June 1991 ISSN 0956-9979

VIRUS BULLETIN

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

CONTENTS

EDITORIAL

The Anti-Virus Commandos

It is unfortunate that certain members of the anti-virus industry are unable to attend the *Virus Bulletin Conference* in September. Their dilemma apparently arises from the fact that the second day of the conference is a **Friday the thirteenth** - that infamous day and date beloved by the press because it is said to be the viral day of reckoning. On this basis, certain key players, in their dedication to duty and customer support, have decided to stay away from the conference lest the dread Jerusalem virus cause international paralysis on the stroke of midnight.

These dedicated souls are the anti-virus commandos of '*SAS International*'. Waiting as they will be on the evening of Thursday September 12th, helicopter blades whirring, in tense readiness (and full body-armour) for the command -

GO! **GO!** **GO!**

- and then springing into action, landing on corporate and governmental roofs - rappelling down beleaguered capitalist facades, hurling anti-virus stun grenades into unsuspecting PCs and freeing hostage disk drives the length and breadth of England.

Those of us basking in the Jersey sunshine must count ourselves fortunate that this brave and selfless devotion will defend the realm against the forces of evil while we are away.

We Gentlemen in Jersey then a-bed
Shall think ourselves accurs'd we were not there,
And hold our manhoods cheap while any speaks
That fought the threat on England's Virus Day. [1]

Unfortunately for the press and the 'anti-virus commandos', the truth about 'Friday the thirteenth' is comparatively mundane. One virus researcher recalls a telephone conversation on a Friday the thirteenth, it went something like this:

Journalist: Had many calls today then?

'Virus Expert': Yes, hundreds.

Journalist: What sort of people have been ringing then?

'Virus Expert': Journalists.

On a serious note, computer security specialists should avoid the 'Friday the thirteenth' mentality as it is misleading - there are simply too many viruses with a multitude of trigger dates, conditions and effects as to make this day any more of a threat than any other day. The Jerusalem virus replicates on *every* day of the year, it is easily detected using scanning software and recovery from its trigger-effects is straightforward.

In the somewhat unlikely event that this particular Friday the thirteenth should turn into the 'mother of all battles' (unlike any previous Friday the thirteenth) help can be obtained from the usual sources, the majority of which have sufficient qualified technical support staff to maintain *all* the normal services as well as sending delegates to the conference. Others will have provided answerphone instructions to contact the *Hotel de France*, Jersey, from where telephone and fax support will also be provided.

The Sleeping Spires of Oxford?

'The mother of all battles' might, however, be a good description for the campaign that the computer staff at *Oxford University* are currently waging against the Spanish Telecom virus (see *VB*, January 1991).

The virus was discovered at two sites, *Oxford University* and at *City University,* London, in October and December 1990 respectively, since which time it has been spreading surreptitiously throughout *Oxford University's* PCs. The virus is known to have infected machines at *Oxford University's* computer services centre, the engineering department and at *Nuffield College* but its actual distribution is certainly far greater.

Spanish Telecom, which overwrites the hard disk on an infected machine on or after the 400th reboot, has triggered on at least three university PCs. Detective Sergeant Gerald Causer of *Thames Valley Police Fraud Squad* issued an official warning following the discovery of the virus at local commercial premises which had used *Oxford University* software. In this instance the virus had triggered on *ten* PCs and had been located in backups dating from April 1991.

Both *Sophos Ltd*. and *S&S Ltd*. have reported an increase in the occurrence of this virus and the police have expressed concern that it could spread to "epidemic proportions".

Detective Sergeant Causer would like to hear from any computer user in the UK who has been infected by this virus and he can be contacted at *Bicester Police Headquarters* on telephone number 0865 846000.

The incident highlights once again the difficulties faced by academic computing faculties which must maintain a relatively unrestrictive computing environment in which all students have access to computing facilities. The disproportionate number of users, the ready exchange of software and the constantly changing student population make controls extremely difficult to implement.

Readers should note that national reporting of computer virus incidents is handled by the *Metropolitan Police Computer Crimes Unit.* The telephone number for this unit has recently changed to 071 230 1177.

[1] Acknowledgements to W. Shakespeare and the bard of Wigston Magna.

TECHNICAL NOTES

Spanish Telecom

In the original analysis of the Spanish Telecom virus (*VB*, January 1991) it was stated that the specimen was capable of infecting both COM files and the Master Boot Sector of the fixed drive. In fact, although the author has developed a functional 'multi-partite' hybrid specimen, live testing (and reports from *Oxford University*) has shown that the virus tends to infect boot sectors rather more often than programs.

The virus uses primitive stealth techniques to conceal itself in the Master Boot Sector which again re-emphasises the age-old warning to boot the PC from a write-protected system diskette before diagnosis commences. It infects all diskettes regardless of density and contains a pernicious trigger effect whereby a counter is incremented upon bootstrapping the computer. Once the counter reaches 400 (190H) the virus will overwrite all sectors of both the first and (if present) second fixed disks repeatedly with random data from boot-time low memory.

The widespread distribution of the virus throughout *Oxford University* and its singularly destructive nature prompted the first ever official virus alert by a United Kingom police force. A search pattern for the virus is repeated here:

```
8A0E EC00 BE70 0003 F18A 4C02 8A74 03C3 ; Offset
0B3H in MBS
```

For detailed information see *VB*, January 1991, pp. 22-24).

Cryptographically Secure?

The following statement has been received from Dr. Alan Solomon in response to this month's product update (see page 18-19) in which Dr. Keith Jackson casts doubt on the cryptographic strength of the checksums used in the *Toolkit.*

> *''For data encryption, you need a strong algorithm, as the attacker is an intelligent human using statistical cryptanalysis and the human ability to recognise intelligible text.*
>
> *For cryptographic checksumming the attacker is a small program (the virus), without the ability to do what a human cryptanalyst can do. So the balance between speed and strength changes; you can use a weaker and faster, algorithm. We believe that the algorithm used by CheckVirus in conjunction with a user selectable password is adequate. Certainly, none of the existing 600 viruses can beat it.''*

The contention surrounds the use of the term *cryptographic* checksumming. The *fundamental* motivation for using cryptographic checksumming techniques is to provide *generic* detection of *any* future virus. This must also include viruses which could aim specifically to avoid detection by a known software package. If Dr. Jackson can reverse-engineer the algorithm, a virus *writer* could do the same - and use this knowledge to write a virus to circumvent the program. Cryptographic checksumming techniques are well documented and *standardised* (*ANSI X9.9, ISO 8731 part 2*), and anti-virus products are available which combine a strong algorithm with reasonable speed.

By claiming that *CheckVirus* detects the 600 viruses known to date, Dr. Solomon is reducing its stature to that of a scanning package - in which case, why bother?

Disk Duplication

An increasing threat to software developers is the danger of distributing shrink-wrapped virus code. In an insecure production environment, infection can occur during software development and testing, or during duplication.

Ideally, QA procedures should prohibit the transfer of binary (executable) code from the development machine to the duplication machine - source code, after all, cannot be infected! The only executables stored on the duplication and/ or master disk production machines should be trusted system files, compiler and image copier. Before software distribution, disks should be spot-checked for known viral infection.

Boot sector virus infection on a duplication machine was responsible for one very recent incident in the UK of shrink-wrapped virus distribution. In this instance, because the software release was highly specialised and all end-users were contactable, no damage resulted. However, this incident once again emphasises the need for vigilance. Software developers who use disk duplication sub-contractors should apply particular scrutiny to duplication QA procedures.

Infected master software is the other obvious path for wide-spread virus distribution. If master software is transferred to the duplication machine (as opposed to source code) the use of reliable virus scanning programs should be regarded as the barest minimum of anti-virus quality assurance.

The GP1 Virus

The GP1 virus contains instructions to subvert *NetWare* security but despite the efforts of various researchers its efficacy in spreading across a network is unknown. Analysis of the virus presents a number of difficulties, not least of which is the fact that *NetWare* operation is undocumented and is the proprietary (and much guarded) property of *Novell.* The MS-DOS sections of the virus are readily understood as, indeed, are the majority of the *NetWare* specific sections of the code but, in this instance, the *exact* effect of a particular interrupt call remains obscure. This virus *is* a deliberate and concerted attempt to circumvent *NetWare* security although it is probable that the code will only function on a specific release of *NetWare.* It is not known whether or not GP1 has been widely circulated - all that can be said conclusively is that *Novell NetWare* has now been targeted.

LETTERS

Table Inaccuracies...

Dear Sir,

I read with interest Mark Hamilton's comparative review of twelve virus scanning programs (*VB*, April 1991). There are a few errors which I should like to correct for your readers. The features table on pages 10 and 11 of the edition indicates that *VPSCAN* 1.1a does not perform memory checking for viruses. In fact *VPSCAN* incorporates scanning of conventional memory for viruses as a command line option.

The table also indicates that *VPSCAN* doesn't provide single file checking. In fact *VPSCAN* has always had this option. One simply specifies the chosen file on the *VPSCAN* command line.

It should also be pointed out that *VPSCAN* provides disinfection of files infected by the most common viruses including Stoned and Jersualem B and provides deletion of any other infected files. *VPSCAN* also detects new viruses specified either in a proprietary format or in the *VB* format.

Finally, the table indicates that the memory-resident scanner that accompanies *VPSCAN* is "non-virus specific". In fact, the *Virex-PC* memory-resident program provides both virus specific and non-virus specific monitoring.

Your readership may like to know that we have recently started to release a fully functional scanner into the BBS community. There is no charge associated with this scanner, its purpose being to demonstrate the power of the full *Virex-PC* package and to provide viral protection for those who might otherwise be unable to purchase our full package.

We look forward to your review of the next release of Virex-PC: we were late in getting it out the door, but we're sure that you'll find it was worth the wait!

Sincerely,

Ross M. Greenberg
Author, *Virex-PC*

Mark Hamilton replies...

I read Ross Greenberg's letter with interest and note the points he makes. The old adage, "no one knows a product better than its author" is so true. When compiling the table of features, I had to rely, in part, on each manufacturer's documentation. Unfortunately, *Virex-PC* was one of the worst products in terms of its documentation. However, I do agree that I overlooked the disinfection/removal capabilities of the *Virex-PC* product. My apologies to Mr. Greenberg for this oversight.

Test-Set Confusion

In looking through the product reviews in the last few issues of *VB*, I noticed an oddity; whereas *VIRSCAN* and the products evaluated in the comparative review were run against a test-suite of 306 file-infectors (and, at least for *VIRSCAN*, 7 boot-infectors), some other products, such as *VISCAN* and *VET*, were run against a test suite of 112 file-infectors and 2 boot-infectors.

I imagine this is just simple disorganisation (were the reviews written at different times, or by different testers?). I'm concerned that, for instance, someone will compare hit percentages when comparing two products reviewed, without realising that the reviews used widely different test suites. Perhaps the *VB* could say just what its policy is as to what test suites will be used for what product and clarify what the precentages mean.

Dave Chess
IBM

Disorganisation? What thorough impudence!

Well, we admit our fallibility and Dave Chess raises an important point which is worth clarifying. There are currently two evaluators working on *VB* product reviews quite independently of each other.

Mark Hamilton conducts reviews using the test-set which was published on page 8 of the April 1991 edition. Thus the results from *VIRSCAN*, which was evaluated by Mr. Hamilton last month (*VB*, May 1991, p.16), are directly comparable with those obtained from the 12 scanners reviewed in April and the 15 scanners which appear in the review update on pages 10-11 of this month's edition.

Dr. Jackson concentrates rather more on 'standalone' reviews which provide a more in-depth insight into the capabilities of anti-virus products whether they be scanners, checksummers, monitors or, as will be the case next month, hard cards.

Dr. Jackson has assembled a rather smaller test-set which is published alongside his reviews. The only exception to this is in this month's review of *Dr. Solomon's Anti-Virus Toolkit*, where Dr. Jackson has deliberately omitted scanner accuracy and speed results because this information is provided on page 11 of this edition. To repeat this exercise was felt to be a wasteful duplication of effort. Indeed, duplicated effort became apparent by the publication of Dr. Jackson's review of *Bates Associates' VISCAN* (*VB*, April 1991, pp.26-27) in which the product was tested independently aginst two test-sets, one of which was a sub-set of the other!

We hope to produce a single 'official' test-set in the near future so as to avoid any further confusion. As the number of both viruses and virus scanners continues to proliferate and as efforts intensify to evaluate anti-virus products, this sort of organisational anomoly is liable to occur. We apologise for any confusion that may have arisen in this case.

BRIEFING

Jim Bates

A Novell-Specific Virus

[*Editor's note*: A computer virus called GP1 was received in April 1991 from the Netherlands which contains instructions to subvert network security. There has been a previous unsubstantiated report of the existence of such a virus (*VB*, February 1990, p.2) as well as persistent claims by Dr. Jon David of the United States to have witnessed a virus subverting *NetWare* security - claims which *Novell* denied and which no other recognised virus researcher or security expert was able to validate (*VB*, December 1990, pp.2-4). The appearance of GP1 confirms that a *NetWare*-specific virus has now been developed but no information about the extent of its distribution is available. The origins of the virus are shrouded in mystery - it is believed to have been developed in Leiden, Holland, and is rumoured to be the result of a challenge. On the premise that rumours are best ignored, it was decided that the virus should be reported from a structural and functional viewpoint. The GP1 virus was fully disassembled and subversive instructions located within its code. Readers should note that to date this virus has not been tested on an active LAN and an assembly listing of the code suggests its intended operation has been disabled intentionally. In the following report, detailed information about certain *NetWare* addresses and calls is not included.]

The GP1 Virus

Although this virus is based on the Jerusalem series, the specimen examined is unusual in several respects. Firstly, this is the first time I have had an original virus generating program to examine, meaning that the sample was **not** an infected file but the *original* code by which a virus could be introduced into a system. Secondly, this is the first virus I have seen which contains code specifically designed to confirm the existence of *Novell* network code before becoming infective. Finally, accompanying the sample is an uncommented assembler listing which shows definite signs of being a disassembly of a Jerusalem virus with *Novell*-specific code and other modifications added to it.

The replication code is unremarkable - infection takes place only during LOAD and EXECUTE function calls (4B00H) to DOS INT 21H. Both COM and EXE files are infected but COMMAND.COM is specifically excluded. The 'standard' Jerusalem infection pattern is apparent whereby COM files have the virus code *prefixed* to the host file, while EXE files have it *suffixed* (with the attendant changes to the file header). The infective length varies (because of various paragraph alignment calculations) between 1563 and 1580 bytes and setting the HIDDEN attribute on files provides no defence against it. The virus marks infected files with a signature of "MsDOS" at the last five bytes of the file and after infection restores the Date and Time settings of the file.

The code contains no stealth routines and is not encrypted, so detection is a fairly simple matter (a recognition pattern was published in last month's *VB*). The code remains resident in memory and monitors only INT 21H although a temporary dummy Critical Error handler (INT 24H) is installed during infection. This virus is of the resident type which uses an 'Are you there?' call, in this case placing a value of 0F7H into the AH register and issuing an INT 12H instruction will return a value of 0300h in AX if the virus is resident. There is no trigger routine although the interference with *Novell*-specific operations may cause unpredictable effects.

An Experiment?

The sample analysed displays all the appearances of an experimental program, and this is one of the aspects which provokes most interest. The actual structure of the program (a genuine EXE file) can be divided into several distinct sections - there is the resident section of code, consisting of the COM file entry routine, EXE file entry routine and the usual INT 21H handler (which contains the infection routines). There is a section (towards the end of the code) which is outside the resident area and this contains the program's direct entry point. Within this non-resident area, the code has two distinct purposes depending upon whether the virus code is resident and if a parameter has been added to the command line used to invoke the program in the first place. If the virus is **not** resident, this initialisation code installs it in memory and exits through a normal TSR (Terminate and Stay Resident) function call. If the virus **is** resident but no parameter was given, the code simply exits without doing anything. However, if the virus is resident **and** a parameter of 'i' is given when the program loads, the virus is removed from memory and a message is displayed which says : 'GP1 Removed from memory.'. In my experience of disassembling virus code, it is unique to come across a program which functions in this way and this, in conjunction with one or two other observations leads me to conclude that this is the original 'virus generating' code.

It should be noted that the collection of the normal *Novell* network handler address is accessed via a FAR JMP instruction rather than a FAR CALL (thus giving an incorrect return address on the stack to the handler). This effectively disables the virus from spreading over a network. Having said that, the uncommented assembler file received with the virus has a conditional structure which includes this FAR JMP instruction, thus making it possible to change the whole complexion of the program code from a relatively harmless exercise into a highly infective virus by simply changing one instruction. For the purposes of further analysis, I therefore assumed this instruction to have been changed accordingly.

The *Novell*-specific aspects of the program naturally formed the focus of most attention during disassembly, but for the first time during analysis I was unable to verify my analysis by direct observation of the virus under actual live conditions. It is understandably difficult to persuade anyone with a *Novell* network to allow virus code loose on their system, especially when its operation is still fairly speculative.

Having said this, most of the code is obvious in its operation and in only **one** area am I unable to verify the *exact* functionality of the virus.

Operation

As mentioned above, this virus is **not** infective unless *Novell* networking software is present on the system. The virus issues a special function call to verify the existence of the network software and if this fails, processing aborts back to the host program. Once the presence of network software has been verified, the entry point of the network IPX service routine is collected and stored within the virus code for future use.

The virus' INT 21H handler monitors all DOS function requests but only intercepts four of them. Two of these will service the virus' own 'Are you there?' call and the peculiar way in which the Jerusalem virus handles the preliminary execution of infected COM files. The third function intercepted is 4BH as described on page 5. The fourth and most interesting function intercepted is the 0E3H request for service to *Novell NetWare*. When such a request is received, it is checked to see whether the subfunction is requesting a user logon procedure. Any other request than this is allowed to pass directly on to the normal DOS/*NetWare* INT 21H handler but the logon request is executed under control of the virus so that the return code can be examined. If the return code indicates that the logon was unsuccessful then processing is allowed to continue unmolested.

If the logon *is* successful, a series of highly specific network calls are issued (to a socket number reserved by *NetWare*) in a sequence which culminates in the sending of a 'packet' (a network message) onto the network, which is coded so that the virus gains *privileged access* to network drives.

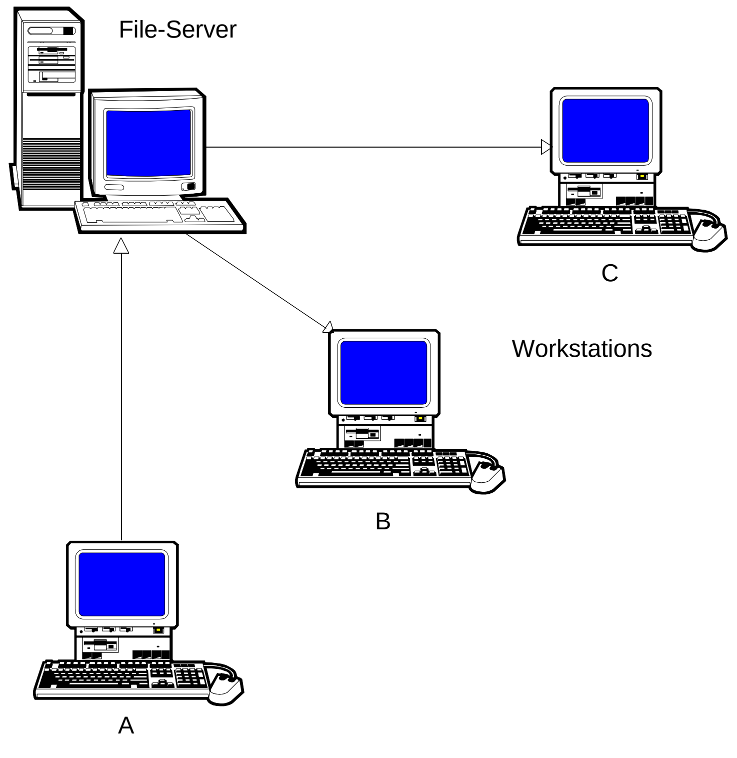

Figure 1. Network vulnerability. The GP1 virus contains code which is designed to circumvent *NetWare* security by gaining privileged system access on a *Novell* network.

No testing of this virus has been undertaken on an active Local Area Network. Examination of the code indicates that if the virus is executed on workstation A by a user with limited system privileges, who subsequently logs on to the network, it will infect the file-server regardless of *NetWare's* security settings. Subsequently users on workstations B and C executing infected programs on the network will import the virus and infect local drives. Running shared programs in this way would effectively spread the virus across the network.

The actual mechanisms by which this is achieved display an intimate knowledge of the way the network operates - however, it is neither necessary nor prudent to describe these mechanisms in detail within this analysis. Suffice it to say that the effect is to allow the virus to spread across the network. The extent of this spreading is the one area I was unable to verify without access to a suitable networked system.

Conclusions

The arrival of a PC virus containing network specific code is an event which has been predicted for some time by genuine researchers. The prudent maxim adopted by most of us is ‘if it can be done - it will be done.’

A recent incident in the United States where an individual suggested that a copy of Jerusalem had propagated across a network was unsubstantiated. It appeared that the network in question may not have been securely configured and the virus sample was never produced for accurate disassembly.

In the case of the GP1 virus however, there is no doubt about the nature of the code, which is obviously designed to sidestep normal network security arrangements. The only minor surprise is that it has taken so long to arrive.

There are many recorded instances of viruses propagating across various types of PC network, but this is invariably when the configuration of network security has been lax or non-existent (see *VB*, December 1990, p.3). Such propagation occurs purely because the network software is designed to be as transparent as possible, even at DOS level. However, the security considerations are generally very efficient at preventing unauthorised access to areas specified during configuration and to date I know of no virus (apart from GP1) designed to avoid the protection measures built into *Novell NetWare*.

It now appears that the warning that **no** protection system can be 100 percent effective against virus code without introducing fundamental changes in machine functionality, has been proven once again.

Examination of the original assembler routine seems to indicate that extensive testing was conducted with the intention of violating network security in a number of different ways. This virus appears to be only one result of such tests, intelligent guesswork indicates some other possibilities which would have equally serious repercussions.

A written report, together with the sample, the assembler listing and my own disassembly and analysis has been passed both to *Novell*’s head office in Provo, Utah, USA and the *Computer Crimes Unit* at New Scotland Yard. Further testing will need to be conducted to assess the extent to which this virus *actually* spreads in a live *NetWare* environment under different software releases and with different configurations. It is to be hoped that investigation will quickly identify the perpetrators before any damage is done.

PC Network Security

Although the IBM PC set a single standard which has been widely adopted, PC networks have been developed by several manufacturers and there are no standards comparable to those for the PC.

Simple networks allow several users to access the file-server which acts as big shared disk. Any user can write to it and access any directory. Such networks offer no general security and no protection against viruses. If a user’s PC becomes infected with a parasitic virus and the user executes a program residing on the file server, the program on the file server will become infected. Any other user executing that program from then on will become infected.

Few networks used today are so primitive. Most offer some degree of protection against writing to designated areas such as the directory containing executables. The best security features at present are offered by *Novell NetWare* which provides four different aspects of file-server security: the **login procedure**, **trustee rights**, **directory rights** and **file attributes**.

➤ **The login procedure** requires all users to identify themselves by a username and a password.

➤ **Trustee rights** are granted to each user by the ‘network supervisor’ and allow each user to undertake various actions such as reading files, writing to files, searching for directories etc.

➤ **Directory rights** (read, open, close, delete) limit access to certain directories.

➤ **File attributes** (read-only, read-write, share) can be set separately.

The above security aspects mean that even if a user’s PC becomes infected, the infection can be controlled on the file-server. The security breaks down only if the *network supervisor* installs and executes a virus infected program.

However, the GP1 virus interferes with the login procedure in order to gain privileged access so that it may spread unhindered across the network.

KNOWN IBM PC VIRUSES (UPDATE)

Amendments and additions to the *Virus Bulletin Table of Known IBM PC Viruses* as of 20th May 1991. The full table was last published in January 1991 and will be published again in the next edition of *VB*. Hexadecimal patterns can be used to detect the presence of the virus with the 'search' routine of disk utility programs or, preferably, can be added to virus scanning programs which contain upgradeable pattern libraries.

Type Codes

C = Infects COM files **E** = Infects EXE files **D** = Infects DOS Boot Sectors (Logical sector 0 on disk)

M = Infects Master Boot Sector (Track 0, Head, Sector 1) **N** = Not Memory-resident after infection

R = Memory-resident after infection **P** = Companion virus

SEEN VIRUSES

217-A - CN: A minor modification of the 217 virus - perhaps changed to bypass some scanner. Detected by the '217' pattern.

268-Plus - CN: When this virus is run it will infect all COM files in the current directory increasing the first one by 268 bytes, the second by 269 bytes, the third by 270 bytes and so on. The virus is encrypted and is awaiting analysis.

```
268-Plus        8EC1 0650 BE00 0156 31FF B90B 01F3 A4BD ; Offset 005
```

1028 - CER: Virus is 1028 bytes long. Awaiting analysis.

```
1028            0606 005E 561E 0E33 FF8E DFC5 0684 002E ; Offset 0E9
```

Arf - CN: A 1000 byte variant of the Violator virus. Will display "Arf Arf! Got you!" when it activates. Detected by the 'Violator' pattern.

Backtime - CR: A 528 byte virus which is awaiting analysis.

```
Backtime        2125 CD21 8CC8 8ED8 8EC0 58BB 0001 53C3 ; Offset 1F8
```

Bandit - EN: This 2653 byte virus is detected by the 'Old Yankee' pattern. Awaiting analysis.

Bljec - CN: A family of small viruses, all awaiting analysis. The following variants are known: Bljec-3 (231), Bljec-4 (247), Bljec-5 (267), Bljec-6 (270), Bljec-7 (287), Bljec-8 (358) and Bljec-9 (369)

```
Bljec           B980 00BE 7FFF BF80 00F3 A4B8 F3A4 A3F9 ; Offset variable
```

Boys - CN: A 500 byte virus containing the text "The good and the bad boys". Awaiting analysis.

```
Boys            BE01 01AD 0503 0050 8BF0 BF00 01B9 0500 ; Offset 042
```

Carfield - CER: A 1508 byte variant of Jerusalem. Detected by the 'Jerusalem-1' pattern.

Damage - CER: Two related viruses, 1063 and 1110 bytes long, which cause ''Sector not found'' errors, by reformatting selected areas of diskettes. Detected by the 'Diamond' pattern.

Darth Vader - CR: A family of small viruses, probably from Bulgaria. Some of the 4 known variants contain code which will only work on '286' processors and above. Awaiting analysis.

```
Darth Vader     B820 12CD 2F26 8A1D B816 12CD 2F ; Offset variable
```

Diamond-1173, David - CER: A modification of the Diamond-B virus, produced by inserting NOP instructions and making other minor changes. Contains errors which will generally cause infected COM files to crash. Detected by the 'Diamond' pattern.

Discom - CR: A 2053 byte variant of the Jerusalem virus. Awaiting analysis.

```
Discom          57CD 2172 1F8B F18B FAB8 0242 B9FF FFBA ; Offset 139
```

Eddie-1801 - CER: A minor variant of the Eddie (Dark Avenger) virus, only one byte longer and detected by the same pattern.

ETC - CN: A 700 byte virus, containing the text "Virus, (c) ETC". Awaiting analysis.

```
ETC             8B16 0201 83C2 33CD 2172 CD89 D68B 043D ; Offset 061
```

Evil Empire B - MR: An encrypted variant, probably written by the same author as the variant reported last month.

```
Evil Empire B   8CC8 8ED8 8EC0 BF05 00B9 9A01 FC8A 0504 ; Offset 19F
```

Gremlin - CER: A 1146 byte variant of Diamond. Detected by the 'Diamond' pattern.

Guru, Bhaktivedanta - CER: A 1250 byte variant of the Murphy virus, contianing the text ‘‘Bhaktivedanta Swami Prabhupada (1896-1977). Detected by the pattern for the HIV virus.

Horse, Hacker, Black Horse - CER: A family of viruses probably from Bulgaria. Currently 8 different variants are known, which can be divided into two groups, with a different pattern required for each group. Awaiting analysis. The first group contains Horse-1 (1154), Horse-2 (1158), Horse-2B (1160) and Horse-7 (1152).

```
Horse (1)          00A3 0001 8B46 02A3 0201 B800 018C CAEB ; Offset variable
```

The second group contains Horse-3 (1610), Horse-4 (1776), Horse-5 (1576) and Horse-6 (1594).

```
Horse (2)          570E 07B9 0800 F3A4 B02E AAB9 0300 F3A4 ; Offset variable
```

Keypress-1228 - CER: Only slightly different from the 1232 byte variant, but was discovered in Kansas. It is detected by the pattern published in the January 1991 edition.

MG-1A - CR: A very minor modification of the MG virus.

MIR - CER: A 1745 byte variant of the Eddie virus (Dark Avenger). The first generation sample contains the text “M.I.R. *-*-*-* Sign of the time!”, but it is corrupted in later generations. Detected by the ‘Dark Avenger’ pattern.

Murphy-3 - CER: A 1284 byte variant of the Murphy virus. Detected by the ‘HIV’ search pattern.

Murphy-4 - CER: A 1480 byte variant of the Murphy virus. Detected by the ‘Murphy-2’ search pattern.

Mutant - CN: Three variants of this virus are known, of which two, 123 and 127 bytes long, are only able to infect small files correctly, but this is corrected in the third variant, also 127 bytes long. The viruses have no interesting effects.

```
Mutant             C98B D1B8 0042 CD21 5972 065A 52B4 40CD ; Offset variable
```

NTKC, C-23693: A 23693 byte variant of Vienna, detected by the ‘Vienna (4)’ pattern.

Pixel - CN: Several new variants of the Pixel/Amstrad virus have been discovered, most of which are very similar to previous variants, and are detectable by published patterns.

```
                   Pixel-257,275,283,295 : detected by the 'Pixel-277' pattern
                   Pixel-892 : detected by the 'Pixel-345' pattern
                   Pixel-779,837,850,854 : detected by the 'Amstrad' pattern
```

Pixel-936 - CN: A 936 byte variant of the Pixel/Amstrad virus.

```
Pixel-936          C706 0001 0001 2E8C 1E02 012E FF2E 0001 ; Offset 198
```

Raubkopie - CR: This virus adds 2219 bytes in front of COM files, but much of that is occupied by a text message in German, directed against pirated software.

```
Raubkopie          0500 013D 0002 7204 25FF 0142 B104 D3E8 ; Offset 537
```

Smack, Patricia - CER: A variant of the HIV virus, containing a message for Patricia Hoffman. Two variants are known, 1835 and 1841 bytes, both probably written by the same person, who calls himself “Cracker Jack”. Both variants can be detected by the ‘HIV’ pattern published in the May edition.

South African 416 - CN: Minor variant of the South African virus.

```
S African 416      FF36 0301 FF36 0501 B43F B903 00BA 0301 ; Offset variable
```

Sylvia-2 - CN: This version of the Sylvia virus has been patched to avoid detection, but appears functionally equivalent to the Sylvia virus. Like the original it is 1332 bytes long, just as the original, and detected by the ‘Sylvia’ pattern.

Tequila - EMR: An encrypted, multi-partite, self-modifying virus from Switzerland. Contains encrypted text ‘Welcome to T. TEQUILA’s latest production”, “Contact T. TEQUILA/ P.O. Box 543/6312 St’hausen/Switzerland’. No pattern for infected files is possible. Displays a crude Mandelbrot set pattern on screen. (*VB*, June 1991)

```
Tequila            B82A 0250 B805 028B 0E30 7C41 8B16 327C in MBS
```

VCS 1.0 - CN: A 1077 byte virus which will delete AUTOEXEC.BAT and CONFIG.SYS when it activates. Generated by a German program called ‘Virus Construction Set’.

```
VCS 1.0            89FE AC32 C4AA E2FA C35E 81EE 0301 56E8 ; Offset 00E
```

Vienna-645 - CN: A 645 byte variant of Vienna, detected by the ‘Vienna-1’ pattern.

Warrior - EN: This virus adds 1012 bytes to any files it infects. It contains the following text: ‘...and justice to all! (US constitution) Dream over ... And the alone warrior is warrior. The powerfull WARRIOR!’ Awaiting analysis.

```
Warrior            AC2C 8032 E403 F826 8035 01E2 F3B4 19CD ; Offset 0AE
```

SCANNER UPDATE

Virus Scanners - A Progress Report

Following publication of the comparative review of virus scanners which appeared in April, *VB* has decided to publish a regular progress report designed to monitor the development and performance of these programs.

Note: *Sophos Ltd* (*SWEEP*) and *Bates Associates* (*VISCAN*) have regular access to the *VB* virus collection; both packages are thus included for control purposes. For a full explanation of the evaluation protocol, declaration of interests and provisos see *VB*, April 1991, pp. 5-8.

Updates and New Entries

There is a certain similarity between the contents of the table on the opposite page and in the one which appears on page 15 of the April 1991 edition of *Virus Bulletin*. The reason for this is straightforward: few of the suppliers have issued updates in the intervening period. The issue of updates (or rather the lack of them) will become more apparent over the coming months. Only eight of the 16 packages achieve a 90% detection rating which has been established as a benchmark.

Updated products have been received from *Bates Associates*, *RG Software* (with the release of version 6.00 of *Vi-Spy*) and *S&S*; none of the other suppliers, whose products were reviewed in April, have issued updates - with the exception of *Sophos*, but this update did not arrive in time for this review. Skulason's F-PROT version 1.16 is due for release within the next week. Results from IBM's *Virscan*, *Central Point Anti-Virus* (reviewed on pages 20-22 of this issue) and *Defiant Systems' VSCAN* scanner are included.

The *S&S* update was accompanied by a Beta-test copy of the company's new memory-resident scanner, *Virus Guard.* The official release of this program is now available and we hope to publish a review presently. (A retrospective review of *Dr. Solomon's Anti-Virus Toolkit* appears on pages 18-19 of this month's *VB.)*

Bates Associates' VIS Utilities have been improved and a new manual produced. This includes a chapter entitled "Bedtime Reading" which provides a light-hearted introduction to the world of computer viruses. Interestingly, the documentation includes a section called *Notes for Software Reviewers*, which implores magazine editors not to succumb to pressure from software developers who withdraw advertising, or use other dirty tricks to suppress unfavourable reviews.

'New entries' include *Central Point's Anti-Virus* which achieves a high detection rate - it is interesting to note the disparity in scanner accuracy when the program is run in its default 'turbo' mode and in its most secure setting.

IBM's *VIRSCAN* was reviewed in last month's *VB* - the flexibility of the IBM package, particularly its command line options and the ability to update the scan data even with search strings that include 'wild cards', make it highly suitable as a supplementary scanner. IBM has just sent us a file (ADDENDA.LST) which contains all *VB* search patterns for incorporation in *VIRSCAN* - unfortunately, this arrived too late for the results to be recorded in this edition.

The other new entry is *the VSCAN* program contained in *Virus Hunter* from *Defiant Systems Ltd*. of the UK. If you believe everything that appears in print, then this package is "A complete solution to the PC virus problem". If only that were true. This package, more than any other, demonstrates the disparity in detection accuracy when the program is run in 'Turbo' and 'Secure' modes. On hard disks in particular, the 'Secure' mode imposes a heavy time penalty. The documentation provides a useful insight to viruses, but this is countered by the software's poor detection rating. The update frequency of this package is not known at the time of writing. The price of the package is £99.95. Telephone UK +44(0)752 603746.

To the Software Suppliers

It is our intention to publish updates to this column and chart the progress of various virus scanners at regular intervals. Manufacturers wishing their scanner to appear regularly in this feature should send the software to the *VB* office as and when new versions appear. The latest version of the software will always be used, provided it arrives in good time. The *VB* virus test-set will be revised occasionally. To maintain continuity, the same test-set will be used for at least three review-months.

Test Conditions

All the scanners were executed from a 3.5 inch diskette. Where timing measurements were taken, the times included the time required to load the program from the diskette, perform any initialisations and (where applicable) automatic memory scans. Disk caching software was disabled.

Two different PCs were used for the tests. The first was a Compaq Deskpro 386/16. This is a 16 MHz 386 ISA PC with 6 Mb RAM and two 42 Mb hard disks, each of which was partitioned into two 21 Mb logical drives. The hard disk speed test was conducted on a 21 Mb partition containing 887 files (of which 316 were COM or EXE executables) occupying 20.5 Mb. The floppy test was conducted using a 360 Kb 5.25 inch floppy disk (Microsoft C V5.1 Setup disk) which contained 10 files, of which 3 were executable, and occupied 354,747 bytes. This PC was used for the timing tests and the boot sector recognition tests.

The virus test-set was installed on an Apricot Qi 486-25-320. This is a 25 MHz 486 MCA PC fitted with 16 MB of RAM and a 320 MB SCSI hard drive which was partitioned into 10 logical drives. Part of the extended memory was configured as a RAM disk thus providing drives A to M inclusive.

306 different parasitic viruses were used to generate genuine COM and EXE file infections. Seven diskettes were infected with different boot sector viruses.

VIRUS BULLETIN **©1991 Virus Bulletin Ltd, 21 The Quadrant, Abingdon Science Park, Oxon, OX14 3YS, England. Tel (+44) 235 555139. /90/$0.00+2.50** This bulletin is available only to qualified subscribers. No part of this publication may be reproduced, stored in a retrieval system, or transmitted by any form or by any means, electronic, magnetic, optical or photocopying, without the prior written permission of the publishers.

IBM PC VIRUS SCANNERS (UPDATE)

RESULTS TABLE - SCANNING SPEEDS [TESTS 1(i),1(ii), 2(i) 2(ii)] (See *VB*, April 1991, pp. 6-7)

Product	**Version**	**Supplier**	**Hard Disk ‘Turbo ’**	**Hard Disk ‘Secure’**	**Diskette ‘Turbo’**	**Diskette ‘Secure’**
CP ANTI-VIRUS	1.0	Central Point	2:13	120:39	0:05	4:34
F-FCHK	1.14a	Skulason	6:23	11:47	0:35	1:06
FINDVIRUS	4.31	S&S	1:11	2:22	0:36	0:41
HTSCAN	1.12	Harry Thijssen	2:18	3:35	0:39	0:52
NORTON A/V	1.01	Symantec	1:56	N/A	0:39	N/A
PC-EYE	2.0b	PC Enhancements	1:12	3:57	0:24	0:43
SCAN	V74-B	McAfee Associates	3:41	6:14	0:59	1:26
SWEEP	2.23	Sophos Ltd	3:38	5:25	0:39	0:50
TBSCAN	2.0	ESaSS	1:25	2:53	0:14	0:32
VIRFIND	1.4	Visionsoft	N/A	84:39	N/A	5:10
VIRSCAN	2.0	IBM	3:16	4:03	0:51	1:09
VISCAN	3.14	Bates Associates	3:17	3:25	0:19	0:24
VI-SPY	6.0	RG Software	3:02	5:01	0:31	0:55
VPCSCAN	1.1a	Microcom	1:07	4:11	0:17	0:46
VSCAN	3.3	Defiant Systems	1:53	14:33	0:23	0:56

RESULTS TABLE - SCANNER ACCURACY [TESTS 3/4] (See *VB*, April 1991, pp. 6-7)

Product	**306 Parasitic Viruses**		**7 Boot Sector Viruses**		**Accuracy Percentage**	
	‘Turbo ’	**‘Secure’**	**‘Turbo ’**	**‘Secure’**	**‘Turbo’**	**‘Secure’**
CP ANTI-VIRUS	279	288	7	7	91.37%	94.42%
F-FCHK	301	301	6	6	98.08%	98.08%
FINDVIRUS	288	288	6	6	93.92%	93.92%
HTSCAN	226	226	6	6	74.12%	74.12%
NORTON A/V	216	N/A	6	N/A	70.92%	N/A
PC-EYE	287	299	7	7	93.93%	97.76%
SCAN	285	285	7	7	93.29%	93.29%
SWEEP	306	306	7	7	100.00%	100.00%
TBSCAN	222	226	7	7	73.16%	73.16%
VIRFIND	N/A	109	N/A	5	N/A	36.42%
VIRSCAN	235	239	7	7	77.3%	78.59%
VISCAN	306	306	7	7	100.00%	100.00%
VI-SPY	294	294	6	6	95.85%	95.85%
VPCSCAN	177	177	5	5	58.15%	58.15%
VSCAN	117	189	5	5	38.97%	61.98%

OVERVIEW

Virus Scanners - Optimisation and Maintainance

There has been increasing controversy over the relative merits of different software packages which are designed to identify the presence of known virus code within executable files. With most software applications, the user can easily judge for himself which suits him best. However, with virus search programs, unless he has a comprehensive selection of live virus samples, he is completely unable to evaluate the most important aspect of these programs: their accuracy.

Known as 'scanners', such programs are bound to suffer from the problems associated with any virus-specific package, and that is the continuing arrival of new (and unknown) virus code from around the world. All scanning programs can only be as good as the research which went into the collection and collation of the virus information that they contain. Similarly, scanning programs are always at least one step behind the virus writers and the user can therefore summarily discard scanners which claim to detect all viruses (known and unknown) both now and in the future.

The principle behind scanning programs is quite simple: a recognisable sequence of bytes is extracted from analysed virus code and put into a routine which will search target files for an exact match to that sequence. Any computer literate schoolboy could write such a program and detect viruses with 100% accuracy. Unfortunately life is more complex and it is how the various packages deal with these complexities that gives rise to their differences.

There are several factors to be considered concerning both the search engine and viruses themselves. Obviously accuracy in detecting virus code is the prime consideration and at the same time, the number of 'false positive' indications of infection should be kept to a minimum. Both of these requirements are a direct function of just how unique an extracted byte pattern is, while still retaining a minimum number of elements. When note is taken of the increasing number of self-encrypting viruses, it will be realised that the extraction of reliable byte patterns often requires lengthy research and considerable skill.

Speed Optimisation

A secondary consideration is the practical one of just how fast a scanning program is in day-to-day operation. If the scanning process is too slow and laborious, there is every chance that operators will not use it as often as they should. These two major requirements immediately produce a conflict since for total security it is desirable to check *all* bytes of *all* files and use large search patterns. However, limiting the number of files and the length of the patterns will generally improve speed. As the number of virus patterns increases, it is expected that the speed of scanning packages will deteriorate and optimisation of the search engine will become a priority.

Wildcards and Meta Languages

As mentioned briefly above, an additional consideration is the introduction of self-encrypting viruses. These present special problems which can only be addressed by building special capabilities into the search mechanism of a scanner.

The Achilles heel of such viruses is that somewhere within them will be a decryption routine which cannot itself be encrypted. Some viruses adopt a method of randomly selecting one of a number of routines which they maintain in their code. However it is done, there will be recognisable code somewhere within the virus. Such options as "wildcard", "multiple wildcard" and "floating wildcard" control characters within a search pattern are of immense value when searching for self-modifying encryption routines since they allow the selection of recognition patterns which might include variable bytes. Such options usually add considerable overhead in terms of time and thus make the scanner slower.

The other approach to this problem is to develop a scanner using a meta language into which a range of variable descriptive characteristics of the virus can be defined - such methods must be sufficiently flexible and fast to describe and identify each such virus. This approach may possibly cope with 10,000 viruses, beyond which the programs will become too slow and cumbersome to be of practical use. Before such saturation occurs, it is probable that viruses using self-modifying encryption and sparse infection techniques will have appeared which will not be detectable by virus-specific measures (see 1260 Revisited, *VB*, April 1990 p.10).

Area and File-Specific Search

So, the ideal scanning program must be both accurate and fast. Accurate information about virus code can limit the search to those files *known* to be at risk and thereby improve the scanning speed. Such information can further limit the search to the areas of files where virus code might be found. Adding such information into the list of recognition patterns turns them into a true 'signature' of each virus and will certainly improve the efficiency of a scanner.

Stealth Viruses

However, there is one major obstacle which **must** be considered. Certain memory-resident viruses have built-in self protection mechanisms designed to detect any attempt to locate them in memory or on disk. This 'counter-detection' will then prevent recognition by re-directing any search to a 'clean' file (or disk) area. In the case of 'stealth' viruses such as Dark Avenger, if the scanning program is not 'aware' of their existence in memory, not only will it not find infected files, but the process of opening, reading and closing files all over the disk may be used by the virus as a vehicle for spreading the infection further. Thus it is of paramount importance that the system is reliably clean before scanning commences.

It is theoretically possible to detect virus code in memory and to ‘unhook’ it from the operating system, thus rendering it inoperative. Unfortunately, this operation requires highly detailed knowledge of the virus and possibly different strains which may function slightly differently. The fool-proof and therefore prescribed way to detect stealth viruses is to reboot the machine from a clean write-protected system disk before beginning the scan. Knowing that there will always be users who will ignore such advice, the scanner should ensure security by searching memory for signs of known resident viruses and then, if any are found, abort the scan until the user reboots the machine from a clean system disk.

An alternative approach is to assume initially that the machine *is* infected and then write the scanner in such a way as to avoid the various intercept routines which viruses may install. This is a direct access method which can be most successful but requires a much higher degree of programming skill since the scanning program will contain code to access only the ROM BIOS of the machine and manipulate it according to the requirements of finding and scanning files. Thus in this case, the files are not ‘opened’ and ‘read’. The process consists of finding the whereabouts of the various sections of each file and then collecting them directly using only the ROM disk access routines.

Unfortunately, as the innards of machines are explored, problems of incompatibility and divergence from the IBM PC standard become readily apparent and such techniques become less portable between processors. Another problem arises when using direct access, in that such access is not usually available across network links and scanners may be limited to checking local drives only.

Interface

The current crop of scanning programs vary widely in the interfaces that they offer to the users. Some use a menu driven system in which the user selects his requirements on a ‘point and shoot’ basis while others use a range of command line options. The interface and desired options are generally a matter of personal choice but should include such things as the option to search just a single directory, or even a single file. It should also be possible to search *all* files for *all* viruses lest some devious soul has renamed an infected file to try to avoid scanning protection.

Compression

If files are compressed using utilities such as *PKZIP* or *PKARC* the ability to unpack and search such archives might be extremely useful. Dynamic decompression programs, notably *LZEXE* (*VB*, June 1990, p.12), present particular difficulties as any compressed program infected with a virus will automatically execute upon decompression. Some scanning programs already check for files compressed in this way and alert the user should they be found. However, diagnosing the presence of compressed virus code is an impracticable proposition.

Updates and Distribution

This need for constant updating gives rise to the practical problem of distributing updates to the users. The best and most secure solution, albeit the most expensive one, is to mail updates regularly to all users, maybe on a monthly or bi-monthly basis. The problem with updates is the effort needed to distribute them. If a large organisation routinely installs scanning software on the hard disks of its computers (which is a questionable practice in the light of stealth viruses) installing updates could easily take several man-months of work.

As some anti-virus software producers do not offer this type of update service, or users may not be willing to pay for it, updates to the database may be distributed via services like *BIX*, *CIX* or *Compuserve*. Unfortunately, BBS software distribution is vulnerable to Trojanisation and other subversion - hence this month’s report of yet another Trojanised version of *McAfee Associates’ SCAN* program (see page 24.).

Subversion

Another problem with virus-specific programs is the ease with which they can be bypassed, simply by making some minor changes to an existing virus, which prevent it from being detected. This is easier if the search strings used are easily accessed, and not stored in encrypted form. Encrypted search patterns do not prevent viruses from being tampered with and patched, but do make this subversion more difficult.

Exponential Virus Development

Perhaps the most serious problem in the long run, is the rate at which new viruses arrive. The number of known PC virus variants has been increasing at an exponential rate during the past few years (effectively doubling every 9 months) and is expected to surpass 1,000 by 1992. The effects of this are twofold. First, the programs and the accompanying database will continue to grow in size, as new viruses are written. Secondly, it will be more and more difficult to keep the anti-virus programs up to date. This is difficult today, and will become practicably impossible if more than two new variants appear per day.

The first signs of confusion are becoming apparent with software manufacturers quoting a range of wholly disparate figures as to the number of ‘known’ virus samples. For example, a recent virus collection from the *NCSA* in the US, which purported to contain 1700 infected files, was found to contain only two samples previously unreported by *VB*!

STRAINS & FAMILIES

Fridrik Skulason

'Stealth' Viruses - An Overview

Stealth viruses were the subject of an article in *VB*, September 1990, which outlined some of the criteria by which a virus may be classified as 'stealthy'. This article will look at the problem from a slightly different perspective, in particular considering how effective anti-virus software is against stealth viruses, and how they can be detected.

A 'stealth' virus can be defined as a virus which attempts to avoid detection by hiding the virus code in the infected media while the virus is active in memory.

Stealth viruses are just as likely to be destructive as other viruses, but the major problem associated with them is their ability to hide - an infection may remain unnoticed for a considerable time, which increases the chance of infection spreading between computers.

Sophisticated stealth viruses are more difficult to write than ordinary viruses, and only a few of them have achieved a significant distribution - 4K (aka Frodo, 4096) is the only one currently 'at large'. This may change in the near future, as the source code to stealth viruses is in circulation on various Bulletin Board Systems around the world.

A significant change expected in the future is an increase in the relative number of viruses using stealth techniques. Currently only 3-5 % of known viruses use any stealth methods, but this number is expected to increase in the future.

Software Subversion

Stealth viruses pose a serious problem to some types of anti-virus software, in particular scanners and checksumming programs. For example, if the virus is active while an infected file is being examined by a checksumming program, only the original non-infected program will be seen, and the infection will not be detected.

Although it has been stated before, it should be mentioned once again - no matter how sophisticated the checksumming algorithm is, with a stealth virus active in memory, correct operation of a checksumming program cannot be guaranteed.

The easiest way to avoid this subversion is to boot the computer from a clean write-protected system diskette before the anti-virus program is executed, but this simple and foolproof precaution is often ignored. For assured integrity, checksums of the trusted executables should be stored on diskette and these should be compared periodically with the state of the executables on the fixed disk.

Some other types of anti-virus software may be effective against stealth viruses, **provided that the virus is not active in memory when they are run**. Virus-specific monitoring programs can stop known stealth viruses when the virus is copied or read into memory (see *VB*, May 1991, p.6), and generic monitors, such as *FluShot+* may also be effective, unless the virus accesses the operating system directly.

It is possible to detect most known stealth viruses by scanning memory for virus patterns, but this may not be not be effective against the next generation of stealth viruses.

How Do Stealth Viruses Work?

The first stealth virus was the old and well-known Brain virus, although it was not formally classified as such until a year ago. The stealth features of Brain are quite simple. It intercepts INT 13H and checks for any attempt to read logical sector 0 - the DOS Boot Sector. When an infected boot sector is read, the virus will discard it and read the original non-infected boot sector instead.

Although the virus can easily be detected in memory, it will not be found on infected diskettes while active, unless the scanner makes a jump into ROM to read the diskette, instead of issuing an INT 13H call.

Several other boot sector viruses, such as Azusa and E.D.V. use the same method, and being relatively easy to program, it is expected to become a common feature in the future.

Parasitic Viruses

Implementing stealth functions in a parasitic virus requires considerably more coding than in the case of a boot sector virus.

All parasitic stealth viruses which change the length of infected programs must make the length increase 'disappear' when the user issues a DIR command.

This is usually done by intercepting the INT 21H FindFirst and FindNext function calls. Directory entries for infected files are also changed in some subtle way, for example by setting the 'seconds' field of the time stamp to 60 or 62. If FindFirst or FindNext returns a file which is marked in this way, the virus subtracts its own length from the reported length of the file. This can produce strange results if a very small file is marked as infected, as the resulting file length will be negative. As DOS treats the length as an unsigned number, the file appears to be 4 Gigabytes or so in length.

While this technique is necessary in a stealth virus, it is by no means sufficient. A virus using only this technique, such as SVC 4.0, can easily be found by looking at the file, as Read operations are not intercepted. Alternatively, any virus which only intercepts FindFirst/FindNext can be found by comparing the value returned by LSEEK (EOF) to the length returned by

FindFirst/FindNext. Any mismatch is an indication of viral activity - either indicating a corrupted directory or the presence of the virus itself. The virus can naturally intercept the LSEEK function as well - Spanish Telecom is an example.

If a virus is to be classified as a 'stealth' virus, several other DOS functions must be intercepted as well. *The second condition which must be met is that any program reading from an infected file must not read the virus code, only the non-infected original program*.

One simple method is for the virus to intercept the 'Open' function and disinfect the file on disk whenever it is read, usually re-infecting it as it is closed. Although this method will normally work, it is useless if the infected file resides on a write-protected diskette. As a side effect, if the virus does not re-infect the files as they are closed, it is possible to disinfect the system just by scanning every file for viruses, even though the scanner will not find anything suspicious.

The alternative adopted by such a virus might involve intercepting most of the file-handling functions, and 'disinfecting' the file in memory while it is being read. This avoids the problem with write-protected media, but is considerably more complicated to implement.

It is also possible to implement something similar on a lower level. Instead of intercepting INT 21H, the virus could intercept INT 13H, the disk I/O interrupt. No such virus was known to exist last September, so this possibility was not discussed, but the INT13 virus now uses this method.

STEALTH & STEALTH-LIKE VIRUSES

The following boot sector viruses use the stealth methods described above, intercepting INT 13H operations intended to read infected boot sectors.

Azusa: Redirects attempts to read infected diskette boot sectors, but the infection of the Master Boot Sector is easily detectable. (*VB*, April 1991)

Brain: Redirects attempts to read infected diskette boot sectors.

E.D.V.: Redirects attempts to read infected boot sectors.

Evil Empire: Redirects attempts to read infected Master Boot sector. (*VB*, May 1991)

Joshi: Redirects attempts to read infected boot sectors. (*VB*, December 1990)

Spanish Telecom: Redirects attempts to read or write to infected boot sectors. (*VB*, January 1991)

The following parastic viruses fulfill both conditions to qualify as stealth viruses. They are among the most innovative from a technical point of view and most of them have already been analysed in *VB*.

4K (Frodo, 4096): The virus 'disinfects' programs as they are read. (*VB*, May 1990, November 1990)

Fish 6: Related to 4K and uses similar methods.

INT13: Intercepts INT 13H and redirects attempts to read from infected files. (*VB*, March 1991)

Number of the Beast: Changes to the length of file are concealed even when the virus is not active in memory. Redirects attempts to read from infected files. (*VB*, May 1990)

Tequila: (*VB*, June 1991)

Whale: While this virus uses stealth methods, it places a heavy load on system resources and is usually easy to discover when active. (*VB*, November 1990)

Zerohunt (Minnow): As the virus overwrites unused space in files, it does not change the length of infected files. It will attempt to 'disinfect' programs when they are opened for reading, but is easily detected on write-protected diskettes.

Many viruses are able to hide the increase in the length of infected files, by intercepting the FindFirst and FindNext functions. They are not considered to be stealth viruses however, as they make no attempt to hide the virus code when an infected file is read. Examples include:

3445

Diamond, 1024

Dir: The file infection routine is a part of the FindFirst/FindNext handler.

Eddie (Dark Avenger): The 2000 and 2100 byte variants use this method, but not the earlier 1800 byte variant. (*VB*, February 1990)

Eddie-2: (*VB*, June 1990)

MG

PcVrsDs: (*VB*, April 1991)

Spanish Telecom: Also intercepts the LSEEK function. (*VB*, January 1991)

SVC

Zero Bug

Finally, the Bulgarian '800' virus has been reported elsewhere as using stealth methods. It does not.

VIRUS ANALYSIS

Richard Jacobs

Tequila - A Cocktail of Viral Tricks

The Tequila virus was originally sent to *VB* under the guise ‘Yugoslavian Virus’, however the virus contains an encrypted text message declaring its Swiss origin. This message starts with the greeting:

```
'Welcome to T.TEQUILA’s latest production'
```

This message suggests that the writer of this virus has also written other, less sophisticated viruses. The text goes on to give a Swiss post office box number, through which the author, supposedly, can be contacted. (Since this report was compiled, the Tequila virus has been identified in the wild throughout Europe. The alleged authors of this virus were arrested by Swiss police on May 20th, see page 24. Ed.)

Self-Modifying Encryption

Tequila is a sophisticated, self-modifying, encrypted virus using techniques similar to those of Mark Washburn in his V2P6 virus (see *VB*, April 1991). It was expected that Washburn’s tactics would be adopted quickly by virus writers in their efforts to conceal their programs - this virus provides further and somewhat alarming evidence that this is the case. It should be noted that the encryption method used in this virus has not been copied from any of Washburn’s viruses - the author has developed a proprietary method. The virus is also the latest in the current trend of multi-partite viruses, infecting both EXE files and the Master Boot Sector of fixed disks. No hexadecimal search pattern can be extracted to identify infected programs but a search pattern can be located in infected boot sectors.

Trigger Routine

The virus contains a non-destructive memory-resident routine which conceals increases in the size of infected files and which triggers the virus’ only visible side-effect. Under certain conditions, depending on the date and number of files infected, the virus displays a crude on-screen Mandelbrot (fractal) set. The virus then prompts the user to execute an INT 21H function, which displays a text message giving the name T. Tequila and the Swiss P.O. box number. The message continues with the text:

```
'Loving thoughts to L.I.N.D.A. BEER and TEQUILA
forever !'
```

When an infected file is run the virus decrypts itself in memory, reads the Master Boot Sector of the first fixed disk and checks it for previous infection. If the disk has not already been infected, this sector and the virus are written out to the last six sectors of the DOS partition. The size of the partition is reduced by 6 sectors so that the virus will not be overwritten. The virus boot code is copied over the copy of the Master Boot Sector in memory, preserving any error messages and the Partition Table. This new infected sector is written out to the normal Master Boot Sector, which closely resembles a normal boot sector in all but execution. This is one of several features to reduce the likelihood of detection. Finally, the virus returns control to the original program.

“It was expected that Mark Washburn’s tactics would be adopted quickly by virus writers in their efforts to conceal their programs.”

Operation

The majority of the virus only executes when a PC is booted from an infected disk. Should this happen, the virus reserves the top 3 Kbytes of base memory, loads itself into this area and transfers control to this copy of itself. Next the virus reads the original Master Boot Sector. The interrupt vectors for INT 1CH and INT 21H are then read and INT 1CH is redirected to point to a routine within the virus, before control is transferred to the original Master Boot Sector and the boot procedure is allowed to continue.

INT 1CH is the clock tick interrupt and is generated 18.2 times a second. This routine checks whether or not the INT 21H vector has changed (i.e. whether or not DOS has been loaded). If it has not, normal processing resumes. Once INT 21H has been altered, INT 1CH is returned to normal and INT 13H (BIOS disk services) and INT 21H (DOS functions) are redirected to routines within the virus.

The INT 13H intercept passes calls straight through to the normal routine unless an attempt is made to read, or write to, the Master Boot Sector of the first fixed disk. In this case the call is diverted to the original Master Boot Sector, thus concealing the presence of the virus.

The INT 21H routine has four different functions. The first of these is a simple check to find out whether the virus is already memory-resident. The second intercepts any FindFirst, or FindNext, file calls. Both ASCII and FCB calls are intercepted. Calls to ascertain the length of files are subverted. If a file has its seconds field set to 62, then the length of the virus is subtracted from the file length. Again, this is another stealth feature to render the virus invisible to the operator.

The third routine handles the screen display, when triggered.

The final INT 21H function intercepted by the virus is function 4BH. This is the DOS Load & Execute call and is the normal way of loading programs to be executed. First, the virus checks the name of the file. If the name contains “SC” or “V”, then the file will not be infected. This is probably a crude attempt to avoid infecting **V**irus **SC**anning software. Otherwise INT 24H (Critical Error handler) is disabled and the file attributes are read and saved, before being cleared. Next the file is opened and the date and time it was last written to are saved. The first 28 bytes of the file are then read. The file will only be infected if it is an EXE file.

62 Seconds Stamp

Although Tequila uses the ubiquitous 62 seconds stamp to identify infected files in the FindFirst/Next file routines, it does not rely on this for determining whether or not to infect files. Instead the checksum stored in the file header is read. This checksum is calculated by the linker when the program is first created, but is not used by any current versions of DOS. When the virus infects a file, it overwrites the checksum with a word taken at random from the virus decryption routine.

In order to check for a previous infection, the virus scans its own decryption routine for this checksum. If it finds the checksum with the decryption routine, it assumes the file is already infected. The virus then sets the seconds field of the file time stamp to 62, closes the file and restores INT 24H and the file attributes. If there is no match with the checksum, the virus assumes the file has not already been infected.

Variable Decryption

The encryption method used in this virus is somewhat more advanced than that used in the 1260 (*VB*, March 1990, p.12) and V2P2 viruses but less flexible than that of V2P6 (*VB*, April 1991, pp.18-20).

Once an uninfected file has been identified, the virus adjusts the values stored in the EXE file header to provide a stack for the virus, to set up the virus entry point and to allow for the increased length of the infected file.

The virus then generates a new copy of itself and attaches it to the end of the original file. This process involves generating a new decryption routine. First, the old decryption routine is overwritten with a random word obtained from the system time. This random number is then used to build up the decryption routine. The routine is generated as several modules which can be put together in any combination. This provides a much more flexible set of routines than seen in the Whale virus (see *VB*, November 1990, pp. 17-20).

Each module can also contain randomly placed instructions that have no effect on the functionality of the virus, but further increase the differences between each copy of it. Although Tequila always uses the same registers for the decryption routine, it does not always use each register for the same purpose - for example, SI and BX can be interchanged.

Likewise, the virus does not always use the same instructions in the decryption routine. For instance the actual encryption can be performed either by an XOR or by adding a key to the value of each byte in the virus.

One unusual feature of this virus is that the decryption routine itself is used as the *key* for decryption. Once the new decryption routine has been created, the virus is encrypted in memory, written to the end of the file and then decrypted in memory again. The file time stamp is then set to 62 seconds and one word of the decryption routine is written to the checksum in the file header, for identifying infected files. The new file header is then written out to the file. Finally, the file date and time are restored along with INT 24H and the file attributes. Control is then returned to DOS, which will load and execute the program normally.

Detection and Removal

Removal of this virus from infected files is by the normal procedure of deleting the files and replacing them with write-protected backup copies of the master software.

Disinfection of the Master Boot Sector sector is less straightforward unless you keep a backup of the Master Boot Sector on floppy disk. If such a backup is available removal of this (and all other currently known boot sector viruses) is simple! Otherwise some work with a disk editor is required to locate and replace the original Master Boot Sector. In both cases reboot the PC from a clean, write-protected system floppy disk before starting. **Unless all of the infected program files are replaced, the boot sector will be reinfected immediately an infected program is run**.

Detection of the virus in the Master Boot Sector can be determined using a straightforward hexadecimal pattern. However, no pattern can be used to detect the virus in program files. The use of professional virus scanning software capable of algorithmic detection is essential if the following pattern is located in the Master Boot Sector.

```
B82A 0250 B805 028B 0E30 7C41 8B16 327C
```

Conclusions

The Tequila virus displays a veritable cocktail of programming techniques designed to increase its chances of spreading undetected. The use of various stealth techniques, self-modifying encryption and multi-partite characteristics infecting both programs and boot sectors places it in the ‘hybrid’ category.

Ironically, despite all the author’s efforts to conceal this program, it is comparatively easy to develop reliable detection routines for inclusion in scanning software.

PRODUCT UPDATE

Dr. Keith Jackson

Dr. Solomon's Anti-Virus Toolkit - A Return Visit

This is a reassessment of *Dr. Solomon's Anti-Virus Toolkit*, as it was originally reviewed in the first ever issue of *Virus Bulletin* almost two years ago (July 1989).

My intention is to see how the available facilities in the *Toolkit* have advanced in the intervening months and to assess how useful the product now is. The *Toolkit* contains anti-virus programs which fit into the following categories:

➤ Checking for the presence/effects of viruses

➤ Providing protection against viruses

➤ Removing viruses after infection has taken place

➤ Inspection of disks, and other miscellaneous utilities

The effects of an extant virus infection are best dealt with by restoring infected files from clean backups and the *Toolkit* documentation acknowledges that inoculating files against virus infection, and/or removing a virus after it has infected parts of a disk, are both secondary lines of defence. However, they can prove useful if backups are unavailable. The first of the four categories described above contains the most important parts of the *Toolkit* and this review will concentrate on these components.

Documentation

Two years ago, the *Toolkit* came with a 68-page manual which although brief, nevertheless provided a good introduction to Trojan horses, logic bombs, time bombs and viruses.

The documentation has since grown much larger. A hefty A5 manual includes an introduction to viruses, sections explaining how to use the various components of the *Toolkit*, detailed explanations of the more common viruses (95 in total), plans of action and other useful advice and information.

For some unfathomable reason the individual pages are not numbered. The manual is about 15mm thick, and at a guess comprises about 250 pages. Given that the pages are not numbered, production of an index would seem to be difficult and it is no surprise that the manual does not have one.

The *Toolkit* contains many small utilities, lots of descriptions of individual viruses, and several chapters of general advice - if ever a manual cried out for an index this is it!

FINDVIRUS

The virus scanning component of the *Toolkit* (*FINDVIRUS*) was discussed in the comparative review of virus scanning programs published in the April 1991 issue of *VB*. It would be pointless to repeat this exercise here. However, in passing it should be noted that the speed at which disks can be scanned has improved enormously since the original *Toolkit* review. The *Toolkit* used to be able to scan a hard disk at 22 Kbytes per second. My original review complained that this speed was unbearably slow. Even though exact comparisons are not possible (I no longer have the computer used for the original review), the latest version of the *Toolkit* scans my hard disk at 778 Kbytes per second - a quantum improvement.

When the scanner from the original *Toolkit* was executed, it stated that it would scan for the Brain, Italian, Pentagon Yale, Stoned, 405, 648, 1168, 1701, 1704 and 1813 viruses. Oh how we've moved on in just two years to the state that there are (depending upon whose figures you believe) somewhere between 250 and 500 unique PC viruses known to exist [1]. Another change is that Alan Solomon used to use numeric nomenclature for all viruses which infected files. This scheme has been dropped over the intervening period of time.

Checksumming Algorithm

In my original review, a major gripe was with the algorithm used to calculate checksums for files (and/or areas of disk/memory). These checksums can be verified to ensure that the file has not been altered in any way (either by a virus or for more prosaic reasons).

The algorithm used by the original *Toolkit* was very simplistic and could in no way be described as having any cryptographic merit. To prevent reverse-engineering of the checksums by a clever virus, such an algorithm must be cryptographically strong. If it is not, it may be possible to deduce the algorithm from inspecting checksums and/or files, and write a virus which can alter a file, recalculate the correct checksum, and replace the new (correct) checksum. The original algorithm was trivial in the extreme and I recommended that the checksum program provided with the *Toolkit* should not be used until it had been given some cryptographic strength. The algorithm provided with the current version of the *Toolkit* is different from the original algorithm, but in seemingly minor ways, and after cursory investigation I would be hard pushed to describe it as being of any greater cryptographic complexity. It still seems vulnerable to reverse-engineering.

My original comment on the algorithm used by the *Toolkit* to calculate checksums was 'It is trivial in the extreme, don't use it'. I can see nothing to change my conclusion in the latest version of the *Toolkit*. The reason why an algorithm of greater cryptographic complexity has not been used is probably to maintain the fast execution speed of the checksumming program provided with the *Toolkit* (*CHKVIRUS*). There is a balance that must be struck between cryptographic strength

[1] If all variants are counted, the number approaches 700. Tech Ed.

(which is essential) and speed of execution (which is extremely desirable). The *Toolkit* has still **not** got this balance correct and given that this product now sells worldwide in large numbers, a virus author may well latch on to this weakness at some future date and exploit it.

There are other products on the market offering checksum schemes using algorithms that are known to be cryptographically strong and I am on record as stating that in nearly all cases they are too slow to use routinely. There is, therefore, no simple solution to the speed of execution versus cryptographic strength equation. In my opinion the algorithm used by *CHKVIRUS* has got this equation wrong. [2]

Improved Front-End

It is a shame that the checksumming component has not improved commensurate to the many other vast improvements incorporated in the current version of *CHKVIRUS*. Given the integration offered by the front-end software this is now an easy-to-use utility in which the user just specifies the details of the disk(s) to be checksummed, and whether *CHKVIRUS* execution should create a list of files to be checksummed, checksum this list of files, or verify a set of existing checksums. A program is also provided which just does a quick check of each file by looking for changes to the Master Boot Sector active DOS Boot Sector, file size, date and/or time. All of the checksumming facilities are now very easy to use (this was not previously the case), and the individual programs can be executed directly from a batch file.

"I used to think that the Toolkit was most suitable for virus researchers rather than for a user of a computer infected with a virus. . . I've now changed my mind. . ."

Unlike the original program, most of the utilities provided to inoculate against viruses, remove the effects of viruses, and the various other miscellaneous utilities now have a reasonable user-interface and error reporting. Their integration through a consistent front-end software package (see immediately below) has helped matters enormously.

I used to think that the *Toolkit* was most suitable for virus researchers rather than for a *user* of a computer infected by a virus. I've now changed my mind on this matter. From being a mere collection of disparate programs, the new front-end has added a cohesion which was formerly lacking and given the components of the *Toolkit* a common feel. All the facilities of the *Toolkit* are available in a drop-down menu format, coupled with mouse operation and hypertext on-line help. Although I may not require this front-end software to use the *Toolkit*, I would conjecture that many users do.

Virus Guard

The *Toolkit* came with a copy of a new product called *Virus Guard*. The disk label says 'Put the disk in and type Help'. If you follow these instructions nothing happens, presumably because there is no executable file called HELP present on the disk. No documentation was provided with *Virus Guard*, just a file detailing the current list of known viruses. This is a return to the bitty standard of the original *Toolkit* of two years ago. From other sources (mainly the *CIX* conferencing system) I gather that *Virus Guard* is a utility that resides in memory and monitors the PC in real-time for virus activity. Given the problems that **always** occur with interaction between memory-resident utilities, I'm hanged if I'm going to install this on my PC without any documentation. *VB* will no doubt be back to examine *Virus Guard* when it has matured somewhat.

Conclusions

Far be it for me to pretend that front-end software is the most important part of any product. However if it is included, care is needed to ensure that it is both consistent and correct. In my opinion this has now been achieved with exception of a few minor inconsistencies (see above), and the glaring lack of page numbering and an index in the documentation.

With the exception of the algorithm used for checksum calculation, the broad principles of the product are sound, and if you intend to maintain barriers against virus infection, cure virus infections, or if you want to learn nitty gritty details about viruses, then the *Toolkit* provides value for money.

During the past two years the *Toolkit* has become a coherent product, whereas it used to be merely a collection of small programs given a grandiose title. The front-end package provided with the *Toolkit* is excellent and if I can keep the copy provided for review I intend to use it myself in future.

Product Details

Developer and Vendor: *S&S International Ltd.*, Berkley Court, Mill Street, Berkhampstead, Herts. HP4 2HB, UK. Tel 0442 877877, Fax 0442 877882.

US Distributor: *Ontrack Computer Systems*, 6321 Bury Drive, 15-19, Eden Prairie, MN 55346, USA. Tel 612 937 1107.

Availability: IBM PC/XT/AT, PS/2, or any close compatible running MS-DOS or PC-DOS.

Version Evaluated: v4.32

Serial Number: TT34219

Price: £99/$279.95 (includes 4 quarterly upgrades)

[2] See Technical Notes, 'Cryptographically Secure?', p.3. Ed.

PRODUCT REVIEW

Mark Hamilton

Central Point Anti-Virus

Until *Symantec* announced its *Norton Anti-Virus* product last September following the buyout of *Peter Norton Computing*, the PC anti-virus marketplace was the preserve of the smaller specialised manufacturer. Now rival utilities company *Central Point Software* is making its bid for a slice of the action.

Unlike *Symantec*, which has been active in the anti-virus field for over two years with its Macintosh product, *Central Point* has opted to buy in an exisiting package. For the core technology, *Central Point* went to Israel and contracted with *Carmel Software* whose *Turbo Anti-Virus* was the subject of a recent review (see *VB*, February 1991, pp. 18-19).

The package has been customised by the addition of *Central Point's* new screen library (which is also being used in its awaited *PC Tools* version 7) while an aggressive marketing campaign has centred on the familiar numbers game - "Protects against over 500 known plus unknown viruses" - and other hyperbole.

I conducted this review according to the protocol published in the April 1991 edition of *VB*. Readers should refer to that issue for full details of the test-set used, the testing criteria and details of the hardware environment.

The Carmel Connection

If proof were required that this software is a reincarnation of Turbo Anti-Virus, the following text is contained within the main program:

```
Turbo Anti-Virus(tm) Copyright 1988, 89, 90 This
software was created by Yuval Sherman & Eli Shapira
All rights reserved world wide to: CARMEL Software
Engineering Hamachshev LTD. Hahistradrut Av. 20
Haifa, Israel POB 25055. TEL: 972-4-416976, FAX:
972-4- 416979 Have a nice day!!!
```

With the exception of the *Microsoft Windows* specific support files, all the original *Turbo Anti-Virus* programs have their equivalent counterparts in *Central Point's* offering.

Documentation

There is a single over-sized A5 perfect-bound book that serves as both a user-manual and as the product's reference material. Spot colour, icons, margin notes and photographic screen representations give the impression of a professionally produced work. The manual's first seven chapters provide clear instructions on the use of the product. Chapter 8 details most of the viruses that *Central Point Anti-Virus* can detect, while the last three chapters provide technical reference.

Nowhere in the manual, do the authors emphasise the importance of regular verified backups as being the best protection against data loss - and this from a company which is a major player in the backup software market! Nor is there any precaution to boot your PC from a clean write-protected system disk. Significantly, the documentation fails to mention that you can run the product from a floppy disk; conversely, it details instructions on installation to a local hard drive.

A number of inaccuracies appear and the non-standard terminology used is confusing. *Central Point* is totally out of step with convention in its classification of virus types. For instance, it classifies overwriting viruses as 'Trojans'!

The Software

The software is delivered on two 5.25 inch and one 3.5 inch disks and its 25 files occupy some 653,369 bytes of disk space. In addition to all the program files, there is a README which details changes to the program since the documentation was printed, as well as a text file - VIRLIST.DOC - which explains how you can obtain a list of all the viruses that the product claims to detect.

The software consists of four main components:

BOOTSAFE which saves and restores images of boot sectors and partition tables to and from disk files. This program is functionally the same as its *Turbo Anti-Virus* counterpart.

CPAV is the main program which incorporates a scanner, an inoculation facility, a checksum generator and checker, and file disinfector. These are similar facilities to those in TNTVIRUS, *Turbo Anti-Virus'* main program.

VSAFE is a configurable memory resident-monitor which claims to detect viral behaviour as well as scanning executable files for known viruses. The documentation states that it requires 22 Kb of memory - the true figure was closer to 24.6 Kb. This program is functionally equivalent to *Turbo Anti-Virus'* TSAFE program.

VWATCH is a memory-resident scanner which claims to check executable files for the presence of known viruses as they are opened for any reason. Again, the documentation claims that this utility requires less than 8 Kb of memory - it in fact needs 11 Kb. VWATCH is equivalent to the *Turbo Anti-Virus* DEFENDER program.

VSAFE and VWATCH are provided both as command line invocable TSR programs and as device drivers, but you can only have one utility loaded at any one time - not that you'd require both since one is a subset of the other. In addition, there are a number of support files, most of which allow the above-mentioned programs to communicate with *Windows 3.*

Installation

The INSTALL program installs the software to a directory of your choice on (one of) your hard disk(s); this defaults to C:\CPAV unless you change it. Prior to installing the software, the installation program executes a scan of the hard drive. This is actually quite a good idea, but it would have been better had the installation program reminded you that you should boot from a clean, write-protected system disk immediately before installing the software.

The installation process will also alter the PC's start-up configuration if you elect to have either VSAFE or VWATCH loaded at bootup time. It does this by modifying either CONFIG.SYS or AUTOEXEC.BAT having previously saved the contents of the original files.

Microsoft Windows support is provided and the installation program will attempt to locate one of the *Windows* initialisation files - WIN.INI. However, it failed to locate this file on my system. Also, you have to edit another *Windows* initialisation file, SYSTEM.INI, to add the name of a driver in the '[386Enh]' section, if you intend to run *Windows* in 386 Enhanced Mode. Unfortunately, the installation program does not do this for you. Hardly 'user-friendly'.

"There are very few programs in the 'real world' which can be inoculated in this way - too few for it to be a worthwhile strategy"

Resident Software Performance

Using the 4K virus, I tested both VWATCH and VSAFE to see whether they would allow me to copy infected files and also whether they would allow an infected program to be executed. VSAFE, the larger of the two, disallowed both operations but, VWATCH, while preventing an infected file from executing, allowed me to make copies of it. These monitors offer no choice in the matter; if they detect what they think is a virus, the operation is disallowed.

The PC's performance did not degrade unduly with VWATCH resident, but system performance worsened noticeably with VSAFE in memory. Unless *Central Point* can reduce the memory footprint of these two utilities, I fear few users will tolerate this overhead, particularly on DOS 4 machines operating on a network. One also has to remember that these performance overheads will worsen as more and more viruses are added to the monitors' detection capabilities.

Interface

On suitably equipped PCs (those with EGA or VGA displays), CPAV redefines the character set to provide a "fake graphics" appearance. This extends to iconic representations of file types and sub-directories - it differentiates iconically between executable and non-executable files - as well as various 'windows- type' buttons, arrows and sliders.

There are two basic operating modes, Express - which provides limited functionality - and Full. The Express menu option to revert to Full mode can be password-protected. The Full Menu mode has menu options called Scan, Options, Configure and Help, while the Express Mode has five buttons labelled Detect, Detect & Clean, Select New Drive, Full Menus and Exit. Personally, I found the screen display of the Full mode to be very 'busy'. Also, the full menus are none-too-logically arranged. For example, the sub-menu command to change the drive to be scanned is not found in the Scan Menu, as one might expect. Rather it is to be found under the Configure Menu and is entitled 'Change Work Drive'. Inexplicably, you'll find the Exit (to DOS) command under the Scan menu (!?!).

Scanner Performance

In its 'Turbo' mode of operation, its scan speed is very respectable and in both modes its detection rating is good when one considers that this is a new (?) product (see table on page 22).

In its 'Turbo' mode, this product missed nine instances of viral infection which *were* detected when the product commenced a 'Secure' mode search. Unfortunately, in this secure mode, its disk scanning speed is excrutiatingly slow.

Checksumming

In addition to virus-specific scanning, the CPAV program can optionally create checksum files. If the checksumming option is invoked, CPAV creates a file called CHKLIST.CPS in each sub-directory, in which there is an individual entry for every file whose extension is .COM, .EXE, .SYS, .BIN, .DRV, .OV? or .DLL. By enabling the option "Verify Integrity" (not in the "Configure" menu), the scanner calculates and compares each file's checksum with that previously recorded.

For this to be effective, a checksum of the *entire* file must be taken. This does not appear to be the case here, since, in "Turbo" mode, there was no difference in its speed regardless of whether the checksum checking options were On or Off. Nowhere is the user instructed to maintain checksums of trusted executables on diskette - running the program on hard disk and only running it from the fixed drive is prone to subversion by stealth viruses. No indication of the strength of the algorithm used is provided, which further reduces one's confidence in the security provided. This checksumming protection, might, therefore, be described as less than optimal!

Inoculation

Now the thorny subject of inoculation. *Central Point* offers file protection by appending self-checking code, in the same way as an appending virus, to the program files being protected (this is sometimes irreverently referred to as the ‘condom’ method). The idea is that if the self-checking code detects program modifications, you are given the option of repairing the file, allowing the program to continue, or preventing its execution.

Unfortunately, it is impossible to inoculate programs with inbuilt overlays (or other information at the end of the file), COM files larger than 63 Kb, files with built-in integrity checking, *Windows* or OS/2 files - and this list is by no means exhaustive. In short, there are very few program files in the ‘real world’ of end-user computing which can be inoculated in this way - too few for it to be a worthwhile strategy. In fact this method can be summarily dismissed as a means of generic defence and could well prove counter-productive by engendering a false sense of security.

Technical Support

At its offices near Heathrow, UK, *Central Point* has a European headquarters from which it aims to provide technical support for the UK and Europe. The problem with *Central Point’s* approach is that its technical support staff have to be conversant with all the products that the company markets. As a result the company currently has no ‘in house’ virus expertise.

I am bound to report that the product’s two developers are currently in Israel and are not due back at *Central Point’s* Oregon, USA, headquarters until later this summer. The company has also decided to centralise all its research endeavours in the US which means that there could be delays in updating the product.

The company advertises a 24-hour *Virus Hotline* service. This is “manned” by an answering machine which simply states that you can obtain signature file update disks at a cost of £7.95 and also gives you a bulletin board number to call from which you can also obtain the updates.

Conclusion

The product looks and feels reasonably accomplished which gives the initial impression that it is well designed and engineered. However, upon examination, many of its components appear to be fundamentally flawed. There is nothing wrong with a large software company promoting and selling others’ inventions. However, such a company - in this case, *Central Point* - in order to have any impact ought to provide support and technical expertise at least as good as the existing and more specialised industry leaders. There are currently no indications that this is the case. However, the jury will wait a while longer before returning a final verdict.

CENTRAL POINT ANTI-VIRUS

Product *Central Point Anti-Virus* v.1.00

Distributor: *Central Point Software*, 15220 N W, Greenbrier Parkway, Suite 200, Beaverton, Oregon 97006, USA. Tel 503 690 8090, Fax 503 690 8083

Central Point Software, 3 Furzeground Way, Stockley Park, Uxbridge, Middlesex UB11 1DA, UK. Tel 081 848 1414, Fax 081 569 1017.

Price	Product	£115.00
	Quarterly Updates	£19.50 each
	Signature Files	£7.95 each

Standard Memory Check	Yes
Upper Memory Check	No
Network Aware	Yes
Single File Check	Yes
Definition Format	Proprietary
Virus Removal	Disinfection
Access to *VB* Test-Set	No
User Upgradeable	Only using *Central Point* detection data
Resident Scanner/Monitor	Yes

Scanning Speeds

Hard Disk - ‘Turbo’	2 mins 13 secs
‘Secure’	2 hours 39 secs
Floppy Disk - ‘Turbo’	0 mins 5 secs
‘Secure’	4 mins 34 secs

Scanner Accuracy

Parasitic - ‘Turbo’	279 out of 306
‘Secure’	288 out of 306
Boot Sector - ‘Turbo’	7 out of 7
‘Secure’	7 out of 7

Accuracy Percentages

‘Turbo’	91.37 %
‘Secure’	94.42 %

For information about the entries in this table refer to the evaluation protocol published in *VB*, pp.6-7 April 1991.

BOOK REVIEW

A Short Course on Computer Viruses

The author of this book is one Dr. Frederick B. Cohen and if you're interested in computer viruses its possible you'll have heard of him. Dr. Cohen has written numerous learned papers on the subject of viruses (many of which contain incomprehensible algebra) and he's spoken about the subject to quite a few people (10,000 in all by his own estimation). He is also, of course, the man who conducted the first virus 'experiments' when in 1983 he wrote a 200 line C program which spread like wildfire on his university's Unix network. When he published the results of his experiments in 1984, many people were dismissive - the reactions ranged from disbelief to contempt. Attitudes change, however, and Dr. Cohen has now logged thousands of air miles and lecture hours explaining the virus phenomenon and how best to deal with it. *A Short Course on Computer Viruses* presents his findings and conclusions as well as some good jokes.

The consistent thread running through this book is the author's assertion that all computers, however configured, are inherently vulnerable to viral infection. The author takes great pleasure in demonstrating how viruses have completely undermined conventional computer security wisdom which emphasises *confidentiality* and not *integrity*. Even high security architectures based on the *Bell-LaPadula* model (a hierarchical system in vogue with the US military whereby the user cannot *read* from 'higher' security levels and can't *write* to 'lower' security levels) are wide open to viral and Trojan attack. The reason? Simply that there are no controls to prevent higher level users from executing lower level user's programs - suitably seductive software will tempt people regardless of their assigned security clearance! Moreover viruses are persistent; unlike conventional security problems where trap-doors and holes can be identified and closed, viruses don't go away - you can clean out a system only to find that the poison has re-entered via backups or infected media. Worse still, because computer architectures are upwardly compatible, viruses remain functional even as operating systems evolve - as Cohen says: 'If you wrote a computer virus for the IBM 360 in 1965, chances are it would run on every IBM compatible mainframe computer today.'

Cohen isn't coy and pulls no punches when it comes to describing what viruses can do and what they may do in the future. A catalogue of horrors is presented in an unashamedly upfront manner. Odious examples include viruses which alter decimal points and spreadsheet cells, viruses which interfere with the backup process, viruses which subvert system security settings, viruses which paralyse networks, viruses which destroy user confidence, viruses which exploit covert channels and even an 'executive error virus' specifically designed to get your business rival fired from his job!

Anecdotes are liberally peppered throughout the book, the most extraordinary of which concerns Ken Thompson, co-author of Unix, who is rumoured to have done something horribly devious with his C compiler - the ramifications of which don't even bear contemplation!

The second part of the book is devoted to defences, specifically those which work and those which don't. Cohen pushes his 'integrity shell' theories down the reader's throat while summarily condemning most of the defences which are actually in widespread use. He has some time for cryptographic checksumming techniques but is dismissive of scanning software which he variously describes as 'expensive', 'ineffective' and engendering 'a false sense of security'. (This despite the fact that scanners have probably detected more real world viruses than all the other methods combined.) In fact, Dr. Cohen (a man known for taking a longer term view of the world's problems) is right in concluding that scanning software must fall into demise - he even produces a cost analysis to prove the point. Some of the statistics provided to support his assertions are exaggerated - most scanners which conduct a byte-by-byte search of files do not take 5 minutes per megabyte as suggested.

As might be expected of a man who has devoted the best part of a decade to analysing the virus problem, the author comes to the inevitable conclusion that inoculation and static analysis programs which search for suspicious instructions or data areas should be consigned to the dustbin. Perhaps his most contentious points concern backups; everyone is agreed that backups are prone to failure (which is why they should be verified) and the data contained on backups can be corrupted by viruses (or sunlight, or spilt coffee etc.) but is Cohen *really* suggesting that they are a waste of time as a defence? In the real world backups offer the *only* opportunity for recovery from a range of disasters and to infer otherwise seems wilfully perverse. Cohen also labours the point that viruses reinfect systems from backups - it would be more helpful if he proffered advice to backup *data* and not executables!

These objections apart, and discounting some of the more contrived mathematical gymnastics which appear spasmodically throughout the text, this book is fun to read (a rarity in this field) and seemingly near faultless in the majority of its conclusions. Considering the author's considerable achievements, the book is also written with admirable modesty. It is important to know what Dr. Cohen believes (even if you don't agree with the man) - he has, after all, dedicated himself to this subject far longer than the hundreds of self-proclaimed virus 'experts' who have crept out of the woodwork in recent years. Ultimately, Cohen has clout.

A Short Course on Computer Viruses (196 pp.)
Dr. Frederick B. Cohen
ISBN 1-878109-01-4
ASP Press, PO Box 81270, Pittsburgh, PA 15217, USA
Tel 412 422 4134, Fax 412 422 4135.

END-NOTES & NEWS

Virus Arrests. On 20th May 1991 Swiss police arrested two men aged 18 and 21 in the village of St. Hausen, Switzerland in connection with the development and distribution of the Tequila virus (see pages 16-17). The father of one of the men is a shareware dealer and the virus has been reported as having been widely distributed in games software. The Tequila virus has spread throughout Europe - notification of the first UK infection was received on 20th May.

The **trial of Dr. Joseph Lewis Popp** for his alleged involvement with the AIDS Information Diskette Trojan has been set for November 11th 1991 and will take place at Southwark Crown Court, London.

McAfee Associates have issued a warning that ***VIRUSCAN* version 78 contains a Trojan horse**. The hacked version of *SCAN* has been uploaded to BBSs in Michigan, USA under the filename SCANV78.ZIP. The authentication methods used by *SCAN* have been subverted. A file named TB1.COM contains the Whale virus. Enquiries to Aryeh Goretsky, *McAfee Associates*, USA. Tel 408 988 3832 or e-mail aryehg@tacom-emh1.army.mil.

The *Common Cold & Nasal Research Centre* at the *University College of Wales Cardiff* is used to common bugs but has recently had difficulty in combating the WDEF A virus on its Macintosh computer network. The infection, reported by *Computer Weekly*, disrupted ongoing research until the *Disinfectant* anti-viral utility was wheeled into action. Mr. John Norstad, author of *Disinfectant* and specialist in Macintosh viruses, will be speaking at the *VB Conference* in September.

The US *National Computer Security Association* (*NCSA*) is to issue a number of '**consumer reports' on anti-virus utilities** with publication schedules planned through to March 1992. The reports will cover scanners, CRC and cryptographic checksumming programs, TSR monitors, hardware solutions, and recovery tools. Tel *NCSA* (USA) 202 364 8252.

Breaking with the tradition of discretion, *Central Point Software* has announced that a UK division of ***Citibank* has ordered *Central Point Anti-Virus*** for its 1,000 strong network of PCs in the UK. The *VB* review of this product appears on pages 20-22 of this edition.

RG Software Systems, Inc. has released **version 6.00 of Vi-Spy**. The program provides a RAM resident option to scan diskettes automatically upon their first access in a drive. Version 6.00 includes memory checking, checksumming and monitoring to warn of unauthorised TSR activity. Tel *RG Software* (USA) 602 423 8000.

Sophos Ltd continue a series of introductory and advanced **Computer Virus Workshops** in which participants can gain 'hands on' experience with live computer viruses. The next available workshops take place in Oxford on the 11th and 12th July 1991. Contact Karen Richardson, *Sophos* UK, Tel 0235 559933.

S&S Ltd is holding a **Virus Seminar for Managers** in Great Missendon, Buckinghamshire on 26th June 1991. Contact Janet Rudkin, *S&S* UK, Tel 0442 877877.

Quote of the Month

...From the 'Bedtime Reading' section of *Bates Associates' VIS Utilities Manual*:

BIMBOWARE = 'delightful packaging and fun to play with, but no good for a serious relationship.'

VIRUS BULLETIN

Subscription price for 1 year (12 issues) including delivery:

USA (first class airmail) US$350, Rest of the World (first class airmail) £195

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