these solutions will need to evolve to provide adequate protection for customers. As virus writers have done in the past, we expect that they will evolve their worms to be even more difficult to detect and remove.

With all of these changes, home users, corporations and government entities need to seriously reconsider their security policies, and anti-virus companies need to start working on the next generation of anti-virus protection. The game continues.

ⁱ IBM Research, 1997. researchweb.watson.ibm.com/internettech/cover0397.html

ⁱⁱ Findings of Fact, United States District Court For The District of Columbia, 1999. usvms.gpo.gov/findfact.html

ⁱⁱⁱ USA Today, Jan 22, 1998.

^{iv} Symantec AntiVirus Research Center, 1999. www.symantec.com/avcenter/venc/data/mailissa.html, www.symantec.com/avcenter/venc/data/worm.explore.zip.html

^v ICSA 1998 Computer Virus Prevalence Survey, 1998, p. 13

^{vi} 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.

^{vii} LDAP standard for Lightweight Directory Access Protocol. An LDAP directory is a directory system which *speaks* the LDAP protocol. These directories can store user information, passwords, e-mail addresses, phone numbers, etc.

^{viii} C. Nachenberg, 'Computer Parasitology', Proceedings of The Ninth International Virus Bulletin Conference, 1999. p. 7

^{ix} S. Gordon, 'The Generic Virus Writer', Proceedings of The Fourth International Virus Bulletin Conference, 1994.

^x S. Gordon, 'The Generic Virus Writer II', Proceedings of The Sixth International Virus Bulletin Conference", 1996.

^{xi} This is not to say that a determined attacker won't target the Solaris platform; however, based on its availability to today's virus writers and its compatibility with existing hardware, we expect virus writers to target this platform in the future.

^{xii} Symantec AntiVirus Research Center, 1999.

www.symantec.com/avcenter/venc/data/prettypark.worm.html,

www.symantec.com/avcenter/venc/data/backorifice.html

^{xiii} S. Gordon, 'When Worlds Collide: Information Sharing for the Security and Anti-virus Communities', Proceedings of The Ninth International Virus Bulletin Conference, 1999.

^{xiv} Symantec AntiVirus Research Center, 1999.

www.symantec.com/avcenter/venc/data/prettypark.worm.html,

www.symantec.com/avcenter/venc/data/backorifice.html

^{xv} IRC stands for Internet Relay Chat, and is a multi-user, Internet-based chatting service.

^{xvi} D. Chess and J. Morar, 'Is Java Still Secure?', Proceedings of The Ninth International Virus Bulletin Conference, 1999.

^{xvii} P. Szor. 'Memory Scanning Under Windows NT', Proceedings of The Ninth International Virus Bulletin Conference.