

Changing threats, changing solutions: A history of viruses and antivirus

David Emm, Security Consultant Kaspersky Lab

It is more than 20 years since the first PC virus appeared. Since then, the nature of threats has changed significantly. Today's threats are more complex than ever before. Much of today's malicious code, and this includes a wide array of Trojans, exploits, rootkits, phishing scams, spam and spyware as well as classic viruses and worms, is purpose-built to hijack users' machines to make money illegally. The connectivity provided by the Internet means that attacks can be launched on victim machines very quickly, as widely or selectively as malware authors, and the criminal underground that sponsors them, require. Malicious code may be embedded in e-mail, injected into fake software packs, or placed on web pages for download by a Trojan installed on an infected machine. The scale of the problem, in terms of numbers alone, has also continued to increase. Kaspersky Lab antivirus databases contain 570,500 records¹ and around 3,500 new records are added weekly.

In any field of human activity, the latest generation stands squarely on the shoulders of those who went before, learning from what has been done before, re-applying what has proved successful and also trying to break new ground. This is no less true of those who develop malicious code. Successive waves of malicious code have re-defined the threat landscape.

What's also clear is that security solutions have also had to evolve to match each successive generation of threats. As a result, both the disease and the cure differ greatly from the situation when the virus problem first appeared. But what are the specific factors that have influenced the development of malicious code? And how have security solutions had to evolve to deal with each emerging threat?

The first PC viruses: boot sector viruses

The first PC virus, *Brain*, appeared in 1986. Brain was a boot sector virus. Boot sector viruses work by modifying the first sector on floppy disks. The life-cycle of a boot sector virus is as follows: the virus executes, and loads its code into memory, when a user boots from an infected disk. The disk doesn't have to be a system disk: any disk will do. In most cases, the user doesn't mean to boot from the disk at all. Typically, they simply forget to remove the disk when they shut down the machine and then forgot it's there when they boot up the next day. If the BIOS is configured to boot from floppy disk (and, of course, a growing number of PCs these days do not come with a floppy disk drive) the system detects the disk in drive A and automatically loads whatever code is in the boot sector: in the case of an infected disk, the virus. The user realizes they've tried to boot from floppy disk by mistake when they see the message 'Non system disk or disk error, replace and press any key when ready'. They then remove the disk and continue working, suspecting nothing about what has just happened. What happens next depends on the operating system being used. Boot sector viruses infect at a BIOS level, before the operating system is loaded. So they're operating system independent². However, they use DOS calls to go memory resident and spread to other floppy disks: if the operating system doesn't support DOS, they don't get the chance to load and spread. They're effectively sidelined by any operating system other than DOS, Windows 3.x (which sits on top of DOS) or Windows 9x (which may sometimes use DOS access to floppy disks). The only damage they can do on other operating systems is if the virus is coded to carry out any damage routine at a BIOS level, before the operating system loads. This is true of *Michelangelo*, for example, which overwrites the start of the hard drive as soon as the PC is booted on 6 March ... before the operating system loads.

The writers of boot sector viruses had no need to implement social engineering tricks to spread their creations. On the contrary, very little user interaction was required beyond inadvertently leaving an infected

¹ February 2008.

² Given the limitations outlined above, you could be forgiven for thinking that boot sector viruses have become extinct. And they have ... almost! However, in January 2007 we learned of two infections of Junkie virus, one in Russia, the other in the Netherlands; and in September 2007 laptops infected with Angelina virus were shipped by a Danish vendor.

floppy disk in the drive. At the time, floppy disks were the main means of transferring data from computer to computer and from user to user. So it was almost inevitable that, sooner or later, the user would pass on an infected floppy disk to a friend, colleague or customer and spread the virus.

In the years that followed the appearance of Brain, boot sector viruses were further refined and developed. Whereas Brain infected floppy disks only, most of its successors were designed to infect the hard disk also. In most cases, this meant writing code to the MBR (Master Boot Record). Some, however (notably *Form*), infected the boot sector of the hard disk. And a small number (e.g. *Purcyst*) infected both the MBR *and* the boot sector.

DOS file viruses

Until 1995, boot sector viruses represented around 70% of all infections found in the field³. However, they weren't the only type of virus. This period also saw the emergence of viruses designed to infect DOS executable files, first COM files, then later EXE files. These viruses modified the host file in such a way that the virus code ran automatically when the program was run. There were many different methods used to infect files.

One typical method was to add the virus code to the end of the host file and modify the header of the file so that it loads first the virus code and then the original program instructions. There were several variations on a theme, however. Some viruses inserted their code at the start of the file. A few inserted their code in several locations within the file. And some, called overwriting viruses, replaced the original code entirely: of course, the host program failed to run, so such viruses were not commonly found in the field; their presence was too obvious for them to survive for long. A few viruses chose an alternative infection method. Companion viruses stored their code in a separate 'companion' file. For example, the virus might rename the standard RUNME.EXE file to RUNME.EXD and create a new RUNME.EXE containing the virus code: when the user executed the program, the virus loaded first and then passed control to the original program, so the user wouldn't see anything suspicious. Another alternative was the 'link virus', which spread by manipulating the ways files were accessed under the FAT file system used by DOS. The user would receive an infected file (on floppy disk, for example) and run it. The virus loaded into memory created a (typically hidden) file on the disk: this file contained the virus code. The virus would then modify the FAT to cross-link other files to the disk sector containing the virus code. As a result, whenever the infected file was run, the system would jump first to the virus code and run it.

One reason why file viruses were less common than boot sector viruses is that users didn't often exchange programs, particularly in a business environment. They did exchange floppy disks, but typically to transfer data. That said, there were also file viruses found in the field during these early days. The most successful and fast-spreading of them were designed to go memory resident: these so-called indirect-action file viruses could monitor activity on the system and infect any file the user chose to run. By contrast, direct-action file viruses simply infected a file (or files) when the infected program was run and would then hibernate until the next time an infected file was run: such viruses were much less effective at spreading. One other factor that helped file viruses to spread was the mass distribution of infected media: on a number of occasions this happened through the distribution of infected disks on magazine covers.

The virus landscape of the 1980s

While the overall number of file viruses grew steadily from the late 1980s, the scene was dominated by a small number of very successful viruses. *Jerusalem*, for example, spread across many enterprises, academic institutions and government agencies and on 13 May 1988 (which became known as 'Black Friday') it caused the first major virus epidemic. The *Vienna* virus spawned numerous variants following the publication of its source code. And *Cascade*, notable for being the first encrypted virus, continued to be common well into the 1990s.

³ The term 'in the wild' wasn't used until Joe Wells created The WildList, as a means of tracking real-world infections, in April 1993.

As time went on, some virus authors tried to get the best of both worlds by developing viruses that were combination boot sector viruses and file viruses. *Tequila*, *Junkie* and *Natas* were all successful examples of what became known as multipartite viruses.

At this time, it was almost completely a *virus* problem. There had already been some worms, most notably the *Morris worm* in November 1988: this successfully infected about 6,000 or so vulnerable systems (around 10% of all computers connected to the Internet in 1988). However, at this time, the Internet was used almost exclusively by government and academic institutions. The Internet worm's time had not yet come.

In addition, there were only a small number of Trojans. The term Trojan (short for Trojan Horse) is taken from the wooden horse used by the Greeks to sneak inside the city of Troy and capture it. The first Trojans, which appeared in the late 1980s, masqueraded as innocent programs. Once the unsuspecting user ran the program, the Trojan would deliver its harmful payload. Hence the copy-book definition given by most anti-virus vendors: a non-replicating program that appears to be legitimate but is designed to carry out some harmful action on the victim computer.

The fact that Trojans don't spread by themselves was the key feature that distinguishes them from viruses. Viruses are parasitic, adding their code to an existing host. So they spread from file to file to file: and the longer a user is infected, the further the virus spreads across their machine (and potentially across the network too, if the user is able to access a network). Trojans, by contrast, have no on-board replication mechanism. So at this time, Trojan authors had to find some way of distributing their code manually: upload it to a BBS (Bulletin Board System) in the guise of a useful application, deliberately plant it in a corporate network, or use the postal service to send it to a pre-defined list of victims.

Twelve Tricks, for example, was a hacked version of a hard disk benchmarking program. When installed, the Trojan wrote itself to the MBR of the disk and performed one of its twelve 'tricks', many of which made it look as though the victim had a hardware problem. Unfortunately, there was also a chance that the Trojan would format the track on the hard disk containing the boot sector, or cause gradual corruption of the FAT.

Another example of an early Trojan was the Aids Information Disk. In late 1989, 20,000 floppy disks containing this Trojan were mailed to addresses stolen from PC Business World and the World Health Organization, by a company called PC Cyborg. The disks supposedly contained information about HIV and the author was clearly playing on widespread public concern about the disease. When the user ran the installation program, the Trojan wrote itself to the hard disk, created its own hidden files and directories and modified system files. After the PC had been booted 90 times, the Trojan encrypted the contents of the hard disk, making the data inaccessible. The only accessible file remaining on the disk was a README file: this contained a bill and a PO Box address in Panama for payment. Interestingly, the use of 'program mechanisms', including some that would 'adversely affect other program applications', was announced up-front in a license agreement contained on the floppy disk used to distribute the Trojan.⁴

It was not until much later that Trojans were to come into their own.

Stealth and polymorphism

This period also saw the development of techniques specifically intended to extend the 'window of opportunity', the period of time during which a virus could spread undetected. Stealth techniques were developed as a way of hiding the changes a virus made to the system from users. These included suppressing error messages that might give away the presence of the virus, spoofing file information so that it didn't look as though a file had increased in size and intercepting attempts to read system sectors using a sector editing tool. Stealth techniques later became even more sophisticated by filtering out virus code from infected files whenever an anti-virus scanner tried to read the files. This development made memory

⁴ Dr Joseph Popp, the alleged author of the Trojan, was later extradited to the UK. However, he was deemed unfit to stand trial following his behavior in court (an Italian court later found him guilty *in absentia*).

scanning, before looking at the disk, essential (encrypting virus code in memory was so complex that few viruses attempted it).

Polymorphism was another technique designed to help code evade anti-virus scanners. The term 'polymorphic' comes from the Greek for 'many forms'. The idea was to variably encrypt the virus code with each infection, to change the form of the virus so that there was no constant sequence of bytes for an anti-virus program to search for. The first proof-of-concept polymorphic viruses were created in 1990: but it was not until April 1991, with the appearance of the *Tequila* virus, that polymorphic code began to appear in the field. This development made it impossible for an anti-virus program to rely solely on signatures. Other techniques, including emulation of the virus code, or the use of mathematical algorithms to 'see through' the code, became essential.

Both stealth and polymorphism were subsequently refined and re-applied by subsequent generations of virus authors.

Early antivirus solutions

What about the antivirus solutions during this period? The virus problem started as a trickle rather than a flood: new viruses appeared slowly, one at a time. Indeed, for a few years, there were some who considered the appearance of viruses to be just a myth. The first antivirus programs were utilities designed to find and remove specific individual viruses or, as the number of malicious programs slowly increased, a handful of viruses.

Then, as the trickle started to turn into a stream during 1989, anti-virus 'toolkits' were released. The core of these programs tended to be an on-demand scanner designed to search the disk for the dozen or so viruses in existence at the time. Some included cleaning capability, to remove the infected code. Some also included a checksummer, designed to 'fingerprint' files on a clean disk so that any subsequent change became noticeable, even (in theory) if they were modified by a brand new virus.

To start with, antivirus programs were on-demand only. The number of new viruses grew slowly at first. Moreover, their rate of spread was, by today's standards, slow. The key, therefore, was to be able to carry out regular scans of the system and to have the means to remove anything that might find its way into an organisation or a home computer. For this purpose, the clean system disk was an invaluable tool: by booting clean first, you could ensure that there was no virus in memory to interfere with scan and clean operations⁵. In many cases, companies didn't even install anti-virus programs onto individual machines, at least not initially (this tended to change once a company had been hit by a virus). Typically the administrator would conduct regular sweeps of the system and try to screen incoming floppy disks on a stand-alone machine before they were used.

The normal update cycle for those using anti-virus programs was quarterly. Although some anti-virus vendors offered monthly updates, they weren't generally considered necessary and those who needed the extra level of security paid a premium for the greater update frequency. Updates were delivered, of course, on physical media ... floppy disks.

Increasing threats, evolving detection methods

However, by the end of 1990, the number of viruses was approaching 300. In the face of this threat antivirus vendors began to implement real-time protection. This meant developing TSR (Terminate and Stay Resident) programs which, like the successful viruses of the period, monitored the system, intercepting disk and file access to check for the presence of known viruses.

⁵ Although a small number of viruses, like Exebug and Purcyst, modified CMOS settings to try and prevent the machine from clean booting from floppy disk.

In addition, they also started to look for ways of detecting viruses proactively, without the need for a specific signature, by applying heuristic analysis to the problem. Anti-virus researchers had growing experience of the techniques used by malicious programs and they used this experience to create a list of suspicious characteristics, each with a specific score. The scanner would analyse the code for these characteristics: if the score passed a pre-defined threshold, the file was identified as a probable virus.

The growing threat from polymorphic viruses forced anti-virus developers to supplement signature-based analysis with other methods that would allow a scanner to 'see through' the layer(s) of encryption. These included the use of reduced masks, cryptanalysis, statistical analysis and emulation techniques. Code emulation was also used to provide enhanced heuristic detection, by making dynamic code analysis possible.

Early behaviour blockers

One alternative solution to the sharp rise in numbers was behavioural analysis. Whereas traditional anti-virus scanners held signatures of malicious code in a database and cross-checked code against this database during the scan process, behaviour blockers used a program's behaviour to decide if it was malicious: if a program did something that fell beyond a range of pre-defined acceptable actions, it was blocked.

The chief benefit of a behavior blocker, according to its proponents, was that it's able to distinguish between 'good' or 'bad' programs without the need for a professional virus researcher to analyze the code. Since there's no need for ongoing analysis of each specific threat, there's also no need to keep updating virus signatures. So users are protected against new threats in advance, without the need for traditional anti-virus updates.

The drawback, of course, lies in the grey area between actions that are clearly malicious and those that are legitimate. What's malicious in a program specifically designed to cause damage may be good in a legitimate program. For example, the low-level disk writes carried out by a virus, worm or Trojan, perhaps to erase data from a hard disk, are also used legitimately by the operating system. And how is a behavior blocker deployed on a file-server to know whether a modification to (or deletion of) a document is being done legitimately by a user or is the result of a malicious program's activity? After all, a virus or worm is simply a program that copies itself. Beyond this, it may do what any other normal program does. For this reason, mainstream anti-virus programs continued to be based largely on the detection of known malicious programs.

However, behavioural analysis did not disappear. Indeed several security solutions today combine the use of behavioural analysis with other methods of finding, blocking and removing malicious code.

The growing number of threats was just one problem. The development of stealth and polymorphism outlined above highlight the fact that viruses were continually evolving. Although there were many poorly-written, copy-cat viruses, the most skilled virus writers could write very effective code and kept pushing the boundaries in order to get one step ahead of anti-virus researchers. Among them was the virus writer known as 'Dark Avenger', responsible, among other things, for developing both 'fast infectors' that could infect on any type of file access and for the idea of gradually corrupting data (the *Dark Avenger* and *Nomenklatura* viruses). Dark Avenger was responsible for the *Self-Mutating Engine* (known as MtE), a bolt-on module capable of converting a regular virus into a polymorphic virus⁶. His final offering was a virus called *Commander Bomber*, in which the code was distributed across the host program, making it very difficult to scan for at any speed.

Virus exchange and virus construction kits

There were several other developments that made life difficult for anti-virus developers during the first half of the 1990s. The first was the virus exchange (VX) bulletin board. Bulletin boards had been used from the outset as a distribution mechanism for viruses. Until the Internet became widely used, bulletin boards were

⁶ This started a trend. In the years that followed other polymorphic engines appeared, including the 'Trident Polymorphic Engine' and the 'Nuke Encryption Device'.

commonly used to download programs: so infecting files and making posting them for download was an effective way to seed an infection. VX was a further development. The idea was simple: the VX bulletin board hosted a collection of viruses, but you could only access the collection if you uploaded a virus. Of course, this not only encouraged virus writing, it also caused the more widespread dissemination of viruses that might otherwise not have spread. In a parallel development at this time, a number of people, both in Europe and the US, offered virus collections for sale.

Another worrying trend was the emergence of virus construction kits. As the name suggests, they were designed to provide 'build-your-own-virus' capability to those who did not have the knowledge or technical skills to write their own code. Such kits offered a user interface which made it possible to select the characteristics of the virus from a list. Unfortunately for the authors of these kits, the viruses created using such kits normally had a characteristic 'fingerprint' that allowed anti-virus researchers to detect them all using a single signature.

Macro viruses

A major shift in development took place in July 1995, with the appearance of the first macro virus, called *Concept*. Macro viruses were to dominate the threat landscape in the four years that followed.

The author of *Concept* exploited the fact that users exchanged data far more frequently than programs. He used WordBasic⁷ instructions to modify the NORMAL.DOT template and so incorporate the virus code, in the form of several auto macros, into each document subsequently created by the user. As a result, every user who received the infected document became a victim. This simple development, long predicted and feared by anti-virus researchers, had far-reaching implications.

First, the focus of the virus writing community shifted from executable code (program files and disk sectors) to data. Previously, viruses had been generally written in assembly code which requires a certain amount of knowledge. Macro viruses, by contrast, written in WordBasic and later VBA (Visual Basic for Applications), were much easier to create. They were also easier to copy: virus macros were generally clearly visible and could be copied, modified and re-applied easily. Suddenly the field of virus writing was open to a far wider group. As a result, the number of viruses increased significantly: from around 6,000 in June 1995, to 7,500 by the end of this year, to more than 25,000 in December 1998.

Second, macro viruses were the first viruses to (deliberately) infect data files. Since data files were exchanged far more often than programs, this provided macro viruses with a more effective replication channel than previous viruses. This problem was compounded by the emergence of e-mail as a means of communication and data exchange. The increased use of e-mail by users, rather than by geeks, the ease with which files could be attached to an e-mail message, and the emergence of widespread access to the Internet all provided fertile ground for macro viruses to spread.

Third, macro viruses were neither platform-specific, nor OS-specific. They were application-based. Since there were versions of Word for Windows 3.x, Windows 95, Windows NT and Macintosh, this made all these systems susceptible to attack, or at least made them an effective carrier. On top of this, the problem soon extended beyond just Word. As VBA was applied across the Office suite (Word, Excel, PowerPoint, Access and Project) these applications all became targets for macro viruses: there were even cross-application macro viruses that targeted all Office applications!

As mentioned above, each generation stands on the shoulders of those who went before. Macro virus authors took earlier technical developments, like stealth and polymorphism, and re-applied them in this new context.

⁷ The macro writing language built into early versions of Word for Windows.

Mail server and gateway solutions

As the threat landscape changed, so too did the solutions designed to protect businesses from attack. Before the appearance of macro viruses, the centre of gravity for virus scanning was the desktop and, to a lesser extent, the file server. Macro viruses began a process in which the net widened to incorporate mail servers and the Internet gateway. The reason is clear. As email became the chief mechanism by which viruses spread, scanning of email became an efficient way to block the threat before it reached the desktop. Of course, the need for desktop and file server protection didn't evaporate. These remained an essential weapon in the fight against viruses. But they had to be integrated into a wider, multi-layered 'defence in depth' solution.

Anti-virus vendors also started to further develop their proactive detection capabilities, specifically by implementing generic detection. Generic detection refers to the detection and removal of multiple threats using a single virus signature. The starting-point for generic detection is that successful threats are often copied by others, or further refined by the original author(s). The result is a spate of viruses, worms or Trojans, each one distinct but belonging to the same family. In many cases, the number of variants can run into tens, or even hundreds. Generic detection provided a further method of detecting new, unknown threats without the need for a brand new signature.

The birth of spam

The growing use of e-mail as a key business tool saw the emergence of another business problem, junk e-mail, Unsolicited Bulk E-mail (UCE) or spam, as it is variously known. As more and more businesses came to rely on e-mail, those using it became an attractive target for those looking for new ways to advertise goods and services. Such advertising covered a broad range, from products and services that were legitimate, to those that were obscene, illegal or otherwise unwanted.

The emergence and growth of spam brought with it several problems. These included wasted bandwidth, wasted time as staff read non-work-related e-mail and potential HR and legal issues concerning obscene, sexist, racist or other undesirable content. There could be grey areas, of course: clearly one person's spam may be another's valuable and well-received information.

Not only did this period see the development of content filtering, primarily deployed at the Internet gateway, for filtering out spam and other unwanted content. It also saw collaboration with anti-virus vendors, who were focusing increasingly on filtering malicious code at the mail server and Internet gateway.

Mass mailing of malicious code

The appearance of the *Melissa* virus in March 1999 marked the beginning of another quantum leap forward in terms of the virus threat. On the face of it, Melissa seemed to be just another macro virus. However, unlike earlier macro viruses, which waited for the user to send the infected data, Melissa 'hijacked' the e-mail system to distribute itself proactively.⁸ All that was required of the user was to double-click on the infected e-mail attachment. After this, the virus harvested e-mail addresses from the Outlook address book and sent itself directly to the contacts listed in it. This 'mass-mailing' allowed Melissa to spread further and faster than any previous macro virus. Worse still, Melissa represented a threat to the stability of the e-mail infrastructure itself as a result of the sheer volume of e-mail messages created by the virus. Following the initial spread of Melissa (it was posted to one of the sex newsgroups) corporate e-mail systems quickly became clogged with e-mail and many simply bowed under the pressure. It's hardly surprising that Melissa set a trend. For many years to come, nearly all of major viruses and worms to threaten corporate and home users alike included mass-mailing capability.

⁸ Melissa wasn't the first virus that tried to use e-mail to 'mass mail' itself. In 1998, the RedTeam virus included code to send itself via the Internet. However, this virus targeted Eudora mail only and failed to spread in significant numbers.

Melissa brought about a significant change in the nature of the virus threat. Viruses no longer had to wait for an unsuspecting user to distribute the infected file. By hijacking the mail system itself, viruses were able to harness a very effective replication mechanism and bring about global epidemics in days or even hours.

In one sense, Melissa faced in two directions simultaneously. On the one hand, it looked backwards to the macro virus era and was one of the last (and the biggest) macro virus outbreak. On the other, it looked forward to a period that was to be dominated by the worm, in particular the e-mail worm.

Worms: email worms

A worm is a stand-alone replicating program, that is, a virus with no need for a host. Instead of spreading from file to file across the same PC, the worm seeks to use network (or Internet) connectivity to spread itself. There may, therefore, be only a single instance of the worm on any given PC. The re-emergence of the worm took place hand in hand with the decline in macro viruses (as a result of changes Microsoft made to the handling of macros) and the use, once again, of executable files to generate outbreaks.

E-mail worms came in different flavours. They spread as executable files (like Happy99, the first worm of the modern era), as script files attached to e-mail messages (Visual Basic Script or Java Script), or even as script embedded within HTML messages. What they had in common was the use of e-mail to spread, typically using social engineering techniques to lure naïve users into running malicious code.

Social engineering refers to a non-technical breach of security that relies heavily on human interaction, tricking users into breaking normal security measures. In the context of viruses and worms, it typically meant attaching a virus or worm to a seemingly innocent e-mail message. One of the earliest examples was LoveLetter, with its 'ILOVEYOU' subject line and message text reading, 'Kindly check the attached LOVELETTER coming from me'. Or (like LoveLetter, SirCam, Tanatos, Netsky and many others), it could include an attachment with a double extension, to conceal the true nature of the infected attachment: by default, Windows does not display the second (real) extension. Or it could be an e-mail constructed to look like something innocent, or even positively beneficial!

Internet worms

E-mail worms were not the only type of worm. 2001 saw the return of the Internet worm. CodeRed, which appeared in July 2001, was a 'fileless' worm. In a complete departure from earlier virus practice, the worm's code existed in memory only: it made no attempt to infect files on an infected machine. CodeRed used a vulnerability in Microsoft IIS server (MS01-033 'Uncheck Buffer in Index Server ISAPI Extension Could Enable Web Server Compromise') to attack Windows 2000 servers. It spread via TCP/IP transmissions on port 80, launching itself in memory via a buffer overflow and then sending itself in the same way to other vulnerable servers. CodeRed ripped through the Internet in a matter of hours, considerably faster than anything previously seen. This was the first time since the Morris worm to exploit a vulnerability to spread. The success of CodeRed, however, made sure it wouldn't be an isolated incident. Indeed, just a few months later, in September 2001, the Nimda virus exploited a vulnerability in Internet Explorer (MS01-020, 'Incorrect MIME header can cause Outlook to execute e-mail attachment') to produce another global epidemic.

Nimda infected files but, unlike earlier mass-mailing threats, didn't rely on the user to click on an infected EXE file attached to an e-mail message. Instead, it made use of the browser vulnerability to launch itself automatically on vulnerable systems. This was a six month old vulnerability, but a great many systems remained un-patched and vulnerable to attack and the use of this vulnerability helped Nimda to infect systems all over the globe in a few hours.

Exploits

In the years following CodeRed and Nimda, the use of system exploits became commonplace, as malware authors tapped into the 'helping hand' provided by vulnerabilities in common applications and operating systems. Such attack methods had previously been associated with the activities of hackers, rather than virus writers. The combination of 'traditional' virus writing techniques with hacker attacks marked yet another 'leap forward' in malware development.

Some threats have avoided the use of 'traditional' virus techniques altogether. Lovesan, Welchia and Sasser, for example, were Internet worms pure and simple. There was no mass-mailing component and no requirement for the victim to run an infected program. Instead, these threats spread directly across the Internet, from machine to machine, using various exploits. Lovesan exploited the MS03-026 vulnerability ('Buffer Overrun In RPC Interface Could Allow Code Execution'). Welchia exploited the same vulnerability, plus MS03-007 ('Unchecked Buffer In Windows Component Could Cause Server Compromise'). Sasser exploited the MS04-011 vulnerability (a buffer overflow in the Windows LSASS.EXE service).

Nimda set another trend by combining the use of an exploit with other attack methods. As well as mass-mailing itself, it also appended viral exploit code (in the form of infected JavaScript) to HTML files. If the infected machine were a server, a user became infected across the web when they accessed the infected pages. Nimda went even further in its efforts to spread across the corporate network by scanning the network for accessible resources and dropping copies of itself there, to be run by unsuspecting users. On infected machines, the virus also converted the local drive(s) to open shares, providing remote access to anyone with malicious intent. For good measure, Nimda also used the MS00-078 exploit ['Web Server Folder Traversal'] in Microsoft IIS server to infect vulnerable servers by downloading a copy of itself from already infected machines on the network. Nimda's multi-faceted attack strategy led many to refer to it as a 'compound' threat.

Many 'successful' threats that followed (successful from the author's perspective, that is) made use of multiple attack mechanisms and used vulnerabilities to bypass the user and launch code automatically, dramatically reducing the 'lead time' between the appearance of a new threat and it reaching epidemic proportions. Threats spread quicker than ever before, achieving worldwide distribution in hours, riding on the back of business-critical e-mail infrastructure and exploiting the increasing number of vulnerabilities as a springboard into the enterprise.

Responses to increased propagation speed

The spread of new threats at 'Internet speed', and the growing number of global epidemics, placed a greater emphasis than ever before on the speed at which anti-virus vendors responded to new threats. In the 'good old days', quarterly updates were enough for most customers. Later, monthly updates became standard. Then, in the wake of e-mail and Internet worms, most anti-virus vendors switched to weekly signature updates. Now, several of them began to offer daily updates.⁹ Speed of response to new threats also began to feature in independent anti-virus tests.¹⁰

In addition, as a response to the social engineering techniques employed by malware authors, many enterprises began to routinely block certain file types at the Internet gateway, to prevent EXE, PIF, SCR and other attachments reaching their users.

Nevertheless, many were still concerned about the potential gap between the appearance of a new exploit or threat and the means to block it, during which a new threat could spread unchecked. Some questioned the ability of traditional signature-based anti-virus solutions to deal with the growing complexity of malicious code.¹¹ Indeed leading anti-virus vendors began to broaden the scope of the protection they offered.

⁹ Today, one anti-virus vendor, Kaspersky Lab, provides hourly signature updates.

¹⁰ This has been a particular focus of [AV-Test GmbH](#).

¹¹ In 2001, Gartner commented, in its Signature-Based Virus Detection at the Desktop is Dying: 'The signature-based desktop antivirus software used by most enterprises provides marginal value today, and that value is steadily decreasing. Gartner believes that enterprises should begin to augment and eventually replace signature-based techniques with more-robust approaches. Enterprises that don't will be swamped by malicious software riding the coming wave of Web services'.

Personal firewalls

One way of doing this was adding personal firewall capability. Personal firewalls monitor and block unwanted traffic. A personal firewall, as the name suggests (and in contrast to a traditional gateway firewall), is installed on a desktop or server. It works like a desktop 'traffic warden', inspecting inbound *and* outbound traffic on the computer and allowing or blocking connections based on pre-defined policies. There are typically two aspects to personal firewalls. On the one hand, they offer application filtering. That is, they allow rules to be set for commonly used applications like web browsers, ICQ, instant messaging programs and others. Personal firewalls also provide packet filtering: they analyze data transfers (headings, protocol used, ports, IP addresses, etc.) and filter packets based on the policies set.

Intrusion prevention systems

Some security vendors have supplemented 'traditional' technologies with intrusion prevention systems [IPS]. *Host-based* IPS, designed to protect desktops and servers, typically employ behavioral analysis to detect malicious code. They do this by monitoring all calls made to the system and matching them against policies based on 'normal' behavior. Such policies can be quite granular, since behavior may be applied to specific applications. In this way, activity such as opening ports on the system, port scanning, attempts to escalate privileges on the system and injection of code into running processes can be blocked as abnormal behavior. Some IPS systems supplement behavioral analysis using signatures of known hostile code.

Network-based IPS, deployed inline on the network, filter packets for malicious code, looking for abnormal bandwidth usage or for non-standard traffic (like malformed packets). Network-based IPS is particularly useful for picking up Denial-of-Service [DoS] attacks, or the traffic generated by network-based worms.

Behavioural analysis

Several vendors have also blended existing technologies with behavioral analysis, providing real-time monitoring of application activity, blocking of any suspicious actions and even 'roll-back' capability to undo changes that a malicious program has made to the system.

The intended aim of all these 'new' technologies is to defend against attacks designed to steal confidential information, network worms and malicious code that seeks to turn the victim's machine into a 'zombie' for use in spam attacks.

The up-side of both technologies is that they offer generic protection from unknown malicious code attacks, rather than relying on signatures of known threats. The potential down-side is the risk of false alarms. To try and minimize this risk, most of them include an alert-only 'learning mode' that allows the product to build up a picture of what 'normal' behavior looks like in a specific environment. However, they need to be carefully tuned, so there's a much bigger administrative overhead than with traditional anti-virus protection. In the case of IPS, they also require updates (albeit not the weekly or daily signatures required by 'traditional' anti-virus technology) in order to detect new attack methods.

From cyber vandalism to cybercrime

Since 2003 there has been a decline in the number of global epidemics that has reflected a shift in motivation on the part of malware authors. Until a few years ago, viruses and other malicious programs tended to be isolated acts of computer vandalism, anti-social self-expression using hi-tech means. Most viruses confined

themselves to infecting other disks or programs. And 'damage' was largely defined in terms of loss of data as a virus erased or (less often) corrupted data stored on affected disks.

Over the course of the last few years this has changed. Today we're faced with 'crimeware', malicious code created for the purpose of making money illegally. The criminal underground has clearly realized the potential for making money from malicious code in a 'wired' world and much of today's threats are written 'to order'.

We've seen a clear shift in tactics from the writers of malicious code. The decline in the number of global epidemics signals a move away from the use of mass attacks on victims worldwide. From their peak in 2003, the number of global epidemics has fallen steadily. This isn't to say that there aren't any epidemics: it's just that they aren't *global*. Rather, attacks are becoming more targeted.

This is partly because law enforcement agencies across the world have developed far more expertise than ever before in tracking down the perpetrators of e-crime. It's also partly because anti-virus researchers have now had many years practice in dealing with large-scale epidemics. Fast response to new threats, in the form of virus definitions, is just the visible tip of the iceberg here. Anti-virus research teams worldwide have developed 'early warning antennae' giving them early visibility into malicious activity on the Internet. And when an attack occurs, the servers used to gather confidential data harvested from victim machines can be tracked and closed down, mitigating the effects of an attack.

There is a third reason, however, intrinsic to the motives of the criminal underground. Since much 'crimeware' is designed to steal confidential data from victim machines, later used to make money illegally, it follows that the harvested data has to be processed and used. Where millions of victim machines are involved, not only does this make detection more likely, it's also a huge logistical operation. So for this reason too, it makes more sense for malicious code authors to focus their attacks.

Typically, this means targeting machines one thousand at a time in small-scale, low-key 'hit and run' operations. Or it may mean tailoring a piece of code for an attack on a single victim, or a small number of victims.

Such attacks are often carried out using Trojans. In the last few years, we have seen a massive rise in Trojans numbers: they have now become the 'weapon of choice' for authors of malicious code. Of course, Trojans come in many different flavours, each purpose-built to carry out a specific function on the victim machine. They include, Backdoor Trojans, password stealing Trojans, Trojan-Droppers, Trojan-Downloaders and Trojan-Proxies.

They can be used to harvest confidential information (username, password, PIN, etc.), for computer fraud. Or they can be 'conscripted' into a 'zombie army' to launch a DDoS (Distributed-Denial-of-Service) attack on a victim organization. These have been used to extort money from organizations: a 'demonstration' DDoS attack offers the victim a 'taster' of what will happen if they don't pay up. Alternatively, victim machines can become proxies for the distribution of spam e-mail. There has also been a growth in the number of 'ransomware' worms or Trojans, used to try and extort money from individual users. These programs encrypt the user's data and create a 'readme' file that asks the user to transfer money to the author of the program using one of the many e-payment services.

Often, victim machines are combined into networks, using IRC channels or web sites where the author has placed additional functionality. Typically, such 'bot networks' are controlled through a single control-and-command server. This means that they can be taken down once its location becomes known. However, recently we've seen more complex botnets that use a P2P [peer-to-peer] model. The most celebrated of these, Zhelatin [aka the 'Storm Worm'], appeared in January 2007 and has continued to build since then. This model has also been adopted by the MayDay backdoor, which first appeared in September 2007. The persistence of Zhelatin lies in the fact that it is distributed and so there is no central controlling server that can be taken offline to terminate the botnet.

The trend away from global epidemics and towards low-key, localized attacks has gone hand in hand with a further significant change: a relative decline in the use of mass-mailing to distribute malicious code. Until a few years ago, most epidemics involved worms that hijacked the mail system to distribute themselves proactively, harvesting additional contacts from infected machines as they spread. This was the method used by worms like LoveLetter, Klez, Tanatos (Bugbear), Sobig, Mimail, Sober and Mydoom to cause global

outbreaks. Now, increasing numbers of malicious programs are being deliberately spammed to victim machines. This allows the author(s) to control the distribution of their code to a targeted PC population, rather than letting it spread at will.

For the same reason, the malware 'bundle' dropped onto victim machines now often includes a Trojan Downloader. As the name suggests, these Trojans are designed to download malicious code from specified web sites. They are used not only to control the spread of malicious code, but also to automatically update it across the Internet. They are also used increasingly to install non-viral 'spyware' or 'pornware' programs without the knowledge or consent of the user.

Phishing attacks

The use of malicious code is not the only method used by cyber criminals to gather personal data that can be used to make money illegally. Phishing (a conscious misspelling of the word 'fishing') is a specific form of cyber crime. It involves tricking computer users into disclosing their personal details (username, password, PIN number or any other access information) and then using these details to obtain money under false pretences. It's fraud: data theft, followed by theft of money. Phishers rely heavily on social engineering. They create an almost 100% perfect replica of a chosen financial institution's web site. They then spam out an e-mail that imitates a genuine piece of correspondence from the real financial institution. Phishers typically use legitimate logos, good business style and even make reference to real names from the financial institution's senior management. They also spoof the header of the e-mail to make it look like it has come from the legitimate bank. In general, these letters inform customers that the bank has changed its IT structure and is asking all customers to re-confirm their user information. Occasionally, the letters cite network failures, or even hacker attacks, as reasons for requiring customers to re-confirm their personal data.

The fake e-mail messages distributed by phishers have one thing in common: they're the bait used to try and lure the customer into clicking on a link provided in the letter. If the bait is taken, the luckless 'fish' stands in serious danger of divulging confidential information that will give the criminal access to his or her bank account. The link takes the user directly to an imitation site that mimics the real bank's web site very closely. This site contains a form that the user is told they must complete: and in doing so, they hand over all the information the criminal needs to access their online account and steal their money.

The 'bad guys' are playing for high stakes. So much so that they are reluctant to give up the victim machines under their control. It has now become common for malware authors to sabotage security software, by terminating active processes, or deleting code or blocking anti-virus updates. Some also remove 'competitor' malware. One Trojan, Backdoor.Win32.Agent.uu (aka 'SpamThru') even used a pirated copy of one anti-virus program¹² to find and remove other malware on victim machines.

Sophisticated stealth

In addition, the use of rootkits to mask the presence of malicious code and 'spyware' has increased during the last 12 months or so. The term rootkit is borrowed from the Unix world, where it was used to describe tools used to maintain 'root' access while remaining invisible to the system administrator. Today it's used to refer to stealth techniques employed by malware authors to hide the changes they have made to a victim's machine. Typically, the malware author obtains access to the system by cracking a password or exploiting an application vulnerability and then uses this as 'leverage' to gain other system information until he achieves administrator access to the machine. Rootkits are often used to hide the presence of a Trojan, by concealing registry edits, the Trojan's process(es) and other system activity.

¹² The downloaded anti-virus program was Kaspersky® Anti-Virus.

Mobile malware

Until now, the main focus of malware authors has been desktops and laptops. However, since the appearance of Cabir in June 2004, there has been a steady stream of malicious code specifically aimed at mobile devices.

The use of mobile devices within the corporate world continues to grow and with it the use of wireless technologies of one sort or another. These devices are quite sophisticated: they run IP services, provide access to the World Wide Web and offer network connectivity. In fact, there's little you can do with a laptop that you can't do with a handheld computer.

Therein lies the problem. Enterprises operate today in an 'open space', with employees connected, and therefore open to attack, wherever they work: in the work place, at home, or on the road. And mobile devices are intrinsically less secure, operating outside the reach of traditional network security. And as they start to carry more and more valuable corporate data, wireless devices and wireless networks become a more attractive target for the writers of malicious code. The history of software development clearly shows that time and time again ease of access has been delivered ahead of security. And since mobile devices live outside traditional network security, they could easily become the weakest link in the corporate security system.

The first worm for mobile phones, Cabir, appeared in June 2004. Since then Cabir has spread to more than 40 countries across the globe. Cabir spreads using Bluetooth. This is the most common method for wireless transmission of data, so it's no surprise that it has become the chosen means of infection for many virus writers. Significant numbers of Bluetooth-enabled devices are left in discoverable mode: open to infection and open to hackers.

In a very short period of time, we have seen viruses, worms and Trojans for mobile devices; that is, the array of threats that took twenty years to develop on PCs

Currently, there are around ten new mobile threats per week. Many are fairly basic, but it's clear that malware authors are aware of the long-term potential for using mobile devices for making money illegally. In April 2006, the first Trojan Spy for Symbian OS appeared: Flexispy is a commercial Trojan that takes over control of smartphones and sends call information and SMS data to the author or 'master' of the Trojan. It soon became clear to us that its author was selling his creation for \$50. And there has been similar malware for Windows Mobile, currently the second most popular operating system for mobile devices.

Most mobile threats we've seen so far require user interaction (accept the file transfer then agree to run it). At first glance, therefore, it might seem surprising how well they spread. That is, until you consider the success of PC-based worms that require similar user action. The key is social engineering, often using the lure of free pornographic pictures, movie downloads, free services or make-money-fast schemes.

It's no different on mobile phones. For example, the Comwar worm uses MMS [Multimedia Messaging Service] to send itself to contacts found in a phone's address book, at a cost of around €0.35 per message. Research has shown that many users are prepared to accept files transmitted to their devices using Bluetooth, especially if the content is sex-related.

The effects of the mobile threats vary. The phone may become unusable while the worm remains installed: the Skuller Trojan, distributed via download from a variety of mobile sites, replaces system icons with a skull icon: and makes the linked service unavailable. The Mosquit Trojan sends SMS [Short Messaging Service] messages to premium rate numbers. The effects of 'crimeware' like Brador, Flexispy or one of the other mobile Trojans, allow the malware author or 'master' to steal confidential data stored on a mobile device. It's worth noting in this context that users seldom encrypt the data they store on their device, and many don't even use a power-on password.

While the 'bad guys' are still experimenting with mobile technology, we've already seen some interesting developments. These include Lasco, a hybrid virus/worm combination; Cxover, that infects files on mobile devices and PCs; and RedBrowser, a Trojan that targets phones running Java [J2ME], i.e. non-smartphones.

Although it's clear that mobile devices are far from immune to attack, it's hard to predict when the 'proof-of-concept' trickle will turn into a flood. This will depend largely on usage. Once the number of smartphones, and their use for conducting online business, reaches 'critical mass', the criminal underground will target them, just as they target any commonly used system. Today criminals use the data stored on desktops and laptops to make money illegally. Tomorrow they will seek to capture data on mobile devices for the same purpose.

This is why leading anti-virus vendors have developed solutions designed to protect mobile devices, both in the form of software installed on the device itself and for use by mobile service providers.

Today's challenges

Since the appearance of the first PC viruses, the threat to businesses and individuals has changed beyond recognition. In those early days, no one would have anticipated the sheer numbers, or variety, of malicious programs that exist today. Each successive wave of malware development has brought with it new challenges and has required changes to existing solutions, the development of new solutions or the re-application of technologies from outside the anti-virus world. So not only does the threat landscape radically differ to that of 20 years ago, so too are today's security solutions. In particular, we've seen a change in the motivation of malware authors, from 'cyber vandalism' to the use of malicious code to make money illegally. This has placed a greater emphasis than ever before not only on delivering timely protection from the 200 or so new threats that appear daily, but also on delivering solutions that can block new, unknown threats without the need for a new signature. Early anti-virus solutions look one-dimensional compared to the holistic solutions delivered by today's leading security software providers. In addition to this, changing business practices, and the disappearance of traditional network perimeters mean that security solutions have to be capable of protecting the enterprise and its staff wherever they are and however they do business.