THE PC VIRUS CONTROL HANDBOOK

A Technical Guide to Detection, Identification, Disinfection, and Investigation
Includes Model Policy and Procedures

Robert V. Jacobson

Second Edition

THE PC VIRUS CONTROL HANDBOOK

A Technical Guide to Detection, Identification, Disinfection, and Investigation
Includes Model Policy and Procedures

Robert V. Jacobson

Miller Freeman Publications, Inc., San Francisco, California

Second Edition
ISBN 0-87930-194-5

First Printing: November 1990
Printed in the United States of America
Cover design: Mark Zaremba

Please address suggestions, comments and corrections regarding the text to the author at:

International Security Technology, Inc.
515 Madison Avenue - Suite 3200
New York, New York 10022
(212) 288-3101

Acknowledgments

Information about viruses in this Handbook has been compiled from in-house tests of sample viruses using International Security Technology's **VIRUS-PRO** programs and from outside sources. In particular we want to acknowledge in alphabetical order the excellent work of Dr. Klaus Brunnstein and Ms. Simone Fischer-Huebner of the University of Hamburg, FR of Germany, David Ferbrache of Hariot-Watt University in Edinburgh, Scotland, James Goodwin, a contributor to the HomeBase BBS, Aryeh Goretsky of McAfee Associates, Santa Clara, California, Joseph Hirst of the British Computer Virus Research Center in Brighton, England, and Patricia M. Hoffman of Santa Clara, California. We are particularly grateful for the generous advice and support we received from John McAfee, Chairman of the Computer Virus Industry Association, and President of McAfee Associates. Any errors or omissions are solely the responsibility of the author.

The reader must take the information in this Handbook with a grain of salt. Every "popular" virus seems to inspire anonymous programmers to make "improvements." While a virus you encounter may appear to match one of the viruses described here, you may find that it behaves differently. Consequently, a disclaimer is in order.

DISCLAIMER

While it is hoped that the information herein is accurate, IST makes no claim as to its accuracy or completeness. **Those who use the information in this Handbook do so at their own risk, and with the understanding that neither author nor the publisher or any other person or organization is liable for any claim whatsoever resulting from use of this information.**

Introduction

The enthusiastic reception given the First Edition of The PC Virus Control Handbook was dramatic evidence of the seriousness of the threat to PCs posed by computer viruses. This Second Edition includes a new section in Chapter One which discusses the characteristics of so-called stealth viruses and the special difficulties they present. The reader will also find expanded coverage of how DOS uses the Partition Table to control access to fixed disk drives and the workings of the DOS **FDISK**, **SYS** and **FORMAT** programs as an aid in understanding how to disinfect boot sectors and Partition Tables.

The information in this Handbook will help users of IBM-compatible personal computers (PCs) to cope with the growing incidence of virus infections. As we predicted in the First Edition, there has been an explosive growth in the number of PC viruses during 1990. We have added descriptions of more than 50 new viruses to the Second Edition. This makes it clear that there is no reason to assume that viruses will go away spontaneously. Until PC users are able to trace viruses back to their sources, the number of infections can be expected to grow month by month.

How To Use This Handbook.

This Handbook presents the information you need to deal with virus infections of your organization's PC's. Four topics are covered.

Virus Technology.

Chapter One defines the term computer virus, and describes how viruses spread from one PC to another. You will also find an explanation of the four different kinds of virus infection mechanisms, and the special problems created by the stealth virus technology. Finally there is a discussion of virus detection techniques.

Identifying Viruses.

Chapter Two begins with a procedure for responding to reports of virus infections, and checklists of overt indicators which you can use as a starting point in identifying viruses. Indicators of infection are linked to detailed descriptions of more than one hundred viruses. Only viruses which attack IBM-compatible personal computers running under MS- or PC-DOS are included. Viruses rumored to exist but which have not been "captured" and analyzed by qualified researchers are not included.

Most infections are caused by a few well-known viruses described in **Chapter Two**. Regrettably it is relatively easy to modify the virus code in an infected program to change its infection signature or logic bomb trigger. Consequently, virus variants may differ from the information given here. One must treat any identification as tentative, and assume that damage effects may not be as listed.

If you discover a virus infection, do not delay disinfection because the virus is said to be harmless. Even if a virus is thought not to have a logic bomb with the implication that it is harmless, it may still corrupt files and prevent them from executing properly.

Since many viruses are known by more than one name, there is a cross reference to more than 200 virus names in Appendix A. The cross reference to the identifying codes used by the popular McAfee Associates **ViruScan** software in Appendix B is a new addition to ***The PC Virus Control Handbook.*** This cross reference makes it easy to relate the viruses detected by the McAfee programs with the virus descriptions in **Chapter Two.** Appendix C lists a number of helpful virus books.

How To Recover from and Investigate an Infection.

In **Chapter Three** you will find detailed instructions on how to disinfect PCs and how to investigate the source of the virus infection. Experience shows that most organizations must make three attempts before disinfection is successfully. Reinfection within a month is common. A careful, systematic approach is essential to success.

Anti-Virus Policy and Procedures.

For more than two years experts have been offering advice on how to deal with the threat of virus infections. The growing number of infections makes it clear that simplistic lists of Dos and Don'ts are not enough. In **Chapter Four** we present prototype statements of realistic anti-virus policy and procedures. These are also available on a diskette. See the order form in the back of the book for details.

The **VIRUS-PRO** Software Package.

International Security Technology, Inc. is the developer of the **VIRUS-PRO** software package. **VIRUS-PRO** provides facilities to detect and recover from virus infections with a minimum of lost data. Various **VIRUS-PRO** programs are mentioned to illustrate detection, recovery and investigation techniques.

It should not be inferred from this that **VIRUS-PRO** is required to use the information presented here. Several different kinds of anti-virus software are available from reputable software vendors. A combination of software techniques will provide the most effective defense against viruses. It is unwise to depend on only one approach. Please refer to Section 1.4 and Chapter 4 for more about this topic.

Robert V. Jacobson
New York City, September 1990

Table of Contents

Chapter One
Virus Technology

This Handbook addresses the computer viruses which infect IBM-compatible Personal Computers running under the MS-DOS and PC-DOS operating systems for the simple reason that these viruses are causing the greatest amount of damage. We will refer to these PC/MS-DOS viruses simply as viruses, the computers as PCs, and the operating system as DOS.

To deal intelligently with PC viruses, it is important for you to understand how they function, where they hide, and how they spread from one PC to another. This Chapter covers these topics. Chapter Two has the information you need to identify known viruses. Chapter Three presents step-by-step instruction for recovering from a virus infection and investigating its source. In Chapter Four you will find anti-virus policy and procedures which you can adapt to your organization.

There is no official body which assigns names to viruses. Researchers choose names when they discover new viruses. Names are usual selected for one of four reasons: (1) the size increase of infected programs, (2) a characteristic of the observable effect of the virus, (3) wording of an overt message or internal ASCII string, or (4) the locale where the virus was first "captured." A popular virus may be referred to by more than one name. To facilitate reference to specific viruses, we have assigned an identifying number to each virus in the form: IST# n.nnnn, e.g., IST# 1.3760. The leading digit identifies the infector type as follows: 1.0 - program infectors, 2.0 - boot sector infectors, 3.0 - Partition Table infectors, and 4.0 - multiple infector viruses. The four types are defined in Section 1.3. Appendix A on page 151 of this Handbook is a cross reference of virus names and these numbers.

Section 1.1 A Introduction to Virus Technology.

We are using the term **computer virus** to refer specifically to program code segments which have these characteristics:

1.) The virus code resides in a "host" program which performs a useful function having nothing to do with the virus.

2.) When the virus code segment is executed as a result of executing the host program, the virus code seeks out at least one other program, the "target" program, and attempts to copy itself into the target program. The virus designer has two objectives. The target program must continue to function correctly after being infected. The virus code must execute each time the target program executes. In short the target program acquires the same virus characteristics as the host program. Some viruses make themselves memory resident, and continue to function after the host program has completed execution.

3.) In addition to the infector code, the virus code segment may also include (a) an indicator which appears each time the virus code executes, e.g., the "Bouncing Ball" virus, IST# 2.0300, and (b) a logic bomb which when triggered, performs an action such as reformatting the hard drive, e.g., the "Dark Avenger" virus, IST# 1.5200.

Once a target program has been infected, its virus code can infect yet another target program and so on. Thus a virus spreads from program to program, and from PC to PC. This ability to propagate is the essential characteristic of computer viruses.

Figure 1 illustrates the concept of a virus spreading from program to program in a logical drive. Step 1 shows four uninfected programs: A, B, C, and D, stored on a disk. In Step 2 program E which is virus infected, is added to the disk and executed. In Step 3 we see that E has infected program B, and in Step 4, B has infected D. Note that the infections have increased the size of B and D. This is typical of program infector viruses.

While some viruses appear to be harmless, others include logic bombs which can destroy programs and data files. The danger to an organization is that all of its PCs will become infected resulting in the *simultaneous loss* of functionality when

the logic bomb is triggered. This is not a theoretical risk. Simultaneous disablement of thousands of PCs has already occurred. In the final analysis, there is no such thing as a harmless or beneficial virus. Even the most benign may corrupt an infected program or interfere with normal operation of the PC.

There are two basic ways in which program infector viruses select programs to infect. One virus type searches one or more DOS directories for executable programs to infect. The other type makes itself memory resident when it is activated. It becomes what is called a Terminate and Stay Resident (TSR) program, and "hooks into" DOS so it can monitor the initiation of new processes. When the typical memory resident virus sees that a program is about to be executed, it infects the program. The virus will continue to infect other programs until the PC is rebooted.

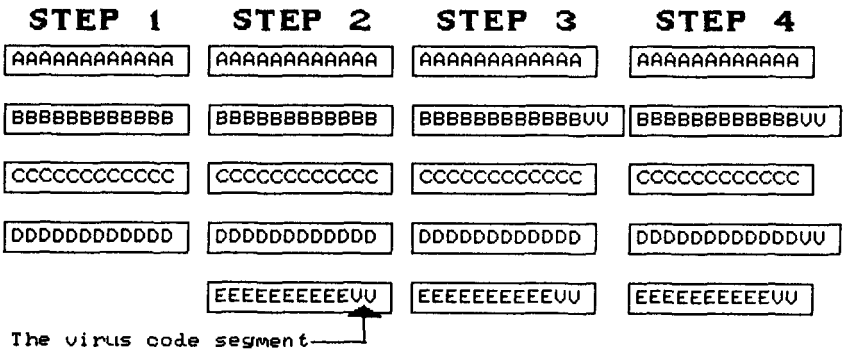

Figure 1. Step-by-step spread of a virus from E to B to D.

Most viruses use the same basic technique to inject themselves into targeted programs. First the virus injects a copy of itself to the target program. To simplify this process for EXE files which are relocated at load-time, the virus may align the copy along a paragraph (16 byte) boundary of the target program. This insures that the virus offsets will be correct. The virus segment will automatically be adjusted at load-time. As a result of this adjustment the increase in the program size will vary over a 16 byte range. IST#'s 1.14000, and 1.18000 are examples of this effect.

Once the virus has injected itself into the target program, it makes use of the fact that PC programs all have a defined entry point. This is the place in the program where the first instruction is located. The virus changes the jump instruction

at the entry point to transfer control to the virus near instead of to the next instruction of the target program. This will cause the virus code to execute first when the target program is run. A few viruses inject themselves into the beginning of the program, after copying the overwritten program code to the end of the program. IST# 1.1600 is an example.

Finally, the virus adds instructions at the end of the virus code to cause execution to continue at the entry point of the target program, as it did prior to infection. If the virus does all this correctly, the target program will still function normally after the virus code segment executes.

Some viruses use slightly different infection techniques. For example, as you may recall, DOS assigns disk storage space to files in "clusters" where a cluster is two or more sectors. (The number of sectors per cluster is set at the time a disk is formatted. The number will always be a power of two, e.g., two, four, eight, etc.) Unless a program is exactly an integral number of clusters long, there will be unused space in the last cluster assigned to a program. A few viruses check before infecting a program to see if this space is large enough to store the virus. If it is, the virus will hide itself in the extra space.

One virus, IST# 1.0000, specifically targets the DOS **COMMAND.COM** program and hides in an internal buffer. This allows it to infect the program without changing its size.

A few viruses, such as IST# 1.0100, simply overwrite the beginning of the target program. Of course, this damages the program so it cannot execute. As a result these viruses immediately call attention to themselves, and are not likely to become widespread.

Boot sector and Partition Table viruses are a special case. These viruses insert themselves into the boot sector or Partition Table code segments of disks and gain control of the PC at the time it is booted up. See Sections 1.2, 2.4, and 2.5 for details. Recently viruses have appeared which use two or more types of infectors. For example when the Ghost virus, IST# 4.0010, executes, it will attempt to infect the boot sector of the current logical drive, and also will attempt to infect COM programs. This gives the virus two different ways to spread.

To infect a target program without damaging it program infector viruses must identify the end of the target program correctly. Apparently some viruses when infecting programs that use overlays, do not do this correctly. As a result the virus may be injected into the middle of the target, rather than at the end. As a result it overwrites some of the target program

code. Of course, the program will crash when control flows randomly from the program into the virus code. As a result some infected programs cannot be repaired by "clean-up" programs. If one does a before-and-after **VIRUS-PRO** scan and analysis of the "repaired" program, the truncation will be quite apparent. These programs can only be recovered by reinstalling uninfected copies.

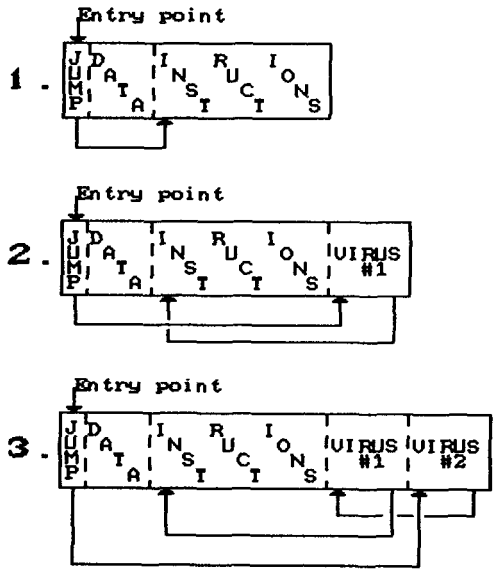

Figure 2. How a virus infects a program.

Figure 2 shows the typical program infection process graphically. At 1 in the Figure we see the uninfected program. Notice that the entry point (the point at which execution of the program begins) has a jump instruction which transfers control to the next instruction of the program. The arrows represent the flow of control. At 2 the virus has added itself to the end of the program, and has changed the entry point jump instruction to transfer control to the first instruction of the virus. Finally the virus has put a jump instruction at the end of its own code segment pointing back to the first instruction of the target program. Thus, when the target program runs, the virus code executes first, and then the target program code executes. In 3 a second virus has used the same technique to add itself to the program. Now the execution sequence is: virus

#2, virus #1, target program.

The "ideal" virus (from the virus designer's point of view) *will avoid detection until its logic bomb is triggered.* Avoiding detection can be a challenge for the virus designer. For example, some early viruses could not tell if the target program had already been infected, and would inject another copy of themselves into the target. As a result the target would increase in size each time it was infected until it becomes too big to load into memory. This would cause the PC owner to realize that something was wrong before the logic bomb was triggered. Sometimes because of a design bug a virus may not perform the infection correctly so that the infected program is corrupted and causes the PC to crash. For example, by appending itself to a large COM file, a virus may make the file size exceed the 64K limit imposed by DOS, e.g., IST# 1.5000.

Virus designers soon realized that a mechanism was needed to avoid repeat infections. Several techniques have been used. The simplest is to put a marker character string at a predetermined location in the target program. If the virus finds its own marker in a target program, it assumes that it has already infected the program and aborts the infection process. Another marker technique, used by the "Vienna" virus, IST# 1.1600, is to set the seconds field of the infected program's time-of-creation directory entry to 62, a value which never appears normally.

By definition viruses *change* their targets, and it is these changes, and to some extent the infection markers, which enable us to detect and identify virus infections. Please refer to Section 1.4 on page 26 for more about detecting virus infections. You may also want to consult the books listed in the Bibliography in Appendix C for more information.

Section 1.2 Where Viruses Hide.

In this Handbook, we classify viruses based on the three places where viruses can hide in a PC since this determines the infection recovery procedure. Bearing in mind that by definition a virus is an executable code segment, *viruses can only exist in the context of executable code which can be modified.* By following the process whereby an IBM-compatible PC begins operation ("is booted up"), we can identify the three contexts which meet these requirements. These are:

1) The executable program files. These are files with the extensions **COM, EXE, BAT, SYS** and program overlay files with extension such as **OVR** and **OVL**. The three DOS system programs are a special subset of these programs because they execute before any other programs, e.g., virus detection programs, are executed.

2) The boot strap program which is a part of every logical drive, and[1]

3) The Partition Table program stored in the Partition Table which is a part of every fixed disk,

It is generally understood that executable programs can become virus infected. The Partition Table and boot strap programs are less well known as virus infection sites. In general these later two are the targets of the more dangerous virus, and so it is important to understand what these two special programs do.

Section 1.2.1 The Boot Strap Program.

At the time DOS formats a logical drive it divides the drive into storage units called sectors. In almost every case a DOS sector stores 512 bytes of data. DOS designates the first sector on every logical drive, e.g., the A drive, the C drive, etc., as the "boot sector". The boot sector is created by DOS when the logical drive is formatted using the DOS **FORMAT** program. Here is what the **FORMAT** program does:

1) It formulates the bytes needed for an appropriate boot sector and copies them to the zero sector of the logical drive

2) It initializes the File Allocation Tables (FATs) and root directory, and copies them to the logical drive. The **FORMAT** program adjusts the FAT and root directory to match

---

1. If the difference between physical fixed disks and logical drives is not clear, simply bear in mind that DOS stores data and program files in drives. These are the "logical" representation of hardware devices which receive and process commands from the CPU via the bus to store and retrive data. Controllers are plugged into the bus. The controllers receive commands to read and write disks and other media which store data. The system software assumes that the controller with address number 00 has floppy diskette drives connected to it, and that the controller with address 80 hex has fixed disk drives connected to it. DOS permits only one logical drive on a floppy disk, but fixed disks may be partitioned into more than one logical drive.

the storage capacity of the drive being formatted. Note that the FATs might better be called the Cluster Allocation Tables since each FAT entry represents a Cluster (group of contiguous sectors). A cluster is the smallest unit of disk space DOS can allocate to a file. Each FAT entry shows the status of the corresponding cluster: unused, the number of the next cluster in a chain of linked clusters which store a file, the last cluster of a chain, or a bad (unreadable) cluster.

3) It verifies that all of the remaining clusters on the drive are error-free. If a bad cluster is detected, its status flag in the FATs is set to "bad" so DOS will not use it for data storage.

The boot sector itself (item 1 above) has two elements. The first is a table of parameters that DOS uses to manage data storage on the drive. The table includes: bytes per sector, sectors per cluster, total sectors, and so on. All early versions of **FORMAT** put a set of parameters into the first 30 bytes of the boot sector which define basic storage parameters. DOS version 3.+, DOS version 4.+, and proprietary formatting programs create expanded parameter tables with added parameters. Some OEM versions of DOS create a nonstandard parameter table if the drive is being formatted with a capacity greater than than 32 Mbytes.

The second boot sector element is the boot strap program[2], a master copy of which is stored in the **FORMAT** program. A hex-dump of a typical boot sector appears in Figure 3. The boot strap program begins at address 000036 hex. You can see the OEM label on the first line. The boot-up messages begin at address 000179 hex. As a part of the formatting process, the program is merged with the parameter table and copied into the zero logical sector of the drive. (DOS starts counting sectors at zero instead of one.)

---

2. It is traditional to refer to this kind of program as a "boot strap" program because the PC can be thought of as "pulling itself up by its boot straps" when it loads the DOS operating system.