COMPUTER VIRUSES AND ETHICS

Eugene H. Spafford

CSD-TR-91-061
August 1991

Computer Viruses and Ethics

Purdue Technical Report CSD-TR-91-061

Eugene H. Spafford
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907–1398
spaf@cs.purdue.edu

August 20, 1991

Abstract

There has been considerable interest in computer viruses since they first appeared in 1981, and especially in the past few years as they have reached epidemic numbers in many personal computer environments. Viruses have been written about as a security problem, as a social problem, and as a possible means of performing useful tasks in a distributed manner. Most users of computers view viruses as annoying or dangerous. Some people, however, claim that (at least some) viruses are beneficial.

This paper begins with a description of how computer viruses operate, and the various ways simple viruses are structured. It then discusses the most common reasons put forth to explain the writing of viruses, and discusses whether those reasons justify the resultant damages.

1 Introduction

There has been considerable interest of late in computer viruses, and in the motives behind the writing of viruses. Some individuals attempt to justify the writing of viruses by claiming it is for a greater good.

This paper is a condensed, high-level description of computer viruses — their history, structure, and how they affect computers It provides a general introduction to the topic without requiring an extensive background in computer science.

To appear in a special issue of the journal *Collegiate Microcomputing*, volume X, 1992.

The interested reader might pursue [12, 3, 4] and [7] for more detail about computer viruses and their properties. The description in the next few sections of the origins of computer viruses and their structure is derived from [12].

2 What is a Computer Virus?

The term *computer virus* is derived from and analogous to a biological virus. The word *virus* itself is Latin for *poison*. Viral infections are spread by the virus (a small shell containing genetic material) injecting its contents into a far larger body cell. The cell then is infected and converted into a biological factory producing replicants of the virus.

Similarly, a computer virus is a segment of machine code (typically 200-4000 bytes) that will copy its code into one or more larger "host" programs when it is activated. When these infected programs are run, the viral code is executed and the virus spreads further. Viruses cannot spread by infecting pure data; pure data is not executed. However, some data, such as files with spreadsheet input or text files for editing, may be interpreted by application programs. For instance, text files may contain special sequences of characters that are executed as editor commands when the file is first read into the editor. Under these circumstances, the data is "executed" and may spread a virus. Data files may also contain "hidden" code that is executed when the data is used by an application, and this too may be infected. Technically speaking, however, pure data itself cannot be infected.

2.1 Worms

Worms are another form of software that is often referred to by the uninformed as a computer virus. The Internet Worm of November 1988 is an example of one of these programs.[10, 9]

Unlike viruses, worms are programs that can run independently and travel from machine to machine across network connections; worms may have portions of themselves running on many different machines. Worms do not change other programs, although they may carry other code that does, such as a true virus.

In 1982, John Shoch and Jon Hupp of Xerox PARC (Palo Alto Research Center) described the first computer worms. [8] They were working with an experimental, networked environment using one of the first local area networks. While searching for something that would use their networked environment, one of them remembered reading *The Shockwave Rider* by John Brunner, written in 1975. This science fiction novel described programs that traversed networks, carrying information with them. Those programs were called *tapeworms* in the novel. Shoch and Hupp named their own programs *worms*, because in a similar fashion they would travel from workstation to workstation, reclaiming file space, shutting off idle workstations, delivering mail, and doing other useful tasks.

Few computer worms have been written in the time since then, especially worms that have caused damage, because they are not easy to write. Worms require a network environment and an author who is familiar not only with the network services and facilities, but also with the operating facilities required to support them once they have reached the machine. The Internet worm incident of November, 1988 clogged machines and networks as it spread, and is an example of a worm.

Worms have also appeared in other science fiction literature. Recent "cyberpunk" novels such as *Neuromancer* by William Gibson [6] refer to worms by the term "virus." The media has also often referred incorrectly to worms as viruses. This paper focuses only on viruses as defined here. Many of the comments about viruses and artificial life may also be applied to worm programs.

2.2 Other Threats

There are many other kinds of *vandalware* that are often referred to as viruses, including *bacteria, trojan horses, logic bombs,* and *trapdoors.* These will not be described here. The interested reader can find explanations in [12] and [4].

2.3 Names

As the authors of viruses generally do not name their work formally and do not come forward to claim credit for their efforts, it is usually up to the community that discovers a virus to name it. A virus name may be based on where it is first discovered or where a major infection occurred, e.g., the *Lehigh* and *Alameda* viruses. Other times, the virus is named after some definitive string or value used by the program, e.g., the *Brain* and *Den Zuk* viruses. Sometimes, viruses are named after the number of bytes by which they extend infected programs, such as the *1704* and *1280* viruses. Still others may be named after software for which the virus shows an affinity, e.g., the *dBase* virus. In the remainder of this paper, viruses are referred to by commonly-accepted names. Refer to [12] or [14] for detailed lists of virus names and characteristics.

2.4 A history lesson

The first use of the term *virus* to refer to unwanted computer code occurred in 1972 in a science fiction novel, *When Harley Was One*, by David Gerrold.[1] The description of *virus* in that book does not fit the currently-accepted definition of computer virus—a program that alters other programs to include a copy of itself. Fred Cohen formally defined the term *computer virus* in 1983. [3] At that time, Cohen was a graduate student at the University of Southern California attending a security seminar. The idea of writing a computer virus occurred to him, and in a week's time he put together a simple virus that he demonstrated to the class. His advisor, Professor Len Adelman, suggested that he call his creation a computer virus. Dr. Cohen's thesis and later research were devoted to computer viruses.

It appears, however, that computer viruses were being written by other individuals, although not named such, as early as 1981 on Apple II computers.[2] Some early Apple II viruses included the notorious "Festering Hate," "Cyberaids," and "Elk Cloner" strains. Sometimes virus infections were mistaken as trojan horses, as in the "Zlink virus," [sic] which was a case of the Zlink communication program infected by "Festering Hate." The "Elk Cloner" virus was first reported in mid-1981.

It is only within the last three years that the problem of viruses has grown to significant proportions. Since the first detected infection by the *Brain* virus in January 1986, up to 31 December, 1990, the number of distinctly different IBM PC viruses grew to over 225. If current trends continue, one projection estimates that by 1 January 1994 we will have almost 39,00 different strains of virus for the IBM PC alone! Examined another way, in 1989, a new virus appeared every week; in 1990, a new virus appeared approximately every two days; in 1991, we are seeing about six new viruses every day.[13]

The problem is not restricted to the IBM PC, and viruses now affect all popular personal computers. Mainframe viruses do exist for a variety of operating systems and machines, but all reported to date have been experimental in nature, written by serious academic researchers in controlled environments.[3]

Where viruses have flourished is in the weak security environment of the personal computer. Personal computers were originally designed for a single dedicated user — little, if any, thought was given to the difficulties that might arise should others have even indirect access to the machine. The systems contained no security facilities beyond an optional key switch, and there was a minimal amount of security-related software available to safeguard

[1] The recent reissue of Gerrold's book has this subplot omitted.

[2] Private communication from Joe Dellinger.

[3] E.g., the UNIX virus described in [5].

data. Today, however, personal computers are being used for tasks far different from those originally envisioned, including managing company databases and participating in networks of computer systems. Unfortunately, their hardware and operating systems are still based on the assumption of single trusted user access, and this allows computer viruses to flourish on those machines.

2.5 Formal structure

True viruses have two major components: one that handles the spread of the virus, and a manipulation task. The manipulation task may not be present (has null effect), or it may act like a logic bomb, awaiting a set of predetermined circumstances before triggering. These two virus components will be described in general terms, and then more specific examples will be presented as they relate to the most common personal computer: the IBM PC. Viruses on other machines behave in a similar fashion.

2.5.1 A Note About Mainframe Viruses

As already noted, viruses can infect minicomputers and mainframes as well as personal computers. Laboratory experiments conducted by various researchers have shown that any machine with almost any operating system can support computer viruses. However, there have been no documented cases of true viruses on large multi-user computers other than as experiments. This is caused by, in part, both the greater restrictions built into the software and hardware of those machines, and the way they are usually used. My further comments will therefore be directed towards PC viruses, with the understanding that analogous statements could be made about mainframe viruses.

2.5.2 Structure

For a computer virus to work, it somehow must add itself to other executable code. The viral code must be executed before the code of its infected host (if the host code is ever executed again). One form of classification of computer viruses is based on the three ways a virus may add itself to host code: as a shell, as an add-on, and as intrusive code.

Shell viruses A shell virus is one that forms a "shell" (as in "eggshell" rather than "Unix shell") around the original code. In effect, the virus becomes the program, and the original host program becomes an internal subroutine of the viral code. An extreme example of this would be a case where the virus moves the original code to a new location and takes on its identity. When the virus is finished executing, it retrieves the host program code and begins its execution.

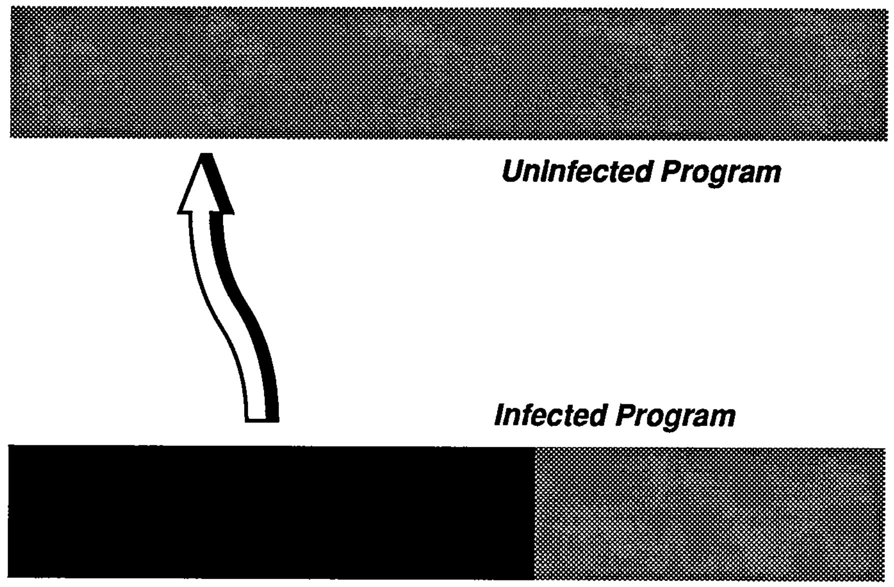

Figure 1: Shell Virus Infection

Add-on viruses Most viruses are add-on viruses. They function by appending their code to the end of the host code, or by relocating the host code and adding their own code to the beginning. The add-on virus then alters the startup information of the program, executing the viral code before the code for the main program. The host code is left almost completely untouched; the only visible indication that a virus is present is that the file grows larger.

Intrusive viruses Intrusive viruses operate by replacing some or all of the original host code with viral code. The replacement might be selective, as in replacing a subroutine with the virus, or inserting a new interrupt vector and routine. The replacement may also be extensive, as when large portions of the host program are completely replaced by the viral code. In the latter case, the original program can no longer function.

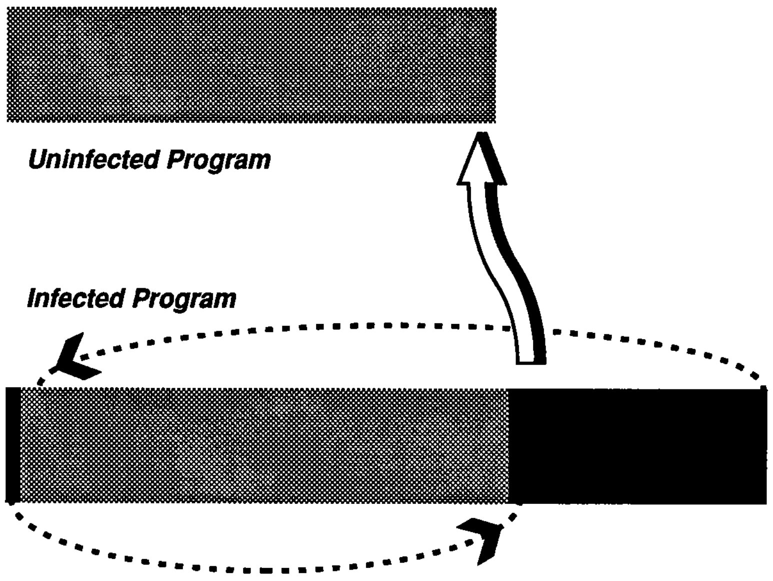

Figure 2: Add-on Virus Infection

2.5.3 Triggers

Once a virus has infected a program, it seeks to spread itself to other programs, and eventually to other systems. Simple viruses do no more than this, but most viruses are not simple viruses. Common viruses wait for a specific triggering condition, and then perform some activity. The activity can be as simple as printing a message to the user, or as complex as seeking particular data items in a specific file and changing their values. Often, viruses are destructive, removing files or reformatting entire disks.

The conditions that trigger viruses can be arbitrarily complex. If it is possible to write a program to determine a set of conditions, then those same conditions can be used to trigger a virus. This includes waiting for a specific date or time, determining the presence or absence of a specific set of files (or their contents), examining user keystrokes for a sequence of input, examining display memory for a specific pattern, or checking file attributes for modification and permission information. Viruses also may be triggered based on some random event. One common trigger component is a counter used to determine how many additional programs the virus has succeeded in infecting—the virus does not trigger until it has propagated itself a certain minimum number of times. Of course, the trigger can be any combination of these conditions, too.

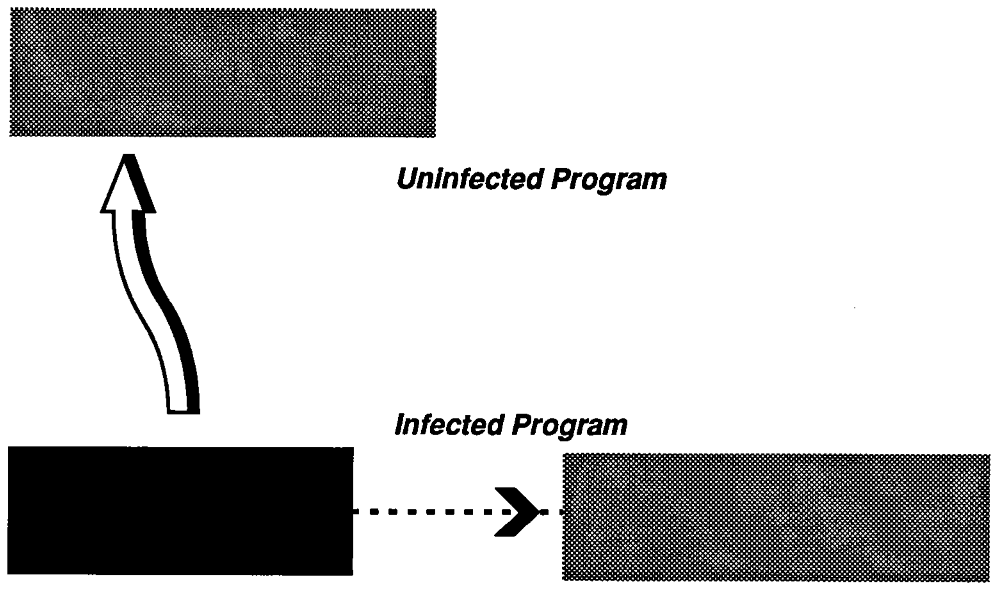

Figure 3: Intrusive Virus Infection

2.6 How do viruses spread?

Computer viruses can infect any form of writable storage, including hard disk, floppy disk, tape, optical media, or memory. Infections can spread when a computer is booted from an infected disk, or when an infected program is run. It is important to realize that often the chain of infection can be complex and convoluted. A possible infection might spread in the following way:

- A client brings in a diskette with a program that is malfunctioning (because of a viral infection).
- The consultant runs the program to discover the cause of the bug—the virus spreads into the memory of the consultant's computer.
- The consultant copies the program to another disk for later investigation—the virus infects the copy utility on the hard disk.
- The consultant moves on to other work preparing a letter—the virus infects the screen editor on the hard disk.

- The system is switched off and rebooted the next day—the virus is cleared from memory, only to be reinstalled when either the screen editor or copy utility is used next.
- Someone invokes the infected screen editor across a network link, thus infecting their own system.

2.7 The three stages of a virus's life

For a virus to spread, its code must be executed. This can occur either as the direct result of a user invoking an infected program, or indirectly through the system executing the code as part of the system boot sequence or a background administration task.

The virus then replicates, infecting other programs. It may replicate into only one program at a time, it may infect some randomly-chosen set of programs, or it may infect every program on the system. Sometimes a virus will replicate based on some random event or on the current value of the clock. The different methods will not be presented in detail because the result is the same: there are additional copies of the virus on your system.

Finally, most viruses incorporate a manipulation task that can consist of a variety of effects (some odd, some malevolent) indicating the presence of the virus. Typical manipulations might include amusing screen displays, unusual sound effects, system reboots, or the reformatting of the user's hard disk.

2.7.1 Activating a virus

The IBM PC can be used as an example to illustrate how a virus is activated. Viruses in other types of computer systems behave in similar manners.

The IBM PC boot sequence This section gives a description of the various points in the IBM PC boot sequence that can be infected by a virus. I will not go into extensive detail about the operations at each of these stages; the interested reader may consult the operations manuals of these systems, or any of the many "how-to" books available.

The IBM PC boot sequence has six components:

- ROM BIOS routines
- Partition record code execution
- Boot sector code execution

- IO.SYS and MSDOS.SYS code execution
- COMMAND.COM command shell execution
- AUTOEXEC.BAT batch file execution

ROM BIOS When an IBM PC, or compatible PC, is booted, the machine executes a set of routines in ROM (read-only memory). These routines initialize the hardware and provide a basic set of input/output routines that can be used to access the disks, screen, and keyboard of the system. These routines constitute the basic input/output system (BIOS).

ROM routines cannot be infected by viral code (except at the manufacturing stage), as they are present in read-only memory that cannot be modified by software. Some manufacturers now provide extended ROMs containing further components of the boot sequence (e.g., partition record and boot sector code). This trend reduces the opportunities for viral infection, but also may reduce the flexibility and configurability of the final system.

Partition Record The ROM code executes a block of code stored at a well-known location on the hard disk (head 0, track 0, sector 1). The IBM PC disk operating system (DOS) allows a hard disk unit to be divided into up to four logical partitions. Thus, a 100Mb hard disk could be divided into one 60Mb and two 20Mb partitions. These partitions are seen by DOS as separate drives: "C," "D," and so on. The size of each partition is stored in the partition record, as is a block of code responsible for locating a boot block on one of the logical partitions.

The partition record code can be infected by a virus, but the code block is only 446 bytes in length. Thus, a common approach is to hide the original partition record at a known location on the disk, and then to chain to this sector from the viral code in the partition record. This is the technique used by the New Zealand virus, discovered in 1988. (See figures 4 and 5.)

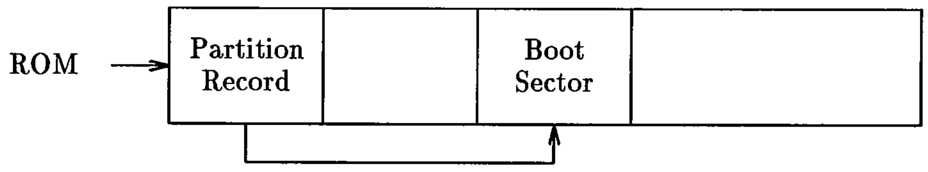

Figure 4: Hard disk before infection

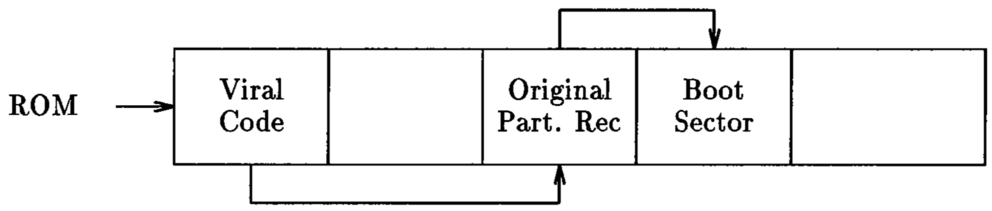

Figure 5: Hard disk after infection by New Zealand Virus

Boot sectors The partition record code locates the first sector on the logical partition, known as the boot sector. (If a floppy disk is inserted, the ROM will execute the code in its boot sector, head 0, track 0, sector 1.) The boot sector contains the BIOS parameter block (BPB). The BPB contains detailed information on the layout of the filing system on disk, as well as code to locate the file IO.SYS. That file contains the next stage in the boot sequence. (See Figure 6.)

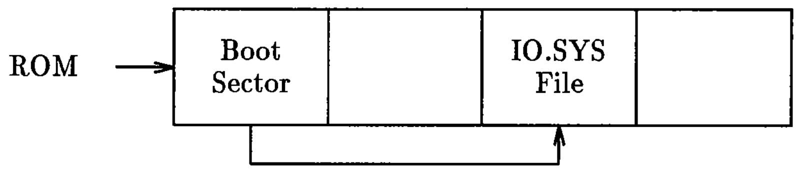

Figure 6: Floppy disk before infection

A common use of the boot sector is to execute an application program, such as a game, automatically; unfortunately, this can include automatic initiation of a virus. Thus, the boot sector is a common target for infection.

Available space in the boot sector is limited, too (a little over 460 bytes is available). Hence, the technique of relocating the original boot sector while filling the first sector with viral code is also used here.

A typical example of such a "boot sector" virus is the *Alameda* virus. This virus relocates the original boot sector to track 39, sector 8, and replaces it with its own viral code. (See Figure 7.)

Other well-known boot sector viruses include the *New Zealand* (on floppy only), *Brain, Search*, and *Italian* viruses. Boot sector viruses are particularly dangerous because they capture control of the computer system early in the boot sequence, before any anti-viral utility becomes active.

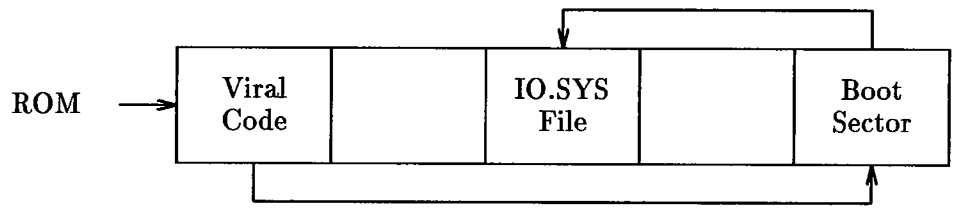

Figure 7: After Alameda Virus Infection

MSDOS.SYS, IO.SYS The boot sector next loads the IO.SYS file, which carries out further system initialization, then loads the DOS system contained in the MSDOS.SYS file. Both these files could be subject to viral infection, although no known viruses target them.

Command shell The MSDOS.SYS code next executes the command shell program (COMMAND.COM). This program provides the interface with the user, allowing execution of commands from the keyboard. The COMMAND.COM program can be infected, as can any other .COM or .EXE executable binary file.

The COMMAND.COM file is the specific target of the *Lehigh* virus that struck Lehigh University in November 1987. This virus caused corruption of hard disks after it had spread to four additional COMMAND.COM files.

AUTOEXEC batch files The COMMAND.COM program is next in the boot sequence. It executes a list of commands stored in the AUTOEXEC.BAT file. This is simply a text file full of commands to be executed by the command interpreter. A virus could modify this file to include execution of itself. Ralf Burger has described how to do exactly that in his book *Computer Viruses—A High Tech Disease.* His virus uses line editor commands to edit its code into batch files. Although a curiosity, such a virus would be slow to replicate and easy to spot. This technique is not used by any known viruses "in the wild."

Infection of a user program A second major group of viruses spreads by infecting program code files. To infect a code file, the virus must insert its code in such a way that it is executed before its infected host program. These viruses come in two forms:

Overwriting The virus writes its code directly over the host program, destroying part or all of its code. The host program will no longer execute correctly after infection.

Non-overwriting The virus relocates the host code, so that the code is intact and the host program can execute normally.

A common approach used for .COM files is to exploit the fact that many of them contain a jump to the start of the executable code. The virus may infect the programs by storing this jump, and then replacing it with a jump to its own code. When the infected program is run, the virus code is executed. When the virus finishes, it jumps to the start of the program's original code using the stored jump address. (See Figure 8.)

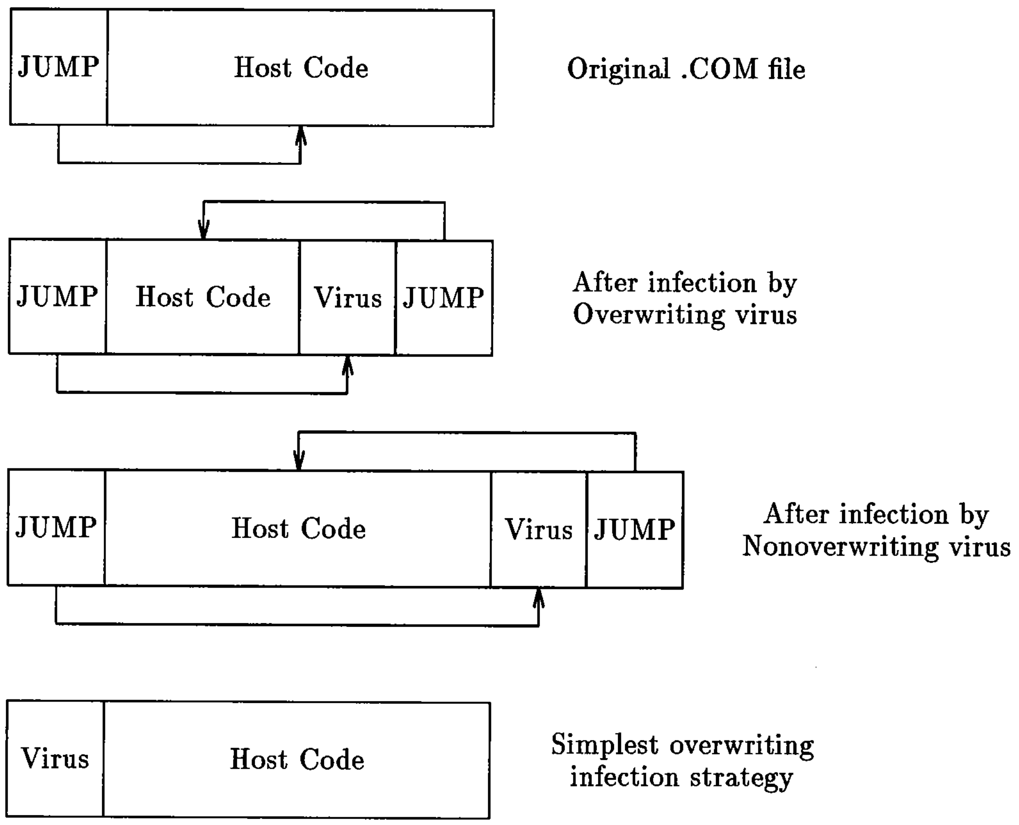
JUMP
Host Code
Original .COM file
JUMP
Host Code
Virus
JUMP
After infection by Overwriting virus
JUMP
Host Code
Virus
JUMP
After infection by Nonoverwriting virus
Virus
Host Code
Simplest overwriting infection strategy

Figure 8: Infection of user applications

Notice that in the case of the overwriting virus, the more complex infection strategy often means that all but a small block of the original program is intact. This means that the original program can be started, although often it will exhibit sporadic errors or abnormal behavior.

Memory-resident viruses The most "successful" viruses to date exploit a variety of techniques to remain resident in memory once their code has been executed and their host program has terminated. This implies that, once a single infected program has been run, the virus potentially can spread to any or all programs in the system. This spreading occurs during the entire work session (until the system is rebooted to clear the virus from memory), rather than during a small period of time when the infected program is executing viral code.

Thus, the two categories of memory-resident virus are:

Transient The viral code is active only when the infected portion of the host program is being executed.

Resident The virus copies itself into a block of memory and arranges to remain active after the host program has terminated. The viruses are also known as TSR (Terminate and Stay **R**esident) viruses.

Examples of memory-resident viruses are all known boot sector viruses, the *Israeli, Cascade*, and *Traceback* viruses.

If a virus is present in memory after an application exits, how does it remain active? That is, how does the virus continue to infect other programs? The answer is that it also infects the standard interrupts used by DOS and the BIOS so that it is invoked by other applications when they make service requests.

The IBM PC uses many interrupts (both hardware and software) to deal with asynchronous events and to invoke system functions. All services provided by the BIOS and DOS are invoked by the user storing parameters in machine registers, then causing a software interrupt.

When an interrupt is raised, the operating system calls the routine whose address it finds in a special table known as the *vector* or *interrupt* table. Normally, this table contains pointers to handler routines in the ROM or in memory-resident portions of the DOS (see figure 9). A virus can modify this table so that the interrupt causes viral code (resident in memory) to be executed.

By trapping the keyboard interrupt, a virus can arrange to intercept the CTRL-ALT-DEL soft reboot command, modify user keystrokes, or be invoked on each keystroke. By trapping the BIOS disk interrupt, a virus can intercept all BIOS disk activity, including reads of boot sectors, or disguise disk accesses to infect as part of a user's disk request. By trapping the DOS service interrupt, a virus can intercept all DOS service requests including program execution, DOS disk access, and memory allocation requests.

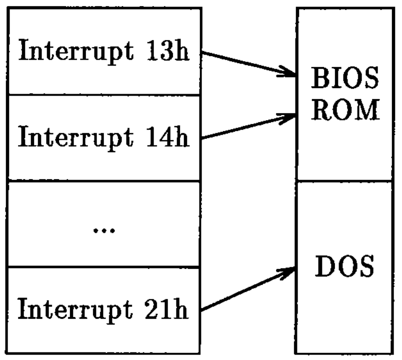

Figure 9: Normal interrupt usage

A typical virus might trap the DOS service interrupt, causing its code to be executed before calling the real DOS handler to process the request. (See figure 10.)

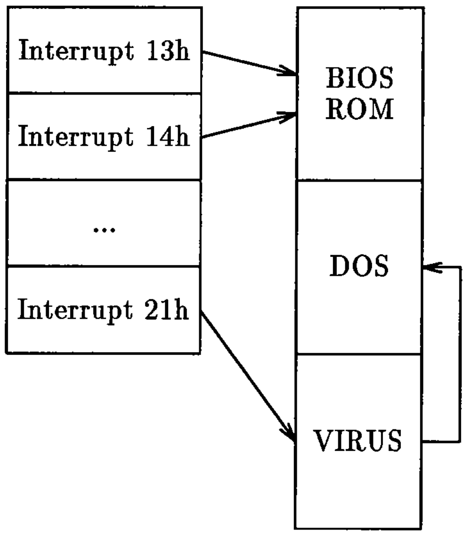

Figure 10: Interrupt vectors with TSR virus

2.7.2 Replication strategies

Types Viruses can be grouped into four categories, based on the type of files they infect:

- Boot sector viruses that only infect boot sectors (or rarely, partition records)

- System viruses that are targeted against particular system files, such as the DOS command shell
- Direct viruses that scan through the DOS directory structure on disk looking for suitable files to infect
- Indirect viruses that wait until the user carries out an activity on a file (e.g., execution of a program) before infecting it

Transient viruses are always direct in that they attempt to infect one or more files (usually in the same directory or home directory) before terminating. Resident viruses can be either direct or indirect (or worse, both). The recently reported *Traceback* virus infects any file executed (indirect), while also incrementally scanning the directory structure (direct).

In general, indirect viruses are slower to spread, but often pass unnoticed as their infection activities are disguised among other disk access requests.

Signatures to prevent reinfection One problem encountered by viruses is that of repeated infection of the host, leading to depleted memory and early detection. In the case of boot sector viruses, this could (depending on strategy) cause a long chain of linked sectors. In the case of a program-infecting virus (or link virus), repeated infection may result in continual extension of the host program each time it is reinfected. There are indeed some viruses that exhibit this behavior (e.g., the Israeli virus extends .EXE files 1808 bytes each time they are infected).

To prevent this unnecessary growth of infected files, many viruses implant a unique *signature* that signals that the file or sector is infected. The virus will check for this signature before attempting infection, and will place it when infection has taken place; if the signature is present, the virus will not reinfect the host.

A virus signature can be a characteristic sequence of bytes at a known offset on disk or in memory, a specific feature of the directory entry (e.g., alteration time or file length), or a special system call available only when the virus is active in memory.

The signature is a mixed blessing. The virus would be easier to spot if reinfections caused disk space to be exhausted or showed obvious disk activity, but the signature does provide a method of detection and protection. Virus sweep programs are available that scan files on disk for the signatures of known viruses, as are "inoculation" routines that fake the viral signature in clean systems to prevent the virus from attempting infection.

2.8 Other Characteristics

Recent viruses exhibit "stealth" characteristics that prevent their detection and elimination. These viruses set intercepts of system interrupts to alter data returned by BIOS calls, thus "sanitizing" buffers that might otherwise show the virus code. Some viruses, referred to as "stage 4" viruses, encrypt themselves when infecting a program. This prevents virus scanning programs from detecting them, because they have no distinctive pattern for which to scan.

Even without stealth characteristics, viruses have spread widely. One study of over 2500 large computer installations show that the incidence of computer virus infections is growing exponentially. The rate of growth shown in that study is greater than 3% per month, and over 10% if incidents affected more than 25 machines at a time.[13] Experience with these viruses indicate that the majority are strains written over two years ago but that are now making a significant impact.

3 Reasons

There have been many reasons given for the writing of computer viruses. Each has been an attempt to justify the propriety of writing a computer virus.

In the following sections, I will outline the most common reasons given, and briefly discuss the suggested rationale. This discussion is not a rigorous philosophical discussion, but rather a practical view by a computer scientist. The reader is left to draw his or her own conclusions based on personal ethics.

3.1 Experimentation

Undoubtedly, some viruses have been written as experiments. Work by Cohen and Duff, for instance, fall into this category.[3, 5] However, very few instances of virus writing are appropriate with this as a rationale.

Experiments, when done appropriately, involve careful formulation of both a main hypothesis and a null hypothesis. This allows the research to distinguish the results from a random occurrence. Next, the experiment is conducted under a carefully controlled set of circumstances where observations are noted. When sufficient data has been collected, the experiment is stopped.

Computer viruses, when released into an uncontrolled population, fail to meet the requirements of good science. Although a hypothesis may have been formulated ("I bet this will really wreak havoc!"), a carefully formulated null hypothesis is seldom present. The experiment is not well-controlled as the author has no control over the spread of the virus, and no capability of closely observing effects over time. Furthermore, many viruses never "expire." Instead, the viruses continue to spread from machine to machine.

Perhaps the biggest problem with the excuse of "experimentation" is that the subjects have not given informed consent. A pervasive concept within the scientific community is that when people are involved in an experiment, they should be informed of the consequences and side-effects, and allowed to choose whether or not they want to participate. People who find their computers infected with a virus have not given this informed consent, and they are victims rather than experimental subjects.

Claiming that writing of computer viruses is experimental is akin to saying that mixing chemicals together in a flask to see if it explodes is a scientific experiment. Worse, the flask is then placed inside an unmarked box and put on a supermarket shelf for unsuspecting shoppers to detonate. Although a computer virus is not as life-threatening as a bomb, the analogy is reasonable because of the damage that can be caused to an unsuspecting victim.

3.2 Education

Some virus authors have claimed that they have written viruses to further their knowledge of how PCs work. This is an excuse that is often given by people who break into computer systems — they claim they are learning about computer security.[11] As an educator, I would equate this with the claim that one can learn more about automotive engineering by putting sugar in the gas tank and setting fire to the upholstery.

To learn more about how a computer works, one should study the manuals an literature, and build some projects. Writing a full-screen editor would provide as much (or more) education about how a particular computer works than would writing a virus. Writing and then releasing a virus provides no educational value that cannot be achieved otherwise, and with far less damage to others.

3.3 Accident

The claim that the release of a virus was accidental is closely related to the attempts to justify virus-writing as an experimental or educational activity. The authors claim that they wrote the virus for one of those purposes and it "escaped" by accident.

Certainly, accidents do happen. However, one must look at the underlying reason the virus was written. Using an "accident" as an excuse does not justify the writing of the virus in the first place. Even if the virus was written for a legitimate purpose (something I argue is unlikely), the accidental release likely evidences careless or reckless behavior. Fires and explosions cannot be excused away by accidents, and neither can viruses. If a researcher were doing work with recombinent DNA on the bacilli that cause Black Plague, and the bacilli accidently escaped the laboratory, the researcher would still be morally culpable.

3.4 Political Statements

Many computer viruses have been written that convey some political message when activated. The Den Zuk, Peace, and Dukakis viruses are all examples of this;[12] a more recent example is the "Evil Empire" virus.[1]

In general, we may see some activities as morally justified, even if they are illegal. The peaceful demonstrations of Ghandi and Martin Luther King, for example, were illegal but most people today view them of high moral character because of the wrongs they sought to correct. Even in the realm of computers, computer tampering in support of a political agenda may be viewed as somewhat heroic, such as protesting an oppressive communist government.[2]

The case of computer viruses is not quite the same, however. Viruses disrupt the systems of the supporters as well as the opponents of any particular cause. Furthermore, the virus is likely to be around far longer than some of the causes espoused: the Den Zuk virus made statements about Ronald Reagan and Margaret Thatcher, both of whom are no longer in office; the Dukakis virus was written well in advance of the 1988 US Presidential campaign and Michael Dukakis has now dropped from public life; and the Evil Empire virus has been rendered moot by Operation Desert Storm and the subsequent withdrawl of troops. Unfortunately, all of these viruses are likely to remain in the population of personal computers indefinitely.

Using a virus as a form of political expression is more an act of terrorism than a sophisticated political statement. Rather than garnering sympathy from a selected audience, it is likely to antagonize potential supporters. As an act of terrorism, its ethical nature should be obvious.

3.5 Not Illegal, Boredom

A great number of viruses have recently appeared from authors in Eastern European countries, particularly Bulgaria. Some of the virus authors have been identified. When questioned about their motives, they have responded that they write the viruses because they are bored and because there are no laws against it (in their countries).

This is obviously a silly excuse. Common vandalism of all kinds is often explained as being the result of boredom, but it does not make the activity ethical. The damage caused by computer viruses is global in scope and may cost hundreds of millions of dollars a year. Boredom certainly does not justify this amount of havoc, especially when there are other outlets for challenge and interest (e.g., writing applications or games).

Some philosophical systems state that society determines ethics through its laws. Under those philosophies, the fact that there are no laws against writing viruses might seem to make virus-authorship ethical. However, the viruses have continued to propagate to locales where there *are* laws against such behavior. The authors know this, but continue to write new viruses; their concern is not whether the activity is illegal somewhere else, and thus potentially unethical. Rather, their concern is whether they may be caught and punished locally. This shows their awareness that their activity may be unethical and harmful. It also explains why very few viruses ever carry identification indicating the original author.

4 Concluding Remarks

Some other reasons have rarely been given to explain the writing of computer viruses, but I have never seen documented evidence that they were offered by actual virus authors. Instead, they have been speculation by others as to the reasons why the viruses are being written. Noteriety and blackmail are two examples of these other reasons, for instance. A little thought about these justifications should suffice to discern their ethical nature.

We have three methods of limiting the writing and spread of new viruses. One way is to improve the security of our existing and future computing base. Unfortunately, the vast investment in software and hardware make any such improvement limited in scope and distribution. Most computers are used for business and educational purposes, and putting awkward or expensive constraints on those systems would render them unsuitable for their intended purposes. Only through slow evolution of systems brought to market will we be able to transition to safer systems. This is indeed unfortunate, as this is the only sure way to limit the spread of many forms of computer virus.

A second method of limiting computer virus spread is to legislate and enforce strict laws and penalties against the writing of computer viruses. This is unlikely to have much effect, however, for two reasons. First, it would require the uniform effort of governments world-wide to pass such laws. It is not currently possible to get the nations of the world to agree on simpler issues such as those of intellectual property — it seems unlikely that there will be any universal computer virus laws anytime soon. Second, investigation of computer virus writing is very, very difficult. Unless an investigator is extremely lucky, or an author admits under oath to having written a virus, it would seem to be almost impossible to prove guilt in a court of law. To date, in five years of malicious virus writing with many hundreds of viruses released, I am unaware of any prosecution being initiated in either a criminal or civil court against a virus author.

The third method to defeat computer viruses is to inculcate a sense of shared moral outrage about the writing of computer viruses. As a community of professionals, we should realize the threat to our profession and futures posed by computer viruses (and other vandalware). Some limited thought on the subject, along the lines give in the preceding section, should make it obvious that there is no justification for anyone to ever write a virus and release it. We should make this clear to our peers, our students, and our employers. We need to make it clear that writing viruses is not done for "fun," and neither is it acceptable behavior.

Developing a sense of community against the development of viruses will not eliminate them, certainly. It may, however, help discourage the developers of viruses who currently give little thought as to the effects and scope of the results. Unfortunately, even if no new viruses were ever written, we would continue to face the untoward effects of the existing viruses for years to come. It is never too late, however, to do what is right.

References

[1] Jim Bates. Virus analyses: Evil empire — hypocrisy writ large. *Virus Bulletin*, May 1991.

[2] J. J. Buck BloomBecker. *Spectacular Computer Crimes.* Dow-Jones-Irwin, 1990.

[3] Fred Cohen. *Computer Viruses.* PhD thesis, University of Southern California, 1985.

[4] Peter J. Denning. *Computers Under Attack: Intruders, Worms and Viruses.* ACM Press (Addison-Wesley), 1990.

[5] Tom Duff. Experiences with viruses on Unix systems. *Computing Systems*, 2(2), Spring 1989.

[6] William Gibson. *Neuromancer.* Ace/The Berkeley Publishing Group, 1984.

[7] Lance J. Hoffman. *Rogue Programs: Viruses, Worms, and Trojan Horses.* Van Nostrand Reinhold, New York, NY, 1990.

[8] John F. Shoch and Jon A. Hupp. The 'worm' programs—early experiments with a distributed computation. *Communications of the ACM*, 25(3):172–180, March 1982.

[9] Eugene H. Spafford. An analysis of the internet worm. In C. Ghezzi and J. A. McDermid, editors, *Proceedings of the 2nd European Software Engineering Conference*, pages 446–468. Springer-Verlag, September 1989.

[10] Eugene H. Spafford. The internet worm: Crisis and aftermath. *Communications of the ACM*, 32(6):678–687, June 1989.

[11] Eugene H. Spafford. Are computer break-ins ethical? *Journal of Systems and Software*, 13:88–101, January 1992.

[12] Eugene H. Spafford, Kathleen A. Heaphy, and David J. Ferbrache. *Computer Viruses: Dealing with Electronic Vandalism and Programmed Threats.* ADAPSO, Arlington, VA, 1989.

[13] David Stang. Virus trends: Up, up, up. *NCSA News*, 2(3):2, March–April 1991. National Computer Security Association.

[14] David J. Stang. *Computer Viruses.* National Computer Security Association, Washington, DC, 2nd edition, March 1990.