Fortinet-1017

December 1991 ISSN 0956-9979

VIRUS BULLETIN

THE AUTHORITATIVE INTERNATIONAL PUBLICATION
ON COMPUTER VIRUS PREVENTION,
RECOGNITION AND REMOVAL

CONTENTS

GUEST EDITORIAL

Steve R. White

The Mother Of All False Positives

In last month's *VB* (November 1991), *Virus Bulletin* published a signature for the Gosia virus, a new virus which has not been seen in any actual infection incident. Shortly thereafter, *VB* discovered that the signature incorrectly identified COMMAND.COM from DOS 3.3 and above as being infected. *VB's* signature gave a 'false positive' indication on this almost universal file. (See '*Living Together - Without False Alarms!*, *VB*, November 1991, pp. 19-20.)

It could have happened to anyone. Indeed, it already has. Most anti-virus vendors have had false positive problems. To its credit, *VB* went out of its way to notify its subscribers and ward off potential problems. It warned people not to use the signature by telephone, FAX, e-mail and letter. Others have not been as forthcoming.

False positives are a common occurrence in the anti-virus industry. But should they be? More precisely, should customers resign themselves to periodic 'virus outbreaks' that turn out to be caused by faulty signatures in anti-virus programs? I don't think so. It's time to stop asking customers to do quality assurance for the anti-virus industry. It's time for vendors to prevent their problems from becoming their customer's problems.

Like manic butterfly collectors, the industry has become obsessive about how many new specimens it can find. Never mind that most viruses it finds have never been seen in an actual incident. Never mind that many viruses are so buggy that we can't make them spread even if we try. Advertising copy, as well as reviews of anti-virus software (including some of those conducted by *VB*), have concentrated on the *quantity* of viruses detected rather than the overall *quality* of the protection provided. We have, perhaps, lost track of the real problem - reducing the *actual* risk to which our customers are exposed.

Anti-virus vendors can be more careful in a variety of ways. The obvious thing to do is to select signatures carefully. Longer signatures can provide a margin of safety, since a longer sequence of bytes is less likely to be found in an arbitrary program than is a shorter sequence. (See '*Selecting and Testing Virus Patterns*', *VB*, September 1991, pp. 3-4. Note, however, that *VB's* testing did not include false positive testing at this time!)

Signatures should be carefully tested before they are released, on as large a corpus of uninfected programs as possible, to help reduce the incidence of obvious false positives. At *IBM*, we have been testing our signatures on a corpus of hundreds of megabytes of normal programs, written in dozens of languages, for some time now. (This simple step would have prevented *VB's* Gosia problem. One hopes that *VB*, and everyone else, will introduce stringent testing of signatures in the future!)

It is also important to write anti-virus programs in such a way as to avoid their being identified as infected by other anti-virus programs. Most vendors have already learned not to leave unaltered binary signatures in memory for just this reason. Most but not all. (See '*Troublesome Concubines in the Anti-Virus Harem*', *VB*, November 1991, p. 18.)

But even this is not enough. It's not possible to have *every* program in the world in your corpus and new programs may be written in the future which cause false positive problems with today's anti-virus software. What is needed is a way to characterize 'normal' programs and determine that a particular signature is unlikely to be found in them. Our lab has made good progress on this problem recently. When we received *VB's* errant Gosia signature, our characterization programs identified it as very likely to be found in normal programs, even apart from COMMAND.COM.

Another approach is to use verification programs, to certify that a virus is byte-by-byte identical to the virus indicated by the signature. (See '*Virus Verification and Removal*', *VB*, November 1991, pp. 7-11.) This is a two-edged sword. It can eliminate false positives if a signature is mistakenly found in a normal program and that is useful. The downside of this approach, is that if anti-virus software only reports *exact* matches, it may miss new viruses that are small variants of existing viruses, which would be unfortunate.

Beyond our individual actions, the industry can help encourage higher quality in anti-virus software. Reviews should be expanded to include false positive testing. Customers should know which products have been prone to problems and what, if anything, the vendor has done to solve them. Vendors should reduce their reliance on the 'numbers game' in advertising. They should encourage customers to buy their product because it reduces the *real* risks, doesn't cause new problems, and is easy to install, use and update.

Finally, the industry should encourage an open discussion of what, exactly, it means to have a high quality anti-virus product, both to educate its customers and to raise its own consciousness.

Yes, it could have happened to anyone. But let's take this opportunity to ensure that it doesn't continue to happen to our customers.

[Steve White is Manager of *IBM's High Integrity Computing Laboratory* based at the *Thomas J. Watson Research Center*, Yorktown Heights, New York.]

TECHNICAL NOTES

Gosia

The identification string published for the Gosia virus in the November edition of *VB* turned out to be unusable, as it produced a false positive in a program which is found in virtually all MS-DOS machines, namely COMMAND.COM.

An examination of the virus revealed that it contains a block of code which is copied from COMMAND.COM, but is not used by the virus. Unfortunately, last month's string was selected from this area, which seemed a natural choice - containing rather unusual code, which was not thought likely to be found in any program at random.

The following amended string should be used instead:

```
Gosia  8BD6 81C2 7001 B001 B900 00B4 43CD 2172
       358B D681 C270 01B0
```

Form Disinfection

In last month's *VB* it was erroneously stated that the widespread Form virus (*see the Prevalence Table overleaf*) relocates the original DOS Boot Sector to the last sector of the active DOS partition. In fact, the virus relocates the DOS Boot Sector to the very last sector *on disk*: the virus code itself is stored in the first sector of the active DOS partition and the penultimate sector on disk.

DOS Compatibility

Hardened *VB* readers know that the magic object in detecting and removing computer viruses is the clean write-protected system diskette. However, it is important that DOS compatibility is considered when booting the PC.

Machines must be booted from the same version of DOS (or a *higher* version of DOS) than is on the PC itself.

Prior to DOS 4.xx, the operating system could only manage 32 Megabyte partitions. DOS 4.xx (and upwards) introduced an expanded form of sector editing as well as the ability to handle volumes larger than 32 Megabytes. Thus booting a machine running DOS 4.xx from a DOS 3.xx system diskette can result in spurious results as the operating system attempts to contend with unmanageable partitions.

There are a few exceptions to this rule - for instance the editor's PC runs under Compaq DOS 3.31 which manages volumes larger than 32 Megabytes gracefully. The same is also true of Zenith DOS 3.30 and upwards.

Spurious results can also be expected if, for example, a PC running DR-DOS from *Digital Research* is booted with a version of DOS from *Microsoft* or *IBM* etc.

Smart Scanners Not So Brainy After All...

Viruses generally modify the first instruction of the programs they infect, - some viruses add code at the front of COM files, others overwrite the beginning of programs, or modify the initial CS:IP instruction (in the case of .EXE files). All these types of modifications result in the first few instructions executed being different after infection than before.

The new Brainy virus is an exception to this rule. It infects .COM files which start with a JMP instruction. This instruction is not changed, but the virus code is inserted into the program at the target address of this JMP instruction.

Brainy differs from the majority of .COM infectors in an important way - the virus code can be found *anywhere* in the file, so virus scanners which only search a small block (usually 2000-4000 bytes) at the beginning and end of programs will not find it. This is not a radical new development, as the Bulgarian 800 virus works in a similar way, although it overwrites the first 3 bytes with a JMP to the virus.

It used to be possible to check for a virus infection simply by looking at the first instruction executed and a few bytes following it. These could then be compared with their original values, but Brainy invalidates this approach.

One could envisage a virus which would combine features of Brainy (where no changes are introduced at the beginning or end of a file), and of the 'Number of the Beast' virus (where no visible increase in file length is apparent, even when the virus is not active in memory). Such a virus would render obsolete any checksumming program which only checked blocks at the beginning and end of a program, as well as verifying that the program size was unchanged. Some checksumming software available today provides exactly this method as a 'Quick' option, but although it is faster than checking the entire file it is not fool-proof.

Correctly implemented cryptographic checksumming, by necessity, involves the creation of a checksum value of the *entire* executable image followed by a *full* comparison of that image on each and every subsequent check.

Semi-Stealth Viruses

The terms 'Semi-Stealth' and 'Sub-Stealth' have been used to describe those parasitic viruses which fulfil *one* of *two* essential stealth criteria: they change the length of infected programs and subsequently make the increase in file length 'disappear' when a DIR command is issued.

In stark contrast to those viruses which are *fully stealth*, semi-stealth viruses do not present a serious problem to anti-virus software developers, even when active in memory. A checksumming program will report that the infected program has been altered while a virus scanner is able to read and analyse the program without difficulty.

8-10-16-24 Bytes and Climbing

A few months ago an editorial decision was made to extend *VB* virus identification patterns from 16 bytes to 24 bytes. (Back in July 1989, it was thought sufficient to publish identification patterns of just 8-10 bytes!)

Longer patterns have a lower chance of occurring by chance in a non-infected program and increasing the average length of the identification string will generally have the effect of reducing false positive indications. Using longer strings has some side-effects, not all of which are necessarily desirable.

If long strings are used, it is not always possible to find 'generic' patterns for an entire family of viruses - separate patterns may have to be used for each variant. Some manufacturers of virus scanners actually prefer this, as the search patterns thus provide a primitive form of automatic variant identification. Other developers adopt a 'scatter gun' approach - patterns are selected in order to maximise the likelihood of detecting minor variants. The selection of longer patterns imposes certain limitations on the packages which employ the latter tactic - not least is the need to include more search patterns in the program's search database than were previously considered necessary.

Another limiting factor of long search patterns is a consequent drop in the chances of detecting a new variant of a previously known virus. Primarily, this is because it might not be possible to isolate a long search pattern which does not contain absolute memory references. This problem can be partially resolved by using wildcards in place of any addresses or constants which might change.

Virus Prevalence Tables

The distinction between 'lab' viruses and those found in the wild has been apparent for some time now, but until recently there has been a dearth of information about the prevalence of different viruses.

The following tables were produced from statistics collated by *Virus Bulletin*. Table 1 shows recorded virus infections during the period January 1st to October 31st 1991. Unfortunately, this information is incomplete, as careful recording of *every* virus incident was only implemented after the *New Scotland Yard Computer Virus Strategy Group* initiative in March of this year. Even following this initiative, incidents have gone unrecorded in the 'fog of war' - for instance the Spanish Telecom virus is *much* more widespread in the United Kingdom than table 1 indicates. This information in table 1 is also confused by the inclusion of non-UK reports which somewhat diminishes its regional accuracy.

Table 2 provides more accurate and up-to-date data about virus prevalence in the United Kingdom. This shows incidents reported to *VB* during October 1991 and includes *all* verified reports of virus infection in the UK.

Virus Prevalence Table 1

This table shows the ten most prevalent viruses reported to *Virus Bulletin* between January and October 1991.

Virus Name	Reports	Total Infections (%)
New Zealand 2	68	25.66
Form	23	8.68
Cascade	19	7.17
Tequila	18	6.79
Joshi	14	5.28
Dark Avenger	14	5.28
Jerusalem	13	4.91
4K	12	4.53
Spanish Telecom	11	4.15
Nomenklatura	8	3.02
Yankee	8	3.02
Other	57	21.5
Total	265	100

A table will feature in each future edition of *VB* showing the 'top ten' viruses in the UK during the preceding month.

As Mark Twain said 'there are lies, damned lies and statistics' - the figures shown in these tables do not accurately portray the *full* extent of the problem in the UK as they do not include statistics from the myriad of other agencies involved in combating computer viruses.

Virus Prevalence Table

The following table is a break down of virus infections in the UK reported to *Virus Bulletin* during October 1991.

Virus Name	Reports	Total Infections (%)
Form	9	22.5
New Zealand 2	8	20
Joshi	5	12.5
Tequila	4	10
Spanish Telecom	4	10
Michaelangelo	3	7.5
Cascade	2	5
4K	2	5
Nomenklatura	1	2.5
Jerusalem	1	2.5
Flip	1	2.5
Total	40	100

KNOWN IBM PC VIRUSES (UPDATE)

Updates and amendments to the *Virus Bulletin Table of Known IBM PC Viruses* as of 20th November 1991. Hexadecimal patterns may be used to detect the presence of a virus with a disk utility program, or preferably a dedicated virus scanner.

Type Codes

C = COM files **E** = EXE files **D** = Infects DOS Boot Sector (logical sector 0 on disk)
M = Infects Master Boot Sector (Track 0, Head 0, Sector 1) **N** = Not memory-resident after infection
R = Memory-resident after infection **P** = Companion virus **L** = Link virus

Big Joke - CN: A harmless 1068 byte virus, which contains a long message, warning about a harmful variant soon to come.

```
Big Joke        558B EFCD 105D B9FF FF49 83F9 0075 FA46 59E2 DFE8 AA00 59E2
```

Blinker - CR: A 512 byte variant of Backtime (q.v.) and detected by the pattern for that virus. This also applies to a 496 byte variant which was made available as 'Joker'.

Brainy - CR: A 1531 byte virus of Bulgarian origin, which appears to do nothing but replicate, but is interesting from a technical point of view because it may insert itself into the middle of another program, without modifying the program's starting instructions. Brainy uses a simple 'byte-swap' encryption.

```
Brainy          1B90 8BEC 0E1F BC34 00FC AD86 C489 44FE 4444 81FC 0003 F272
```

Cascade-1701-S - CR: A minor modification of the Cascade virus, with the encryption routine changed, probably to bypass some scanner. Reported to be written in Sweden.

```
1701-S          FA8B ECE8 0000 5B81 EB31 01F6 872A 0101 740F 8DB7 4D01 BC82
```

CB 1530 - CER: This 1530 byte virus is detected by the previously published 'Dark Avenger' pattern.

CSL, Microelephant - CR: A 381 byte virus from Eastern Europe, which contains the text '26.07.91.Pre-released Microelephant by CSL'. This virus does nothing but replicate.

```
CSL             A184 008B C8B8 9200 BB84 0089 07A1 8600 8BD0 8CC0 8947 028E
```

Day/10 - CN: This 674 byte virus was made available to virus researchers under the name of 'Numlock', but that is just the name of the original sample. The effects of the virus have nothing to do with the NumLock key - instead it will overwrite the first 80 sectors on the hard disk which only happens if the date of the month is divisible by 10.

```
Day/10          8E06 2C00 B900 10FC 33FF B050 F2AE 7518 B641 2638 3575 F347
```

DIR-II(1) - LCER: Two new variants of this virus, which was described last month, have now appeared. The following pattern will detect all three variants.

```
DIR-II(1)       26FF 77FE 26C5 1F8B 4015 3D70 0075 1091 C640 18FF 8B78 13C7
```

F-709 - CR: This 709 byte virus is reported to have originated either in Sweden or in Finland. It has not been fully analysed, but appears to do nothing but replicate.

```
F-709           8BF2 33FF F3A5 068C C633 C08E C026 A184 0026 8B0E 8600 0726
```

Gotcha-C - CER: A 906 byte variant of the Gotcha virus. Awaiting analysis.

```
Gotcha-C        9C3D DADA 7458 5251 5350 5657 1E06 3D00 6C74 4280 FC56 7426
```

Haifa - CER: A variable-length, self-modifying, encrypted virus from Israel. No search string is possible. Currently being analysed.

Hary Anto - CR: A 981 byte virus. Reported 'in the wild' in the UK. Currently being analysed.

```
Hary Anto       B904 00D3 E8BB 3E01 8907 40B9 0400 D3E0 505A 33C9 B800 428B
```

Hey You-928 - CER: Unlike an earlier 923 byte sample, this version replicates. Awaiting analysis.

```
Hey You         2181 F9C7 0772 1C80 FE02 7217 80FA 1972 1233 C08E C026 F606
```

Jabberwocky - CER: A 812 byte virus, containing the text 'BEWARE THE JABBRWOCK'. Awaiting analysis.

```
Jabberwocky     0500 108E C0BE 0000 BF00 00B9 FFFF F3A4 1E07 89D6 BF00 01B9
```

Jerusalem-Nemesis - CER: A minor mutation of the original virus. Detected by previously published Jerusalem-USA pattern.

Lozinsky-1018 - CER: A close relative of the 1023 byte variant previously reported.

```
Lozinsky-1018   E800 005E 2E8A 44FC BF20 0003 FEB9 CB03 2E30 0547 E2FA B8DD
```

Karin, Redstar - CN: This German virus adds either 1090 or 1134 bytes to the programs it infects. It is harmless, but will activate on October 23rd when it displays the message 'Karin hat GEBURTSTAG' (It's Karin's Birthday).

```
Karin           BB00 0153 F3A4 BE00 F8BF 8000 B980 00F3 A433 FF33 F633 C033
```

Kuku - CN: This 448 byte virus may infect files in an ordinary way, or overwrite them with a small program, which will display the word 'Kuku!' on the screen when it is run.

```
Kuku            241F 3C0A 750C B42C CD21 80E6 0775 E3BD 0100 A11A FA3D C501
```

Little Brother - P: A 299 byte 'companion' virus, which appears to be incomplete.

```
Little Brother  7418 5253 501E 063D 004B 7503 E810 0007 1F58 5B5A 9D2E FF2E
```

Maltese Amoeba, Irish, Grain of Sand - CER: A destructive virus which overwrites the first four sectors of tracks 0 to 29 of the hard disk, and any diskette in the disk drive, if the date is November 1st or March 15th (any year). A psychedelic screen effect follows. When the machine is powered up a fragment of a poem (*Auguries of Innocence*) by William Blake (1757-1827) appears on screen and the machine hangs. The virus employs self-modifying encryption. No search pattern is possible. (*VB*, Dec 91)

Minimal-30-B: This is in most respects the same virus as the Minimal-30 reported earlier, but it has been assembled with a different assembler, which has produced minor differences. At only 30 bytes this is currently the smallest known virus.

```
Minimal-30-B    3DBA 9E00 CD21 93B4 4089 F28B CECD 21C3
```

Mono-1063 - CR: A 1063 byte Polish virus, which deletes files when it activates if it is running on a PC with monochrome display.

```
Mono            FDF3 A406 E800 0059 83C1 0651 CB2E 8C4F 048D 4FF6 F3A4 2E8C
```

MPS-OPC - CN: Three Polish viruses, 469, 640 and 654 bytes long. Awaiting analysis.

```
MPS 1.1         B447 CD21 5E8B FE81 C72D 0232 C0B9 4000 F2AE 4FC7 055C 00B9
MPS 3.1         0ADB 7441 B42C CD21 3ADA 7304 2AD3 EBF8 8ADA BA80 0203 D6B9
MPS 3.2         0ADB 7441 B42C CD21 3ADA 7304 2AD3 EBF8 8ADA BA8E 0203 D6B9
```

MSTU - CEN: This virus contains the text 'This program was written in MSTU,1990' Awaiting analysis. Virus length is 532 bytes.

```
MSTU            BB16 0026 8B07 3DEB 55C3 5E8B C6B1 04D3 E805 6400 0E5B 03C3
```

Pixel-897, **899A**, **899B**, **905** - CN: Four variants, which are all detected by the Pixel-936 pattern. Contains code to format track 1.

Plovdiv-1.3 - CR: This 1000 byte virus is related to the 800 byte Plovdiv virus reported last month. According to a text string inside the virus, it should be named 'Damage', but this name was rejected to avoid confusion with the Diamond/V1024-derived Damage virus. The virus is 'semi-stealth', hiding increases in file length when it is active.

```
Plovdiv 1.3     80E2 1F80 FA1E 7506 2681 6F1D E803 079D 5A5B EB02 CD32 CA02
```

Ps!ko - CER: A 1803 byte variant of the Eddie (Dark Avenger) virus, and detected by the same pattern as the original.

QMU-1513 - CR: This virus has not been fully analysed yet, but it appears to contain an entire boot sector.

```
QMU-1513        5053 8BDA B000 4338 0775 FBB8 4F4D 3947 FE74 04F9 EB02 90F8
```

Seventh son - CN: A 332 byte virus which contains the text 'Seventh son of a seventh son'. It seems to do nothing but replicate.

```
Seventh son     1F5A B824 25CD 215A B801 33CD 210E 0E1F 07B8 0001 50C3 FCB8
```

Shaker - CR: A variant of Backtime, similar to Blinker, and probably written by the same author. Produces a 'shaky' screen when an infected program is run. Detected by the Backtime pattern (q.v.).

Simulation - CN: This is a variable length, self-modifying encrypted virus, which adds around 1300 bytes to the files it infects. When it activates it displays a message announcing the infection or an effect or message which is normally associated with a different virus, such as April 1st (Suriv 1), Frodo, Datacrime or Devil's Dance. No effective search pattern is possible.

Socha - CR: This 753 byte virus contains code which will only activate if the year is set to 1981. Awaiting analysis.

```
Socha           C0BF F5FF 268B 0547 4726 3305 4747 2633 0547 4726 3305 8D36
```

South African-623 - CN: This variant of the South African virus was discovered in New Zealand. It will activate on any Friday the 13th, just like the original, and is detected by the same pattern.

Spanz - CN: A 639 byte virus, which does not seem to do anything but replicate. It contains the text 'INFECTED! * SPANZ *'

```
Spanz           807D 043D 7506 83C7 051F EB0F B9FF 7F33 C0F2 AE80 3D00 75DB
```

Squeaker - CER: A 1091 byte virus awaiting analysis.

```
Squeaker        80FC 7F75 03B4 80CF 80FC 4B74 052E FF2E 2C00 5053 521E 06E8
```

StinkFoot - CN: This virus from South Africa uses instructions which do not exist on 8088/8086 and it will crash on such machines. It adds 259 bytes to the beginning of files, and 995 bytes at the end.

```
StinkFoot       600E 59BA 0400 B435 B024 CD21 061F 890F 8957 0261 071F C31E
```

SVC 6.0 - MCER: A 4644 byte version of the SVC virus. This is a complex multipartite 'stealth' virus. (*VB* Dec 91)

```
SVC 6.0          8ED7 8BE3 FB06 5633 D2B4 84CD 215E 5681 FA90 1975 0A2E 3ABC
```

Tony - CN: This 200 byte Bulgarian virus will only infect files with a name starting with 'B' on the first day of any month. On the second day it will only infects files with a name beginning in 'C' and so on. The virus uses some curious undocumented features.

```
Tony             CC8C C880 C410 8EC0 BE00 0133 FF8B CEF3 A4BA 0001 B41A CCB4
```

V472 - CR: A 472 byte virus, probably from Eastern Europe, which does nothing but replicate.

```
V472             01D6 31DB 8EC3 BB84 0026 8B0F 890C 4646 4343 268B 0F89 0CBE
```

Vienna-656 - CN: A non-remarkable 656 byte variant.

```
Vienna-656       895C 018C 4403 07BA 6000 01F2 B41A CD21 0656 8E06 2C00 BF00
```

Vienna Dr. Q - CN: A 1161 byte variant, which includes encryption of the data area. Not yet analysed.

```
Vienna Dr.Q      8E06 2C00 BF00 005E 5683 C61A ACB9 0080 F2AE B904 00AC AE75
```

Violator-B - CN: This 716 byte variant is detected by the Violator pattern.

Violator-B3 - CN: An 843 byte virus, related to Violator and Christmas Violator and probably written by the same author(s).

```
Violator-B3      803E D003 0274 0B80 3ED0 0303 7407 C3CD 21C3 CD13 C3CD 26C3
```

Virdem-1542 - CN: A longer variant of the Virdem virus, but detected by the same pattern as the original.

W13-REQ! - CN: This 494 byte member of the W13 group contains the text 'REQ ! Ltd (c) 18:41:22 3-I-1991'. It is of Polish origin.

```
REQ              8B4F 1683 E11E 83F9 1E74 EC81 7F1A 00FA 77E5 817F 1A10 0272
```

Reported Only

408 - CR: Does nothing but replicate.

1661 - CR: 1661 bytes.

1840 - CER: Adds 1838-1891 bytes to infected files. Contains the text 'NV71.EXE'

Cannabis: A Dutch boot sector virus.

Caz - CER: 1204 bytes.

Dutch Tiny - CER: A series of small viruses from The Netherlands.

Got-You - EN: Adds 3052-3067 bytes to the programs it infects. Will only replicate the first half of the year, but activates in the second half, interfering with certain operations, such as printing over a network.

Grapje - CN: A 1039 byte virus from the Netherlands.

Hitchcock - CR: 1247 bytes. Plays music shortly after being installed.

Lowercase - CN: 864 byte virus, which attempts to change 'IBM' to 'ibm'. Probably the same virus as '864' reported last month.

Manta - CN: 1077 bytes. Based on the VCS virus.

Miky - CER: A 2350 byte virus from Bolivia.

Mini-97 - CN: The smallest known non-overwriting virus at only 97 bytes. It reportedly originated in The Netherlands.

Newcom - CER: 3045-3060 bytes.

Pathhunt - CEN: 1231 bytes.

Pirate - CN: A 609 byte overwriting virus from Portugal.

Poem - CR: Adds 1825-1888 bytes to infected files, which activates on December 21st, displays a poem and overwrites the beginning of the hard disk. Originated in South Africa.

Pregnant - CR: A 1199 virus which may rename files to 'PREGNANT'.

Relzfu - CN: Alias for the Fake-VirX virus reported last month.

Tokyo - ER: A 1258-1273 byte virus from Japan.

Topo - ER: 1542-1552 bytes. Reported to be related to the Mosquito virus.

LETTERS

Dirty Rotten Scoundrels...

18/10/91

TO BE PUBLISHED IN FULL OR NOT AT ALL.

re: Software Reviews

Before I wrote my note about your software reviews (September *VB*, p. 12) I carefully read the article, and also the 'Protocol' in the April issue. I did not notice the hardware specifications on page 8, but this is not surprising, as the protocol does not refer to them, and you apparently do not regard them as part of it, as you always refer to 'published in *VB*, April 1991, pp.6-7'. In your comment on my letter you waxed indignant (always the first line of defence of the scoundrel) and then quoted 'all comparative tests should be done on the same machine with exactly the same file configuration'. However this statement contains the following gaping holes;

i. It is not clear if this applies to all tests at any time, or only to a given series of tests.

ii. The protocol itself does not specify which machine should be used.

iii. The word 'should' implies a degree of flexibility; provided the reviewer does not live in Yorkshire, or provided the test was not done on a Sunday or what?

iv. The review in question considered a single program, so presumably it was not a comparative test, so this clause did not apply?

As *Virus Bulletin* is owned by an anti-viral software company its reviews will always be regarded with some suspicion. It can never establish a reputation for independence unless it convinces users that all products are treated in exactly the same way. To do this it is essential that you set up ironclad specification which has no terms like 'should' or 'at least', and which specifies that all tests are done on a specific machine, and a specific set of disks and that the arrangement of files on these are fully described. As an engineer I understand the word 'specification'. I am not too sure what a 'Protocol' is, but I have noticed that the word is used mainly by diplomats, and of course diplomacy is the art of appearing to co-operate with the enemy without committing yourself in any way.

With Best Wishes,

Roger Riordan
Cybec Pty, Australia

Editor's reply:

You raise several points which I shall deal with in turn:

Your concern with the actual hardware used is laudable but irrelevant. The important point is stated in the Protocol (*VB*, April 1991, p.6, section *Hardware*) and restated and emphasised in my published reply to your note: when conducting comparative tests of any software it is self-evident that 'All comparative reviews should be conducted on the same machine with exactly the same file configuration.' Whether the test machine is an early XT or the latest 486 does not matter as long as all products under a comparative test are run on it. Referring to your itemised points:

i. The testing protocol gives instructions to the evaluator for carrying out a single comparative review (see also iv., below).

ii. As noted above, the Protocol does not state the particular machine used since it already states 'on the same machine'. Even so, the hardware and file configuration used is *always* stated or referred to in actual reviews.

iii. You feel that the word 'should' implies 'a degree of flexibility'. I refer you to the passage on correct and idiomatic usage of the words *will, would, shall* and *should* in *Fowler's Modern English Usage.* Briefly this states that the words *shall* and *should* are used to express 'an influence that affects the result, as a ***demand*** does, but a hope or fear doesn't'.

iv. A comparative review is one which compares two or more products. A review of a single product clearly does not.

The common ownership of *Virus Bulletin* and *Sophos* is well known and no attempt is made to conceal it. This connection has never interfered with *VB's* impartiality; if you choose to regard *VB* reviews with suspicion, that is your prerogative. As editor, I don't *write* the reviews - I merely *commission* them. Using your own reasoning, any comments *you* make concerning the accuracy and fairness of *VB's* reviews must necessarily be taken as a single view from someone with a vested interest in selling his *own* anti-virus software (*VET Anti-Virus, VB,* May 1991). I notice that the few letters of vilification which I have received over the last thirty months have come exclusively from such vested interests.

Finally, your implied reference to me as a 'scoundrel' reminded me of the Earl of Sandwich's similar bluster against Mr. John Wilkes more than 180 years ago:

> Sandwich: *By God, sir, I do not know whether you will die upon the gallows, or of the pox!*
>
> Wilkes: *That will depend, my Lord, upon whether I embrace your principles or your mistress.*

[This correspondence is now closed.]

Fair Dinkum...

October 26 1991

Dear Ed,

G'day. How ya goin' mate?

On behalf of the whole team of men (an women) in cork-brimmed hats from down-under, we would like to extend our thanks to Mark and your team for your most positive product review. Your review of *Virus Buster* is valued and can only inspire us to exceed our already high standards. What more motivation do we need, but to become what a cold *XXXX* tinnie is to a sheep shearer. (We have one on our staff!)

Future updates our assured. We're confident that we will meet and exceed the high aspirations you have for our product. On this note, I hope you and your colleagues "...'ave-a-good-weekend" and we'll be thinking of you when we get together for a barbie and crack a cold tinnie or two tonight.

Regards,

Elizabeth M. Gunn
Sales Manager
(from the whole team at *Leprechaun Software*), Australia

Czech IT!

Dear Sir,

I read with great pleasure Jim Bates' analysis of the DIR-II virus in the November issue of your magazine. It was (as usual) a very interesting article but I found one sentence with which I cannot agree:

> *'Any computer technology from these countries (i.e. Eastern bloc) carries a high degree of risk. If the authorities let them play like this with software, who is to say some of them haven't introduced malicious or mischievous code into some machine ROM chips, or even safety critical software.'*

I think this statement is totally wrong. I don't know of the situation throughout the Eastern bloc so I will confine myself to Czechoslovakia.

Software is protected by law in Czechoslovakia and the authors of malicious software can be prosecuted here. As far as I can establish, the number of viruses created in Czechoslovakia is very low, I am not sure if any known virus was actually written here.

Virus spread in Czechoslovakia is very limited compared with neighbouring countries both in the East and West. If our customers report new viruses these come from foreign sources, mostly from Taiwan or West Germany. Taiwan is a typical virus producer (together with the USSR and Bulgaria). It also produces a lot of widely used computer technology, but we rarely hear of the risks from this source.

For the reasons outlined above I think such general statements simply cannot be right and could lead to misunderstandings.

Yours Sincerely,

Pavel Baudis
ALWIL Software, Czechoslovakia

ON FIDONET

The following message from Mark Washburn (developer of the 1260 virus and the V2Pn series) appeared recently on *Fidonet*. His reference to associates and subscribers of the *Virus Bulletin* as 'criminals' (which, incidentally, is libellous) is ironic - if any of his virus code results in an official complaint to UK authorities he may well find himself the subject of an extradition warrant and facing a five-year term of imprisonment!

Mr. Washburn's self-appointed role as developer and distributor of potentially extremely dangerous virus code and his parallel involvement in the commercial development of an anti-virus program is seen by many as breathtaking duplicity.

VB's writers have in the past dismissed Mr. Washburn's virus programs as infantile, attention-seeking stunts of no research value whatever; this may account for the histrionics at the end of his message.

```
From: MARK WASHBURN
To: ALL
Subj: VIRUS RESEARCH (V2P7)

V2P7 will only be shipped to virus researchers.
An execution of a nondisclosure agreement will be
required. Each demonstration version of the V2P7
type virus will be uniquely serialized.
Send written request with statement of purpose to:

V2P7
c/o Mark A. Washburn
4656 Polk Street N.E.
Columbia Heights, MN 55421

Any associate of or subscriber to the Virus
Bulletin (published in England) is specifically
excluded from this offer. Under the guise of a
research publication, this unworthy tabloid
propagates virus sensationalism and ONLY APPEARS
to offer 'inside information' (for those who can
afford their over-priced 'trash'). I consider
this publication completely BOGUS. THESE
CRIMINALS SHOULD BE IMPRISONED. MAW
```

STRATEGY & TACTICS

JGA Norman
SGS-THOMSON Microelectronics

Practical Virus Avoidance

Any company with personal computers is vulnerable to virus attack. When a company has over 3,000 PCs and 5,000 users scattered around the world in 70 sites that possibility of attack becomes a certainty unless measures are taken to reduce it. *SGS-THOMSON* has put in place a methodical risk-reduction approach which has (so far) proved effective.

1 PROBLEMS

This was the situation which faced us in the summer of 1990:

1.1 Lack of Knowledge

Most of our 5,000 PC users were not, and did not want to be, software experts. Their only knowledge of viruses was formed by sensational newspaper articles and they had no idea of the logical cause and effect relationships which govern virus infection. The small number of PC users who were software experts were, in some ways, even more dangerous than the non-experts because they accumulate lots of exotic software and are therefore more likely to pick up a virus.

We had office automation (OA) experts supporting each site. They understood PC software but knew nothing of how a PC virus functions or how to get rid of an infection.

1.2 Exposure to Infection

PCs were exposed to infection from a number of sources, of which the most important seemed to be the use of unofficial software (games, bulletin board software, personal toolkits).

Another significant source of infection was diskettes given to people inside the company by people outside.

The third common source was home PCs. Many enthusiastic and hard-working members of staff liked to take work home with them on a diskette and continue with it in the evening using their home PC. Unfortunately, other members of the family might also use the PC to play games, access bulletin boards and engage in other high-risk activities. We did not want to discourage people from taking work home, but we certainly did not want them infecting company PCs.

1.3 Detection, Identification, Disinfection

A number of virus attacks had been discovered already by noticing the side-effects and bugs present in various viruses (easy to see in many of the early specimens). This was far from satisfactory! It meant that the virus had time to replicate and infect backups and other machines which resulted in significant data loss on a number of occasions.

We needed to be able to detect a virus as soon as it appeared, to identify it accurately and predict its effects, and to be able to remove the virus with minimum effort and loss of data.

1.4 Software and Data Transfer

People frequently transfer data files (and occasionally program files) to each other by handing over a diskette or by sending files via electronic mail (e-mail). This was obviously a perfect way to spread an infection - especially with e-mail, where a single infected program could be sent to many recipients all over the world.

The company makes heavy use of communications, so there was a real risk of a worldwide epidemic following a local infection.

1.5 Protection of Customers

We work hard to build an image of quality and reliability. Infecting a customer with a virus would undermine that image, so protecting customers was seen as important to our success as a business. A customer could potentially become infected either through buying one of our software products or by receiving a diskette from one of our sales or support staff.

2. ACTION

To counter the threat, we put in place a five-point action plan, covering education, detection, backup, audit and QA, with formal corporate procedures to give it force.

2.1 Education

Education was the first priority, since by educating our users we could simultaneously make them more careful and commission their help in looking out for viruses.

Sophos Ltd had just released a video [1] which seemed to meet our needs, so we bought a copy for each main site and based our internal anti-virus procedures on those recommended in the video. English is the official language of *SGS-THOMSON* but many ordinary users in our overseas offices have only a limited command of English, so we bought versions in French and Italian for the sites where these languages predominate.

The video and supporting booklet gave a good introduction for both users and Office Automation support people. For more detailed information, we recommended that each site should subscribe to *Virus Bulletin*.

Buying the videos was only the first step. Getting all users to attend meetings to see and discuss the video was a major

effort and a major expenditure in time (5,000 users multiplied by 1 hour totals approximately 2.5 man-years). Some sites took a lot of persuading, but 9 months into the program we had reached over 3,000 of our 5,000 users.

2.2 Detection and Identification

To detect and identify viruses we needed reliable tools.

As a starting point we bought at least one copy of a virus-detection software package for each site, which we built into a 'Quarantine PC' used only for virus checking of incoming diskettes and for copying diskettes to be sent out to customers.

Obtaining 'spare' PCs for quarantine purposes was not easy, and at some sites it was decided to conduct virus checking on a PC used for other work although it is recognised that this increases the risk.

In addition we had some additional copies of the virus-specific pattern-recognition component of the software on write-protected diskettes, to use in checking other PCs.

Our virus-specific pattern-recognition software and *Virus Bulletin* are complementary, because they use the same names for viruses. So if the virus-specific detection software finds a virus, we can look up details of the virus in back copies of *Virus Bulletin* and work out how best to tackle the infection. This knowledge of the characteristics of viruses we meet is very important.

An important consideration in the selection of our virus-detection software was that its virus recognition patterns are updated monthly to keep up with new viruses, so we needed to make sure that every copy in the company is updated regularly. To do this, the supplier sends one copy of the new version to our corporate OA support site in Milan, who send it out via e-mail to the local OA person responsible for each site. The local OA person is then responsible for updating every copy on the site. The new version of the software is itself scanned for viruses at each stage using the previous version.

2.3 Backup

The first rule for recovery from virus attacks is to have a backup copy of your data and programs, so that you can restore anything which a virus (or anything else) may destroy.

This is not easy to enforce among a widespread user community, so the main thrust is educating users about the need to take backups (via the *Sophos* video). Our OA staff do not have time to back up users' data for them, but they do hold a central copy of each software package in use, so that any corrupted software can be restored.

In practice, those PCs which handle critical company data are backed up regularly. Many of the others are backed up only occasionally.

2.4 PC Audit

A virus check using virus-specific pattern-recognition software is included in the regular PC audit. This checks not only the PC itself, but also all the backup diskettes associated with that PC.

About half our PCs are audited in a year so this provides a check on each PC every second year. We aim in future to audit every PC every year, resources permitting.

The main objective of this virus check during the audit is to ensure that viruses do not lurk undetected in backups or little-used programs, but we have found that our audit records are also helpful in tracing the source of a virus infection. Knowing that PC 'X' was clean on a certain date helps to eliminate suspected routes of infection.

2.5 Virus QA Checks on Software Products

SGS-THOMSON is primarily a semiconductor manufacturer, but we develop and sell some software products which support our semiconductor products. We also distribute data diskettes to customers and agents giving information on our products.

A policy decision was taken that infection of any customer with a virus would be totally unacceptable. To enforce this decision, virus checking was included in the formal software QA procedure.

One problem with software products is that they are often sold in compressed form (using *PKZIP* or similar) to save space on the distribution media. When a program is compressed, any virus code in the program is compressed too, and becomes invisible to virus scanners.

To overcome this problem:

1. The master copy to be used for production is unpacked onto the quarantine PC and checked for viruses in its unpacked form, using virus-specific pattern-recognition software.

2. The compressed copy is 'fingerprinted' (by a cryptographic checksummer) so that even a 1-bit change in the software can be detected.

3. When the master copy is used to produce a batch for distribution, a random production sample is withdrawn for inspection and its fingerprint is checked as part of the standard QA check.

2.6 Procedures

In addition to the education and the technical measures already mentioned, we defined corporate procedures for the use of PCs which have the force of law inside the company. Essentially they are a set of rules for minimising the risk of virus damage.

The main principles are:

- ➤ Do not take chances.
- ➤ If you pass on software or data, make sure it is not infected.
- ➤ If you receive some software or data, protect yourself by checking it.

Some of the more detailed points include:

- ➤ Games must not be run on any company computer.
- ➤ Any software other than the standard company tools must be checked and approved by the site security officer before it can be installed. This always includes a check for viruses.
- ➤ Every diskette from outside the company must be checked with virus-specific pattern-recognition software before it is used on a company PC. This includes diskettes containing only data as well as those containing programs, and it includes diskettes which have been used on a home PC.
- ➤ Every diskette going outside the company must be checked with virus-specific pattern-recognition software, to ensure that it is virus-free.
- ➤ Following the initial integrity check, diskettes transferred within a site should not need checking if 100% of incoming diskettes are checked, but extra checks are advisable.

The PC user is responsible for:

- ➤ Backups.
- ➤ Checking incoming and outgoing diskettes using the quarantine PC.
- ➤ Checking executable software sent or received via e-mail.
- ➤ Reporting any indication of a possible virus infection.

The IT security officer of each site (often the DP manager) is responsible for:

- ➤ The education program.
- ➤ Provision of virus-checking facilities.
- ➤ Immediate notification of the corporate IT security manager if a virus is found or suspected.
- ➤ Virus removal operations.

3. RESULTS SO FAR

The virus-detection software and the training video were distributed to sites in September 1990. Some existing infections were found and removed. For example, out of 4,000 hard disks and diskettes in one site, 73 (2%) were found to be infected. In another (smaller) site, 50% of diskettes were infected.

Incidents continued at the rate of 1-2 per month across the whole company, but so far as we can tell all were found before they could spread. There is some indication that the attack rate is slowing as the education process permeates through the company. This is valuable, because even a single infected PC requires a major clean-up operation encompassing all PCs and data that the infected PC might have been associated with. This inconveniences users and is very time-consuming for OA staff.

Data was lost in two attacks before we implemented the new procedures, but no data has been lost since.

3.1 Sources of Infection

Sources identified (or suspected) include:

- ➤ Infected software supplied with a new PC (2 incidents).
- ➤ Utility software brought in by a new employee for his personal use. (2 incidents)
- ➤ New version of application software from a supplier (this was industrial control software, not a mainstream product) (2 incidents).
- ➤ Local technical college where a member of staff was taking an evening course.
- ➤ Bulletin Board System software.
- ➤ Local supplier of formatted diskettes.
- ➤ Old diskette, which contained a boot sector virus, which was re-used to copy software.
- ➤ Data diskette sent in by a customer.

There were also a number of incidents where we were unable to identify the source of the virus.

Viruses identified:

New Zealand (8 times)
Cascade (3)
Jerusalem (2)
Vienna (2)
Italian (1)
Beijing (1)
Keypress (1)

4. FUTURE DEVELOPMENTS

4.1 Assertions

First, the threat from MS-DOS viruses seems likely to increase further, because the number of DOS viruses continues to grow. The number of distinct viruses reported in *Virus*

Bulletin is growing at a consistent rate; if we project the figures forward (always a risky business) they indicate approximately 1700 known viruses by November 1992! Even if virus-specific pattern-recognition software keeps up with the new viruses, the time taken to scan must inevitably get longer and longer.

Second, users are becoming harder to protect. They are increasingly using laptop PCs and public communication networks, so it is becoming more difficult to restrict them from likely sources of infection.

Third, the potential business impact of infections will increase, as PC applications become integrated into the flow of critical business data. Examples include treasury PCs used for transferring funds between banks, and sales PCs used for entering orders.

Fourth, the problem stems from MS-DOS. Not that it is a bad operating system, but its detailed internal structure is known to a lot of people, and it has no effective mechanism for controlling 'rogue' programs.

5. CONCLUSIONS

The measures adopted by *SGS-THOMSON* have proved reasonably effective. In a period of increasing threat they have stabilised the number of virus incidents and reduced the number of PCs affected.

The education program has been a very worthwhile foundation, because it helps the users to understand the necessity for the other measures and without active user support those other measures would be ineffective.

Our current tools and procedures will remain the basis of our protection for the next year or two at least, but they will need to evolve to meet the changing environment, for example:

- A change of emphasis from scanning using virus-specific pattern-recognition software to the use of cryptographic checksumming software.
- More copies of the anti-virus software for use by individuals who are particularly at risk.
- More quarantine PCs, to make it easier for everyone to use them.
- An automatic means of screening software sent via e-mail.

In the longer term, the company may move towards Unix as the basis for office automation, mostly for reasons of synergy, but the increasing MS-DOS virus threat will be a contributing factor.

References

[1] Video: *Viruses on Personal Computers - A Growing Threat*. (1990) *Sophos Ltd*.

CASE STUDY

It Slipped Through The Net...

On November 1st 1991 an organisation in Liverpool telephoned *VB* and enquired whether the following text was familiar within computer virus circles:

```
"To see a world in a grain of sand
And a heaven in a wild flower,
Hold infinity in the palm of your hand
And eternity in an hour.

THE VIRUS 16/3/91"
```

The poem appeared on the screen of a computer which had been switched on that morning - the computer itself had hung. From the symptoms described, it appeared that a Trojan or virus program had triggered and damaged critical areas of the hard disk. Unfortunately the victim organisation was woefully ill-prepared to conduct any diagnostics; no disk editor or clean DOS diskette was readily available to ascertain what exactly had happened.

Inadequate Tools and Information

The organisation was informed that the poem was not familiar (at least to *VB's* editor who is a self-confessed philistine) and that recovery from an overwriting or formatting routine (which was suspected) would almost certainly require the assistance of a professional data recovery service. A specimen of the virus was requested on the assumption that infected files or boot sectors (at this stage no-one knew which) might be located on diskettes found at the stricken site. Finally, as in all cases, the organisation was provided with the telephone number of the *Computer Crimes Unit* in London and advised to contact it.

Further Reports

Approximately half an hour later, Richard Jacobs of *Sophos* informed *VB* that he had taken two calls from other sites both of which described exactly the same symptoms. Again no accurate advice could be given as *Sophos* had no specific knowledge of this particular virus (or Trojan) involved.

A telephone call to Jim Bates at *Bates Associates* shed no further light on the situation - he had not received any calls but was currently working on two infected files received in the post the previous day, one of which had been supplied by a customer and contained a previously unseen virus from Malta. This file had actually been sent encrypted via e-mail to *Sophos* on October 31st for analysis but, like so many other virus specimens, simply took its place in the queue and was listed as 'awaiting disassembly'.

No viruses had been known to trigger on Halloween (the previous day), and by midnight of October 31st, the most obvious trigger date of that month thus passed without incident. There were sighs of relief all round - no-one knew that the Maltese virus 'awaiting disassembly' would trigger the next morning.

A Call For Information

By the afternoon of November 1st further reports of the virus triggering had been received. Ray Glath of *RG Software Inc*. in the United States reported a virus called Amoeba which had struck in Damascus, Syria and had issued 'Shakespearean verse' to screen. *Sophos* uploaded a call for information on the popular *CIX* virus conference during the afternoon of Friday, November 1st. Notice the errors in the message, particularly the inexact transcription of the verse and the confusion over the exact trigger data and conditions.

```
Two reports received this morning of a virus which
triggered on 1st November or 31st October.

Message displayed by virus:

"To see a world in a grain of sand
A heaven in a wild flower
Hold infinity in the palm of your hand
and Eternity in an hour."

The Virus 16/3/91

The virus leaves the hard disk motor running and
crashes PC. It trashes the partition table(?).
Sample urgently requested.

Sophos Technical Support
Tel 0235 559933
Fax 0235 559935
```

No responses to this message appeared on November 1st. However, the poem was recognised by Dr. Peter Lammer as verse by William Blake (1757-1827), as it was by Jim Bates who subsequently identified it as the first verse of *Auguries of Innocence* (erudite chaps, these virus researchers).

A Solution At Last

By 9.30 am on Saturday, November 2nd, Bates had disassembled the Maltese virus in its entirety and had located the Blake poem which is stored encrypted within its code. A second trigger date of March 15th (the Ides of March) was included in the code and the trigger effect entailed overwriting cylinder 0 through cylinder 29 on the hard drive followed by the same sequence on any diskettes found to be present. Text sections referred to an Amoeba virus, which corresponded with Glath's message of the previous day.

By 9.41 pm on November 2nd, it became clear that the team at *S&S* were also having an eventful weekend. A *CIX* message was posted that evening by Ray O'Connell of *Virus News International*. It provided a reasonably detailed analysis of the virus and announced the availability of updated detection and eradication tools made available on *CIX* to registered users of *Dr. Solomon's Anti-Virus Toolkit*.

Unfortunately, despite all the noble intentions of the anti-virus community, these efforts were simply too late...the virus had spread, it had avoided detection and now it had triggered.

Scanner Limitations

Prior to November 2nd 1991, no commercial or shareware scanner (of which *VB* has copies) detected the Maltese Amoeba virus. Tests showed that not *one* of the major commercial scanners in use (the latest releases of *Scan*, *Norton Anti-Virus*, *Vi-Spy*, *VISCAN*, *Findvirus*, *Sweep*, *Central Point Anti-Virus et al.*) detected this virus. Nor did any anti-virus software vendor or computer security agency warn of its imminent trigger date; no individual or organisation even alluded to the virus' existence prior to its triggering!

At the time of writing (November 10th), only two packages are known to detect it, both scanners (*VISCAN* and *Dr. Solomon's Anti-Virus Toolkit*) were updated over the weekend of November 2nd-3rd 1991 - a classic case of shutting the stable door after the horse had bolted.

If we are to believe the authors of the virus, it was written and presumably released on 16th March 1991 and has spread undetected since then from Malta to locations in the UK, Eire, France, Germany and Syria (at the time of writing). In simple terms, this virus spread for *seven* months entirely unnoticed.

Unknown Factors

How many more undiscovered viruses are currently in circulation? In scanning terms, what is the total number of false-negative indications among all the scanners combined at any given time? No-one knows. It would be comforting to think that virus specimens are always discovered and combated quickly but this need not necessarily be the case.

Furthermore, with the number of known viruses now heading towards 1,000 it is becoming increasingly difficult to know *which* ones contain pernicious trigger effects and *what* these trigger effects are. The collective knowledge even about those samples which have been isolated and are 'in captivity' is synonymous with the tip of an iceberg - hundreds of known viruses still remain unanalysed, while heaven knows how many trigger routines, dates and conditions remain undiscovered in this ever-swelling pile of binary.

The lessons learned? (1) *Virus-specific detection is useless in the face of unknown threats*; (2) *Both computer users and the anti-virus community were taken entirely unawares by this virus - it is not the first time and it will not be the last*; (3) *Checksumming is the inevitable long-term approach to computer virus detection*.

VIRUS ANALYSIS 1

James Beckett

Maltese Amoeba...Poetic Injustice

On Friday 1st November this year, several reports were received of computers which refused to boot; the usual power-up messages failed to appear and were replaced instead by a few lines of poetry. The symptoms were not immediately familiar and we did not know of any virus which triggered on this date, but after a couple of distress calls from unrelated organisations it became obvious that this was indeed a virus.

The event is unusual because there were no reports of the virus being found by any of the popular virus detection packages. Several of the victim organisations subscribed to well-known scanning programs and it appears that *not one of them* detected it prior to the virus' trigger data. Computer virus samples are usually in the hands of researchers and anti-virus software producers long before they appear in the wild (if ever they do), this virus was a rare exception to the rule.

Ironically, it now transpires that a sample had already been received the previous evening. An infected executable had been provided by the virus researcher Jim Bates and analysis was proceeding even as calls came in.

'No Search Pattern Is Possible'

After verifying that the sample I had was infectious, analysis was carried out on an executable designed as a virus sample carrier (sometimes referred to as a sacrificial or goat file). It was immediately apparent that: *the main body of the virus was encrypted*; *the initial decryption routine was largely composed of unnecessary instructions*; and *the decryption routine modified itself upon each infection*. This meant that once again, we were faced with a virus which could not be detected by using any simple pattern, but which had to be analysed exhaustively to determine exactly in which ways it would mutate in order to devise a detection algorithm.

The encryption itself is a trivial XOR of the remainder of the virus with a word value that varies between instances of the virus (totalling 65, 536 different encryptions). However, if the decryption routine is single-stepped, the program code itself is open to ready analysis. Although the initial few instructions might be construed as minor 'armouring' to obstruct further single-stepping, there is no serious attempt to prevent disassembly.

At this point a near-complete listing was available and the text in the code gave enough clues to establish a link between this virus and the one which had triggered in the morning. Unfortunately, this knowledge was of little to those computer users who had already been hit!

Are You There?

Viruses, in common with some respectable programs, employ 'Are-You-There' calls to determine whether another copy is already resident in memory - in order to avoid going resident themselves. This virus issues and responds to several such calls, interacting with different viruses. It initially uses a DOS Set-Date call with an invalid value to check for itself, then also checks for the PSQR virus (a Jerusalem variant) using PSQR's own Are-You-There call. If either of these are answered, control is returned to the host program and the virus does not infect. Otherwise, it goes on to examine the code at the start of the INT 21H function chain for the presence of code for processing an Are-You-There call similar to Murphy-1, and if found, removes that virus from memory by restoring the previous INT 21H handler address.

Memory-Resident Software Subversion

After all these checks, Maltese Amoeba installs itself into memory by manipulating DOS Arena pointers thus bypassing any resident programs which monitor the standard Terminate-and-Stay-Resident (TSR) functions to indicate virus activity.

Trigger Routine

After installing itself and intercepting the DOS Function despatcher (INT 21H), Amoeba checks the PC's date setting and the process to inflict deliberate damage commences. If the date is the first of November (any year), or the 15th of March (any year), the first thirty tracks of side zero on the hard disk have the first four sectors overwritten with virus code. The virus goes on to do the same with any diskettes present and if it succeeds in wreaking such destruction without being stopped it congratulates itself with a psychedelic flashing screen display, which loops indefinitely.

The code written into the Master Boot Sector during this process contains a second level of decryption to display a section of poetry - in fact the first four lines of Blake's *Auguries of Innocence*, from *The Pickering Manuscripts* (See Case Study, page 13). The verse displays the next time that the PC is booted. The PC then hangs.

[*The theme of Blake's* Songs of Innocence *is the all-pervading presence of divine love amidst trouble and sorrow. They are followed by the altogether more cynical* Songs of Experience, *which present a pessimistic, dark view of the world. Blake's work is often considered 'anti-establishment', concentrating on the denial of authority, the overthrow of order and imposed morality and the triumph of anarchy. Elements of his thought might well appeal to the virus writer. Ed.*]

Data Areas

The remainder of the boot sector contains a message in plain text lambasting the *University of Malta*, at which the author(s) apparently studied and warns that a further version of the virus is in the making. The text is the source of the name 'Amoeba'.

```
"AMOEBA virus by the Hacker Twins (C) 1991 This is
nothing, wait for the release of AMOEBA II - The
Universal infector, hidden to any eye but ours!
Dedicated to the University of Malta - the worst
educational system in the universe, and the
destroyer of 5X2 years of human life."
```

Clues To Investigation

It is always dangerous to interpret text messages in computer virus code literally but I would tentatively suggest that there are sufficient clues within this code for its authors to be identified and, if Maltese law permits, apprehended. Two students, one or both of whom are computer literate, who joined the *University of Malta* (in 1986-87?), who were probably still in attendance in March of this year (a five year degree / diploma course or possibly post-graduate education) and who empathise with a poem by William Blake (is there an English literature faculty within the University?) must surely be traceable. Considering the destructive nature of this virus and the veiled threat to release yet more viruses a preliminary investigation into these obvious leads seems warranted.

Concealment

While resident in memory, the virus intercepts INT 21H to infect further files and also answers Are-You-There calls. As well as its own recognition sequence, it responds to the PSQR and Murphy viruses, thus preventing either of them from activating. Both the DOS load-and-execute and open file functions are used to infect COM and EXE files. The intercept checks for re-entrant calls, so the virus can use the DOS open file call in its own infection process. The critical-error handler is also trapped to avoid the user seeing any of the familiar 'Abort, Retry, Ignore' messages which might appear during attempted infection of a write-protected diskette, for example.

Several checks are made on files prior to infection and steps are taken to reduce the chance of the infection process being noticed. A checksum is produced of the filename; this has two functions: 'COMMAND.COM' produces a CRC of 7478H and is not infected while infected files have the checksum appended as an infection marker to avoid multiple infections. Of course, if a file were to be renamed, the CRC could change and it could be re-infected. Files with the SYSTEM attribute are avoided and any other attributes are overridden and then reset - so read-only files are not immune to infection. Timestamps are also preserved. These latter two precautions are taken by many viruses today. Files larger than 64 Kilobytes or smaller than 450 bytes are not infected, but large COM files could still be pushed over the 64 Kilobyte limit after having been infected.

The Encryption Method

Many more viruses are now employing self-modifying encryption (the basic methodology involved in this process was widely disseminated by Mark Washburn). Self-modifying encryption produces numerous different permutations of code on a random basis in an attempt to thwart 'dumb' scanners.

The 'mutation engine' in this virus comprises three parts, starting with a template containing the actual bones of the decryption routine. This contains a number of gaps which are first filled at random with one-byte instructions, none of which affect the working of the routine: NOP, CLI, CLD, CLC, CMC, STC, SAHF. The first is an official no-operation instruction and the rest affect irrelevant processor flags. Subsequently some of these instructions are randomly swapped around, again preserving the functionality of the decryption routine.

After some processing specific to the type of executable targeted, a series of two-byte null instructions is inserted into the routine (such as exchanging registers with themselves, and jumping to the next instruction).

Finally, the virus is written to disk: an image of the virus is encrypted with a random key while the decryption routine, having had the same key coded into it, is written out, followed by the encrypted virus. The initial few bytes (either a COM file's JMP instruction, or an EXE file 'MZ' header) are modified to link the virus code in front of the host.

Detection and Removal

Detecting which files have become infected is a non-trivial process. No search pattern is possible.

A simplified approach might involve checking for certain instructions at the start of the virus, bearing in mind that a variable number of other instructions may lie between them. The remainder of the routine is made up of the non-instructions mentioned above, to a maximum total size of 99 bytes. Also, the code is generated differently for COM and EXE file infections. The rest of the virus is some 2457 bytes long, so file sizes will be increased by a variable amount. Timestamps and attributes are preserved. The virus' own infection marker checksum could be used, but this is an unreliable method.

Anti-virus software is currently being updated by a number of the major companies involved in scanner development. Two packages (*Dr. Solomon's Anti-Virus Toolkit* from *S&S Enterprises* and the *VIS Utilities* from *Bates Associates*) were updated over the weekend of November 2nd-3rd 1991. A *Sophos Sweep* update to detect this will be available this month as will a copy of Fridrik Skulason's *F-PROT*. Updates from other major manufacturers are doubtless being prepared.

[*Due to its existence in the wild and the fact that the virus will next trigger on 15th March 1992, VB will conduct tests over the next couple of editions to determine which scanners detect the Maltese Amoeba virus. Ed*.]

Disinfection is best achieved by deleting the infected files and restoring from write-protected backups or master software.

VIRUS ANALYSIS 2

Jim Bates

The SVC Series - The Latest Stealth Viruses

Just as an average pianist can recognise poor musicianship, so an average programmer can easily recognise poor code. A case in point concerns the SVC series of viruses which have recently caused something of a stir in some sections of the research community.

These viruses display an increasing complexity in successive versions which has now reached laughable proportions with the convoluted 'spaghetti code' which comprises the SVC6.0 virus. The reason for the research interest centres around the virus' stealth capabilities. Just as Mark Washburn wasted so much time and effort by single-mindedly attempting (unsuccessfully) to produce an undetectable virus - so the author of the SVC viruses has concentrated on producing a dementedly tortuous stealth virus. Stealth, in this context, refers to virus code which attempts to avoid detection by other software while it is resident in a system.

It is useful that we have a number of SVC viruses (probably comprising a development series) to examine, since the progress between versions gives an insight into the thinking of the writer. In this instance I have disassembled and analysed both SVC5.0 and SVC6.0, the additions and alterations (and the update process) that the author has made in the later version are quite revealing.

In general terms, both viruses are fairly standard parasitic viruses containing no major encryption capabilities and with respective infective lengths of 3103 and 4644 bytes. There are no deliberately destructive trigger routines but both viruses will cause program file corruption and SVC6.0 will certainly cause complete system failure on some PS/2 machines.

ANATOMY AND FUNCTIONING

Let us first examine the SVC5.0 virus:

Installation Routine

The code first issues an 'Are you there?' call to the system by placing a value of 84H into the AH register and issuing an INT 21H request call. If the virus is resident, this call returns a value of 1990H in the DX register and the version number of the virus in the BH register. An encrypted copy of the virus' resident code segment is also returned in AX but more of this later. It should also be mentioned that the virus contains an earlier 'Are you there?' response routine which returns only the 1990H in the DX register if a function 83H request is received. The virus goes resident by using the standard tactic of calculating the highest available segment in conventional memory and moving itself there (removing around 3000 bytes from available memory in the process).

The memory is given a legal Memory Control Block by the virus, and this is marked as being owned by DOS. During the installation process, the virus hooks into the DOS system interrupt (INT 21H) and the system timer interrupt (INT 08H), while maintaining a copy of the original (clean?) INT 21H vector address for its own internal use. When the installation is complete, the host program file is repaired in memory and processing is passed to it.

Virus Operation

Once hooked into the DOS Interrupt, the virus maintains a comprehensive monitoring of no less than seventeen distinct function calls in an attempt to avoid detection by both resident and non-resident virus checking and scanning software. The usual stealth feature of hiding the change in infected file length is there, along with more insidious redirection of system services. Taking them in order, the changes introduced during each function intercepted are as follows:

- Function 4B03H - Load Overlay

- Function 4B01H - Load, relocate, but do not execute

Both of these functions are treated the same - the virus checks the time field of the target file for a value of 60 or 62 seconds which it uses as an infection indicator. If the file is marked as infected, the content of the attached virus is checked to see whether it is SVC5.0 - if so, the file is disinfected (see below) and processing is allowed to continue. If the file is not infected by this particular version, then processing is allowed to continue without further interference.

- Function 4B00H - Load and Execute

Interception of this function causes the target file to be checked for infection as before. However, in this case, if the file is 'clean' it is then infected. For ease of reference, the specific infection details are given below. Special provision is included for the execution of *CHKDSK* since the various infection/disinfection/truncation routines would cause it to produce multiple allocation errors. If the virus detects that *CHKDSK* is about to be executed, an indicator is set to warn other routines to modify their operation.

- Function 4CH - Exit program

Interception here simply clears the special CHKDSK indicator (OK chaps - he's gone, everybody relax!)

- Function 3DH - Open file

Exactly similar to the Load and Execute intercept, but with an added routine which prevents the virus from infecting files resident in drive A: or B:. Quite why this happens is not clear

since no such conscience is shown by the Load and Execute infection routine. Additionally, when a file is opened its handle is stored by the virus for future reference.

An extra routine is invoked during a file open request. This checks whether the name of the file is 'AIDSTEST.C' and if so, a virus 'copyright' message is appended to the end of the file. This is not an infection process, but simply adds a single line of text. The 'copyright' message is:

```
-* (C) 1990-91 by SVC, Vers 6.0 *-
```

This self-test is presumably a left-over routine from the 'development cycle' of the virus. [*AIDSTEST is the name of the most popular anti-virus utility in Russia. Tech Ed.*]

- Function 3CH - Create a File

When this function request is received, the virus simply stores the handle of the file being created (for use later). The SVC5.0 virus can only store a single handle in this way whereas SVC6.0 maintains a table of up to 8 handles.

- Function 3EH - Close a File

If the file handle is collected from a Create request, the file will be infected - otherwise it will be ignored and the Close request will continue normally.

- Functions 11H, 12H, 4EH and 4FH - Find a File

These functions constitute the main file location system of DOS. They are classified as FINDFIRST (11H and 4EH) and FINDNEXT (12H and 4FH) and search either via FCB (File Control Block) methods or through filename searching. A similar series of interception routines is used on all of them whereby the relevant 'found' file is checked for infection and, if infected, has the length of the virus subtracted from its file length field. The virus also clears the seconds portion of the time field on infected files.

- Function 3FH - Read file

With this request, the virus first completes the request on behalf of the calling routine, and then checks to see whether the information collected contains any virus information (i.e. part of the appended virus code or the modified header). If so, the file is disinfected and the read request is re-issued.

- Function 40H - Write to file

Since only one handle can be maintained at a time by the SVC5.0 virus code, this function can conceivably cause corruption. If the target of the write request is that referred to by the virus handle, the file will be disinfected before the write is allowed. However, if two or more files have been opened simultaneously, one or more of them could write over or beyond sections of virus code. Modifications in SVC6.0 make this problem much less likely.

- Function 4202H - SEEK to End of File

This function is subverted by the virus which ensures that the file position pointer is always returned pointing to the original end of file and never to the end of the appended virus code.

- Function 5700H - Get File DATE/TIME

There has been much uninformed talk about the ubiquitous 62 second marker on files. The plain fact is that only a small proportion of viruses use this method of marking their trail - so the practice of setting this marker to avoid infection is about as effective as wearing a cardboard 'PLEASE DON'T SHOOT ME' poster while walking around downtown Beirut. In this instance, since the virus checks the file internally for a version number, the external marker is merely a convenience. However, it may also be a giveaway since there are *still* packages which will check for this marker as a possible indication of virus activity. To avoid this, the virus intercepts the Get DATE/TIME request and clears any marked seconds field to zero before returning the requested information to the calling routine.

This concludes the range of functions intercepted by the virus but it is useful to examine the actual infection and disinfection routines in a little more detail.

File Infection

Target files are COM, EXE and files containing executable code loaded via the various subfunctions of Function 4BH. Any file containing 'MM' or 'BM' in its filename is specifically excluded, thus avoiding COMMAND.COM and files like IBMBIO.COM. Files with the system attribute set are similarly excluded from infection.

Both SVC5.0 and SVC6.0 use similar processes to infect files and only the size range of suitable files and the infective length of the virus code differ. These are as follows (SVC6.0 details in brackets): Infective length is fixed at 3103 (4644) bytes and if the target file contains a standard 'MZ' header then any length file will be infected. For COM files, non-standard EXE files and overlays, the target file length must be between 3103 and 60896 (4644 and 50711) bytes inclusive before infection is attempted. The values for the infection of device driver files by SVC6.0 are different and are listed below.

The infection mechanism consists of appending the virus code to the target file and modifying the header or initial jump to ensure that the virus code is executed first. The original header information is contained within the virus code in encrypted form and the encryption key is a pseudo-random number collected from the system clock. The encryption key is also stored as a plain value within the virus code so that the disinfection routine can be invoked when required. The increase in file length is concealed from all but the most determined inspection by the stealth features of the virus.

Self-Disinfection

This is achieved by the virus opening the file and collecting the encrypted header information and the encryption key bytes. Then a toggle routine is used to unscramble the header information. The 'clean' header is then written back to the beginning of the file and the file length is truncated back to its original size by simply subtracting the virus code.

Bugs

There are several mistakes within both viruses, most of which have a negligible effect on virus operation. I am not in the business of debugging virus code for the benefit of the execrable individuals who write them, but in this case there is one bug which may assist users in locating these viruses. The same mistake exists in both viruses and its effects may vary between different machines. In some cases is that the execution and subsequent infection of certain overlay files could result in their attributes being replaced wrongly by the virus code. This could make the overlay file a *System* and or *Read Only* file and might even change it into a volume label or a hidden subdirectory. Anti-virus developers should note these possibilities when devising detection methods.

"Setting this 62 seconds marker to avoid virus infection is about as effective as wearing a cardboard 'PLEASE DON'T SHOOT ME' poster while walking around downtown Beirut."

Active Monitor Subversion

A primitive form of monitor detection is incorporated into SVC5.0 whereby regular checks (courtesy of the installed INT 08H handling routine) are made to see whether any resident monitoring software has attempted to hook into INT 01H (Single Step) or INT 03H (Breakpoint). If such an attempt is detected, the virus re-installs its own dummy handling routines in place of the monitor. If the dummy handlers detect any interference, the machine is rebooted! A more sophisticated version of this system can be found within SVC6.0

SVC6.0 - ADDITIONAL FEATURES

The foregoing constitutes a reasonably comprehensive description of the SVC5.0 virus, but SVC6.0 - as its increased length would imply - contains several extra routines which increase its virulence within a PC environment.

The first of these new 'tricks' is the introduction of a boot infection capability making SVC6.0 a multi-partite virus.

Boot Infection

Within the installation routines of SVC6.0 is a section which checks for the existence of a primary fixed disk containing a standard, active DOS partition. If found, the Master Boot Sector and Track zero of the primary drive are infected.

The original Master Boot Sector (Track 0, Head 0, Sector 1) is read into a buffer in memory and the Partition Table is checked to ensure the existence of an active DOS partition. Then the first three bytes of this buffer are collected and stored within the virus code. Three newly-generated bytes are then placed at the start of the Master Boot Sector memory image - these three bytes constitute a jump instruction to code at offset 17CH. Code from within the virus is then copied into the memory image at this offset and the modified buffer is then written back to the Master Boot Sector.

The final stage is to write the whole of the virus code to sectors 2 to 11 of Track 0, Head 0. As remarked in previous analyses, overwriting any reserved sectors on Track 0 can cause problems on some machines. In this case it is certain that machines which contain configuration and signature information on sectors within Track 0, Head 0 will not reboot correctly (if at all) after being infected by this virus. Tests on an IBM PS/2 model 80 revealed an immediate configuration error on attempting a reboot after infection with SVC6.0 and the boot process could not be completed.

Boot and BIOS Stealth

In order to provide some protection against detection of the modified boot sector, SVC6.0 has an additional interception routine hooked into the BIOS interrupt INT 13H. This is necessarily different from usual boot sector virus redirection since SVC6.0 does not maintain a copy of the original Master Boot Sector. The interception routine detects requests to read or write to the Master Boot Sector and subverts them accordingly. A read request is held while the original three bytes from the beginning of the boot sector are replaced and the additional virus code is deleted (to zeros). Then the modified sector is returned to the caller. Any request to write to the Master Boot Sector is simply aborted.

The actual operation of the boot code simply loads the virus, hooks it into INT 08H and INT 13H, repairs the original boot code and then jumps into it. Since the original infection of the Master Boot Sector does **not** check for code in the area it overwrites (this would be zeros on a standard boot sector), protection systems which depend upon placing extra code within the boot sector could well become corrupted.

While the boot infection is truly a complete infection (involving the whole of the virus code) it does *not* transfer to floppy disks. It seems simply concerned with getting the virus into

memory from the existing hard disk. Thus the only way this virus can travel between machines is via infected files on diskettes or communication channels.

Device Drivers

Another addition incorporated into SVC6.0 is a series of routines designed to enable the infection to be passed to device drivers. The file inspection routines have been modified to recognise the header sections of device driver files and infection is thereafter accomplished by appending the virus code to the file and modifying both the Strategy and Interrupt (Command) routine addresses to ensure its invocation on execution. The size limitations of the device driver files which the virus targets are different to those applied to other files. There appears to be no lower limit but device driver files greater than 62247 bytes in length are not infected.

Another series of routines works in conjunction with the boot configuration of the virus in such a way that the loading of a batch file (AUTOEXEC.BAT?) is detected and attempts are made to infect individual filenames invoked from within the batch processing. These routines make extensive use of spare video memory above B800:0FA0H which is just beyond the 80x25 text configuration of standard colour configurations. Any machines with a specific screen configuration that uses more memory than this (80x43 or 80x50) may cause system malfunction with this virus resident in memory.

Anti-Debugging Code

As mentioned above, SVC6.0 has one or two other minor 'enhancements' over its predecessor and these include the ability to keep track of more than one file handle (during the infection/disinfection cycles) and an increased sensitivity to the possibility of monitoring software.

This latter addition cannot accurately be described as 'armouring' but it can cause slight problems when attempting to analyse the operation of certain sections of the virus code under control of debugging software. The virus actually installs its own INT 01H and INT 03H handlers and these regularly check to ensure that they have sole control of these sensitive interrupt vectors. The obsession with stealth is even extended to these areas by providing a link/unlink routine to make it appear to a cursory examination that the INT 01H and INT 03H system vectors have not been tampered with.

An Unexplained CMOS Routine

One final routine, which caused a raised eyebrow, attempts to identify the host machine by checking the machine ID byte in ROM. If an AT type machine is indicated, the routine goes on to write the string - 'SVC 6.0' into CMOS memory at offset 34H. There is little standardisation over exactly what lives where within various CMOS environments but those reference books that I consulted generally concur that offset 34H is 'reserved'. Quite what the programmer hopes to achieve by writing the virus name and version into CMOS is questionable - perhaps a future recognition capability? Certainly within this code, no further reference to CMOS is made.

Conclusions

These viruses may cause some minor problems to anti-virus software developers engaged in the maintenance of memory-resident programs.

The irony, of course, is that this concerted attempt to avoid detection is immediately undermined by simply rebooting from a clean write-protected system diskette! Once the virus is neutralised it is completely exposed to detection and subsequent removal. Even when the virus is resident, the stealth capabilities will not fool any competent detection software.

Detection

The following hexadecimal search patterns will locate the virus both on disk and in memory. In all cases, it is strongly recommended that machines are cold booted from a clean write-protected DOS diskette before scanning commences.

```
SVC 5.0    2E89 84A6 0B2E 8C84 A80B C406 2000
           2E89 84A2 0B2E 8C84 A40B

SVC 6.0    2E89 8471 112E 8C84 7311 C406 2000
           2E89 846D 112E 8C84 6F11
```

The following pattern to detect the SVC 6.0 virus is found in infected Master Boot Sector (Track 0, Head 0, Sector 1)

```
SVC 6.0    BC00 96FB 0E1F 0E07 BEAD 7DBF 007C
           FCA4 A5B4 02B0 0ABB 007E
```

Disinfection

All diagnosis must be run in a clean DOS environment having booted from a secure system disk. Program files infected by the SVC viruses should simply be deleted and replaced from clean write-protected master software.

Boot sector disinfection is a more involved task. The original three bytes of the clean boot sector are stored unencrypted at offset 1ADH of the Master Boot Sector. Assuming that no corruption has occurred, the three bytes at offset 1ADH, 1AEH and 1AFH should be transferred back to offset 00H, 01H and 02H. The three-byte JMP instruction should then point into clean boot sector code as before.

If the virus has overwritten sensitive areas of the boot sector commencing at 17CH, disinfection can only be achieved by restoring from a backup of the clean boot sector or, as a last resort, a low level format. As in all cases of boot sector virus infection, it would be wise to seek professional advice before commencing disinfection.

PRODUCT REVIEW

Mark Hamilton

F-PROT

Fridrik Skulason, *VB's* Technical Editor and renowned Icelandic virus hunter, has released version 2.00 of his popular *F-PROT* anti-virus software. A diversity of small utility programs that comprised version 1 have now been incorporated into a single menu-but-not-mouse-driven program.

Apart from *F-PROT* and its documentation, the distribution disk (or ZIP file if you obtain *F-PROT* from a bulletin board) contains one other program called *VIRSTOP* which is a memory-resident, virus-specific, active monitoring program. This claims to prevent the execution of any program infected by a virus known to *F-PROT*. (Unfortunately, I was unable to test the efficacy of *VIRSTOP* as it refused to work on any of the test machines under either DR-DOS 6 or MS-DOS 5; any attempt at executing the program resulted in the PC 'freezing' requiring a power-off, power-on restart.)

F-PROT

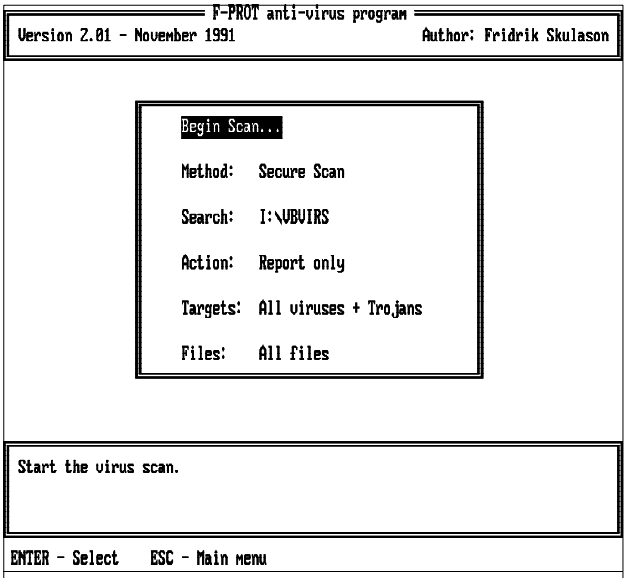
Once *F-PROT* has loaded its definitions, it proceeds by scanning memory for viruses. Memory scanning is not foolproof; it is possible to leave traces of virus code in the DOS buffer area (after copying a virus-infected file) which remain undetected - this was certainly the case with *F-PROT*. Uniquely and surprisingly, Skulason allows you to interrupt the memory scan. Why he should allow this is beyond me, because the memory scan is actually very rapid (it was completed in 15 seconds on the test hardware).

Once the memory scan has been completed (or interrupted), the program's main menu is displayed and provides six options: *Scan*; *Install*; *Viruses*; *Analyse*; *Program*; and *Quit*.

Scanning Options

When you select the Scan menu, you are given three choices of scanning method: *Full Scan*; *Quick Scan;* and *Secure Scan*. *Quick Scan* - as its name implies - is the fastest and least secure method and searches in just one place in the file. It also does not distinguish between variants of a virus so while either of the other two methods might tell you that a file is infected with 'Antiscan', the *Quick Scan* simply says that the file is infected with 'Jerusalem'.

Full Scan uses a single signature for each virus and if that matches, it will then check the file for the presence of other signatures belonging to the same virus. The *Secure Scan* mode uses all the signatures for all the viruses and is, therefore the slowest. For the timing and identification tests, I used the *Secure* and *Quick Scan* modes.



A directory is specified and a secure scan commences. In this instance the scanner is instructed simply to report the presence of any viruses. *F-PROT* can be configured to disinfect files automatically - the program contains the most extensive set of disinfection routines currently available.

Having selected the scanning method, you then select the target, or where you want to scan and you are given the choice of diskette drive, hard drive, network drive or a user-specified drive-directory combination.

The next choice (not activated under *Quick Scan*) enables you to select what action you wish to take in the event that a virus is discovered. You can choose from: *Report only*; *Query Disinfect*; *Automatic disinfection*; *Query Delete*; *Automatic Deletion*; and, *Rename*. For this review I did not test the accuracy of the disinfection routines. It should be noted, however, that *F-PROT* contains the most extensive set of disinfectors of any anti-virus package currently available.

The next option (confusingly called 'targets') enables the user to select the virus patterns and identities to be searched for. This option differentiates between: *All viruses*; *Only file viruses*; *Only boot sector viruses*; *All viruses and Trojans*; and *User-defined patterns*. Elsewhere in the program, you can enter hexadecimal patterns for new viruses (such as those published in *VB*) which are not known to *F-PROT*.

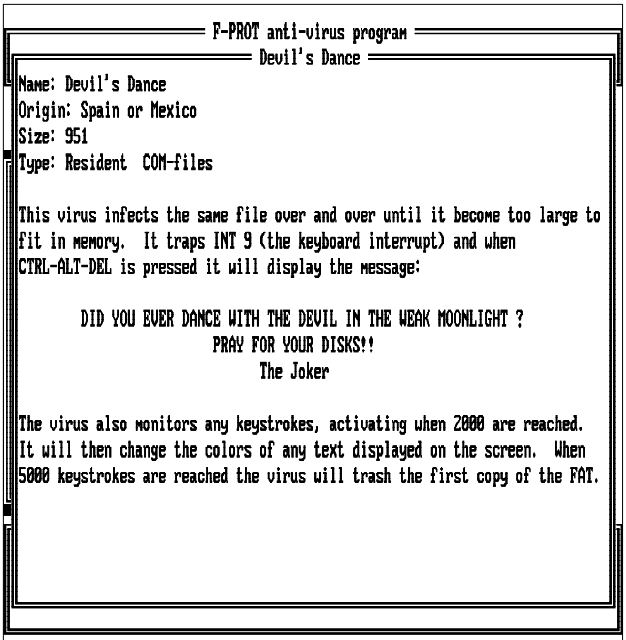
An anomaly means that recognition strings included by the user are omitted from the search routine if the user selects any option other than that to scan for user-defined stings. Two separate searches must thus be undertaken in order to run the complete library of patterns. This is surely unnecessary. The options 'All viruses' and 'All viruses and trojans' should mean exactly that; not 'all viruses except those for which you've added independent search patterns'.

Finally, you choose the objects to be scanned. You have a choice of standard executables (*.APP, *.COM, *.EXE, *.OVL and *.OVR); all files, or those with a user-defined extension. Unlike some other packages (for example *Virus Buster*, see *VB*, November 1991), you can not edit the list of standard executables to be scanned - for example to include PIF or DLL files (found in *Windows 3* and some DOS applications). However, the *user-defined* option can be configured to scan whichever extensions the user wishes.

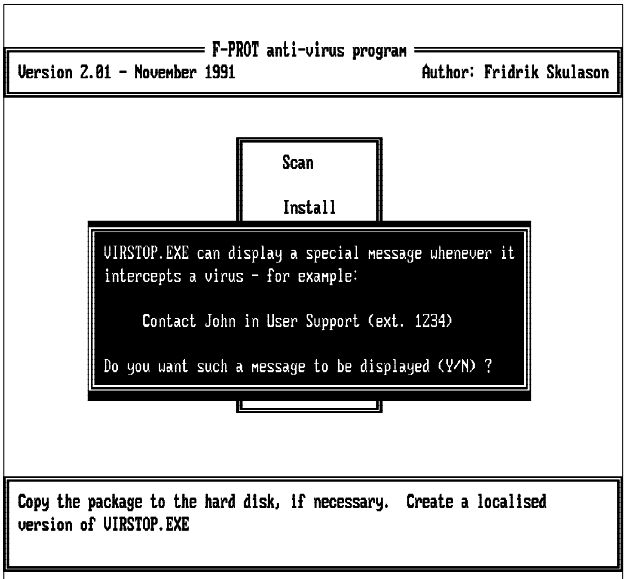
Having selected all the various options, you then begin the scan. *F-PROT* indicates the progress of the scan by displaying the full pathname of the file being scanned. It was interesting to note that *F-PROT* does not indicate infected files or the identity of the offending virus(es) until its scan has finished.

Accuracy

F-PROT is a classic example of the wide disparity between the Turbo and Secure modes of operation. In Turbo mode, it missed 14 viruses (as compared to just two viruses in its secure mode). To scan 428 executable files (by its definition) in Turbo mode on a hard disk, it took just 36 seconds while 988 files in Secure mode took nearly 12 minutes. Although it found most of the infections in the Stamina Test (see *Virus Bulletin*, October 1991, pp 7-11), it failed to detect Spanish Telecom 2 just as its predecessor in the original October tests.



F-PROT contains an extensive database containing information about most of the viruses which the package detects. This listing is shorter but more accurate than that provided by Patricia Hoffman in the United States



VIRSTOP is a memory-resident virus-specific scanner designed to prevent the execution of infected files. Alert messages can be configured by *PC Support*. Installation of this program proved impossible on the hardware used to conduct this review.

Code Analysis

Skulason is experimenting with file analysis to try to determine whether a file is infected with an unknown virus. *F-PROT* currently includes an *experimental* program which examines a suspect program's code and looks for unusual code sequences. These include undocumented DOS calls and any disk writes. Such analysis is highly likely to cause false positives and Skulason has identified and documented a number of programs prone to such false alarms including the *Windows 3 Kernel*, *Microsoft Word* and *IBMBIO*.

Against two of the more recent viruses, Spanish Telecom 1 and Tequila, this code analysis caused inconsistent results. While it did identify four COM files and one boot sector infected with Spanish Telecom 1 as suspect, it passed all five Tequila infections as clean. In the light of these results, I would not currently recommend the use of this code analysis as a serious diagnostic method. However, if the author further hones his analysis engine, this method of detection may prove more reliable.

[Skulason has pointed out that his code analysis methods are still at an early *experimental* stage and are *not* intended as a replacement for traditional scanning. In his own tests, the author claims a 90 percent detection rating against his virus collection while code analysis has proved equally effective against new, unanalysed samples. A number of public domain and shareware programs have appeared recently which incorporate static analysis methods - these detection programs will feature in a forthcoming comparative review. Ed.]

F-PROT

Version Reviewed	Version 2.01
Scanning Speeds [1]	
Test 1 (i) Hard Disk - Turbo	36 secs
Test 1(i) Hard Disk - Secure	11 mins 55 secs
Test 2 (i) Floppy Disk - Turbo	5 secs
Test 2 (ii) Floppy Disk - Secure	51 secs
Scanner Accuracy [2]	
Parasitic Viruses - Turbo	349/363
Parasitic - Secure	361/363
Boot Sector Viruses - Turbo	8/8
Boot Sector - Secure	8/8
Accuracy Percentage - Turbo	96.22%
Accuracy Percentage - Secure	99.46%
Stamina Test - Encrypting Viruses [3]	
Multiple Test: Flip	Pass
Multiple Test: Suomi	Pass
Multiple Test: Tequila	Pass
Multiple Test: Spanish Telecom 1	Pass
Multiple Test: Spanish Telecom 2	Fail
Multiple Test: Group II	Pass
Multiple Test: Group III	Pass

[1] This speed test is outlined in the test protocol described in *VB*, April 1991, pp 6-7.

[2] The test-set is outlined in *VB*, September 1991, p. 18

[3] This test to determine a scanner's ability to detect encrypted viruses was first conducted in *VB*, October 1991, pp. 7-11.

Virus Database

Detailed information about most of the viruses which *F-PROT* detects is available by selecting the 'Virus' option from the main menu. This is a particularly useful feature which, if nothing else, will save you flicking through back issues of *Virus Bulletin*!

Selections can be made from an alphabetical listing. For each virus, Skulason lists its origin (where known), its size and type (for example: 'Resident COM/EXE files') together with what is known about the virus, including any effects and destructive triggers. Where there are variants, these are listed.

Although not as comprehensive as Patricia Hoffman's *VSUM* database, it appears to be more accurate and has the benefit of adhering closely to *Virus Bulletin* published names - which the Hoffman listing does not.

Conclusion

Overall, *F-PROT* is a very good package marred by some annoying quirks. For the most part, it functions correctly - I cannot comment on the efficacy of *VIRSTOP* (its failure to install is baffling).

I don't have any feel for *F-PROT's* acceptance by the corporate computing community but I suspect that its distribution principally by bulletin boards doesn't help it any. The lack of any printed documentation is a definite drawback but since the software is (almost) free, one can't expect more than a disk-based manual.

Minor criticisms apart, *F-PROT* is a useful and *very* affordable addition to the anti-virus armoury. The package is properly maintained, fully supported and (unlike some other anti-virus packages) regularly updated. [In recent tests by the *National Computer Security Association*, *F-PROT* received the highest accuracy rating among 16 other commercial and shareware scanners tested. Ed.]. Like many other shareware programs, it is a *far* better package than most of the principal commercial offerings currently available.

Technical Details

Test Conditions: The testing for this review was conducted on two PCs. The first, a Compaq DeskPro 386/16 running under DR-DOS 6 was used for the speed tests. There are 28 megabytes in 988 files of which 428 files are binary executables and they occupy 15.2 megabytes.

For the floppy read speed tests, the 360 Kbyte Setup disk for Microsoft C 5.1 was used. This contains a total of 12 files requiring 354,804 bytes, of which 4 (238,913 bytes in volume) are executable.

The virus identification testing was conducted on an Apricot 486/25 which houses the test library. For specific details of the viruses used in all the tests, please refer to *VB*, September 1991, p.18.

Product: *F-PROT*

Developer: *Frisk Software International*, Postholf 7180, 127 Reykjavik, Iceland. Tel +354 1 694749, Fax +354 128801.

Availability: IBM, PC, AT, PS2 or compatible running MS-DOS version 2.1 or greater.

Version Evaluated: 2.01

Serial Number: N/A

Price: Free for personal use. US$1.00 per computer for commercial use with a US$20.00 minimum fee.

PRODUCT UPDATE

Mark Hamilton

The Sophos Utilities

One of the hardest and the most under-rated jobs in any software company is that of providing technical support. This is especially true for anti-virus software houses. Not only do you need to know ‘all about viruses’ but when you’re trying to talk someone through a boot sector disinfection routine, a myriad of different sector editors must be taken into account, each with differing capabilities (and bugs).

In an attempt to make life easier for its customers (or support staff?), *Sophos* recently released the ‘*Sophos Utilities*’ (*SU*) with its *Sweep* scanner. If an infection occurs, *Sweep* subscribers can request a faxed disinfection procedure or ‘cure sheet’ from *Sophos* which can be implemented using *SU*.

Two major observations regarding this program: it has been designed *specifically* for dealing with boot sector virus infections and, secondly, it cannot be used to make minor modifications to either disk files or the disk structure.

To start the program simply type ‘SU’ - but if you want to write back to disk the changes you have made, you must append a space and add ‘-wr’ to the command line. This is an intentional safety feature included to prevent a nasty accident.

```
Sophos Utilities. Press F1 for HELP or F2 to QUIT. Use cursor keys to move
around the screen and Enter to make a selection.

  Please select action:

  1  View item

  2  Copy item

  3  Search item

  4  Disk drive info, mapping and interrupts

  5  Special functions

  6  Select new drive

  7  Return to the operating system
```

Sophos Utilities (SU) main menu - providing PC Support with limited disk editing utilities to access and copy areas of disk liable to boot sector infection. Not as flexible as *The Norton Utilities*, *SU* which is incorporated within the *Sweep* scanner, was designed specifically for virus disinfection.

```
Press F1 for more info, F2 to QUIT, F3 to view as hex, F4 as text, F5 as
directory, F6 as large FAT, F7 as small FAT, F8 as partition.
FA33C08E D0BC007C 8BF45007 501FFBFC BF0006B9 0001F2A5  ·3└Ä╨.¦ï⌠P· P▼√ⁿ┐.♠╢.⊡∑Ñ
EA1D0600 00BEBE07 B304803C 80740E80 3C00751C 83C610FE  Ω↔♠..╛╛•│♦Ç< Çt♫Ç<.u∟â├►■
CB75EFCD 188B148B 4C028BEE 83C610FE CB741A80 3C0074F4  ╥u∩=↑ï¶ïL☻ï∈ â├►■╥t→Ç<.t⌠
BE8B06AC 3C00740B 56BB0700 B40ECD10 5EEBF0EB FEBF0500  ╛ï♠¼<.t♂V╗•. ┤♫═►^δ≡δ■┐♣.
BB007CB8 010257CD 135F730C 33C0CD13 4F75EDBE A306EBD3  ╗.|╕☺☻W═‼_s♀ 3└═‼Ouφ╛ú♠δ╙
BEC206BF FE7D813D 55AA75C7 8BF5EA00 7C000049 6E76616C  ╛┬♠┐■}ü=U¬u╟ ï⌡Ω.|..Inval
69642070 61727469 74696F6E 20746162 6C650045 72726F72  id partition  table.Error
206C6F61 64696E67 206F7065 72617469 6E672073 79737465   loading ope rating syste
6D004D69 7373696E 67206F70 65726174 696E6720 73797374  m.Missing op erating syst
656D0000 575EBE16 00000000 00000000 00000000 00000000  em..W^╛▬.... ............
00000000 00000000 00000000 00000000 00000000 00000000  ............ ............
00000000 00000000 00000000 00000000 00000000 00000000  ............ ............
00000000 00000000 00000000 00000000 00000000 00000000  ............ ............
00000000 00000000 00000000 00000000 00000000 00000000  ............ ............
00000000 00000000 00000000 00000000 00000000 00000000  ............ ............
00000000 00000000 00000000 00000000 00000000 00000000  ............ ............
00000000 00000000 00000000 00000000 00000000 00000000  ............ ............
00000000 00000000 00000000 00000000 00000000 00000000  ............ ............
00000000 00000000 00000000 00008001 0100043F 201E2000  ............ ..Ç☺☺.♦? ▲ .
0000E0F7 00000000 011F053F 602F00F8 00000088 08000000  ..α≈....☺▼♣? `/.°...ê◘...
00000000 00000000 00000000 00000000 00000000 00000000  ............ ............
00000000 000055AA                                      ......U¬
                           Head 0, Cylinder 0, Sector 1: byte 446 (1be hex)
```

A healthy boot sector in hex. Boot sectors (which can be copied to disk) can also be viewed in Partition Table format

In the ‘View item’ menu provided you must chose at what level you want to look at the disk, be it at an absolute or logical sector level, at the cluster level, or on a file basis. Remember that this program has been specifically designed to remove boot sector viruses - this is an important point because although you can copy data around, data is always copied in 512-byte sectors. With this in mind and for the purposes of this review, I will concentrate on the Absolute Sector mode of operation primarily.

When you select Absolute Sector, you are given the choice of which sector to view. This defaults to a display of Head 0, Cylinder 0 Sector 1 (the Master Boot Sector of hard disks) in Partition Table format. The display can be changed to hex, text, directory, large FAT, small FAT or back to Partition Table using function keys F3 to F8. F1 is reserved for help and F2 is the Quit key.

Unlike similar programs - including *The Norton Utilities* and *PC Tools* - *SU’s* screen display is extremely slow and reminiscent of some of the earliest PC programs written before programmers discovered that they could write direct to screen memory for very fast screen updates. This sluggishness is particularly noticeable when displaying a complete 512-byte sector in hexadecimal mode.

Sophos supplied copies of their ‘cure sheets’ for the New Zealand, Joshi, Spanish Telecom and Tequila boot sector viruses and I was able to cure infected disks without a hitch. The instructions contained on these sheets are clear and unequivocal and uses the View and Copy item main menu

selections. It seems that all the other menu choices - all of which work - are provided for the benefit of the inquisitive! Quite why you have the choice between small (12-bit) and large (16-bit) FAT displays is beyond me since the software could work this out automatically (as *The Norton Utilities* do) and display the information in the correct format.

One of the most useful aspects of this program - apart from its obvious disinfection capability - is its ability to save (and subsequently restore) the Master and DOS boot sectors to disk files: a highly recommended practice. A specific back-up routine for these sectors is included. However, in common with all similar utility programs which save sensitive sectors, this routine does not *automatically* save the hard drive configuration information of PS/2s which is contained in Head 0, Cylinder 0, Sector 2. Without this sector being intact, it is impossible to access the hard drive of a PS/2. SVC6.0 is but one virus which overwrites this information as part of its infection routine (see pp. 17-20). This is a relatively minor grumble as the sector can, of course, be transferred to disk with *SU* by simply accessing it and copying it - provided that you know which sector to copy!

For the purposes for which they were designed, the *Sophos Utilities* work and work well. The documentation is somewhat spartan and devoid of any diagrams or screen dumps. It would be nice to see, for example, what a proper boot sector should look like! However, a virus scanner which includes purpose-built recovery utilities such as those provided by *Sweep* with *SU* provides a realistic alternative to using untrusted virus-specific software disinfectors.

```
Interrupt table.  Press PgUp for previous 16 interrupts, PgDn for the next 16.
Press F2 to QUIT.

640K of RAM            Checksum of INTs 0 to 3F: d7892c62 and 0 to ff: 1117e0b4
  Checksum of 8 bytes pointed to by INTs 0 to 3F: 5bcbee2F and 0 to ff: ef459e1f

Int  Normal use          Segm:Offs  Addrs  Points to       First 8 bytes

 00  Divide error        0978:39f1  0d171  Transient area  b88b0d8e d8b80300
 01  Single step         0070:06f4  00df4  Transient area  cf2e891e 12002e8c
 02  NMI                 06cc:0016  06cd6  Transient area  5006b800 f000ec026
 03  One-byte interrupt  0070:06f4  00df4  Transient area  cf2e891e 12002e8c
 04  Overflow            0070:06f4  00df4  Transient area  cf2e891e 12002e8c
 05  Print-screen        f000:cc86  fcc86  ROM BIOS         501eb840 008ed8b0
 06  Undefined opcode    f000:98c8  f98c8  ROM BIOS         501e52b8 0bffba20
 07  No math coprocessor f000:98c8  f98c8  ROM BIOS         501e52b8 0bffba20
 08  IRQ0 Timer          0978:1831  0afb1  Transient area  9c2eff1e 2816fa2e
 09  IRQ1 Keyboard       0978:18cd  0b04d  Transient area  fa2e883e 58160075
 0a  IRQ2 Int control 2  06cc:0057  06d17  Transient area  eb10c898 00f04b42
 0b  IRQ3 COM2           06cc:006f  06d2f  Transient area  eb10c898 00f04b42
 0c  IRQ4 COM1           06cc:0087  06d47  Transient area  eb10c898 00f04b42
 0d  IRQ5 Fixed disk     06cc:009f  06d5f  Transient area  eb10c898 00f04b42
 0e  IRQ6 Diskette       06cc:00b7  06d77  Transient area  eb1057ef 00f04b42
 0f  IRQ7 Printer        0070:06f4  00df4  Transient area  cf2e891e 12002e8c
```

The Interrupt Table - interrupts and their locations are checksummed and an upper limit of RAM is displayed.

BETA-TEST

Dr. Keith Jackson

File Protector

File Protector is an anti-virus program which is quite different from competing software packages; it works by adding a small amount of code to each executable file. This code checks that the file has not been altered *before* execution is allowed to proceed. The version of *File Protector* received for review was very new; it was provided only as a floppy disk, with the documentation contained in a README file. I shall therefore provide no comments on the documentation (complete documentation is promised for the final version). When *File Protector* is executed, the initial screen states that it is a beta-test copy. *VB* will review such products, but the fact that the product is still in beta-test will not be permitted to excuse any glaring problems. *VB* is not a beta-test site and its reviewers test software *exactly* as it is received.

Operation

File Protector is an MS-DOS program which claims to protect executable files against viruses, or for that matter anything else that attempts to change an executable file. In its simplest form, *File Protector* ensures that the file size, date and time have not been altered from when *File Protector* code was first added to the executable file. Optionally a checksum facility can conduct a thorough byte-by-byte examination of the file.

File Protector is operated via a clear, easy to use, menu driven interface which offers the following options: *Install*, *Remove*, *Driver*, *Options*, *Quit*. The first two options are used to add/remove the extra code which *File Protector* introduces into an executable file. The addition of extra code only applies to COM and EXE executable files. The third option provides a facility to add a device driver. According to the documentation this 'helps File Protector to detect stealth viruses'. No further explanation is provided! This device driver can be executed when the computer is first powered on. The only other way of executing PC software is through the boot sector and the documentation states that protection for this area of disk will be available very soon.

I tested *File Protector* by installing its protection on a dozen executable files, randomly chosen, of various sizes. Seven of the test files were EXE files, ranging in size from 26 Kbytes to 224 Kbytes. The remaining five test files were COM files, ranging in size from 640 bytes to 46 Kbytes. Under MS-DOS , a COM file can never be greater than 64 Kbytes in size - as *File Protector* makes each executable somewhat larger (see over page), the documentation warns against using *File Protector* on COM files which are bigger than 62 Kbytes.

In most cases *File Protector* installed correctly, the only visible effect being an increase in size of the file under test. For the record, the largest file increase which I saw was 26 Kbytes when protecting an EXE file which was 195 Kbytes long, with the checksum option activated.

Quite correctly, *File Protector* notices when a file has already been protected and refuses to install itself twice. However, *File Protector* went decidedly awry when operating on *PROCOMM*, a shareware communications package. It thought that the file PROCOMM.EXE either contained an embedded overlay file (which for all I know may be true, but so what?), or was corrupt (which it definitely wasn't). *File Protector* decided to protect it as a COM file; I can't think why. The original *PROCOMM* file was 165 Kbytes long (by no means the longest file tested), but the resulting COM file was only 6 Kbytes long. Mmmm! *File Protector* did however have the sense to make a backup copy of the original file.

In no other case did the protection added by *File Protector* have any affect on a program's functionality and *File Protector's* checksum test successfully detected every single bit change which I made to the test files. This is in spite of the fact that single bit changes to the test files are far more difficult to detect than the gross changes introduced by viruses. Bit changes could in theory (if not yet in practice) be made to a file by a virus while the file resides on disk.

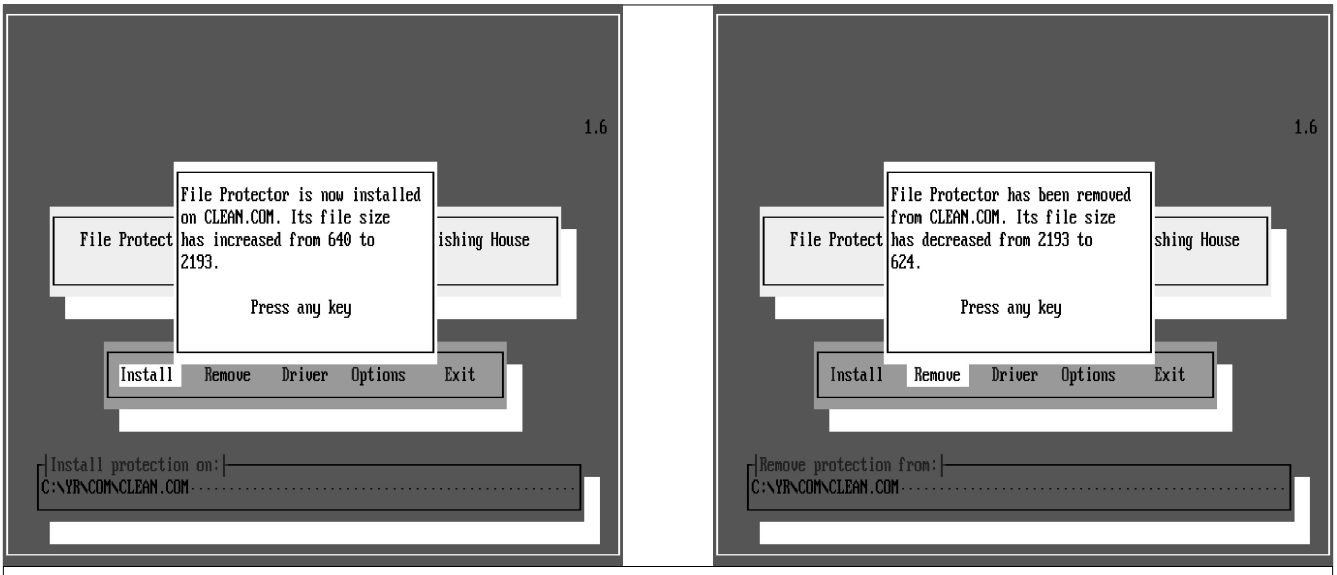
Removing File Protector

The additional code installed by *File Protector* can be removed easily from a file and this worked correctly (the original file size was reproduced) on all of my test files bar one. The exception was a small COM file called *CLEAN* (a utility which moves the disk head to facilitate usage of a head cleaner disk). The original file was only 640 bytes long. However, after *File Protector* protection had been added, then removed, the file had decreased in size to 624 bytes. File Protector did not comment on this reduction in file size. Even more curiously *CLEAN* still seemed to execute correctly. It is rather noticeable that the original file was exactly a multiple of 128 bytes in size, perhaps *File Protector* decided to remove some redundant code. Either way round, a change in file size is disturbing and would definitely agitate any other checksumming program in use.

Execution Overhead

Given that the protection offered by *File Protector* is in the form of some extra code executed only when a file is loaded from disk, this must increase the time taken to load and execute a program. I could only measure this effect accurately when *File Protector's* checksum option was in operation and a large executable file was being tested. In all other cases, the overhead introduced by *File Protector* was negligible.

Using the three test files which were larger than 200 Kbytes, I measured increases in load/execute time of 46%, 54% and 49% respectively. Therefore, a rough rule of thumb suggests that *File Protector* imposes a 50% increase in load/execute time for my particular processor/hard disk combination. This overhead is not onerous. In absolute timings the average load/execute time for a 200 Kbyte file increased from 2.3 seconds to 3.3 seconds, therefore on my test computer *File Protector* can perform its checksum calculations at the rate of approximately 400 Kbytes per second.



Shome mishtake, Ed! *File Protector* adding its protective code (left) to the *CLEAN* utility and subsequently removing itself (right). At some stage in this process some 16 bytes of CLEAN.COM file are truncated with no effect on program functioning!

Checksumming

I do not know the checksumming algorithm used in *File Protector* - it seems to be more complicated than the simple algorithms which I have criticised in the past. When a single bit of a test file is altered, *File Protector* changes three bytes near the end of the protected file. When eight bytes of the test file are altered, thirteen bytes at the end of the file change. This is symptomatic of a non-trivial checksumming algorithm.

Stealth Rears Its Ugly Head (Again!)

In common with other checksumming methods of protection, *File Protector* has a problem with stealth viruses that are actually active in memory. For example, consider the 4K virus which monitors DOS calls when it is active in memory, and can 'disinfect' any program infected by itself before loading the program, i.e. it is capable of stripping its own code from an executable file during the loading process.

I tested this problem by first executing a known virus-free test program (LIST.COM, the file viewer), which had previously been protected by *File Protector*, on a computer which had the 4K virus active in memory. The program executed normally, and *File Protector* did not report anything amiss. This infected the test program with the 4K virus (I confirmed with this with two 'scanner' programs) and again I executed the test program with 4K active in memory. Once again *File Protector* did not report any problems. This happens because the code added by 4K is removed at load time, 4K locates itself elsewhere in memory and the original file is loaded. *File Protector's* tests see nothing wrong.

Even though I confirm elsewhere in this review that *File Protector* is capable of detecting any single bit alteration, this is useless when it only 'sees' the original unchanged file. Once a stealth virus is active in memory, I cannot see any way round this problem without having other programs monitoring PC operation for currently active viruses. *File Protector* is therefore almost by definition useless against any effective stealth virus. Even booting from a known virus-free floppy disk before running the program does not prevent this problem when the *File Protector*-protected program itself already carries a stealth infection - and if the protected program must be assumed to be clean, the protection is inherently superfluous. [*A device driver which purportedly defends against stealth activity is supplied with the program. The documentation did not explain its correct use, and it was not tested, due to lack of information. The evaluator initially misunderstood its purpose altogether - what hope for the poor user?* Ed.]

Dynamic Decompressors

File Protector performs correctly with dynamically compressed EXE files so long as the compression is introduced *before File Protector's* code is added to the file. However, I found that using either *LZEXE* or *PKLITE* (the two best known dynamic file compressors) after *File Protector* had been installed produced an error on every subsequent execution of the file. Given that the act of compressing a file must change the structure and contents of the file, this is unsurprising. However, I think that it must be possible for *File Protector* to detect that such file compression had been added to a file and the user could be advised of a suitable course of action. This shortcoming is not unique to *File Protector*, all checksumming anti-virus programs suffer from this problem.

Conclusions

I have grave reservations about the philosophy behind the design of *File Protector*. It should be crystal clear by now that *File Protector* works in a manner almost identical to the way a virus would act: extra code is added to an executable file, and this code is executed every time that the file is used. This rather goes against the grain. I believe that one of the best ways to fight viruses is to avoid or detect the changes introduced by them and here I'm faced by an anti-virus utility which actually advocates changing every single executable file on a hard disk (there seems little point in adding *File Protector* to just some of the executable files). Furthermore, I fail to see what *File Protector* offers an end user over and above installing a checksumming program using a cryptographic protection algorithm.

File Protector is obviously useless for protecting any program which alters its *own* executable image, but this applies equally to *any* checksumming program. Inherent in the process of protecting each executable file is the great weakness that it gives no protection whatsoever when a non-protected program is executed. Users should be aware of this and must either take physical precautions, or use another security product to prevent unprotected programs being executed.

A far more fundamental problem is that presented by stealth viruses which is seemingly insurmountable. In a clean DOS environment *File Protector* detected every single bit file change which I introduced and would detect any more traumatic changes such as those introduced by viruses. However, the product simply keeled over and died with the ubiquitous 4K virus in memory! The device driver referred to earlier in the text *may* provide a *partial* solution to this problem for *certain* viruses under *particular* conditions. In fairness to the product, we were unable to test this.

Technical Details

Product: File Protector

Developers: Kivunim Publishing House, 20 Achad Ha'am St., Rehovot, Israel. Tel +972 8 470791/6, Fax +972 8 462834.

Availability: IBM PC/XT/AT, PS/2 or compatible. No details of system specification provided.

Version Evaluated: 1.6

Serial Number: beta

Price: US$40.00

Hardware Used: A Toshiba 3100SX laptop with a 40 Mbyte hard disk, 5 Mbytes of RAM, a 16 MHz 80386 processor and a single 3.5" floppy drive.

END-NOTES & NEWS

VB '92 - Call For Papers

Abstracts of 300 - 1000 words are invited for papers to be presented at the *2nd International Virus Bulletin Conference*, *The Edinburgh Sheraton Hotel*, Edinburgh, Scotland, September 2nd-3rd 1992. The conference will be in two streams: Stream one will address the **management of the virus threat** within the corporate environment, while Stream two will concentrate on **technical developments** including virus disassembly, detection and classification. Abstracts are welcomed from individuals or groups active in research, software or hardware development, quality assurance, the law, corporate security management, or any other field related to countering computer viruses and malicious software. **Abstracts should be completed by February 15th 1992** and should be sent to The Editor, *Virus Bulletin*, 21 The Quadrant, Abingdon Science Park, Abingdon, Oxon OX14 3YS. Fax 0235 559935.

XTree* has released *AllSafe described in its advertising as 'the ultimate in PC protection'. The package combines virus prevention with access control and security reporting. *XTree's* advertising (*Personal Computer World*, December 1991), speaks volumes: 'Even if the virus is unknown, AllSafe studies and learns its signature immediately, letting you *automatically* update the virus signature database. Then you can use AllSafe to scan other disks or systems right away before an infection can spread without waiting for a new signature from the software publisher. So, if you use a standalone PC, fear not! It's over for the evil virus!' (A free virus expert with *every* copy?) *XTree's* UK distributors are *Softsel* (Tel 081 568 8866), *P&P* (0706 217744), *Ingram Micro* (0908 260160), *Xitan* (0703 899321).

Whatever your business it always pays to have the full address and telephone number of the ***Advertising Standards Authority***. This august body is the principal watchdog of advertising standards and practices in the United Kingdom. For information write to: *ASA*, Brook House, 2-16 Torrington Place, London WC1E 7HN. Tel 071 580 5555.

Patricia Hoffman who maintains the Hypertext *VSUM Virus Information Summary List* has announced a new certification scheme for anti-virus software products called the **Scanning Product Certification Scheme**. For a fee of US$500, manufacturers can submit scanners for testing against a battery of 685 live viruses. Somewhat ominously, the final line in the information sheet states: 'This software [the scanner] will not be returned in order to avoid any problem in the unlikely event it becomes infected.' Information from Patricia Hoffman, USA. Tel 408 988 3773, Fax 408 246 3915, BBS 408 244 0813.

The ***Fifth Annual Computer Virus & Security Conference*** sponsored by the *ACM* and *IEEE*, New York, March 14-15th 1992. Tel 212 663 2315.

The *S&S Consulting Group* is holding a two-day seminar on **The Virus Threat**, Great Missenden, Bucks, UK, 15-16th January 1992. Tel 0442 877877.

Sophos is holding a two-day **computer virus workshop** in London, January 15-16th 1992. Technical and management streams are available. Tel 0235 559933.

Beware *Trisec Designs Drivelock* from South Africa! This is a 3.5 inch square chunk of plastic that you shove into the disk drive and lock with what purports to be a padlock. *VB* 'evaluated' it - the padlock fell open with the aid of a paper clip in approximately 30 seconds. A hack-saw would cut through the plastic retaining collar with ease but should such subtleties seem too time-consuming, a hefty pull on the device will also remove it from the disk drive. Strictly a Christmas cracker amusement. Suitable for ages 2-5.

VIRUS BULLETIN

Subscription price for 1 year (12 issues) including first-class/airmail delivery:

UK £195, Europe £225, International £245 (US$395)

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, 21 The Quadrant, Abingdon Science Park, Abingdon, OX14 3YS, England

Tel (0235) 555139, International Tel (+44) 235 555139
Fax (0235) 559935, International Fax (+44) 235 559935

US subscriptions only:

June Jordan, *Virus Bulletin*, 590 Danbury Road, Ridgefield, CT 06877, USA
Tel 203 431 8720, Fax 203 431 8165