The Evolution of Viruses and Worms

Thomas M. Chen
Dept. of Electrical Engineering
SMU
PO Box 750338
Dallas, TX 75275-0338 USA
Tel: 214-768-8541
Fax: 214-768-3573
Email: tchen@engr.smu.edu

Jean-Marc Robert
Alcatel Canada Inc.
600 March Road
Ottawa, Ontario, Canada K2K 2E6
Tel: (613) 784-5988
Email: jean-marc.robert@alcatel.com

Abstract - Computer viruses and network worms have evolved through a continuous series of innovations, leading to the recent wave of fast-spreading and dangerous worms. A review of their historical development and recent outbreaks leads to a number of observations. First, while viruses were more common than worms initially, worms have become the predominant threat in recent years, coinciding with the growth of computer networking. Second, despite widespread use of firewalls and other network security equipment, worm outbreaks still occur and will likely continue to be a threat for the near future. Third, recent worms are appearing as a series of quick successive variants. Unlike the independent efforts of early viruses, these variants suggest an increasing level of coordination among worm creators. Fourth, recent worms have shown capabilities to spread faster and exploit more infection vectors. This trend implies a more urgent need for automated, coordinated protection measures. Finally, more dangerous payloads are becoming commonplace. This suggests that worm creators are using worms for other objectives than simply infection, such as data theft and setting up denial of service networks.

1. Introduction

Computer viruses and worms are characterized by their ability to self replicate. The modern computer virus was conceived and formalized by Fred Cohen as a USC graduate student in 1983. Cohen wrote and demonstrated the first documented virus in November 1983 [1]. Like biological viruses, computer viruses reproduce by taking advantage of the existing environment. A biological virus consists of single or double-stranded nucleic acid (DNA or RNA) surrounded by a protein shell (capsid). The capsid gives specificity to bond with those particular hosts with matching surface receptors, while the inner nucleic acid gives infectivity or potency to subvert the infected host's cellular machinery. A virus is incomplete and inactive outside of a living cell but becomes active within a host cell by taking over the host's metabolic machinery to create new virus particles that spread the infection to other cells.

Computer viruses replicate themselves by attaching their program instructions to an ordinary "host" program or document, such that the virus instructions are executed during the execution of the host program. As a simplification of Cohen's definition, a basic computer virus can be viewed as a set of instructions containing at least two subroutines attached somehow to or within a host program or file [2]. The first subroutine of the virus carries out the infection by seeking out other programs and attaching or overwriting a copy of the virus instructions to those programs or files [3]. The method of infection or propagation is called the "infection vector" (borrowed from epidemiology). The second subroutine of the virus carries the "payload" that dictate the actions to be executed on the infected host. The payload could be almost anything in theory, for example, deletion of data, installation of backdoors or DoS (denial of service) agents, or attacks on antivirus software [4]. An optional third subroutine could be a "trigger" that decides when to deliver the payload [5].

Computer networks have created a fertile environment for worms, which are related to viruses in their ability to self-replicate but are not attached parasitically to other programs. Worms are stand-alone automated programs designed to exploit the network to seek out vulnerable computers to infect with a copy of themselves. In contrast to viruses, worms are inherently dependent on a network and not dependent on any human action (such as to execute a program infected with a virus). Worms have become more prevalent since Internet connectivity has become ubiquitous. The Internet increases the vulnerability of all interconnected computers by making it easier for malicious programs computers to move among computers.

Recent worm outbreaks, such as the Blaster worm in August 2003 and the SQL Sapphire/Slammer worm in January 2003, have demonstrated that networked computers continue to be vulnerable to new attacks despite the widespread deployment of antivirus software, firewalls, intrusion detection systems, and other network security equipment. We review the historical development of viruses and worms to show how they have evolved in sophistication over the years. The history of virus and worm evolution is classified into four “waves” spanning from the Shoch and Hupp worms in 1979 to the present day (the term “generations” is less preferred because viruses and worms of one wave are not direct descendents from an earlier wave). This historical perspective leads to a number of observations about the current state of vulnerability and trends for possible future worm attacks.

We classify the evolution of viruses and worms into the following waves:

• first wave from 1979 to early 1990s

• second wave from early 1990s to 1998

• third wave from 1999 to 2001

• fourth wave from 2001 to today.

These waves represent periods where new technological trends began and appeared in a significant number of cases. For example, the third wave was dominated by so-called mass e-mailers: viruses and worms that exploited e-mail programs to spread. The classification dates are approximate and not meant to imply that waves can be separated so distinctly. For example, mass e-mailers characteristic of the third wave are still common during the fourth wave.

2. First Wave: Early Viruses and Worms

The first wave of viruses and worms from roughly 1979 to early 1990s were clearly experimental. The early viruses were commonly boot-sector viruses and targeted mostly to MS DOS. The early worms were prone to programming bugs and typically hard to control. The term “worm” was created by John Shoch and Jon Hupp at Xerox PARC in 1979, inspired by the network-based multi-segmented "tapeworm" monster in John Brunner's novel, *The Shockwave Rider*. Shoch and Hupp used worm to refer to any multi-segmented computation spread over multiple computers. They were inspired by an earlier self-replicating program, Creeper written by Bob Thomas at BBN in 1971, which propelled itself between nodes of the ARPANET. However, the idea of self-replicating programs can be traced back as early as 1949 when the mathematician John von Neumann envisioned specialized computers or “self-replicating automata” that could build copies of themselves and pass on their programming to their progeny [6].

Shoch and Hupp invented worms to traverse their internal Ethernet LAN seeking idle processors (after normal working hours) for distributed computing [7]. Since the worms were intended for beneficial uses among cooperative users, there was no attempt at stealth, intrusion, or malicious payload. Even under cooperative conditions however, they observed that worm management was a key problem to ensure that worm growth could be reliably contained. Their worms were designed with limited lifetimes, and responsive to a special packet to kill all worms. Despite these safeguards, one of the worm programs mysteriously ran out of control and crashed several computers overnight.

In 1983, Fred Cohen conceived, wrote and demonstrated the first computer virus while a graduate student at USC. The 1986 DOS-based Brain virus, supposedly written by two Pakistani programmers, was interesting in its attempt to stealthily hide its presence by simulating all of the DOS system calls that normally detect viruses, causing them to return information that gave the appearance that the virus was not there.

In 1987, the "Christma Exec" virus was among the first to spread by e-mail among IBM mainframes. It is also an early example of social engineering where the user is tricked into executing the virus because it promised to draw a Christmas tree graphic. The worm does produce a Christmas card graphic on the computer screen (drawn using a scripting language called Rexx) but also sends a copy of itself in the user's name to his list of outgoing mail recipients. The recipients believe the e-mail is from the user so they open the e-mail.

On November 2, 1988, the famous Morris worm disabled 6,000 computers in a few hours (constituting 10 percent of the Internet at that time) [8]. The worm was written by a Cornell student, Robert Morris Jr. Investigations conducted later (ending in his conviction in 1990) concluded that his motives for writing the worm were unknown but the worm was not programmed deliberately for destruction. Instead, the damage appeared to be caused by a combination of accident and programming bugs. It was detected and contained only because an apparent programming error caused it to re-infect computers that were already infected, resulting in a noticeable slowdown in infected computers. It was among the first to use a combination of attacks to spread quickly: cracking password files; exploiting the debug option in the Unix "sendmail" program; and carrying out a buffer overflow attack through a vulnerability in the Unix "finger" daemon program.

In October 1989, the WANK (Worms Against Nuclear Killers) worm apparently learned from the Morris worm and infected VMS computers on DECnet. It spread using e-mail functions, exploited default system and field service accounts and passwords for access, and tried to find accounts where the user name and password were the same or the password was null.

3. Second Wave: Polymorphism and Toolkits

The second wave in the approximate period from the early 1990s to 1998 saw much more activity in viruses than worms, although the technical advances in viruses would effect the later evolution of worms. In this period, viruses began to move from Microsoft DOS to Windows as the primary target; cross-platform macro viruses appeared; mass creation of polymorphic viruses became easy; and a trend towards e-mail as the preferred infection vector began.

In the late 1980s, the idea of using encryption to scramble the appearance of a virus was motivated by the fact that antivirus software could detect viruses by scanning files for unique virus signatures (byte patterns). However, to be executable, an encrypted virus must be prepended with a decryption routine and encryption key. The decryption routine remains unchanged and therefore detectable, although the key can change which scrambles the virus body differently. Polymorphism carries the idea further to continuously permute the virus body. A polymorphic virus reportedly appeared in Europe in 1989. This virus replicated by inserting a pseudorandom number of extra bytes into a decryption algorithm that in turn decrypts the virus body. As a result, there is no common sequence of more than a few bytes between two successive infections.

Polymorphism became a practical problem in 1992 when a well-known hacker, Dark Avenger, developed a user-friendly Mutation Engine to provide any virus with polymorphism [9]. Other hackers soon followed with their own versions of mutation engines with names such as TPE, NED, and DAME. In 1994, Pathogen and Queeg were notable polymorphic DOS-infecting viruses that were produced by Black Baron's SMEG (Simulated Metamorphic Encryption enGine) [5].

The “virus creation lab” was a user-friendly, menu-driven programming toolkit that allowed hackers with little programming skill to easily generate hundreds of new viruses [9]. Other virus creation toolkits soon followed such as PS-MPC. Perhaps the best known product of a virus toolkit was the 2001 Anna Kournikova virus [10]. This virus was carried in an e-mail attachment pretending to be a JPG picture of the tennis player. If the Visual Basic Script attachment is executed, the virus e-mails a copy of itself to all addresses in the Outlook address book.

The 1995 Concept virus was the first macro virus, written for Word for Windows 95. It infected Word's "normal.dot" template so that files were saved as templates and ran the infective AutoOpen macro. An infected file was accidentally shipped on a Microsoft CD called "Microsoft Windows95 Software Compatibility Test." Later, Microsoft UK shipped an infected file on another CD called "The Microsoft Office 95 and Windows95 Business Guide." The vast majority of macro viruses are targeted to Microsoft Office documents. Macro viruses have the advantages of being easy to write and cross-platform. However, most people now know to disable macros in Office so macro viruses have lost their popularity.

4. Third Wave: Mass E-Mailers

The third wave spanning roughly 1999 to late 2000 is highlighted by a large number of mass e-mailers, beginning with the “Happy99/Ska” worm. E-mail continues to be a very popular infection vector today. In January 1999, the Happy99 worm spread by e-mail with an attachment called Happy99.exe [11]. When the attachment is executed, it displayed fireworks on the screen to commemorate New Year's Day 1999, but secretly modified the WSOCK32.DLL file (the main Windows file for Internet communications) with a Trojan horse program that allowed the worm to insert itself into the Internet communications process. The original WSOCK32.DLL file is renamed to WSOCK32.SKA. Every e-mail sent by the user generated a text-less second copy that carried the worm to the same recipients.

In March 1999, the Melissa macro virus spread quickly to 100,000 hosts around the world in 3 days, setting a new record and shutting down e-mail for many companies using Microsoft Exchange Server [12]. It began as a posting on the Usenet newsgroup "alt.sex" promising account names and passwords for erotic web sites. The attached Word document actually contained a macro that used the functions of Microsoft Word and the Microsoft Outlook e-mail program to propagate. Up to that time, it was widely believed that a computer could not become infected with a virus just by opening e-mail. When the macro is executed in Word, it first checks whether the installed version of Word is infectable. If it is, it reduces the security setting on Word to prevent it from displaying any warnings about macro content. Next, the virus looks for a certain Registry key containing the word "Kwyjibo" (apparently from an episode of the television show, "The Simpsons"). In the absence of this key, the virus launches Outlook and sends itself to 50 recipients found in the address book. Additionally, it infects the Word “normal.dot” template using the VBA macro auto-execute feature. Any Word document saved from the template would carry the virus [13].

The PrettyPark worm became widespread in the summer of 1999. It propagates as an e-mail attachment called "Pretty Park.exe." The attachment is not explained but bears the icon of a character from the television show, "South Park." If executed, it installs itself into the Windows system directory and modifies the registry to ensure that it runs whenever any .EXE program is executed. This can cause problems for antivirus software that runs as an .EXE file. Furthermore, the worm e-mails itself to addresses found in the Windows Address Book. It also transmits some private system data and passwords to certain IRC (Internet relay chat) servers. Reportedly, the worm also installs a backdoor to allow a remote machine to create and remove directories, and send, receive, and execute files.

In June 1999, the ExploreZip worm appeared to be a WinZip file attached to e-mail but was not really a zipped file [14]. If executed, it would display an error message, but the worm secretly copied itself into the Windows systems directory or loaded itself into the registry. It sends itself via e-mail using Microsoft Outlook or Exchange to recipients found in unread messages in the inbox. It monitors all incoming messages and replies to the sender with a copy of itself.

In early 2000, the BubbleBoy virus (apparently named from an episode of the television show, "Seinfeld") demonstrated that a computer could be infected just from previewing e-mail without necessarily opening the message. It took advantage of a security hole in Internet Explorer that automatically executed Visual Basic Script embedded within the body of an e-mail message. The virus would arrive as e-mail with the subject "BubbleBoy is back" and the message would contain an embedded HTML file carrying the viral VB Script. If read with Outlook, the script would be run even if the message is just previewed. A file is added into the Windows start-up directory, so when the computer starts up again, the virus e-mails a copy of itself to every address in the Outlook address books. Around the same time, the KAK worm spread by a similar exploit.

In May 2000, the fast-spreading Love Letter worm demonstrated a social engineering attack, which would become common in future mass e-mailing worms [15]. It propagates as an e-mail message with the subject, "I love you" and text that encourages the recipient to read the attachment. The attachment is a Visual Basic Script that could be executed with Windows Script Host (a part of Windows98, Windows2000, Internet Explorer 5, or Outlook 5). Upon execution, the worm installs copies of itself into the system directory and modifies the registry to ensure that the files were run when the computer started up. The worm also infected various types of files (e.g., VBS, JPG, MP3, etc.) on local drives and networked shared directories. When another machine is infected, if Outlook is installed, the worm will e-mail copies of itself to anyone found in the address book. In addition, the worm makes an IRC connection and sends a copy of itself to anyone who joins the IRC channel. The worm had a password-stealing feature that changed the startup URL in Internet Explorer to a web site in Asia. The web site attempted to download a Trojan horse designed to collect and e-mail various passwords from the computer to an address in Asia.

In October 2000, the Hybris worm propagated as an e-mail attachment [16]. If executed, it modifies the WSOCK32.DLL file in order to track all Internet traffic. For every e-mail sent, it subsequently sends a copy of itself to the same recipient. It is interesting for its capability to download encrypted plug-ins (code updates) dynamically from the "alt.comp.virus" newsgroup. The method is sophisticated and potentially very dangerous, since the worm payload (destructive capability) can be modified at any time.

5. Fourth Wave: Modern Worms

The fourth wave of modern worms began in 2001 and continues today. They are represented by worms such as Code Red and Nimda that demonstrate faster spreading and a new level of sophistication including

• blended attacks (combined infection vectors)

• attempts at new infection vectors (Linux, peer-to-peer networks, instant messaging, etc.)

• dynamic code updating from the Internet

• dangerous payloads

• active attacks against antivirus software.

Linux was first targeted by the Bliss virus in 1997. In early 2001, Linux was hit by the Ramen and Lion worms. The Lion worm seemed to be unusually dangerous. After infection of a new victim, the worm carries out several actions. The worm sends e-mail containing password files and other sensitive information, e.g., the IP address and username of the system and bundles the contents of the "/etc/password," "/etc/shadow," and "/sbin/ifconfig" files into the mail.log file before transmission. The worm installs the binary toolkits "t0rn" and the distributed denial of service (DDoS) agent "TFN2K" (Tribal Flood Network 2000). The t0rn rootkit deliberately makes the actions of the worm harder to detect through a number of system modifications to deceive the utility "syslogd" from properly capturing system events. The worm creates a directory named "/dev/.lib" and copies itself there. The “/etc/host.deny” file is deleted to make any protection offered by TCP Wrappers useless against an outside attack. The worm also installs backdoor root shells on TCP port 60008 and TCP port 33567. A Trojanized version of SSH on port 33568 changes the port setting to listening. This port setup could be done to launch future DDoS attacks using the TFN2K agent.

In February 2001, the Gnutelman/Mandragore worm infected users of Gnutella peer-to-peer networks by disguising itself as a searched file.

In May 2001, the Sadmind worm spread by targeting 2 separate vulnerabilities on 2 different operating systems, and set a precedent for subsequent viruses that could combine multiple attacks [17]. It first exploited a buffer overflow vulnerability in Sun Solaris systems and installed software to carry out an attack to compromise Microsoft IIS web servers.

A buffer overflow vulnerability in Microsoft's IIS web servers was announced on June 18, 2001, referred to as the Index Server ISAPI vulnerability. About a month later on July 12, 2001, the first Code Red I (or .ida Code Red) worm was observed to exploit this vulnerability [18]. Upon infection, the worm set up itself in memory and generated up to 100 new threads, each an exact replica of the original worm. The first 99 threads were assigned to spread the worm by infecting other IIS servers. The worm generated a pseudorandom list of IP addresses to probe but used a static seed which (unintentionally) resulted in identical lists of IP addresses generated on each infected host. As a result, the spread was slow because probes repeatedly hit previously probed or infected machines, although 200,000 hosts were infected in 6 days. A side effect was congestion between these hosts, causing a DoS effect. The last 100th thread was assigned to check the language of the IIS system; if English, then the worm would deface the system’s web site with the message “Hacked by Chinese.” Next, each worm thread checked for the file “c:\notworm” and finding it, the worm would go dormant, which seemed to be a mechanism to limit the worm spread. Finally, each worm thread checked the date. If it was past the 20th of the month, the threads would stop scanning and instead would launch a DoS attack against port 80 of “www.whitehouse.gov.” On the 27th of the month, the worm would go dormant permanently although an unpatched machine could be re-infected with a new worm.

A week later on July 19, a second version of Code Red I (Code Red v2) was noticed which spread much faster. Apparently, the programming had been fixed to change the static seed to a random seed, ensuring that the randomly generated IP addresses to probe would be truly random [19]. Code Red v2 worm was able to infect more than 359,000 machines within 14 hours [20]. At its peak, 2000 hosts were infected every minute. By design, Code Red I stopped spreading itself on July 20.

On August 4, 2000, a new worm called Code Red II (carrying a different, more dangerous payload than Code

Red I but self-named "Code Red II" in its source code) was observed exploiting the same buffer overflow vulnerability in IIS web servers [21]. After infecting a host, it laid dormant for 1-2 days and then rebooted the machine. After rebooting, the worm activated 300 threads (but 600 for Chinese language systems) to probe other machines to propagate. It generated random IP addresses but they are not completely random; about 1 out of 8 are completely random; 4 out of 8 addresses are within the same class A range of the infected host's address; and 3 out of 8 addresses are within the same class B range of the infected host's address. The attack code ran for 24 hours (but 48 hours for Chinese language systems). The enormous number of parallel threads created a flood of scans, in effect a DoS attack. Then it would write a Trojan horse "explorer.exe" into the root directories of the hard drives, and cause a reboot to load the Trojan. The Trojan did several things to conceal itself and open a backdoor.

On September 18, 2001, the Nimda worm/virus raised new alarms by using 5 different ways to spread and carrying a dangerous payload. The fast spreading had a side effect of traffic congestion. Initially, many sites noticed a substantial increase in traffic on port 80. Nimda spread to 450,000 hosts within the first 12 hours. Although none of the infection vectors was new, the combination of so many vectors in one worm seemed to signal a new level of complexity not seen before [22].

- It sent itself by e-mail with random subjects and an attachment named "readme.exe." It finds e-mail addresses from the computer's web cache and default MAPI mailbox. The worm carries its own SMTP engine. If the target system supports the Automatic Execution of Embedded MIME types, the attached worm will be automatically executed and infect the target. Infected e-mail is resent every 10 days.
- It infected Microsoft IIS web servers, selected at random, through a buffer overflow attack called a Unicode Web Traversal Exploit (known for a year prior). This vulnerability allows submission of a malformed URL to gain access to folders and files on the server's drive, and modify data or run code on the server.
- It copied itself across open network shares. On an infected server, the worm writes MIME-encoded copies of itself to every directory, including network shares. It creates Trojan horse versions of legitimate programs by prepending itself to .EXE files.
- It added Javascript to web pages to infect any web browsers. If it finds a web content directory, it adds a small piece of javascript code to each .html, .htm, or .asp file. This javascript allows the worm to spread to any browser listing these pages and automatically executing the downloaded code.
- It looked for backdoors left by previous Code Red II and Sadmind worms.

Upon infection, it takes several steps to conceal its presence. Even if found, the worm was very difficult to remove because it makes numerous changes to Registry and System files. It creates an administrative share on the C drive, and creates a guest account in the administrator group allowing anyone to remote login as guest with a blank password.

Beginning with the Klez and Bugbear worms in October 2001, “armored” worms contain special code designed to disable antivirus software using a list of keywords to scan memory to recognize and stop antivirus processes and scan hard drives to delete associated files [23,24]. Other recent examples of armored worms include Winevar (November 2002) and Lirva (January 2003). More destructively, the Lirva worm, named after the singer, Avril Lavigne, will e-mail cached Windows dial-up networking passwords to the virus writer, and e-mail random .TXT and .DOC files to various addresses [25]. It will connect to a web site “web.host.kz” to download Back Orifice giving complete control to a remote hacker .

In March 2002, Gibe spread as an attachment in an e-mail disguised as a Microsoft security bulletin and patch. The text claimed that the attachment was a Microsoft security patch for Outlook and Internet Explorer. If the attachment is executed, it displays dialog boxes that appear to be patching the system, but a backdoor is secretly installed on the host.

Another trend to dangerous payloads includes installation of keystroke logging software that record everything typed on the keyboard, usually recorded to a file that can be fetched by a remote hacker later. The Bugbear, Lirva, Badtrans (November 2001), and Fizzer (May 2003) worms all install a keystroke logging Trojan horse.

The SQL Slammer/Sapphire worm appeared in January 2003, exploiting a buffer overflow vulnerability Microsoft SQL Server announced by Microsoft in July 2002 (with a patch) [26]. It is much simpler than previous worms and fits in a 376-byte payload of a single UDP packet. In contrast, Code Red was about 4,000 bytes and Nimda was 60,000 bytes. The sole objective seems to be replication due to the absence of a payload. It appeared to test the concept that a small, simple worm could spread very quickly. The spreading rate was surprisingly fast, reportedly infecting 90 percent of vulnerable hosts within 10 minutes (about 120,000 servers) [27]. In the first minute, the infection doubled every 8.5 seconds, and hit a peak scanning rate of 55,000,000 scans/second after only 3 minutes. In comparison, Code Red infection doubled in 37 minutes (slower but infected more machines). The probing is fast because infected computers simply generate UDP packets carrying the worm at the maximum rate of the machine.

6. Current Developments

The week beginning on August 12, 2003, has been called the worst week for worms in history, seeing the fast-spreading Blaster, Welchia (or Nachi), and Sobig.F worms in quick succession. Blaster or LovSan arrived first, targeted to a Windows DCOM RPC (distributed component object model remote procedure call) vulnerability announced only a month earlier on July 16, 2003 [28]. The worm probes for a DCOM interface with RPC listening on TCP port 135 on Windows XP and Windows 2000 PCs. Through a buffer overflow attack, the worm causes the target machine to start a remote shell on port 4444 and send a notification to the attacking machine on UDP port 69. A tftp (trival file transfer protocol) "get" command is then sent to port 4444, causing the target machine to fetch a copy of the worm as the file MSBLAST.EXE. In addition to a message against Microsoft, the worm payload carries a DoS agent (using TCP SYN flood) targeted to the Microsoft Web site “windowsupdate.com” on August 16, 2003 (which was easily averted). Although Blaster has reportedly infected about 400,000 systems, experts reported that the worm did not achieve near its potential spreading rate due to novice programming.

Six days later on August 18, the apparently well-intended Welchia or Nachi worm spread by exploiting the same RPC DCOM vulnerability as Blaster. It attempted to remove Blaster from infected computers by downloading a security patch from a Microsoft Web site to repair the RPC DCOM vulnerability. Unfortunately, its probing resulted in serious congestion on some networks, such as Air Canada's check-in system and the US Navy and Marine Corps computers.

The very fast Sobig.F worm appeared on the very next day, August 19, only seven days after Blaster [29]. The original Sobig.A version was discovered in January 2003, and apparently underwent a series of revisions until the most successful Sobig.F variant. Similar to earlier variants, Sobig.F spreads among Windows machines by e-mail with various subject lines and attachment names, using its own SMTP engine. The worm size is about 73 kilobytes with a few bytes of garbage attached to the end to evade antivirus scanners. It works well because it grabs e-mail addresses from a variety of different types of files on the infected computer and secretly e-mails itself to all of them, pretending to be sent from one of the addresses. At its peak, Sobig.F accounted for one in every 17 messages, and reportedly produced over 1 million copies of itself within the first 24 hours. Interestingly, the worm is programmed to stop spreading on September 10, 2003, which suggests that the worm was intended as a proof-of-concept. This is supported by the absence of a destructive payload, although the worm is programmed with the capability to download and execute arbitrary files to infected computers. The downloading is triggered on specific times and weekdays, which are obtained via one of several NTP servers. The worm sends a UDP probe to port 8998 on one of several pre-programmed servers which responds with a URL for the worm to download. The worm also starts to listen on UDP ports 995-999 for incoming messages, presumably instructions from the creator.

If these recent worm incidents are viewed as proof-of-concept demonstrations, we might draw the following observations:

- Blaster suggests a trend that the time between discovery of a vulnerability and the appearance of a worm to exploit it is shrinking (to one month in the case of Blaster)
- Blaster demonstrates that the vast majority of Windows PCs are vulnerable to new outbreaks
- Sobig shows that worm writers are creating and trying out successive variants in a process similar to software beta testing, and if variants are being written by different people, it suggests an increasing degree of coordination among worm writers.

One observation is certainly evident from recent worm incidents. Despite firewalls, intrusion detection systems, and other network security equipment, worm outbreaks continue to spread successfully. No one doubts that new worm outbreaks can be expected in the future. Indeed, outbreaks have become so commonplace that most organizations have come to view them as a routine cost of operation.

The reasons for the continued state of vulnerability are complex. The problem is seen by some at a simplistic level as a struggle between virus writers and the antivirus industry. However, the problem is much larger, involving the entire computer industry. Worms and viruses will continue to be successful as long as computers have security vulnerabilities that can be exploited. These vulnerabilities continue to exist for a number of reasons. First, software has become enormously complex, and perfectly secure software has become more difficult. For example, buffer overflow attacks have been widely known since 1995 but this type of vulnerability continues to be found often (on every operating system). Hence, the root problem is that unsecure software continues to be written. Second, when vulnerabilities are announced with corresponding software patches, many people are unaware of them or slow to apply patches to their computer for various practical reasons. Frequent patching takes significant time and effort, and faith that the patches will not cause troublesome side effects. Third, many people are unaware or choose to ignore the security problems. This problem is particularly relevant for home PC owners who often must be educated about security and then make the effort to acquire antivirus software, personal firewalls, or other means of protection.

In addition to technical causes, a sociological reason is the lack of accountability for worm writers. The long-held general perception has been that worms and viruses are low risk crimes. It is notoriously difficult to trace a virus or worm to its creator from analysis of the code, unless there are inadvertant clues left in the code. Even if a virus creator is identified and arrested, cases are difficult to prosecute and prove malicious intention (not just recklessness). Historically, long prison sentences have been perceived as overly harsh for convicted virus writers who have tended to be teenagers and university students. Therefore, sentences have tended to be light. The author of the 1988 Internet worm, Robert Morris, was sentenced to three years of probation, 400 hours of community service, and a $10,000 fine. Chen Ing-hau was arrested in Taiwan for the 1998 Chernobyl virus but released when no official complaint was filed. Onel de Guzman was arrested for writing the 2000 LoveLetter virus resulting in $7 billion of damages, but he was released due to the lack of relevant laws in the Philippines. Jan De Wit was sentenced for the 2001 Anna Kournikova virus to 150 hours of community service. David L. Smith, creator of the 1999 Melissa that caused at least $80 million in damages, was sentenced to 20 months of custodial service and a $7,500 fine. In the absence of a serious legal deterrent, the general perception persists that virus writers can easily avoid the legal consequences of their actions.

7. Conclusions

This review of the evolutionary development of viruses and worms leads to a number of conclusions about our current situation. First, new virus and worm outbreaks may be expected for the foreseeable future. Software will continue to have vulnerabilities that will be expoited by virus writers. Software patching will continue to be problematic for various practical reasons. Second, new worms may appear sooner after a vulnerability is discovered, increasing their opportunity to spread quickly through a largely unprotected population. Third, new worms have been successfully exploring new infection vectors (e.g., peer-to-peer networks, instant messaging, wireless devices). This also increases the chances that a new outbreak may be successful. Fourth, future worms will likely see a rapid succession of variants, again increasing the chances that at least one variant will spread successfully.

The general conclusion is that a new fast-spreading worm outbreak is a real possibility. Continuing recent trends, a new worm may be able to saturate the vulnerable population within minutes or possibly seconds. This points to an urgent need for better, automated means of protection. This protection could take the form of a global coordinated antivirus system. This system would need the capabilities to reliably and quickly detect signs of a new worm outbreak anywhere in the world; broadcast notifications of a new worm throughout the system; and automatically quarantine the outbreak by selectively isolating the infected computers from the rest of the Internet.

References

[1] F. Cohen, "Computer viruses: theory and experiments," *Computers and Security*, vol. 6., pp. 22-35, Feb. 1987.
[2] F. Cohen, *A Short Course on Computer Viruses*, 2nd ed., John Wiley & Sons, NY, 1994.
[3] R. Grimes, *Malicious Mobile Code: Virus Protection for Windows*, O'Reilly & Associates, Sebastopol, CA, 2001.
[4] M. Ludwig, *The Giant Black Book of Computer Viruses*, American Eagle Publications, Show Low, AZ, 1998.
[5] D. Harley, R. Slade, R. Gattiker, *Viruses Revealed*, Osborne/McGraw-Hill, NY, 2001.
[6] J. von Neumann, A. Burks, *Theory of Self-Reproducing Automata*, U. of Illinois Press, Urbana, IL, 1966.
[7] J. Shoch, J. Hupp, "The 'worm' programs - early experience with a distributed computation," Commun. of ACM, vol. 25 , pp. 172-180, March 1982.

[8] E. Spafford, "The Internet worm program: an analysis," *ACM Comp. Commun. Rev.*, vol. 19 , pp. 17-57, Jan. 1989.

[9] G. Smith, *The Virus Creation Labs: A Journey into the Underground,* American Eagle Publications, Tucson, AZ, 1994.

[10] CERT advisory CA-2001-03, "VBS/OnTheFly (Anna Kournikova) malicious code," http://www.cert.org/advisories/CA-2001-03.html.

[11] CERT incident note CA-1999-02," Happy99.exe trojan horse," htttp://www.cert.org/incident_notes/IN-99-02.html.

[12] S. Cass, "Anatomy of malice," *IEEE Spectrum*, pp. 56-60, Nov. 2001.

[13] CERT advisory CA-1999-04, "Melissa macro virus," http://www.cert.org/advisories/CA-1999-04.html.

[14] CERT advisory CA-1999-06, "ExploreZip trojan horse program," http://www.cert.org/advisories/CA-1999-06.html.

[15] CERT advisory CA-2000-04, "Love letter worm," http://www.cert.org/advisories/CA-2000-04.html.

[16] CERT incident note IN-2001-02, "Open mail relays used to deliver Hybris worm," http://www.cert.org/incident_notes/IN-2001-02.html.

[17] CERT advisory CA-2001-11, "Sadmind/IIS worm," http://www.cert.org/advisories/CA-2001-11.html.

[18] CERT incident note IN-2001-08, "Code Red worm exploiting buffer overflow in IIS indexing service DLL," http://www.cert.org/incident_notes/IN-2001-08.html.

[19] H. Berghel, "The Code Red worm," *Commun. of ACM*, vol. 44, pp. 15-19, Dec. 2001.

[20] S. Staniford, "Analysis of spread of July infestation of the Code Red worm," http://www.silicondefense.com/cr/july.html.

[21] CERT incident note IN-2001-09, Code RedII: another worm exploiting buffer overflow in IIS indexing service DLL," http://www.cert.org/incident_notes/IN-2001-09.html.

[22] CERT advisory CA-2001-26, "Nimda worm," http://www.cert.org/advisories/CA-2001-26.html.

[23] F-Secure Security Information Center, "F-Secure virus descriptions: Klez," http://www.europe.f-secure.com/v-descs/klez.shtml.

[24] Sophos, "W32/Bugbear-A," http://www.sophos.com/virusinfo/analyses/w32bugbeara.html.

[25] Symantec Security Response, "W32.lirva.C@mm," http://securityresponse.symantec.com/avcenter/venc/data/w32.lirva.c@mm.html.

[26] CERT advisory CA-2003-04, "MS-SQL server worm," http://www.cert.org/advisories/CA-2003-04.html.

[27] D. Moore, et al., "The spread of the Sapphire/Slammer worm," http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html.

[28] CERT advisory CA-2003-20, "W32/Blaster worm," Aug. 11, 2003, http://www.cert.org/advisories/CA-2003-20.html.

[29] Symantec Security Response, "W32.Sobig.F@mm," http://securityresponse.symantec.com/avcenter/venc/data/w32.sobig.f@mm.html.