Beating the Superbug: Recent Developments in Worms and Viruses

Viruses and worms are significant risks in today's increasingly networked computing environment. This paper will examine the differences between worms and viruses, and then discuss recent developments in virus and worm technology. Some defensive techniques will be examined, and an attempt will be made to predict future possible techniques that may emerge in viruses or worms.

Michael Clarkson **GIAC GSEC Practical (1.4)**

Beating the Superbug: Recent Developments in Worms and Viruses

Introduction

Viruses and worms are significant risks in today's increasingly networked computing environment. This paper will examine the differences between worms and viruses, and then discuss recent developments in virus and worm technology. Some defensive techniques will be examined, and an attempt will be made to predict future possible techniques that may emerge in viruses or worms.

Definitions

Virus

Some computer programs, known as "Malware", have been designed to perform malicious actions. A subset of malware is computer viruses. Dr Cohen defined a computer virus as "A program that can infect other programs by modifying them to include a possibly evolved copy of itself"[1]. He wrote the first paper on computer viruses back in 1984[2], and is widely credited with inventing the term "virus". His definition is now commonly accepted. Simply put, a virus is a program that copies itself to other places, and attaches itself to programs already there.

Mechanism

A virus is simply a piece of program code. Before it is executed, it is just stored instructions on a disk. It needs to be executed to become active. A virus can attach itself to another program. This means that when the program is executed, the virus will itself become active. The virus will look for a program that is regularly executed, to ensure that it will become active in the near future. Different viruses attach themselves to different programs, and different types of executable object. Once the program is executed, the virus is activated, and then goes looking for other files to attach itself to.

Early Example: Brain

The one of the first known personal computer viruses (©Brain) was discovered in 1986[3]. Every disk contains a "boot sector." This boot sector may contain a small portion of program code that tells the computer how to start up (not all disks are "bootable" as this sector may contain bad information, or be blank.) The ©Brain virus infected the boot sectors of floppy disks. When an infected disk was used to

---

1 Dr Fred Cohen, Computational Aspects of Computer Viruses, Computers and Security, Volume 8, 1989, p325

2 Dr Fred Cohen, "Computer Viruses: Theory and Experiments", Proceedings of the 7th National Computer Security Conference, pp. 240-263, 1984

3 F-Secure, "Computer Virus Information Pages: Brain". URL: http://www.europe.f-secure.com/v-descs/brain.shtml

boot up a computer, the virus would be activated, and copy itself from the boot sector and into memory. The virus would then monitor the floppy disk drive, and then copy itself to any uninfected disks used.

When infecting a disk, the virus would mark a portion of the disk as bad (in the File Allocation Table.) It then copied the original boot sector information of the floppy disk to that bad portion, and copied itself to the boot sector. It monitored any calls to the floppy disk, and would redirect any calls to the boot sector to the original data (now stored on the "bad" portion of the disk), thereby hiding the infection. By marking the disk bad, the virus prevented other programs from accidentally accessing the original boot sector data, and revealing the infection. A virus needs some executable code to attach itself to. Here, that code is contained in the boot sector. This early example illustrates the necessary elements for a virus, a place to store program code, a means of executing that code, and a communication vector to get that code some-place else.

Worm

A computer worm is also a piece of software that copies itself elsewhere. However, unlike a virus, it does not attach itself to, or modify, other files. The worm is a stand-alone piece of code - although it may need to use another program to spread, it does not change that program in any way. The term "worm" is taken from the book "Shockwave Rider"[4] and refers to a self-replicating piece of code (called the "tapeworm") created by a character in that book.

Morris Worm

The first known Internet worm was the Morris worm. This crippled the Internet for several days in 1988. It exploited buffer overrun flaws in the common "sendmail" and "fingerd" programs (as well as rsh/rexec trust issues and weak user passwords)[5]. A buffer overrun occurs when a program does not check the length of an input string against the amount of memory it has available to store that string. If the entered string is too long, then it overwrites other memory in the computer. In the case of the Morris worm, the string contained garbage data, followed by instructions for the computer to send the same message to other available machines. The worm (Morris claims) was supposed to spread slowly, and in a controllable fashion, but due to a bug in the code spread much faster than anticipated. This brought about the first large scale Denial of Service (DoS) condition on the Internet itself. An infected machine could have its resources entirely controlled by the worm in as little as one hour. The courts were not impressed by Morris' actions, and sentenced him to 3 years probation, 400 hrs community service, and $10,050 fines[6].

---

4 Brunner, John. *Shockwave Rider*. NY: Del Rey, 1978 (originally published in 1975 by Harper & Row). ISBN 0-345-32431-5. C.I.P

5 Zeltser, Lenny. "The Evolution of Malicious Agents." April 2000. URL: http://www.zeltser.com/agents/agents.html

6 Boettget, Larry. "The Morris Worm: how it Affected Computer Security and Lessons Learnd by it". December 24, 2000.
URL: http://wbglinks.net/pages/reads/misc/morrisworm.html

Worm vs Virus

This is the difference between a worm and a virus, a virus "infects" programs on the victim machine, whereas a worm may simply use that program's functionality to launch itself to other computers.

Environment

Moore's law (named after Gordon Moore, co-founder of Intel) predicts that computer power will double every 18 months. This allows more complex programs, with more features and "friendlier" user interfaces. The Internet has developed over the last two decades and has changed the ways in which information is shared. New operating systems, combined with larger hard disk drives and increased processing power have changed the way users interact with their machines. Viruses and Worms rely on the architecture of the computers they are running on, but also of the networks connecting those machines to each other. As the computers and networks evolve, so must the viruses that exploit them.

InterNet

The Defense Advanced Research Projects Agency developed the Internet in the late 1960's[7] as a decentralised means of command in the event of a nuclear strike. It was then adopted by researchers and academics, and later business groups. The Internet is used ubiquitously today to communicate all forms of information.

Net connections

There were estimated to be 533 million people using the Internet in 2001[8]. Home Internet technology has improved from early modems to cable modems, ADSL or satellite connections etc. The technology supporting the Internet such as routers have increased in speed and bandwidth, so that now more information is being passed than ever before. Although very useful, this increased functionality comes at a cost – it is a perfect medium for viruses and worms to spread. A worm replicates directly around a computer network. Without a sufficiently large network, the worm does not have sufficient targets to infect, and its lifecycle is accordingly limited. There have been worms released on Local and Wide Area Networks, for example the Ladex worm which could only spread across local area networks[9]. The Internet is however the most effective setting for a worm.

---

7 The Moschovitis Group. "Chapter 3, What Does a Network Do?" History of the Internet. May 1999. URL: http://www.historyoftheinternet.com/chap3.html

8 Computer Industry Almanac. "Press Release". March 21, 2002. URL: http://www.c-i-a.com/pr032102.htm

9 McAfee Security. "WNT/Ladex.worm." 2002 URL:http://vil.nai.com/vil/content/v_99590.htm

Psychology

A virus is simply code that replicates and infects other files. It is unlikely but possible that a virus could spontaneously occur through random corruption of a hard disk drive, and then spread into "the wild" (this phrase was coined by David Chess in 1990[10], and refers to active threats to real world computers). Much more likely is that a virus would be deliberately created for some purpose, and then either be released or escape. The symptoms and actions taken by that virus will depend on its design, which will itself depend upon the purpose for its release. Therefore, by examining the motivations behind the creation of a virus, we are more able to predict possible future developments in viral technology.

Infamy (Media)

A virus may be released out of a desire for media exposure, or notoriety. The world is becoming more reliant on computers and network technology. The media recognise this fact, and are giving more news time to computer related developments. Often the media reports are unnecessarily alarmist, and short on technical detail. Presumably an "armageddon scenario" makes a better story. The "Love Bug" and "Code Red" viruses for example received a large amount of media attention. A virus may be created for this reason, giving many people a reason to hate the virus's creator, but without knowing who that is. This mindset is similar to vandalism, or arson. Also, the virus author may trust their peer group, and tell them who they are. In this case, they will be able to gain increased respect from those peers for writing and releasing an effective virus. Or the author might just enjoy the "15 minutes of fame" until the virus is subdued and fades from public view. People creating viruses for this reason are likely to include a media "hook" to get public attention. This may be a destructive payload, such as reformatting a hard disk drive, or simply displaying a memorable or noteworthy image etc. Alternatively, they may design the virus to spread as far and as fast as possible, in an attempt to cripple Internet systems.

Curiosity

Intellectual curiosity is another reason that someone might create a computer virus. "To see if I could do it." A piece of code that copies itself elsewhere, and seeks to "survive" is an interesting technical trick that poses several challenges. This is closely linked to "Proof of concept" viruses in which the virus author simply creates a virus for a specific target to prove that it can be done. Viruses created for this reason will likely not include a destructive payload, and may include helpful "comments" in the code to help others to understand this new viral technique.

Teaching

Some Computer Science classes have to dissect a computer virus to examine its

10 Dwan, Berni. "The Computer Virus – From There to Here." Computer Fraud & Security. Issue 12, December 2000 (2000): 13-16

methods of manipulating operating systems and communication protocols. Another, WordMacro/Polite may have been created for teaching purposes, as it asks permission to infect particular files[11]. For a virus of this type to get out into the wild would require the creator to be careless, but not as careless as the victims who agree to become infected. A virus created for teaching methods is likely to be quite simple, so that students do not have too much difficulty in understanding its operation.

Political, Malice

A virus may also be created out of malice, or for political reasons. A virus may be written with a specific system or network in mind, for reasons personal to the virus's creator. To use a virus as a tool of terror seems a little short-sighted. Viruses infect every possible target. A potential terrorist releasing a damaging virus would likely be hurting themselves be its release. Some viruses seek to work around this problem, for example by only infecting computers that meet a certain requirement. The MyParty virus only infects computers that are not using the Russian keyboard layout[12]. The CodeRed v2 virus infects all vulnerable targets, but creates twice as many hostile threads on computers that use Chinese or Taiwanese as the default language[13], and spreads for twice as long. Due to the close linkages between computers, infecting some will have a detrimental (if indirect) effect on the operation of the others.

First Malicious

Following the release of the Morris worm in 1988, a similar worm called "WANK" was released in 1989. Whereas Robert Morris claimed that his worm was a technological investigation, WANK was designed with a destructive payload. WANK may have been the first malicious worm[14].

Helpful Virus

Some have also claimed that there is a place for helpful viruses – viruses that serve a useful purpose. An example of this is the KOH virus[15], that asks permission to infect a Hard Disk, and encrypts the drive's contents. It decrypts the contents on the fly, protecting the drive from unauthorised access. There was also suggestion of releasing a virus that would spread using the same vector as a previous virus, but then close the security flaw that allowed them entry. This virus would spread until there were no more vulnerable machines, and then disappear. Helpful viruses are rare, as they limit the choice of the system administrator, as well as creating rather interesting legal issues.

---

11 F-Secure, "Computer Virus Information Pages: Polite". URL: http://www.f-secure.com/v-descs/polite.shtml

12 F-Secure, "Computer Virus Information Pages: Myparty". URL: http://www.f-secure.com/v-descs/myparty.shtml

13 McAfee Security. "Virus Information Library: W32/CodeRed.c.worm" 2002
URL: http://vil.mcafee.com/dispVirus.asp?virus_k=99177&&cid=2399

14 CERT, "CERT Advisory CA-1989-04 WANK Worm On SPAN Network." October 17, 1989.
URL: http://www.cert.org/advisories/CA-1989-04.html (September 17, 1997)

15 Metropolitan Network BBS Inc, "KOH" URL:http://www.avp.ch/avpve/boot/koh.stm

Helpful Worm

A worm can also designed to perform a useful function. In the late 70's, a worm was created (called the "vampire" worm) that would collect maths intensive computations, and then run them at night, when the system was being used less[16]. The worm in that case had a flaw, that caused the system to crash during the day, and a piece of counter-software had to be executed. Nevertheless, the worm was written with good intentions.

Creation

Information more available (BBS, Tutorials)

There used to exist Bulletin Boards (such as Caustic Contagion[17]) that contained virus information. These had a limited selection and often were difficult to find. The Internet has no such restrictions. It is the single largest information resource in human history[18]. There are tutorials available on the Internet[19] on how to create a virus for most computing platforms. Source code and compiled versions of existing viruses are also available, often along with commentaries on the tricks employed by the original virus creator to make the virus more difficult to detect or contain. These reference materials have made the knowledge required to create a virus significantly easier to come by.

Operating System Knowledge

To create a virus used to require detailed knowledge both of the assembler language of the target operating system, and also of the function calls etc that the operating system used to function. Some viruses exist that are thought to have been written in a high level "compiler" language (for example the Hooters.5000 and AIDS viruses[20]) but these were slow and inefficient in contrast to their contemporaries. The virus creator needed to know how to read, write, and manipulate memory (code pointers etc), as well as disk drive functions, in detail. Now, most operating systems provide more complex "function calls". These are easier to use, and reduce the size of the viral code. However, they are also often stricter in what they will and will not do, and so require the virus programmers to be craftier in getting around systems' (often deliberate) limitations.

Creation Kits

---

16 "Benefits of the Computer Virus". URL: http://www.greyowltutor.com/essays/virus.html

17 Smith, George. "An Excerpt from 'The Virus Creation Labs'." 1996. URL:http://www.soci.niu.edu/~crypt/other/vcl.htm

18 Lesk, Michael. "How Much Information Is There In The World?" URL:http://www.lesk.com/mlesk/ksg97/ksg.html

19 VX Heavens, "Library/VX." URL:http://vx.netlux.org/lib_vx.shtml

20 Metropolitan Network BBS Inc, "HLL Family (High Level Language viruses)" http://www.avp.ch/avpve/file/h/hll.stm

Another relatively recent development is virus creation kits. These pieces of software allow people with no technical knowledge to choose from options listed, and create a new (and possibly dangerous) virus in minutes. Many of these "tools" even come with Graphical User Interfaces! This makes creating a virus extremely simple. The first known virus creation kit was "Virus Creation Lab"[21] but this created rather simple viruses. By comparison, today's virus creation kits allow for polymorphism, varied payloads, etc. Visual Basic Script (VBS) Worm Generator is a virus creation kit that creates VBS viruses. These viruses use the Windows Scripting Host to spread. This piece of Windows software automatically runs VBS scripts in some circumstances. The VBS Worm Generator can generate worms that spread through various vectors, including Outlook/e-mail exploitation, Internet Relay Chat, and infecting files[22]. The author of this creation kit (known as [K]alamar) originally decided not to include options for destructive virus payloads, intending the creation kit to be used to educational purposes, but the new version of this creation kit includes the option to link the virus to an executable file[23]. These could execute arbitrarily destructive actions. However, viruses created in this way are quite limited. The only options about the virus's operation are those foreseen by the initial author of the virus creation kit. This kit would be unable to use new vulnerabilities or functionality to spread. The viruses would also share a common structure, and set of assumptions, and could possibly be detected through a common signature. These tools have made creating a basic virus a "point & click" process, but creating a new and original virus is still a difficult task.

Scripting Languages

Recently scripting languages have become a target for virus writers. The first scripting language virus was "Concept" discovered in 1995[24]. A scripting language is a simple programming language that is used in documents, to increase functionality. One such language is Visual Basic for Applications (VBA). Various programs use this language, including Microsoft Word, Powerpoint, and Excel. The code is included in the document itself. Like regular viruses, this code can include instructions such as copying itself to other open documents. Virus writers have exploited all these programs. Some viruses are limited to a single application, but others like TriState[25] can infect multiple document types. The Melissa virus is a recent scripting language virus that sent itself out to e-mail addressed listed in the user's Outlook address book[26]. This is only possible through the facilities provided by the Visual Basic language supported in Word.

---

21 CKnow, "Virus History." URL:http://www.cknow.com/vtutor/vthistory.htm

22 Grazi, Alberto. "VBS Worms Generator." February 21, 2001. URL:http://rr.sans.org/malicious/VBS_worms.php

23 Vamosi, Robert. "New worm generator kit sparks the threat of virus epidemics." March 13, 2001. URL: http://www.zdnet.com/filters/printerfriendly/0,6061,2695860-3,00.html

24 The Macro Virus Help Center. "Did you know?". 1996. URL: http://training.csd.sc.edu/virus/macro.htm

25 F-Secure, "Computer Virus Information Pages: Tristate". URL: http://www.f-secure.com/v-descs/tristate.shtml

26 CERT, "CERT Advisory CA-1999-04 Melissa Macro Virus". March 27, 1999. URL: http://www.cert.org/advisories/CA-1999-04.html (March 31, 1999)

Scripting languages have two advantages for virus writers. Firstly, they are easier to write. Assembler is a rather difficult language to learn, and requires more significant time investment. Viruses written in higher level languages tend to be less successful. High-Level compilers are often not optimised to reduce the size of the compiled code, so the final virus contains many unnecessary instructions. Modern scripting languages tend to be forms of BASIC (Beginners All-purpose Symbolic Instruction Code[27]), which is an easy language to gain proficiency in.

The second advantage of scripting languages is that they are platform independent. As a scripting virus targets the application, rather than the operating system, it can infect any computer running the target application. A Microsoft Word Macro virus can infect an Apple Macintosh running Word just as it would a PC.

Open Source

An open source program is one to which the source code is freely available. The code can be studied by other programmers, and any flaws, alterations, bugs or improvements suggested to the author (or altered and compiled directly). There are even open source operating systems, such as Linux (Based on Minix, a Unix variant, and written and released by Linus Torvalds in 1991). This source code can be studied, as it is much easier for humans to understand than assembly or machine code. This makes it easier to find a security flaw, and then to create a worm based on that flaw. This is more difficult in proprietary systems, and requires more effort[28].

White Hats

However, there are also groups of programmers studying the code from the other point of view, looking to close flaws. Their job is also easier under the open source model. Many will notify the original authors of any flaws found. This makes finding an unknown flaw in the system less likely. The "white-hat" community (hackers and analysts working to improve security, rather than exploit flaws) may be the first to document a new flaw. Even if this is the case, there will still likely be un-patched systems amongst which a worm could propagate.

Virus in Source

Open source software is often distributed as source code, which can be compiled on each user's computer. This makes distributing the virus more difficult. It is much harder to hide a virus in source code than in an executable file[29]. Although many users may not examine the source code in detail, those that do may notice the malicious code, and notify the distributor. This is less likely with proprietary systems, and some large software vendors have distributed software infected

---

27 Webopedia. "BASIC". May 28, 2001. URL: http://www.webopedia.com/TERM/B/BASIC.html

28 Zenkin, Denis. "The Invulnerable Penguin?" 2000 URL: http://www.scmagazine.com/scmagazine/sc-online/2000/linux/index.html

29 Yeargin, Ray. "The short life and hard times of a Linux virus". June 10, 2000. URL: http://www.librenix.com/?inode=21

with viral code[30].

Distribution

The Internet has now become the single most used computer transmission medium[31]. This makes it the ideal medium for the distribution of computer code, and viruses are no exception.

Initial Release

The initial outbreak of a virus is very important to its overall lifecycle. If a virus is discovered and contained early, all vulnerable systems can be hardened appropriately before becoming infected. A virus may be sufficiently virulent that it saturates the vulnerable systems before an appropriate defense is available. This makes the defenders job much harder. An aggressive and ubiquitous virus could cause a denial of service condition on the Internet itself, effectively preventing system administrators from researching and installing counter measures (the Morris worm accomplished this, and several recent e-mail based viruses have come close.)

Sneaker-Net vs Internet download

Previously, a virus would be perhaps installed on a single system, or several systems, and then left to spread naturally. However, the Internet is a very fast, very global, single source of infection. Recent viral threats have been Internet (rather than disk) based, and are also thought to have been initially launched from the Internet. There are various ways that this could be done. A virus can be installed in a piece of downloadable software, either also written by the virus creator (although this is somewhat easy to discover and prosecute), or through more nefarious means into software created by someone else. A new release of popular software can be downloaded many times in a very short space of time. This causes an explosion of the virus when the software is run for the first time.

Multiple Initial Infectious Hosts

Another option for the initial release of the virus would be through multiple "zombie" viral infectors[32]. In this scenario, several computers would be subverted and controlled by the virus author. They would be loaded with the viral code, and given a list other potential targets. At a specified time, all the zombie machines would begin infecting other computers from their list. This would maximise the spread of the virus, and the difficulty of containing the infection. The Internet is also useful in hiding the origin of the virus. Early viruses could be traced back to

30 Leyden, John. "Microsoft security fixes infected with FunLove virus". April 25, 2001. URL: http://www.theregister.co.uk/content/8/18516.html

31 Larson, Ray. "Bibliometrics of the World Wide Web: an exploratory analysis of the intellectual structure of cyberspace". 1996. URL: http://sherlock.berkeley.edu/asis96/asis96.html

32 Staniford, Stuart et al. "How to 0wn the Internet in Your Spare Time". 14 May 2002. URL: http://www.icir.org/vern/papers/cdc-usenix-sec02/

an initial geographical release point, due to their slow spread. A virus could now be released over the Internet in such a way that tracing it would be almost impossible. Further, it could be released through a publicly accessible Internet terminal, such as in a cyber-café. This way, even if the initial outbreak was identified, there would be no way to locate the user responsible.

Hit list Block Scanning

One more step that the viral creator could take is to slowly scan portions of the internet, looking for vulnerable machines. This scan may take several months or more. The machines that are initially found to be vulnerable to the virus are stored in a list. Parts of this list are distributed to the slave machines, so that many vulnerable machines may be infected in the initial stages of the outbreak. Once many machines have become infected, and are all looking for other victims, saturation can quickly be achieved.

Self Spread (Speed vs Dormancy)

The Internet is a good target for the self-replication of a virus. Having infected a computer connected to the Internet, all other Internet computers become potential victims. The Internet has virtually no information transfer overhead compared to a physical medium (which requires transporting from one computer to another.) Given the small amount of data required for a virus, this makes the spread much faster. It is possible for a virus to infect a large number of machines very quickly. Two common techniques used for Internet spread are IP scanning and address book exploitation. In the first technique the virus simply picks a random block of IP addresses, and scans that block for vulnerable computers. Any vulnerable computers discovered during this scan are then infected by the virus. Although thorough, this method can use a lot of bandwidth, and is very overt. A computer monitoring several or more IP addresses can notice the scan, and block the infected computer from communicating with it. Another technique is address book exploitation. E-mail viruses particularly use this method, although other malware (for example the Nimda worm) has also used this technique. The virus opens the address book and sends itself to the e-mail addresses stored there. The first virus to use this method of propagation was the Melissa virus[33], that used the Outlook address book to spread. It would send itself to the first 50 addresses in the address book, which led to rapid worldwide distribution.

Dormancy

One disadvantage of the Internet is that is makes it harder for viruses to hide. A common source of reinfection from viruses used to be old floppy disks. A virus would use a floppy disk to gain access to a victim machine. The virus would be detected on that machine, and subsequently cleaned off. However, the virus would still be on the floppy disk. It could later reinfect that machine, if the disk

33 Forland, Olav. "What to learn from visits by Melissa and her siblings". April 28, 1999. URL: http://www.norman.no/documents/melissa_experiences.shtml

was used again without being disinfected. There is no such possibility on the Internet. The reason a virus could hide on the floppy disk and escape detection is that the disk was not connected to the computer being scanned. A virus that came in over the Internet however must be stored on the local hard disk drive. A virus scanner could therefore access the virus, and disinfect it. So, although the Internet can speed up virus distribution, it also shortens the virus' possible dormancy period.

Program Specific

The Internet was primarily designed as a communications medium. Accordingly, information is sent across the Internet in pieces, which need to be reassembled and then displayed at the receiving end. It is up to the software to know what the information should be displayed as, and how it should be assembled. This means that viruses have to be deliberate about which program they exploit, as it will be up to that particular program to reassemble and display the information, rather than the operating system.

Infection

Sites for infection

There are various places on the drive that contain executable instructions. The locations of these places depend on the computer, and the operating system used. This means that a virus (generally) can only target files on one type of computer, or operating system. So, viruses use three main methods to find files that they are able to infect. These methods are direct action, terminate and stay resident, and cluster infection.

1

In the first method, the virus would be activated, and would then go out and attach itself directly to other executable files. An example of this type is the "Boza" virus[34]. This virus attaches itself to win32 executable files. When an infected file is run, the virus searches for and attaches itself to (up to) three other win32 executable files, then returns control to the operating system. This virus does not remain in memory. A virus does not necessarily need to alter the original program, as long as it can logically attach itself to it. A companion virus takes advantage an aspect of MS-DOS (Microsoft Disk Operating System.) When a command is typed, MS-DOS first looks for a .COM file with that name. If none is found, MS-DOS looks for an .EXE file. A companion virus creates a new .COM file with the same name as an existing .EXE file, but containing the viral code. When the user attempts to execute the .EXE file, the .COM file is executed first, and the virus is activated. An example of this type of virus is the AIDS2 virus[35]. This virus then runs the original .EXE file, so that if the virus was not so

34 Sophos. "W95/Boza." URL: http://www.sophos.com/virusinfo/analyses/w95boza.html

35 McAfee Security. "AIDS 2" 2002 URL: http://vil.nai.com/vil/content/v_98149.htm

blatant (it plays a tune and displays a message), the user might not know that they were infected at all.

2

In the second method (Terminate and Stay Resident) the virus is activated when the program it has infected is run. It stays in the computer's memory, even after the original program has finished (terminated.) When another executable file is accessed, the virus attaches itself to that file. This method has the advantages of not requiring unexpected disk activity (the drive would be using the disk anyway, so the user would not notice sporadic and random disk usage), and only infecting the files that are actually executed by the user. An example of this type of virus is "Sayha Watpu"[36]. This virus remains in memory once activated, and infects any .COM or .EXE files that are modified or accessed.

3

The third approach is the cluster technique. The virus inserts itself into the boot sector, so that it executes when the computer is started up. The virus remains in memory. When a floppy disk is used, the virus attaches itself to the boot sector of that disk also. In this manner, the virus spreads from disk to disk. An example of this type is the "Brain" virus.

Multiple file types

The first two of these methods (direct action, and terminate and stay resident) could potentially infect every executable file on a disk drive. The third method is not especially virulent, unless combined with another method, as in the Kiev.2049 virus[37]. This is transmitted in infected .EXE files. Once an infected file is run, the virus is copied into memory. It also copies itself into the drives boot sector, so that it will go memory resident again the next time the computer is booted. Once in memory it infects some .EXE files as they are executed. Many early viruses could only infect one type of file, ie .COM or .EXE. This is due to the different code format in these types of files. More modern viruses can infect multiple different file types, such as Natas[38], which infects executable (EXE), command (COM), and overlay (OVL) files.

Any file on a disk

Viruses written in machine code (assembler) are fast, and small. They also require a specific instruction set. This means that although they could only run on a specific set of computers (and operating systems)[39], they would run in exactly the same fashion on all of those computers. A virus could attach itself to any executable file, and be able to run in the same manner. This meant that a virus

36 McAfee Security. "Sayha Watpu" 2002 URL: http://vil.nai.com/vil/content/v_1059.htm

37 McAfee Security. "Kiev 2049" 2002 URL: http://vil.nai.com/vil/content/v_674.htm

38 F-Secure, "Computer Virus Information Pages: Natas". URL: http://www.f-secure.com/v-descs/natas.shtml

39 Scientific-Computing.Co.UK. "Machine Code and Assembler." URL: http://www.scientific-computing.co.uk/level.htm

could infect an entire machine given sufficient resources.

Function Calls

New operating systems monitor system calls and prevent any direct reads or writes[40] to certain portions of the disk. This means that most viruses can no longer use these methods. Instead, they have to use a specific program to spread. An example of this is the "LoveBug" virus. This uses features in Microsoft Outlook to copy itself to all the email addresses in Outlook's address book. The only place this virus can hide is in an Outlook "In Box." This makes finding it much easier.

Computers for infection

Assembler viruses were extremely fast and efficient, but could only operate on a specific type of computer. Recently a new type of virus has emerged – macro viruses. As mentioned, these viruses have the capability to infect any computer running software that uses that scripting language. Currently these scripting languages do not provide some functionality, which limits the vectors a macro virus can exploit. This functionality could develop in time.

Sheer number of Possible Victims

Although viruses can only use certain programs to spread, there are now many machines connected to the Internet (and Local Area Networks) using those programs. Certain software vendors have significant numbers of machines using their products. In some cases, these products are not especially secure, which aids in the spread of viruses. It does however give some security to users using less common products, as they are less likely to be targeted. Over fifty thousand viruses are currently known, and significantly over half of these are targeted specifically at MS-DOS or Windows operating systems[41]. This is likely due to the large role Microsoft plays in the marketplace, which makes them a large target.

Multiple Victim Types

The evolution of viruses has led to a point where a virus can infect and spread using multiple programs, possibly in different ways. For example the Nimda virus which can spread through email attachments (it includes its own smtp engine), browsing past an infected web site with javascript running on a vulnerable browser, IIS exploits, clicking on a particular file with active desktop enabled, and through accessible mapped drives. The Nimda virus is also sometimes referred to as the "Concept" virus due to a text string it contains. It is more usually referred to as "Nimda" as a "Concept" virus was already in circulation when Nimda was discovered[42]. Although viruses such as Nimda have several possible

---

40 Newmarch, Jan. "System Software Directories – Win32API." November 19, 1997 URL: jan.netcomp.monash.edu.au/ssw/directories/win32.html

41 CKnow, "Number of Viruses." URL: http://www.cknow.com/vtutor/vtnumber.htm

42 Delio, Michelle. "Nimda Lives; What a Concept". November 9, 2001.

exploitable vectors they can still only use known and flawed services. This makes detecting the virus and preventing it reinfecting a system much easier. The "LoveBug" spread through a security flaw in Outlook, which is now easy to close. Not only does this prevent reinfection from the "LoveBug", but a number of other viruses that might use the same entry point to a system.

Detection

A good virus scanner has three main aspects. Firstly, it avoids false negatives. A false negative is where the program decides that a file is free from infection, but in fact it is not. Secondly, it avoids false positives. A false positive is declaring that a file contains a virus, but the file is actually clean. Thirdly, it should be easy to use (fast, automatic, and give helpful information.) Avoiding false positives is easy, just declare that no files contain viruses! Similarly, to avoid false negatives, just call every file a virus. Avoiding both types of mis-detection is a worthy goal for virus software, but has been proved to be theoretically impossible[43]. False negatives are more damaging in the short term (leaving the computer infected with the virus.) False positives are more insidious, leading the user to disbelieve the virus scanner, increasing their work-load, and possibly causing the user to disable the virus scanner.

Baselining

It has become harder to detect noticeable effects of a virus. In a single user/ process environment, if something strange started happening, then that was likely a virus. Machines were slow and small enough that a small change would be obvious. Computers today have much more complicated operating systems, as well as being faster and larger. This makes small changes much harder to detect. A sudden burst of unexpected disk activity may mean a virus, or it may mean a word auto-save, or some random Windows background task taking place. This variability makes getting an accurate idea of how the computer usually performs (base-lining) almost impossible. Virus checkers are therefore necessary.

There are three main methods used to detect viral activity, activity monitoring, scanning, and file integrity checking.

Activity Monitoring

Activity monitoring involves looking at every disk read or write, for suspicious activity. "Suspicious" may mean writing to the drives boot sector, writing to another executable file, directly accessing the File Allocation Table etc. These are all signs that a virus may be attempting to infect a machine, or hide itself. The virus scanner could detect these activities, and report back to the user, to either allow or deny permission for that act. If a program was running that was known to

---

URL: http://www.wired.com/news/technology/0,1282,48262,00.html

43 Chess, David & White, Steve. "An Undetectable Computer Virus." 2000
URL: http://www.research.ibm.com/antivirus/SciPapers/VB2000DC.htm

use these methods, (disk de-fragmentation programs for example) then the user could choose to allow those writes, knowing the activity was not malicious. Many BIOS's (Basic Input/Output System – the computers most basic set of instructions stored on the motherboard) now include virus activity scanning options. Many operating systems also now include similar features. This has become easier recently, as computers are faster, and so some overhead in read and write operations will likely not be noticed. The system uses "function calls" to perform read and write operations for programs. Part of these calls can be to check to see that nothing untoward is going on. As part of the operating system, this process can be completely transparent to the user. Nevertheless, there will be some impact on system performance. A risk analysis should be performed on the system, and this method only used in high availability systems.

File Scanning

Another method used to detect viruses is file scanning. This involves the virus scanner going out and scanning every file (matching certain criteria) on the hard disk. The scanner can look for known signatures, the faster method, or look for suspicious code. This second technique, known as heuristic scanning, assumes that viruses need to act in a specific way to replicate. It uses this assumption to report on any files containing certain types of instructions, such as code to "Terminate and Stay Resident." This method is slower, and more prone to false positives (for example a legitimate mouse driver will also contain TSR code.) File scanning is slow, and reliant on viral signatures (unless heuristic scanning is used) meaning that a DoS on the signature distribution centre would render the virus scanner (temporarily) useless.

File Integrity Checking

File integrity can also be used to detect virus infection. This method stores a "checksum" for every (important) file on the drive. A checksum is a large number that represents all the data in a file. If any byte in that file changes, then the checksum will also change, and then that change can be detected. This indicates virus activity (unless the file has been changed for some other reason.) This method is quite fast, and has the ability to detect even unknown viruses. It can also be used to detect direct hacker activity, or even disk corruption. However, it can only detect a virus once that virus has been activated, and written to a (previously checked) file. Further, some files can not plausibly be checked in this way, as they are constantly changing. An example of this is the Windows registry file. This file is often being modified, such that an integrity checker would report a change on many system reboots. If an unknown program appears suddenly in the "Run" (or "RunOnce") sections of the registry, then that is likely a virus.

Finding Macro Viruses

Macro viruses are relatively easy to detect. The viral code needs to be stored in a document related to the application that can interpret that code. Ie, a word macro virus needs to be stored in a word document. Searching all word documents will reveal conclusively whether or not a simple word macro virus is present in the system. Similarly, a purely mass e-mail virus will be located in the in (and out) boxes of the e-mail program. Many modern viruses are much harder to detect than this. Nimda can copy itself in several forms, including in embedded malicious java code. To locate this virus would include searching all web pages

and shared directories stored on a computer.

Polymorphism

Some viruses are "polymorphic" – meaning that they change their code with each copy (the code is changed in known ways, so that the virus always functions in the same manner.) Magistr is an example of this type of virus. Magistr is polymorphic (it appears to include 2 polymorphic engines) and uses several tricks to make detecting and eliminating it more difficult[44]. Another technique used to disguise a virus is encryption. The virus includes a small decryption engine at the start of the virus, which executes the rest of the viral code. Some viruses use sophisticated polymorphic methods to disguise this decryption engine, and reduce the amount of similar code from copy to copy.

Old Dog, New Tricks

As virus technology has developed, virus scanner technology has also evolved. A new method used by some virus scanners is to simulate running the code of a suspected program, so as to use the decryption engine provided by the virus. Once the virus has been decrypted, the code can be examined and compared to known viral samples. Another method new virus scanners use is "heuristic" scanning. This looks at the code itself, to detect known viral techniques. Redirecting system function calls is something that most programs would not need to do. The scanner knows this, and notes the file being scanned as suspicious. This technique can be prone to "false positives" (something being thought a virus that is in fact a legitimate program.) The scanner might ask a user whether a certain program should manipulate the boot sector of the drive, or if that was indicative or a virus infection.

Definitions for Creation Kits

Creation kits have formerly posed a problem for virus scanners. A creation kit can create thousands of new viruses with little effort from the user. The anti-virus company would need to go through each possible virus created with the kit to prevent them all. Alternatively, the company could look at each of a large number of created viruses, and attempt to find similarities between them. Once a similarity is found that is unique to those viruses, that similarity can be used to identify them. This requires a significant amount of work by the anti-virus researchers, but only needs to be done once for each virus creation kit.

Damage Control

Not all viruses can be avoided. A new virus could determine that the computer was vulnerable, and then infect it before a remedy was available. In this case, the only viable option is to seek to limit the damage the virus causes. This option will help avoid more serious damage to the system. Such measures include

-User training. A computer does not spontaneously develop a virus. Users should be made aware of the ways viruses can spread, symptoms of a virus

44 Metropolitan Network BBS Inc, "I-Worm.Magistr" URL: http://www.avp.ch/avpve/worms/email/magistr.stm

infection, what to do in that instance, and the risks associated with incoming computer information.

-Virus scanners. A virus must be introduced to the system through some input device. The virus scanner should include scanning (or stopping) e-mail attachments to prevent infection. Regularly update virus definitions.

-Ensure "Macro Virus Protection" is enabled in all applications that support it. Do not execute any unexpected e-mail attachment. Update e-mail software with the latest patches available.

-Using a "quarantine" machine. This is a computer that is not connected to the rest of the network, and is not responsible for any necessary operations. It can (and should only) be used to test new software and configurations. If a new piece of software contains a virus, it will not be able to spread from this computer. Where a virus is detected, that media should obviously not be used in any other machines until it has been fully disinfected.

-Backups. All data should be regularly backed up. Although most viruses in the wild today do not destroy data, some do. In the case that a new and destructive virus gets through the other defense mechanisms, the data should be recoverable. Data should be backed up on a rotational (Grandfather, father, son type) scheme, so that it can be recovered some time into the future. Backups should be regularly tested, and stored off-site (to prevent non-viral threats from destroying the backups at the same time as the computers.) Note: When a virus has infected a system, a decision must be made whether to disinfect the system, or rebuild it. Some viruses might hide in unexpected places, such that rebuilding the system is the safest option.

Real-Time Countermeasure

David Chess, an IBM research, proposed a system for a computer to automatically develop a virus signature and inoculate the local network[45]. This system requires significant overhead, but could potentially be set up in a large network environment. It requires firstly that a "Tripwire" type program monitor the integrity of files on the computer. This should be combined with access logs that record any programs making unauthorised changes to files. Then, when any program changes a file, that program will be identified. The system should then trace back to try to identify where that program entered the network. This is the viral entry point. The network should then close down the entry point, possibly by automatically disabling the exploited service until a secure version can be developed. The system would then restore the original versions of the files from a secure backup. By comparing the original files, and the infected versions, a virus signature could be created. This would then be spread around the network, to prevent other computers in the network from becoming infected. The computer with the "Tripwire" access logging functionality could be set up in a "Honeypot" type fashion, ie with no critical services, and designed to be vulnerable.

45 Chess, David. "The Future of Viruses on the Internet." (This paper was originally presented at the Virus Bulletin International Conference in San Francisco, California, October 1-3, 1997.)
URL: http://www.research.ibm.com/antivirus/SciPapers/Chess/Future.html

Consequences

Viruses have varied consequences. The action a virus takes (other than spreading) is known as the "payload." Once detected, a virus will certainly take up resources – time and manpower. However, some viruses also destroy or alter data, deface web sites, use the modem to make toll calls, etc. Until recently, the payload had been focused on the victim, seeking to make their life difficult. A new development in virus authoring is using the virus to serve a specific purpose, beneficial to the author. Viruses are also (generally) having less damaging payloads than previously. Viruses written in assembler can contain any payload the author chooses. The Casino virus[46] copies the entire File Allocation Table into memory, erasing it from the hard disk drive. The virus then requires the user to play a virtual "slot machine". If the user loses, then the virus crashes the computer, meaning that the FAT is lost. As some modern viruses are written in scripting languages, this payload would not be possible. The only tools the virus creator has to work with are those included in the scripting language. This usually includes disk reads and writes, but probably not to specific disk sectors, and usually not reformatting drives. This somewhat limits the damaging potential of most viruses.

DoS

Some viruses are now being used as Denial of Service (DoS) tools. This is where each infected machine consumes some network resources, such that none are left for legitimate users. The virus is allowed to spread (usually over the Internet.) At a predetermined time, the virus uses some resource of the target. This could simply be connecting to the web site, or a ping (echo reply) packet, or sending them an e-mail. Either way, the goal is that the virus will be so widely distributed that the mass of requests from all the infected computers will cause the target server to fail. This will cause it to stop processing legitimate requests, thereby denying service to that site. This is a very difficult attack strategy to counter. An example of this type of virus is the CodeRed virus. This would send ping packets to the www.whitehouse.gov web site between the 20th and the 28th of each month, in an attempt to crash that server[47]. However, the virus had a hard-coded IP address for the White-house web site, which the Whitehouse were able to change before the virus struck – avoiding the DoS attack. The more recent "Slapper" worm exploited a flaw on Linux computers to spread. This worm created a massive peer to peer network with other infected computers, that could collectively be used to launch an arbitrary Denial of Service attack.[48]

Payloads

---

46 Metropolitan Network BBS Inc, "Casino.2330" URL: http://www.avp.ch/avpve/file/c/casino.stm

47 Symantec (Chien, Eric), "CodeRed Worm." July 16, 2001. URL: http://www.symantec.com/avcenter/venc/data/codered.worm.html (July 19, 2002)

48 Lemos, Robert. "Slapper worm smarting less." September 20, 2002. URL: http://news.com.com/2100-1001-958758.html

Various other payloads exist. The SirCam virus sends files from the victim's hard disk to a random e-mail contact[49]. This does not seem to serve any direct purpose, except providing the recipients with a voyeuristic thrill. The CodeRed virus opens a back door on victim machines, allowing scripts to execute code that was formerly unavailable. The BugBear virus installs a keyboard logger. This records keys typed, and sends them to the virus creator the next time that the computer is connected to the Internet[50]. This could include Internet passwords, or credit card numbers. Viruses no longer simply affect the infected machine, but can have global consequences. This dramatically increases a user's obligation to protect themselves, as it is no longer just themselves that they are protecting.

Counter-Measures

The Internet has made simplified the process of eliminating viruses. The remedy used to involve getting a new disk with updated viral definitions from the anti-virus team. Although laborious, this method was difficult to interfere with. Now, the Internet is the modality of choice for most users. The user suspects that they might have a virus, so they go onto the Internet and download a specific remedy (most Anti-Virus software does this automatically). In most cases it is just that simple. This means that virus definitions can be updated much more frequently. This is necessary due to the more rapid spread and decline in a particular viral outbreak. However, much as retroviruses used to target anti virus software, new viruses may create a DoS condition on anti-viral vendors' web sites. As the Internet becomes the usual way of getting software, other methods tend to fade away. If a virus could successfully maintain a DoS condition on anti-viral software teams, then it could potentially continue indefinitely. This would be impracticable, but a virus could increase its staying power by either targeting known response sites with a DoS attack, or just cutting off the Internet entirely (after spreading first, of course.) by attacking a local router, or even DoS'ing the local tcp/ip stack. This would prevent or delay applying any remedy to the machine in question.

Predictions

Viruses are changing. Viruses are different in several respects from their earlier forms. As new anti-virus techniques are developed, new viruses are discovered (or created) that avoid or even exploit them. Successful viruses are disassembled, and their techniques copied. There are several directions virus authors could pursue that have not been fully developed. Techniques should be designed to meet these possible threats before they materialise.

Multi-partitism

Modern viruses tend to have more vectors than previously. For example, the Nimda worm, could travel through various routes. It could use the Internet Information Service vulnerabilities used by CodeRed. It could attach itself to exe

49 Delio, Michelle. "'Sircam' Worm Getting Hotter." July 20, 2001.
URL: http://www.wired.com/news/technology/0,1282,45427,00.html

50 Sophos, "W32/Bugbear-A." URL: http://www.uk.sophos.com/virusinfo/analyses/w32bugbeara.html

files, as a traditional virus. It e-mailed itself out (as did the LoveBug worm). It infected http documents, and could spread through LAN shares. This is nothing new, the Morris Worm exploited several vulnerabilities itself. However, the sheer number of vectors exploited by Nimda is surprising.

Multi-partite evolution is likely to continue. One possibility is an "Octopus" worm or virus. This is a virus that is controlled from a central point. The viruses connect back to this point to get instructions, or updates. This central controller could update the viruses with new vulnerabilities as they are discovered, and possibly new encryption routines to prevent the virus from being detected. The viruses could not talk to the central controller much, to avoid creating a DoS condition on it. The Opaserv virus attempted to download updates from a central site hard-coded into its source-code[51], but the site now appears to be off-line (5th November 2002).

Alternatively, a virus could use levels of control, with several "zombied" computers distributing the information. Worms and viruses will use more and more vulnerabilities to spread. Much as macro viruses are platform independent, more traditional viruses are likely that can infect multiple platforms. An example of this is the recent Simile virus, which can infect both Windows and Linux operating systems. This virus was developed as a "Proof of Concept" virus, and has not been seen in the wild.

Another method using this approach would be a virus that understood file formats across different types of computer. The virus head would contain a "translator" function, which could go through the virus and modify the instructions to operate on the new target computer. The virus would need to know about the specific assembler instructions on the target computer, along with the file allocation table and executable file layouts. The virus would need to identify the target computer before attempting to copy itself there. Operating System fingerprinting can be done without even needing to send the target computer any information[52]. The virus would then reconstruct itself in memory to suit the target computer, and then copy the newly infectious copy to the target. The translation routine would need a look-up table that would not be modified from copy to copy, and unless encrypted this could be used to detect the virus between different types of computer. This type of virus could infect across all the different types of computer that it had been programmed for.

Infecting Different File Types

Another technique a virus could use is to spread itself around different files. The "Perrun" virus claimed to be able to infect JPEG format picture files. However, this virus required that the system already be infected with an extractor

---

51 F-Secure, "Computer Virus Information Pages: Opaserv". URL: http://www.f-secure.com/v-descs/opasoft.shtml

52 Miller, Toby. "Passive OS Fingerprinting: Details and Techniques." URL: http://www.incidents.org/papers/OSfingerprinting.php

executable[53]. It could be argued that the extractor executable was in fact the virus. However, this technique could be used to disguise a virus. The main executable could simply be a decryptor. The main viral code could then be hidden within any file on the computer. The decryptor executable would decrypt the viral body from the other file, and reassemble the virus in memory. This would make the virus much harder to detect when inactive, although when active a memory scan would still reveal it. The decryptor could be hidden by using standard polymorphic techniques. The decryptor could be attached to any standard executable, as with a regular virus, or inserted into the boot sector etc. The second file would not be "infected" per se, but just used to hide the body of the virus. This way only the polymorphic decryption routine would need to be attached to the executable, which would make detection more difficult.

Blended Threats

The line between a virus and a worm is clear in theory. However, several recently released pieces of malware have exhibited characteristics of both. The Nimda worm could both infect .EXE files, as a virus, and spread through IIS vulnerabilities, like a worm. Many viruses contain "logic bombs" – functionality design to cause damage if a particular condition is met. Although all these elements have existed previously, combining them into a single threat would make that threat harder to combat.

Serving a Goal

The future will likely see the creation of more "trojan horse" type viruses. These viruses will infect a system, and then create a back door, so the virus author can further exploit that system. The new version of VBS Worm Generator allows for the inclusion of arbitrary executable files[54]. This could be used to install a trojan horse (such as Sub7[55]) on infected systems. Further, a virus could install a back door, and then close off its own entry point. This would prevent subsequent viruses from exploiting the same vulnerability, as well as disguising the method of entry. Another development is viruses being bundled with "root kits", for example t0rn[56]. These are software packages that replace standard system tools to disguise certain actions. The "root kit" might replace the directory-listing command (ls on unix systems) with one that does not show certain directories. This "root kit" type virus would be harder to detect on infected machines, and could allow the machine to be used for other illegitimate purposes. The Lion worm used this technique[57]. Similarly it would be an effective technique to combine a worm's spread with the ability of an autorooter[58] to compromise

53 Symantec (Knowles, Douglas), "W32.Perrun." June 13, 2002. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.perrun.html (June 20, 2002)

54 Weil, Nancy. "Enhanced tool can create tougher Anna worm". March 15, 2001. URL: http://www.cnn.com/2001/TECH/industry/03/15/VBS.worm.generator.idg

55 HackGuard. "Sub7 Information." URL: http://www.hackguard.net/sub7.htm

56 Miller, Toby. "Analysis of the T0rn rootkit." URL: http://www.sans.org/y2k/t0rn.htm

57 Sophos, "Linux/Lion". URL: http://www.sophos.com/virusinfo/analyses/linuxlion.html

58 Tanase, Matt. "Introduction to Autorooters: Crackers Working Smarter, not Harder" August 21, 2002.

systems. Another possibility is that of "bespoke" viruses – ones written to affect and compromise a particular system.

Multi-stage attacks

Already viruses have knowledge of each other's work. The Nimda virus can spread using the "Root.exe" backdoor left by the CodeRed or Sadmind/IIS worm[59]. It is likely that this trend will continue. A virus will alter a system in a way that is difficult to detect but causes a security flaw. A later virus will exploit that flaw to spread, and possibly cause further harm. This cumulative effect means that even if the system administrator is able to work out the entry point of a virus, and close that hole, their system may still be compromised and vulnerable. Another possibility is that viruses may use their entry into a system to infect the system with other viruses. An example of this is the Klez worm. Having infected a computer, a variant of the Klez worm (Klez-H) deploys the Elkern virus onto that computer[60]. The Elkern virus does not reciprocally infect computers with the Klez worm.

Less User Intervention Required

Originally, e-mail was not thought to be a possible viral vector. E-mails at that stage could only contain text. People then devised ways to attach files to e-mail messages. To become infected by a virus required double-clicking an executable attachment. Then, through the "double extension" trick, clicking apparently non-executable attachments was enough. With Word (etc) macro viruses, opening a .DOC attachment can lead to infection. With Frethem, reading an infected message is enough[61]. And now, with Kak[62] and Bubbleboy[63] etc, simply previewing e-mail is enough to trigger a viral consequence. As new programs and operating systems have more features, it is likely that this trend will continue – and increasingly less user intervention will be required to trigger an infection.

Multiple Versions

Often several different versions of viruses have been released. The Frethem virus has over 12 known variants. This may be from the original author modifying and improving their code. Alternatively, it may be from later people obtaining copies of the virus (or source code) and changing it themselves, and releasing a "copycat" virus. It is probable that this will continue. A virus could be released with several different exploits, updating earlier versions of the virus. A virus could be constructed in a modular fashion, so new exploits are easily to add

---

URL: http://online.securityfocus.com/infocus/1619

59 Hayes, Bill. "VIRUS ALERT – W32/Nimda@MM Internet Worm" September 25, 2001. URL: http://www.unl.edu/security/virus_alerts/nimda.htm

60 Sophos, "W32/Klez-H". URL: http://www.sophos.com/virusinfo/analyses/w32klezh.html

61 Ikarus Software. "Worm FRETHEM.K (Variants FRETHEM.Family)". URL: http://www.ikarus-software.com/news/frethem.html

62 McAfee Security. "JS/Kak@M" 2002 URL: http://vil.nai.com/vil/content/v_10509.htm

63 McAfee Security. "VBS/Bubbleboy@MM" 2002 URL: http://vil.mcafee.com/dispVirus.asp?virus_k=10418

functionality for. Then, if a newer version of the virus discovered an old infected machine then it could just update the virus on that machine to include new infection techniques.

Random Polymorphism

Polymorphic viruses are also likely to continue to develop. A step not yet seen in the wild is a virus that randomly "mutates". Polymorphic viruses change their code instructions, but in pre-programmed ways that maintain the original functionality. A new type of virus might randomly change a single byte in its code, or several bytes. This could occur only infrequently, say 1 in 10 copies. If the virus was well designed originally, it might infect tens or hundreds of thousands of computers worldwide. Even if most of the mutated versions were useless, there could still be some functional modified copies. In simulations (ie the Tierra[64] software) this can be done safely simulated. Self-replicating programs in this environment evolve surprisingly quickly, to efficient new forms. In the wild, over the Internet, an evolving virus could be extremely difficult to contain.

Conclusions

Viruses will continue to develop, as do the hardware and software that supports them. 2002 has been a quiet year for viruses[65]. This is possibly due to harsher prison sentences for virus writers, or the goodwill in the aftermath of the tragic events of the previous September 11th. Either way, it is unlikely that this trend will continue. Continued computer evolution will attract interest, and inevitably a percentage of users will be drawn towards using their skills destructively. Faster Internet connections will lead to more sudden and virulent virus outbreaks. Although most viruses are now targeted at a specific OS/Software combination, there are enough big players that this is still a very large number of victims. Virus scanning technology and firewalls can prevent most of the damage, but an unexpected type of new virus could cause considerable harm before being contained. Systems should be designed with multiple layers of protection, to make a virus author's job as hard as possible.

64 Ray, Tom. "Welcome to the Tierra home page". URL: http://www.isd.atr.co.jp/~ray/tierra/index.html

65 Warner, Bernhard. "New computer security dilemma: lack of viruses." August 12, 2002. URL: http://www.infoworld.com/articles/hn/xml/02/08/12/020812hnnovirus.xml

Last Updated: August 17th, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Chicago 2017	Chicago, ILUS	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VAUS	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS San Francisco Fall 2017	San Francisco, CAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FLUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NVUS	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, IE	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MDUS	Sep 25, 2017 - Sep 30, 2017	Live Event
Data Breach Summit & Training	Chicago, ILUS	Sep 25, 2017 - Oct 02, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, DK	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, GB	Sep 25, 2017 - Sep 30, 2017	Live Event
Rocky Mountain Fall 2017	Denver, COUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, NL	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS DFIR Prague 2017	Prague, CZ	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS Oslo Autumn 2017	Oslo, NO	Oct 02, 2017 - Oct 07, 2017	Live Event
SANS October Singapore 2017	Singapore, SG	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS AUD507 (GSNA) @ Canberra 2017	Canberra, AU	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZUS	Oct 09, 2017 - Oct 14, 2017	Live Event
Secure DevOps Summit & Training	Denver, COUS	Oct 10, 2017 - Oct 17, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VAUS	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Brussels Autumn 2017	Brussels, BE	Oct 16, 2017 - Oct 21, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, JP	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS Berlin 2017	Berlin, DE	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS Seattle 2017	Seattle, WAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, AE	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Miami 2017	Miami, FLUS	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Amsterdam 2017	Amsterdam, NL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Milan November 2017	Milan, IT	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Sydney 2017	Sydney, AU	Nov 13, 2017 - Nov 25, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MDUS	Nov 13, 2017 - Nov 20, 2017	Live Event
SANS Paris November 2017	Paris, FR	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS Adelaide 2017	OnlineAU	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced