Computer Viruses—A Form of Artificial Life?

Technical Report CSD-TR-985

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

September 19, 1991

This appears in *Artificial Life II, SFI Studies in the Sciences of Complexity,* vol. XII, Eds. D. Farmer, C. Langton, S. Rasmussen and C. Taylor, Addison-Wesley, 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.

However, only recently have some scientists begun to ask if computer viruses are not a form of artificial life—a self-replicating organism. Simply because computer viruses do not exist as organic molecules may not be sufficient reason to dismiss the classification of this form of "vandalware" as a lifeform.

This paper begins with a description of how computer viruses operate, and the various ways simple viruses are structured. It then examines how viruses meet properties associated with life by researchers in the area of artificial life and self-organizing systems. This paper argues the author's view that computer viruses meeting the definition probably does not mean that they represent life, but rather that the definition is deficient.

1 Introduction

There has been considerable interest of late in computer viruses. One aspect of this interest has been to ask if computer viruses are a form of artificial life, and what that might imply.

This paper is a condensed, high-level description of computer viruses—their history, structure, and how they relate to some properties that might define artificial life. It provides a general introduction to the topic without requiring an extensive background in computer science.

The interested reader might pursue [10, 1, 2] and [5] for more detail about computer viruses and their properties. The description in this paper of the origins of computer viruses and their structure is taken from [10].

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.

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. [7] 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. [9, 8]

Worms have also appeared in other science fiction literature. Recent "cyberpunk" novels such as *Neuromancer* by William Gibson [4] 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 [10] and [2].

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 [10] or [11] 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. [1] 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 infection by the *Brain* virus in January 1986, up to April 1, 1990, the number of known viruses has grown to nearly 60 distinctly different IBM PC viruses. The problem is not restricted to the IBM PC, and now affects 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.

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.

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 due, in part, both to the greater restrictions built into the software and hardware of those machines, and to the way they are usually used. Our 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.

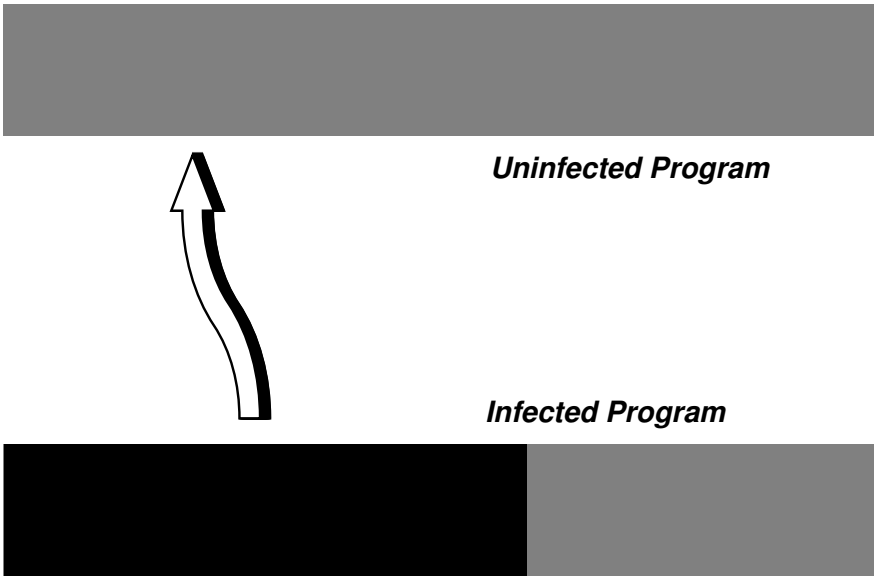

Figure 1: Shell Virus Infection

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.

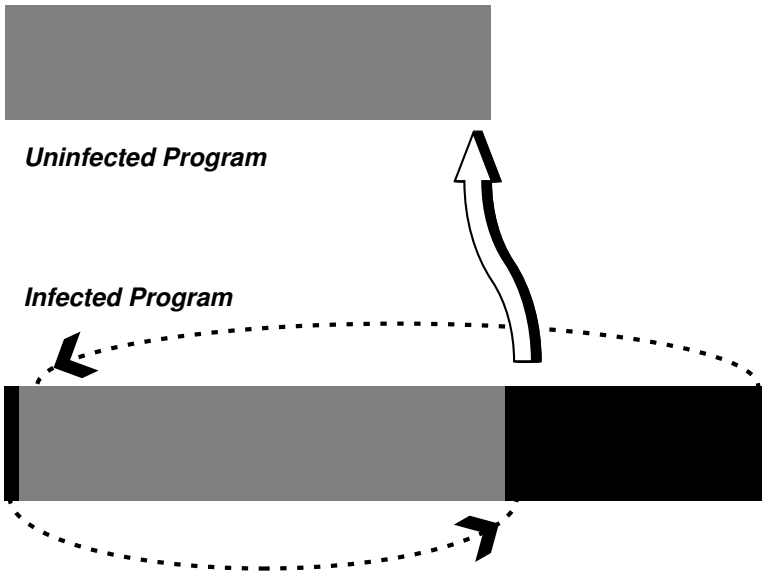

Figure 2: Add-on Virus Infection

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.

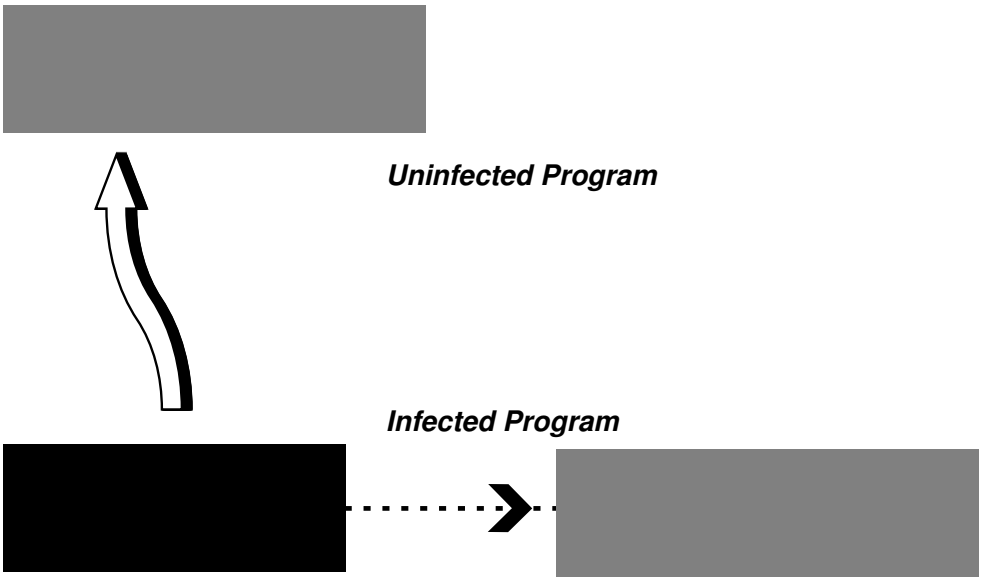

Figure 3: Intrusive Virus Infection

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 detailed description of the various points in the IBM PC boot sequence that can be infected by a virus. We 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.)

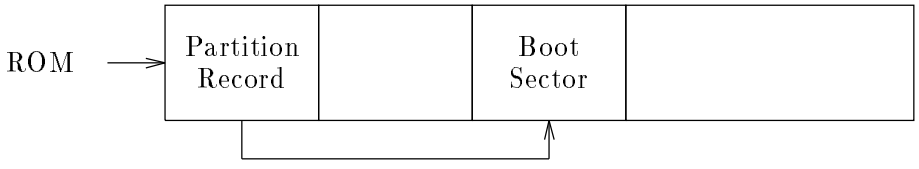

Figure 4: Hard disk before infection

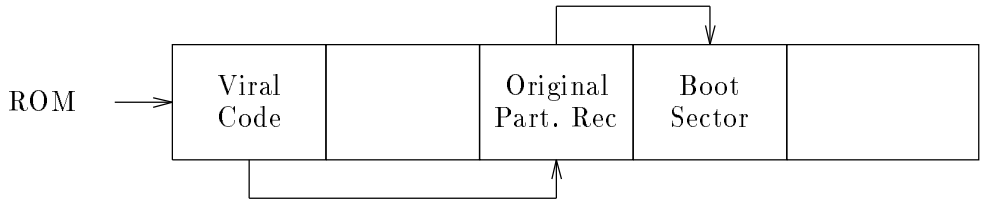

Figure 5: Hard disk after infection by New Zealand Virus

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.)

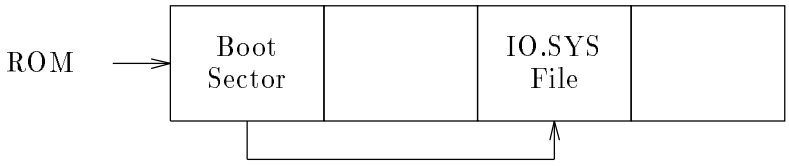

Figure 6: Floppy disk before infection

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.

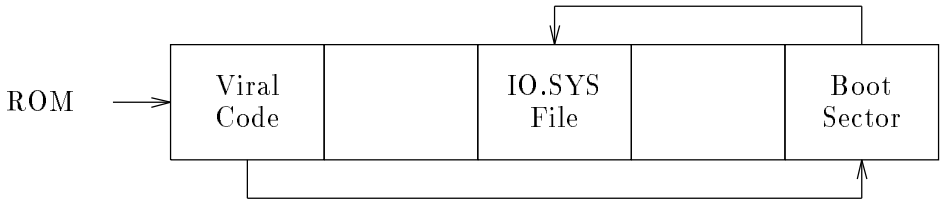

Figure 7: After Alameda Virus Infection

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.)

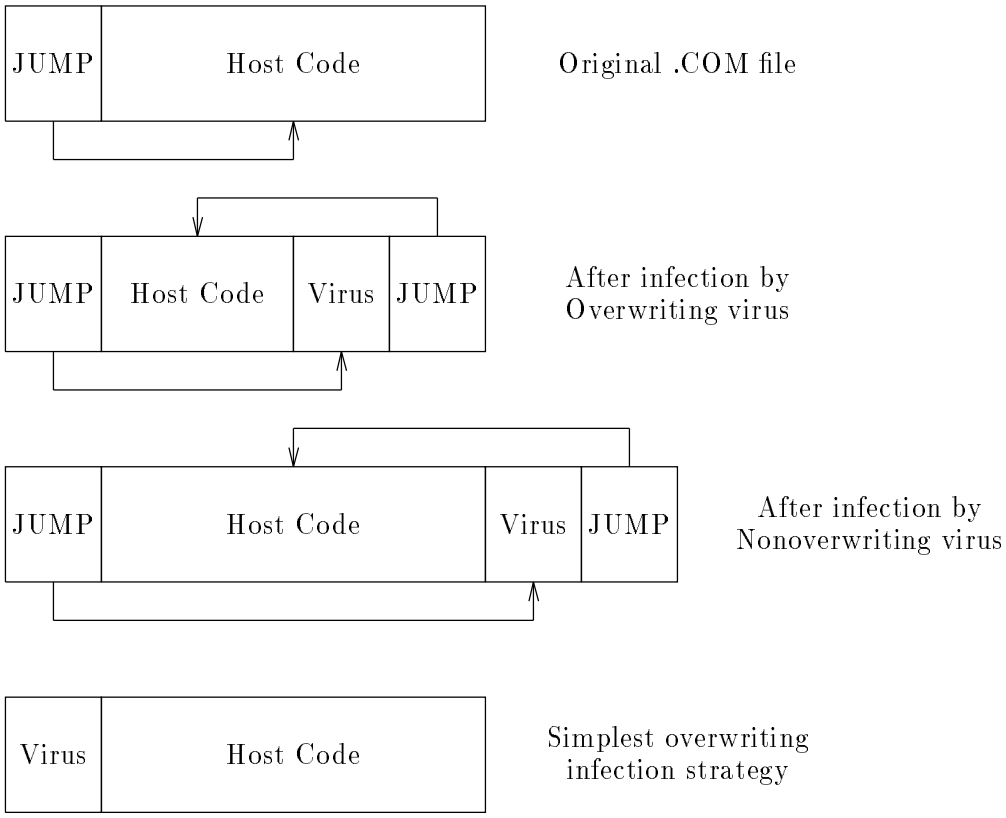

Figure 8: Infection of user applications

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 (**T**erminate and **S**tay **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.

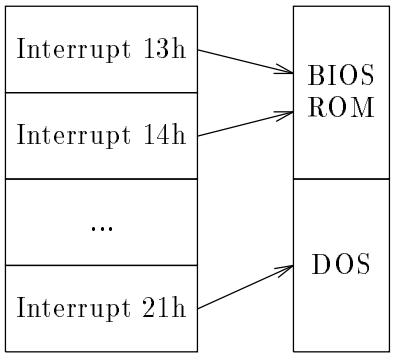

Figure 9: Normal interrupt usage

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.)

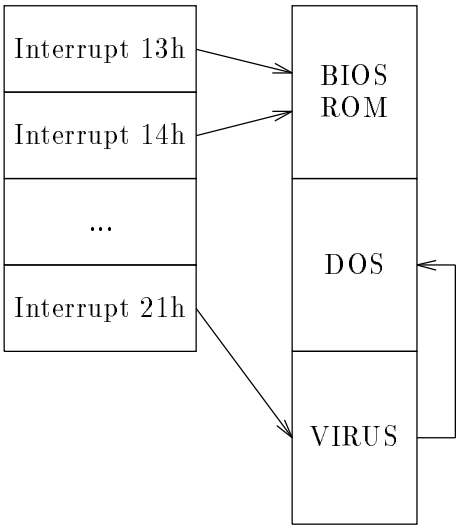

Figure 10: Interrupt vectors with TSR virus

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.

3 Viruses as Artificial Life

Now that we know what computer viruses are, and how they spread, we can ask if they represent a form of artificial life. The first, and obvious, question is "What is life?" Without an answer to this question, we will be unable to say if a computer virus is "alive."

One list of properties associated with life was presented in [3]. That list included:

- *Life is a pattern in space-time* rather than a specific material object.
- *Self-reproduction*, in itself or in a related organism.
- *Information storage of a self-representation.*
- *A metabolism* that converts matter/energy.
- *Functional interactions with the environment.*
- *Interdependence of parts.*
- *Stability under perturbations* of the environment.
- *The ability to evolve.*
- *Growth or expansion*

Let us examine each of these characteristics in relation to computer viruses.

3.1 Viruses as patterns in space-time

There is an obvious match to this characteristic. Viruses are represented by patterns of computer instructions that exist over time on many computer systems. Viruses are not associated with the physical hardware, but with the instructions executed (sometimes) by that hardware.

3.2 Self-reproduction of viruses

One of the primary characteristics of computer viruses is their ability to reproduce themselves (or an altered version of themselves). Thus, this characteristic is met.

3.3 Information storage of a self-representation

This, too, is an obvious match for computer viruses. The code that defines the virus is a template that is used by the virus to replicate itself. This is similar to the DNA molecules of what we recognize as organic life.

3.4 Virus metabolism

This property involves the organism taking in energy or matter from the environment and using it for its own activity. Computer viruses use the energy of computation expended by the system to execute. They do not convert matter, but make use of the electrical energy present in the computer to traverse their patterns of instructions and infect other programs. In this sense, they have a metabolism.

3.5 Functional interactions with the virus's environment

Viruses perform examinations of their host environments as part of their activities. They alter interrupts, examine memory and disk architectures, and alter addresses to hide themselves and spread to other hosts. They very obviously alter their environment to support their existence. Many viruses accidentally alter their environment because of bugs or unforeseen interactions. The major portion of damage from all computer viruses is a result of these interactions.

3.6 Interdependence of virus parts

Living organisms cannot be arbitrarily divided without destroying them. The same is true of computer viruses. Should a computer virus have a portion of its "anatomy" excised, the virus would probably cease to function normally, if at all. Few viruses are written with superfluous code, and even so, the working code cannot be divided without destroying the virus.

3.7 Virus stability under perturbations

Computer viruses run on a variety of machines under different operating systems. Many of them are able to compromise (and defeat) anti-virus and copy protection mechanisms. They may adjust on-the-fly to insufficient storage, disk errors, and other exceptional events.

Some are capable of running on most variants of popular personal computers under almost any software configuration—a stability and robustness seen in few commercial applications.

3.8 Virus evolution

It is here that viruses display a difference from systems we traditionally view as "alive." No computer viruses evolve as we commonly use the term, although it is conceivable that a very complex virus could be programmed to evolve and change. However, such a virus would be so large and complex as to be many orders of magnitude larger than most host programs, and probably bigger than the host operating systems. Thus, there is some doubt that such a virus could run on enough hosts to allow it to evolve.

Mutations of viruses do exist, however. There are variants of many known viruses, with as many as 15 known for some IBM PC viruses. The variations involved can be very small, on the order of two or three instructions difference, to major changes involving differences in messages, activation, and replication. The source of these variations appears to be programmers (the original virus authors or otherwise) who alter the viruses to avoid anti-viral mechanisms, or to cause different kinds of damage.

There is also one case where two different strains of a Macintosh virus are known to interact to form infections unlike the "parents," although these interactions usually produce "sterile" offspring that are unable to reproduce further. [6]

3.9 Growth

Viruses certainly do exhibit growth. Some transient viruses will infect every file on a system after only a few activations. The spread of viruses through commercial software and public bulletin boards is another indication of their wide-spread replication. One reasonable set of estimates had the number of computer virus infections in 1989 at a level 50% above the 1988 rate.[3] The number of new virus "species" reported in the first four months of 1990 has undergone a 15-fold increase over the same period in 1989. Clearly, computer viruses are exhibiting major growth.

3.10 Other behavior

As already noted, computers viruses exhibit "species" with well-defined ecological niches based on host machine type, and variations within these species. These species are adapted

[3] Personal communication, Assistant U.S. Attorney Bill Cook, quoting Bell Labs' estimates.

to specific environments and will not survive if moved to a different environment.

Some viruses also exhibit predatory behavior. For instance, the DenZuk virus will seek out and overwrite instances of the Brain virus if both are present on the same system. Other viruses exhibit territorial behavior—marking their infected domain so that others of the same type will not enter and compete with the original infection.

4 Summary and Comments

Our examination of computer viruses leads us to the conclusion that they are very close to what we might define as "artificial life." Rather than representing a scientific achievement, this probably represents a flaw in our definition. To suggest that computer viruses are alive also implies to me that some part of their environment—the computers, programs, or operating systems—also represents artificial life. Can life exist in an otherwise barren and empty ecosystem? A definition of "life" should probably include something about the environment in which that life exists.

I would also be disappointed if computer viruses were considered as the first form of artificial life, because their origin is one of unethical practice. Viruses created for malicious purposes are obviously bad; viruses constructed as experiments and released into the public domain are likewise unethical, and poor science besides: experiments without controls, strong hypotheses, and the consent of the subjects. Facetiously, I suggest that if computer viruses evolve into something with artificial consciousness, this might provide a doctrine of "original sin" for their theology.

More seriously, I would suggest that there is something to be learned from the study of computer viruses: the importance of the realization that experimentation with systems in some way (almost) alive can be dangerous. Computer viruses have caused millions of dollars of damage and untold aggravation. Some of them have been written as harmless experiments, and others as malicious mischief. All have firmly rooted themselves in the pool of available computers and storage media, and they are likely to be frustrating users and harming systems for years to come. Similar but considerably more tragic results could occur from careless experimentation with organic forms of artificial life. We must never lose sight of the fact that "real life" is of much more importance than "artificial life," and we should not allow our experiments to threaten our experimenters.

References

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

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

[3] J. Doyne Farmer and Alletta d'A. Belin. Artificial life: The coming evolution. In *Proceedings in Celebration of Murray Gell-Man's 60th Birthday.* Cambridge University Press, 1990. To appear.

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

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

[6] John Norstad. *Disinfectant On-line Documentation.* Northwestern University, 1.8 edition, June 1990.

[7] 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.

[8] 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.

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

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

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