Twenty years before the mouse.

Aryeh Goretsky
Distinguished Researcher

Table of Contents

Title note:

With apologies to author Charles Erskine's "Twenty Years Before the Mast: With the more thrilling scenes and incidents while circumnavigating the globe under the command of the late Admiral Charles Wilkes 1838-1842." Boston (privately printed), 1890.

Introduction

For the past several years, I have been deep in the dark bowels of ESET, LLC's Research Department—as the department's Special Projects Manager, working on tasks that are vital at antivirus companies but generally go unnoticed by the public, testing things, making things, aggregating data from disparate sources, providing commentary and analysis and all the other myriad tasks one has to perform as a manager. One such regular responsibility is drafting new research topics alongside security expert Jeff Debrosse, the head of Research at ESET.

Jeff is no stranger to visitors of ESET, LLC's blog or readers of its white papers. He has written many of both and has been involved in the creation of others to varying extents. But as Senior Director, he is not just a frequent author or speaker but responsible for the operation of the department as well. And that means occasionally getting one of us to write a white paper. During a review last year, Jeff asked me how long it had been since I had started in the antivirus industry. September would mark my twentieth year of fighting computer viruses, or, as they're now called, malware[1], I replied. Jeff sat back in his chair and commented about what a good opportunity it would be for a white paper comparing those earliest days of the antivirus industry with how things stand today.

I am afraid this will be less of a normal white paper and more of a personal retrospective, a fact that made it much more difficult and far longer to write than I had anticipated. However, I think readers will find it helpful to help bring to light the earliest days of the antivirus industry and see how far we have progressed—or have not—in the intervening two decades.

So, Jeff, thank you for the pushing and prodding to get this paper done. I would also like to thank my fellow longtime computer virus combatants David Harley and John McAfee for the feedback they provided during innumerable revisions of this white paper.

[1] A portmanteau of *malicious* and *software*.

Fiat Lux

One of the advantages of growing up in Silicon Valley was my early exposure to computers. From the late 1970s onward, I had used a variety of 8-bit computers, such as the Apple II series and the Commodore 64 and had even gained a little bit of experience with IBM's PC XT.

After graduating from high school, I went on to college and found myself looking for a part-time job. In the middle of September 1989, I asked my friend John McAfee for a job, and, surprisingly, he agreed. At the time I knew John as an eccentric BBS[2] operator who had a side business selling antivirus software out of his house. I had even spent several afternoons stuffing envelopes for a computer antivirus industry association he had started. By this time, John had already received several mentions in the press and had been interviewed on television due to the Datacrime [1] virus and Morris Worm [2], and it seemed likely that this was going to become his full-time occupation. I figured that he would need someone to answer phones, type letters, provide technical support and perform whatever other front-office type functions he needed. Thus, I became the first full-time employee at McAfee Associates.

That very first day, John explained to me how viruses operated in MS-DOS on the PC by drawing a series of boxes on a piece of scratch paper, with each box representing a program instruction, and then showing me how a virus would prepend instructions to the beginning of the file, add its code to the end and then transfer execution back to the original instructions. In short, how a simple parasitic file-infecting virus added itself to a .COM file. He also explained how some viruses were designed to simply overwrite the beginnings of files but that these were unlikely to remain undetected for very long because the damage they caused was readily apparent. With these five minutes of instruction, I was ready to take my first phone call as a technical support engineer. Later that day, the first call came in: a request for assistance with the Pakistani Brain [3, 4] computer virus, and I screwed it up completely.

Brain Damage: Rootkits 1980s-Style

First, a little history lesson is in order: The Pakistani Brain infected the *boot sector* of a floppy diskette. A boot sector is the very first physical sector on a floppy diskette that is read into memory after the computer has completed its POST [5] (Power On Self Test) process to begin loading an operating system [6]. For MS-DOS (PC-DOS used different filenames), the boot sector would load the operating system's kernel files, `IO.SYS` [7] and `MSDOS.SYS` [8], which, in turn, would load `COMMAND.COM` [9], the command interpreter. The boot sector even checked to see if these files were not present and, if so, displayed a `"Non-system disk or disk error. Replace and strike any key when ready."` error on the screen to alert the user. For all this functionality, the boot sector is not a file but rather a small assembly language program shoehorned into the first sector on a diskette, which is 512 bytes in length[3]. What the Pakistani Brain did was to copy the original boot sector to the end of the disk, along with the remainder of the viral code, as the Pakistani Brain virus was too large to fit into a single sector. It would then overwrite the original boot sector with a program that loaded and ran the virus and only then finally loaded the original boot sector. To prevent its code from accidentally being overwritten, the virus marked the sectors containing it on the diskette as bad, which made them unavailable for file storage.

Once the virus was in memory, it would check diskettes as they were accessed to see whether they contained the virus and, if not, infect them. As a method to avoid detection, the virus would monitor any attempt to access the boot sector and redirect it to the saved copy. This technique of redirecting attempts to see the viral code to a clean object is called *stealth* and is commonly seen today in other malware, notably rootkits. It is important to keep in mind, though, that the Pakistani Brain computer virus dates back to at least 1986 and there are hints of a prototype, Ashar, from even earlier, although the exact date is a matter of some conjecture.

[2] Bulletin Board System

[3] Actually, the assembly language code had to be 440 to 446 bytes in length, with the remaining space reserved for the partition table of data and the signature for the master boot record.

On-the-Job Training

So, what happened with my first technical support phone call? After several unsuccessful attempts to find out from the customer what *file* was infected with the virus, I ended up handing the phone to John McAfee, who promptly told the user to copy the files off of the floppy diskette and reformat it. At this point, I was terrified: I had completely and utterly failed at answering my first question about removing a computer virus and figured my first day on the job was about to become my last.

I looked at John, unsure of what to say to him. John looked at me and, without saying anything, grabbed another piece of scratch paper and then started to draw a concentric series of circles like an onion. He then began explaining how these were the tracks on a diskette, that they were divided up into sectors and how the first one contained the boot sector. For someone who had grown up using computers where the operating systems were embedded in ROM, the concept of having to boot an operating system from a diskette and loading it into RAM before being able to use the computer was somewhat foreign and took some getting used to, despite John's helpful drawing. But I eventually understood how PCs booted up, from initializing their hardware and then testing it to passing control off to software.

This process was the beginning of a bootstrapping sequence that eventually resulted in the loading of an operating system, all ready to run applications. I learned the points at which these processes could be subverted to load malicious software, how to check them and then clean them.

While computer viruses were the dominant form of malware throughout the late 1980s and early 1990s, they were not the only form of malware. BBS systems, like corporate mainframes, were sometimes the targets of logic bombs and Trojan horses uploaded by people who sought to crash them; and the Christmas Tree Exec worm and Robert T. Morris Jr.'s Internet Worm had all made their presence known.

One of the first PC-based malicious programs to receive widespread attention was, in fact, not a virus (i.e., self-replicating malware) at all but rather a Trojan horse program, or "trojan" in the modern parlance. Much like their Greek namesake, Trojan horse programs purport to do something useful but, instead, intentionally perform some malicious activity. While the definition is somewhat vague, a good definition of a Trojan horse program is "a program you wouldn't want on your computer if you knew what it actually did."

Ransomware: Then and Now

In December 1989, the AIDS Introductory Information Diskette Trojan horse [10] became the first Trojan for PCs to receive widespread media coverage. This was not because of its success—perhaps a few hundred computers were affected by it—but because of the mechanisms surrounding its distribution: The Trojan horse was mailed on diskette to perhaps 10,000 recipients. It came with a sheet explaining how it allowed them to assess their knowledge of the AIDS virus and risk factors for infection. It also contained a license agreement explaining the program must be licensed by sending payment to a post office box in Panama. When a person ran the program from the diskette, it gave them something more than a quiz: The Trojan horse modified the `AUTOEXEC.BAT` file on the hard disk drive to keep track of the number of times the computer was started and, after ninety boots, encrypted filenames on the hard disk. At this point, the extortion began with a harsh warning that a license was now required.

Although we had not yet received a sample of the Trojan horse (sending floppy diskettes over modem was expensive and time-consuming in 1989), John had received a faxed copy of the sheet that came with it. One of my first tasks was to go to the local copy shop and get paper in the right size (5.5" × 8.5") and color (light blue) as the sheet. I then retyped the message from the fax, choosing a typeface and layout as close to the original message as possible, and then printed it

out on the paper. John showed this printout to the reporters who came by his house, and it appeared on several news broadcasts for the next few days.

The author of the program was eventually identified as Dr. Joseph Popp. Far from being enriched by his scheme, Dr. Popp was arrested but did not stand trial due to being declared mentally unfit. Perhaps the only mystery about this Trojan horse was how the doctor was able to create such a program and distribute it: Doing so required renting mailing lists, duplicating the diskettes and shipping them. According to Robert Slade's "Guide to Computer Viruses," four principals apart from Dr. Popp were identified. I am unsure if any of them were criminally charged.

As for my transcription, when I asked John about whether it was right to be showing such a thing to reporters, he reminded me that he had not claimed that it was the actual paper from the Trojan horse's author. On the other hand, he did not go out of his way to state that it was a quickly transcribed copy. This was one of the first indications I had that my new career was going to be interesting.

Perhaps the most notable thing about this attempt at *ransomware* [11] was that it was the last we would see of any high-profile malware-based extortion schemes for about fifteen years. While there have been several notable attempts at computer-mediated extortion, these largely relied on the use of remote-access programs to steal data and then sell it back to the owner. It was not until late 2004 that a series of Trojan horse programs appeared in Russia that encrypted the data on computers and then displayed instructions on how to purchase a decryption key. While such attacks normally fail when criminals attempt to collect their funds, the author(s) of the Russian ransomware have kept the price of decryption at nuisance levels (sometimes as low as about $20.00) and made use of electronic payment through services such as e-Gold and Yandex.

War of the Parasites

Viruses like the Pakistani Brain virus, which affected the boot code of floppy diskettes and later that of hard disk drives, continued to be a problem through the 1980s until the release of Windows 95. However they typically travelled no faster than a diskette could be couriered overnight or sent via mail. Since floppy diskettes were the most common means of distributing software, though, infections did spread effectively, albeit at a slower pace than today.

It is important to keep in mind that in the 1980s and into the early 1990s, networked computers—let alone Internet access—were relatively uncommon and expensive, generally limiting their installation to large organizations. Mailing diskettes or carrying them from computer to computer were typical ways to install software and spread computer viruses: It was not uncommon for technicians upgrading computers' software via "sneakernet" to infect their floppy diskettes when deploying software or updates, subsequently transferring the virus (or, sometimes, viruses) to every other computer on which they worked.

Unlike their boot-sector-based brethren, file-infecting viruses could spread as quickly as they could be downloaded using a modem over BBSes, commercial online services or the fledgling Internet.

Initially, file-infecting viruses were quite simple: They affected more simply structured .COM files (as opposed to more complex .EXE files), typically either prepending or appending their code. Few were memory-resident and were instead "direct action" file-infecting viruses, spread using simple methods such as infecting one uninfected file at a time in the current directory when an infected host file was run before passing control back to the host program.

As virus writers developed their skills, file-infecting viruses bloomed. Some of the techniques they used included:

- Memory residency. MS-DOS was not designed with multitasking (or networking) in mind but was rather an operating system that hosted a single user running a single application at a time. It was, initially, quite rare to see a system that needed to load memory-resident programs such as device drivers or TSRs [12], since computer manufacturers sold copies of MS-DOS modified to support whatever custom hardware add-ons they provided. Only as the IBM PC's hardware ecosystem expanded to encompass third-party vendors did it become common to load device-driver software to support new hardware such as sound cards and graphics cards, and we saw the dawn of IBM-compatible computers. Likewise, TSRs were popularized by programs such as Borland's Sidekick [13, 14], which allowed the operator to switch quickly to a personal information manager regardless of what other program was currently running.
- .EXE file infection. As mentioned previously, file infectors initially only infected .COM files, which had a simple structure—assembly language programs of 64KB or less that were loaded into a bank of the computer's memory and executed. In such cases, it was simple to prepend or append viral code to a .COM file so that it could perform whatever actions it needed before passing control to the host program.

  For more complicated programs that could not fit into a single 64KB bank of memory, DOS provided .EXE files, which had a more complicated structure, allowing software authors to create larger, more complex programs. Commonly, file-infecting viruses were designed to append their code to the end of .EXE files and patch the beginning instructions of the infected program file so that the viral code was jumped to and executed before passing control back to the host; however, there were also viruses that prepended their code.

  Regardless of where virus code was added to a file, it increased the size of the file on disk. A quick way to determine whether or not a virus was present was to list a directory, examine the size of the files in it, then run one (or more) of them, issue another directory command, and look at the increase in file sizes of the infected files. In some instances, it was possible to determine the family or even the specific virus that infected a system simply by looking at the increase in file size.

  This was not a substitute for running antivirus software, but it did indicate that a computer virus was present. As virus writers grew more sophisticated, they came up with more sophisticated ways to store viral program code in files: Some viruses looked for slack space in files and placed their code in them. Others split their program code throughout a file so that no large sequences of the virus appeared exclusively at the beginning or end of a file—a technique designed to avoid detection by virus scanners that looked only in these locations for viruses as a means of boosting performance.
- Fast "on-access" infectors. The first file infectors were not memory-resident and typically worked by searching the current directory for new hosts and infecting the **first** uninfected file they came across. This may seem like a slow method of spreading an infection—not to mention cumbersome—however, it is important to remember that during the earliest day of the IBM PCs, most computers were only equipped with a floppy diskette drive (or two) and most computer users were quite aware of how long it took for their computer to boot and run various programs.

  In some cases, they were used to the specific sounds from their floppy diskette drives as these occurred. The additional time required to seek out files, coupled with the noise of the floppy diskette drive's read-write heads as they moved over the surface of the diskette—were sometimes enough to notify an alert computer operator to the presence of an infection.

  As computers became more powerful and hard disk drives became more common (and quieter), observation of a virus's infective activities became less of a risk for virus writers, and as memory residency became more common, they created viruses that would search for uninfected files when *any* file I/O operations were performed, such as listing directories and copying files. It also became practical to infect all the executable files in a directory and search across the hard disk drive for uninfected files to contaminate.

- Stealth. When early viruses like Pakistani Brain were resident in a computer's memory, they prevented access to its boot sector code on infected diskettes by redirecting attempts to access the boot sector to the location where it stored the original code. Such stealth techniques were adopted by other virus writers, as well. A memory-resident file-infecting virus could monitor attempts to read program files and, when it saw that a file being loaded contained a copy of itself, could remove the actual viral code from the file before transferring itself into memory, thus allowing whatever program was accessing the file, such as a disk editor (or even antivirus software), to see a clean, uninfected copy.

  Such techniques could be bypassed by booting the computer from the original floppy diskettes containing the operating system, which were typically shipped on notchless or write-protected[4] diskettes, but sometimes the computer operator did not have them or they were already infected, whether by the operator in the course of trying to troubleshoot the problem or because they were shipped that way with the computer.

Mass outbreaks typically occurred when shared computers became infected. School computer labs sometimes became hot zones, and infections became pandemic when diskettes containing viruses were duplicated and distributed as commercial software, driver diskettes for hardware, magazine cover disks and so forth.

The 1980s to the mid-1990s marked the heyday of MS-DOS, and the number of DOS viruses infecting it grew from a handful to tens of thousands before the arrival of Windows 95, which marked something of a change in the threat landscape.

File-infecting viruses were legion, prepending or appending their code to executable files. Those that damaged their hosts did not, generally speaking, survive for very long. One exception to this was the Jerusalem virus [15]. One of the earliest .COM and .EXE file infectors, the Jerusalem virus contained coding errors that caused it, over time, to damage some .EXE files as it appended its code to them. One of the features of the .EXE file structure was that it gave programmers the ability to load a program overlay into itself in order to get around the memory constraints of DOS.

As an example, a word processor might have separate modules for a thesaurus and spellchecker, displaying a graphical "WYSIWYG" representation of a document[5], equation editing and document printing. To take advantage of this functionality, a program needed to contain *stack space*, empty locations within the main program file. Once the program was in memory, what were empty spots inside the file on disk became the location into which subprograms could be loaded and run from disk. Two bugs in the Jerusalem virus's code caused the virus, not to mention infected users, quite a bit of difficulty:

1. Viruses often placed a marker inside infected programs in order to detect whether or not they had already been infected. Markers might be a specific string of data or even part of the virus's own code. If the marker was present, then the virus would not infect that program. The Jerusalem virus placed a marker in infected files; however, it did not correctly detect its own marker in .EXE files, causing it to reinfect .EXE files over and over again. Each time an .EXE file was run on an infected system, it grew by approximately 1,863 bytes.

   One small advantage of this coding mistake was that it made it easy to detect the presence of the Jerusalem virus: All one needed to do was note the size of an .EXE file, then run it and check the size of the file again. This technique worked for many other file-infecting viruses as well, although a clean, uninfected file had to be run each time if the virus properly recognized its infection marker. An effect of the continuous .EXE file growth from the Jerusalem virus

[4] On the hardware architecture for the IBM PC, a notch was cut into the upper-right corner of a 5.25" floppy diskette's outer jacket to indicate that a disk was writeable. Boxes of blank floppy diskettes came with sheets of opaque stickers, which were applied over the notch to indicate a disk could not be written to. The actual mechanism was typically an infrared photocell inside the floppy diskette drive mechanism, which, if light was admitted, allowed write operations to the diskette.

Many years ago, I had a long conversation with a customer who was convinced he was infected with a computer virus that bypassed this hardware-based write-protect mechanism used on floppy diskettes. After spending over an hour on the phone with the customer, verifying the diskettes were write-protected, inserting them into the computer and listening as he described the increase in file size as programs became infected, it finally occurred to me to ask him what sort of stickers he was using to write-protect the diskettes. It turned out he was using cellophane tape, which is generally transparent. After a short discussion about infrared light and the visible spectrum, I was able to successfully close the incident.

[5] Due to the speed of processors, video chips and available RAM, most business applications under DOS displayed text mode screens in order to provide enough performance for users. Some applications offered graphic display modes, but this was largely for preprint proofing, as in graphics mode a it might take several seconds to refresh the screen.

was that they would eventually no longer fit in memory, causing DOS to report an error when the computer operator tried to load them.

2. When infecting .EXE files, the Jerusalem virus did not look at the size of the file on disk to determine where to append its viral code but instead looked at the size of the file indicated in a program's header. While these values were nominally the same, .EXE files, which loaded overlays, stored the location at which overlays were supposed to load there instead. This meant that the Jerusalem virus would stick its program code somewhere inside the file instead of appending it at the end.

While neither of these two bugs in the code of the Jerusalem virus was fatal enough to prevent the virus from spreading to a large number of hosts, in combination the errors they caused could lead to programs either crashing or not running at all, leading an astute computer operator to check for viruses.

If this was not enough, the Jerusalem virus periodically displayed a black box on the screen and, on every Friday the 13th, would *delete* program files instead of *infecting* them when the user tried to run them.

The Jerusalem virus was prevalent and its code was widely copied and modified by inexperienced virus writers. The most frequently changed actions in the viral code were the color and location of the on-screen box and the trigger date for file deletion, largely because those were very simple modifications. Fixing recognition of the virus's marker and appending its code correctly to overlay-loading .EXE files were not done as often. Perhaps as novice virus writers began to master assembly language programming, they became more interested in writing their own creations from the ground up instead of fixing others' code. Or perhaps they didn't care.

Quite a few viruses used dates as triggers for their logic bombs, and, on Friday the 13th, Michelangelo's birthday and other trigger dates, it became common at antivirus companies to ramp up technical support by having everyone take support calls, regardless of whether they were in sales, accounting or even John McAfee himself. At McAfee Associates, the only exception to this press-ganging was the programmers: One of the senior programmers, after a couple hours' long phone session attempting to help remove a particularly stubborn piece of malware, finally recommended that the customer remove the hard disk drive from the computer, drive a wooden stake through its platters and bury it upside down at a crossroads. As endearing as this was, it was decided then that programmers should not speak to customers.

Writing Viruses for Fun and Profit

Shortly after my arrival at McAfee Associates, I heard for the first time the accusation that antivirus companies were responsible for writing computer viruses. As a matter of fact, I still hear it today. In those early days, it was not uncommon for people to call, write or fax us (remember, Internet access was still relatively uncommon in the early 1990s) just to argue that there were no such things as computer viruses, or, that if there were, it was because we were responsible for creating them. The callers usually prefaced their comments by announcing that they had a Ph.D. in computer science, taught computer science at a university or had a CNE[6]. In their defense, there was not much reliable information about computer viruses in those days and the type of information that did tend to get passed along through BBSes, user groups and regional computer magazines was often lacking. Once the pontification was done, though, it was usually quite easy to explain programming concepts such as memory-resident programs and self-modifying code and how by those two techniques together one could create parasitic, replicating programs.

As for the actual writing of malware, or commissioning thereof, I will categorically state that during the time I was at McAfee Associates (1989 through 1995) we neither wrote viruses ourselves nor paid for them to be written. In many cases, though, we did provide the people who sent us new viruses with updated copies of our software for free to use for removing the threat.

[6] Short for Certified NetWare Engineer, a profession accreditation to indicate the possessor was capable of setting up and managing a network based on Novell NetWare as well as being capable of badgering antivirus technical support engineers for hours. Often when they needed to go the bathroom. Badly.

In one case, I recall we received a floppy diskette from a library that was infected with a virus that our software was unable to remove. The librarian, citing the usual dismal financial status of libraries everywhere, requested that the diskette be returned after our analysis. Not wanting to send a potentially infectious diskette, we instead shipped back a box of blank diskettes. A nice bonus for the cash-strapped library, but one I doubt would lead librarians to a life of virus-writing in an attempt to extort office supplies from antivirus companies.

In the late 1980s, we did occasionally receive requests for viruses from enterprise clients wishing to evaluate the software, as well as from journalists wishing to review our software. Once in a while viruses were sent out in response; however, for us this practice ended in the early 1990s. One of the journalists who reviewed antivirus software at a magazine used to ask McAfee Associates for viruses every time a review came up. Although the magazine no longer exists, at least in print form, he still reviews anti-malware software, and when contacting ESET to review our software, requests actual malware to show off our software's capabilities. This is not provided, and while our products never score the highest in those reviews, they don't do too badly, either.

So, if we didn't write the computer viruses ourselves, or pay for them to be written, then where did they come from?

Well, most of them were uploaded to our BBS. With Internet access still rare outside of academia, government and big business, and online services such as CompuServe still considered expensive for most home users, most people decided to send us computer viruses by uploading them to a protected section of our BBS. This was checked several times a day—often hourly—for submissions, which were then copied onto floppy diskettes and carried over to the programmers for examination. For floppy diskette boot-sector infectors, we received a stream of diskette mailers through the mail and courier services.

Of course, those are not the only ways we received computer viruses, and the truth is sometimes stranger than fiction.

Throughout 1989 we had received reports of a virus called Disk Killer [16] that displayed the following message:

```
Disk Killer -- Version 1.00 by COMPUTER OGRE 04/01/1989

Warning !!  Don't turn off the power or remove the diskette

While Disk Killer is Processing!

PROCESSING
```

Unfortunately, by the time that message appeared, the computer's hard disk drive was corrupted and we were unable to get an intact sample of the virus.

One day, though, we had a bit of luck: A call came in from a gentleman whose computer was infected with the virus. While we could not help him, he had isolated the infection to a new laptop that was purchased from the U.S. office of a Taiwanese notebook vendor. As luck would have it, this office was only several miles away from John's house in Santa Clara.

John quickly put on a tie, grabbed a sports jacket and a badge he had with "SECURITY OFFICER" emblazoned on it, and off we drove down Montague Expressway to Milpitas.

Once there, John identified himself to the receptionist as being with the Computer Virus Industry Association (a trade organization he had established), showed them his badge and asked to speak with whomever was in charge. I waited in the lobby, trying to make small talk with the receptionist ("Have you worked here long?" "How do you like working here?"), while John spoke with the owner. He emerged a few minutes later with a handful of floppy diskettes and a very nervous owner in tow. He asked me to write the man a receipt (I had a notepad with me), and we returned to John's house. Within a few hours, John had determined the virus was a boot-sector infector and released a new version of VIRUSCAN to detect it.

While we did make a handful of other visits to local offices with computer virus outbreaks, none of them were quite like that.

Somebody Set Us Up the Bomb

Computer viruses, despite their comparatively small size, had to be fairly complex programs, capable of installing themselves in memory, handling file and disk I/O (Input/Output) activity to spread their code, checking for their markers to avoid accidentally reinfecting programs, and so forth. This required some skill at assembly language programming, and it was easy for novice virus writers to accidentally introduce bugs into their program code.

When viruses did not work properly, they could unintentionally crash programs and systems, which made it that much harder for a virus to spread and made it more likely that the computer operator would begin checking for them. Much like their biological counterparts, computer viruses have to strike a balance between infecting and killing their hosts in order to ensure that they can spread to a large enough population before causing such unintentional catastrophic damage as might lead to their premature extinction. Of course, there is always intentional, willful damage programmed into a computer virus by its author(s), which is another matter altogether.

One of the hallmarks of computer viruses of the DOS era was that many contained a payload of some sort. Some of these payloads were rather nasty logic bombs and did things such as deleting files or erasing hard disk drive volumes when a specific trigger occurred, such as a particular time or date being matched. Others might include a component of a more graphical or audible nature, such as moving a dot across the screen (perhaps in bizarre homage to the video game Pong), causing all the letters on the monitor to cascade [17] to the bottom of the screen, displaying subtly mocking messages to the user or playing music over the PC's buzzer.

Of course, some viruses were just *spreaders*, without any intentional trigger mechanisms or payloads. Without those, the viruses were not considered very interesting unless they had a novel means of spreading or infecting. Some viruses did contain messages inside their code. These were not displayed on screen but present as a way for the authors to take credit for their creations, as boasts to other virus writers or to taunt antivirus vendors.

News reports of the day tended to emphasize the payloads of viruses, which lead to them being popularized in other forms of media, such as books and movies. I think this influenced a generation of virus writers to conceptualize their programs as pranks[7] and had to contain some method of making their presence known to the computer operator. After all, a prank without a victim was not a very good prank, at least from the virus writer's point of view.

Every so often, a computer virus writer—or someone claiming to be one—would call the office asking to speak to John McAfee. Being a rather busy person, these calls were often routed to me instead. This was before Caller ID was readily available, and I am sure they took some steps to mask their location. Here's what I learned from these callers:

- Age-wise, they tended to be in their teens, typically around fourteen to sixteen years of age. One virus writer stated that he was twelve, and occasionally others said that they were in their early twenties. All were male.
- Those of school age claim they went to school (not altogether unsurprising) and received excellent to average grades.
- Some specifically went out of their way to state that they were not stereotypically awkward geeks or nerds, as portrayed by the media, but were instead popular, interested in sports and even had girlfriends.
- They generally lived in two-parent homes, which were upper middle class or wealthy. Keeping in mind that PCs were much more expensive than today, that even an affluent household might contain a single computer for the entire family, and that dial-up connectivity meant local or long-distance calls to BBSes or expensive online services, this translated into the likelihood that the virus writer had his own computer and a dedicated phone line for its modem.

[7] Of course, pranks can sometimes be very severe to the point of being fatal.

Despite having seemingly privileged backgrounds, one other thing they all had in common was that they relished the destruction their creations caused. They believed that anyone who was affected by one of their computer viruses *deserved* to be infected, that they were completely blameless and that the fault lay entirely with the infected party. While such a complete lack of empathy sounds like it borders on the sociopathic, my theory is that this was actually a combination of an inability to accept responsibility for their own actions due to immaturity and feelings of being completely powerless in their daily lives. Their only outlet was to write computer viruses as a means for getting back at parents, teachers and anyone else they thought had slighted them. The fact that they tended to disappear from the virus-writing scene after reaching about college age enforced this perception

One thing that I want to make clear is that these callers were most likely from the United States or Canada, judging from the accents and quality of the phone lines. I understand that in other parts of the world the people who were writing computer viruses—and their motivations—were very different.

To this day, I think the public perception of malware is still somewhat stuck in this two-decade-old preconception of virus authors and virus behavior. Parasitic, recursively self-replicating programs—e.g., classic computer viruses—today account for less than 10% of the malware that we come across, and none of those viruses contain a payload that is anything remotely resembling a prank.

Just as the authors of yesterday's viruses grew up, today's malicious ware has grown up as well. From adware to browser redirection, spyware, rootkits and now fake antivirus programs, today's malicious software authors are motivated by profit—how much they can steal from others matters far more to them than any feelings of superiority or revenge. Or perhaps those teenagers who wrote viruses grew up and decided the best revenge was living well…by stealing from others.

Profits of Doom

Like the Pakistani Brain virus a year before it, the Stoned [18] virus infected the boot sectors of floppy diskettes; however, unlike Brain, it also infected the Master Boot Record[8] code on hard disk drives. As hard disks became more commonplace in computers, this provided an additional method for the virus to spread. Once a computer was booted from its infected hard disk drive, it went on to infect all of the floppy diskettes it accessed. The Stoned virus itself was notable for three things:

1. It contained two messages, "`Your PC is now Stoned!`" and "`LEGALISE MARIJAUANA!`". The first message had a one-in-eight chance of being displayed when an infected disk was booted. The latter message, with its non-American English spelling of *legalize*, was never displayed.
2. The Stoned virus was endemic throughout the DOS era, partly because floppy diskettes were a dominant means of exchanging files and partly because the virus was accidentally duplicated and distributed numerous times on commercial diskettes. This, in turn, led to it becoming further widespread. As copies of the virus were readily available, it was modified dozens of times. Some of these variants were trivial, such as changing the messages in the virus or the location where they stored a disk's original boot code. Others were more complex, such as Azusa, Empire, Monkey and NoInt, to name a few.
3. The Stoned virus begat the Michelangelo virus.

Ah, yes, the Michelangelo virus. While other viruses periodically received some attention in the media, the hype surrounding the Michelangelo virus was like nothing we had seen before. After first being detected in New Zealand in April 1991, it had spread around the world. Unlike Stoned, though, its payload was not the semi-random display of a prankster's message but rather something more sinister: The virus checked the date each time it was run. If the date was March 6th[9],

[8] On a hard disk drive, the Master Boot Record describes the number and sizes of volumes the hard disk drive is partitioned into, as well as which of the volumes contained the boot sector to load an operating system. This allowed an IBM PC to support multiple operating systems, one of which could be booted from the hard disk drive, and others that could be accessed by bootstrapping the machine from an operating system's bootable floppy diskette before going on to load the rest of its code from one of the non-bootable disk volumes.

instead of spreading, the virus wrote random garbage to the first seventeen sectors of the first 255 tracks on the first four cylinders of a hard disk drive.

This overwrote the master boot record on the hard disk drive; the boot sector of the first volume on the drive; and, depending upon the configuration and size of the disk, one or possibly both copies of the file allocation table (FAT), which is the index of where files are stored on a disk. Without the FAT, there is no way to tell which clusters of sectors are in use on a disk volume or which files they contain. So, in other words, successful activation of Michelangelo's payload left the computer operator with a corrupted hard disk drive. On the very largest hard disk drives, it might be possible to recover some of the data stored on it if the virus's damage routine had left the second copy of the FAT intact, but that was rare, and the cost of recovering data from a hard disk corrupted by Michelangelo was prohibitive.

So, we had a virus that had spread globally and a damaging payload with a trigger almost a year away. What we didn't have, though, was a good scope of how prevalent the virus was. Which, by the way, was nothing unusual for the time: There were many computer viruses that had triggers and damaging payloads, and antivirus programs of the era had no way to provide telemetry by "phoning home" to their developers with statistical information.

A reporter contacted John McAfee about the Michelangelo virus to ask him for an estimate of how many computers were infected by the virus. John replied that he didn't know; that it could be 5,000 or five million, but that due to the lack of data there was no way to be certain. What otherwise would have been a footnote in a conversation became a statement. In much the same way that viruses "evolve" and "mutate" through modification by virus writers, John McAfee's statement "evolved" from "as many as five million" to a hard figure of "five million" infected PCs.

What happened next was nothing short of amazing.

The phones had begun to ring. Constantly. One and a half seconds after we ended a conversation the phone began to ring again (that was the fastest the phone company could switch the circuit). All of the phone lines on the BBS's modems became constantly in use as people called in to download McAfee's antivirus software. John scrambled to add more telephone lines and modems to the system. With people being unable to reach the BBS to download the software, we began to tell them to instead visit CompuServe and download from there. Our nascent forum on CompuServe became the most heavily trafficked section in their history as people went there to download antivirus software.

When people could get through to us on the phone, it was to ask how to determine if they were infected, to report that they had a found a virus after running our software and ask for assistance in disinfecting their computer, or to buy our software. We were so overwhelmed that we eventually began to ask people calling to make a purchase just to download it, use it and send us a check later.

In the first week of March we found ourselves working eighteen and finally twenty hour days, trying to deal with the rush of calls and faxes and letters and messages on our BBS. Journalists and television reporters came by the office constantly to interview John McAfee and provide their audiences with a view from the epicenter. At one point, I was sitting at my table with three national network camera crews surrounding me, which made it difficult to get out of my chair.

Finally, March 6th arrived. All of us were exhausted, glad that the insane rush was over and looking forward to picking up the scraps of work we has discarded in order to respond to the tidal wave of communications that had engulfed us, not to mention our personal lives.

Afterward, the antivirus industry—and John McAfee in particular—received a lot of criticism for our handling of the

[9] One of the first antivirus researchers to look at the then-unnamed Michelangelo virus looked up the March 6th date to find out what notable events occurred on that date. What he found was the birth of a certain Italian Renaissance artist, and the name stuck. Interestingly enough, we received an international long-distance phone call from someone claiming to be the author of the virus and demanding to speak to John McAfee. When the call finally reached John's phone, an obviously angry, young-sounding man with a New Zealand accent told John the virus had nothing to do with Michelangelo's birthday and then hung up before John could utter a reply.

virus. Some accused us of scaremongering, and many competitors were incensed that we had capitalized on the online distribution of software through our BBS, CompuServe and the Internet to respond to the threat posed by the virus while they had no software to provide for download and could not get boxes onto the shelves of computer stores quickly enough to take advantage of the publicity. The Michelangelo virus had another effect on us as well: Despite having to turn back sales calls during the threat, sales grew fourfold afterward as organizations decided they needed antivirus software.

One question you are probably wondering about is whether all of the hype was justified. Was the amount of damage caused by the virus worthy of the attention it received? At the time Michelangelo occurred, our data collection efforts were manual and spotty, but we had about reports of about 30,000 infected computers in the weeks leading up to March 6th, and on that day received reports of another 30,000 or so damaged by it. So, the number of Michelangelo-infected computers reported to McAfee Associates was approximately 60,000 computers total. Of course, that does not include all the people who found infections and did not contact us, found and removed the virus using a competitor's product, or went to go use their computers one day and found that the hard disk drives had mysteriously crashed.

In 1992, McAfee Associates was one of perhaps two dozen antivirus companies, some of which were larger than we were. While the actual number of computers affected by the Michelangelo virus will remain a mystery, I suspect it was far higher than ever reported.

Interestingly enough, on March 6th, between all the calls from people requesting assistance with their infected or damaged computers, I received a call from an American expatriate living in France. He had set his clock ahead a few days in order to avoid the date-sensitive payload in the Michelangelo virus, a technique that had been popularized in the media. Unfortunately, his computer was, in fact, infected with a virus named *Maltese Amoeba* [19], which displayed a bit of poetry from William Blake and erased the beginning of his computer's hard disk drive.

One of the consequences of Michelangelo was the use of date-specific triggers by virus authors, typically with damaging payloads, such as deleting files or, like Michelangelo, corrupting disks. Of course, other triggers, such as the number of infections, system time or probability (e.g., a one-in-*n* chance) were popular as well. Some antivirus companies began to provide calendars of virus trigger dates so that customers would know when to be extra vigilant and, perhaps, to let reporters know when to contact them for a story.

These days, it is rare for malicious software to have any date-specific trigger mechanisms or malicious payloads, as the authors of such programs are far more interested in making money from infected computers than destroying them. And, the last calendar I saw from an anti-malware company just had regular holidays listed on it.

In the past two decades we have seen several other pieces of malware rise to media prominence, and in the early 2000s it seemed to become an annual event, with worms such as ILOVEYOU [20], Code Red [21], Blaster [22] and SQL Slammer [23] spreading so quickly they caused problems with email, database and web servers, not to mention home computers.

By the mid-2000s, network-clogging malware had become less prevalent, in part due to better security but also because malware authors were looking for ways to monetize their creations. Malware authors used Trojan horses and bots (general purpose programs that could follow a pre-scripted set of commands or respond to them in real time) to distribute affiliate marketing-supported adware, collecting a few cents or perhaps even a dollar for each computer they infected.

While the individual results were not very impressive, the ability to automatically install adware on hundreds, thousands and tens of thousands of computers at a time allowed the operators of such networks of bots to receive commission checks for thousands of dollars.

As for the companies behind the adware, as they sold more and more advertising, some of them received venture capital funding, which put them in a precarious position of keeping their investors happy. Increased public attention on such operations; action by the FTC; and, more importantly, a bottoming out of revenue in the glutted online advertising market helped reduce the effect of adware.

Other attempts to monetize malware included distributed denial of service (DDoS) attacks for hire, where the operators of bots blackmailed web-based businesses, threatening to disrupt their web sites with blasts of network traffic so high that their web sites would crash, being unable to handle the requests generated by bot-infected networks of computers. The botnet operators, or bot herders, primarily targeted adult and gambling web sites, likely because they expected them to be fellow operators in a shadowy illegal economy. But such businesses are legal in various locations around the globe, and after several high-profile arrests for extortion and blackmail, DDoS-for-hire activities are greatly diminished.

It was commonly suggested that we would not see any more high-profile malware infestations, as such events tend to point a finger back at their creators. However, November 2008 saw the rise of the Conficker [24] worm. Initially infecting through the MS08-067 [25] vulnerability that had been released the previous month, the worm was modified on a near-monthly basis in order to increase its infectiousness. Eventually, multiple means of infecting new computers were added to the Conficker worm, ranging from the aforementioned Windows vulnerability to brute-force password hacking to spreading via USB flash drives. Regardless of how it initially entered a network, once it was present it spread rapidly and, using multiple infection vectors, increased the likelihood of reinfection. This was particularly problematic in organizations that did not have centralized management of their computers, as all it took was one unpatched computer or plugging in one infected USB flash drive to begin the infection cycle anew.

So what exactly did the worm do? Aside from making itself resident on computers and disabling security-related programs and services on them, it also blocked access to various security vendor's programs and web sites and attempted to connect to various web sites to verify Internet connectivity as well as to check the current date. Some variants used a private peer-to-peer network to check for instructions and updates from the operators of the worm. Perhaps the most troubling aspect of the worm was that it created a random list of web site addresses. The worm would then attempt to connect to one of these addresses to check for new instructions and further updates. This action was particularly troubling, since an update could cause the worm to go from beyond merely spreading to causing much more active forms of damage. Initial versions of the randomly generated list contained several hundred addresses, but by March 2009 a new version of the worm appeared that would begin generating a list of 50,000 web site addresses a day, beginning in April 2009.

A coalition of security and Internet companies formed the Conficker Working Group [26] to block and prevent any of these new domains from being registered or used, as well as to collect and share intelligence about the worm. According to the Conficker Working Group, anywhere from three to fifteen million computers may have been infected by the Conficker worm.

One major difference in two decades of anti-malware software is that today's programs are persistently connected to the Internet, which allows them to not only download updates but upload telemetry as well, such as the number of files scanned, threats found, the web sites they were downloaded from (if applicable) and so forth. ESET calls its telemetry service ThreatSense.Net, and other anti-malware vendors have similar services in their products. Through 2009, the Conficker worm was endemic, at times accounting for as much as about 20% of the total reports received worldwide. On a country-by-country basis, the infection rate was higher: For example, Conficker infections in Ukraine approached 30%.

So, what happened with the Conficker worm? Not much, actually. April 1, 2009, came and went, and with the worm unable to download any updates due to the efforts of the Conficker Working Group, no hard disks drives were erased; no mass mailings of spam or copies of the worm were sent; and no DDoS, espionage or other attacks were seen. With all of the attention the worm had received in the news and the high level of monitoring by security and network companies and researchers, it seems likely that Conficker's authors found themselves in the same position as jewel thieves who have stolen gems that are too hot to fence. Any attempt to begin monetizing their botnet would likely cause them unwanted scrutiny, up to and including arrests, even if they happen to be in countries with no cybercrime laws or lax enforcement of them, such as China, Russia and Ukraine.

Today, the Conficker worm is still out there and accounts for just under 10% of the threats we see on a monthly basis, which places it in the top twenty families, and sometimes the top ten, of malware reported to ESET. Conficker continues to spread and continues to build a list of 50,000 different domains to check on a daily basis. But it seems to be abandoned by its creators. Headless and without purpose, the Conficker worm will likely live on for more years, though, as long as computers capable of hosting it are connected to the Internet. As for security companies, most that I have spoken to claim a rise in sales of about 10–15% due the Conficker virus, assuming they can attribute it. Not bad, but nothing like the 400% increase in sales seen during Michelangelo's heyday.

This Way to the Egress

In the past twenty years, we've seen personal computers go from being the tools of the largest enterprises and agencies (as well as a few dedicated hobbyists) to the point of ubiquity, where it is not so much a question of whether there is a computer in the home but how many there are. Likewise, we have seen malicious software evolve from pranks and occasional acts of vandalism to the tools of choice for blackmail, espionage, theft and, perhaps, cyber war.

Anti-malware companies, such as ESET, will continue to fight the good fight, in conjunction with the developers of operating systems, networks and, yes, our competitors. Even the biggest companies are not able to make a go of it alone, and it might be a single security researcher toiling in obscurity who provides the necessary intelligence to take down the authors of the next Michelangelo virus or Conficker worm.

Criminal and illegal acts have existed since the dawn of human history, long before the birth of the computer and pervasive network connections. Technology just makes them easier to engage with less chance of being caught. Even when the bad actors are caught, the patchwork of cybercrime laws and treaties makes it difficult to charge them with crimes and successfully prosecute them. While I do not think we should just give up trying to prosecute criminals; I think anti-malware vendors, governments and, most importantly, citizens need to work together toward better laws that make it easier to prosecute cybercriminals. This becomes even more important as an increasing number of criminal activities move from the physical to the cyber world.

Acknowledgements

On a final note, this white paper was written over the course of several weeks, often late at night and from my own, somewhat-hazy recollection of events long ago. There were a number of people who were helpful in the creation of this white paper either by sharing their recollections with me or by being actual participants, and I would like to thank them for their assistance:

Michael Albers, Linnaea Avenell, Jared Bergeron, John Bitow, Richard Bitow, Vesselin Bontchev, Larry Bridwell, David M. Chess, Spencer B. Clark, Ileah Crawford, David Debenham, Jim Dennis, Brian Denton, Chris Dunn, Michael Durkin, Joseph J. Esposito, Gadi Evron, Paul Ferguson, Mark P. Fister, W. Bullitt Fitzhugh, Richard Ford, Garry and Monique Gayles, Michael Gilardino, Sarah Gordon, Ross M. Greenberg, Richard Gugeler, Wallace Hale, Harold J. Highland, Norman Hirsch, Zach Hornbaker, Robert V. Jacobson, Eric Johnson, Fred Kolbrener, Victor Kouznetsov, C. Jimmy Kuo, Mikael Larsson, Ola Larsson, Kelly D. Lucas, Jim Lynch, Michael and Sandra Mansfield, Dan McCammon, William S. McKiernan, Kevin McPherson, Christopher T. Morgan, Thomas Nofsinger, Pat O'Leary, David and Margaret Perry, Keith Petersen, A. Padgett Peterson, Paul K. Peterson, Chad F. Routh, Timo Salmi, Chris and Martha Schram, Morgan R. and Melissa Schweers, Fridrik Skulason, Robert Slade, Alan Solomon, Eugene Spafford, Heather Stern, Brian Thomas, Nigel Thompson, Roger Thompson, Lorraine Walker, Kenneth R. van Wyk, Randal Vaughn, J.J. Webb, Dennis Yelle, John Young, James A. Zoromski and Righard Zwienenberg.

Despite my best efforts, it is likely I have made mistakes in some part of this paper or otherwise gotten my facts wrong. If that is the case, I would appreciate hearing from you so that I might correct it in a future edition.

Aryeh Goretsky

Sources

[1] Simondi, Tom. "1989 Datacrime" Computer Knowledge, http://www.cknow.com/cms/vtutor/1989-datacrime.html

[2] Wikipedia contributors, "Morris worm," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Morris_worm

[3] Elmer-Dewitt, Philip; Munro, Ross H. Technology 1988. Technology: You must be punished. TIME Magazine. September 26. (retrieved from http://www.time.com/time/magazine/article/0,9171,968490,00.html)

[4] ESET Virus Lab, "Threat Encyclopedia: Brain," ESET, http://www.eset.com/threat-center/encyclopedia/threats/brain

[5] Wikipedia contributors, "Power-on self-test," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Power-on_self-test

[6] Wikipedia contributors, "Booting," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Booting

[7] Wikipedia contributors, "IO.SYS," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=IO.SYS

[8] Wikipedia contributors, "MSDOS.SYS," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=MSDOS.SYS

[9] Wikipedia contributors, "COMMAND.COM," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=COMMAND.COM

[10] Wikipedia contributors, "AIDS (trojan horse)," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=AIDS_(trojan_horse)

[11] Wikipedia contributors, "Ransomware (malware)," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Ransomware_(malware)

[12] Wikipedia contributors, "Terminate and Stay Resident," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Terminate_and_Stay_Resident

[13] Wikipedia contributors, "Borland," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Borland

[14] Wikipedia contributors, "Sidekick," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Sidekick

[15] Wikipedia contributors, "Jerusalem (computer virus)," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Jerusalem_(computer_virus)

[16] ESET Virus Lab, "Threat Encyclopedia: Disk Killer," ESET, http://www.eset.eu/buxus/generate_page.php?page_id=3160

[17] http://www.eset.com/threat-center/encyclopedia/threats/cascade

[18] Wikipedia contributors, "Stoned (computer virus)," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Stoned_(computer_virus)

[19] ESET Virus Lab, "Threat Encyclopedia: Maltese Amoeba," ESET,
http://www.eset.com/threat-center/encyclopedia/threats/malteseamoeba

[20] Wikipedia contributors, "ILOVEYOU," Wikipedia, The Free Encyclopedia,
http://en.wikipedia.org/w/index.php?title=ILOVEYOU

[21] Wikipedia contributors, "Code Red (computer worm)," Wikipedia, The Free Encyclopedia,
http://en.wikipedia.org/w/index.php?title=Code_Red_(computer_worm)

[22] Wikipedia contributors, "Blaster (computer worm)," Wikipedia, The Free Encyclopedia,
http://en.wikipedia.org/w/index.php?title=Blaster_(computer_worm)

[23] Wikipedia contributors, "SQL slammer (computer worm)," Wikipedia, The Free Encyclopedia,
http://en.wikipedia.org/w/index.php?title=SQL_slammer_(computer_worm)

[24] Wikipedia contributors, "Conficker," Wikipedia, The Free Encyclopedia,
http://en.wikipedia.org/w/index.php?title=Conficker

[25] Microsoft, "Microsoft Security Bulletin MS08-67 – Critical; Vulnerability in Server Service Could Allow Remote Code Execution (958644)," Microsoft Security Bulletins, http://www.microsoft.com/technet/security/Bulletin/MS08-067.mspx

[26] "Conficker Working Group Home Page," Conficker Working Group, http://www.confickerworkinggroup.org

www.eset.com

ESETDCBWP20100614