infections. Other file changes, notably configuration variations, will not trigger the alarm. If, however, you should ever desire a full check that detects ANY file changes, this option takes care of it. Be aware that this option slows down the system considerably, so we don't recommend its use in normal circumstances.

secure (s).

TbCheck normally asks whether you want to continue or cancel when a file has been changed or when there is no checksum information available. In a business environment it may be unwise to leave such decisions to employees. Option SECURE makes it impossible to execute new or unknown programs, or programs that have been changed.

NOTE:
Be aware that the SECURE option also disables the OFF and REMOVE options.

3.5.4 Understanding the Scanning Process

This section adds to your knowledge of TbCheck by explaining a little more about the scanning process.
Whenever a program wants to execute, TbCheck steps in to see if it really has the authority to do so. During that time it displays the message "*Checking*" in the upper left hand corner of the screen. TbCheck operates at lightning speed, so the message appears only momentarily.

TbCheck quickly checks a program when the program loads. If TbCheck detects that a file has changed, a notification message appears. At this point, you can choose to either continue, or to abort the program's execution.

If there is no information in the ANTI-VIR.DAT file about the program, TbCheck also informs you of this. You can either choose to continue without checking, or to abort the program's execution.

TIP:
You can prevent users from executing unauthorized software by using

Understandably, many users wish to test the product they are using. In contrast to a word processor, for example, it is very difficult to test a smart integrity checker like TbCheck. You cannot change a random 25 bytes of an executable file just to find out whether TbCheck detects the file change. On the contrary, it is very likely that TbCheck will NOT detect it because the program checks only the entry area of the file, whereas the changed bytes might reside in another location within the file. But again, if a virus infects the file, this entry area will definitely change, so this is perfectly adequate to detect all infections.

3.6 Using TbClean

In case a virus infects one or more files, and you wish to remove the virus from those files (for example, in case you do not have a clean backup of the files), you can use TbClean. TbClean is the program that can remove viruses from infected files, even without knowing the virus itself. This section explores TbClean.

3.6.1 Understanding TbClean

TbClean isolates viral code in an infected program and removes it. It is then safe to use the program again, since TbClean securely eliminates the risk of other files becoming infected or damaged.

Understanding the Repair Cleaner

TbClean works differently from conventional virus cleaners because it does not actually recognize any specific virus. TbClean's disinfection scheme is unique, employing ThunderBYTE's heuristic (learn as you go) technology so that it works with almost any virus.

Actually, the TbClean program contains two cleaners: a "repair" cleaner, and a "heuristic" cleaner. The repair cleaner needs an ANTI-VIR.DAT file generated by the TbSetup program before the infection occured. This ANTI-VIR.DAT file contains essential information such as the original file size, the bytes at the beginning of the program, a cryptographic checksum to verify the results, etc. This information enables TbClean to disinfect almost every file, regardless of the specific virus that has infected it, even if it is unknown.

Understanding the Heuristic Cleaner

In the heuristic cleaning mode TbClean does not need any information about viruses either, but it has the added advantage that it does not even care about the original, uninfected state of a program. This cleaning mode is very effective if your system becomes infected with an unknown virus and you neglected to let TbSetup generate the ANTI-VIR.DAT files in time.

In the heuristic mode, TbClean loads the infected file and starts emulating the program code to find out which part of the file belongs to the original program and which belongs to the virus. The result is

successful if TbClean restores the functionality of the original program, and reduces the functionality of the virus to zero.

NOTE:
This does not imply that the cleaned file is 100% equal to the original. Please read on.

When TbClean uses heuristic cleaning to disinfect a program, the file most likely will not be exactly the same as in its original state. This does not imply a failure on TbClean s part, nor does it mean the file is still infected in some way.

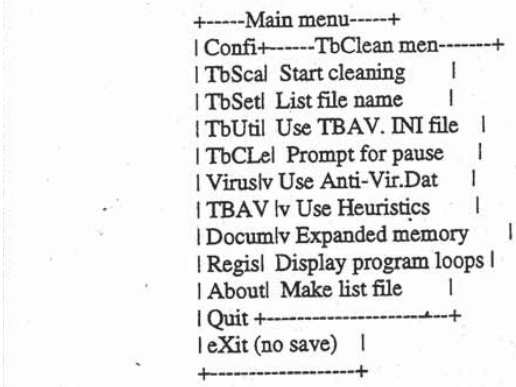
It is actually normal that the heuristically cleaned file is still larger than the original. This is normal because TbClean tries to be on the safe side and avoids removing too much. The bytes left at the end of the file are dead code, that is, instructions that will never execute again since TbClean removes the jump at the beginning of the program. If the cleaned file is an EXE type file, it is likely that some bytes in front of the program (the EXE-header) are different. There are several suitable solutions for reconstructing the EXE-header, so TbClean cannot, of course, know the original state of the program. The functionality of the cleaned file will nevertheless be the same.

NOTE:
This applies only to heuristic cleaning. If there is a suitable ANTI-VIR.DAT record available, the cleaned program will normally be exactly the same as the original clean file.

It's also possible for a virus to infect a file with multiple viruses, or multiple instances of the same virus. Some viruses keep on infecting files, and in such cases the number of infected files keeps growing. If TbClean used its heuristic cleaning mode, it is very likely that TbClean removed only one instance of the virus. In this case, it is necessary to repeat the cleaning process until TbClean reports that it cannot remove anything else.

3.6.2 Working with the TbClean Menus

Selecting TbClean from TBAV's Main Menu displays the following menu:



We'll now explore these menu options.

The "Start Cleaning" Option

After tracking one or more viruses, all you should do is select the Start cleaning option. After specifying the relevant filename, TbClean goes into action. Before beginning, however, you can select various parameters. We will explore these in the following sections.

The "List File Name" Option

By selecting this option you can specify a filename to use as a list file (see also the Make list file option below).

The "Use TBAV.INI File" Option

If you enable this option, the TbClean configuration values, saved in the TBAV.INI file, will also be valid if you run TbClean from the DOS command line. Be careful, however, since if you specify options in the TBAV.INI file, you cannot undo them on the command line. See the "Configuring TBAV" section of Chapter 1 for details about TBAV.INI.

The "Prompt For Pause" Option

This option instructs TbClean to stop disassembling information after

The "Use ANTI-VIR.DAT" Option

If you turn this option off, TbClean acts as if there were no ANTI-VIR.DAT records available and therefore performs heuristic cleaning.

The "Use Heuristics" Option

If you turn this option off, TbClean does not try to apply heuristic cleaning, even when there are no ANTI-VIR.DAT records available.

The "Expanded Memory" Option

If you select this option, TbClean detects the presence of expanded memory and uses it in heuristic mode. You might want to disable EMS usage if it is too slow or if your expanded memory manager is not very stable.

The "Show Program Loops" Option

By default TbClean keeps track of looping conditions to prevent repetitive data from appearing on your screen thousands of times. If you select this option, TbClean "works out" every loop.

CAUTION:
Using this option drastically reduces TbClean's performance speed. Also, do not combine this option with the "Make list file" option, because the list file might grow too big

The "Make List File" Option

Selecting this option instructs TbClean to generate an output file with a chronological disassembly of the virus being removed.

Maximizing TbClean

Now that you know how to use TbClean's menus, you can more easily understand the power of using it from the command line.

3.6.3 Using TbClean Command Line Options

When you run TbClean from the DOS command line, it recognizes command line options (often called "switches" in DOS terms). These options appear as "key-words" or "key-letters." The words are easier to memorize, so we will use these in this manual for convenience.

You can maximize TbClean's performance by using its command line options. The following table lists these options:

option	parameter	short	explanation
help		he	display on-line help
pause		pa	enable pause prompt
mono		mo	force monochrome display output
noav		na	do not use ANTI-VIR.DAT records
noheur		nh	do not use heuristic cleaning
noems		ne	do not use expanded memory
showloop		sl	show every loop iteration (slow!)
list	[=<filename>]	li	create list file

The explanations in the above table serve as a quick reference, but the following descriptions provide more information about each option.

TIP:
Remember that you can display these options from the command line by entering TBCLEAN ?.

help (he).

Specifying this option displays the above options list.

pause (pa).

This option instructs TbClean to stop disassembling information after each full screen, enabling you to examine the results. The PAUSE option is available for registered users only.

mono (mo).

This option enhances the screen output on some LCD screens or color-emulating monochrome systems.

noav (na).

If you specify this option, TbClean acts as if there were no ANTI-VIR.DAT records available and therefore performs heuristic cleaning.

noheur (nh).

If you specify this option, TbClean does not try to apply heuristic cleaning, even when there are no ANTI-VIR.DAT records available.

noems (ne).

If you specify this option, TbClean does not detect the presence of expanded memory and use it in heuristic mode. You might want to disable EMS use if it is too slow, or if your expanded memory manager is not very stable.

showloop (sl).

By default TbClean keeps track of looping conditions to prevent repetitive data from appearing on your screen thousands of times. If you select this option, TbClean "works out" every loop.

CAUTION:
Using this option drastically reduces TbClean's performance speed. Also, do not combine this option with the "Make list file" option, because the list file might grow too big

list [=<filename>] (li).

This option instructs TbClean to generate an output file with a chronological disassembly of the virus being removed. The LIST option is available for registered users only.

Here are two examples of using TbClean from the command line:

1. This command:

TBCLEAN VIRUS.EXE

instructs TbClean to make a backup of the file VIRUS.EXE using the name filename VIRUS.VIR, and then disinfect VIRUS.EXE.

2. This command:

TBCLEAN VIRUS.EXE TEST.EXE

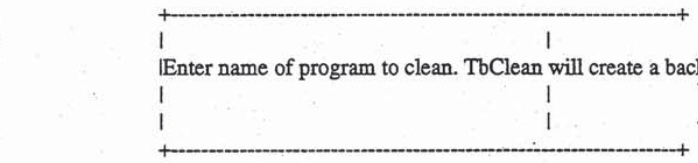
instructs TbClean to copy the file called VIRUS.EXE to the new filename TEST.EXE and then disinfect TEST.EXE.

3.6.4 Understanding the Cleaning Process

TbClean's cleaning process is extremely important. To better illustrate it, let's look at a sample file cleaning.

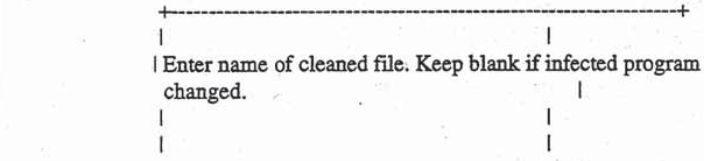
Assume you want to clean a file called COMMAND.COM, which resides in the TMP directory on drive G. To do so, you would follow these steps:

1. Select the "Start cleaning" option on the TBAV menu. The following window appears:



The ThunderBYTE utility cleans on a file-by-file approach; that is, it cleans one file, verifies the result, and continues on to the next file. This helps you keep track of which file is clean, which file is damaged and should be restored from a backup, and which file is still infected.

2. Specify the name of the file. In this case, you would type G:\TMP\COMMAND.COM and press ENTER. The following window appears:



3. Type a new file name and press ENTER. In this case, we'll use G:\TMP\TEST.EXE. TbClean now begins the cleaning process.

By specifying a different name you ensure that the cleaned file cannot overwrite the original file. In this example TbClean copies COMMAND.COM to TEST.COM and disinfects TEST.COM.

If you do not specify a backup filename, TbClean creates a backup with the .VIR extension. In this example, the TbClean would copy the original file to COMMAND.VIR and then clean COMMAND.COM.

During the cleaning process, TbClean displays as much information as possible about the current operation, as illustrated below. All the major actions appear in the emulation window at the lower half of the screen, which displays a disassembly and the register contents of the program under scrutiny, as well as a progress report. The top-left and top-right status windows reveal useful details of the infected file and (if TbClean can find a suitable ANTI-VIR.DAT file) the file's original status. You can abort the cleaning process by pressing Ctrl+Break.

```
+-------------------------------------------------------------+
| Thunderbyte clean utility     (C) 1992-95 Thunderbyte B.V.  |
+--------Infected state----------++--------Original state-------+
| Entry point (CS:IP)  34BF:0012 || Entry point (CS:IP) 34BF:0012|
| File length                || File length        UNKNOWN! |
| Cryptographic CRC    9F90F52A || Cryptographic CRC   UNKNOWN! |
+--------------------------------++-----------------------------+
|                                |
| Starting clean attempt. Analyzing infected file...           |
| Anti-Vir not found: original state unknown. Trying emulation... |
| Emulation terminated:                                        |
|                                |
| G:\VIRUS\COMMAND.COM                                         |
| CS:IP   Instruction    AX BX CX DX DS SI ES DI SS SP |
| 9330:0101  mov ah,40   FFFE9330FFFFEFFFD382FFEDEFFEFFFF9520007E|
| 9330:0103  mov bx,0002  40FE9330FFFFEFFFD382FFEDEFFEFFFF9520007E|
| 9330:0106  mov cx,0016  40FE0002FFFFEFFFD382FFEDEFFEFFFF9520007E|
| 9330:0109  mov dx,cs    40FE00020016EFFFD382FFEDEFFEFFFF9520007E|
| 9330:010B  mov ds,dx    40FE000200169330D382FFEDEFFEFFFF9520007E|
| 9330:010D  mov dx,0117  40FE00020016933093309330FFEDEFFEFFFF9520007E|
| 9330:0110  int 21       40FE00020016011793309330FFEDEFFEFFFF9520007E|
| 9330:0112  mov ax,4CFF  40FE00020016011793309330FFEDEFFEFFFF9520007E|
| 9330:0115  int 21       4CFF00020016011793309330FFEDEFFEFFFF9520007E|
| 9330:0115  <End of emulation>                                |
+-------------------------------------------------------------+
```

A successful purge is not the end of the story! Your job is only partially complete. Some viruses damage data files. They could randomly change bytes on your disks, swap sectors, or perform other nasty tricks. A cleaning utility can never repair data!

4. Check your data files thoroughly and consult a virus expert to find out what the virus is capable of doing. If there is any doubt, restoring the data is definitely the most reliable option.

WARNING:

Under no circumstances should you continue to use cleaned software! Cleaning is a temporary solution that simply enables you to delay a large restore operation until a more practical time. You should never rely on a cleaned program for any length of time. This is not a criticism of anti-viral cleaning agents. If your data is valuable to you, you should care for it as much as possible, and sticking to original software is simply an elementary precaution. In other words, restore the original programs as soon as possible!

3.6.5 Understanding Cleaning Limitations

Although TbClean has a very high success rate and is able to clean programs that other cleaners refuse to process, it simply cannot remove all viruses and cannot clean every file. Examples of computer viruses that TbClean (or other virus cleaners) cannot clean include:

Overwriting viruses. This type of virus does not add itself to the end of the original program, rather it copies itself over the original file. Further, it does not attempt to start the original program but simply hangs the machine or returns you to DOS after it activates. Since it overwrites the original file, no cleaner can restore the file.

Some encrypted viruses. TbClean is usually able to decrypt the virus. However, some viruses use anti-debugger features that TbClean cannot yet cope with (but we re working on it!).

The construction of some program files makes them impossible to clean, making reinstallation the only option. Some of these file types include:

EXE-programs with internal overlays. TbScan marks these files with an "i" flag. Any infection is sure to cause major damage to these files. Some viruses recognize such programs and do not

infect them, but most viruses infect these programs anyway and corrupt them. No cleaner can repair this kind of damage.

Programs with sanity check routines. Some programs (mostly anti-virus software or copy-protected programs) perform their own kind of sanity check. Heuristic cleaning of an infected program normally results in a program that is not physically identical to the original. So, although TbClean removes the virus from the program and the program is functionally identical to the original, the program's internal sanity check usually detects the slight changes and aborts the program.

Cleaning Multiple Files

TbClean has no provisions for cleaning multiple programs in one run. There are two reasons for this omission:

1. TbClean cannot search for viruses automatically since it does not know any virus.

2. We recommend that you clean the system on a file-by-file basis. Clean one file, verify the result, and go on to the next file. Again, this helps you keep track of which files are clean, which files are damaged and should be restored from a backup, and which files are still infected.

3.7 Using TbMem

TBAV provides three extra utilities that help you build a massive security wall around your computer system. This set includes: TbMem, TbFile and TbDisk. In this section, we'll introduce these three utilities collectively as a set and then examine each individual utility.

3.7.1 Introducing the TbMem, TbFile & TbDisk Utilities

As the old saying goes, An ounce of prevention is worth a pound of cure, and the computer virus threat gives this old saying new meaning. TBAV is the best product on the market for removing viruses, but if this is all it did, it would be of little use. It's much wiser to prevent virus infection than wait until you get one and remove it.

This is where a set of three small memory-resident (TSR) programs come in. These utilities are shipped with TBAV for DOS; they monitor specific areas of your system and protect against virus infection. These three utilities are:

TbMem.

This program detects attempts by programs to remain resident in memory and ensures that no program can remain resident in memory without permission.

TbFile.

This program detects attempts by programs to infect other programs.

TbDisk.

This program detects attempts by programs to write directly to the disk (bypassing DOS), attempts to format disks, and other such destructive actions.

3.7.2 Loading TbMem, TbFile and TbDisk

The TbMem, TbFile and TbDisk programs load in the same way. The following sections contain specific information on each of the programs, but here we present loading information that is common to all of them.

CAUTION:
You must load TbDriver before you can load any of the TbMem, TbFile or TbDisk utilities. These utilities will refuse to load without it.

There are three possible ways to load TbMem, TbFile or TbDisk. Please note that we call the programs TbXXX here. Naturally, you will replace the XXX with either Mem, File, or Disk when you load each utility.

1. From the DOS prompt or within the AUTOEXEC.BAT file:
   <PATH>TBXXX

2. From the CONFIG.SYS file as a TSR (DOS 4 or higher):
   INSTALL=<PATH>TBXXX.EXE
   The INSTALL= CONFIG.SYS command is NOT available in DOS 3.xx.

3. From the CONFIG.SYS as a device driver:
   DEVICE=<PATH>TBXXX.EXE

NOTE:
Executing one of the utilities TbMem, TbFile or TbDisk as a device driver does not work in all OEM versions of DOS. If it doesn't work, use the INSTALL= command or load the desired program from within the AUTOEXEC.BAT. TbMem, TbFile and TbDisk should always work correctly after being started from within the AUTOEXEC.BAT file. Also, unlike other anti-virus products, you can load the ThunderBYTE Anti-Virus utilities before starting a network without losing the protection after the network starts.

In addition to the three loading possibilities, if you are using DOS version 5 or above, you can load the TbMem, TbFile or TbDisk programs in an available UMB (upper memory block) from AUTOEXEC.BAT using the following command:

```
LOADHIGH <PATH>TBXXX.EXE
```

You can load TbMem, TbFile or TbDisk high from within the CONFIG.SYS using the following command:

This file wrapper was thoroughly reviewed by our technical staff. Pages 110 is missing in the file.

This has been brought to your attention so that you will know it has not been overlooked.

programs load themselves into memory, remain resident in memory, and perform some task in the background. Programs in this category include: disk caches, print spoolers and network software. These programs are often referred to as TSR (Terminate and Stay Resident) programs.

Like a TSR program, most viruses also remain resident in memory, and it is for this reason that TbMem should be usedto control the process of becoming resident in memory.

If a program attempts to become resident, TbMem offers you the option to abort the attempt. It does this by guarding the DOS TSR function calls while also monitoring important interrupts and memory structures. TbMem uses the ANTI-VIR.DAT records to determine whether it will allow a specific program to remain resident in memory.

TbSetup recognizes many common TSRs. If it doesn't recognize a TSR, however, TbMem asks your permission for the TSR to load. It then maintains permission information in the ANTI-VIR.DAT files to prevent TbMem from bothering you when an approved TSR is loading.

TbMem also checks the contents of the CMOS configuration memory after each program termination to ensure that programs have not changed. TbMem offers you the option of restoring the CMOS configuration when it changes. Once you teach TbMem which programs are TSRs and which are not on a PC, you can use TbSetup to set the permission flag of these files on other machines.

TbMem also installs a hot key that you can use to escape from nearly all programs.

TbMem is fully network compatible. It does not require you to reload the checker after logging onto a network.

3.7.5 Working with TbMem

Since TbMem is a memory resident program, you can execute and configure it from the command line or from within a batch file. It is more efficient, however, to load TbMem at boot up from either CONFIG.SYS or AUTOEXEC.BAT. See the "Introducing the TbMem, TbFile and TbDisk Utilities" section earlier in this chapter for details.

CAUTION:

You must load TbDriver before you can load TbMem. TbMem will refuse to load without it.

3.7.6 Maximizing TbMem

You can maximize the performance of TbMem by using its command line options. The first four options in the table below are always available. The other options are available only if TbMem is not yet memory resident.

option	parameter	short explanation
help	?	display on-line help
remove	r	remove TbMem from memory
on	e	enable checking
off	d	disable checking
secure	s	do not execute unauthorized TSRs
hotkey<=keycode>	k	specify keyboard scancode for the program cancel hotkey
nocancel	n	do not install the cancel hotkey
nocmos	m	do not protect CMOS memory

The explanations in the above table serve as a quick reference, but the follow descriptions provide more information about each option.

TIP:
Remember that you can display these options from the command line by entering TBMEM ?.

help (?).

Specifying this option displays the brief help as shown above.

remove (r).

This option disables TbMem and attempts to remove the resident part of its code from memory and return this memory space to the system. Unfortunately, this works only if you loaded TbMem last. An attempt to remove a TSR after you load another TSR leaves a useless gap in memory and could disrupt the interrupt chain. TbMem checks whether it is safe to remove its resident code; if not, it simply disables itself.

on (e).

This option reactivates TbMem after you disable it using the OFF option.

off (d).

Specifying this option disables TbMem but leaves it in memory.

secure (s).

TbMem normally asks the user to continue or to cancel when a program tries to remain resident in memory. In some business environments, however, employees should not make this choice. If you use this option, it is no longer possible to execute new or unknown resident software. It is also no longer possible to use the REMOVE or OFF options.

hotkey (k).

TbMem offers you a reliable way to escape from any program by pressing a special key combination. You can not only use this feature to escape from programs that "hang," but also from software that seems to be malicious (although we recommend powering down and rebooting from a write-protected system disk). Instead of the default combination (Ctrl+Alt+Insert), you can specify another keyboard combination using the HOTKEY=<KEYCODE> option. You must specify the scancode using a 4-digit hexadecimal number; the first two digits specify the shift-key mask, and the last two digits specify the keyboard scancode. Consult your PC manual for a list of "scan codes." For example, the default scan code is 0C52, but you can change this to another code, such as 0C01, the code for Ctrl+Alt+Esc.

nocancel (n).

TbMem normally installs the program cancel hot key (Ctrl+Alt+Insert). If you do not want to use the program cancel hot key, specify this option, since this saves a few bytes of memory.

nocmos (m).

TbMem normally protects the CMOS memory if available. If you do not want TbMem to do this, you can specify this option.

The following command loads TbMem as a device driver in the CONFIG.SYS, configures the "program cancel hot key" as Ctrl+Alt+Esc, and cancels protection of CMOS memory:

```
DEVICE=C:\TBAV\TBMEM.EXE HOTKEY=0C01 NOCMOS
```

To achieve the same functionality, you could execute TbMem from the DOS command line rather then specifying the TbMem command line in the CONFIG.SYS by entering the following command at the DOS command line:

```
C:\TBAV\TBMEM.EXE HOTKEY=0C01 NOCMOS
```

3.7.7 Understanding TbMem's Operation

If TbMem detects that a program tries to remain resident in memory, it displays a pop-up window displaying a message to that effect. You can either choose to continue, or to abort the program's loading. If you answer "NO" to the question "Remove program from memory?" the program continues undisturbed, and TbMem places a mark in the ANTI-VIR.DAT file about this program. Next time you invoke the same resident program, TbMem will not disturb you again.

There are many programs that normally remain resident in memory, such as: disk caches, print spoolers, and others. How, then, does TbMem distinguish between these programs and viruses?

TbMem uses the ANTI-VIR.DAT records generated by TbSetup to keep track of which files are normal TSRs and which are not. It marks most common resident software as being common so you don't have to worry about these files.

If TbMem pops up with the message that a program tries to remain resident in memory, you have to consider the purpose of the program mentioned. For example, is the program supposed to continue to operate in the background? The answer is obviously yes if the program is a disk cache, print spooler, pop-up utility or system extension software.

If, on the other hand, the message appears after you have exited your word processor, database, spreadsheet application, something is definitely wrong! You ought to terminate the program immediately and use a virus scanner to check the system. The same applies when software that

operates normally without staying resident in memory suddenly changes its behavior and tries to remain resident in memory.

3.8 Using TbFile

This section concerns another resident TBAV utility, TbFile, which checks programs for virus infections as they begin to load.

3.8.1 Understanding TbFile

The two most dangerous virus categories are the boot sector and the file variants. File viruses all have a common purpose, namely, to infect programs. Infecting a program involves very unusual file manipulations that are quite dissimilar to normal file handling procedures, so in order to detect viral activity it is essential to keep an eye out for program file changes involving peculiar actions.

TbFile monitors the system and detects attempts by programs to infect other programs. Unlike other file guards, TbFile monitors the system only for virus specific file modifications. TbFile doesn't generate an alarm when a program modifies itself for configuration purposes, nor does it bother you when you update a program or create one yourself. On an average system, configurations should never cause a false alarm. TbFile has a very sophisticated infection detector and will not give a false alarm when you perform standard file operations. In normal configurations you will never get a false alarm!

TbFile not only detects attempts to infect programs, it also offers you the option of aborting the infection process and continuing a program's execution.

TbFile also detects other suspicious activities, including setting the seconds value of time stamps to an illegal value.

TIP:

> As many users know, you can protect files against unwanted modifications by means of the read-only attribute. Without TbFile, however, someone can easily circumvent this standard DOS protection. TbFile detects any attempts to sabotage the read-only attribute. This gives you added security by enabling you to use this uncomplicated method to fully protect your files against destruction and infection.

TbFile is fully network compatible. It does not require you to reload the checker after logging onto a network. In contrast, other resident anti-virus utilities force you to choose between protection BEFORE you start the network, or protection AFTER you start network, but not both.

3.8.2 Working with TbFile

Since TbFile is a memory resident program, you can execute and configure it from the command line or from within a batch file. It is more efficient, however, to load TbFile at boot up from either CONFIG.SYS or AUTOEXEC.BAT. See the "Introducing the TbMem, TbFile and TbDisk Utilities" section earlier in this chapter for details.

CAUTION:
You must load TbDriver before you can load TbFile. TbFile will refuse to load without it.

3.8.3 Maximizing TbFile

You can maximize the performance of TbFile by using its command line options. The first four options in the table below are always available. The other options are available only if TbFile is not yet memory resident.

option	parameter	short explanation
help	?	display on-line help
remove	r	remove TbFile from memory
on	e	enable checking
off	d	disable checking
secure	s	all permissions denied
allattrib	a	readonly check on all files
compat	c	allow CPM-style file I/O calls

The explanations in the above table serve as a quick reference, but the following descriptions provide more information about each option.

TIP:
Remember that you can display these options from the command line by entering TBFILE ?.

help (?).

Specifying this option displays the brief help shown above.

remove (r).

This option disables TbFile and attempts to remove the resident part of its code from memory and return this memory space to the system. Unfortunately, this works only if you loaded TbFile last. An attempt to remove a TSR after you load another TSR leaves a useless gap in memory and could disrupt the interrupt chain. TbFile checks whether it is safe to remove its resident code; if not, it simply disables itself.

on (e).

This option reactivates TbFile after you disabled it using the OFF option.

off (d).

Specifying this options disable TbFile, but leaves it in memory.

secure (s).

TbFile normally asks you to continue or to cancel when a program tries to perform a suspicious operation. In some business environments, however, employees should not make this decision. If you use the SECURE option, it is no longer possible to allow suspicious operations. It is also no longer possible to use the OFF and REMOVE options.

allattrib (a).

TbFile normally protects only the read-only attribute of executable files (program files with the extension COM and EXE). If you want to have the read-only check on all files, add this option. In this case you always get an alarm when something attempts to remove the read-only attribute of any file.

compat (c).

DOS still contains some CPM (an earlier operating system) internal functions, even though DOS programs no longer use these functions. Some viruses, however, use these functions to bypass anti-virus software. TbFile closes these backdoors by default, but you can prevent this by specifying this option.

The following command loads as a device driver in CONFIG.SYS and it guards the read-only attribute of all files:

```
DEVICE=C:\TBAV\TBFILE.EXE ALLATTRIB
```

To achieve the same functionality, you could execute TbFile from the DOS command line rather then specifying the TbFile command line in the CONFIG.SYS by entering the following command at the command line:

```
C:\TBAV\TBFILE.EXE ALLDRIVES
```

3.9 Using TbDisk

This section deals with TbDisk, which prevents viruses from damaging data on your hard disk.

3.9.1 Understanding TbDisk

Many viruses try to damage the data on disk. They accomplish this by various actions, such as, formatting the disk, overwriting the FAT, and swapping disk sectors, among others. Almost anything is possible!

Another category of malicious software, known as boot sector virus droppers, install a boot sector virus on the disk. The program itself is not a virus, so detection with virus scanners and other anti-viral software is very difficult. The only way to detect such a program is by monitoring its behavior.

The main problem in all this lies in the way these programs manage to avoid the usual DOS procedures: they go directly to the BIOS (Basic Input/Output System). This is the reason you need TbDisk, to monitor the system and to ensure that no program can write directly to disk without permission. TbDisk draws attention to any software that attempts to write directly to disk, thereby reducing the likelihood of a virus remaining unnoticed. TbDisk prevents viruses from damaging data on your disk and stops boot sector virus droppers in their tracks.

TbDisk not only informs you when a program tries to write directly to the disk, it also offers you the option to abort the program before it can cause any damage.

TbDisk is able to detect stealth techniques, that is, attempts to single step through the BIOS software, and even monitors the use of undocumented calls that could cause disk damage. For example, TbDisk is able to distinguish whether DOS or an application makes direct write attempts via Int 13h (a system call implemented in the BIOS of your computer). Direct writes are perfectly legal for DOS, but unusual for application software.

TbDisk does require a little maintenance. TbDisk uses the ANTI-VIR.DAT records to determine if it should allow a program (including popular disk utilities, which TbSetup recognizes) to write directly to the disk. In the absence of an ANTI-VIR.DAT record, TbDisk asks your permission first and, if granted it, updates the record accordingly to avoid repeated warnings about the same program.

TbDisk is fully network compatible. It does not require you to reload the program after logging onto a network. Other resident anti-virus utilities force you to choose between either protection BEFORE the network is started, or protection AFTER it starts, but not both..

TIP:
TbDisk also comes in handy if you ever need to write protect a hard disk. This bonus feature often helps when testing new software.

3.9.2 Working with TbDisk

Since TbDisk is a memory resident program, you can execute and configure it from the command line or from within a batch file. It is more efficient, however, to load TbFile at boot up from either CONFIG.SYS or AUTOEXEC.BAT. See the "Introducing the TbMem, TbFile and TbDisk Utilities" section earlier in this chapter for details.

CAUTION:
You must load TbDriver before you can load TbDisk. TbDisk will refuse to load without it.

In addition to all this, there are several special considerations in using TbDisk.

Loading TbDisk

Improper installation of TbDisk can cause excessive false alarms! If you want to install TbDisk in your CONFIG.SYS or AUTOEXEC.BAT file, we recommend that you use the INSTALL option of TbDisk first. If the system continues to behave normally and TbDisk does not give false alarms when you copy files on your hard disk, TbDisk is installed correctly and you can remove the INSTALL option from the command.

WARNING:
Failure to use the Install option when you install TbDisk in CONFIG.SYS or AUTOEXEC.BAT file might cause loss of data! Please read on.

While the INSTALL option instructs TbDisk to allow all disk accesses, it also displays a message as it would do in normal mode. If no false alarms occur when you copy files on your hard disk, TbDisk is installed correctly and you can remove the INSTALL option.

If TbDisk causes false alarms, load TbDisk further ahead in your CONFIG.SYS or AUTOEXEC.BAT file until it works as it should.

CAUTION:
Unlike the other TBAV utilities, we recommend that you load TbDisk after other resident software! Failure to do so can cause false alarms!

TbDisk detects if Windows is running and automatically switches into multitasking mode if necessary. You can even disable TbDisk in one window without affecting the functionality in another. If you are using Windows fast 32-bit disk access, you might need to use TbDisk's WIN32 option if Windows displays an error-message.

3.9.3 Maximizing TbDisk

You can maximize TbDisk's performance by using its command line options. The first four options are always available. The other options are available only if TbDisk is not yet memory resident.

option	parameter	short explanation
help	?	display on-line help
remove	r	remove TbDisk from memory
on	e	enable checking
off	d	disable checking
wrprot	p	makes hard disk write protected
nowrprot	n	allow writes to hard disk
win32	w	allow Windows 32-bit disk access
secure	s	deny access without asking first
notunnel	t	do not detect tunneling
nostealth	a	do not detect stealth disk access
install	i	installation test mode

The explanations in the above table serve as a quick reference, but the following descriptions provide more information about each option.

TIP:
Remember that you can display these options from the command line by entering TBDISK ?.

help (?).

Specifying this option displays the brief help as shown above. After loading TbDisk into memory, not all options appear.

remove (r).

This option disables TbDisk and attempts to remove the resident part of its code from memory and return this memory space to the system. Unfortunately, this works only if you loaded TbDisk last. An attempt to remove a TSR after you load another TSR leaves a useless gap in memory and could disrupt the interrupt chain. TbDisk checks whether it is safe to remove its resident code; if not, it simply disables itself.

on (e).

This option activates TbDisk after you disabled it using the OFF option.

off (d).

Specifying this option disables TbDisk but leaves it in memory.

wrprot (p).

Hard disks are more difficult to protect against writing than floppies, which adds considerable risk when doing such things as testing new software. Sometimes you might want to find out what this software does to your hard disk and how this could possibly affect your valuable data. Using the "WRPROT" option makes this safer to do. Whenever a program wishes to write to a protected disk, you will see a message such as:

Write protect error writing drive C: A)bort, R)etry, I)gnore?

You can then take the appropriate action.

CAUTION:

Software write protection is not absolutely reliable. Some viruses can bypass this protection, but fortunately they are few and far between. Despite its shortcomings, this option can be a valuable shield against most malicious software.

nowrprot (n).

Use this option to undo the WRPROT option.

win32 (w).

Windows 386 Enhanced Mode uses some undocumented DOS calls to retrieve the original BIOS disk handler when you enable 32-bit disk access. Since TbDisk guards these calls, 32-bit disk access will no longer be possible, unless you specify the WIN32 option when you initialize TbDisk.

CAUTION:
Use this option only in Windows 386 Enhanced Mode with fast 32-bit disk access enabled as it reduces anti-viral security to some extent.

secure (s).

TbDisk normally asks whether the user wants to continue or cancel when a program tries to perform direct disk access. In some business environments, however, employees should not make this decision. This option disables direct disk access permission to new or unknown software. It also disables the OFF and REMOVE options.

notunnel (t).

"Tunneling" is a technique viruses apply to determine the location of the DOS system code in memory, and to use that address to communicate with DOS directly. This inactivates all TSR programs, including resident anti-virus software. TbDisk is able to detect these "tunneling" attempts, and informs you about it. Some other anti-virus products also rely on tunneling techniques to bypass resident viruses, thereby causing false alarms. If you are currently executing other anti-viral products, the NOTUNNEL option disables TbDisk's tunneling detection.

nostealth (a).

TbDisk tries to detect direct calls into the BIOS. If such an attempt occurs, TbDisk pops up with a message that something is

accessing the disk in an unusual way. If this feature causes false alarms, you can use this option to turn it off.

install (i).

Incorrect installation can result in a large number of false alarms. You should use this option when installing TbDisk because it reduces the risk of canceling a valid disk write operation as a result of false alarms.

3.9.4 Understanding TbDisk's Operation

What is Direct Disk Access? Programs usually access files through the operating system (DOS). Whenever a program wants to update a file, for example, it asks DOS to write the data to disk. It is also possible, however, to write to a disk without using DOS. This is called direct disk access.

While normal programs do not write to the disk directly, there are some programs that need to do so, including:

Format utilities. Direct disk access is the only way to format a disk.

Disk diagnosis utilities (such as the Norton Disk Doctor, and DOS's CHKDSK command and ScanDisk utility).

Disk optimizers and defragmenters (such as Norton SpeedDisk and DOS's Defrag utility).

Since many viruses can perform direct disk access, it is essential to control this. TbDisk can distinguish between legitimate programs and a virus with the help of the ANTI-VIR.DAT records, which you can generate using TbSetup.

Whenever TbDisk pops up a message that says a program accesses to the disk directly, consider its purpose carefully. While it is perfectly acceptable for a format utility or a disk optimizer to format or edit disk sectors, this is not acceptable for a word processor or database. When TbDisk warns you that a spreadsheet or some other normal program is about to format a sector, you can be sure that something is wrong. Terminate the program pronto! Then check things out with a virus scanner before the worst happens.

3.10 Using TbUtil

This section describes TbUtil, which is designed primarily to make a precautionary backup of clean partition tables and boot sectors.

3.10.1 Understanding and using TbUtil

TbUtil provides a defense against partition table and boot sector viruses. TbUtil can be used to:

Copy the partition table, boot sector and CMOS data area into a file. You can use TbUtil on a regular basis to compare both the current and the original versions of the partition table, boot sector and CMOS data area. After an accident virus, (virus or otherwise), you can restore the copy using the TbUtil program.

Remove a partition table virus without having to low-level format the hard disk, even if there is no backup of the partition table.

Remove boot sector viruses and creates a partition table that has some first-line virus defenses built-in.

Replace the infected or clean boot sector with a safe TBAV boot sector.

NOTE:

What is a partition table? A physical hard disk might consist of more than one "partition" (or division). Each partition is a logical disk drive and has it own ID, such as C:, D:, and E:. The partition table, then, contains the disk lay-out and the starting and ending cylinder of every partition. The partition table also contains information about the operating system of a partition and which partition should be used to boot. The partition table (also called the Master Boot Record, or MBR) always resides at the very first sector of the hard disk.

Unlike most file viruses, partition table viruses are hard to remove. The only solution is to low-level format the hard disk and to make a new partition table, or to make use of scantily documented DOS commands.

TbUtil, however, makes a backup of the partition table and boot sector and uses this backup to compare and restore both the original partition table and boot sector once they become infected. You no longer have to

format your disk to get rid of a partition table or boot sector virus. The program can also restore the CMOS configuration.

Optionally, TbUtil replaces the partition table code with an immunized partition table containing facilities against viruses. The TbUtil partition code executes before the boot sector gains control, so it is able to check the boot sector in a clean environment. Once the boot sector executes, it is difficult to check it because the virus is already resident in memory and can deceive a protection scheme. Instead of booting from a clean DOS diskette just to inspect the boot sector, the TbUtil partition code performs a CRC calculation on the boot sector just before passing control to it.

If TbUtil detects a change in the boot sector, the TbUtil partition code warns you about it. The TbUtil partition code also checks the RAM layout and informs you when it changes. TbUtil does all of this every time you boot from your hard disk.

TbUtil can replace infected and clean diskette boot sectors with a new and specialized boot sector, which has several advantages over the standard boot sector:

It has boot sector virus detection capabilities.

It performs a sanity check.

It offers you the possibility to redirect the boot process to the hard disk without opening the diskette drive door.

3.10.2 Working with the TbUtil Menu

The TbUtil module contains several programs, which you can execute from either the TbUtil Menu or, in case of an emergency, from a TbUtil recovery diskette using the DOS command line. The menu, however, offers some additional menu options. Selecting the "TbUtil" option from the TBAV Main Menu displays the following menu:

```
+------Main menu------+
| Confi+-----------TbUtil menu-----------+
| TbSet| System maintenance menu      >|
| TbSca| Immunize/clean bootsector A:  |
| TbUti| Immunize/clean bootsector B:  |
| TbCLe| Immunize/clean partition code |
| Virus+-------------------------------+
| TBAV Monitor     >|
| Documentation    >|
| Register TBAV     |
| About             |
| Quit and save     |
| eXit (no save)    |
+-------------------+
```

We'll now explore these menu options.

The "System Maintenance Menu" Option

Selecting the "System maintenance menu" option displays the System Maintenance menu:

```
+------Main menu------+
| Confi+-----------TbUtil menu-----------+
| TbSet| Syste+-------System maintenance-------+
| TbSca| Immun| Execute TbUtil               |
| TbUti| Immun| Describe this machine        |
| TbCLe| Immun| Save system configuration    |
| Virus+-------|v Compare system configuration |
| TBAV Monitor | Restore system configuration |
| Documentation|v process CMOS memory         |
| Register TBAV|v process Partition code      |
| About        |v process Bootsector          |
| Quit and save+-------------------------------+
| eXit (no save)    |
+-------------------+
```

This menu contains the actual TbUtil program. The program takes care of saving, restoring or comparing the system configuration of your PC. It stores the backup system configuration on a diskette in a file with either a default name or a name you can specify yourself.

WARNING:

You can only restore a system configuration data file on the machine that created the data file. Restoring a configuration file from one

The "System Maintenance Menu" contains the following items:

Execute TbUtil.

Before activating this option, you must select one of the optional functions: Save, Compare, or Restore the system configuration. Move to the desired option you want to activate and press ENTER. A check mark indicates that an option is active.

Describe this machine.

Enter a meaningful description of the machine. Enter something like, "486DX4 @ 100MHz, 32Mb, 2 Gb SCSI disk, room 12, Mr. Smith." You do NOT have to remember this description; TbUtil displays it on the screen when comparing or restoring, which helps you to verify that the data file belongs to the machine.

Save system configuration.

This option stores the partition table, boot sector and CMOS data area into the TbUtil data file.

WARNING:
Since the PC is completely inaccessible to DOS if the partition table becomes damaged, we RECOMMEND that you store both the TbUtil data file AND the program TBUTIL.EXE itself on a "rescue" diskette! If the partition table is damaged or destroyed, then the only solution to the problem may reside on the "rescue" diskette, since your hard drive may be inaccessible!

When loading TbUtil from the command line you must specify a filename after the STORE option. In contrast, using the TBAV menu, you can use the default filename TBUTIL.DAT. If you own more than one PC, we recommend that you create one TbUtil diskette with all TbUtil data files of all your PC's on it. Use the extension of the file for PC identification, as in the following:

A:TBUTIL.<NUMBER>

Compare system configuration.
This option enables you to check on a regular basis that everything is still okay. If you specify this option, TbUtil compares the

information in the TbUtil data file against the partition table, boot sector, and CMOS data areas. It also displays the comment stored in the data file. Using this option also guarantees that the TbUtil data file is still readable.

Restore system configuration.

This option enables you to restore the partition table, boot sector, and CMOS data area. It asks you to confirm that the data file belongs to the current machine. Finally, it can restore the partition table, boot sector of the partition to be used to boot, and the CMOS data area.

Process CMOS memory,
Process Partition code, and
Process Boot sector.

By default, TbUtil restores the partition code, boot sector, and CMOS if you specify the "Restore system configuration" option. If you use one of the above options in combination with the "Restore option," TbUtil restores only the items you specify.

The "Immunize/Clean Boot sector A: [or] B:" Options

You can use these options to clean diskettes infected by a boot sector virus or to replace the standard boot sector with a boot sector that has advantages over the original one:

The TBAV boot sector has virus detection capabilities. The TBAV boot sector checks that it resides on the correct place on the diskette, and that Int 13h and/or Int 40h still exist in system ROM. This makes it possible to detect even stealth and boot sector viruses.

The TBAV boot sector can load the system files if they are available on the disk, but if the DOS system files are not on the disk, the TBAV boot sector displays a small menu offering you two possibilities: retry the boot operation with another diskette, or boot from the hard disk. If you select the latter, you don't have to open the diskette drive door.

The "Immunize/Clean Partition Code" Option

This is an extremely powerful option, which you can use to clean an infected partition table if there is no TbUtil data file. It saves the original partition code in a file and replaces the existing partition table code with a new partition routine that contains some virus detection capabilities. You must execute TbUtil from a floppy drive or you have to specify the name of the file (the specified drive should be a diskette drive) to store the original partition code.

If the original partition table becomes irreparably damaged and can't be used to build a new one, TbUtil scans the entire disk for information about the original disk layout. TbUtil also searches for TbUtil data files on the hard disk.

CAUTION:
While it is a good idea to keep a copy of the data file on the hard disk, we recommend that you store the data file on a diskette. Just in case!

If your system configuration changes, that is, you update your DOS version or change the amount of memory, you need to update the information stored in the immune partition as well. You can do this by using this option.

In the unlikely event that the system does not boot properly, you can restore the original partition table using the TbUtil RESTORE option (refer to The "System Maintenance Menu Option" section above) or by using the DOS version 5 or above FDISK /MBR command (which creates a new partition table).

TIP:
If you have installed two hard drives in your computer, you can immunize the partition code of the second hard drive by specifying the physical drive number rather than the drive ID (i.e., execute the command TbUtil 2:)

If the new partition code works properly, you should make a backup copy of it on a diskette using the TbUtil STORE option (refer to The "System Maintenance Menu Option" section above).

3.10.3 Maximizing TbUtil

This section describes how to fully maximize TbUtil in three ways: use command line option, use the anti-virus partition, use the TbUtil diskette.

Now that you know how to use TbUtil's menus, you can more easily understand how to maximize its performance by using its command line options.

option parameter	short	explanation
immunize <drive>	im	Immunize/Clean boot sector or MBR of <drive>
getboot <drive>	gb	Save boot sector/MBR into file
store [<filename>]	st	Store system information
restore [<filename>]	re	Restore system information
compare [<filename>]	co	Compare system information

Sub-options of immunize option:

norepeat	nr	Do not ask for next diskette
nomem	nm	Do not check for amount of RAM
batch	ba	Do not prompt to insert a disk

Sub-options of store option:

description=<descr.>	de	Add description to data file

Sub-options of restore option:

part	pt	Restore partition table
boot	bo	Restore boot sector of hard disk
cmos	cm	Restore CMOS data memory

The explanations in the above table serve as a quick reference, but the following descriptions provide more information about each option.

Immunize <floppy drive> (im).

You can use this option to clean diskettes infected by a boot sector virus or to replace the standard boot sector by a boot sector that has advantages over the original one:

The TBAV boot sector has virus detection capabilities. The boot sector checks that it still resides on the correct place on the diskette, and that Int 13h and/or Int 40h still exist in system ROM. This makes it possible to detect even stealth and boot sector viruses.

The TBAV boot sector is able to load the system files if they are available on the disk, but if the DOS system files are not on the disk, the TBAV boot sector displays a small menu offering you two possibilities: retry the boot operation with another diskette, or boot from the hard disk. If you select the latter, you don't have to open the diskette drive door.

Immunize c: (im c:).

This is an extremely powefful option, which you can use to clean an infected partition table if there is no TbUtil data file. It saves the original partition code in a file and replaces the existing partition table code with a new partition routine that contains some virus detection capabilities. You have to execute TbUtil from a floppy drive or you have to specify the name of the file (the specified drive should be a diskette drive) to store the original partition code.

TIP:

If you have installed two hard drives in your computer, you can immunize the partition code of the second hard drive by specifying the physical drive number rather than the drive ID (i.e., execute the command TbUtil 2:)

If the original partition table becomes irreparably damaged and consequently can't be used to build a new one, TbUtil scans the entire disk for information about the original disk layout. TbUtil also searches for TbUtil data files on the hard disk.

CAUTION:

While it is a good idea to keep a copy of the data file on the hard disk, we recommend that you store the data file on a diskette. Just in case!

If your system configuration changes, that is, you update your DOS version or change the amount of memory, you need to update the information stored in the immune partition as well. You can do this by using this option.

In the unlikely event that the system does not boot properly, you can restore the original partition table using the TbUtil RESTORE option (refer to The "System Maintenance Menu Option" section above) or by using the DOS version 5 or above FDISK /MBR command (which creates a new partition table).

getboot <drive> (gb).

With this option you can copy the boot sector of the specified drive into a file.

store [<filename>] (st).

This option stores the partition table, boot sector and CMOS data area into the TbUtil data file.

WARNING:
Since the PC is completely inaccessible to DOS if the partition table becomes damaged, we RECOMMEND that you store both the TbUtil data file AND the program TBUTIL.EXE itself on a rescue diskette! If the partition table is damaged or destroyed, then the only solution to the problem may reside on the "rescue" diskette, since your hard drive may be inaccessible!

When loading TbUtil from the command line you must specify a filename after the STORE option. In contrast, using the TBAV menu, you can use the default filename TBUTIL.DAT. If you own more than one PC, we recommend that you create one TbUtil diskette with all TbUtil data files of all your PC's on it. Use the extension of the file for PC identification, as in the following:

A:TBUTIL.<NUMBER>

restore [<filename>] (re).

This option enables you to restore the partition table, boot sector, and CMOS data area. It asks you to confirm that the data file belongs to the current machine. Finally, it restores the partition table, boot sector of the partition to be used to boot, and the CMOS data area.

compare [<filename>] (co).

This option enables you to check on a regular basis that everything is still okay. If you specify this option, TbUtil compares the information in the TbUtil data file against the partition table, boot sector, and CMOS data area. It also displays the comments stored in the data file. Using this option guarantees that the TbUtil data file is still readable.

norepeat (nr).

By default, TbUtil prompts you for the next diskette after you have immunized a diskette. This option disables this function.

nomem (nm).

If you specify this option when you are immunizing your partition code, the partition code skips the RAM check while booting. This is necessary for some systems that change the memory setup during the boot process.

batch (ba).

If you specify this option, TbUtil will assume a disk has already been inserted in your disk drive. This option is particularly useful with batch files.

description =<descr.> (de).

For <desc.> enter a meaningful description of the machine. Enter something like, "486DX4 @ 100MHz, 32 Mb, 2 Gb SCSI disk, room 12, Mr. Smith." You do NOT have to remember this description; TbUtil displays it on the screen when comparing or restoring, which helps you to verify that the data file belongs to the machine.

part (pt),
boot (bo), and
cmos (cm).

By default, TbUtil restores the partition code, boot sector, and CMOS if you specify the RESTORE option. If you use one of these options in combination with the RESTORE option, however, TbUtil restores only the items you specify.

In the following two examples TbUtil simply store system information gathered from the partition table and boot sectors of your fixed disk(s) and the CMOS data area into a file in the current directory called TBUTIL.DAT.

```
TBUTIL STORE
TBUTIL ST
```

The following example does the same as the previous, except that TbUtil stores the information on a diskette instead of in the current directory.

```
TBUTIL STORE A:TBUTIL.DAT
```

It's a good idea to describe the machine from which you are saving information about the partition table, boot sectors and CMOS data. You can use the DESCRIPTION option to add a small, single-line description of the machine:

```
TBUTIL STORE A:TBUTIL.DAT DESCRIPTION = "TEST MACHINE"
```

You can always fall back on the information TbUtil stores if you suspect an infection by a boot sector virus. Suppose the information gathered earlier by TbUtil is stored in the file A:\TBUTIL.DAT. To compare the current system information with the information stored in the TbUtil data file, you could use this command:

```
TBUTIL COMPARE A:TBUTIL.DAT
```

Now suppose that TbUtil informs you that the current system information (that is, the partition table and the CMOS data area) does not match the information stored earlier. If you did not change the configuration of your computer, it is most likely that a virus is guilty of the change. You could restore the old system information using this command:

```
TBUTIL RESTORE A:TBUTIL.DAT PART CMOS
```

In case of a boot sector virus infection, we recommend that you disinfect (clean) all diskettes. Using the following command, TbUtil cleans and immunizes the boot sector of the diskette in drive A: and then repeats the action after asking you to insert other (possibly) infected diskettes into the disk drive:

```
TBUTIL IMMUNIZE A:
```

In case of a virus infection you should always make certain that the Master Boot Record of your fixed disk is not infected. The following command specifies an extra option, which you must use in case your computer changes its memory setup during the boot process:

```
TBUTIL IMMUNIZE C: NOMEM
```

You can easily view the contents of a TBUTIL.DAT by using the DOS TYPE command:

```
TYPE A:TBUTIL.DAT
```

3.10.4 Using the Anti-Virus Partition

If you install the ThunderBYTE partition code (by using TbUtil's IMMUNIZE option), you will see the following when booting a clean system:

```
Thunderbyte anti-virus partition (C)1993-95 Thunderbyte BV.

Checking boot sector CRC -> OK!
Checking available RAM -> OK!
Checking INT 13h -> OK!
```

In contrast, if there is a virus in the boot sector or partition table, you will see this message:

```
Thunderbyte anti-virus partition (C)1993-95 Thunderbyte BV.

Checking boot sector CRC -> OK!
Checking available RAM -> Failed!

System might be infected. Continue? (N/Y)
```

Other messages that might appear are:

"No system." This message means that there is no active partition on the disk.

"Disk error." The meaning of this message is obvious.

3.10.5 Using the TbUtil diskette

To use the TbUtil diskette, follow these steps:

1. Take a new diskette and format it as a bootable diskette (by using the DOS FORMAT /S command).

2. Copy the TbUtil files onto the diskette using this command:

COPY TBUTIL.* A:

The TbUtil files you need are TBUTIL.EXE and TBUTIL.LNG.

3. In case of an emergency (such as a damaged or infected partition table, for example), boot from the TbUtil diskette.

4. Run the TbUtil program, using the IMMUNIZE option:

A:\TBUTIL IMMUNIZE C:

This cleans the partition table.

5. You should now be able to boot from your hard disk normally.

3.11 Using TbLog

This section describes TbLog, which is designed primarily to create log files in response to various TBAV alert messages.

3.11.1 Understanding and using TbLog

TbLog is a memory resident TBAV utility that writes a record into a log file whenever one of the resident TBAV utilities pops up with an alert message. It also records when a virus is detected.

This utility is primarily for network users. If all workstations have TbLog installed and configured to maintain the same log file, the supervisor can easily keep track of what's going on. When a virus enters the network he is able to determine which machine introduced the virus, and he can take action in time.

A TbLog record provides three pieces of information:

The time stamp of when the event took place.

The name of the machine on which the event occurred.

An informative message about what happened and which files were involved.

This information is very comprehensive and takes only one line.

3.11.2 Working with TbLog

Since TbLog is a memory resident program, you can execute and configure it from the DOS command line or from within a batch file. You should, however, load TbLog automatically and when the computer boots, preferably during the execution of AUTOEXEC.BAT, or better yet, CONFIG.SYS.

You should install TbLog on every workstation. If you want to use all workstations to maintain the same log file, we recommend that you load TbLog after starting the network.

By default, TbLog maintains a log file with the name TBLOG.LOG in the TBAV directory. If you want to use another filename or another disk and/or directory, you can specify a filename (and path) on the TbLog

command line. In a network environment, we recommend that you put the log file on a server disk.

CAUTION:
Be sure to load TbDriver before trying to load TbLog. TbLog will refuse to load without it.

There are three possible ways to load TbLog:

1.From the DOS prompt or within the AUTOEXEC.BAT file:

<PATH>TBLOG

2.From CONFIG.SYS as a TSR (DOS 4 or above):

INSTALL=<PATH>TBLOG.EXE

The INSTALL= CONFIG.SYS command is NOT available in DOS 3.xx.

3.From CONFIG.SYS as a device driver:

DEVICE=<PATH>TBLOG.EXE

NOTE:
Executing TbLog as a device driver does not work in all OEM versions of DOS. If you encounter problems, use the INSTALL= command or make sure to load TbLog from the AUTOEXEC.BAT. Also, unlike other anti-virus products, you can load the ThunderBYTE Anti-Virus utilities before starting a network without losing the protection after the network is started.

In addition to the three loading possibilities, if you are using DOS version 5 or above, you can load TbLog into an available UMB (upper memory block) from AUTOEXEC.BAT using this command:

LOADHIGH <PATH>TBLOG

You can also load TbLog into high memory from within the CONFIG.SYS using this command:

DEVICEHIGH=<PATH>TBLOG.EXE

If you are using Microsoft Windows, you should load TbLog BEFORE starting Windows. When you do this, there is only one copy of TbLog in memory regardless of how many DOS windows you might open. Every DOS window (that

is, every virtual machine) has a fully functional copy of TbLog running in it.

TbLog automatically detects if Windows is running, and switches itself into multi-tasking mode if necessary. You can even disable TbLog in one window without affecting its functionality in another window.

3.11.3 Maximizing TbLog

You can maximize TbLog's performance by using its command line options. The first five options in the following table are always available. The other options are available only if TbLog is not yet memory resident.

option parameter	short	explanation
help	?	Display some on-line help
remove	r	Remove TbLog from memory
on	e	Enable TbLog
off	d	Disable TbLog
test	t	Log test message
machine=<descr.>	m	Description/name of your machine
secure	s	Do not allow removal of TbLog

The explanations in the above table serve as a quick reference, but the following descriptions provide more information about each option.

help (?).

Specifying this option displays the brief help as shown above.

remove (r).

This option disables TbLog and attempts to remove the resident part of its code from memory and return this memory space back to the system. Unfortunately, this works only if you loaded TbLog last. An attempt to remove a TSR after you load another TSR leaves a useless gap in memory and could disrupt the interrupt chain. TbLog checks whether it is safe to remove its resident code; if not, it simply disables itself.

on (e).

This option reactivates TbLog after you disabled it using the OFF option.

off (d).

Specifying this option disables TbLog but leaves it in memory.

test (t).

Use this option to record a test message. If you use this option at the initial loading of TbLog, it records the time and machine name into the log file. If you use this option after the initial loading, it simply places a test message into the log file.

machine (m).

Using this option, you can specify the name of the machine on which TbLog is running. This machine name appears in the log file. By default, TbLog uses the network machine name on NetBios compatible machines. On other networks, such as Novell, you must enter the network name on the TbLog command line.

secure (s).

If you specify this option, it is not possible to use the OFF and REMOVE options.

The following command loads TbLog, disables, the OFF and REMOVE options, specifies that the logfile reside in directory F:\SECURITY, and identifies the machine as DESK3:

```
C:\TBAV\TBLOG F:\SECURITY\TBLOG.LOG SECURE MACHINE=DESK3
```

The following CONFIG.SYS command loads TbLog, creates the logfile in directory X:\LOGS, and specifies that the first line of the log file contain a date/time stamp and the name of the computer:

```
DEVICE=C:\TBAV\TBLOG X:\LOGS\TBLOG.LOG MACHINE=JOHN TEST
```

3.12 Using TbNet

TBAV for DOS can cooperate with TBAV for Networks, another ThunderBYTE product, via the program called TbNet. If you do not want to use the combination of TBAV for DOS and TBAV for Networks, you can skip this section.

NOTE:
For more information about TBAV for Networks, please refer its documentation. If you did not purchase TBAV for Networks yet, your local dealer can inform you about this product.

3.12.1 Understanding TbNet

TbNet is a memory resident TBAV utility that implements the communication between TBAV for DOS and TBAV for Networks. TBAV for Networks has several options for controlling remote workstations. For Windows workstations, TBAV for Windows contains all logic needed to implement the communication between the workstation and TBAV for Networks. For DOS workstations you need TbNet for this communication.

3.12.2 Working with TbNet

Since TbNet is a memory resident program, you can execute and configure it from the DOS command line or from within a batch file. You should, however, load TbNet automatically when the computer boots, preferably during the execution of AUTOEXEC.BAT, or better yet, CONFIG.SYS.

You should install TbNet on every workstation.

CAUTION:
Since TbNet uses a public network directory for its communication with TBAV for Networks, you must load TbNet after starting the network.

There are three possible ways to load TbNet:

1. From the DOS prompt or within the AUTOEXEC.BAT file:

   <PATH>TBNET

2. From CONFIG.SYS as a TSR (DOS 4 or above):

INSTALL=<PATH>TBNET.EXE

The INSTALL= CONFIG.SYS command is NOT available in DOS 3.xx.

3. From CONFIG.SYS as a device driver:

DEVICE=<PATH>TBNET.EXE

NOTE:
Executing TbNet as a device driver does not work in all OEM versions of DOS. If it doesn't work, use the INSTALL= command or load TbNet from AUTOEXEC.BAT. TbNet should always work correctly if you load it from AUTOEXEC.BAT. Also, unlike other anti-virus products, you can load the ThunderBYTE Anti-Virus utilities before starting a network without losing the protection after the network is started.

In addition to the three loading possibilities, if you are using DOS version 5 or above, you can load TbNet into an available UMB (upper memory block) from AUTOEXEC.BAT using this command:

LOADHIGH <PATH>TBNET

You can also load TbNet into high memory from within the CONFIG.SYS using this command:

DEVICEHIGH=<PATH>TBNET.EXE

We recommend that you do not use TbNet if you use MS-Windows, but use TBAV for Windows instead. TBAV for Windows has built-in functionality for communication with TBAV for Networks.

If you do want to use TbNet with MS-Windows for some reason, you should load TbNet BEFORE starting Windows. When you do this, there is only one copy of TbNet in memory regardless of how many DOS windows you might open. Every DOS window (that is, every "virtual machine") has a fully functional copy of TbNet running in it.

TbNet automatically detects if Windows is running, and switches itself into multi-tasking mode if necessary. You can even disable TbNet in one window without affecting the functionality in another window.

3.12.3 Maximizing TbNet

You can maximize TbNet's performance by using its command line options. The help and remove options in the following table are always available. The other options are available only if TbNet is not yet memory resident.

option parameter	short	explanation
help	?	Display some on-line help
remove	r	Remove TbNet from memory
netname=<netname>	n	Netname of the workstation
commdir=<path>	c	Communication directory used by workstation
frequency=<seconds>	f	Poll frequency (default is 30 seconds)
buffers=<number>	b	Number of disk buffers (default is 2)

The explanations in the above table serve as a quick reference, but the following descriptions provide more information about each option.

help (?).
Specifying this option displays the brief help as shown above.

remove (r).

This option disables TbNet and attempts to remove the resident part of its code from memory and return this memory space back to the system. Unfortunately, this works only if you loaded TbNet last. An attempt to remove a TSR after you load another TSR leaves a useless gap in memory and could disrupt the interrupt chain. TbNet checks whether it is safe to remove its resident code; if not, it simply disables itself

netname (n).

TBAV for Networks distinguishes workstations by their unique netnames. These netnames are assigned by TBAV for Networks; the agents software running at the workstations (i.e., TbNet or TBAV for Windows) receive this netname upon registering the workstation with TBAV for Networks. You need to specify this netname for correct behavior of TbNet.

commdir (c).

The communication between TBAV for Networks and the agent software running at the workstations (i.e., TbNet or TBAV for Windows) takes

place via a special "communication directory," a directory that is public to all users. You must specify the path of this directory when loading TbNet.

frequency (f).

TbNet checks the communication directory every once in a while, to see if messages originating from TBAV for Networks need to be processed. You can change the default period of 30 seconds by specifying the FREQUENCY option.

buffers (b).

TbNet internally needs some buffers to speed up the communication with TBAV for Networks. The number of these disk buffers used by TbNet can be changed by using the BUFFERS option.

The following command loads TbNet, for workstation 001AE3, making use of the communication directory J:\TBAVNW.NET.

```
C:\TBAV\TBNET NETNAME=001AE3COMMDIR=J:\TBAVNW.NET
```

4 Understanding Advanced User Information

This chapter presents some advanced information on using memory, TbSetup, TbScan, and TbClean. It also introduces you to another TBAV utility, TbGenSig, signature file compiler. While some of this material is simply for a better understanding of the utilities and might not be of interest to you, we recommend that you at least look at the first section on memory considerations.

4.1 Understanding Memory Considerations

This section presents the memory requirements for each of the TBAV utilities and how you can reduce the requirements of each utility.

4.1.1 Understanding Memory Requirements

The following table lists the memory requirements for each of the TBAV utilities:

TBAV Utility	Memory needed to load	Memory consumed after exiting
TbScan *	200 Kb	-
TbScanX **	10 Kb	800 bytes
TbCheck	4 Kb	600 bytes
TbUtil	64 Kb	-
TbClean ***	96 Kb	-
TbMem	4 Kb	600 bytes
TbFile	5 Kb	1 Kb
TbDisk	4 Kb	800 bytes
TbDriver	5 Kb	3 Kb
TbLog	5 Kb	1 Kb

* If you decide to use a log file, TbScan requires an additional 16 kilobytes of memory for the log file buffer. If TbScan uses its own built-in file system, it uses additional memory to keep the FAT in memory. Note that the memory requirements are independent of the number of signatures. The current memory requirements are adequate to manage at least 2500 signatures.

** The amount of memory TbScanX requires depends on the number of signatures. If you enable all features, TbScanX uses 30 kilobytes of memory when scanning for 1400 family signatures. If you enable swapping, TbScanX normally uses only one kilobyte of memory. You can swap to EMS and XMS memory. Naturally you can load the remaining kilobyte of TbScanX into upper memory.

*** In the heuristic cleaning mode TbClean requires much more memory, depending on the size of the infected file. TbClean can also use expanded memory (EMS).

4.1.2 Reducing Memory Requirements

Most PC users try to maintain as much free DOS memory as possible. The memory resident TBAV utilities (TbScanX, TbCheck, TbMem, TbFile, TbDisk, TbLog and TbDriver) use only a small amount of DOS memory. To decrease the memory requirements of these utilities even further, do the following:

Load the programs from within the CONFIG.SYS file. When loaded as a device driver, a TBAV utility has no Program Segment Prefix (PSP, a DOS-internal memory area), which saves 256 bytes for each TBAV utility.

If you load the TBAV utilities from within the AUTOEXEC.BAT file, load them before establishing environment variables. DOS maintains a list of environment variables for every resident program, so keep this list small while installing TSRs. Once you install all TSRs, you can then define all environment variables without affecting the memory requirements of the TSRs.

Make use of memory swapping. If you use the EMS or XMS option, TbScanX swaps itself to non-DOS memory, leaving only one kilobyte of code in DOS memory. It is better to swap to expanded memory (EMS option) because it is faster.

Use high memory if possible. If you have DOS 5 or higher, try to load the program into an upper memory block using the LOADHIGH or DEVICEHIGH commands. We recommend that you also enable swapping to limit the use of upper memory.

Use one of the processor specific versions of the relevant TBAV utility. They all consume less memory than the generic versions. Processor optimized versions are available on any ThunderBYTE support BBS.

Use memory-saving program options. Consider using TbDriver's NOSTACK option, TbMem's NOCANCEL option, and TbScanX's NOBOOT, EMS and XMS options.

4.2 Understanding TbSetup

This section presents advanced user information about TbSetup. It explains the design of ANTI-VIR data files, editing the TBSETUP.DAT file, and how to easily install TBAV on several machines.

4.2.1 Understanding ANTI-VIR.DAT File Design

Most ThunderBYTE Anti-Virus utilities expect every directory on your system with executable files to contain its own ANTI-VIR.DAT file. Some other anti-virus products maintain a somewhat similar fingerprint list of all executable files, but in one large file rather than a separate file in each directory. TBAV's approach is superior for several reasons:

One file in each directory is easy to maintain. If you want to remove the complete product, you can remove the accompanying ANTI-VIR.DAT file as well.

It consumes less disk space because it is not necessary to store full path information in the information file.

The TBAV utilities perform faster because they do not have to search through a huge file to locate the information for one specific file.

Installation is easier and more reliable in network environments. On a network, it is not unusual that the same files have different drive ID's on different workstations. If there is only one information file, the drive-IDs should be stored as well, so every workstation should maintain its own list. The supervisor can quickly lose control in this type of situation.

4.2.2 Editing the TBSETUP.DAT File

Editing the TBSETUP.DAT file is useful to TBAV site installation (see the next section). Therefore, some information on the format of this file is necessary.

Understanding the Format of TBSETUP.DAT

The format of the TbSetup.Dat file is quite simple. You can either ignore empty lines, lines starting with a semi-colon (;), and lines starting

with a percentage symbol (%), or you can treat them as comment lines. The lines with a preceding percentage symbol also appear in TbSetup's upper window.

Each entry in the TBSETUP.DAT file has four items:

1. The filename. The filename MUST appear in capital letters and without spaces.

2. The length of the file in hexadecimal notation. This field might contain a single asterisk [*] if an exact file length match is not required.

3. The file's 32-bit CRC in hexadecimal notation. You can use a single asterisk if an exact checksum match is not required.

4. The hexadecimal number representing flags you want set when the listed file is found on the system.

You can use the rest of the line for a brief comment.

You can use the following flags. If several flags require setting for a file, you can combine them using the bitwise OR operation:

```
bit 0:  (0001)  Do not perform heuristic analysis
bit 1:  (0002)  Ignore CRC changes (self-modifying file)
bit 2:  (0004)  Scan for all signatures (LAN remote boot file)
bit 3:  (0008)  Do not change read-only attribute of this file
bit 4:  (0010)  The program stays resident in memory
bit 5:  (0020)  The program performs direct disk access
bit 6:  (0040)  Program is allowed to remove read-only
                attributes
bit 15: (8000)  Interrupt rehook required for TBDRIVER.EXE
```

The following are a few example entries from a TBSETUP.DAT file:

```
; filename   Length 32-bit CRC Flags  Comment

; Files that trigger the heuristic alarm of TbScan:
4DOS.COM     19FEA       *  0001   ;4Dos 4.0a
AFD.COM      0FEFE  4B351A86  0001   ;AFD debugger
ARGV0FIX.COM 001D8  431E70C0  0001   ;Argv[0]fix
EXE2COM.EXE  00BEA  49276F89  0001   ;Exe to Com conv. util
KILL.EXE     00632  74D41811  0001   ;PcTools 6.0 utility
WATCH.COM    003E1  2353625D  0001   ;TSR monitoring util
```

```
; Files that need to be scanned completely, for ALL viruses:
NET$DOS.SYS     *       * 0004   ;Disk-image Novell boot

; Files without fixed checksum due to internal config area's:
Q.EXE           *       * 000A   ;Qedit (all versions)
TBCONFIG.COM    *       * 000A   ;all versions
```

Defining New Entries in TBSETUP.DAT

If you have any files that we should include in TBSETUP.DAT, please let us know! We would like to receive a copy to enhance our products and keep TBSETUP.DAT up to date. Candidates for inclusion are any programs that trigger the heuristic analysis of TbScan.

Whenever you choose "V)alidate program" in the TbScan message window, you will discover that on subsequent occasions TbSetup displays the value "0001" in the flags field. If your company has several files like this installed on multiple machines, you might want to include these files in the TBSETUP.DAT file yourself. To do this, execute TbSetup for the file in question and make a note of its file length and 32-bit CRC, as displayed on the screen. Then edit the TBSETUP.DAT file, entering the exact filename, the file length, and the CRC number, plus the number of any flags you wish to set for that file. If you now use TbSetup on another machine (using the updated TBSETUP.DAT file), it sets the appropriate flags automatically.

TIP:
You can manually set or clear a flag field value when executing TbSetup at the DOS prompt using the SET or RESET option as follows:

```
TBSETUP TEST.EXE SET=0001.
```

4.2.3 Simplifying Installation on Several Machines

If you need to install the TBAV utilities on several machines in one company, it would be tedious, for example, to run every TSR and disk utility on each machine to "teach" TBAV which programs are valid and which are not. Fortunately, this is not necessary. We present here some examples of how to simplify installation on several machines.

If a resident utility named, for example, TSRUTIL.EXE, is in use throughout the company, you can predefine permission by using TbSetup. First, use TbSetup to determine the length and CRC of the program.

Second, put the name of the program, along with its other information, in the TBSETUP.DAT file, and then assign the flag 0010 to it:

```
TSRUTIL.EXE 01286 E387AB21 0010 ;OUR TSR UTILITY
```

If a disk utility named, for example, DISKUTIL.EXE, is in use throughout the company, you can predefine permission by using TbSetup. First, use TbSetup to determine the length and CRC of the program. Second, put the name of the program, along with its other information, in the TBSETUP.DAT file, and then assign the flag 0020 to it:

```
DISKUTIL.EXE 01286 E387AB21 0020 ;OUR DISK UTILITY
```

If a utility named, for example, UTIL.EXE, causes TbScan to give false positives and is in use throughout the company, you can use TbSetup to "teach" TbScan to avoid heuristic scanning of the program. First, use TbSetup to determine the length and CRC of the program. Second, put the name of the program, along with its other information, in the TBSETUP.DAT file, and then assign the value 0001 to it:

```
UTIL.EXE   01286 E387AB21 0001 ;OUR UTILITY
```

If you now run TbSetup on every machine (you have to do this anyway), it recognizes the utilities you added in the TBSETUP.DAT file. Additionally, all the TBAV utilities automatically adapt their behavior for those files.

TIP:
Consult the TBSETUP.DAT file itself. It contains useful comments on this subject.

4.3 Understanding TbScan

This section offers advanced information about TbScan, including: heuristic scanning, integrity checking, program validation, algorithms, and the TBSCAN.LNG file.

4.3.1 Understanding Heuristic Scanning

What makes TbScan so unique is that it is not just a signature scanner, but it is also a disassembler. It disassembles files for the following purposes:

By disassembling a file, the scanner restricts itself to the area of the file where the virus might reside, reducing false alarms and speeding up the process. Disassembling a file makes it possible to use the algorithmic detection method on encrypted viruses whose signatures would otherwise remain invisible to the scanner.

Disassembling the file makes it possible to detect suspicious instruction sequences.

This detection of suspicious instruction sequences is "heuristic scanning." This extremely powerful feature enables you to detect new or modified viruses and to verify the results of the signature scan. You no longer have to rely on the scanner's publisher having the same virus as you might have. In normal cases a scanner can find a virus only if the scanner's publisher had a sample of that virus and includes that virus's signature in a signature file. In contrast, heuristic scanning does not require signatures, enabling the scanner to detect yet unknown viruses by looking for the characteristics of a virus instead of a signature.

Never underestimate the importance of heuristic scanning, since every month at least 50 new viruses are reported, and it is extremely unlikely that a publisher is the first one to get a new virus.

TbScan distinguishes two heuristic levels. The following table describes the properties of these levels:

Heuristic Level 1	Heuristic Level 2
always enabled	only enabled with command-line option "heuristic", or TBAV menu option "High heuristic sensitivity," or after a virus has been found
detects 50 % of (yet) unknown viruses	detects 90 % of (yet) unknown viruses
almost never causes false alarms	might cause few false alarms
displays "Probably infected"	displays "Might be infected"

The following lines show the effect of scanning four files, each having its own characteristics. Please note the heuristic flags that appear next to the word "scanning."

```
FILE1.EXE  scanning...OK    (no flags)
FILE2.EXE  scanning...ROK   (nothing serious)
FILE3.EXE  scanning...FRM   might be infected by unknown virus
FILE4.EXE  scanning...FRALM# probably infected by unknown virus
```

It is obvious from these four examples that heuristic scanning (resulting in the heuristic flags) is very powerful for finding yet unknown viruses.

4.3.2 Understanding How Heuristic Scanning Works

Every program contains instructions for the computer's microprocessor. By looking into the file's contents and interpreting the instructions, TbScan is able to detect the purpose of these instructions. If the purpose appears to be formatting a disk, or infecting a file, TbScan issues a warning. There are many instruction sequences that are very common for viruses but are very uncommon for normal programs. TbScan, therefore, assigns every suspicious instruction sequence to a character called a heuristic flag. Every heuristic flag denotes a score. If the total score (that is, the sum of scores for each flag that triggered) exceeds a predefined limit, TbScan assumes the file contains a virus.

There are actually two predefined limits. The first limit is quite sensitive and can be reached by some normal innocent programs. If the suspicious program reaches this limit, TbScan highlights the heuristic flags that appear on the screen and increases the suspicious item's counter. TbScan does not indicate the existence of a virus unless you specify the heuristic or high heuristic sensitivity option. If you do specify this option, TbScan informs you that the file Might be infected by an unknown virus.

In contrast to the first option, many viruses trigger the second heuristic limit, while normal programs do not. If a suspicious program reaches this limit, TbScan informs you that the file is Probably infected by an unknown virus.

NOTE:

TbScan performs heuristic analysis only near the entry-point of a file. Therefore, TbScan does not detect direct writes to disk by some disk utilities nor does it detect some programs as TSR programs. This is simply the result of a specific approach that minimizes false alarms. In case of a virus, the offending instructions are always near the entry-point (except when the virus is over 10Kb in size), so TbScan detects suspicious phenomena in these situations anyway.

4.3.3 Understanding Integrity Checking

TbScan performs integrity checking while scanning. For this purpose, you must use TbSetup to generate the ANTI-VIR.DAT files. Once these files exist on your system, TbScan verifies that every file being scanned matches the information maintained in the ANTI-VIR.DAT files. If a virus infects a file, the maintained information no longer matches the now changed file, and TbScan informs you of this.

NOTE:
There are no command line options to enable this feature. TbScan performs integrity checking automatically if it detects the ANTI-VIR.DAT files.

Note that TbScan reports only those file changes that could indicate a virus. While internal configuration areas of program files might also change, TbScan normally does not report these. If a file becomes infected with a known or unknown virus, however, the vital information does change and TbScan does indeed report it to you!

In contrast, there might be files that change themselves frequently or change frequently due to another cause. In such a case you might want to exclude the program from integrity checking to avoid future false alarms. If TbScan detects such a change, it informs you of it. Additionally, TbScan offers the possibility to Validate the program, which is the subject of the next section.

Understanding Program Validation This section applies only if you use TbSetup to generate the ANTI-VIR.DAT records. Without these records, program validation is not an option.

TbScan performs as intended on most programs. There are some programs, however, that require special attention in order to avoid false alarms. TbSetup recognizes most of these programs automatically. Nevertheless it is certainly possible your PC contains some program files that trigger the heuristic alarm of TbScan and/or programs files that change frequently.

If TbScan finds an infection using heuristic analysis or integrity checking, and if there is an ANTI-VIR.DAT record available, it offers an additional option in its virus-alert window, namely, V)alidate program.

If you are sure that the indicated program does not contain a virus, you can press V to set a flag in the program s ANTI-VIR.DAT record. This avoids future false alarms.

There are two validation modes. If TbScan alarms you to a file change, the validation applies to future file changes only. If the alarm is due to heuristic analysis, the validation applies only to heuristic results. If you exclude the file from heuristic analysis, TbScan still performs an integrity check. Conversely, if you exclude the file from integrity checking, TbScan still performs heuristic analysis.

CAUTION:

If you replaced a file (for example, because of a software upgrade) and you did not apply TbSetup to the changed files, TbScan pops up its virus alert window to inform you of the file change. Do not select the validation option in this case, because this would exclude the file from future integrity checking. You should abort TbScan and execute TbSetup on the changed file(s) instead.

4.3.4 Understanding the Scan Algorithms

When TbScan processes a file it displays one of the following messages:

Looking.

"Looking" indicates that TbScan has successfully located the entry point of the program in one step; that is, it has identified the program code so it knows where to search without the need of additional analysis. TbScan uses "Looking" on most known software.

Checking.

"Checking" indicates TbScan has successfully located the entry point of the program, and is scanning a frame of about two kilobytes around the entry point. If the file is infected, the virus signature appears in this area. "Checking" is a very fast and reliable scan algorithm, so TbScan applies it to most unknown software.

Tracing.

"Tracing" means that TbScan has successfully traced a chain of jumps or calls while locating the entry point of the program and is scanning a frame of about two kilobytes around this location. If the

file has been infected, the signature of the virus appears in this area. "Tracing" is a fast and reliable scan algorithm. TbScan uses it primarily for memory resident COM programs. Most viruses force TbScan to use "Tracing."

Scanning.

"Scanning" indicates that TbScan is scanning the entire file (except for the EXE-header that cannot contain any viral code). It uses this only if it can't safely use "Looking," "Checking," or "Tracing." Such is the case when the entry point of the program contains other jumps and calls to code located outside the scanning frame, or when the heuristic analyzer finds something that you should investigate more thoroughly. Because Scanning is a slow algorithm, it processes almost the entire file, including data areas, and it is more likely to trigger false alarms. TbScan uses this algorithm when scanning boot sectors, SYS files, and BIN files.

Skipping.

"Skipping" occurs only with SYS and OVL files. It simply means that the file will not be scanned. As there are many SYS files (such as CONFIG.SYS) that contain no code at all, it makes absolutely no sense to scan these files for viruses. The same applies to .OV? files. Many overlay files do not deserve the name overlay because they lack an EXE-header. Such files cannot execute through DOS, which in-turn makes them just as invulnerable to direct virus attacks as .TXT files. If TbScan reports that a virus has infected an .OV? file, that file is one of the relatively few overlay files that does contain an EXE-header. In such a case, the infection was the result of the virus monitoring the DOS exec-call (function 4Bh) and thereby infecting any program that executes that way, including real overlay files.

Decrypting.

TbScan detected that the file is encrypted, and decrypts it to be able to "look inside." TbScan performs signature scanning and heuristic analysis on the decrypted code since that is very reliable and also reveals polymorphic viruses.

4.3.5 Understanding the TBSCAN.LNG File

The TBSCAN.LNG file contains all the text that TbScan displays. You can translate or customize the messages with any ASCII editor. A dollar sign [$] separates the messages.

The first message displays our address and registration information. You can edit this message as you please, adding, for example, your company name and logo.

CAUTION:

Take care in customizing messages so that you don't change the essence of the message.

You can also add color codes to the TBSCAN.LNG file. You must precede a color code with the "pipe" [|] character. Each color code consists of a foreground (or highlight) color and a background color. The following table lists the available color codes (all numbers are in hexadecimal notation):

Color	Foreground	Highlight	Background
Black	00	08	00
Blue	01	09	10
Green	02	0A	20
Cyan	03	0B	30
Read	04	0C	40
Magenta	05	0D	50
Brown	06	0E (yellow)	60
Gray	07	0F (white)	70

To make characters blink, add 80 to the background color codes.

Here are few examples of defining colors:

To make a highlighted green character on a red background, use the color code 0A+40=4A. To make the character blink, add 80h to the result (4A+80=CA). To display white characters on a blue background, use the color code 0F in combination with color code 10: 0F+10=1F.

If you prefer a cyan background with a gray foreground, you should add 30 to 07 (30+07=37). If you want the characters to blink, the color code becomes 37+80=B7.

4.3.6 Understanding the TBAV.MSG File

The TBAV menu displays the contents of a file named TBAV.MSG, if it exists in the ThunderBYTE directory. You can use this feature to display your company logo on the TBAV screen. As in the TbScan language file, you can embed color codes in this file. Consult the previous section for more information about color codes.

4.4 Understanding TbClean

This section takes a look at how TbClean works by explaining how a virus goes about infecting a file and the difference between conventional cleaners and generic cleaners.

4.4.1 Understanding how a Virus infects a file

To understand how a cleaning program works, try to imagine how a virus usually goes about infecting a program. The basic principle is really quite simple. A virus, which is simply another computer program, adds itself to the end of the program it infects. The additional viral code obviously increases the size of the program.

Simply appending a viral program to another program, however, is not enough to do any real harm. To do damage, the viral code must first be executed. To accomplish this, the virus grabs the first few bytes at the start of the program and replaces them with a jump instruction to its own viral code. That way the virus is able to take control when the program starts. Chances are you will never even notice the momentary delay while the extra code executes and does whatever the virus has been programmed to do. The virus then restores the original instructions and restarts the program (jumps to the original start of the program). Your program, more often than not, works as usual, and of course, any virus worth its salt makes sure it doesn't draw undue attention to itself, at least not too soon.

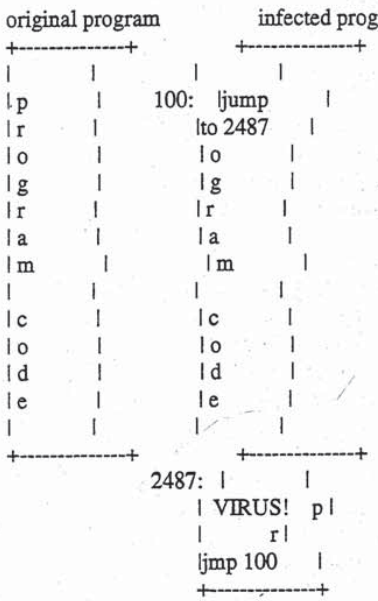
So, in order to purge a program, we must first restore the starting instruction bytes, which the virus replaced with the jump to its own code. The virus is going to need these bytes again later on, so it stores them somewhere in the viral code. The cleaner starts out to find those bytes, puts them back in their proper place, and trims the file to the original size.

Cleaner programs basically come in two types: the conventional type, for specific types of viruses, and the far more advanced generic cleaner, which offers a much wider scope. Let's take a closer look at both cleaner types and find out where they differ.

4.4.2 Understanding Conventional Cleaners

A conventional cleaner has to know which virus to remove. Suppose one of your programs is infected with a Jerusalem/PLO virus. This means that the

infected program has grown in size in comparison with the original program, and that the first few bytes have been replaced by a "jump" instruction to the viral code. The following drawing ilustrates this process:



When you start a conventional cleaner, a procedure much like the following takes place:

> "Hey, the signature file tells me this file is infected with the Jerusalem/PLO virus. Okay, let's see, this virus tacks on 1873 bytes at the end and overwrites the first three bytes of the original program with a jump to itself. The original bytes are located at offset 483 in the viral code. So, I have to take those bytes, copy them to the beginning of the file, and then remove 1873 bytes of the file. That's it!"

But there are several pitfalls to worry about in a scenario like this. For one thing, the cleaner obviously must have some means to recognize the virus it should remove. A conventional cleaner cannot cope with a virus unless it knows exactly what to look for.

To make matters worse, it's even more important to establish whether or not the virus is exactly the same one that the cleaner knows about. Imagine what would happen if the virus in our example had been modified

and is now 1869 bytes in size instead of 1873. The cleaner would remove too much! This is not an exceptional case at all. On the contrary, there is a virtual epidemic of countless so-called mutant strains. The Jerusalem/PLO family, to name but one example, now has more than 100 mutant members!

4.4.3 Understanding Generic Cleaners

A generic cleaner works on the principle that any kind of virus, whether or not it has made the signature "charts," is just plain bad news. That's why TbClean works with a completely different disinfection scheme that is effective with almost all viruses; it doesn't even need to recognize them. Actually, TbClean represents two cleaners in one: a "repair" cleaner and a "heuristic" cleaner.

Repair cleaning

Repair cleaning needs an ANTI-VIR.DAT file generated by TbSetup before the infection occurred. The ANTI-VIR.DAT file stores vital information about programs, including their original size, the first few instruction codes, and a cryptographic checksum. This information is usually all it takes to disinfect a file, no matter what virus, known or unknown, caused the infection. The cleaner simply restores the bytes at the beginning of the program, trims the file to its original size, and verifies the result using the original checksum. It's just that simple (and effective).

Heuristic cleaning

TbClean is the first cleaner in the world that has a heuristic cleaning mode. Like the repair cleaner, this mode does not need any information about viruses either, but it also has the added advantage that it doesn't even care about the original, uninfected state of a program. This cleaning mode is very effective if your system becomes infected with an unknown virus and you neglected to let TbSetup generate the ANTI-VIR.DAT files before infection.

In heuristic mode, TbClean loads the infected file and starts emulating the program code. It uses a combination of disassembly, emulation and, sometimes, execution to trace the flow of the viral code, pretending to do more or less exactly what the virus would normally be doing. When the virus gets to the original program's instructions and jumps back to the original program code, TbClean stops the emulation process, with a

tongue-in-check thank you to the virus for its cooperation in restoring the original bytes.

The actual cleaning process involves almost the same three steps as with repair cleaning. First, TbClean repairs the program startup code and copies it back to the file. Second, it removes the now ineffective code for the sake of security. Third, it does a final analysis of the purged program file.

4.5 Using TbGenSig

This final section of Chapter 4 introduces you to TbGenSig, an advanced user utility that enables you to define your own virus signatures.

4.5.1 Understanding and using TbGenSig

TbGenSig is a signature file compiler. Since we distribute TBAV with an up to date, ready-to-use signature file, you do not really need the signature file compiler.

If, however, you want to define your own virus signatures, you will need this utility. You can use either published signatures or define your own, if you are familiar with the structure of software.

One way or another, you need to do this only in case of an emergency, such as in the unfortunate event that a yet unknown, and thus unrecognized, virus attacks your machine, or even your company. We recommend that you send a few samples of the virus to some of our researchers, to insure that they can be examined and the results included in one of the subsequent updates to our software.

NOTE:
Since it's not possible to explain the whole subject of virus hunting in one manual, this section assumes you have enough experience and knowledge to create your own virus signatures.

TbGenSig searches for the USERSIG.DAT file in the current directory. This file should contain the signatures you want to add to the TBAV signature file TBSCAN.SIG. TbGenSig checks the contents of the USERSIG.DAT file and applies it to the TBSCAN.SIG file.

If you want to delete or modify your signatures, just edit or delete the USERSIG.DAT file and run TbGenSig again.

TbGenSig lists all signatures in the TBSCAN.SIG file on screen as it runs.

4.5.2 Working with TbGenSig

This section describes how to use TbGenSig. It outlines how to format the text in the USERSIG.DAT file, add published signatures, define your own signatures, and other procedures.

Formatting Text in USERSIG.DAT

You can create and edit the USERSIG.DAT file using any DOS text editor (such as DOS 5+ EDIT program) that uses un-formatted (ASCII) text. All lines starting with a semicolon (;) are comment lines. TbGenSig ignores these lines. Lines starting with a percentage character (%) appear in the upper TbGenSig window.

The first line should contain the name of a virus, the second line contains one or more keywords, and the third line contains the signature itself. We call this combination of three lines a signature record. A signature record should look like this:

```
TEST VIRUS
EXE COM INF
ABCD21436587ABCD
```

You can use spaces in the signature for your own convenience; TbGenSig will just ignore them.

Adding a Published Signature

As outlined above, adding an already published signature is simply a matter of editing or creating the USERSIG.DAT file to convert the signature to an acceptable format for TbGenSig. Format the three lines to include the virus name, keywords, and the signature, as in the following:

```
NEW VIRUS
EXE COM BOOT INF
1234ABCD5678EFAB
```

After editing the file, execute TbGenSig.

4.5.3 Defining a Signature with TbScan

This section is for advanced users who have registered their copy of ThunderBYTE Anti-Virus.

Although the TBSCAN.SIG file updates frequently, new viruses appear every day, outpacing the regular upgrading service of the TbScan signature file. It is possible for your system to become infected by a recently created virus not yet listed in the signature file. TbScan will not

always detect the virus in such cases, not even with its heuristic analysis. If you are sure that your system has become infected without TbScan confirming this, this section will supply you with a valuable tool to detect unknown viruses. This section offers step-by-step assistance in creating an emergency signature that you can (temporarily) add to your copy of TbScan.Sig

1. Collect some infected files and copy them into a temporary directory.

2. Boot from a clean write-protected diskette.

WARNING:
Do NOT execute ANY program from the infected system, even though you expect this program to be clean.

3. Execute TbScan from your write-protected TbScan diskette using the EXTRACT option. Make sure that the temporary directory where you stored the infected files is TbScan's target directory. Using the EXTRACT option, TbScan will NOT scan the files but, instead, displays the first instructions that it finds at the entry-point of the infected programs.

NOTE:
We recommend that you also set TbScan's LOG option to generate a log file.

4. Compare the "signatures" extracted by TbScan. You should see something like this:

```
NOVIRUS1.COM   2E67BCDEAB1290909 09090 ABCD123490CD
NOVIRUS2.COM   N/A
VIRUS1.COM     1234ABCD5678EFAB9 09090 ABCD123478FF
VIRUS2.COM     1234ABCD5678EFAB9 01234 ABCD123478FF
VIRUS3.COM     1234ABCD5678EFAB9 A5678 ABCD123478FF
```

If the "signatures" of the files are completely different, the files are either probably not infected, or they have become infected by a polymorphic virus that requires an algorithmic detection module to detect it.

5. If there are some differences in the "signatures," you can use the question mark wildcard (?). A signature to detect the virus in the example above could be:

1234ABCD5678EFAB ?3 ABCD123478FF

The "?3" means that there are three bytes at that position that should be skipped. Note that two digits in the signature represent a byte in your program.

6. Add the signature to USERSIG.DAT. Give the virus a name in the first line of its entry, specify the COM, EXE, INF, and ATE in the second line, and enter the signature in the third, as in the following:

NEW VIRUS
EXE COM ATE INF
1234ABCD5678EFAB?3ABCD123478FF

7. Run TbGenSig. Make sure the resulting TbScan.Sig file is in the TBSCAN directory.

8. Run TbScan again in the directory containing the infected files. TbScan should now detect the virus.

9. Send a couple of infected files to a recommended virus expert, preferably to the ThunderBYTE Corporation.

Congratulations! You have defined a signature all by yourself! Now you can scan all your machines in search of the new virus.

CAUTION:
Keep in mind that this method of extracting a signature is a "quick-and-dirty" solution to viral problems. The extracted signature might not detect the presence of the virus in all cases. You can make a signature guaranteed to detect all instances of the virus only after complete disassembly of the new virus. For these reasons you should NEVER distribute your home-made "signature" to others. In most cases, the signature eventually assembled by experienced anti-virus researchers may be different from your homemade version.

4.5.4 Understanding Keywords

You can use keywords for several purposes. You can separate them by spaces, commas, or tabs and use a maximum line length of 80 characters. You also should specify at least one of the following flags: BOOT, COM, EXE, HIGH, LOW, SYS, or WIN.

These seven flags fall into three categories: "Item Keywords," "Message Keywords," and "Position Keywords."

Using Item Keywords

Item keywords tell the scanner where to search for viruses with those keywords. For example, the BOOT keyword tells the scanner that the accompanying virus signature can reside only in a boot sector or partition table. The Item keywords include the following:

BOOT. Specifies that the signature can be found in boot sectors and/or partition tables.

COM. Specifies that the signature can be found in COM programs. This flag instructs the scanner to search for this signature in executable files that do not have an EXE header or device header.

NOTE: Always keep in mind that the file content determines the file type, not the filename extension!

EXE. Specifies that the signature can be found in EXE programs. This flag instructs the scanner to search for this signature in the load module of EXE type files. EXE files are files that have an EXE header. (See the Note under the COM keyword.)

HIGH. Specifies that the signature can be found in HIGH memory (above program).This flag instructs the scanner to search for this signature in memory above the memory allocated by the scanner. This keyword is for resident viruses that allocate memory at "system boot" or viruses that decrease the size of the last MCB (Memory Control Block). Please note that the flag HIGH does not mean that the signature should be searched in UPPER memory.

LOW. Specifies that the signature can be found in LOW memory. This flag instructs the scanner to search for this signature in memory below the PSP (Program Segment Prefix) of the scanner and in the UMBs (Upper Memory Blocks). This keyword is for viruses that remain resident in memory, using the normal DOS TSR (Terminate and Stay Resident) function calls.

SYS. Specifies that the signature can be found in SYS programs, such as device drivers.

WIN. Specifies that the signature can be found in Windows programs.

Message keywords

Message keywords describe the type and behavior of the virus. For each keyword, this results in the scanner displaying a different message when it finds such a virus. These keywords include the following:

DAM. Message prefix: damaged by.
DROP. Message prefix: dropper of.
FND. Message prefix: found the.
INF. Message prefix: infected by.
Message suffix: virus.
JOKE. Message prefix: joke named.
OVW. Message prefix: garbage: (not a virus).
PROB. Message pre-prefix: probably.
TROJ. Message prefix: trojanized by.

Position keywords

Position keywords indicate special file areas where the virus can be found. If you use a position keyword, the virus must reside at the specific position. TbGenSig can handle three position keywords:

UATE. Specifies that the signature starts directly at the unresolved entry-point of the viral code. With some polymorphic viruses, it might be possible to create a signature from the degarbling routine, although it might be either too short or give false positives with a global search. An initial branch instruction can be part of the signature. The unresolved entry-point is defined for COM-, EXE-, and Windows-type files:

COM type files: top of file (IP 0100h).

EXE type files: CS:IP as defined in the EXE-header.

WIN type files: Non-DOS CS:IP of the new EXE-header.

NOTE:

The UATE keyword is not allowed for BOOT, SYS, LOW, HMA, or HIGH type signatures.

ATE. Specifies that the signature starts directly at the entry-point of the viral code. With some polymorphic viruses, it might be possible to create a signature from the degarbling routine, although it might either be too short or give false positives with a global search. Therefore, use the ATE keyword to ensure that the scanners do not scan the entire file for the signature, but only look at the entry-point for the signature.

The first instruction that is not equal to either a "JUMP SHORT," a "JUMP," or a "CALL NEAR" instruction defines the entry point of a virus.

Let's examine the following code fragment:

```
Unresolved entry point:  1   JUMP SHORT 3
                         2   ...
                         3   JUMP 5
                         4   ...
                         5   CALL NEAR 7
                         6   ...
                         7   CALL NEAR 9
                         8   ...
Resolved entry point:    9   POP <reg>
```

The entry-point of the above fragment is Line 9, as this is the first instruction to execute that is not a "JUMP SHORT," a "JUMP," or a "CALL NEAR."

NOTE: You can determine the entry-point by a code analyzer to cope with tricks such as coding an NOP or DEC just before the branch instruction. Therefore test the results of the scanner carefully. In case of trouble, use the TbScan EXTRACT option to find out what TbScan considers to be the entry point of the program. Also, the ATE flag is not allowed for BOOT, SYS, LOW, HMA or HIGH type signatures.

XHD. Specifies that the signature can be found at offset 2 of the EXE header, but is rarely used. You should use it only to detect the also very rare high-level language viruses, viruses written in a programming language such as C or Basic. These viruses normally contain standard setup

routines and library routines that are not suitable to defining a signature. Use this keyword as a last resort to detect such viruses.

NOTE:
You can use this flag only for EXE or WIN type signatures.

Using Wildcards

You can use wildcards characters in a virus signature to recognize so called polymorphic (self-modifying or self-mutating) virus code. TbGenSig distinguishes two wildcard categories: position wildcards and opcode wildcards (note that all numbers are in hexadecimal):

Using Position Wildcards

Position wildcard affect the position where the parts of the signature match.

Skip fixed amount of bytes

?n Skip n bytes and continue.(0h <= n <= Fh)
?@nn Skip nn bytes and continue.(00h <= nn <= 7Fh)

Skip variable amount of bytes

*n Skip up to n bytes and continue. (0h <= n <= Fh)
*@nn Skip up to nn bytes and continue. (00h <= nn <= 1Fh)

Using Opcode wildcards

The opcode wildcards detect instruction ranges.

Low opcode
nL One of the interactions in the range of n0h to n7h.

High opcode

nH One of the interactions in the range of n8h up to nFh.

Since the opcode wildcards are rather difficult to understand, let's explore an example.

Suppose a polymorphic virus puts a value in a word register (using a MOV WREG,VALUE instruction), increments a register (using an INC WREG instruction), and pops a word register from the stack (using a POP instruction). Both the registers and the value are variable. This means that the signature you are writing to detect this virus should be able to detect all code sequences for every value of the registers and the value, but this is far too much work. Now, consider that B8-BF are the opcodes for MOV WREG,VALUE, that 40-47 are the opcodes for INC WREG, and that 58-5F are the opcodes for POP REG.

By using the opcode wildcards, you can detect a sequence of these three instructions using the following signature fragment:

```
bH4L5H
```

4.5.5 Understanding a Sample Signature: Haifa.Mozkin

To show the power of using the appropriate keywords and wildcards, here is the signature of the Haifa.Mozkin virus. This virus is highly polymorphic and encrypted. It contains a small variable decryptor to decrypt the virus.

There are two problems here: most bytes are encrypted or variable, thus not suitable to be part of a signature, and the remainder is short and would cause dozens of false alarms.

Using the appropriate keywords and wildcards, however, it s possible to define a reliable signature. TbScan actually uses the signature below to detect the Haifa.Mozkin virus.

```
Haifa.Mozkin
com exe ate inf
bh?2bh?109?2*22e80?24l4h75fl
```

Now let's analyze this signature. The first line describes the name of the virus. The second line tells the scanner to search for this signature in COM and EXE type files. It also tells the scanner that it should report the file as infected if the signature matches. The keyword ATE instructs the scanner to match this signature only at the resolved entry-point of the file. The virus starts, of course, by decrypting itself, so it is certain that the scanner will scan this location. The ATE instruction limits the scope of this signature to just one position in a file, so this significantly reduces the chances of false alarms.

The third line is the signature definition. Let's reverse engineer it:

bh?2 Means a byte in the B8-BF range is followed by two variable bytes. B8-BF is a MOV WREG,VALUE instruction. From the register we only know it is a word register; the value is unknown as well.

bh?109 Means another MOV WREG,VALUE instruction. The register is a word register, and from the value we know that it is in the range 0900 to 09FF.

?2*2 Means skip two to four bytes. The virus inserts this instruction to make it harder to define a signature.

2e80?2 Means that the virus performs an arithmetic byte sized operation with an immediate value (decrypts one byte) with a CS: segment override. The exact operation, the memory location, and the value are unknown.

4l Means a byte in the 40-47 range. This is an INC WREG instruction. The virus increments the counter to the next byte to be decrypted.

4h Means a byte in the 48-4F range. This is a DEC WREG instruction. The virus decrements the iteration count.

75fl Opcode 75 is a JNZ instruction. If the decremented register did not reach zero, the virus jumps back and repeats the operation. How much does it jump? That tells the fl part: somewhere between -16 (F0h) to -8 (F7h) bytes.

NOTE:

Although the signature language of TbGenSig is extremely powerful, there are viruses that are simply so highly polymorphic that they require even more sophisticated wildcards, keywords, or even special detection algorithms. The explanation of these wildcards, keywords, and algorithmic detection definitions, however, is beyond the scope of this user manual.

Appendices

Appendix A: TBAV messages

The TBAV utilities might display various messages when run. Most messages are self-explanatory, but here is some additional information about specific messages.

A.1 TbClean

ANTI-VIR.DAT record found: information matches the currentstate of the file.
The ANTI-VIR.DAT record has been found, but the information matches the current state of the file.

The ANTI-VIR.DAT file was created after the infection. Trying emulation...
The ANTI-VIR.DAT record was created after the file became infected, or the file is not changed at all. TbClean is going to emulate the file to clean it heuristically.

ANTI-VIR.DAT record found: reconstructing original state...
The ANTI-VIR.DAT record that belongs to the infected file has been found. The information will be used to reconstruct the file.

ANTI-VIR.DAT record not found: original state unknown. Trying emulation...
The ANTI-VIR.DAT file did not exist or did not contain information about the infected program, so the original state of the infected program is unknown to TbClean. TbClean switches to its heuristic mode to determine the state of the original file.

NOTE:
To prevent this situation, use the TbSetup program to generate the ANTI-VIR.DAT records. These records are of great help to TbClean. After infection, it's too late to generate the ANTI-VIR.DAT records.

Emulation terminated: <Reason>
The emulation process terminated for the reason specified. TbClean now consults the collected information to see if it can disinfect the file. The reason for termination can be one of the following:

Jump to BIOS code. The virus tried to perform a call or jump directly into BIOS code. TBAV cannot emulate this process, so aborts. The infected program probably cannot be disinfected.

Approached stack crash. The emulated program is approaching a crash. Something went wrong while emulating the program so it aborts. The infected program probably cannot be disinfected.

Attempt to violate license agreements. TbClean will not disassemble this program for obvious reasons.

Encountered keyboard input request. The emulated program tries to read the keyboard. This is very unusual for viruses, so the file is probably not infected at all.
Encountered an invalid instruction. The emulator encountered an unknown instruction. For some reason the emulation failed. The infected program probably cannot be disinfected.

DOS program-terminate request. The emulated program requests DOS to stop execution. The program is either not infected at all, or infected by an overwriting virus that does not pass control to its host program. The infected program cannot be disinfected.

Jumped to original program entry point. The program jumped back to the start position. It is very likely infected, but can probably be disinfected.

Undocumented DOS call with pointers to relocated code. This is very common for viruses that add themselves in front of the COM type program. The program can probably be disinfected.

Encountered an endless loop. TbClean encountered a situation in which the program is executing the same instruction sequences repeatedly for hundreds of thousands of times. It is unlikely that the program will ever escape from this loop, so the emulation aborts.

Ctrl-break pressed. The user pressed <Ctrl>-<Break> so the clean attempt aborts.

Emulation aborted for unknown reason. If this message appears, please send a copy of the file being emulated to the ThunderBYTE organization or one of the support BBS .

Sorry, the collected information is not sufficient to clean file... The heuristic cleaning mode of TbClean aborts with success. The only option left is to restore the file from a backup or to re-install the program.

Collected enough information to attempt a reliable clean operation... The emulation of the virus provided TbClean with all information needed to disinfect the file.

Some DOS error occurred. TbClean aborted! Some DOS error occurred while trying to clean the file. Check that no files are read-only or located on a write protected disk, and make sure there is a reasonable amount of free disk space.

The clean attempt seems to be successful. Test the file carefully! TbClean thoroughly and reliably removed the virus from the file. However, take care and test the file carefully to see if it works as correctly.

Reconstruction failed. Program might be overwritten. Trying emulation... TbClean tried to reconstruct the original file with the help of the ANTI-VIR.DAT record, but the attempt failed. TbClean is going to emulate the file to try to clean it heuristically.

Reconstruction successfully completed. TbClean has reconstructed the file to its original state with the help of the information in the ANTI-VIR.DAT record. The CRC (checksum) of the original file and the cleaned file are completely equal, so it is almost certain that the cleaned file is equal to the original file.

Starting clean attempt. Analyzing infected file... TbClean is analyzing the infected file and trying to locate the ANTI-VIR.DAT record.

A.2 TbDriver

Another version of TbDriver is already resident!
You started a TBDRIVER.EXE with another version number or processor type than the TbDriver already in memory.

Cannot remove TbDriver. Unload other TSRs first!

You tried to remove TbDriver from memory, but other resident software was loaded after TbDriver. You can only remove resident programs from memory by unloading them in reverse order.

LAN support was already installed.
You tried to use the NET option a second time, or TbDriver already enabled network support automatically.

TbDriver not active. Load TbDriver first!
The resident TBAV utilities need TbDriver, so you need to load TbDriver first.

TbDriver is not <version>.
The version of TbDriver found in memory does not match the version number of this resident TBAV utility. Be sure you do not mix version numbers!

This version of TbDriver requires a <typeID> processor.
You are using a processor optimized version of TbDriver that the current processor cannot execute.

A.3 TbScan

Cannot create logfile.
The specified log file path is illegal, the disk is full or write protected, or the file already exists and cannot be overwritten.

[Cannot read datafile]
TbScan needs access to its data file to be able to tell you the name of the virus. If it cannot access the data file, it displays this message instead of the virus.

Command line error.
You specified an invalid or illegal command line option.

No matching executable files found.
The specified path does not exist, is empty, or is not an executable file.

Sanity check failed!
TbScan detected that its internal checksum no longer matches. It is possible that TbScan is contaminated by a virus. Obtain a clean copy of TbScan, copy the program on a write protected system diskette, boot from that diskette, and try again.

A.4 TbScanX

Data file not found.
TbScanX cannot locate the data file.

Not enough memory.
There is not enough free memory to process the data file. Try to enable swapping, or if you are already doing so, try another swapping mode. See also the Understanding Memory Considerations section in Chapter 4.

Appendix B: TbScan Heuristic Flag Descriptions

This appendix describes TBAV's heuristic flags.

- Decryptor code found
The file possibly contains a self-decryption routine. Some copy-protected software is encrypted, so this warning might appear for some of your files. If, however, this warning appears in combination with, for example, the "T" warning, there could be a virus involved and TbScan assumes contamination. Many viruses encrypt themselves and trigger this warning.

! - Invalid program.
Invalid opcode (non-8088 instructions) or out-of-range branch. The program has either an entry point that located outside the body of the file, or reveals a chain of jumps that can be traced to a location outside the program file. Another possibility is that the program contains invalid processor instructions. The program being checked is probably damaged and cannot execute in most cases. At any rate, TbScan avoids risk and uses the scan method to scan the file.

1 - 80186+ instructions.
The file contains instructions which cannot be executed by 8088 processors, and require an 80186 or better processor.

@ - Strange instructions
The file contains instructions which are not likely to be generated by an assembler, but by some code generator like a polymorphic virus instead.

? - Inconsistent header.
The program being processed has an EXE-header that does not reflect the actual program lay-out. Many viruses do not update the EXE-header of an EXE file correctly after they infect the file, so if this warning pops up frequently, it appears you have a problem.

c - No integrity check
This warning indicates that TBAV found no checksum/recovery information for the indicated file. We recommend you use TbSetup in this case to store the file's information. TBAV uses this information for integrity checking and to recover from virus infections.

h - Hidden or System file.
The file has the Hidden or the System file attribute set. This means that the file is not visible in a DOS directory display but TbScan scans it anyway. If you don t know the origin and/or purpose of this file, you

might be dealing with a Trojan Horse or a joke virus program. Copy such a file onto a diskette, remove it from its program environment, and then check if the program concerned is missing the file. If a program does not miss it, you not only have freed some disk space, but you might also have prevented a future disaster.

i - Internal overlay.
The program being processed has additional data or code behind the load-module as specified in the EXE-header of the file. The program might have internal overlay(s) or configuration or debug information appended behind the load-module of the EXE file.

p - Packed or compressed file.
This means that the program is packed or compressed. There are some utilities that can compress program files, such as EXEPACK and PKLITE. If the file became infected after compression, TbScan is able to detect the virus. However, if the file became infected before compression, the virus was also compressed in the process, and a virus scanner might no longer be able to recognize the virus. Fortunately, this does not happen very often, but you should still beware! A new program might look clean, but can turn out to be the carrier of a compressed virus. Other files in your system will become infected too, and it is these infections that will be clearly visible to virus scanners.

w - Windows or OS/2 header.
The program can be or is intended to run in a Windows (or OS/2) environment. TbScan offers a specialized scanning method for these files.

A - Suspicious Memory Allocation
The program uses a non-standard way to search for, and/or to allocate memory. Many viruses try to hide themselves in memory, so they use a non-standard way to allocate this memory. Some programs (such as high-loaders or diagnostic software) also use non-standard ways to search or allocate memory.

B - Back to entry.
The program seems to execute some code, and after that jumps back to the entry-point of the program. Normally this results in an endless loop, except when the program also modifies some of its instructions. This is quite common behavior for computer viruses. In combination with any other flag, TbScan reports a virus.

C - File has been changed
This warning appears only if you use TbSetup to generate the ANTI-VIR.DAT files and means the file has been changed. Upgrading the software would

trigger this message. Otherwise, it is very likely that a virus infected the file!
NOTE:
TbScan does not display this warning if only some internal configuration area of the file changes. This warning means that code at the program entry point, the entry-point itself, and/or the file size has been changed.

D - Direct disk access
This flag appears if the program being processed has instructions near the entry-point to write to a disk directly. It is quite normal that some disk related utilities trigger this flag. If several files that should not be writing directly to the disk trigger this flag, your system might be infected by an unknown virus.
NOTE:
A program that accesses the disk directly does not always have the "D" flag. Only when the direct disk instructions are near the program entry point does TbScan report it. If a virus is at fault, the harmful instructions are always near the entry point, so it is only there that TbScan looks for them.

E - Flexible Entry-point
This flag indicates that the program starts with a routine that determines its location within the program file. This is rather suspicious because sound programs have a fixed entry-point so they do not have to determine this location. For viruses, however, this is quite common. Approximately 50% of the known viruses trigger this flag.

F - Suspicious file access
TbScan has found instruction sequences common to infection schemes that viruses use. This flag appears with those programs that are able to create or modify existing files.

G - Garbage instructions.
The program contains code that seems to have no purpose other than encryption or avoiding recognition by virus scanners. In most cases there won't be any other flag since the file is encrypted and the instructions are hidden.
NOTE:
This flag appears occasionally on "normal" files. This simply indicates, however, that these are poorly designed, not infected..
J - Suspicious jump construct.
The program did not start at the program entry point. The code has either jumped at least twice before reaching the final startup code, or the program jumped using an indirect operand. Sound programs should not

display this kind of strange behavior. If several files trigger this flag, you should investigate your system thoroughly.

K - Unusual stack.
The EXE file being processed has an odd (instead of even) stack offset or a suspicious stack segment. Many viruses are quite buggy by setting up an illegal stack value.

L - Program load trap
The program might trap the execution of other software. If the file also triggers the "M" flag (memory resident code), it is very likely that the file is a resident program that determines when another program executes. Many viruses trap the program load and use it to infect the program. Some anti-virus utilities also trap the program load.

M - Memory resident code.
TbScan has found instruction sequences that could cause the program to hook into important interrupts. Many TSR (Terminate and Stay Resident) programs trigger this flag because hooking into interrupts is part of their usual behavior. If several non-TSR programs trigger this warning flag, however, you should be suspicious. It is likely that a virus that remains resident in memory infected your files.
NOTE:
- This warning does not appear with all true TSR programs, nor can you always rely upon TSR detection in non-TSR programs.

N - Wrong name extension.
Indicates a name conflict; that is, the program carries the extension .EXE but appears to be an ordinary .COM file, or it has the extension .COM but the internal layout of an .EXE file. A wrong name extension might in some cases indicate a virus, but in most cases it does not.

O - code Overwrite.
This flag appears if TbScan detects that the program overwrites some of its instructions. However, it does not seem to have a complete (de)cryptor routine.

R - Suspicious relocator
Indicates a suspicious relocator. A relocator is a sequence of instructions that changes the proportion of CS:IP. Viruses often use this. Those viruses have to relocate the CS:IP proportion because they were compiled for a specific location in the executable file; a virus that infects another program can hardly ever use its original location in the file as it is appended to this file. Sound programs know their location in the executable file, so they don t have to relocate

themselves. On systems that operate normally, only a small percentage of the programs should trigger this flag.

S - Search for executables
The program searches for *.COM or *.EXE files. This by itself does not indicate a virus, but it is an ingredient of most viruses, since they have to search for suitable files to spread themselves. If accompanied by other flags, TbScan assumes the file is infected by a virus.

T - Invalid timestamp.
The timestamp of the program is invalid; that is, the number of seconds in the time stamp is illegal, or the date is illegal or later than the year 2000. This is suspicious because many viruses set the time stamp to an illegal value (such as 62 seconds) to mark that they already infected the file so they won't infect a file a second time. It is possible that the program being checked is contaminated with a virus that is still unknown, especially if several files on your system have an invalid time stamp. If only very few programs have an invalid time stamp, you d better correct it and scan frequently to check that the time stamp of the files remains valid.

U - Undocumented system call.
The program uses unknown DOS calls or interrupts. These unknown calls can be issued to invoke undocumented DOS features, or to communicate with an unknown driver in memory. Since many viruses use undocumented DOS features, or communicate with memory resident parts of a previously loaded instance of the virus, a program is suspicious if it performs unknown or undocumented communications. This does not necessarily indicate a virus, however, since some tricky programs also use undocumented features.

V - Validated program
The program has been validated to avoid false alarms. The design of this program would normally cause a false alarm by the heuristic scan mode of TbScan, or this program might change frequently, and TbScan excludes the file from integrity checking. Either TbSetup (automatically) or by TbScan (manually) stores these exclusions in the ANTI-VIR.DAT.

Y - Invalid boot sector.
The boot sector is not completely according to the IBM defined boot sector format. It is possible that the boot sector contains a virus or has been corrupted.
Z - EXE/COM determinator.
The program seems to check whether a file is a COM or EXE type program. Infecting a COM file is a process that is not similar to infecting an EXE

file, which implies that viruses able to infect both program types should also be able to distinguish between them. There are, of course, innocent programs that need to find out whether a file is a COM or EXE file. Executable file compressors, EXE2COM, converters, debuggers, and high-loaders are examples of programs that might contain a routine to distinguish between EXE and COM files.

Appendix C: Solving Incompatibility Problems

Although TBAV utilities cooperate very well with other resident software, other software might not behave so well. This can cause system errors or even more serious problems. This section describes some common problems and their solutions.

PROBLEM:
If a TBAV utility tries to display a message, the text message "file <filename> could not be opened" appears.

Specify the FULL path and filename of the file to use as a message file after the TbDriver loading command. The default file name is TBDRIVER.LNG.

PROBLEM:
One of your utilities is loading a TSR into memory without an executable filename extension, such as .EXE or .COM. Since TbSetup creates ANTI-VIR.DAT records only for files with an executable extension, there is no ANTI-VIR.DAT, so TbMem is not able to record the TSR permission information.

Run TbSetup and specify the exact filename of the TSR. TbSetup creates an ANTI-VIR.DAT record, regardless of the filename extension, so TbMem can now record its information.

Although the ANTI-VIR.DAT record exists, TbScan does not use it to check the CRC to avoid false alarms.

PROBLEM:
You are running a network, and one of the following problems arises:

1. TbScanX is installed, but does not display the *scanning* message while accessing files. It also does not detect viruses.

2. TbCheck is installed, but does not display the *checking* message while accessing files. It also does not detect viruses.
3. TbFile is installed, but does not detect anything.
4. TbMem is installed, but does not detect TSRs.

Use the "TbDriver net" command after the network loads.

PROBLEM:
The system sometimes hangs when the message *scanning* is on the screen.

Try TbScanX without the EMS or XMS option. If TbScanX now works without any problems, add the EMS or XMS option again along with the COMPAT option. On some systems, you cannot use the TbScanX XMS option at all because these systems do not allow resident software to use extended memory.

If the problem relates to the XMS option and still occurs when you use the COMPAT option, you can use the XMSSEG = <VALUE> option to change the XMS swap segment address. The value should be between 2000 and 8000. The default value is 4000.

PROBLEM:

After you have given permission for a program to remain resident in memory, TbMem asks the same question the next time.

First, the SECURE option of TbDriver is in use. Remove this option, reboot and try again.

Second, the program mentioned does not appear in the ANTI-VIR.DAT file and, therefore, TbMem cannot permanently store the permission flag. Use TbSetup first to generate this program's ANTI-VIR.DAT record.

Third, for some reason it is not possible to write to the Anti-Vir.Dat file. The file might reside on a write protected diskette, on a network in a read-only directory, or the Anti-Vir.Dat file has the read-only attribute set.

PROBLEM:

The system sometimes hangs when you answer "YES" (abort program) to a TbMem message.

A solution here is difficult. Some resident programs seriously interfere with the system, and once rejected from memory, the system becomes unstable.

PROBLEM:

When you load TbDisk from the DOS command prompt, everything works fine. When you install TbDisk from within the CONFIG.SYS or AUTOEXEC.BAT file, however, it continually warns that programs write to disk directly.

Load TbDisk at the end of your AUTOEXEC.BAT file.

PROBLEM:

You formatted the hard disk using DOS FORMAT, but TbDisk did not display a message until the process was almost complete.

This is not a problem. A high level format program such as DOS's FORMAT.COM does not actually format the disk (that is, divide the disk into tracks and sectors), rather it reads all tracks to locate possible bad spots and clears the FAT and directory structure. Only this last step implies a disk write, so it is the only one TbDisk detects.

PROBLEM:
After you give permission for a program to perform direct disk access, TbDisk asks the same question the next time.

First, the SECURE option of TbDriver is in use. Remove this option, reboot and try again.

Second, the program mentioned does not appear in the ANTI-VIR.DAT file and therefore TbDisk can not permanently store the permission flag. Use TbSetup first to generate this program's ANTI-VIR.DAT record.

PROBLEM:
If you try to use Windows fast 32-bit disk access, Windows displays an error message.

Use the WIN32 option on the TbDisk command line.

Appendix D: TBAV Exit Codes and Batch Files

All TBAV utilities return to DOS with an error code that you can use with DOS's ERRORLEVEL command. The chief use of these error codes is in batch files. This appendix lists these error codes. Consult your DOS manual for information how to use error codes in batch files.

D.1 TbScan Exit Codes

TbScan terminates with one of the following exit codes:

Errorlevel	Description
0	No viruses found/ No error occurred
1	No files found
2	An error occurred
3	Files have changed
4	Virus found using heuristic analysis
5	Virus found using signature scanning
255	Sanity check failed

D.2 TbUtil Exit Codes

TbUtil terminates with one of the following exit codes:

Errorlevel	Description
0	No error occurred
1	Option "compare" failed/An error occurred

D.3 General Exit Codes

All the TBAV utilities except TbScan and TbUtil (see above) exit with one of the following exit codes:

Errorlevel	Description
0	No error occurred
1	A error occurred

D.4 Program Installation Check

To detect within a batch file whether a resident TBAV utility loaded, you can check for the device names. All TBAV utilities install a device name, whether they load from CONFIG.SYS or AUTOEXEC.BAT.

You can use the DOS IF EXIST batch file command to check for the device names. The following example, illustrating a part of a batch file, uses this construction to test whether TbScanX is loaded:

```
@ECHO OFF
IF NOT EXIST SCANX ECHO TBSCANX HAS NOT BEEN LOADED!
```

You could also branch to a label by using the GOTO command:

```
@ECHO OFF
IF NOT EXIST SCANX GOTO NOSCANX
ECHO TBSCANX EXISTS !
GOTO END
:NOSCANX
ECHO TBSCANX DOES NOT EXIST !
:END
```

Finally, the following table lists the device names used by the TBAV utilities:

TBAV program	Device name
TbScanX	SCANX
TbCheck	TBCHKXXX
TbMem	TBMEMXXX
TbFile	TBFILXXX
TbDisk	TBDSKXXX
TbLog	TBLOGXXX

Appendix E: Virus Detection and Naming

E.1 How Many Viruses Does TbScan Detect?

Most of the TbScan signatures are family signatures; that is, one signature detects an entire set of viruses. All these viruses relate to one another. The Jerusalem signature, for example, covers more than 100 viruses. For this reason, there is no way of knowing how many viruses TbScan detects.

Some competitive products treat each virus mutant as a separate virus, thus claiming to detect over 4000 viruses. TbScan, however, detects viruses using only 2000 signatures. If you want to compare virus scanners, you have to rely on the tests frequently published in magazines.

E.2 The Virus Naming Convention

TbScan follows the CARO virus naming recommendations. CARO is an organization in which leading anti-virus researchers participate. The CARO approach groups viruses in a hierarchical tree, which indicates to which family viruses belong. TbScan shows the complete CARO name where possible.

In contrast, however, many other anti-virus products simply indicate the family name or the member name. For example, many products might refer to the Leprosy.Seneca.493 using the family name Leprosy or member name Seneca, or even by the variant name 493. Worse yet, anti-virus products developed by non CARO members might even use a completely different name.

TbScan, however, tries to display as much of the name as possible. Building on the previous example, if TbScan can t distinguish between the Leprosy.Seneca.493 and Leprosy.Seneca.517 viruses, it indicates both by the name Leprosy.Seneca

Some viruses mutate themselves frequently. To detect all instances of such a virus, it is sometimes necessary to use multiple signatures. Although these signatures cover exactly the same virus, they do have a slightly different indication. Behind the name of the virus you will see a number in angle brackets. This number has nothing to do with the name of the virus, but is there just for maintenance reasons.

Index