Norman Book on Computer Viruses

Peace of Mind

Norman is one of the world's leading companies within the field of data security. With products for virus control, spam control, email control, download control, personal firewall, encryption, data recovery, certified data erasure and computer forensics, the company plays an important role in the data industry.

NORMAN®

www.norman.com

Norman ASA is not liable for any other form of loss or damage arising from use of the documentation or from errors or deficiencies therein, including but not limited to loss of earnings.

In particular, and without the limitations imposed by the licensing agreement with regard to any special use or purpose, Norman ASA will in no event be liable for loss of profits or other commercial damage including but not limited to incidental or consequential damages.

Contributors to *The Norman Book on Viruses*:

Snorre Fagerland, Sylvia Moon, Kenneth Walls, Carl Bretteville

Edited by Yngve Ness

Last revised February 2003.

Contents

Introduction

It's hard to believe that the first IBM personal computer (PC) was introduced in August, 1981. In the beginning they were used by a small group of people. Today, however, we can't imagine life without them, both at work and at home. Look around your office when the electricity goes out, and you'll see people standing around talking because they feel they can't get any work done without their computers.

We have become dependent on these machines and the information stored within. As the importance of a "thing" rises, it becomes equally as important, if not more, to secure it. (How many of you have alarm systems in your cars?)

A large portion of modern computing life is to secure the information that we are creating and processing. There are many aspects of information security, ranging from physical access to ensuring that the information has not been changed in any way.

"The only secure computer is one that's unplugged, locked in a safe, and buried 20 feet under the ground in a secret location...and I'm not even too sure about that one."
Attributed Dennis Huges, FBI

One of the most high-profile threats to information integrity is the computer virus. Surprisingly, PC viruses have been around for two-thirds of the IBM PC's lifetime, appearing in 1986. With global computing on the rise, computer viruses have had more visibility in the past two years. In fact, the entertainment industry has helped by illustrating the effects of viruses in movies such as "Independence Day", "The Net" and "Sneakers".

Note that computer viruses are also found on Macintoshes and other platforms, but in this book, we will focus on PC viruses. The topics we will cover are:

- what a virus is
- the evolution of the virus problem
- viruses on different operating systems
- solutions to the virus problem
- how Norman Virus Control products help

What is a virus?

The terms “computer virus” and “virus” are used very loosely in everyday conversation and have become synonymous with “trouble”.

A virus is usually not something that creates cool screen effects and enables you to hack into Pentagon. The “Launching virus” screen as seen in Hollywood movies bear no resemblance with real life viruses. In reality, a virus infection is most often invisible to the user. The machine may slow down a little. Some programs may be unstable and crash at irregular intervals, but then again that happens ever so often on clean systems too.

Still, some viruses have some sort of screen effect. The Windows virus “Marburg” fills the desktop with red circles with a white “X” inside”. A couple of viruses will make desktop icons escape the mouse cursor. Such effects are not particularly common, since they expose the existence of the virus.

In order to explain such vexing programs, we will need to look at what programs really are.

What is a program

A program is a recipe for a computer’s behavior. Now, computers do not read these things as we humans do. They cannot understand free text messages – instead they have to rely on numbers, because computers are really only glorified calculators. For example, let’s look at the instruction for “do nothing” in ordinary Intel processors (yes, there is an instruction for that) – it’s the number 144. If the number 144 is translated into binary it can be written as 10010000 – which physically means voltage on, off, off, on, off, off, off, off in wires going into the processor.

When a program is run on your computer, what happens is that the operating system, for example Windows, reads the program

from the disk, examines it and determines what kind of program it is. From there the processor is fed the numbers in the program. Modern operating systems are multitasking – that is, they can juggle around with many programs simultaneously. That is why you can have several program windows open at the same time.

What is residency

"Residency" is a term you will come across far and wide in this book. It means "active in memory". A resident program is a program that exists in the computer's memory for an extended period of time. The term was more relevant in the DOS heyday, when most programs were non-resident – i.e. they did what they were supposed to do and died. In the Windows world, however, it's fair to say that most programs are resident. They stay active until you close them.

Malware classes overview

Viruses, Worms, Trojan horses, Logic Bombs etc. are all examples of what is called malicious software programs, or malware for short.

Malware is primarily an unwanted, uninvited, potentially dangerous set of programs, but there are important distinctions among the different subtypes. The following overview defines a few of the most important categories:

Virus

Viruses require a host, and their goal is to infect other files so that the virus can "live" longer. Some viruses perform destructive actions although this is not necessarily the case. Many viruses attempt to hide from being discovered.

Remember: Viruses are simply software programs.

Replicates?

Yes. All viruses make copies of themselves, infecting boot sectors, programs, or "data files" as the opportunity arises.

Worm

A host is not required, although one in some cases may argue that a worm's host is the machine it has infected. As such, some researchers define worms as a subtype of viruses. In the beginning worms were considered to be mainly a mainframe problem. This changed after Internet became widespread; worms quickly got accustomed to the Windows operating system and started to send themselves via e-mail, IRC, and other network functions. In addition we have lately seen a re-emergence of the UNIX-based worms, which exploit security holes in the different flavours of UNIX.

Replicates?

Yes. A worm makes copies of itself as it finds the opportunity.

Trojans, backdoors, security risks

Do not require a host. The word Trojan is derived from the term "Trojan horse", and although it sometimes refers to the destructive code contained in the program, the term is more often used to refer to the entire program file. Trojans are programs that perform some unwanted action while pretending to be useful. Most trojans activate when they are run and sometimes destroy the structure of the current drive (FATs, directories, etc.), obliterating themselves in the process.

A special type is the backdoor trojan, which often does not do anything overtly destructive, but sets your computer open for remote control and unauthorized access. Unfortunately, some of the commercially available remote administration tools can be used as trojans in certain settings. Tools that do not have enough precautions against being used for malicious purposes may be detected by Norman Virus Control as a "security risk".

Replicates?

No.

Denial-of-service tools, nukers, mail bombers

These categories are software weapons. They do not pose any direct threat to the computer where they are installed, but are designed to disrupt the operation of other networked computers.

Sometimes these weapons can be installed silently to be used from unsuspecting users' computers, and in this respect such tools also fit the description trojans.

Denial-of-service (DOS) tools are used to bombard other computers with connection attempts to such a degree that the computer that is under attack cannot handle the traffic load, and legitimate requests are neglected. A special case of denial-of-service is the so-called "Distributed Denial-of-Service", or DDOS. DDOS occurs when several machines start a coordinated attack against the same target.

Nukers send malformed network requests to try to confuse the attacked machine and cause a crash.

Mail bombers are pretty self-explanatory – they are used to annoy people by filling up their mailbox.

Replicates?

No. None of these replicate by themselves, but it is possible to combine viruses and some of these attack methods.

Hacking tools, virus creation kits

There are quite a few people who engage in shady activities, and there are plenty of tools available to help them.

Hacking, which unfortunately has come to mean gaining unauthorized access to remote computers, has been a problem since long before the first computer viruses emerged. There are a number of obtainable tools that can be used to gain knowledge about and break into other computers.

There are also quite a few programs that in turn can create computer viruses. These programs are made as help for would-be virus authors, and is one of the main reasons for the current virus situation. These tools are so easy to use that persons with no programming skills can make new viruses. Such programs are called virus generators, virus creation kits, or just kits.

Replicates?

No. A virus creation kit creates new viruses, but does not replicate by itself.

Bugs, logic bombs, time bombs

These are program malfunctions. You can say that they require a host — programmers cannot write a bug without at the same time writing other code — although it's fair to say that most programmers do not intentionally write bugs. Logic bombs and time bombs are malfunctions intentionally inserted in otherwise "good" code.

Replicates?

No. This code generally has better things to do than making copies of itself. Logic bombs and time bombs wish to remain hidden, with only their effects being visible. Bugs do just about everything except make more bugs.

Hoax

A hoax is a chain letter, typically sent over e-mail, which carries false warnings about viruses or trojans. This causes well-meaning users to send the warning on in the belief that they are doing other users a favor. Often such warnings apparently stem from well-known companies and organizations, but this is not the case. Hoaxes may also contain other messages that are supposed to trick people to send the message on, for example they will offer money or a cell phone as a reward for forwarding the message to friends.

If you receive a warning about a virus, do not pass the warning on to other users! This rule applies even if the virus actually does exist, and applies doubly if the warning asks to be sent on. It whips up anxiety and increases the workload.

Replicates?

No, not by itself. They trick the user into making copies instead.

Virus/worm types overview

When speaking about viruses and worms, we normally speak about these main categories:

Binary file virus and worms

File viruses infect executables (program files). They are able to infect over networks. Normally these executables and viruses consist of instructions that are created for easy machine interpretation, so-called machine code. To the untrained eye, machine code is incomprehensible, as it is basically a row of numbers to be queued into the processor. File worms are also written in machine code, but instead of infecting other files, worms focus on spreading to other machines. See page 24 for details.

Binary stream worms

CodeRed is a binary stream worm that employs the network.

Stream worms is a group of network spreading worms that never manifest themselves as files. Instead, they travel from computer to computer just as pieces of code that exist only in memory. The most renowned of this group is the *CodeRed* series of worms that spread between IIS systems. See page 26 for details.

Script file virus and worms

A script virus is technically a file virus, but script viruses are written as pure text and thus easily readable for everybody. Since computers cannot understand text instructions directly, the text first has to be translated from text to machine code. This procedure is called “interpretation”, and is performed by separate programs on the computer. For example, Visual Basic Script (VBS) is interpreted by the program WSCRIPT.EXE, and old DOS batch language (BAT) is interpreted by COMMAND.COM. Script viruses infect other script files, but even more common are the script worms that travel from machine to machine, preferably over e-mail. See page 27 for details.

Macro virus

Macro viruses infect data files, or files that normally are perceived as data files, like documents and spreadsheets. Many “data file types” have the possibility to include instructions along with the normal content – f.ex. Microsoft Word files can contain instructions that tells Word how to show a particular document,

or instructions that tells Windows to do certain actions. Just about anything that you can do with ordinary programs on a computer can be done through such so-called macro instructions.

Macro viruses are among the most common viruses today. These are able to infect over networks. See page 25 for details.

Boot virus

Boot viruses infect boot sectors of hard drives and floppy disks. These are **not** able to infect over networks.

Multipartite virus

Multipartite viruses infect both executable files and boot sectors, or executable files and data files. These are able to infect over networks.

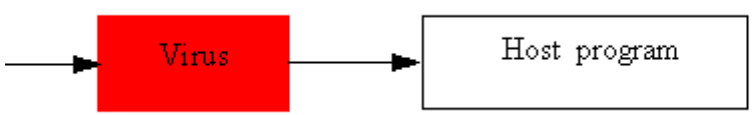
You may also have come across terms like “polymorphic”, “stealth”, and “encrypted”. These are not types of viruses per se, but rather methods that viruses use to disguise themselves from anti-virus products.

The next sections describe binary, script, macro, and boot viruses more thoroughly.

Binary file virus

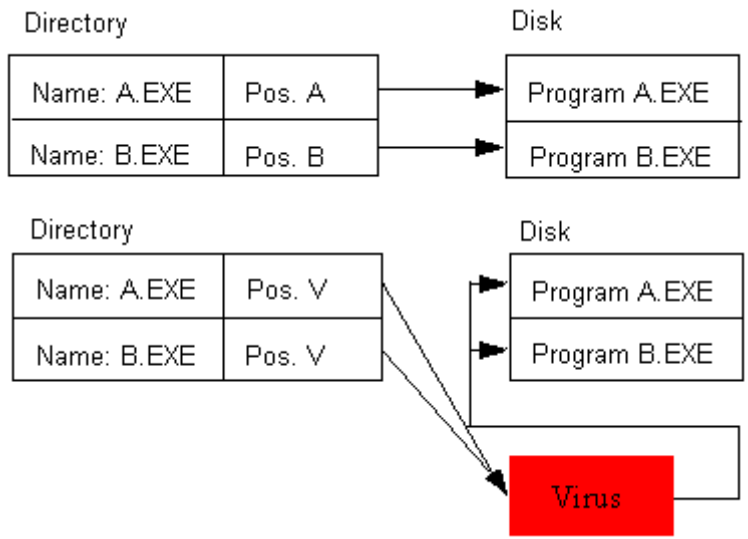
A file virus attaches itself to a program file (the host) and uses different techniques in order to infect other program files.

There are several basic techniques for infecting an executable file: companion, link, overwrite, insert, prepend, append, and others.

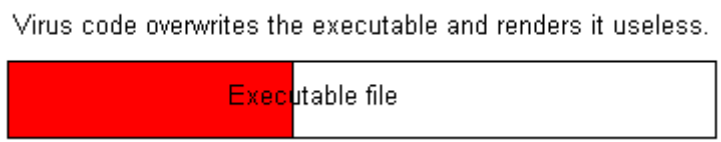


A companion virus does not modify its host directly. Instead it maneuvers the operating system to execute itself instead of the host file. Sometimes this is done by renaming the host file into some other name, and then grant the virus file the name of the

original program. Or the virus infects an .EXE file by creating a .COM file with the same name in the same directory. DOS will always execute a .COM file first if only the program name is given, so if you type “EDIT” on a DOS prompt, and there is an EDIT.COM and EDIT.EXE in the same directory, the EDIT.COM is executed.

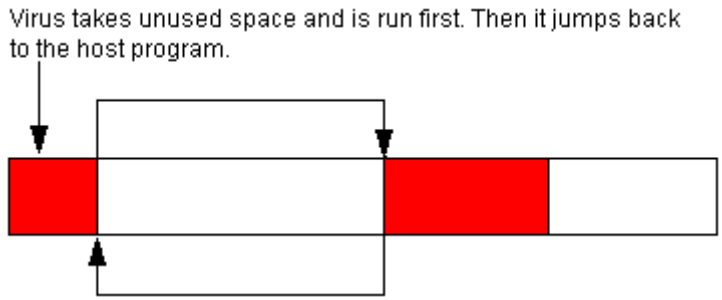


A link virus makes changes in the low-level workings of the file system, so that program names do no longer point to the original program, but to a copy of the virus. It makes it possible to have only one instance of the virus, which all program names point to.

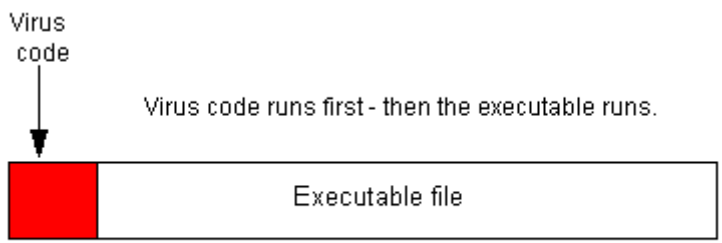


An overwriting virus places itself at the beginning of the program, directly over the original program code, so the program is now damaged. When you try to run this program, nothing happens except for the virus infecting another file.

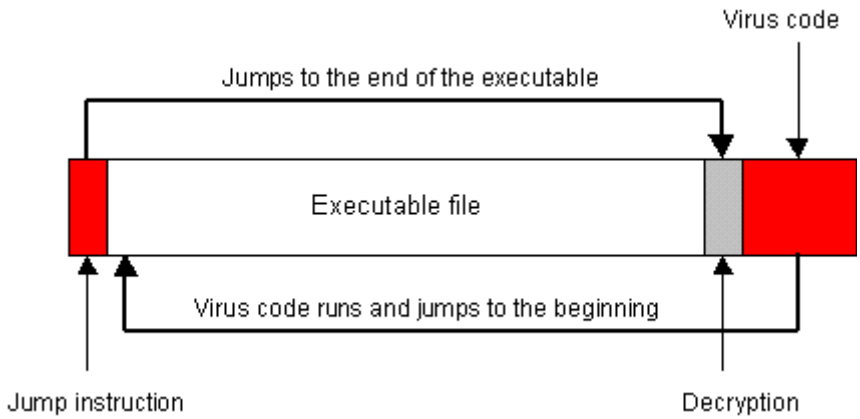
Such viruses are easily apprehended and destroyed by users and user support staff, so they actually spread very poorly in the wild. You have almost no chance of ever getting an overwriting virus in your machine.



An inserting virus copies itself into the host program. Programs sometimes contain areas that are not used, and viruses can find and insert themselves into such areas. The virus can also be designed to move a large chunk of the host file somewhere else and simply occupy the vacant space.



The pure prepending virus may simply place all of its code at the top of your original program. When you run a program infected by a prepending file virus, the virus code runs first, and then your original program runs.



An appending virus places a “jump” at the beginning of the program file, moves the original beginning of the file to the end of the file, and places itself between what was originally the end of the file and what was originally at the beginning of the file. When you try to run this program, the “jump” calls the virus, and the virus runs. The virus then moves the original beginning of the file back to its normal position and then lets your program run.

This was a brief overview of how a virus attaches itself to a program file. It uses different techniques in order to infect. Many file viruses go memory-resident so that they can monitor all actions and infect other program files as they are run or otherwise accessed. Other file viruses infect by “direct action”, which means that they infect other program files right away, without going memory resident. Under Windows this distinction becomes blurred, as many viruses are resident and “direct action”.

Several other methods exist.

Script file viruses

Script file viruses are a not really a new class of viruses, but has only quite recently evolved into a major threat. As mentioned, scripts are pure text instructions that are interpreted by some program. There are quite a few scripting languages:

Visual Basic Script: These scripts are normally found as separate *.VBS files or inserted into web pages. VB scripts have a

functionality that is a subset of the Microsoft Visual Basic language, and it's expandable by importing functions from other programs. For example, many of Microsoft Words' functions can be used through VB script.

JavaScript: The scripting language that was introduced by Sun Microsystems alongside the development of the HTML standard. Standard JavaScript is usually quite safe, as it does not affect the file system. You'll normally find JavaScript on web pages.

JScript: The Microsoft version of JavaScript. It is about as flexible and expandable (and unsafe) as Visual Basic Script. JScript is found in *.JS files or on web pages.

DOS BAT language: When you wanted venerable old DOS to do something, you used to type the command on the command line. E.g. displaying files in a directory was performed by typing DIR <Enter>.

However, sometimes you instructed DOS to perform certain tasks when you weren't around to type in the commands. The BAT (batch) language was developed for this purpose: enter the commands into a text file and then type the filename on the command line to give DOS a set of commands to process. Such files are always called batch files and have the extension *.BAT.

UNIX shell script: This is similar to DOS batch language only it was developed for the different varieties of UNIX. UNIX has very wide set of commands available from the command line, so shell scripts are quite powerful and can do a lot of different things.

IRC scripts: Internet Relay Chat is a chat system for the Internet. Chat systems can be scripted to perform certain tasks automatically, like sending a greeting to someone who just joined the chat room. However, the scripts also support sending of files, and many worms and viruses spread over IRC. Known IRC programs that have been exploited are the popular mIRC, pIRCH and VIRC clients.

Other scripting languages: Many other scripting languages exist. Corel Draw, Visual Foxpro, SuperLogo, InstallShield etc. can be scripted and have been used for malicious purposes.

Macro virus

Since the introduction of the first macro virus in August 1995 and until quite recently, this virus type has been the fastest growing category. The first time we discussed this phenomenon in this publication, in January 1997, the number of known macro viruses was 100. Some four years later, Norman had identified more than 8,000 macro viruses, and the number is still growing. Even though bigger threats have emerged since the heyday of the macro virus, the problem still exists and cannot be ignored.

How it works

Traditional file viruses do not attempt to infect data files, for data files are not an ideal ground for replication. That is, one does not "run" a data file — one "reads" and "edits" a data file. However, in the past few years, organizations have been building upon open systems, in which data is shared more readily. This in turn means that there is little security. Macro viruses take advantage of the fact that many applications now contain **macro programming languages**. These languages allow users (and virus authors) more flexibility and power within the application than ever before, and in fact convert what used to be data files into programs. Often macro viruses are not detected early enough because many users are not familiar with the nuances of macros. As a result, macro viruses have an infection rate much higher than traditional file and boot viruses.

In the beginning, the most targeted macro programming language was WordBasic, the language within early versions of Microsoft Word. Later, the predominant macro programming language used in viruses became Visual Basic for Applications, or VBA. This programming language is shared by a lot of applications – Word, Excel, Access, PowerPoint, Project, Visio and many others.

Why it's such a risk

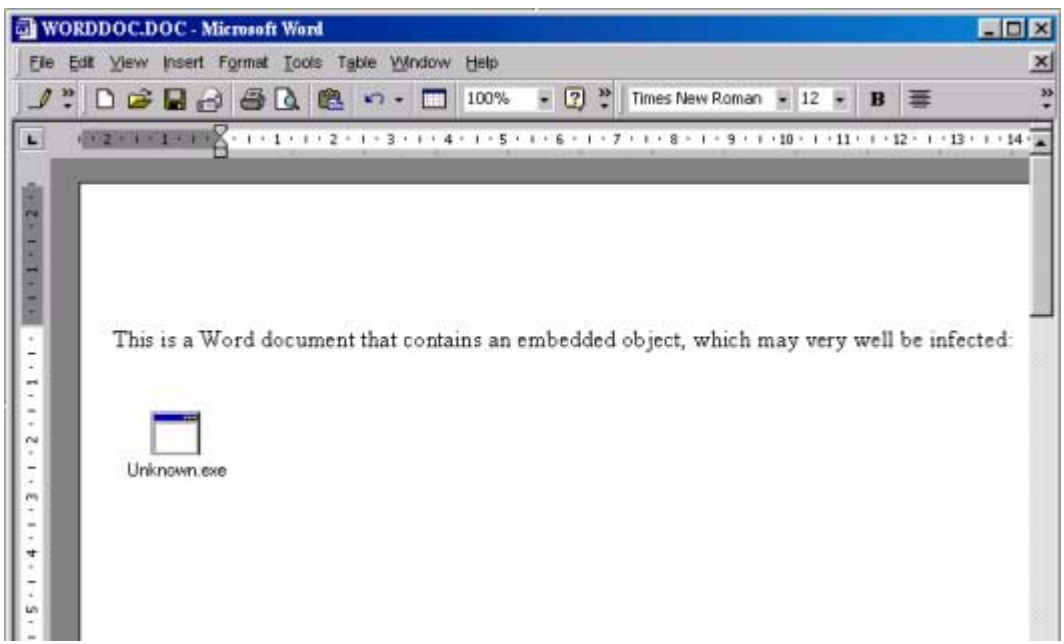
Since data files, and in particular documents, are shared more frequently than executable (program) files, the security threat posed by macro viruses is very real. VBA is also a very powerful

programming language which can be used to control almost anything on the computer.

Some macro viruses contain destructive code and some even create and execute traditional file and boot viruses. While traditional file and boot viruses affect the operation of a machine, macro viruses affect the quality and reliability of **information** contained within data files.

Embedding and linking

The open systems in many of Microsoft's applications utilize OLE in order to combine different data types. You can **embed** an object such as a bitmap or an executable within a Word document. Embedding an object means that any edits to the object will not be reflected in any other copies of the object. You can also **link** an object such as an Excel spreadsheet to a Word document. Linking an object means that you may edit the object in either its source application or from within the application to which it is linked, and all copies of the object will be updated.



Microsoft Office products have the ability to embed and link objects. In addition, they have the ability to be embedded and linked in other applications. Thus there is a risk that you can run a Word macro virus from within another application, or an infected executable file from within Microsoft Office.

MS Word

By far the most common macro viruses have been written for Microsoft Word. This is partly because Word was the first application to be riddled with macro viruses, but also because Word documents are exchanged more frequently than any other file type. The first Word macro virus, WM/Concept.A was created in the middle of 1995 and was quickly one of the most widespread viruses in the world - in spite of the fact that it contained no mail-spreading functionality that became popular towards the turn of the century.

MS Excel

It did not take long after the first Word Macro virus before the first Excel virus appeared: *XM/Laroux.A*. This was an event that was expected, as the techniques necessary to create such a virus are the same as for Word macro viruses.

The difference between the first viruses for Word and Excel is that viruses for Word were written in WordBasic, whereas viruses for Excel were written in VBA3 (Visual Basic for Applications version 3). The format was different, and the macros were not stored inside the spreadsheet (viruses for Word 6/7 are stored in the Word document), but in separate streams. This technique complicated detection, identification, and removal.

Macro viruses for Excel can sometimes pose a bigger threat than Word viruses, because of the possible practical implications. Imagine that a macro virus for Excel multiplies a certain cell by a factor 10 and that this particular cell specifies your salary. Definitely not the end of the world, but what if this cell was *divided* by 10?

These are minor inconveniences compared to similar changes to calculation formulas for estimating the strength of concrete for a

skyscraper. Spreadsheets are often large, and anomalies are not easily recognized.

Office 97, Office 2000, Office XP

The introduction of Office 97 also included changed formats for almost all programs in the suite, but at least the changes were consistent. Both Excel and Word are using VBA5, based on VBA3 with many extensions.

VBA5 is not compatible with WordBasic, which should indicate that macro viruses written for previous versions of Word would not affect Word 8.0 in Office 97.

However, Microsoft initiated a WordBasic to VBA5, and a VBA3 to VBA5 conversion to upgrade existing macros to the new formats.

The same thing happened with document conversion from VBA5 to VBA6, which is the VBA version used by the newer Office formats Office 2000 and Office XP.

Consequently, macro *viruses* for previous versions of Word and Excel can also be ‘upgraded’. Not every virus will work after the conversion, but we know that quite a few do.

Boot viruses

Boot viruses infect System Boot Sectors (SBS) and Master Boot Sectors (MBS).

The MBS is located on all physical hard drives. It contains, among other data, information about the partition table (information about how a physical disk is divided into logical disks), and a short program that can interpret the partition information to find out where the SBS is located. The MBS is operating system independent. The SBS contains, among other data, a program whose purpose is to find and run an operating system.

Because floppy diskettes are exchanged more frequently than program files boot viruses are able to propagate more effectively than file viruses.

Refer to “Viruses on different operating systems” on page 36 for more information.

The booting process

To understand boot viruses, it is necessary to understand the booting process.

The BIOS (Basic Input/Output System), which controls the booting process, is initiated as soon as the power is switched on.

The next process that runs is called POST (Power On Self Test). It ensures that the computer is in working order. One POST function that all users will recognize is the display that counts the amount of RAM (Random Access Memory) in the machine.

POST’s final act is to kick off the booting process. The first task is to determine whether or not there is a diskette in the floppy drive. If there is, the System Boot Sector on the floppy is read, and the machine attempts to boot from it.

If the diskette is not bootable (see below for more details), then you will see the following message on the screen:

```
Non system-disk or disk error.

Replace and strike any key when ready.
```

This text is written to screen by a small program that exists on the SBS of the diskette if it is non-bootable.

Normally, however, no diskette is present, and the Master Boot Sector on the hard drive is read. Then the System Boot Sector on the boot partition of the hard drive is read, and it will start the operating system.

This process remains the same on machines running DOS, Windows, Windows 9x/ME, Windows NT/2000, Linux and OS/2. The differences appear when the operating systems themselves are loaded.

A bootable diskette

When a floppy disk is formatted, a System Boot Sector is created. The diskette can have two functions: contain program files and data files and/or be a bootable diskette.

A bootable diskette is one that can be used to bypass the booting process on the hard drive. Instead, the boot process runs from the diskette.

To create a bootable diskette, you must either format the diskette with the “system” option (/S) or you must use DOS’ SYS command on the diskette.

A formatted diskette always has a System Boot Sector regardless of whether or not the diskette is bootable. The SBS happens to be the place that a boot virus calls home, so any formatted diskette that you have can potentially be infected with a boot virus.

How a boot virus infects

If a diskette is left in the drive, and the machine is set up to boot from floppy, then the SBS of the diskette will be read. If the SBS contains a boot virus, the boot virus will become active, go memory resident, infect the system areas of the hard drive, and attempt to infect other write-enabled diskettes that are accessed.

Users tend to leave floppies in the drive when they turn their machines off in the afternoon and then forget about them when they turn their machine on in the morning. Thus, many people boot from their floppies without even realizing it. As a result, boot viruses were the most common viruses for a long period.

Special case: The CIH virus (W95/CIH.1003.A)

Type:	**Binary file virus**
Infects:	**Windows 32-bit executable programs**

Until 26 April 1998 it was true that viruses could inflict serious damages on software, but not on hardware. On this particular day, the virus *W95/CIH.1003.A* struck for the first time. The victims had to replace the flash BIOS chip, and even (especially on laptops) the PC’s motherboard. In the months to follow, the CIH virus was reported in the wild from most parts of the world. It now exists in a heap of different variants, some triggering on the 26th of the month.

The CIH virus infects executable files under Windows 95/98 in a very covert manner. For example, infected files do not increase their length. Often, an unexpected change of file length for a binary file is a sign of virus activity.

The technical description of what actions the virus take when it triggers is beyond the scope of this book. Simply put, when the flash BIOS is reprogrammed by the CIH virus, the PC is lobotomized and forgets its internal language. When this happens, there is no cure but replace this piece of hardware. The virus can also partially overwrite the hard disk and render it useless.

The CIH virus is a reminder that virus writers sometimes have detailed information on undocumented internal procedures deep down inside the operating systems. When they use this knowledge to write almost bug-free viruses as nasty as CIH, the need for adequate virus protection is a must.

It's common knowledge that many CIH victims were infected after downloading files from Internet gaming sites. Therefore, we think it's appropriate to remind all surfers on the web about the everyday perils out there.

However, we do recommend that you use antivirus software with a qualified updating mechanism. The days when you could get away with quarterly updates of your virus protection software are definitely over.

Special case: The Melissa virus (W97M/Melissa.A@mm)

Type: **Macro virus**
Infects: **Word 97 and Word 2000 documents**

Late March '99, we received reports that a document called `password.doc` seemed to be sending itself via e-mail. The number of mails sent was so oversized that several e-mail servers were unable to handle the traffic load and crashed. We were looking at the first massmailing e-mail virus. The document contained a macro virus called *W97M/Melissa.A*, which not only spread itself to the users documents, but also sent copies of itself

to the fifty first entries in the user's address book. A delicate side effect was the involuntary distribution of documents to others. Word documents quite often contain sensitive or semi-sensitive information, so the damage because of this spread pattern may have been substantial.

Special case: The CodeRed worm (NT/CodeRed.A)

Type: **Binary stream worm**
Infects: **Windows 2000 machines running IIS**

Until August 2001, the presence of malicious software was generally in *files*. Programs are indeed present in memory when they run, but since they normally have to be started from file, antivirus tools have traditionally focused on detecting malware in files.

This perception changed with *CodeRed*. *CodeRed* is a program, almost like any other program. However, there's a striking difference: it never manifests itself on the hard disk. This makes it difficult for many antivirus tools to see the worm at all. It moves into the computer from the network as a kind of data stream. The receiving machine perceives the event as a program and executes it from its place in memory. Clearly, this is not something that should be allowed to happen, but *CodeRed* exploits a known bug or malfunction in the Internet Information Server in order to accomplish this.

Once running on the local machine, it will again attempt to contact other machines in order to disseminate. The worm is easily stopped by patching the IIS server to eliminate the bug that the worm employs, but other worms may exploit other security holes in the future.

Special case: The LoveLetter virus (VBS/LoveLetter.A@mm)

Type: **Script file virus**
Infects: **VBScript files**

In the beginning of May 2000 the phones started to ring at antivirus vendors' offices everywhere. "Do you provide detection for the I love you virus?" The question was followed by a roaring silence before the hesitant answer: "What I love you virus?"

This virus epidemic demonstrated a very nasty feature of modern e-mail viruses and worms: they are fast. Before the antivirus industry had seen the virus, users all over the world had received copies in their mail boxes. *LoveLetter* is a textbook example for the importance of frequent antivirus updates and the importance of using good heuristics ("unknown virus detection").

Sidenote: *VBS/LoveLetter.A* is a real virus, as it overwrites VB Script files with itself. However, the wormish functionality is what you see. The distinguishing feature of fast propagation is why *LoveLetter,* not surprisingly, sometimes is referred to as a worm.

Special case: Nimda (W32/Nimda.A@mm)

Type: **Binary executable virus**
Infects: **Windows 9x/Me/NT/2000/XP machines**

This is a very interesting case. This virus employs a conglomerate of techniques to spread. However, the most intriguing part is the Internet spreading technique.

Before *Nimda* appeared, a few virus authors had experimented with infecting users from web servers. The problem from the virus writer's point of view, was that this created a single point of attack. As soon as the offending web page was identified, it would be shut down, thus effectively killing the virus. Other

worms, like *CodeRed*, have tried to break into or infect web servers. *Nimda* was the first to combine these effects.

When *Nimda* is started on a machine (remeber, it's an executable), it will start to look for web servers. This is done by generating random Internet addresses and attempting to connect to these. When a connection is established, *Nimda* hacks the web server effectively, and copy itself over. When it runs on the web server, it modifies web pages in ways that infect surfers on these pages. This two-way infection creates a two-sided problem: users are not used to consider a web page as a possible source for infections, and there is no longer any single web server that can be shut down to stop the virus. The web spreading mechanism is more mischievous considering that the user does not necessarily need to click on any attachment to be infected - *Nimda* exploits a programming error in certain versions of Internet Explorer to be executed automatically as the web page (or mail body) is viewed.

Nimda possesses other spreading mechanisms that are more commonplace—it infects binary executable files, it spreads over the local network shares, and it e-mails itself around.

Special case: Sircam (W32/Sircam.A@mm)

Type:	**Binary executable worm**
Infects:	**Windows 9x/Me/NT/2000/XP machines**

This middling worm was not expected to be much of a threat at the face of it. It spreads by ordinary e-mail, and it has a semi-fixed mail body text, so it's easy to recognize. However, this worm has proved to be very persistent and hard to uproot. Why is that?

The most distinguishing feature with *Sircam* is the trick it does when it sends itself via e-mail. It finds a random document, spreadsheet, or zip file on the hard disk, and simply append it to itself. The resulting file has the name of the original document, but adds an extension. For example, `mydocument.doc` becomes `mydocument.doc.exe`. If you are not paying close

attention when you read the mail, the file may easily be mistaken for an original uninfected file.

A grim consequence of this trick is that personal and/or confidential information may be sent out unintentionally. The worm doesn't care whether it sends an innocent memo or business-sensitive financial information. We have experienced that very personal letters from infected users have arrived in our mailbox. This demonstrates the hazard of today's computing—even your own hard disk can be exposed for everyone to see. Sensitive information should always be encrypted and preferably stored on a removable media such as floppies or CDs.

Special case: Klez (W32/Klez.H@mm)

Type:	**Binary executable virus**
Infects:	**Windows 9x/Me/NT/2000/XP machines**

The *Klez* family consists of about ten different variants with slightly different properties. Only two of these have become really widespread—the E and H variants. Of these, the H variant is totally dominant and is at the time of this writing (December, 2002) by far the most widespread e-mail capable virus worldwide.

It arrives in mails of very unpredictable appearances—different subjects, body texts and attachment file names. The texts used for these fields are randomly picked from files on the user's hard disk and from word lists inside the virus itself. The most insidious feature, however, is that the "FROM:" address is faked. *Klez* collects e-mail addresses from various places on the infected machine and use them randomly for both "TO:" and "FROM:" addresses. So, the next time you receive a *Klez*-infected e-mail, remember that the (seemingly) sender is most likely not infected.

Klez also spreads through a number of other mechanisms. It copies itself over network shares, both as a standalone executable file and as a RAR archive with itself inside. It also

sometimes acts as a virus— it stores legitimate programs in compressed files and takes their place.

It doesn't stop there; it also plants **another unrelated virus** on machines which it infects. This virus, called *W32/Elkern.C*, is not capable of spreading over mail by itself. *Elkern* can be very difficult to remove because of its ability to attach to already running processes in memory. You may clean up the entire hard disk, but the viral process may lurk inside legitimate running processes and continue to infect.

Special case: Friends Greetings (W32/FriendGreet)

Type:	**Binary executable, commercial**
Infects:	**Windows 9x/Me/NT/2000/XP machines**

This is not a virus or worm, even though it has "worm-like" capabilities.

It arrives as an e-mail that looks like this:

Bill, Bob has sent you a greeting card -- a postcard from Friend-Greetings.com. You can pick up your greeting card at Friend-Greetings.com by clicking on the link below.

http://www.friend-greeting.com/pickup.html?code= <removed> &id=<removed>

Message:

Bill,

I just sent you a greeting card - please pick it up.

Bob

Now, this looks innocent enough. However, if you click on the link, you will be prompted to download and install software to view the "postcard". During the installation of this software, you will be informed via license information screens that the software will mail such cards **to everyone in your address**

book. Many users accept this, either because they don't read the license information, or because they don't care. In this manner the software spreadsfar because it has permission by the user to do so.

It is advisable to read all license agreements carefully so that you do not get inadvertently caught in such schemes.

Predictions for the future

Norman expects that macro viruses still will represent a serious threat to data security, even though there is reason to believe that the growth rate will flatten.

Some scripting languages have made it abundantly clear that they have the potential to cause a lot of issues security wise, and new scripting languages show up quite often. Script viruses will remain a problem for the foreseeable future.

The last couple of years have also indicated a reemergence of the binary file virus and worm. This group of malware was actually losing terrain in the early and mid-nineties, because the exchange of executable files became less and less common with the introduction of Windows, while e-mail systems at the time were not sophisticated enough to facilitate for e-mail worms. Internet and modern Windows software development tools have changed all that. Now there is an active exchange of files over e-mail and via other network functions, and with them, viruses. Binary file viruses will be around as long as programs are exchanged freely.

Other areas of computer usage are changing. The Internet as "environment" becomes more complex and interconnected, and this will create new opportunities for malicious software. Exactly how the terrorists of the Internet will exploit the new reality is hard, if not impossible to predict. It's not unlikely that next really serious virus epidemic will emerge along one of these unpredictable new trails. We have already seen from the *CodeRed* incident that part of the Internet infrastructure is vulnerable and will play a role in future outbreaks.

Norman will continue to keep abreast of the virus issue. We have some of the world's most prominent experts working to monitor the evolution of computer security, and we seek to be prepared to intercept new threats before the damage is done.

How many viruses are there...

Anti-virus vendors are asked this question all the time. The answer is difficult for several reasons:

1. There is no central organization that counts the number of viruses.
2. New viruses appear every day. Some experts say that the growth of new viruses is exponential and others say that it's quadratic. If we were able to count them all, then the count would be valid only for a short period of time such as a day.
3. Often we find that many variants are made based upon one virus, and often there is disagreement among the virus research community on the definition of "variant".
4. There is no standard naming convention for viruses, and as a result it is possible to have several different names for the same virus.

   This brings up the question of how viruses get their names. Sometimes the virus author puts text into the virus that indicates a name for the virus or for him/herself (e.g., The xxx virus is here; Greetings from yyy). But most of the time, names are given by people who discover them. Different methods are used, such as: estimates of place of origin or place of detection (e.g., the Lehigh virus), number of bytes that the virus adds to files, what the virus does, and so on.

With those caveats in mind, Norman Virus Control products detect over 56,000 virus variants as of this writing (December, 2002).

...and does it matter?

To the ordinary user, it's immaterial if the number of viruses is this or that, as long as your anti-virus software keeps your machine virus free. The number of known viruses does not really reflect what is going on in the virus makers' world. Statistics are

sometimes useful as a visible manifestation of the evolution of computer viruses, for example.

The computer virus problem is best evaluated by analyzing the nature of the individual virus and looking at what they do, rather than keeping track of how many there are. This may have been the message from the virus maker(s) who sent some 14,000 brand new viruses to anti-virus vendors all over the world (fall of 1998). All of these viruses had been generated automatically, and they were therefore not technically sublime. In fact, most of these viruses could be detected by heuristic methods. Nevertheless the *number* of viruses detected by our virus definition files almost doubled overnight, while the overall virus threat remained unchanged.

In the wild viruses

Although virus researchers know of thousands of viruses, you need not worry about all of them. Of those thousands, most of them exist only in research labs, and the remaining handful are actually seen in homes and organizations around the world.

As a result, virus researchers group viruses into two categories: “in the wild” and “in the zoo”, sometimes referred to as “ITW” and “ITZ” respectively.

Viruses that are “in the wild” have been seen outside the research labs. In the wild viruses comprise about 10% of the viruses that we know about, and it is these viruses that you and your organization should concern yourselves with.

If you are interested in more details, contact your nearest dealer or Norman directly.

Corporations and single users need to protect themselves by frequent updates of their virus control tools. This in turn involves the antivirus industry to constantly update and distribute definition files. A definition file holds the virus signatures (fingerprints of known viruses) and is used by the scanning engine to detect and remove computer viruses.

Any virus scanner is only as effective as its most recent update, so obtaining frequent virus signature updates is critical to maintaining a secure computing environment.

Norman Virus Control version 5 or higher makes this easy for you by downloading and installing updates from Norman automatically.

The evolution of the virus problem

In the beginning, computers were not connected together very well, and computer viruses spread extremely slowly. Files were transmitted via BBSs (bulletin board systems) or on diskette. As a result, the transmission of infected files and boot sectors was geographically limited.

However, as soon as connectivity increased, mostly by the use of computers in the workplace, the boundaries of computer viruses widened. First there was the local area network (LAN), then there was the wide area network (WAN), and now there is the Internet. The extensive use of e-mail has also contributed to the meteoric rise in the number of macro virus incidents.

We are now living in a society in which global technology has taken the forefront, and global commerce is driven by communication pathways. Computers are an integral part of this technology and so the information they contain (as well as the malicious code they unwittingly contain) also becomes global.

Consequently, it is much easier to get a virus today than it was a few years ago. However, the **types** of viruses that are common today are different than those that were common two years ago.

Steve White, Jeff Kephart, and David Chess of the IBM Thomas J. Watson Research Center have been following the evolution of viruses, and (among other things) they have concluded that the prevalence of certain types of viruses have been in part determined by changes in operating systems.[1]

In the following sections, we will discuss how viruses operate in different operating systems.

1. Steve R White, Jeffrey O Kephart and David M Chess, 'The Changing Ecology of Computer Viruses' *Proceedings of the Fifth International Virus Bulletin Conference*, Brighton, UK, 1996.

Viruses on different operating systems

Computer viruses were first created as early as around 1981, when a program called “Elk Cloner” was created for the Apple IIe computer. It appeared on some bulletin board systems, but never really spread much and was never referred to as a “virus”. Dr. Fred Cohen coined that term in 1983 when he discussed concepts and experiments with replicating programs (http://all.net/books/virus/index.html). These experiments were conducted in controlled mainframe environments.

Although the first viruses that appeared were made for operating systems such as Apple II and VAX, they did not become a real problem before they were created for MS-DOS. MS-DOS was the first operating system to become a household item, and as such the viruses for this OS had a much larger potential for spreading than viruses for other platforms. It took some years for Windows to stabilize and become popular, so viruses flourished in MS-DOS. In fact, almost all file viruses were DOS based for years after the introduction of Windows, and the first Windows generations (up to Windows 3.11 for workgroups) were actually not bothered much with viruses at all. However, the appearance of Microsoft Word on this platform and the macro viruses that came with it gave us a taste of what was to come later.

OS/2 came on the scene shortly after viruses, but OS/2 never became such a mainstream operating system as DOS. Therefore, virus writers were less likely to be running OS/2 themselves. Even if OS/2 viruses had been written frequently, they would not be as widespread as MS-DOS viruses were. As a result, there is only a handful known OS/2 viruses today.

Today, DOS is outmoded even though a DOS-like functionality still exists on Windows machines. However, in a virus setting, DOS is for all intents and purposes dead as disco.

The current operating systems nowadays are the 32-bit Windows variants (Win 95/98/ME, Win NT/2000/XP) and the different UNIX variants. Let's take a look at how viruses behave on MS-DOS, Windows, OS/2, Windows 95/98, Windows NT/2000, and UNIX.

MS-DOS

Since the macro viruses that we have seen to date infect data files generated from and read by Windows applications, macro viruses are not a problem on MS-DOS-only machines.

Traditional file viruses and boot viruses prosper in MS-DOS machines because MS-DOS has no inherent security features. Viruses, therefore, have free rein to infect memory, and program files as described in "Binary file virus" on page 14.

Windows

When Windows was introduced, users had to change the way they interacted with their computers. The images on the screen were more colorful, navigating around in a program was more intuitive, and the prospect of being able to switch tasks without exiting an application was very enticing.

Since DOS ran "underneath" Windows 3.x, file viruses were able to infect machines that ran Windows, but their lifespans were cut short. In general, file viruses are able to infect Windows executables, but the executables then do not generally work properly. Impatient users would either replace the executables, or if they were frustrated enough, reinstall Windows. This was enough to cause the demise of the traditional file virus. In addition, the structure of the executables in Windows 3.x is more complicated than DOS and even Windows 9x/NT executables, and memory has in some respects better protection. Viruses under Windows 3.x therefore never became the nuisance as they have proved to be under newer Windows versions.

Macro viruses and boot viruses, however, have not suffered the same fate. To date, macro viruses have been written to target Windows applications, and therefore the presence of Windows is required. Combining the wide acceptance of Windows with the fact that macro viruses infect data files rather than program files (see “Macro virus” on page 19) has led to six macro viruses being amongst the ten most common viruses overall.

The actual booting process on a Windows machine is no different than on a DOS-only machine. Therefore, boot viruses have not been hindered by Windows, and they continue to propagate by infecting hard drives, going memory resident, and then infecting floppy disks.

OS/2

As mentioned above, OS/2 is not as widely used as Windows and other Microsoft operating systems. Because of the way that OS/2 was designed, however, it is still susceptible to non-OS/2-specific viruses.

Unlike Windows, MS-DOS does not run “underneath” OS /2. OS/2 is a powerful 32-bit operating system that supports DOS applications, Windows applications, and native OS/2 applications. In order to run DOS applications, OS/2 furnishes VDMs (virtual DOS machines). As the name suggests, VDMs “look” like DOS to DOS programs. Therefore, an infected DOS program can infect other DOS program files within that VDM, but not DOS programs in other VDMs. The newly infected DOS program file can then continue infecting other program files which might be started in VDMs in the future. So the infection path continues.

If Windows applications which include macro programming languages are run on an OS/2 machine, then the OS/2 machine is equally as susceptible to macro viruses as a Windows machine.

Again, since the booting process is the same on IBM-compatible machines prior to the operating system being loaded, boot viruses can infect OS/2 machines. OS/2 handles diskettes differently than DOS and Windows so the likelihood that the boot virus will propagate after it has infected the hard drive is lower on an OS/2 machine than on a Windows or DOS-only

machine. The risk involved is rather one of the boot virus's action on the hard drive. If the boot virus was designed to have a payload, then we can expect it to be delivered, regardless of whether it was able to infect any floppies.

OS/2 supports two file systems: FAT (file allocation table) and HPFS (high performance file system), and you may use just one or both. HPFS is more advanced and stores information in different places, so you can expect serious effects on an HPFS system from a boot virus that expected to only see FAT.

Windows 95/98/ME

Windows 95 was launched at a time when the Internet became public property. Today the world wide web is available for everybody, not just for the seasoned user. Even though the majority of PC users welcome the Internet, e-mail, and chat programs, the flip side is a huge playground for virus makers, sometimes referred to as the Internet terrorists. The widespread use of these facilities has contributed to manifold the propagation of viruses under Windows 9x/ME.

Unlike Windows and DOS, Windows 95/98 is marketed as having built-in security features. Unfortunately, such features are not robust enough to safeguard Windows 95/98 against viruses. In fact, the first virus written especially to target Windows 95 (the Boza virus) emerged late in 1995. Furthermore, Windows 95's workgroup networking environment has no file-level protection and therefore can potentially lead to increases in virus spreading.

After the rather primitive Boza virus, the Windows 95/98 and Windows NT/2000 viruses have increased in numbers and complexity. Like in the DOS environment, the first viruses were amateurish. Then they have become more technically complex as the virus writers have gained experience. Some of the viruses under Windows 95/98 and Windows NT/2000 spread by active use of the network protocol. A temporary "climax" of complexity and destructive capacity was reached with the CIH virus in 1998 (see page 24). Later viruses have become increasingly sly.

Windows 95/98 shares many characteristics with OS/2 with respect to system architecture and interaction with viruses:

Like OS/2, Windows 95/98 is a 32-bit operating system that supports DOS applications, Windows applications, and native Windows 95/98 applications.

Similar to OS/2's VDMs, Windows 95/98 has VMs (virtual machines)— a System Virtual Machine with separate address spaces for Win32 applications and a shared address space for all Win16 applications; and separate virtual machines for individual DOS applications.

DOS file viruses can easily spread on a Windows 95/98 machine because DOS program files' only limitation under Windows 95/98 is that they cannot write directly to the hard drive.

Each DOS VM takes on the characteristics of the system from the point at which the machine was started. Since Windows 95/98 first starts up by running the same programs as a DOS-only machine does, it is possible that an infected program running during the startup process could go on to infect other program files within that VM. In addition, if that infected program originated from the startup process, it would become active in all VMs that were started in the future. Although program files from one VM cannot infect program files in another VM, it is possible for an infected program file to be loaded into a separate VM in the future, thereby continuing the infection path.

The macro viruses that have been written to date target data files generated from and read by Win16 and Win32 applications that are frequently run on Windows 95/98. As a result, macro virus infections abound on Windows 95/98.

Since the Windows 95/98 boot process is the same as a DOS-only or Windows machine (up to a certain point), boot viruses are able to infect hard drives of Windows 95/98 machines. When Windows 95/98 loads, however, boot viruses are often disabled and not allowed to propagate. On the other hand, if the boot virus has a payload, it may deliver it without requiring the virus to replicate beforehand.

Windows NT/2000/XP

As discussed in the sections on OS/2 and Windows 95/98/ME, Windows NT supports DOS applications, Windows applications, and native Windows NT applications. Like Windows 95/98, Windows NT is backwards compatible, and to some extent with DOS and Windows. Despite the fact that NT's security features are more robust than Windows 95/98's, file viruses can still infect and propagate within Windows NT. DOS applications run in separate VDMs (virtual DOS machines), and file viruses can function within the VDM. Some DOS file viruses might not work in the intended fashion under NT, but there is little about NT's security that prevents file viruses from infecting. NT has a feature called System File Checker (SFC), but that can be bypassed.

As with Windows 95/98, Windows NT supports applications that contain macro programming languages, making NT as vulnerable to macro viruses as old Windows machines.

Because Windows NT machines boot the same way that DOS machines do (up to the point at which NT takes over), boot viruses are able to infect NT hard drives. However, when these boot viruses attempt to go memory resident, they will be stopped by NT and therefore be unable to infect floppies. In effect, this stops the infection path, but the user must still deal with any side effects that the boot virus may have on the system — destructive payloads or manhandling NT's boot area in such a way that NT refuses to load.

Some viruses target Windows NT directly. The W32/Funlove and W32/Bolzano viruses undermine the NT security handling, and the recently discovered NT/CodeRed series of viruses exploit security holes found in software that runs exclusively on NT.

Solutions to the virus problem

Establish routines

Unless organizations and single-users have established internal routines for data handling, the chance for running a virus-free computing environment is not likely to succeed. We have seen that when strategies and routines for data handling are initiated at management level, the organization is less exposed to virus infections. And when they occur, routines make it easier to root out the infected files before they spread.

Anti-virus solutions

When people think of anti-virus solutions, they normally think of scanners. Scanners are the most readily available type of anti-virus solution, but they are not the only type.

It's perhaps best to think of anti-virus solutions in terms of:

- what is required to detect the virus
  - generic methods
  - specific methods

and

- when the virus is detected
  - prior to the attempted infection
  - after the infection

A virus can be detected using either generic methods or specific methods. Generic methods look for virus-like behavior rather than specific viruses. As a result, even new viruses can be detected, and there is little need for frequent updates to the tool that is being used. Because generic methods look for behavior rather than specific viruses, the name of the virus is normally not given. Instead users are simply warned that a virus is likely to be

present. Some shy away from this method because it can give false alarms.

Examples of generic detection methods are:

- checksumming and integrity checking
- heuristics
- decoys
- behavior blocking

Specific methods, on the other hand, rely on having prior knowledge of the virus. In this case the tool is able to both detect that the virus is present as well as identify it. As a result, frequent updates to the tool are necessary. Most users like to know what they're "up against" if a virus is found, and the best way to do that is to determine the exact nature of the beast. For this reason, many users prefer this method, but they do not ultimately appreciate how often the tool must be updated.

Examples of specific detection methods are:

- on-demand and scheduled scanning
- on-access (real-time) scanning

Note that the methods above are not always specific – heuristic methods are usually implemented as parts of on-demand and on-access scanning.

An equally important consideration is when the virus is detected. All users would probably agree that the ideal situation would be to prevent viruses from continuing to infect, and the next most ideal would be to identify those areas that have already been infected.

Let's examine where the above-mentioned methods fall:

Method	Detection discussion
Check-summing and integrity checking	Both methods store information about (hopefully) uninfected files in a certain place. Checks against the current status of the files and the stored information are performed periodically. If any change is detected, then a warning is issued. This method provides after-the-fact detection.
Heuristics	This is a method of analyzing files and boot areas in a general sense to determine if the code appears virus-like. Heuristics perform after-the-fact detection.
Decoys	This is a method of lying in wait for viruses, allowing certain files to become infected if a virus is present. Decoys detect viruses as they are infecting and are helpful in raising the warning flag.
Behavior blocking	This is a method of analyzing the behavior of all computing actions to determine if the sum of the parts adds up to a virus-like action. If so, then the action is stopped before infection can occur. Behavior blocking performs before-the-fact detection.
On-demand and scheduled scanning	This is a method of scanning for specific viruses at certain times. This is always an after-the-fact detection.
On-access scanning	This method also uses scanning, but the detection process occurs while other computer processes occur, such as copying a file. As a result, users are notified of existing viruses before they can be triggered.

The subject of removal of viruses and other malware is complex. Some people have a narrow definition of virus removal - if you delete the file or format the hard drive, the virus is no longer there. Please note, however, that such drastic measures are rarely necessary, and that it is much healthier to define removal as

removing the malicious code entirely and leaving behind a usable and clean system state.

Some of the detection methods listed above can also perform removal (as defined the “healthy” way):

Method	Removal discussion
Checksumming and integrity checking	Can remove viruses.
Heuristics	Sometimes can remove boot and macro viruses.
Decoys	Cannot remove viruses.
Behavior blocking	Can remove viruses from memory and boot viruses from floppies.
On-demand and scheduled scanning	Can remove viruses.
On-access scanning	Can remove viruses.

Sandbox

Automatic detection, immediate cure and distribution of updated virus definition files have been a lifelong dream among AV vendors all over the world. Throughout the years, the AV business has invested significant resources to come up with a solution that could fulfil this ambitious goal.

Norman has developed a solution that achieves this objective. It's called **Sandbox**; a term that best describes the technique that is used to check if a file is infected by an unknown virus. The method allows untrusted, possible viral code to play around on the computer. Not in the *real* computer, but in a *simulated* computer. Simply put, this is how a sandbox works: a file that NVC finds suspicious is tossed into the sandbox. When the file is executed, every "viral" move it makes is meticulously monitored and recorded. Remember that any piece of code – hostile or not – that is subjected to the sandbox's scrutiny believes that it is working on a *real* computer. If it's malware, it's caught in the act and immediate antidote can be prepared. At any rate, no harm is done to real files. But the exposure of a possible viral program is very real.

At the Virus Bulletin Conference in 2001, Norman produced a fully functional prototype of a scanning engine with **Sandbox** functionality. Note that this prototype also had eliminated a major obstacle associated with this technology, namely false alarms. In other words, the Sandbox's analysis was very accurate.

About Sandboxing

In the paragraphs below we'll take a closer look at the sandbox concept, and how it's implemented. A major challenge is to integrate the new technology in the product without slowing down scanning speed, for example.

Sandboxing using emulation

A computer virus is a computer program, defined through its behavior. It will transfer code/data to other computer files. When these other computer file in turn is given control, the virus code is somehow activated, trying to infect other computer files. This process is called replication. For a computer program to be called “viral”, it must be able to perform this task recursively.

Norman’s sandbox is a virtual world where everything is simulated. It is powered by an emulator, and together they let possible virus infected binary executables “run” just as they would do on a real system. When execution stops, the sandbox is analyzed for changes.

However, viruses can detect that they run inside a simulated world. Even though we try to cover most aspects, there will always be an API or service that is not provided which viruses can test. It’s only a subset of *DOS/Windows* which is implemented – just enough to run viruses and their hosts. Many viruses are not stable programs. They contain bugs, and often cause your real CPU to hang. This is also a problem for simulation. When does it hang and when is it just a really slow huge virus?

Since nothing runs on your system, except an emulator—it is safe. Everything else is virtualized. Nothing is stored on your hard disk, even if the sample you are testing, wants to. Since sandboxing with the purpose of detecting viruses include “booting” the “computer”, having a full-blown installation of e.g. *Windows 98 SE* would be too slow and inefficient. Norman’s environment is simulated and tailor-made—i.e. it is written by us. It’s painstaking and meticulous work to develop and test the operating systems so that they match the real ones at a satisfactory level. You can “apply” any tailor-made operating system to inspect viral/malicious behavior of a sample. So we have virtualized the entire computer. A new BIOS, CPU feature, OS fix, and so on can be virtually applied, and several worlds can be simulated in turns. Everything is controlled from our definition file.

It runs inside the scanner engine on any computer/OS we're already supporting, and will detect the same viruses on the different platforms, even on access if you like.

Sandboxing using a virtual machine

It is also possible to build a sandbox by creating a VM (Virtual Machine). The idea is to block all exits, so the executable you are examining cannot escape. However, we don't consider this solution as safe enough. There will always be "another" exploit of the PC's processor, some weird interrupt/exception/fault etc. that will allow malicious code to escape a VM, and spread to your real system.

We've seen it before with other attempts to "fence" malware (for example INT 1 tracing). A VM is too specific to the environment you are running, and cannot be a part of a generic scanner engine. A VM doesn't virtualize the computer your running on. The hardware, CPU and BIOS are equal to the one already running on the system. Using a VM, you cannot test suspicious files for other platforms, like Linux. Your system is not necessarily clean - you use what you have. However, executing inside a VM is faster. Everything happens in real-time. The environment is already there; you don't have to "create" it. You use a copy of the machine as it is, which saves development.

Forecasting the sandbox technology

After the break through with the sandbox, we foresee refining the current version, as well as using the experience from research as a stepping-stone to related functionality within AV technology. Futhermore, configuration options in NVC for sandbox related tasks will be implemented, and other automated procedures may very well follow in the wake of the new innovation. They virus makers are definitely getting more sophisticated, but their opponents in the AV industry are making significant progress, too.

Industry facts

Virus statistics

"99,67% of companies surveyed experienced at least one virus encounter during the survey period. 51% claimed they had at least one "virus disaster" during the 12-month period before they were surveyed."

- *Source: 2000 Computer Virus Prevalence Survey, IICSA.net, October 23, 2000*

General Statistics

"Today, 45% of all corporate information and ideas are stored in an organization's email system." denote

- Source: SC Magazine, August 2001

Financial Statistics

"Including hard and soft dollar figures, the true cost of virus disasters is between $100,000 and $ 1 Millon per company."

- *Source: Computer Crime & Security Survey, Computer Security Institute, March 12, 2001*

The Security Threat

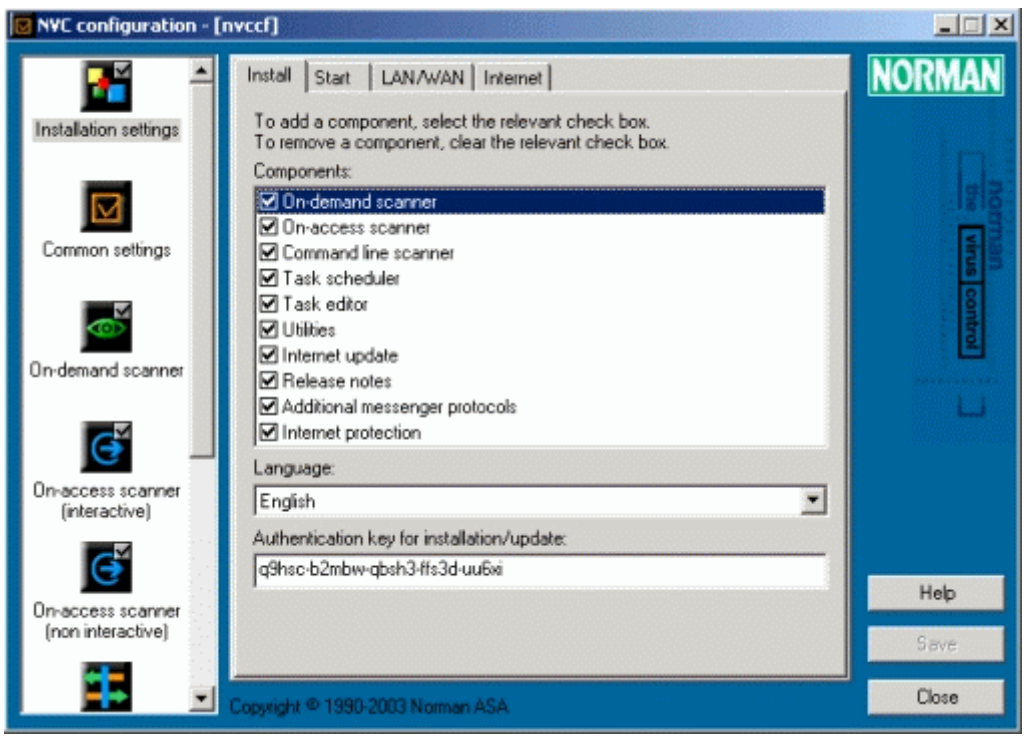
- Reuters reported that over $12 billion in damage was caused by computer viruses in the first 6 months of the year 2000 alone.
- According to "Tippet's Law of Malicious Code", the virus problem doubles about every 14 months.

Norman Virus Control

NVC 5 – a new approach to virus control

Since 1989 Norman ASA has developed one of the world's leading virus control software packages. The current version, NVC 5, is the result of a development cycle that started in 1999 and was finalized a year later.

Virus control has traditionally involved user interaction like 'scanning' of files and boot sectors to detect and remove malign code. Only for the last three or four years have we seen a change towards on-access detection, internet-based updating, and centralized management in virus control. NVC5 takes advantage of the operating systems' potential for full integration. For example, a file system filter driver is installed on Windows NT/2000/XP and OS/2, while a VxD offers the same functionality on Win95/98/Me.



NVC version 5

The product was designed from scratch, emphasizing transparency, reliability and ease of use. Some key objectives were defined, including:

- **Invisibility**
- **Scalability**
- **Accountability**
- **Automatic support and maintenance**

The final design is the result of a philosophy that in many ways is very different from the mainstream virus control products.

Certification

Norman Virus Control 5 is certified by West Coast Labs with the Check Mark level 1 certification, and ICSA labs certified for on-access and on–demand antivirus product.

Awards

Norman Virus Control is one of the most award-winning products in the history of Virus Bulletin Magazine, with a total of 21 Virus Bulletin 100%Awards of 25 possible awards since the start-up January 1998. The 100% award signifies that the product detects 100% of viruses that is reported “in the wild” at the time of testing.

Virus Alert Program

In the wake of the Melissa virus incident (Easter 1999), Norman introduced a virus alert program to assist customers in similar situations in the future.

Norman’s Virus Alert Program offers subscribers a special service in situations when a virus alert situation occurs. To learn more about the program and how to subscribe, see www.norman.no.

Index

—W—

Norman offices

Norway
Norman ASA
Strandveien 37, Lysaker
P.O. Box 43
N-1324 Lysaker
Tel: +47 67 10 97 00
Fax: +47 67 58 99 40
E-mail: norman@norman.no
www.norman.com/no

Denmark
Norman Data Defense Systems AS
Blangstedgårdsvej 1
DK-5220 Odense SØ
Tel. +45 6311 0508
Fax: +45 6313 3901
E-mail: info@normandk.com
www.norman.com/dk

Sweden
Norman Data Defense Systems AB
Birger Jarlsgt. 27
P.O. Box 5044
SE-194 05 Uppland Väsby
Tel. +46 8 988 660
E-mail: sales.se@norman.no
www.norman.com/se

Finland
Norman Ibas Oy
Läkkisepäntie 11
FI-00620 Helsinki
Tel: +358 9 2727 210
Fax: +358 92727 2121
E-mail: norman@norman-ibas.fi
www.norman-ibas.fi

United Kingdom
Norman Data Defense Systems (UK) Ltd
PO Box 5517
Milton Keynes
MK5 6XJ, UK
Tel: +44 (0) 8707 448 044
Fax: +44 (0) 8717 176 999
E-mail: norman@normanuk.com
www.normanuk.com

Germany
Norman Data Defense Systems GmbH
Kieler Str. 15
D-42697 Solingen
Tel: +49 212 267 180
Fax: +49 212 267 1815
E-mail: norman@norman.de
www.norman.de

Switzerland
Norman Data Defense Systems AG
Postfach
CH-4015 Basel
Tel: +41 61 487 2500
Fax: +41 61 487 2501
E-mail: norman@norman.ch
www.norman.ch

The Netherlands
Norman/SHARK B.V.
Postbus 159
NL-2130 AD Hoofddorp
Tel: +31 23 789 0222
Fax: +31 23 561 3165
E-mail: info@norman.nl
www.norman.nl

USA
Norman Data Defense Systems Inc.
9302 Lee Highway, Suite 950A
Fairfax, VA 22031, USA
Tel: +1 703 267 6109
Fax: +1 703 934 6367
E-mail: norman@norman.com
www.norman.com

Norman solutions for clients/workstations
Norman Virus Control for Microsoft Windows 95, 98, Me, NT4.0, 2000, XP, OS/2, Linux (On-Demand scanning) | Norman Internet Control for Microsoft Windows 95, 98, Me, NT4.0, 2000, XP | Norman Personal Firewall | Norman Privacy

Norman solutions for servers
Norman Virus Control for Microsoft Windows XP, 2000, NT40 | Norman Virus Control Firebreak for Novell Netware 4.11 and later | Norman Virus Control for Linux | Norman Virus Control for Lotus Domino (Win32, OS/2) | Norman Virus Control for Firewall-1 (and next generation version)

Norman solutions for web, gateways and mailservers
Norman Spam Control | Norman Email Control | Norman Download Control | Norman Virus Control Net | Norman Virus Control for Microsoft Internet Information Server | Norman Virus Control for MIMEsweeper | Norman Virus Control for Microsoft Exchange

NORMAN®

www.norman.com