Viruses and Worms

Do you have a virus? No instruments, no senses can tell you if you are in the presence of the predator.

—Richard Preston, The Hot Zone

If doctors and pharmacists worked like anti-virus vendors, we’d all be immunized against all illnesses. Would this improve our viability as a species?

—David Harley, Icarus

IN THIS CHAPTER

• Understanding Viruses and Worms 324
• Objects at Risk of Virus Infection 327
• Who Writes Viruses, and Why? 327
• Anti-Virus Utilities 342
• Future Trends in Viral Malware 344
• Publications and Sites 345
This chapter addresses one of the best-known, most-feared and least-understood problems in information security. It explains what viruses and worms really are (and aren’t), summarizes the means of limiting their impact, and, most importantly, includes some pointers to further information.

Understanding Viruses and Worms

Computer viruses are perhaps the most well known and feared security threats of all. Certainly, they’re among the most misunderstood. All viruses entail a certain degree of damage, but their impact, with some very prominent exceptions, is mostly social.

Every virus does cause some (usually) limited denial of service because they all steal disk space, memory, and/or clock cycles (processor time). Some cause unintended (accidental) damage on some systems. Some do intentional damage to files and file systems, and a few can make some hardware effectively unusable by trashing firmware (CIH, for example). At this time, no known virus directly damages hardware, although the possibility of such a virus can’t be discounted. However, some of the most successful viruses (in terms of survival) achieve longevity by virtue of the fact that they do nothing but replicate and therefore aren’t conspicuous. Direct damage tends to be noticeable. However, some viruses cause serious damage to data by slow and insidious corruption, and others continue to survive despite their high damage profile.

It has to be said even an innocuous virus can cause problems just by being there, or even by being misdiagnosed as being there. This can result in secondary damage because of inappropriate action taken by poorly informed virus victims. It can also result in social damage. Such damage can include loss of reputation, scapegoating of the victims of a virus attack, or even legal action. A victim might be accused of failing to apply “due diligence”, of being in breach of contract, or of being in contravention of data protection legislation. He might even be accused of implication in the dissemination of a virus, which is illegal in many countries (even those in which the actual creation of viruses is not in itself a crime).

Viruses that do no intentional damage are sometimes described as benign, in much the same way that a tumor might be defined as malign or benign. However, this usage is potentially misleading because the use of benign in this context does not mean harmless, let alone benevolent, as it might be understood to mean.

The meteoric expansion of Internet usage (especially email) since the early 1990s has raised the status of the virus from an occasional nuisance to everyone’s problem. The vastly increased use of local networks and other means of sharing data and applications has also increased the risks by orders of magnitude. In brief, viruses can travel further and faster than was the case a
few years ago. The big comeback story in the virus field is that of the computer worm. In the early 1990s, Internet usage became less specific to “big iron” mainframes and minicomputers reached via dumb terminals and terminal emulators. The first generation of worms declined in impact accordingly. Virus and anti-virus technology became focused on the individual desktop PC. In the latter part of the decade, however, virus writers began to rediscover worm mechanisms as a means of accelerating dissemination, until worms and worm/virus hybrids have now become one of the most aggravating problems faced by systems administrators.

This chapter, although it addresses worm mechanisms is some detail, isn’t particularly focused on differentiating between viruses and worms. Even within the industry, the terms are often used interchangeably in the context of the hybrid viruses/worms that dominate the current virus scene.

**What Is a Computer Virus?**

Most anti-virus professionals would accept a working definition of the term *computer virus* like this: “a program that replicates by ‘infecting’ other programs so that they contain a (possibly evolved) copy of the virus.” (F. Cohen: *A Short Course on Computer Viruses*.)

Note that the emphasis here is on reproduction by infection. A virus is not *per se* destructive, whereas a destructive program is not *per se* a virus. Furthermore, although most viruses do attempt to operate without the knowledge of the system user, this isn’t a requirement either. The only defining characteristic is *replication*: the primary ‘intent’ of the infective program is to reproduce.

**NOTE**

The term *program* does not necessarily imply a program file, although, most viruses do in some way infect files. Nevertheless, we refer to infected and infective objects in this chapter unless we are specifically considering file infection, so as to include boot sector infectors and macro programs embedded in data files.

*Infection* is sometimes described in terms of attachment of the viral program to one or more programs on the target system. However, *attachment* is perhaps a misleading term, although it is conventionally used in this context because the word *attachment* has a rather different connotation in the context of email. It might be more useful to look at the process in terms of a chain of command. The viral code is inserted *into* the chain of command so that when the legitimate but infected program is run, the viral code is also executed (or in some instances, runs *instead* of the legitimate code).
We often describe infection in terms of the viral code becoming physically attached to the host program, but this isn’t always the case. Sometimes, the environment is manipulated so that calling a given program calls the viral program. Sometimes, the viral program is activated before any program is run. This can effectively “infect” every executable file on the system, even though none of those files are actually physically modified. Viruses that take this approach include cluster or FAT (File Allocation Table) viruses, which redirect system pointers to infected files; companion viruses; and viruses that modify the Windows Registry so that their own code is called before legitimate executables.

Except for a few extraordinarily primitive and destructive examples that actually trash the host program on infection, all viruses work along these lines:

- A computer user calls a legitimate program.
- The virus code, having inserted itself into the chain of command, executes instead of the legitimate program.
- The virus code terminates and hands over control to the legitimate program.

Companion or spawning viruses follow the same sequence, but the virus code is contained in a separate file, which is (characteristically) renamed so that it will be executed instead of the program the victim thought he was launching. (It then normally hands over control to the legitimate program.)

The virus process is a little like the process of biological viral infection, although the analogy is overworked and can be misleading. Think of a person infected with an airborne disease. Whenever he exhales in a public place, he risks infecting others. Similarly, whenever an infected program is executed, the virus’s infective routine also runs, and can infect one or more other objects “in range”. Just as biological viruses infect hosts that are predisposed to infection, computer viruses target certain type of files and system areas, according to virus type.

**What Is a Computer Worm?**

Replication is also the defining characteristic of a worm, and some authorities (including Fred Cohen, the “father” of computer virology) regard worms as a subset of the genus virus. However, worms present particular problems of definition. One viable definition distinguishes between worms and viruses in terms of attachment. Whereas a virus in some sense “attaches” to a legitimate program, a worm copies itself across networks and/or systems without attachment. It can be said that the worm infects the environment (an operating system or mail system, for instance), rather than specific infectable objects, such as files.

Some observers have used the term *worm* to refer to *self-replicating malware* (MALicious softWARE) that spreads across networks. This doesn’t really amount to a meaningful distinction because many viruses can travel between machines on a Local Area Network, for instance,
without being “aware” that a target volume is not on the same machine. This isn’t to say, of course, that viruses are never network aware.

**Objects at Risk of Virus Infection**

Thousands of new viruses have been reported in recent years. Viral mechanisms differ widely, and any type of file can be affected. However, viruses can only spread when code is executed, which means that only files or other objects (such as the boot sector) containing executable code can be carriers for further infections. This doesn’t mean, however, that only binary executable files such as DOS/Windows .EXE and .COM files can be infected.

Some data files can also contain executable code in the form of embedded macros. At present, Microsoft Office includes two of the applications (Word and Excel) that are most vulnerable to virus attacks that take advantage of interpreted macro/scripting languages, such as Visual Basic for Applications and its siblings. Although macro viruses are possible (and exist) for non-Microsoft applications, word processors that store macro code in separate files rather than within document files are arguably less vulnerable. People are far less likely to swap macros than documents.

Shell scripts, batch files, interpretable source code, even Postscript files also contain executable code and could, in theory, be vulnerable to virus attack. The likelihood of such an attack depends on a number of factors, however, such as the popularity of the platform and the access controls native to the operating environment. The restricted write access allowed to unprivileged accounts in a multiuser environment like UNIX or NT does tend to impede the spread of viruses and Trojans in such environments. However, it would be unwise to rely exclusively on this fact for protection of such systems. Some of the earliest experiments with viruses were, in fact, made on UNIX systems.

**Who Writes Viruses, and Why?**

There are certain stereotypes associated with virus writing. On the whole, they’re rarely useful. Most virus writers try, for obvious reasons, to preserve their anonymity, so testing the truth of these images is somewhat problematic. Some virus writers do discuss and display their craft and their angst in more-or-less public forums such as the newsgroup alt.comp.virus. These do seem to tend to be young males, and some research indicates that mostly they “age out” and leave the field as they acquire girlfriends and a life.

However, it’s unsafe to assume that the “virus writers” who dominate such newsgroups are always who they say they are, let alone that they are as talented as they claim to be, or necessarily serious representatives of all virus writers. Indeed, it’s possible that this group represents
a constituency of wannabes rather than a group of real, competent virus writers. Certainly
many successful viruses seem to have been written by focused loners with no particular affiliations, rather than by groups.

It’s also noticeable that many of the most vociferous individuals quoted and feted by the
media, law enforcement agencies, politicians, and others are not widely respected among their peers. Of course, the same is true of other types of computer vandals, not to mention many self-styled security and/or virus experts.

Some virus writers have responded to the very few serious attempts at research in this area. However, quantitative research is not realistically possible, and the research that has been done leans to the ethnographic. That is, rather than try to establish numerical data with large samples, researchers in this field have tended to rely on qualitative data, using interviews with very small samples (that is, just a handful of virus authors).

The acknowledged authority in this area is Sarah Gordon, who has written extensively in this area and in related ethical areas. Her papers for the Fourth and Sixth Virus Bulletin
Conferences on The Generic Virus Writer are particularly relevant. A number of her papers, including both Generic Virus Writer papers, can be found at

A second widespread stereotypical notion is that people who write anti-virus software also write viruses, in an attempt to drum up business for their products. I can’t say with absolute
certainty that no vendor or researcher has ever written a virus, released a virus, or even paid a bounty for samples of original viruses. However, it’s hard to comprehend why any anti-virus professional would see a need to stray toward “the Dark Side” at this stage of the game. There are more than enough amateurs producing viruses. In Generic Virus Writer II, Gordon notes
that older security professionals, especially systems administrators and such, make their own contribution to the virus glut through (probably well-meant) experimentation. However, despite the eagerness of virus writers to implicate “the enemy” in the problem, there is no conspiracy between systems administrators and vendors to keep vendor profits high. Or if there is, no one has offered me a percentage.

There are probably as many reasons for writing viruses as there are virus writers, although the reasons cited by virus writers (actual or wannabe) don’t always stand up to closer analysis. Some appear fascinated by the concept of a self-replicating and/or self-modifying program, and are curious to see how far their creations spread. Indeed, some apologists suggest that virus writing is a legitimate means of research into artificial life forms, or even artificial intelligence. (However, the adaptive behavior displayed by even the most sophisticated viruses is usually rather restricted.)
Many virus authors seem to enjoy matching wits with the anti-virus establishment. Indeed, some viruses go straight from the creator to his favorite anti-virus company without any attempt to spread it through the general population. Others, however, are more concerned with inspiring the admiration of their peers, rather than gaining the attention of the anti-virus professionals. Others don’t make a hard and fast distinction between writing viral and anti-viral software, and might write both. This isn’t normally the case in the anti-virus industry, and those who’ve used their experience on both sides of the barbed wire to support their search for a job in the industry have usually been sadly disappointed. In fact, development teams in the industry have practical as well as ethical reasons for preferring to employ programmers whose experience is in other areas. It saves them having to clean ill-founded technical preconceptions out of the newcomer’s head.

There are, of course, many viruses that are intended to cause widespread damage, although deliberate destruction is the goal far less often than most people seem to believe. (Often, virus damage comes from thoughtlessness or sheer incompetence on the virus writer’s part.) Some virus writers argue that computer users who don’t have the technical savvy to protect themselves deserve everything they get. On the other hand, some virus writers also claim that they have no personal involvement in virus dissemination, and are not responsible for the use made of their code by others. In other words, the distributors are the problem, not the authors. This would be more convincing if such authors never made their creations available as source code and/or binaries on Web sites, in e-zines, and other locations. Then, viruses would be less easily available to anyone who asks, or trawls Vx (Virus eXchange) Web sites.

How Are Viruses Created?

Some people seem to believe that computer viruses appear spontaneously in the same way that biological viruses seem to do. This isn’t quite as silly as it sounds. Completely new viruses don’t just pop out of the primeval soup without warning. However it’s not uncommon for a new variant (not necessarily a viable virus in terms of replication and the capability to infect) to be born without direct human intervention. For instance, a macro virus consisting of a fixed number of modules might mutate by losing some of its constituent macros or gaining unconnected (not necessarily viral) macros. WM/Cap, for example, mutated into many hundreds of variants of the original virus. However, someone had to write the original version.

It’s not impossible that an operating environment might come into general use in which a viral program could be created from scratch without direct human intervention, but it doesn’t seem to have happened yet.

Most virus writers (and a high percentage of the rest of the world) have an exaggerated view of the ability needed to produce a working virus. Undoubtedly, some virus writers produce technically competent code: many more don’t. Furthermore, as we’ve seen, many viruses are
one-trick ponies. They might do the replication trick well or not so well, but replication, even when done efficiently, represents a somewhat limited functionality, compared to that of a compiler or business application.

Older viruses were often written in assembly language. In fact, it’s difficult to write some types of virus in a high-level language, even with the help of an inline assembler. This is an advantage, from the viewpoint of virus victims, in that it takes a certain level of programming expertise to create even a weak virus (or even to modify an existing virus so as to create a variant).

Many variants are, in fact, simply existing viruses with a slight change that doesn’t affect functionality (such as modification to unimportant embedded text). Such a change might require no programming at all.

Some virus writers and their admirers still regard proficiency in assembly language as the hallmark of programming excellence. (This is actually in sharp contrast to the professional programmer, whose choice of tool, given a choice, is liable to be somewhat more pragmatic.) However, the current is, by and large, flowing the other way.

As virus technology developed, some virus programmers turned their attention to creating kits to allow a wannabe virus author to “develop” other viruses without programming. That is, using virus generators to produce virus code. This has not, however, necessarily resulted in an increase in the total number of viruses “in the wild.”

Kit viruses are often not actually viable (that is, they don’t replicate), and are frequently detectable generically. A new kit virus might be identifiable as having been generated by a particular generator, simply by family resemblance. Thus, kit viruses have tended to contribute to the “glut” problem (the sheer weight in numbers), rather than to the “in-the-wild” problem (see next section).

Certainly, assembly language is not necessarily the language of choice among the current generation of virus writers. Interpreted macro languages (especially Visual Basic for Applications) are generally harder to use than kits, but much easier than assembler. Furthermore, disk space and main memory are no longer expensive, and grossly bloated files are less conspicuous in a Windows environment. Thus, it’s become more practical (as well as easier) to write viruses and worms in C++ or Delphi.

**What Does “In the Wild” Really Mean?**

A virus is deemed to be “in the wild” when it has escaped or been released into the general population. The general population refers to computing environments outside the development environment where the virus was created and tested, or the collections of anti-virus vendors, researchers, and collectors. Viruses in these environments are typically (hopefully) processed under controlled circumstances, where no danger is posed to the surrounding communities. However, when a virus escapes a controlled environment, it might be said to be “in the wild”
(often expressed adjectivally in the anti-virus community as In-the-Wild or ItW). Note, however, that in the anti-virus community, the fact that a virus is available on a vx (Virus eXchange) bulletin board or Web site does not make it In-the-Wild. Because access to such resources and exchange of viruses is voluntary, this counts as a controlled environment.

In his conference paper *Counting Viruses* (Virus Bulletin 1999), Paul Ducklin makes the distinction very clearly:

“For a virus to be considered In the Wild, it must be spreading as a result of normal day-to-day operations on and between the computers of unsuspecting users. This means viruses which merely exist but are not spreading are not considered ‘In the Wild’.”

In fact, the definition used by the WildList Organization is far stricter. For a virus to be on the WildList, the nearest thing to an industry standard metric for “In-the-Wildness,” it must be reported by two or more of the virus professionals who report to the WildList Organization. Furthermore, these reports must be accompanied by replicated samples. (Viruses that are reported by only one reporter are put into the supplementary list.) Clearly, this strictness means that the WildList can’t represent all the ItW viruses at a given time, but does represent viruses that are genuinely “out there”. Such data are often more useful than absolute numbers to the organizations and individuals using the WildList as a basis for testing and research. (Note that the WildList indicates a virus’s presence “out there”, but not the total number of virus incidents in which a single virus is implicated. Thus the list only provides a very rough guide to prevalence.)

What matters most for our purposes, however, is the disparity between the number of In-the-Wild viruses at one time (a few hundred according to the prevailing WildList) and the total number of viruses in existence. (At the time of writing between 50,000 and 60,000, depending on how you measure.)

**How Do Viruses Work?**

A virus is, conceptually, a simple program. In its simplest form, a *direct action virus* can be modeled in terms of an algorithm like this:

```plaintext
begin
    Look for (one or more infectable objects)
    If (none found)
        then exit
    else (infect object or objects).
    endif
end
```

They don’t remain in memory, but execute all their code at once, and then hand over control to the host program.
Many viruses go memory resident (install themselves into memory) after the host program is executed, so that they can infect objects accessed after the infected application has been closed.

The term *hybrid* is sometimes used for viruses that stay active as long as the host program is running. It is also (perhaps with more justification) applied to viruses that are both direct action and memory resident.

In fact, all viable boot sector infectors are memory resident—they have to be. Otherwise, their code can only be executed during the boot process, which rather limits their opportunities to infect other boot sectors. We consider boot sector viruses in detail later on in this chapter.

Of course, all but the most incompetent viruses are a little better error-trapped than this, and at least check that the *infectable object* hasn’t already been infected. You’ll notice that I’ve also skated over the *infect object* subroutine. We’ll come to infection mechanisms when we discuss the main virus types later in this section.

Some viruses, of course, do more than just replicate. We sometimes describe viruses as having up to three components: an infective routine, a payload, and a trigger. The previous models demonstrate an infective routine, although it could be said that finding an infectable object is the trigger for the infective routine. However, we more often think of the trigger as being the condition that has to exist before the payload (or *warhead*) can be executed. The payload can, in principle, be any operation that any other program can perform. In real life, however, it tends to be something flippant and irritating, like visual or audio effects, or else downright destructive. So now our model looks more like this:

```
begin
  (Go resident)
  if (infectable object exists)
    then
      if (object is not already infected)
        then
          (infect object)
        endif
    endif
  if (trigger condition exists)
    then
      (deliver payload)
    endif
end
```

The trigger condition might, for instance, be the execution of a file, or a particular date or time. The combination of a trigger and a malicious payload is sometimes called a *logic bomb*.

Viruses can be classified conveniently (but by no means definitively) into five main classes: Boot Sector Infectors (BSIs); file infectors; multipartite viruses; macro viruses; and scripting viruses.
Boot Sector Infectors (BSIs)
These PC-specific viruses infect the Master Boot Record and/or DOS Boot Record. At one time, these viruses accounted for the majority of reported incidents, but now they constitute a dwindling proportion of the total number of threats found in the wild, and new BSIs are something of a rarity. This might reflect the fact that people now increasingly use email and networks rather than floppy disks to exchange files. The fact that these are harder to write than macro viruses and scripting viruses (or even file viruses) is also relevant.

When a modern PC boots up, it goes through a process called Power On Self Test (POST). This stage of the boot process includes checking hardware components. Some of its information comes from information stored in CMOS, especially information relating to disk and memory type and configuration. If the CMOS settings don’t match the actual drive geometry, the machine will not be able to find system areas and files where they should be, and will fail to finish the boot process.

The Master Boot Record (MBR), sometimes known as the Partition Sector, is found only on hard disks, where it is always the first physical sector. It contains essential information about the disk, giving the starting address of the partition(s) into which it is divided. On diskettes, which can’t be partitioned and don’t contain an MBR, the first physical sector is the boot record or DBR. On hard drives, the boot record is the first sector on a partition. The boot record contains a program whose job is to check that the disk is bootable and, if so, to hand over control to the operating system.

By default, if there is a bootable floppy present, most PCs will boot from drive A, the first floppy drive, rather than from drive C, the first hard drive. This is actually an unfortunate default because this is the normal entry point for a boot sector virus. If the PC attempts to boot from a floppy with an infected boot sector (even if the floppy doesn’t contain the necessary files to load an operating system and therefore can’t complete the boot process), the infected floppy will infect the hard drive. Characteristically (although not invariably), once the hard drive is infected, the virus will infect all write-enabled floppies.
The majority of boot sector viruses also contain some provision for storing the original boot sector code elsewhere on the drive. There is a good reason for this. It isn’t because the virus programmer kindly intends to eventually return the MBR to its original state, although retaining a copy of the original boot sector can make disinfecting the virus easier. Rather, it is because he has to. Typically, a virus will keep a copy of the original boot record and offer it whenever other processes request it. This not only enables the system to boot in the first place, but also makes it harder to detect the virus without anti-virus software that specifically recognizes it. However, some viruses simply replace the normal boot sector code with code of their own.

Some BSIs (Form is a particularly well-known and widespread example) only infect the boot record, even on hard disks. This creates particular problems with Windows NT and Windows 2000, and will usually prevent the system from booting at all. Thus a largely innocuous virus has suddenly become a major nuisance in some environments.

NOTE

You might have heard that boot sector viruses can be disinfected without anti-virus software, using FDISK with a (largely) undocumented switch (/MBR), known in some quarters as FDISK/MUMBLE. The good news is that this works a lot of the time. The bad news is that, if you try it with the wrong virus, you can actually lose access to your data. Anti-virus software is a very imperfect technology, but it’s almost invariably better and safer for removing viruses than general-purpose utilities that were never designed for that purpose. FDISK is not recommended as an anti-virus measure unless you know exactly what you’re doing.

The majority of boot sector viruses also contain some provision for storing the original boot sector code elsewhere on the drive. There is a good reason for this. It isn’t because the virus programmer kindly intends to eventually return the MBR to its original state, although retaining a copy of the original boot sector can make disinfecting the virus easier. Rather, it is because he has to. Typically, a virus will keep a copy of the original boot record and offer it whenever other processes request it. This not only enables the system to boot in the first place, but also makes it harder to detect the virus without anti-virus software that specifically recognizes it. However, some viruses simply replace the normal boot sector code with code of their own.

Some BSIs (Form is a particularly well-known and widespread example) only infect the boot record, even on hard disks. This creates particular problems with Windows NT and Windows 2000, and will usually prevent the system from booting at all. Thus a largely innocuous virus has suddenly become a major nuisance in some environments.

TIP

New boot sector viruses are comparatively rare. Nevertheless, even old favorites like Form still circulate among people who still exchange disks. Although reputable and up-to-date anti-virus software is still a must for detecting them, a simple precaution eliminates most of the risk of infection on most PCs, even from unknown BSIs. Most PCs, by default, will attempt to boot from drive A if there is a diskette there. If there isn’t, it tries to boot from drive C. However, nearly all PCs can be reconfigured in CMOS to change this default. On most systems, this is done by modifying the boot order, so that the system always tries to boot from drive C first (or in the order CD drive, drive C, drive A). Other systems (notably some Compaq models) allow the setting of an option to disable booting from the floppy drive altogether. If the system user actually needs to boot from floppy, this simply involves resetting the option to default. Motherboard and PC system vendors use proprietary ways of setting CMOS options. Consult the documentation that came with your system. Note that “file and boot” (multipartite) viruses are less likely to be contained by this precaution.
**File Viruses (Parasitic Viruses)**

File viruses infect executable files. Historically, most file viruses have not been particularly successful in terms of their epidemiology (that is, at spreading). Many thousands have been written, but the number actually seen in the wild has been comparatively small compared to BSIs and, more recently, macro viruses. Nonetheless, those that have survived in the wild have often spread surprisingly well—CIH, for example. Some of the most prevalent contemporary file viruses, however, are more commonly described as worms, as considered later in this chapter.

After a virus infects an executable file by direct attachment, that file, when executed, will infect other files. *Fast infectors* go for instant gratification. Each time the infection routine is executed, it infects a whole directory, all folders on the current path, a whole volume/disk, even all currently mounted volumes. Even file infectors that infect only one or two files at a time can spread quickly across systems and networks in a modern environment, where multiple binary executables are opened and closed many times over a single session. Every time you open an application, at least one executable file is loaded. Some applications will open several files at startup, whereas others periodically open multiple files when performing a particular operation.

Sparse infectors forgo the temptation to infect as many files as possible, usually in an attempt to make themselves less conspicuous. They may not infect every time the virus is executed, but only under very specific conditions, even when an infectable object is there to infect.

**Note**

Binary executables are by no means restricted to .COM and .EXE files, but include DLLs (Dynamic Link Libraries), overlay files, VxDs and other classes of driver, overlay files, and even certain screensaver and font files.

**Multipartite Viruses**

File and boot viruses are the most common example of multipartite viruses, viruses that use more than one infection mechanism. In this case, both boot-sectors and binary executable files might be infected and used as the means of disseminating the virus. However, it’s likely that there will be an increase in multipartite viruses consisting of other combinations of virus types.

**Macro Viruses**

Macro viruses infect macro programming environments rather than specific operating systems and hardware. Microsoft Office applications are by far the most exploited environment. These can be regarded as a special case of file virus, in that they appear to infect data files rather than binary executables. However, this way of looking at the process might actually confuse the
issue. Macros are essentially a means of modifying the application environment, rather than (or as well as) the data file. Indeed, in the case of Microsoft Office applications that support macro programming languages (Visual Basic for Applications and, in earlier versions, WordBasic and AccessBasic), the macro language cannot be unbound from the application’s own command infrastructure. Macro viruses usually infect the global template, and often modify commands within the application’s menu system. Macro viruses are particularly successful against Microsoft applications because they allow executable code (macros) to exist in the same file as data. Applications that segregate macros and data into different files are less susceptible to this kind of attack.

**Script Viruses**

Script is rather an imprecise term, but in this context normally (currently) refers to VBScript and other malware that can be embedded in HTML scripts and executed by HTML-aware email clients through the Windows Scripting Host. Many of the viruses that use this entry point are often better characterized as worms, and are therefore treated under that heading later in the chapter. VBScript and Jscript are more virus friendly than JavaScript (for instance), primarily because they have many of the file I/O capabilities of other variations on the Visual Basic theme. Extant JavaScript malware usually takes advantage of an easily patched vulnerability in Internet Explorer.

This view of script viruses is rather restrictive. A broader definition might include HyperCard infectors, batch file infectors, UNIX shell script infectors, and many more. However, these are of less practical importance, currently.

**Memetic Viruses**

There is a further class of “viruses,” which is unique, in that it comprises viruses that don’t exist as computer code. The term *meme* seems to have been coined originally by Richard Dawkins, whose paper *Viruses of the Mind* draws on computer virology as well as on the natural sciences. A *meme* is a unit of cultural transmission, of replication by imitation, much as a gene is a unit of inheritance (a rather imprecise unit, perhaps). The memes we are most concerned with in this chapter are those sometimes known as *metaviruses*. A metavirus is itself a virus (what Dawkins calls a “virus of the mind, not a computer virus”), but purports to deal with other viruses (which are computer viruses). These viruses don’t happen to exist. In other words, they are virus hoaxes. Virus hoaxes are not only a subclass of memes in general, but a subset of a particular type of meme, the chain letter. However, the virus hoax is particularly relevant to this chapter, because the administrator who manages virus incidents will usually also be the person who has to respond to plagues of virus hoaxes. The same might not be true of other hoaxes and chain letters.
The most commonly encountered hoaxes are derived from the infamous Good Times hoax of the mid-1990s. They conform to a pattern something like this:

[THIS WARNING WAS CONFIRMED BY SYMANTEC AND MCAFEE THIS MORNING.] IF YOU RECEIVE EMAIL WITH THE SUBJECT <GREEN EGGS AND HAM> DO NOT OPEN IT, BUT DELETE IT IMMEDIATELY!!! IT CONTAINS A VIRUS THAT JUST BY OPENING THE MESSAGE TRASHES HARD DRIVES AND CAUSES MOUSE-MATS TO SPONTANEOUSLY COMBUST. MICROSOFT, AOL, IBM, FCC, NASA, CND, AND KKK HAVE ALL SAID THAT THIS IS A VERY DANGEROUS VIRUS !!! AND THERE IS NO REMEDY FOR IT AS YET. PLEASE FORWARD THIS TO ALL YOUR FRIENDS, RELATIVES, COLLEAGUES, AND ANYONE ELSE WHOSE EMAIL ADDRESS YOU HAVE HANDY SO THAT THIS DISASTER CAN BE AVERTED.

By the way, as far as I know there is no Green Eggs and Ham virus or hoax. I’ve just done what many real hoaxers have done and pulled a silly title out of thin air (or in this case my daughter’s bookshelf). In fact, the infuriating aspect of this problem is that most hoaxers are abominably lazy and unoriginal, and the subject of the email which carries the supposed virus is often the only bit of the hoax that varies between two variants.

This sort of hoax only continues to work because masses of people with little technical knowledge of computers (let alone computer viruses) join the Internet community for the first time every day. Each one is at high risk of passing on such a hoax because they don’t know any better. Of course, a hoax can be much more subtle than this, but I’m not here to tell you how to write a hoax that might fool even an expert.

Here are a few of the features that would alert the experienced hoax watcher to the unreliability of the Green Eggs and Ham alert:

- Uppercase is used throughout and the message carries clusters of exclamation marks for emphasis. This doesn’t, of course, prove anything about the accuracy of the alert. Nevertheless, it’s been observed many times that use of uppercase, liberal exclamation marks, and poor spelling, grammar and style characterize most of the common hoaxes. On no account, however, should you assume that an alert is accurate simply because it doesn’t have these characteristics.

- The reference to McAfee and Symantec doesn’t give contact or reference information. It’s just there to add credibility to the hoax. There’s no real indication of when it was written, either. There are hoaxes circulating the Internet right now, saying that IBM announced something “yesterday,” that have been around for years. The “yesterday” is just there to give a false impression of urgency.

- It’s true that some email viruses/worms arrive with a characteristic subject header. However, there are many others that don’t, and it makes more sense to avoid executing
any attachment than to try to remember which silly header goes with which virus. In fact, administrators trying to block particular viruses by filtering mail on subject alone and using inappropriate criteria are responsible for a whole subclass of indirect Denial of Service (DoS) attacks in and on themselves.

- It makes sense to be cautious about email, but just opening a message can only infect your system if you have certain mail programs (Outlook, primarily) set with incautious defaults. Most mailers don’t execute code just by viewing the message. An alert that says that this will happen but doesn’t specify any particular mailer, should be regarded with suspicion.

- It’s implied that the malicious code works on any hardware. This is pretty suspicious. What’s more, a payload that triggered as soon as you opened the message/attachment would be pretty ineffective at spreading. You might think the mouse-mat payload is a bit over the top. Actually, real hoaxes are often as ridiculous as this (although they often conceal their improbability behind technobabble).

- Of all the organizations listed, only IBM has any real expertise in viruses. The others are only listed to impress you.

- It’s claimed that there is no “remedy” for the virus. Anti-virus vendors can usually supply fixes for new viruses in hours, even minutes. Of course, the effects of some viruses might be impossible to reverse, but data recovery firms can perform near-miracles sometimes.

- A virus that trashes your system as soon as you execute it is unlikely to travel very far. What is being described here sounds more like a destructive Trojan, and they don’t generally spread well through email.

- The warning urges you to forward the mail to everyone you know. That makes it a chain letter. Reputable and knowledgeable organizations don’t send alerts that way, although clueless ones sometimes do.

How Do Worms Work?
The 1988 Morris Worm (the Internet Worm) and its siblings, such as WANK and CHRISTMA EXEC, usually targeted heavy-duty mainframe and minicomputer hardware, mail, and operating systems. More recent threats have been aimed primarily at PCs, and, in one highly publicized incident (the AutoStart worm), Apple Macs. However, they might have the incidental effect of bringing down mail servers through the sheer weight of traffic they generate. Some of these have been variously classified by different researchers and vendors as viruses, as worms, as virus/worm hybrids, and occasionally as Trojan horses.

Today’s worms and email viruses tend to be fast burners. They have the potential to spread globally before anti-virus vendors have time to analyze them and to distribute means of detection and disinfection. Some of the malware commonly referred to as worms are actually
specialized viruses that infect only one file. This doesn’t mean, of course, that a virus like Lehigh, which infects only COMMAND.COM, can sensibly be defined as a worm.

Universally accepted classifications of worms don’t exist, but Carey Nachenberg, in a paper for the 1999 Virus Bulletin Conference, proposed a classification scheme along the following lines:

- Email Worms, unsurprisingly, spread via email.
- Arbitrary Protocol Worms spread via protocols not based on email (IRC/DCC, FTP, TCP/IP sockets).

As well as proposing classification by transport mechanism, Nachenberg also proposed classification by launching mechanism:

- Self-launching Worms such as the 1988 Internet Worm require no interaction with the computer user to spread: They exploit some vulnerability of the host environment, rather than in some way tricking the user into executing the infective code. However, KAK and the rather rarer BubbleBoy are examples of self-launching worms. By exploiting a bug in the Windows environment, they can execute without user intervention.

For information on dealing with this problem by applying a patch, see http://support.microsoft.com/support/kb/articles/Q262/1/65.ASP.

- User-launched Worms interact with the user. They need to use social engineering techniques to persuade the victim to open/execute an attachment before the worm can subvert the environment so as to launch itself onto the next group of hosts. Many of today’s VBScript worms fall into this or the Hybrid-launch category.

In fact, some of the worms we’ve seen to date are probably better classified as Hybrid-launch Worms (by Nachenberg’s classification scheme) or multipartite (in terms of conventional virus terminology) because they use both self-launching and user-launched mechanisms.

**Virus Characteristics**

The following characteristics are not necessarily restricted to particular virus/worm classifications, but are of some importance if only because of the way the terms *stealth* and *polymorphism* are so often misused:

- **Stealth.** Almost all viruses include a degree of stealth, that is, they attempt to conceal their presence in order to maximize their chances of spreading. There have been viruses that asked permission before infecting, but this courtesy has not been rewarded by wide dissemination. Conspicuous payloads tend to be avoided, or are delivered fairly irregularly. Stealth viruses use any of a number of techniques to conceal the fact that an object has been infected. For example, when the operating system calls for certain information,
the stealth virus responds with an image of the environment as it was before the virus infected it. In other words, when the infection first takes place, the virus records information necessary to later fool the operating system.

This also has implications for anti-virus tools that work by detecting that something has changed rather than by detecting and identifying known viruses. To be effective, such tools must use generic anti-stealth techniques. Of course, it isn’t possible to guarantee that such techniques will work against a virus that has not yet been discovered. However, virus scanners that detect known viruses are at an advantage in this respect, because vendors will normally compensate for a new spoofing technique when they add detection for the virus that employs it. The trick employed by some BSIs of displaying an image of the original boot sector as if it was still where it belonged is a classic stealth technique. File viruses characteristically (but not invariably) increase the length of an infected file, and can spoof the operating system or an anti-virus scanner by subverting system calls so that the file’s attributes before infection, are reported, including file length, time and datestamp, and CRC checksum.

- **Polymorphism.** Polymorphic viruses are adored by virus authors and feared by nearly everyone else. This is partly because of an over-estimation of the impact of the polymorphic threat. Non-polymorphic viruses usually infect by attaching a more-or-less identical copy of themselves to a new host object. Polymorphic viruses attach an evolved copy of themselves, so that the shape of the virus changes from one infection to another. Early polymorphic viruses used techniques such as changing the order of instructions, introducing noise bytes and dummy instructions, and varying the instructions used to perform a specific function. A more sophisticated approach is to use variable encryption, drastically reducing the amount of static (unchanging) code available to the anti-virus programmer to use to extract a pattern by which the virus can be identified. You might imagine (as many people do) that this makes polymorphism a formidable technology to counter. Indeed, the emergence of polymorphic viruses and plug-in mutation engines (enabling almost any virus author to include variable encryption in his own work without reinventing the wheel) contributed to the disappearance of some of earlier anti-virus packages. However, although polymorphic viruses are popular with virus authors demonstrating their skills, they have been less well represented in the field than in the collections of anti-virus researchers, certification laboratories, comparative testers, and others who need as complete a collection as possible. Anti-virus scanning technology has also moved on, and simple signature scanning for a fixed character string doesn’t play a large part in the operation of a modern scanner.

The classifications of viral malware described earlier do not cover the entire range of objects detected by anti-virus software. Some vendors are quick to point out that what they sell is anti-virus software, not anti-malware software. Nonetheless, nowadays most commercial products
detect some Trojan horses (see Chapter 18) and other objects that barely qualify as malware, let alone viruses. Such objects include intended (non-functioning) viruses, joke programs, DDoS programs (Distributed Denial of Service), even garbage files that are known to be present in poorly maintained virus collections likely to be used by product reviewers.

It might be noticeable that this chapter has been largely PC-centric. Certainly, there are more viruses that infect PC platforms (DOS and all flavors of Windows) than any other operating system. Native Macintosh viruses are far fewer. In fact, there are probably more native viruses on systems such as Atari and Amiga that have never had the same popularity (in corporate environments, at least). However, the fact that Apple Macintoshes share with Windows a degree of vulnerability to Microsoft Office macro viruses makes them the other main virus-friendly environment today.

It should not be assumed, however, that other platforms don’t have virus problems. Access controls can be imposed on unprivileged accounts in UNIX (including Linux), NT, NetWare, and other platforms to restrict infection flow. However, they can’t prevent unprivileged users from sharing files, if only by email. Nor can they prevent a privileged user inadvertently spreading infection. Even systems that don’t support any known native viruses (servers or workstations) can carry infected objects between infectable hosts, a process sometimes known as heterogeneous virus transmission. It’s as important to scan network file servers, Intranet, and other Web servers, regardless of their native operating system. In fact, an increasing number of products detect viruses associated with other operating environments. Thus some Mac products detect PC viruses, and vice versa.

Clearly, viruses do represent a risk on the Internet. That risk is higher for those running DOS, any variant of Windows, or certain macro-capable applications, especially the Microsoft Office applications suite. Mostly this is a matter of market share. Most virus writers target PCs and Windows because that’s what they have access to. However, there are other factors that increase the risk: for example, PC hardware architecture, Microsoft’s rosy view of the lack of need for security on single-user systems, and the dangers of having macro code and data in the same file. There are some tools to help keep systems safe from virus attacks listed later in this chapter. Be aware, though, that the only way to guarantee safety is by obeying Richards’ Laws of Data Security—don’t buy a computer, and, if you do buy one, don’t turn it on. (A tip of the hat to Robert Slade for bringing that one to my attention.) Anti-virus software is mostly reactive: It responds to a perceived threat, and works most effectively against threats it can identify with precision (that is, known viruses). The best defense against unknown viruses is often to work in an environment that doesn’t provide a host to particular classes of threat. Sadly, however, this is often not an option, particularly in some corporate environments where Microsoft products are considered obligatory.
Anti-Virus Utilities

Anti-virus software can generally be defined as generic, malware specific, or hybrid. Generic software commonly includes change detection software (integrity checkers), behavior monitors, and behavior blockers. It deduces the existence of a virus from a change in the environment or an infectable object (a file, for example), or from a process displaying behavior characteristic of malware. (Note that the term malware is increasingly used with particular reference to Trojan horses rather than viruses. Trojans are considered at length in Chapter 8, as is change-detection software.)

Malware-specific software checks infectable objects against a database of virus definitions. If a match is found, it alerts the computer user and might be able to remove the virus from the infected object. This is usually possible with boot sector and macro infectors. File viruses are sometimes harder (and sometimes impossible) to disinfect, and some vendors don’t try, taking the view that it’s always better to replace a binary executable than to risk disinfecting it unsuccessfully.

Scanners can be on-access (real-time or memory-resident) or on-demand. On-access scanners check files and other infectable objects as they are accessed (especially as they’re opened for reading or writing), and can be implemented as a DOS TSR, Windows VxD, NT service, Macintosh System Extension, and so on. Most anti-virus packages include an on-access malware-specific component, but on-access change-detectors do exist. On-demand scanners are executed only when called by the user or by scheduling software. They do their job, then terminate.

Modern malware-specific scanners are better described as hybrid. Although they use more-or-less exact identification, most are also capable of a generic technique known as heuristic analysis, which is related to behavior blocking. Code is checked for characteristics that suggest a virus, either by passive analysis of the code, or by executing it under emulation, so that its behavior can be safely monitored.

Inclusion in the following list of anti-virus products doesn’t necessarily constitute a recommendation. Products change, and what works for one PC, environment, or organization won’t necessarily work well in another. However, these are all competent products. In general, URLs in this chapter have been modified since the previous edition of this book, so that only the relevant domain name is given. Experience indicates that actual pages move around a lot. For a comprehensive list of vendors, try http://www.virustn.com/AVlinks/.

AntiViral Toolkit Pro (AVP)

AVP has been licensed by a number of vendors, but its exact status is uncertain at the time of writing. However, this is a very popular product. Check the Kaspersky Labs site for information about Kaspersky Anti-Virus, at

http://www.kaspersky.com/
Kaspersky also provides a useful virus information site with virus encyclopedia at
http://www.viruslist.com/

**Network Associates**
The NAI range includes the current incarnations of McAfee and Dr. Solomon’s for a wide range of workstation and server platforms, including PCs/Windows, Apple Macs, and UNIX (including Linux). The brand names McAfee and Dr. Solomon’s are now usually applied to the same software, but the Dr. Solomon’s brand is normally only used for the UK/European market. NAI’s Web site is at
http://www.nai.com/

**Norton Anti-Virus**
Norton Anti-Virus is available for a wide range of workstation, server, and gateway platforms including DOS, Apple Macintosh, Windows 9x, and Windows NT/2000.
http://www.symantec.com/

**eSafe**
Eliashim, producer of eSafe and now part of the Aladdin empire, focuses primarily on gateway protection from viruses and other malicious software. Contact them at
http://www.eliashim.com/

**PC-Cillin**
PC-Cillin by Trend Micro can be found along with their InterScan gateway products at
http://www.antivirus.com/

**Sophos Anti-Virus**
Sophos is very focused on the corporate market. Products are available for a wide range of workstation, server, and gateway platforms, including PCs/Windows, Apple Macs, and UNIX (including Linux). Learn more at
http://www.sophos.com/

**Norman Virus Control**
Norman Virus Control (NVC) by Norman Data Defense Systems can be found online at
http://www.norman.com/
**F-PROT Anti-Virus**

A number of products have been based on the F-Prot detection engine. The original product (which is free for personal use) can be found at

http://www.complex.is

The product formerly sold by DataFellows as F-Prot Professional is now known as F-Secure, and is available at

http://www.f-secure.com

The Command Software version of F-Prot Professional is at

http://www.commandcom.com/

**Integrity Master**

Integrity Master, by Stiller Research, combines an advanced change detector with conventional known-virus scanning. The Stiller Web site is a good source of general information (hoax information, for example) and is located at

http://www.stiller.com/stiller.htm

There are hundreds of virus scanners and utilities. We have listed some previously because they have a good reputation, are easily available on the Internet, and are updated frequently. Viruses are found each day, all over the world. Most of them are unlikely ever to be seen In-the-Wild, but sometimes a formerly quiet virus will suddenly “get lucky” and go feral. New worms and other email-borne viruses like Melissa or LoveLetter can go from unknown to global within hours. Strange to think that only a few years ago, it was still normal for anti-virus software to be updated on a quarterly basis.

The second edition of this book included links to sources of freeware and shareware anti-virus utilities. These links have been removed. They haven’t been replaced with more up-to-date links, as it would be doing the reader a disservice to imply that such utilities are still a realistic substitute for commercial software. This applies even for older machines, many of which are still supported by some vendors. In anti-virus as in real life, you generally get what you pay for, or sometimes less.

**Future Trends in Viral Malware**

Virus and anti-virus technologies continue to increase in complexity and sophistication. The likelihood of contracting a virus on the Internet increases as ‘fast burner’ virus dissemination techniques evolve, and the number of potential hosts increases with the expansion of the Internet itself. It depends on where you go. If you frequent the back alleys of the Internet, you should
exercise caution in downloading any file (digitally signed or otherwise). Usenet newsgroups are places where viruses might be found, especially in those newsgroups where hot or restricted material is trafficked. Examples of such material include warez (pirated software) or pornography. Similarly, newsgroups that traffic in cracking utilities are suspect. However, the nature of the virus threat means that you are far likelier to receive an infection from someone you know, someone with no malicious intention, than from a known or anonymous virus author/distributor. We therefore recommend that you look through the guidelines to practicing “safe hex” for computer users and administrators summarized in the final section of this chapter.

Virus technology has been through a number of phases. The first big wave was the PC boot sector infector, mostly overshadowing even the parasitic fast-infector and the “big-iron” infecting worms. The second wave was largely the rise of the macro virus. Among these, the first email-aware macro viruses foreshadowed the coming of the next wave: Melissa, LoveLetter, and the macro and VBScript worms that dominate the scene at the time of writing. Many examples of the current wave of email viruses/worms are less sophisticated than the more complex, “traditional” viruses, relying to some extent on social engineering (psychological manipulation) as much as technical complexity. However, some recent examples (Hybris, MTX) combine technical complexity with social engineering.

It’s been suggested that upcoming operating systems will be so secure that viruses will cease to be a problem. However, experience indicates that as particular loopholes are patched, others are found and exploited. Expect the unexpected.

**Publications and Sites**

The following is a list of articles, books, and Web pages related to the subject of computer viruses. Some are only included or alluded to because they were in the previous edition. Some outdated links and unobtainable references have been removed, and several have been added. (We don’t guarantee that those listed are still available—in fact, you might have trouble getting hold of any but the most recent.) Inclusion of a resource in this section doesn’t necessarily constitute recommendation (as the comments make clear). However, it’s important to know and recognize the more prominent but poor resources, as well as the good ones.


*Robert Slade’s Guide to Computer Viruses: How to Avoid Them, How to Get Rid of Them, and How to Get Help* (Second Edition). Springer. 1996. ISBN: 0-387-94663-2. Four years is a long time in computing, but time has been kinder to Slade’s book than most books on the subject. This was, until recently, easily the best introductory text on the subject.


**CIAC/US Department of Energy.** This Web site has a database of virus information that was recommended in an earlier edition of this book ([http://ciac.llnl.gov/ciac/CIACVirusDatabase.html](http://ciac.llnl.gov/ciac/CIACVirusDatabase.html)). The database is no longer being updated, but is worth checking for information on older viruses. CIAC/DOE have done sterling work in recent years on publicizing the problems associated with virus hoaxes and other chain letters. The relevant pages continue to be maintained and expanded. [http://HoaxBusters.ciac.org/](http://HoaxBusters.ciac.org/)


**Computer Virus Prevalence Survey.** ICSA (formerly the National Computer Association) publishes a yearly survey of virus prevalence, has certification schemes for anti-virus and other security software, papers, discussion groups, and so on. [http://www.icsa.net/](http://www.icsa.net/)


The Virus Creation Labs: A Journey into the Underground. George Smith. American Eagle Publications. ISBN 0-929408-09-8. Smith’s writings have long served as a very effective antidote to some of the self-righteous pomposity found in some corners of the security establishment. His book is an interesting, journalistic, alternative view across the virus/anti-virus divide.

European Institute for Computer Anti-Virus Research. Despite its name, EICAR is not exclusively focused on viruses, and its members include representatives of academia and business. Not all of them are European, either. http://www.eicar.org/

Future Trends in Virus Writing. Vesselin Bontchev. Virus Test Center. University of Hamburg. Crystal-ball gazing is a mug’s game, and even the redoubtable Dr. Bontchev didn’t get every predictive detail right. However, as a thumbnail guide to virus issues from a major authority in the field, this merits close attention. http://www.virusbtn.com/OtherPapers/Trends/

SherpaSoft Web page. FAQs including the VIRUS-L FAQ, the alt.comp.virus FAQ, the Viruses and the Macintosh FAQ, an email abuse FAQ, other papers, resources, and links. http://www.sherpasoft.org.uk

Network Associates. Requests for the Dr. Solomon’s Virus Encyclopaedia cited in the previous edition are redirected to the NAI equivalent at http://vil.nai.com/vil/default.asp.


Virus Bulletin. The only monthly magazine I know of entirely devoted to virus management. http://www.virusbtn.com/
Viruses Revealed: Understanding and Countering Malicious Software. David Harley, Robert Slade, and Urs Gattiker. Osborne. ISBN-0-17-213090-3. (For publication, 3rd quarter 2001.) It’s hardly appropriate for me to advertise my own book here. However, this one is almost unique among recent books on the subject, in that it’s actually written by acknowledged experts in the field. Covers a wide range of issues (technology, history, corporate protection, social issues, ethics). Check the Web site at http://www.viruses-revealed.org.uk.

The Enterprise Anti-Virus Book. Robert Vibert. Segura Solutions Inc. ISBN-0-9687464-0-3. This is the other recent book on the subject written by an expert in the field. The author is a seasoned professional with years of experience in the design and implementation of enterprise anti-virus solutions, and his book focuses on these aspects. Not the book with all the answers, but something arguably more important: the book with just about all the questions. http://www.segurasolutions.com/book.htm.


WildList Organization International. The authoritative source of information on which viruses are known to be in the wild. An essential resource for anti-virus software certification authorities, researchers, and so on. http://www.wildlist.org/

Most anti-virus vendors have virus information databases and other resources, as well as information specific to their products. The following sites are generally dependable (but none are infallible). Precise URLs aren’t given, as such pages move about a lot.

http://www.sophos.com/
http://www.nai.com/
http://www.symantec.com/
http://www.f-secure.com/
http://www.viruslist.com/

Summary

This chapter can only give you an overview of the virus problem. If you have the misfortune to be a systems or network administrator responsible for protecting your customers from malicious software, you will need to do some serious research into virus and anti-virus technology, and I recommend that you take advantage of the information resources listed in this chapter. If you’re an administrator or manager, you certainly can’t afford to rely on vendor sales executives or consultants to make all the decisions for you. More often than not, these people are better acquainted with the interface of their product range than with its real-world application to real-world virus management problems.
For your delectation, we offer some guidelines that should make your computing life safer.

- Check all warnings and alerts with your IT department. If you are a manager or administrator, make sure that there is a known policy by which only authorized personnel can pass on alerts. This cuts down on panic, curbs dissemination of hoaxes and other misinformation, and reduces the risk of inappropriate action that might be worse than no action.
- Don't trust attachments. The sender might have no malicious intent, but he might not be keeping his anti-virus software up-to-date either.
- Remember that worm victims don't usually know that they've sent you an infected attachment. There is no such thing as a trusted account. If someone sends you an attachment, especially if there's no obvious reason they should, confirm with them that they did so knowingly.
- Use anti-virus software and keep it updated. However, don't assume that using the latest updates makes you invulnerable.
- If your environment allows it, disable the Windows Scripting Host. For a good summary of the process, across platforms, see http://www.sophos.com/support/faqs/wsh.html.
- If you use macro-virus-friendly applications like Word, ensure that macros are not enabled by default. Recent versions of Office allow macros in a document to be disabled as a default option. If you receive a document with macros from a trusted source, ask for verification. But don't trust this option absolutely.
- Disable default booting from diskette in CMOS. (This blocks infection from pure boot sector viruses.)
- Keep your browser, mail client, macro-friendly applications, and other vulnerable applications up-to-date with the latest patches.
- Back up, back up, back up.