ISSN 0956-9979 SEPTEMBER 1994

# VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL



## IN THIS ISSUE:

• **KAOS reigns.** A virus has been released on the *Internet*: how great are the risks? For an analysis of the virus, see p.8; for an analysis of the risks, turn to p.6.

• **Comparatively speaking.** *VB* has always advised against the use of virus removal software. However, it has become an integral part of many anti-virus software packages. How effective is this technique? See p.11.

• **Fire, fire!** *Norman Data Defense Systems* has released a server-based anti-virus package: how does it compare to the DOS version of this software, *Norman Virus Control*? Product Review 2 has all the answers.

## CONTENTS

# EDITORIAL

## To Detect or Not to Detect…

When is not a virus a virus... or, put more simply, are there occasions when a virus scanner should label a file as infected, when it is not? The answer to this question seems obvious: virus scanners should detect files which are infected by a fully-functioning virus. However, is this what one actually wants a 'real world' scanner to do?

Consider the following experiment. An innocuous file is encrypted using the Mutation Engine. The file is then virus-checked by a number of different packages. Each scanner will give one of three results. Firstly, the scanner might pass the file as clean. Secondly, the scanner might alert the user to the presence of the Mutation Engine code. Finally, the scanner may display a message informing the user that the file is infected with a virus. Clearly, the most useful statement is the second one. However, almost all scanners fall into either the first or the last category - the second statement, though factually accurate, is not a great deal of use to the user: can he run the file or not? The only way to find out is to analyse it, a task which will be beyond the scope of the casual browser.

The experiment can be made still more complicated. Imagine a polymorphic virus which sometimes does not carry out its infection process correctly, adding a decryption routine to 'infected' files, but failing to add the encrypted virus code. Such a file will not replicate, and is therefore not a virus. However, there is a powerful argument that the user should be alerted to its presence.

The current trend in the industry is a move towards precise or exact virus identification. In many ways, this is highly beneficial, as disinfection can be carried out more accurately, and the user is left with a better picture of what potential effects the virus may have had on his system. With the advent of widely available polymorphic engines, the only way to identify exactly which virus a file is infected with is to decrypt the file and examine the code 'hidden' by the encryption. If this code is identified as a virus, then the file is deemed to be infected. If it is not, the file is clean. This process is slow on infected files, but very quick on clean ones.

*" Surely a user would like to know that a particular program is wrapped in an MtE decryption routine? "*

All well and good... except when the program encounters a file which has a valid decryption routine attached to it, but random code encrypted within it. In such cases, some scanners will not label the file as infected. But is this action unnecessarily pedantic? Surely a user would like to know that a particular program is wrapped in an MtE decryption routine? This is critical in the case of the user who is attempting to clean up a system after an attack by a polymorphic virus. Here, the anti-virus software should identify all those files which contain the virus, or parts of it: that is, those files which have been altered.

Vendors will quite justifiably point out that change detection is exactly what a checksummer does. However, the use of checksummers is hardly widespread, and one feels that there is something unnecessarily pernickety about the failure of a scanner to alert the user to fragments of viruses left scattered across the disk. If files half-infected by a botched virus cannot be detected by the scanner, then it is time for a change of emphasis: the scanner and the checksummer should be combined in a way which is transparent to the user. The checksum information can then be used during a clean-up operation, to identify those executables which have been altered by the virus. It should be noted that during clean-up, whether an altered program is a virus or a dysfunctional attempt at infection is immaterial to the user - all that matters is getting the machine operational as quickly as possible. Some products already use this two-pronged attack, but few seem to make the marriage of the two techniques as harmonious as it could be.

Until the next generation of products is installed upon computers worldwide, the main line of defence against virus attack is the humble scanner. Here the question of 'to detect or not to detect' is still unanswered: some scanners can identify 'half-infected' files, and some cannot. So when is it acceptable to call something a virus when it is not? The current industry consensus on the matter is rather undecided, leaving those unlucky enough to be caught without a backup of their system blundering around in the dark. Anybody care for a light?

# NEWS

## Honecker: The Last Laugh

The latest computer virus story to hit newspaper headlines worldwide is the 'Honecker virus'. The virus triggers on 13 August (the anniversary of Erich Honecker's construction of the Berlin Wall), and displays a caricature of the late East German leader, complete with spectacles, on the screen.

This is followed by a rendition of the national anthem of the former German Democratic Republic, and a message announcing the destruction of programs 'by order of the Council of Ministers of the German Democratic Republic'. The next message reads: 'Honni's last revenge - I'll be back'. It then deletes the AUTOEXEC.BAT file.

Analysis of the 'virus' shows that the program should probably be regarded as a Trojan, as it is incapable of spreading onto floppy disk without actually being copied by the user. The virus is written in a high-level language, and creates a 52480-byte file called DOSINFO.EXE in several sub-directories of the fixed disk. The Trojan then adds code to the start of batch files on the disk to ensure that the program is executed. The program was distributed in an X-rated file, uploaded to German BBSs ▮

## Prism Reaches Out

The *NCSA* (*National Computer Security Association*) has announced the launch of a new program, *Prism*. Services provided by the program include access to on-line help, telephone help-desk support, the 'Underground Research Laboratory', the Virus Research Centre, magazines and newsletters, and national seminars and conferences with internationally-recognised experts in attendance. Members will also be warned by Email of any potential virus attack, and have the use of the product information service.

The program is a logical solution to a resource problem: many corporate IT Managers suffer from an overload of unscreened information, and have to allocate considerable resources to filtering it. *Prism* is designed to carry out this filtering first, supplying information from a wide range of sources, without swamping the company in trivia.

Its member-elected Advisory Council helps to determine the program's direction, assist in establishing special interest groups, provide input to educational programs, and recommend new or amended member services.

*Prism* membership is offered to government and business organisations of all sizes through a multilevel pricing schedule, calculated according to the revenue of the company concerned. Membership starts at US$4,500.00. Further details are available from the *NCSA*, tel. +1 717 258 1816, fax +1 717 243 8642. The *NCSA* can also be reached on *CompuServe* as 75300,2557@compuserve.com ▮

### Virus Prevalence Table - July 1994

| Virus | Incidents | (%) Reports |
|---|---|---|
| Form | 15 | 34.1% |
| Spanish_Telecom | 6 | 13.6% |
| CMOS4 | 3 | 6.8% |
| Flip | 2 | 4.5% |
| Green_Caterpillar | 2 | 4.5% |
| Green_Caterpillar.B | 2 | 4.5% |
| JackRipper | 2 | 4.5% |
| Smeg.Pathogen | 2 | 4.5% |
| Cascade | 1 | 2.3% |
| Eddie_2 | 1 | 2.3% |
| Joshi | 1 | 2.3% |
| New_Zealand | 1 | 2.3% |
| New_Zealand.I | 1 | 2.3% |
| NoInt | 1 | 2.3% |
| Parity Boot | 1 | 2.3% |
| Stoned.O | 1 | 2.3% |
| Taiwan.2900.d | 1 | 2.3% |
| V-Sign | 1 | 2.3% |
| Total | 44 | 100.0% |

## ARCV Case Wrapped Up

The case of *ARCV*, the UK-based virus-writing group *Association for Really Cruel Viruses*, has finally reached its conclusion (see *Virus Bulletin*, November 92, p.3). DC Noel Bonczoszek, at the time an officer of *New Scotland Yard's Computer Crime Unit* recently issued a statement saying that the President, Secretary and two couriers of the group had been identified, arrested, and were subsequently given a police caution. A fifth person was cautioned on another matter, while another arrested at the time was released with no further action taken. However, no victims of viruses written by *ARCV* were identified.

The statement goes on to thank the anti-virus community for their assistance. Commenting on the case, Bonczoszek (now attached to *Marylebone CID*) said, 'a potentially serious problem was nipped in the bud.'

The arrest and cautioning of members of *ARCV* is likely to be met with a mixed response from those in the IT industry. Although the virus-writing group was stopped in its tracks, the lack of convictions will be a source of irritation to some. Part of the reason for members of *ARCV* not being taken to court is believed to be the dearth of reports of their viruses in the wild, highlighting the need for those affected by viruses to report the attack to the appropriate authorities. The *CCU* can be contacted on Tel. 0171 230 1177 ▮

# IBM PC VIRUSES (UPDATE)

The following is a list of updates and amendments to the *Virus Bulletin Table of Known IBM PC Viruses* as of 20 August 1994. Each entry consists of the virus name, its aliases (if any) and the virus type. This is followed by a short description (if available) and a 24-byte hexadecimal search pattern to detect the presence of the virus with a disk utility or a dedicated scanner which contains a user-updatable pattern library.

**Type Codes**

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

**ARCV.Christmas.678**
**CR:** This virus appears to be based on the same source code as the 670-byte ARCV.Christmas virus. Detected with the Ice-9 pattern.

**Barrotes.1310.F**
**CER:** Detected with the Barrotes pattern, as is the 1310.G variant.

**Bupt.1220.C**
**CER:** Detected with the Bupt (Traveller) pattern.

**Burger.441.B**
**CN:** Detected with the Burger pattern and the Virdem and Virdem-fam patterns. The Virdem patterns also match the Burger.382.C virus.

**Cascade.1701.T**
**CR:** Detected with the Cascade(1) pattern, as are the 1701.U and 1701.V variants. The 1701.Q variant requires a new searchstring, as the decryption loop has been modified.

```
Cascade.1701.Q      018B D9EB 0446 4943 418D B74D 01BC 8206 8134 F066 464C 75F8
```

**Chaos.I**
**CER:** Detected with the Chaos (formerly Spyer) search pattern.

**Chaos_Year**
**CER:** An unremarkable 1837-byte virus.

```
Chaos_Year          3D00 4B75 06E8 F102 E97B 0080 FC3D 7506 E84D 04E9 7000 80FC
```

**Chris**
**CR:** This 463-byte virus contains the text '<CHRIS of S.i.t.>'.

```
Chris               80FC 4B74 052E FF2E FC01 061E 5557 5652 5153 5090 901E 5231
```

**Chromo**
**CN:** A 406-byte virus containing the text '[Chromosome Glitch] v1.0 Copyright (c) 1993 Memory Lapse'. This virus may perhaps be reclassified as a member of a family which contains several other viruses by the same author.

```
Chromo              CCC6 8699 0200 C686 9A02 00EB 00B4 4EB9 FF01 8D96 5602 CC3D
```

**Cobra**
**CN:** A 400-byte virus which prepends itself to the files it infects. It contains the text '~Cobra Cou~'.

```
Cobra               A19A 00A3 1B01 E80C 00B4 4FCD 213D 1200 7402 EBE0 C3B9 0500
```

**Cybertech.552**
**CN:** This virus uses variable encryption, and no searchstring is possible. The following text is present within the decrypted code: 'Mourners of a dying world. Too late to reconcile. Into Everlasting Fire. Can't you see it's Satan's world.'

**Cybertech.1066**
**CN:** Another encrypted variant, but the decryption loop is constant. There is also a 1228-byte variant by the same author.

```
Cybertech.1066      E800 005D 83ED 0750 8DBE 1B00 89FE B913 04AC 34?? AAE2 FA
Cybertech.1228      E800 005D 83ED 0750 8DBE 1B00 89FE B9B5 04AC 34?? AAE2 FA
```

**Dicker**
**CR:** A 400-byte virus which prepends itself to infected files.

```
Dicker              80FC 9075 03BB 9900 3D00 4B74 052E FF2E 3401 9C50 5351 5206
```

**FeelBad**
**CN:** This 1124-byte virus probably originated in the Netherlands. Its name derives from the text 'we feel bad about Ritzen'.

```
FeelBad             B840 008E D8BB 6C00 8A07 1F24 033C 0375 06BB 7804 E801 00C3
```

**Fifo**
**CR:** A 300-byte virus containing the text 'FIFO'.

```
Fifo                80FC 3674 03E9 D300 5053 5152 1E06 55B4 19CD 2150 FECA 7804
```

**Filehider.1057**
**CR:** Detected with the Filehider (789) pattern. Similar to a 1067-byte variant reported in July 1993.

**Fission**
**CER:** A 517-byte virus containing the text '[Binary Fission] v 1.0 [ML/PS]'. The text indicates that it is written by the same person who wrote Chromo.

```
Fission             3D00 3D74 283D 013D 7423 3D02 3D74 1E3D 0043 7419 3D01 4374
```

**Flip** **CER:** Three variants of Flip (2153.E, 2153.G and 2365) have not been mentioned before. Like other Flip viruses, they cannot be detected with a single, simple searchstring, but programs capable of detecting earlier Flip variants should also be able to detect these.

**Flue** **CN:** A variable-size virus (1179 bytes long or more) using variable encryption.

**Friday_the_13th.416.C** **CN:** An unremarkable minor variant of this virus, requiring a new searchstring.

```
Friday_13th.416.C   FF36 0201 FF36 0401 B43F B903 00BA 0201 CD21 725F A00E 0138
```

**Green_Caterpillar.1575.H** **CER:** Detected with the Green_Caterpillar (1575) pattern.

**IVP** **CN, CEN:** Three new IVP-generated viruses are known: 260 (CN), April (1676) and Mandela.943.

**Jerusalem.Sunday.L** **CER:** An unremarkable 1636-byte variant, detected with the Jeru-1735 pattern. The same pattern will also detect the new 2064-byte Jerusalem.Tarapa.C virus.

**Keypress.1232.** **CER:** Detected with the Keypress pattern.

**Leprosy.Busted.572** **EN:** Yet another member of this family of primitive overwriting viruses.

```
Leprosy.Busted.572  8B0E 0C02 51E8 1000 5BB9 3C02 90BA 0001 B440 CD21 E801 00C3
```

**Necropolis.C** **CEN:** Very similar to the other two known variants; detected with the Necropolis (1963) pattern.

**PS-MPC** The appearance of the following PS-MPC viruses should not be a surprise to anyone: 339.F (CN), 347.K (CN), 352.M (CN), 574.E (CEN), 578.H (CEN), Alien.733 (CER), ARCV-4.742 (CEN), Asstral (EN, 753), G2.Mudshark.312 (CN), Joshua.964 (CEN), Shiny.934 (CN), Sucker (CR, 572), Tester (CN, 302).

**Trivial** **CN:** There is a constant trickle of new small overwriting viruses which do nothing but replicate. Due to their small size, the patterns are shorter than normal, and should be used with care.

```
Trivial.25.B        BA9E 00CD 212A 2E2A 00B7 4087 D193 EBF3
Trivial.29.B        21BA 9E00 B802 3DCD 2193 5AB4 40CD 21C3
Trivial.30.G        218B D8B4 40B1 1EBA 0001 CD21 2A2E 2A00
Trivial.33          2193 BA00 01B4 40CD 21C3 2A2E 434F 4D00
Trivial.37          0001 CD21 B43E CD21 B44F EBE4 2A2E 2A00
Trivial.38.B        BA00 01B9 2600 CD21 CD20 2A2E 636F 6D00
Trivial.39.B        B440 CD21 B43E CD21 B44F EBE2 2A2E 2A00
Trivial.42.F        21B4 3ECD 21B4 4FEB E2CD 202A 2E63 2A00
Trivial.42.G        CD21 B43E CD21 B44F EBE1 2A2E 636F 6D00
Trivial.43.B        B43E CD21 B44F CD21 73E4 C32A 2E63 2A00
Trivial.43.C        B92B 00BA 0001 CD21 CD20 2A2E 636F 6D00
Trivial.45.E        7473 7920 7275 6C65 7321 202A 2E43 2A00
Trivial.54          EBE0 2E2E 00B4 3B5A BA28 01CD 2173 CBC3
```

**Troi.E** **CR:** Almost identical to the C variant. Detected with the Troi pattern.

**VCL** **CN, PN:** Several VCL-generated viruses have appeared recently: 609, Beepop (PN, 587), Bigtime (676), Butthole (overwriting, 493), Dumbco (3808), Genesis (741), Gif (696), Renegade (5737) and Westward (657). Most are encrypted, and should be detected as other VCL viruses: Westward is not, and is detected with the VCL.VoCo pattern.

**Vienna.648.Oscar** **CN:** Three 648-byte variants have been found recently, all of which contain the text '(C) OSCAR'. Variants A and C are detected with the interceptor pattern, but B requires a new pattern.

```
Vienna.648.Oscar.B  B903 008B D690 83C2 0DCD 218B 5406 8B4C 0483 E1E0 83C9 1D90
```

**Vienna.778** **CN:** Detected with the Dr_Q pattern.

**Vienna.Violator.707.B** **CN:** Detected with the Violator pattern.

**Vienna.Violator.5286.B** **CN:** Detected with the Xmas_Viol pattern.

**Xph.1010** **CER:** Similar to the two variants reported earlier.

```
Xph.1010            3D00 4B74 0580 FC3D 7553 2EC6 060C 0401 8BFA 4774 4280 3D00
```

**YB.316** **CN:** This virus is also known as Silent Runner, as it contains the text 'Silent Runner by Nostradamus [NuKE'94]'. It is 316 bytes long, and has not been fully analysed.

```
YB.316              B802 3DCD 2193 B905 008D 9408 01B4 3FCD 2172 218B 842B 0105
```

**YB.466** **CN:** This virus contains the text 'YB-1 & Handsome Dick Manitoba / Köhntark', indicating that it is by the same author as the KAOS4 virus.

```
YB.466              B802 3DCD 2172 2F93 B905 008D 9494 01B4 3FCD 2172 218B 84C1
```

**YB.647** **CN:** A related virus, containing the text 'YB-2 / Köhntark'.

```
YB.647              B802 3D9C FF9C 6801 72E3 93B9 0500 8D94 5F01 B43F 9CFF 9C68
```

# INSIGHT

## KAOS on the Superhighway?

*Virus Bulletin* readers will have noticed the short 'Stop Press' notice regarding the KAOS4 virus which was included in last month's edition. One month on, it now appears that the spread of the virus has been checked by a prompt response from anti-virus software manufacturers and members of the *Internet* user community. However, there is no doubt that, were it not for the ineptitude of the virus writer, a great deal more damage could have occurred.

With companies climbing over each other in a scramble for increased connectivity, the incident provides an ideal opportunity to review some of the risks associated with *Internet* access.

### Navigating the Internet?

One of the biggest misconceptions about the *Internet* is that it is actually run or controlled by a single body. However, given that there is a blurred definition of what the *Internet* actually is, this may require some further explanation.

Simply put, the *Internet* is a communications network. This may sound rather unimpressive, but estimates of the numbers of computers attached start at a highly conservative million, with more computers being added at a rate of hundreds or thousands a day.

The *Internet* consists of a number of sub-networks. Often these sub-networks are publicly funded, and have owners who recognise that adding connections to other networks enhances their functionality. Thus, as its name implies, the *Internet* is simply a network of networks.

One of the most visible uses of the *Internet* is for sending and receiving Email. Although only text can be transmitted via Email, it is possible to encode binary files as text, allowing executables to be transferred quickly and cheaply worldwide. This provides a way for potentially infected files to enter a system. Unfortunately, such ways are legion.

### Newsgroups

In the case of the KAOS4 virus, an infected file encoded as text was posted to the *Internet* newsgroup alt.binaries.pictures.erotica. That file was downloaded by a number of users, and once on their own machines, decoded, and reconstituted into an executable file.

The *Internet* newsgroups (known as *Netnews* or *Usenet*) are, just like the *Internet*, not run by any individual body. Rather, they have evolved out of a messaging system originally designed to deal with a handful of computers linked together in a *UUCP* (*Unix to Unix Communications Protocol*) network. However, as time (and technology) marched on, this system became unacceptable, and the present system was created.

It was later decided to divide the newsgroups into sub-groups. The most common of these are:

| | |
|---|---|
| **comp** | discussion of computers |
| **news** | discussion of news groups and news |
| **sci** | scientific discussion |
| **rec** | recreational discussion (e.g. pyrotechnics, chess, cycling) |
| **talk** | issue-related discussion (e.g. politics) |
| **misc** | miscellaneous topics. |

This event created tension within the *Usenet* community, which in turn led to the birth of the 'alt' newsgroups (with alt standing for alternative). Even more so than the main-stream newsgroups, the alt hierarchy is completely anarchic, and contains a wide variety of topics, with groups ranging from alt.hackers to alt.swedishchef.bork.bork.bork.

Also included in the alt newsgroups is the 'erotic' picture group alt.binaries.pictures.erotica (abpe). Such newsgroups contain many megabytes of scanned GIF or JPEG files of dubious origin, as well as animation programs or picture viewers. As an interesting aside, the newsgroup abpe is responsible for a significant chunk of the network traffic which makes up *Usenet*.

### Regulatory Bodies

The above discussion may make *Usenet* sound chaotic, but that would not be an unfair description. It is possible to post to *Usenet* anonymously, and (especially in the alt newsgroup hierarchy) there is no filtering or checking of the contents of messages. Thus, downloading any executable file from *Usenet* is a game of chance: although it is likely that the file is exactly what it claims to be, there is a remote possibility that it will contain a Trojan horse or a virus.

One might think that a virus author would be insane to post a new virus, as this would reveal his identity. Unfortunately, this is not the case, as *Usenet* posts can be easily faked and forged. This means that the virus author could disguise a new virus as a utility, and anonymously post the item to the newsgroup. Fortunately, this is very rare.

The user who posted the file infected with KAOS4, *Sexotica*, claims that he did not know that it was infected. This is the case for most of the viruses which have cropped up on the *Internet* so far.

As postings to *Usenet* cannot be trusted, it is worth considering other sources of information on the *Internet*. One of the most popular methods is *gopher* - this is a program which allows inexperienced users to find information or files on a particular topic by searching a simple text-based menu system. Such systems generally allow the user to hop across the globe from site to site, homing in on the area of interest.

Another source of information is ftp sites: large software collections containing many Gigabytes of files, usually operated and maintained by universities or colleges. Unlike postings to *Usenet*, uploaded files are generally placed in a secure area, so that the system manager can check their contents and suitability before allowing other users to access them. Ftp sites can be accessed quickly and easily, and generally do not require the use of a password for access. This provides a certain level of anonymity.

*"one of the first rules for avoiding virus infection via the Internet is exactly the same as for general PC use: do not use software of unknown origins"*

Although ftp sites are an excellent source of information, and often represent well-maintained and catalogued software collections, the same problem of accountability still exists. The ftp site will not always have been sent the file by the author of the package (and even if the site believes that it has been, this is not always easy to prove), so it is still possible that a file could be Trojanised in some way.

Other forms of file distribution on the *Internet* are similarly unreliable. Personal Email is trivial to forge (ask any first-year computer scientist for a demonstration), and can be done automatically by several programs or *Unix* scripts.

### Fighting Back

The preceding information brings little cheer to the average computer user. However, all is not doom and gloom, as many individuals and companies have begun to search for ways in which to make use of the many benefits of *Internet* access more secure.

One of the first rules for avoiding virus infection via the *Internet* is exactly the same as for general PC use: do not use software of unknown origins. In the case of the *Internet*, this will include ftp sites, and more importantly, software encoded as ASCII posted to newsgroups. Obviously such paranoia can only be taken so far. However, for a large network, it seems prudent to follow the oft-stated rule of obtaining software only from trusted sources. The ftp sites maintained by a number of anti-virus software manufacturers are obviously somewhat more reliable, and can be used (with the simple caveat that one can never be completely certain when communicating over the *Internet*).

In order to offer some sort of message authentication system, several encryption programs have been developed. The most popular, *PGP* (*Pretty Good Privacy*) uses a Public/Private key system, so that without a user's private key, it is impossible to fake messages which appear to be from him. Additionally, a message can be sent in such a way that it can only be decrypted by the recipient. Solutions to the problem of mail and file tampering by programs like *PGP* are becoming more common, as users begin to see first hand evidence of forged 'joke' postings.

Apart from encryption systems, most commercial networks connected to the *Internet* have an *Internet Firewall* set up. Although this is principally designed to deter potential hackers, the Firewall can also be configured to prevent users accessing various services and features provided by the *Internet*. The most draconian solution would be to provide access only to Email. Unfortunately even this is not completely effective: several newsgroups also exist in list form, and there are numerous ftp mail servers which can send files to users via Email.

### Conclusions

From a purely virus-related point of view, the *Internet* provides nothing but trouble. However, these problems are not *Internet*-specific: they apply equally to any route by which files can enter a company.

Files which are posted on the *Internet* are not automatically downloaded by unsuspecting users: the user has to access the file, decode it, and run it, for there to be any danger to the host system. Therefore enabling Email is not a risk *per se*, as the user has to take quite deliberate action in order to spread a virus. As GUIs to the *Internet* grow in popularity, this may not always be the case - soon, files may be automatically extracted and restored to their original form.

Of all the ways in which a virus can enter a company, Email and *Internet* access probably rank as two of the lower threats. However, on a global scale it does make a very tempting target, as it allows a reasonably anonymous way to distribute virus code. Therefore, it is important that the usual precautions for dealing with programs are followed.

When made *Internet*-specific, these are:

- Do not use software of doubtful origin (e.g. executable files from public ftp archives and *Usenet* postings).
- Scan all incoming software. Note that most scanners cannot search a binary file encoded as a text file; therefore the file must be decoded first. Some *Internet Firewalls* can be configured to do this automatically.
- When transferring executable code or confidential information, always use a message authentication or encryption system.

These rules, if followed as part of a general policy, will provide an excellent preventative against viruses via Email or the *Internet*. Ignore them at your peril!

# VIRUS ANALYSIS 1

## KAOS4: A Sexually Transmitted Virus?

The KAOS4 virus gained notoriety through its posting to the *Internet* newsgroup alt.binaries.pictures.erotica. Although KAOS4 has, as a result of this method of distribution, become widespread, it appears to be a relatively simple, non-resident COM and EXE file infector, designed to avoid detection by heuristic scanners.

### A Simple Plague

KAOS4 is a rather primitive virus, which makes no attempt to hide its presence, either during or after execution of a file. As the virus does not become memory-resident, no stealth routines are included, and, excepting encryption of some text strings stored in the virus code, disassembly proved to be trivial. It will be stopped by any behaviour blocker, and any of the popular checksumming programs should be able to detect its presence.

### Infection and Operation

The virus infects COM files by appending its code to the host file. When such a file is run, the virus receives control after execution of the starting JMP instruction, and some effort is made to restore the program's original registers before processing continues. No attempt is made to armour the code against disassembly, and the entire virus was pulled apart in a matter of hours.

The virus then sets up its own Disk Transfer Area and decrypts three text strings using a NOT instruction (the decrypted strings are *.COM *.EXE and PATH=). The purpose of this is to avoid detection by scanners which utilise heuristic detection techniques.

A pointer is set up to the string *.COM, and the infection routine is called. Once this routine has completed, the pointer is reset to point to *.EXE, and the process repeated. No checks are made on returning from the infection routine.

The virus then restores the image of the host file in memory, and returns control to it.

### Up the Garden Path

The infection routine contained in KAOS4 is poorly written. It begins by searching the Environment Segment (held in the Program Segment Prefix) for the PATH variable. This is done in such a way that if the environment segment does not contain the character sequence 'PATH=', the code will enter an infinite loop. On all the versions of DOS tested, this string is present even if no path has been set. Although this information will not change while the virus is executing, the search for the path variable is carried out twice every time an infected file is run.

If a match is found, the address of the PATH is stored for later use. The virus then searches for a matching file using the DOS FIND_FIRST function. If no match is found, a routine which attempts to allow the virus to search along the path for an infected file is called. A check is made to ensure that the PATH name has been found, or that all parts of the PATH have been searched. If this is not the case, another search is made for a suitable file by searching along the path set up on the machine.

### Self-infection Checks

One of the continual problems encountered when disassembling a virus is to ascertain what the author thought his code would do when he wrote it. This was the case when examining the KAOS4 virus, where the checks made before infecting a file are somewhat bizarre.

Firstly, the seconds field of the time stamp of the file is checked against a mask of xxx111x1. If a match is found, the file is deemed unsuitable for infection. This will cause the virus to reject files which have a seconds stamp matching this pattern (e.g. 58, 62). This appears to be a self-infection check, as infected files have the value 58 in the seconds field of the time stamp.

*"in this respect, the virus functions very well indeed: infected files do not raise a single heuristic warning flag with ThunderBYTE"*

If the test made on the time stamp is passed, KAOS4 checks whether the internal structure of the file is EXE or COM. This is carried out by checking the first two bytes of the file for the ASCII letters MZ or ZM. The virus is written in such a way that a heuristic scanner will not identify the true functionality of the code. In this respect, the virus functions very well: infected files do not raise a single heuristic warning flag with *ThunderBYTE*; a creditable achievement. For obvious reasons, the precise way in which the virus achieves this is not stated. Suffice it to say that it works, although the effort involved seems to be wasted, given the rather obvious way in which the virus operates.

In the case of COM files, the target file is only infected if it does not begin with the words E9??h ??20h, and if the length of the infected file would be less than 64K. For EXE

files, offsets 18h, 1Ah, and 12h of the EXE header are examined. These are the areas which contain the Relocation Table Offset, the Overlay number, and the Checksum. Given that the virus has already carried out a self-infection check, these further tests seem to be unnecessary. These tests completed, a flag is set to indicate whether the target file is an EXE or a COM file.

The infection routine is standard, and only of note because it does not work correctly. Under certain circumstances, the virus body can become corrupted, allowing subsequent infections to attack only the first COM and EXE file found in each directory on the path. These partial infections do not operate correctly, and can cause the system to hang after they have infected other files.

### Conclusions and Thoughts

Due to the simple-minded way in which the virus is written, KAOS4 poses little long-term threat to the user community. Apart from its novel distribution method, the virus seems to be merely an ego trip for its author. According to a text string stored within the virus, this is none other than 'Köhntark', a virus writer who wrote a rambling description of how to avoid heuristically-based scanners. Users should be grateful that he is appallingly bad at his chosen pastime.

Although KAOS4 can usually replicate successfully, working its way down infected directory trees, on machines with a large PATH variable set up, the large amount of disk activity caused by the virus will soon become noticeable. Notwithstanding its rather obvious behaviour, anyone in the UK who has been affected by the virus is urged to call *New Scotland Yard's Computer Crime Unit*, on 0171 230 1177 so that, should the culprit be found, he can be held responsible for the actions of his creation.

### KAOS4

**Aliases:** None known.

**Type:** Non-resident parasitic file infector.

**Infection:** COM and EXE files.

**Self-recognition in Memory:**

None necessary.

**Self-recognition in Files:**

Checks checksum value in EXE file, or the first four bytes of COM files.

**Hex Pattern:**

```
8C96 D102 2E89 A6D3 028C C88E
D0BC FFEF 2E8A 86B4 022E 8C86
D502 5006 1E0E 0E07 1FFF B6B0
```

**Trigger:** None.

**Removal:** Under clean system conditions, identify and replace infected files.

# VIRUS ANALYSIS 2

## No Smoking, Please!

*Eugene Kaspersky*
*Kami Associates*

Living in limited spaces is a twentieth-century problem to which every creature has had to adapt, in order to survive. Most have managed, including viruses - the biological as well as the computer types. Computer viruses (and their authors) have proven their adaptability: they have outwitted DOS, and the most successful, the more advanced, have even spread to *MS-Windows*, and *OS/2*.

Not all viruses are 'leaders': there are thousands of 'ordinary' DOS viruses. However, if a virus cannot carry out a task on a particular operating system, it will sometimes try and sneak through a chink in the armour of another… meet No_Smoking, the virus which sends *NetWare* messages.

### Infection

No_Smoking is a 1575-byte long self-encrypting COM file infector. It is not memory-resident, but does leave part of its own Int 21h handler operational in memory (see below).

When an infected file is executed, control is passed to the virus' decryption routine. The virus encrypts itself twice, using a different algorithm each time, so two decryption routines are required. The second of these is quite complex: the virus author has altered the stack pointer in order to enable decryption data to be passed back to the virus. This routine will disable every debugger bar a protected mode debugger, and makes single-stepping of the code impossible.

The virus then hooks Int 21h and Int 24h (see details below) and calls the infection routine. This routine scans the directory tree of the current disk (by using the mask '*.*'), searches for five uninfected files (using the mask '*.COM') and writes the body of the virus at the end of any COM files which are found.

On infection, the virus first checks file length - any file over 59860 bytes long will not be infected. It then looks at the file's first instruction: if this is a JMP, it will compare the file entry code with its own code. If the code differs, the file is infected, but if it is the same, no action is taken. No_Smoking has a tendency to corrupt very small files, as it does not check them properly before infection.

Internal file format is not checked by the virus, which will also corrupt EXE files with a COM extension (such files appear, for example, on compression of COM files with certain versions of DIET). The virus saves and restores the file date and time stamp, and the file attributes, on infection. Before opening a file, it will clear the file attributes, as a means of disabling the read-only attribute. Its penultimate

action is to append a random number of NOP instructions to the end of the file. Then, finally, it encrypts itself, and writes itself to the end of the file, modifying the beginning of the file with a JMP VIRUS instruction.

### Interrupt Hooking

When an infected file is executed, the virus first checks to see if its own Int 21h handler is installed. This handler is highly unusual, in that it is not used to infect other files, but to hold the original Int 21h address. Thus, although the virus leaves part of its code resident, it is not classified as a memory-resident virus, as infection only takes place when an infected file is executed. If the virus does not recognise its own handler, it hooks both Int 21h and Int 24h, decreasing the size of the last memory block. The virus' own Int 21h handler is then copied into this area.

*"this handler is highly unusual, in that it is not used to infect other files, but is used to hold the original Int 21h address"*

The Int 24h handler is used by the virus to suppress the DOS critical error handler message. It also causes the virus to store an error flag in the code of the Int 21h handler which is already installed. When an infected file is processed, the virus calls Int 21h with AX=FFFFh. The Int 21h handler returns the pointer to its code, and the main virus code reads that byte which contains the error flag.

This is quite an interesting algorithm, and is used only to pass information to the trigger routine. It is the lengthiest way I have ever seen of passing a single bit of information. Of course, this may be merely an 'anti-antivirus-researcher' ruse, meant to confuse anyone set on disassembly.

### Trigger Routine

If there is an Int 24h call on infection, the virus calls the trigger routine. Should *Novell NetWare* not be installed, the trigger will (of course) not be activated.

The first step towards activation of the trigger routine is for the virus to obtain the name of the server to which the infected computer is connected. This is effected through the function GET FILE SERVER INFORMATION (which is one of the Int 21h, AH=E3h *Novell NetWare* functions). If there is more than one server in the network, the function will return the name of that server which was used as the first on the login procedure.

When this is complete, the virus will find out the number of users connected to that server (using the function GET FILE SERVER INFORMATION), obtain its own computer number (GET CONNECTION NUMBER, Int 21h, AH=DCh), select two connected computers (using its own random number generator) and get the names and network addresses of these computers through the GET CONNECTION INFORMATION function.

After selecting two of the computers which are connected to the network, the virus generates the phrase 'NAME: Text' where 'NAME' is the network name of the first selected computer, and 'Text' is one of the following strings:

```
Friday I'm in LOVE!
No Smoking, please! Thanks.
```

This is then sent to another computer. At a casual glance, a user could be forgiven for thinking that such a message was nothing more than a joke making its way from one user to another. The virus sends its message every time it is executed, with the eventual result that all the users end up asking each other…

### In Conclusion

Viruses which are designed to take advantage of the additional functionality of computers connected to *Novell NetWare* networks are very rare. However, they are likely to grow in number: users and network managers should be aware that virus authors are continually probing systems for signs of weakness. No_Smoking just sends messages… how long until we see a virus which is capable of crashing the the remote PC, or making the network unusable?

## No_Smoking

**Aliases:** None known.

**Type:** Encrypted, parasitic COM file infector. Not memory-resident, but leaves an Int 21h handler in memory.

**Infection:** COM files only.

**Self-recognition in Memory:** Int 21h with AX=FFFFh returns pointer to Int 21h handler.

**Self-recognition in Files:** Compares file entry point code with the start of the virus code.

**Hex Pattern:** Due to the encryption routine, the following hex pattern is very short, and should be used with care.

```
60E8 0000 5E83 C60D B916 0680
34?? 46E2 FA
```

**Intercepts:** Int 21h and Int 24h on infection to prevent DOS error message and to set the flag to call trigger routine.

**Trigger:** Sends message (detailed in text) to *Novell NetWare* stations.

**Removal:** Under clean system conditions, identify and replace infected files.

# COMPARATIVE REVIEW

## Disinfection: Worth the Risk?

Virus disinfection software has always been a bone of contention between *Virus Bulletin* and certain advocates of the technique. This journal's policy, from its very first edition, has been forceful discouragement of virus disinfection, and a believer in removal (deletion of the infected file).

That advice still holds true: it is best, wherever possible, to recover from a virus attack by reinstalling software from a known source, i.e. a backup or the original installation disks. However, as almost all anti-virus software packages now provide some disinfection options, the time has come for a review of such features.

### Boot Sector vs. File Viruses

The issues involved in disinfecting boot sector viruses from disks, as opposed to disinfecting file viruses from infected files, are very different.

When a boot sector virus infects a fixed disk, the original Master Boot Sector (MBS) or DOS Boot Sector (DBS) is usually stored elsewhere on the disk. Boot sector virus disinfection, therefore, is a case of identifying the virus, using knowledge of the virus to locate the original boot code, and copying that code back to the appropriate location. The contents of any additional sectors used by the virus are usually left on the disk. This is a reliable procedure, because it simply involves overwriting the virus code, not attempting to reconstruct an infected object.

In this review, three boot sector viruses were used: Form, JackRipper, and Monkey. The first two present little challenge to anti-virus programs: they are easy to identify, and disinfection consists simply of copying the appropriate sector back to the boot sector of the fixed disk.

Detection and disinfection of Monkey is more difficult, for two reasons. Firstly, this virus overwrites the partition information in the MBS. Thus, if an infected PC is booted from a clean system disk, the operating system cannot find a DOS partition on the fixed disk: this makes drive C inaccessible to DOS. Secondly, the virus XORs the original MBS ('encrypts' is far too grandiose a word). Therefore, before replacing the boot sector, it must be decrypted. These idiosyncrasies cause problems for certain products.

### File Virus Disinfection

While boot sector virus disinfection can be recommended, the question of whether to repair or delete infected files is a thorny one. To understand fully why the process is fraught with pitfalls, the structure of both COM and EXE files in turn must first be considered.

A COM file is the simplest type of executable object on the PC. Such files consist of a sequence of executable instructions which the processor follows. The operating system does not preprocess COM files: they are merely loaded into memory, and executed.

When a virus infects a COM file, the first instruction of that file is usually replaced with a jump to the virus code. After the virus has completed its infection process, the altered instruction is repaired, and control returned to the file.

COM file disinfection, therefore, usually consists of locating the host file's original start instruction, replacing it at the beginning of the file, and removing the virus code from the end of the file. This process relies on the anti-virus software identifying the virus exactly: the disinfection routine operates by reversing the changes made by the virus.

Often, the virus pads the host file out to an exact number of paragraphs in length. When anti-virus software attempts to disinfect the file, it may leave this padding attached to the cleaned file. Although this should not cause the disinfected file to malfunction, certain programs 'disinfected' in such a way will not work correctly.

Disinfection is even more complicated for EXE files: these consist of executable code preceded by an 'EXE header' containing instructions to the operating system on how to load the file. Viruses append their code to EXE files, but rather than altering the first instruction in the file, they make changes to the information stored in the header.

When disinfecting EXE files, such changes must be reversed. This is often difficult: the header includes information about file size in terms of the number of pages, and the last page size (a page being 512 bytes of memory), relocation and segment information, and a checksum.

The checksum field is currently ignored by *MS-DOS*, although future versions of the operating system may not do so. Often the virus destroys some or all of this information irrevocably, and although a disinfected EXE file will probably not be affected by small changes to certain parts of the header, there is no guarantee that this will be the case.

### The Tests

The file-infector tests were constructed using five common viruses: Jerusalem.Standard.A, Jerusalem.Carfield, Cascade.1701.A, Cascade.1704.A, and Smeg.Pathogen. In each case, a number of (where appropriate) COM and/or EXE files of different file lengths were infected. Two test files were multiply-infected with Jerusalem.Standard.A. Anti-virus software was then used to disinfect the files, and a byte-by-byte comparison made of the disinfected file and the uninfected host file.

# Part 1: Boot Sector Viruses

In this section, each product was tested against the three boot sector viruses described above. The machines were all infected with the viruses before anti-virus software was installed, so restoration of the original boot sector from recovery disks, etc., was not possible.

## CPAV v2.2

*Central Point Anti-Virus* gained perfect scores here: in each case, the boot sector virus was identified and disinfected, with no special knowledge required by the user. Excellent.

## Dr Solomon's AVTK v6.66

*Dr Solomon's AVTK* detected the presence of both Form and JackRipper. However, disinfection was less than trivial. No automatic disinfection was provided for Form: the on-line help recommends using the 'SYS' command.

Ripper disinfection had to be carried out using the 'Partition Repair' option. This confronts the user with the message 'Replace partition sector on physical device 128?' followed by a request for a preformatted diskette. After this unnecessarily technical start, disinfection proceeded flawlessly. A nice touch was that the overwritten MBS is also backed up on to drive A, allowing changes to be undone if disinfection proves unsuccessful.

Detection of the Monkey virus was more complicated than it need have been. Attempting to use the *AVTK's* graphical front end was a failure: the large user-friendly button labelled 'C:' was blanked out. Running FINDVIRU directly from the command line was more successful: the virus was identified as Empire.Monkey.A, but no mention of that name could be found in the on-line virus encyclopedia.

At this point, a call to *S&S International* was made. Once put through to technical support, the reviewer was told that the disk had to be disinfected by hand with a sector editor, after booting from the infected drive. When it was pointed out that this would be a lengthy process for any more than a handful of drives, the technical support staff commented that the FDISK /MBR command could be used - a procedure which would render a Monkey-infected disk unusable.

## F-Prot v2.13a

*Frisk Software's F-Prot* performed well when run against fixed disks infected with Form and JackRipper. In each case, the disk was disinfected successfully.

Unfortunately, the product had problems with a fixed disk infected with Monkey. When run using the command line 'F-PROT C:', it scanned the A: drive instead, and reported that 'No viruses or suspicious files/boot sectors were found'. The command line switch '/HARD' was required in order to scan the fixed disk. Once chosen, identification and disinfection was simple. Running the product from the GUI required no customising of options.

## NAV v3.0

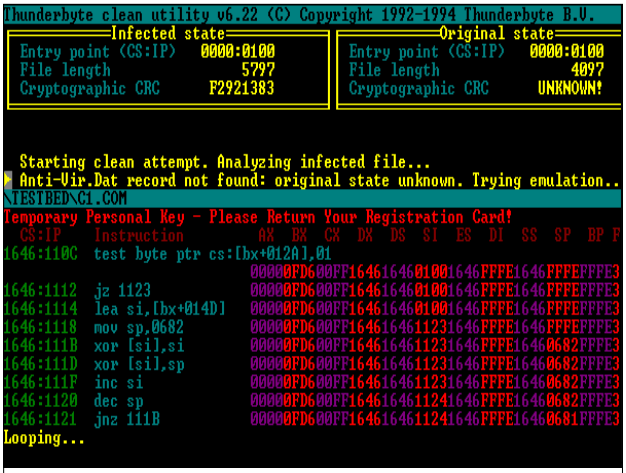
Disinfection of the Form virus was easy using *Norton Anti-Virus* - typing 'NAV' and requesting a scan of the C: drive identified the virus, and offered an option to disinfect it. The disinfection process completed successfully, and the product then offered further information about the virus.

JackRipper disinfection could not be tested, as the product was not capable of detecting the virus on either the fixed disk or infected diskettes.

When confronted with a fixed disk infected with Monkey, running *NAV* from the command line 'NAV' allowed the user to scan drives A or B - no mention was made of the fixed disk. However, during any scan, the fixed disk's MBS is scanned by default. After the default scan, the virus was identified as 'Monkey', an option was supplied to repair the disk, and disinfection was carried out successfully. Activating the program with the command line 'NAV C:' found the virus immediately.

## Scan v117

*McAfee's Scan* identified both JackRipper and Form. In both cases, running the CLEAN program with the appropriate command line options allowed the disk to be disinfected.




*ThunderBYTE's TBClean* approaches the disinfection problem by emulating the way in which the virus repairs an infected file before passing control to it.

When faced with a Monkey-infected fixed disk, however, the results were not as encouraging. The program did not check the MBS of the fixed disk by default, and an additional command-line option, /MAINT, was required. Once this option was selected, the virus was identified.

Running CLEAN with appropriate command-line options produced an error message stating that the program could not scan the C: drive. Running CLEAN with the command line '/?' provided little further information, only displaying a short list of command-line options. However, using the '/MAINT' switch (which was not listed) did disinfect the virus. Oddly, no mention of this switch in the CLEAN documentation could be found.

### Scan v.2.10

One of the new features of version 2 of *McAfee Associates'* scanner is the combination of the SCAN and CLEAN programs: the days of typing CLEAN C: [GenB] will soon be a thing of the past!

In each case, *Scan v2* identified the virus correctly, but when run with the /CLEAN option set, displayed the message 'No remover is currently available'. One improvement from the older version is that no special command-line switches were required in order to detect the Monkey virus (see above).

### Sweep v2.64

*Sophos' Sweep* was capable of identifying and disinfecting all the viruses when run with the command line 'SWEEP C:'. Adding the switch -DI ('disinfect') disinfected each virus successfully - an excellent result.

Attempting to use the *Sweep* graphical interface, SW, when the fixed disk was infected with the Monkey virus was not quite as trouble-free: the program displayed a blue box stating 'Critical error encountered while opening the file B:\SWEEP\SWEEP.CFG. Retry Specify New filename Cancel'. No such file was present on the *Sweep* disk (which was in the A: drive). Selecting 'Cancel' allowed one to use SW, but no option was available for scanning the C: drive.

If a scan of either the A: or B: drive was initiated, *Sweep* identified the virus on the fixed disk, and disinfected it. The program then reported that it had found a virus 'while sweeping drive B:'.

### ThunderBYTE v6.22

*ThunderBYTE* identified both Form and JackRipper on the fixed disk of the machine. However, no automatic cleaning option was provided for the Form virus: the user was advised to use the DOS 'SYS' command instead. In the case of JackRipper, *ThunderBYTE* did not offer to replace the original uninfected MBS, but to overwrite the virus with its own 'virus-resistant' code. This option was selected, and the disk was disinfected successfully.

Detection of the Monkey virus presented a problem after a clean boot. Running the *ThunderBYTE* graphical shell, TBAV, did not provide any obvious method of scanning the fixed disk. Similarly, calling the scanner directly with the command line 'TBSCAN /ALLDRIVES' returned the error message 'No drives to be scanned found'.

### Vi-Spy v12

*Vi-Spy*, from *RG Software*, identified both the JackRipper and the Form virus. Running the software with the /RMV switch set allowed both to be disinfected. One feature lacking from other packages is that the user is urged to back up the fixed disk before disinfection is attempted. This is a thorough measure, and a praiseworthy one.

Like many of the products tested, it fared less well when tested against a fixed disk infected with the Monkey virus. Running the program with the command line 'VI-SPY C:' produced the error message 'Drive C not ready', nor did selecting the 'all drives' option enable detection.

The company was contacted to ensure that the product was being used correctly. *Virus Bulletin* was told that the detection and disinfection routine for Monkey worked from an infected environment, but not from a clean one. When the test was repeated after booting from the hard drive, the disk was successfully disinfected. This is not an ideal solution, and the product should be altered so that it can be used after a clean boot, as described in the manual.

## Part 2: File Viruses

In this section, the file virus disinfection features of all products were tested. Once again, the programs were not given the chance to scan the files before they became infected, and therefore could not utilise any checksum information in order to disinfect the file. The way products cope with a machine upon which they have been preinstalled will be the subject of a forthcoming *Virus Bulletin* review.

### CPAV v2.2

*CPAV* was capable of repairing all files in the test-set infected with either Cascade.1701 (identified simply as '1701') or Cascade.1704 (identified as '1704'). Disinfected files were identical to their uninfected originals.

| PRODUCT | Form | JackRipper | Monkey | Cascade (1701) | Cascade (1704) | Jerusalem (Standard) | Jerusalem (Carfield) | Pathogen |
|---|---|---|---|---|---|---|---|---|
| CPAV v2.2 | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | Not detected |
| Dr Solomon's AVTK v6.66 | No disinfection | ✓ | No disinfection | ✓ | ✓ | ✓ | ✓ | No disinfection |
| F-Prot v2.13a | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| NAV v3.0 | ✓ | Virus not detected | ✓ | ✓ | ✓ | ✓ | ✗ | Not detected |
| Scan 9.30 V117 | ✓ | ✓ | Problems encountered | ✓ | ✓ | (See note 1) | ✗ | Not detected |
| Scan v2.10 | No disinfection | No disinfection | No disinfection | No disinfection | No disinfection | No disinfection | No disinfection | No disinfection |
| Sweep v2.64 | ✓ | ✓ | ✓ | No disinfection (See note 2) | No disinfection (See note 2) | No disinfection (See note 2) | No disinfection (See note 2) | No disinfection (See note 2) |
| ThunderBYTE v6.22 | No disinfection | Replaced with custom boot sector | Problems encountered | (See note 3) | (See note 3) | (See note 3) | (See note 3) | (See note 3) |
| Vi-Spy v12 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | No disinfection |

**Virus disinfection results.** In each case, a tick means that the virus was successfully detected and disinfected by following on-screen instructions. A cross indicates that the software attempted to disinfect the virus, but caused corruption. **Note 1:** *McAfee CLEAN* was the only program which did not restore the COM files to their original length. **Note 2:** *Sweep* does not offer any file virus disinfection options. **Note 3:** *ThunderBYTE* was the only product which did not utilise virus-specific information.

Jerusalem.Standard disinfection on COM files did not succeed: all 'cleaned' files were of zero length. Fortunately, disinfection of EXE files was much better: excepting usual changes to file length and checksum and last page size fields in the EXE header, disinfection worked, even for multiply-infected files. Jerusalem.Carfield, however, could be disinfected correctly, with only small changes left in the EXE header. No detection of Pathogen was provided.

## Dr Solomon's AVTK v6.66

File repair from within *Dr Solomon's Anti-Virus Toolkit* is very easy: the file repair option is selected from the menu, and the machine goes to work. Files which are successfully repaired are not backed up, whereas files which the product cannot repair are renamed to the extension VOM or VXE.

Before infection, the virus is identified (a process which involves examining the whole body of the virus) and the disinfection routine is called. When the disinfection option is specified, there is no prompt before disinfecting files, and the program automatically checks every file on the disk, regardless of extension.

The results were good. The *AVTK* identified the two Cascade variants as Cascade.1701.Standard and Cascade.1704.Standard respectively, and disinfected them. The repaired files were identical to the original host files. Similarly, files infected with Jerusalem.Standard were also identified and disinfected. All infected COM files were repaired exactly. However, although the executable code of EXE files was repaired, the checksum value in the EXE header was not returned to its original value.

Notably, EXE files were the same length as before infection. The two EXE files which were multiply-infected were detected and dealt with automatically, and, with the exception of a difference in the checksum field of the EXE header, repaired perfectly.

The second set of Jerusalem results were not quite as good: disinfected EXE files were always an exact number of paragraphs long, which led to changes in the EXE header checksum and the last page size fields. COM files were returned to their original state. None of the Pathogen-infected files were disinfected - these were all renamed to non-executable extensions.

## F-Prot v2.13a

This product identified all 16 Cascade infections as Cascade.1701.A, and offered a disinfection option. Files were disinfected perfectly, and a byte-by-byte binary file comparison showed them to be identical to the original files. No backup was made of the infected file for use in the event of failure.

Results were identical for Cascade.1704: it was identified as Cascade.1704.A, and files were disinfected correctly. When tested against the Jerusalem viruses, *F-Prot* identified the

virus as Jerusalem.1808.Standard, and offered a disinfection option. This was successful: disinfected COM files were bit-identical to the original uninfected file.

Disinfected EXE files were longer by the virus' padding value, depending on original host file length. Thus, disinfected EXE files sometimes had changes to the header, at offsets 02h, 12h and 13h, the 'Last Page Size' and 'Checksum' fields. The multiply-infected EXE file was automatically re-scanned after disinfection, and the disinfection routine called as necessary.

The second Jerusalem variant was identified as Jerusalem.Carfield. Once again, COM files were perfectly disinfected, but EXE files suffered from the same changes as discussed above.

*F-Prot* correctly identified all files infected with Pathogen, and repaired them. However, clean files were always an exact number of pages long. A binary comparison of disinfected files showed that both files were identical, except for 'padding' at the file end. In the case of EXE files, the header information was still unchanged, although the file was now longer.

## NAV v3.0

*NAV* identified all the Cascade-infected files as Cascade (1), regardless of variant, and offered an option to repair them. When this was selected, the user could choose to disinfect a single file or all infected files. Cascade.1701.A and Cascade.1704.A-infected files were returned to their condition prior to infection, and all files were backed up before being altered.

The Jerusalem.Standard virus was identified as Jeru.1808 on COM files, and Jeru.1808 (x) when attached to EXE files. All COM files were repaired perfectly. However, EXE files were rounded up to an exact number of paragraphs in length, and the checksum and Last Paragraph fields in the EXE header were altered.

The two EXE files which were multiply-infected were discovered and dealt with automatically. The alterations to repaired files were identical to those described above.

When examining the Jerusalem.Carfield samples, *NAV* failed to identify the virus correctly in all EXE files, stating that they were infected with 'Jeru.1808 (x)'. During disinfection, the COM files were completely destroyed: the starting JMP instruction was replaced by three NOP instructions, and the last ten bytes of the file were corrupted. One COM file was also shortened too much.

EXE file disinfection worked, with changes made only to the checksum and last page length fields made to the EXE header, and the file length rounded up to the nearest paragraph. None of the Pathogen samples were detected.

## Scan 9.30 V117

The original plan for this review did not include the old version of *McAfee Scan*. However, as none of the disinfection options seemed to work on the newer version, *Scan v117* was included after all.

Unlike the newer version, virus disinfection required the use of two different programs: SCAN identifies the virus, and CLEAN disinfects it. When run, *Scan* produced a removal name, which was used as a command-line parameter for CLEAN when it was executed.

All samples of Cascade.1701.A and 1704.A were detected, though both were found classified under the same name. Running CLEAN on the files with the appropriate command-line options set disinfected the files successfully: the cleaned files were identical to their uninfected counterparts. No backup of the infected files was made before disinfection was attempted.

The Jerusalem virus was detected by *Scan* as Jerusalem. CLEAN attempted to repair the files: all of the 'cleaned' COM files were correctly restored, but were five bytes longer after disinfection than the original host files. Disinfected EXE files were longer than before infection, with the corresponding last page size and checksum fields altered in the EXE header.

Disinfection of the Jerusalem.Carfield virus failed. Infected files were identified as infected with 'Sunday', and the disinfection routine removed too much code from the end of the host file, making them 123 bytes shorter than they should have been. Additionally, the starting JMP instruction of the cleaned file was corrupted. EXE file disinfection worked correctly, with the usual caveat about the alteration of last page size and checksum fields.

## Scan v.2.10

Files were cleaned by SCAN by using the /CLEAN option. However, the program repeatedly displayed the message 'No remover currently available for this virus'. Both samples of Cascade were identified simply as 'CASCADE'. All samples of Pathogen were discovered. Oddly, the Jerusalem virus was identified as MOCHA, and the Jerusalem.Carfield virus as Jerusalem.Westwood.A.

## Sweep v2.64

All infected files were correctly detected by *Sophos' Sweep*. However, this product provided no file virus-disinfection: infected files can only be renamed, deleted, or 'shredded' (securely erasured).

## ThunderBYTE v6.22

This product found all 16 of the Cascade.1701 samples, and offered options to delete or rename the files. Cleaning was carried out using a separate utility, *TBClean*, which operates one file at a time, backing up each file before any attempt at disinfection is made. Each file was cleaned heuristically, using no virus-specific information.

*TBAV's* attempts were not entirely successful. The original uninfected goat files were all one byte shorter than the 'cleaned' files, and no attempt was made to preserve the original file's time/date stamp. Moreover, a binary file comparison of the files showed discrepancies within the body of the file. Results were identical for Cascade.1704, making disinfection of both viruses a failure.

When examining the Jerusalem-infected files, *TBAV* identified the files as 'infected by Jerusalem-related virus'. Using *TBClean* on the files was once again not entirely successful: each COM file had an extra five-byte 'tail' attached to it. Apart from this, disinfection was successful, as the cleaned files' contents was correctly repaired.

Things were not as successful for the 'cleaned' EXE files: these were approximately 100 bytes longer than their originals, with differences in the header information, including offset 11h, the Initial SP value field. This is likely to be critical to the functioning of most programs.

Results on the multiply-infected file were uninspiring: the program did not identify the file as multiply infected, and had to be run many times. Once the file had had all layers of the virus removed (effected by calling *TBClean* several times), the resulting program was identical to that restored by cleaning a single infection.

Disinfection of the second Jerusalem variant was also unsuccessful: the EXE files no longer ran, as the initial values of CS and IP were incorrect. Cleaned COM files were five bytes too long, but errors were identical apart from this.

The results when run against Smeg.Pathogen were interesting. On some files, the software emulation of the virus code entered an infinite loop, and the program could not disinfect them, leaving the files with an executable extension.

On other files, cleaning was successful, although both COM and EXE files were padded out to an exact number of pages long. Although these disinfected files still ran, eight bytes of the 'clean' file differed from those of the original. In the EXE files, header information (including the original SP value) was altered - a potentially disastrous change.

This is something of an 'apples with oranges' comparison, as all other products were using virus-specific information to effect disinfection. In short, *TBClean* is probably the most technically-impressive package reviewed, but for all its bells and whistles, it cannot quite 'come up with the goods'.

## Vi-Spy v12

*Vi-Spy* identified both the Cascade.1701.A and .1704.A viruses as 'Cascade 1669-1702'. No explanatory text for this virus was available, but running the product with the /RMV option set did allow it to be disinfected. Before this could take place, *Vi-Spy* displayed a message explaining that it was possible that the file would not be '100% recovered', and that disinfected programs should be tested thoroughly.

Such honesty is laudable: the *Vi-Spy* documentation also makes it perfectly clear that disinfection is a last resort. This warning notwithstanding, all Cascade-infected files were returned to their original, uninfected condition.

The first set of Jerusalem viruses was identified as JERUSALEM-B. All COM files were repaired exactly; however, EXE files had their last page length changed, and the checksum entry in the EXE header altered. Excepting these changes, disinfected EXE files were identical to their original, uninfected counterparts. *Vi-Spy* identified the multiply-infected EXE file, and automatically removed the many layers of infection.

The Jerusalem.Carfield virus was identified as Sunday 2. Apart from this, disinfection results were the same as those for Jerusalem.

Although *Vi-Spy* detected all of the Pathogen-infected files, no disinfection routine was offered.

### Conclusions

The most interesting results were the problems encountered with the Monkey virus. Although all the products could detect the virus on diskettes, several handled the scanning of infected fixed disks poorly. The only product which escaped any criticism when tested against the boot sector viruses was *Central Point Anti-Virus*.

The file disinfection results are rather more difficult to interpret. Although most of the products were capable of disinfecting COM files, EXE file disinfection was a less well-defined process, with changes made to the file length and to the EXE header.

No program could restore all of the files perfectly. Though file corruption during disinfection was the exception, there were a few instances of disinfection routines destroying infected files. Encouragingly, *F-Prot* fared well, as did (to a lesser extent) *Dr Solomon's Anti-Virus Toolkit* and *Vi-Spy*.

The results show that, although disinfection can work, it cannot be relied upon as a method of recovery from virus attack. Although many products fare better when installed upon the machine before infection takes place (this topic will be the subject of a forthcoming review), it is no replacement for a regular backup, or a secure collection of software master disks.

# PRODUCT REVIEW 1

## Doctor: Good Medicine?

*Dr Keith Jackson*

*Doctor* is a fairly new release from *Thompson Network Software*, and consists of three major components; scanner, checksummer, and memory-resident monitor. It also has utilities providing password protection, a scheduler, facilities to display/print files, signature extraction from a file, and telephonic upgrade of virus signatures by downloading the latest information from the developer's Bulletin Board. An impressive array of features: how well does it perform?

### Documentation

The A5 ring-bound manual provided is 136 pages long, and indexed: it covers usage of the *Doctor* programs but does not go much, if at all, beyond that. Only two viruses are discussed in the manual (4K and Stoned), each of which has a mere half-page description. The introduction to these descriptions states: 'This is not yet meant to be a definitive collection of virus information.' We'd never have guessed.

Although the manual tries to explain each of the possible error messages which may be displayed, 30% of them are documented simply as: 'This error should never occur. Report instances to…'. Either an error message is worthwhile, both to display and to explain its meaning, or it is not. Placing the onus on users to provide debug information for the developers is not helpful.

### Installation

*Doctor* was provided on two floppy disks, one entitled 'Doctor', the other, 'Medicine Bag'. Both 3.5-inch (1.44 Mbyte) and 5.25-inch (1.2 Mbyte) disks of each were included. The Medicine Bag disk was not required during installation. The installation process adds a memory-resident program to CONFIG.SYS, and executes a batch file from within AUTOEXEC.BAT.

Default installation was straightforward. Drive C was scanned, checksums were calculated, and all specified files were copied across to the hard disk of my test computer. This machine also uses drives D, E and F, all of which were studiously ignored by the installation process. The date/time stamp attached to each installed file was set by the installation process to be the current date/time: this makes it impossible both for a user to tell at a glance which version is in use, and to repeat an installation process exactly (without resetting the clock). This is a less-than-clever feature which should be scrapped.

I then reinstalled *Doctor* using the 'custom' installation procedure, and was asked a series of configuration questions concerned with the location of various files, time of scanner execution, password protection, sound effects, etc. It seems possible to tailor each installation in any way, excepting (as explained above) replication of a previous installation.

There is a problem with checksum verification after the custom installation process has been used. When the PC is next rebooted, the file CHECKSUM.EXE fails its own checksum test. I ran CHECKSUM again, and more files failed the checksum test. This was very odd, and was cured only by remaking the checksums manually: I could find no explanation for this.

Although the developers call this product '*Doctor*', it is obvious that its origins lie in a package called '*Virus Buster*': the programs CHECKSUM.EXE, LIST.EXE and INSTALL.EXE all refer to *Virus Buster* within the executable file, and the installation program even produced an error message stating: 'Can't read Virus Buster Help File'. If the name of a product is changed, the developers should ensure that all references are altered.

### Scanning

The scanner can check any combination of low memory, the boot/system areas of a disk, and the files on disk. The files inspected can be specified using a system of multiple overlapping file specifications. This works well, and permits complex but flexible scans to be initiated.

On default settings, the scanner inspected my test computer's hard disk (249 files, 11.6 MB) in 36 seconds. Unless the number of scanned files is reduced (for example, altering the file specification list to prevent scanning of SYS and BIN files), nothing reduces scanning time significantly: the default settings seem to be very close to the fastest settings.

Under *Windows*, the two scan times quoted increase to 38 seconds, and 2 minutes 1 second respectively. In comparison, *Dr. Solomon's AVTK* scanned the same hard disk in 23 seconds. *Sophos' Sweep* took 25 seconds in fast mode, and 1 minute 13 for a complete scan. Although *Doctor* is not the fastest scanner around, its scanning speed is acceptable, despite being some 50% slower than its competitors.

### Accuracy

When *Doctor* was tested against the viruses in the current *Virus Bulletin* test-set, it failed to detect just ten in 'Quick Check' mode (Suomi, Macho, Power, Sibelsheep (COM and EXE), Halley, Invisible (COM and EXE), NukeHard, and Willow): a detection rate of 96%. When 'Slow' mode was activated, *Doctor* also detected Suomi. Manually setting *Doctor* out of 'Quick Check' mode meant that Halley and Willow were also detected, increasing detection to 97%. I have no idea why these two tests give different results, and can find no explanation in the documentation.

With the exception of Suomi and Macho, the main detection problems were with more recent additions to the test-set - one hopes that the developers will catch up quickly. *Doctor* was impressive when tested against Mutation Engine (MtE) samples, detecting all 500 of the samples correctly.

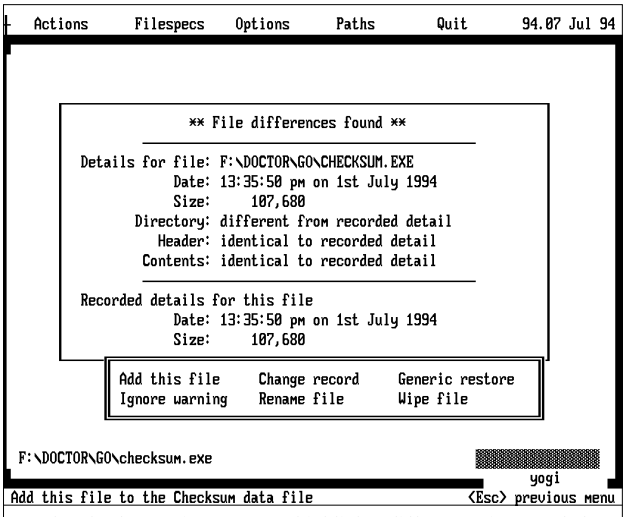
I was surprised to see a message stating 'Found using Offset scan' appear onscreen before details of several test viruses were displayed. This happened with 24 samples: the text message was displayed about five times for each of the viruses, even when the command line switch instructing *Doctor* to operate in 'Hidden' mode was activated. This seems to be a line of debug code which has been left active within the scanner; either way, it should be removed as soon as possible.

### Checksumming

When *Doctor* calculates a file's checksum, it records the date/time of last update, the file size, and an indication of file contents. *Doctor* took 15 seconds to calculate the initial checksums for a hard disk containing 71 files. Subsequent verification of checksums, unsurprisingly, took the same length of time as the initial checksum calculation. This calculation operates by default on the drive where *Doctor* has been installed, whilst the scanner operates by default on drive C (which may or may not be the same drive).

When the checksummer is asked to operate on drive C, it calculates a checksum for its own file image which is resident on a different disk drive. Whilst the checksum program is executing, a message appears on the screen stating that it has 'scanned' a certain number of files - not, however, in the normal virus-specific sense of the word.

The checksummer, like the scanner, has a 'Quick Check' and a 'Slow' mode, though the time taken to verify a set of checksums appears identical no matter which mode is in use. The manual claims that the checksummer only looks at the 'top few kb and the bottom few kb' for each file: I cannot reconcile that with the fact that checksum verification seems to proceed at the same speed no matter what mode is used, and am confused about precisely what differences exist between the two modes.




The checksumming program highlights differences in recorded file detail, but failed its own checksum on installation.

### Memory-resident Monitor

The memory-resident monitor provided with *Doctor* is a 'behaviour monitor and blocker'. It occupies 11.1 Kbytes of low memory, not 9.75 Kbytes as claimed in the product's 'Evaluator's Notes': this overhead can be vastly reduced, however, by loading it into high memory. By default, the monitor attempts to prevent modification of critical disk areas, scans disk boot sectors whenever a disk is accessed (including during a reboot), and scans programs as they are executed and/or copied. Facilities can also be invoked which claim to be able to write-protect COM and EXE files, and to stop direct (non-DOS) writing to disk.

I tested the memory-resident monitor program by copying the entire test-set from one disk drive to another. *Doctor* detected just 45 of 247 (17.5%) infected files: 4K (2 samples), Flip (2), Cascade (2), Fu_Manchu (2), Jerusalem (2), Dark_Avenger, Vienna (3), Sunday (2), Tiny (5), Vacsina (8), Yankee (6), Slow, Nomenklatura, Eddie, Anticad (2), Virdem, 1575, Datalock (2), and Butterfly. Results do not improve when files are renamed to have an executable extension, and do not include the fact that MtE infected files can be copied *ad infinitum* without detection.

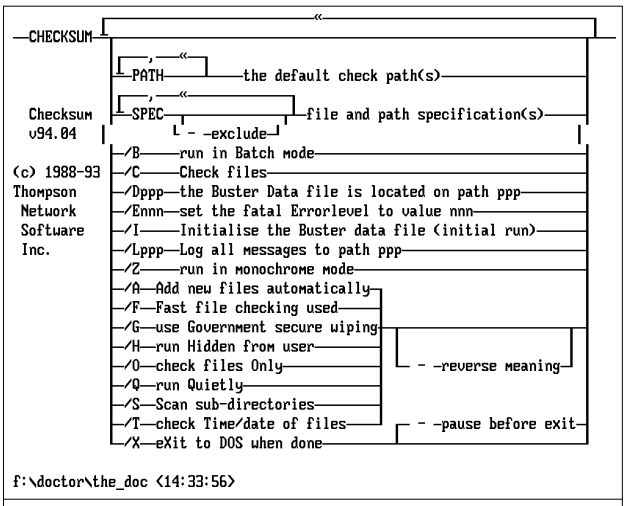
When the memory-resident monitor is scanning for viruses, it uses only those signatures contained in a special signature database file. This must be held in memory, in a file just 4.4 Kbytes long. The documentation states that this only 'contains signatures for non-polymorphic, wild viruses': it omits to say that only 74 virus signatures are included.

The monitor often fails to detect those which are actually present. Alabama, Burger, Keypress, Spanish Telecom, and SVC are present in the database file, but *not* detected when a file infected with one of these viruses is copied. Curiously, only five out of 12 samples of the Tiny virus family were detected - clearly, these were not considered a major threat.

### Inside the Medicine Bag

*Doctor* includes an extra floppy disk called the 'Medicine Bag' which, amongst other things, includes a program called VUPDATE which claims to be capable of dialling the Bulletin Board operated by the developers of *Doctor* and downloading new data files, thereby updating the scanner. I liked this idea, and was interested to see how it worked.

Unfortunately I ran into several problems whilst testing it. It would not dial with my modem (which is attached to the COM3 serial port), and whenever dialling was attempted, VUPDATE simply returned to the main screen. Most unfortunately, attempting to dial a number also had the effect of removing the help system, so that the F1 key produced only a blank screen, just when you need it most!




The *Virus Buster* name still appears in some of the *Thompson Network Software's Doctor* files.

If the communication speed is set to 38400 bps (the speed my modem normally uses), VUPDATE produces a stream of errors, all of which state 'Error - 2 opening COM port 3', and then locks up so thoroughly that Ctrl-Alt-Del is needed to regain control. There were further problems: the settings are saved if the program is exited using the Esc key (which should abort execution), and some keys operate the wrong way round (e.g. Cancel = Start). The plain fact is that this program is not yet finished.

The Medicine Bag disk also contains a program called DOCLITE, apparently a cut-down version of the *Doctor* scanner. No documentation is provided other than a help screen listing of the command line switches, but it seems to work correctly, and takes the same time to scan the hard disk of my test computer as the main scanner. A switch is provided to scan inside ZIP files, but does not appear to do anything, and creates no log entries mentioning its usage.

Finally, the Medicine Bag has a program called BOOTCHEK, and users are warned that this program alters the partition record of the hard disk. It claims that it 'checksums a system's MBR and DBR for integrity checks against boot viruses', and therefore cannot be subverted by a stealth virus. I wish I knew what a DBR was. 'DOS Boot Record' perhaps? The BOOTCHEK help file uses such acronyms without any explanation.

I freely admit that I did not test BOOTCHEK on my test computer: by this time it seemed that programs were left in the 'Medicine Bag', and not included within the main part of *Doctor*, because they were not finished. Life is too short to sort out the havoc a program can create if it fouls up the MBS of a hard disk. My hard disk…

## Other Features

Infected files can be erased, either 'normally' or by 'Government Rules' (overwritten three times). The scanner includes facilities which purport to 'clean' viruses from infected files. Being consistent with my oft-stated proclivity for erasing or replacing files rather than trying to cobble them back together again, I can recommend only 'erasure' features, and did not test the 'cleaning' features.

The utility program which extracts a signature from a file is not quite what it seems: its purpose is to extract a signature from a clean file, and to add that signature to the database of the memory-resident monitor. This file can then be added to a list of 'allowed' files; it will not be possible to execute any others. The term 'signature extractor' is more often used to refer to a utility which can automatically obtain a reliable virus detection string from an infected file: such features have, in the past, proven notoriously optimistic in their aims and claims.

## Conclusions

Overall, I find that *Doctor* suffers badly from a lack of attention to detail. Leaving the old name scattered around in various places is unforgivable - users find anti-virus products bewildering enough already without seeing confusing references to other products. Likewise, adding an extra disk containing sparsely-documented programs which still appear to have serious problems is, to say the least, helping nobody.

In common with many other memory-resident anti-virus programs (*VB*, Sept 1993, pp.15-19), the results obtained from the TSR monitor are poor; so poor that I cannot recommend its use. *Doctor's* insistence on resetting its own files' date/time stamps is a further cause of annoyance.

These problems are a shame: *Doctor* is not difficult to use, and has a reasonable (but not brilliant) scanner and an efficient checksumming program. With more polish and further development, this product could be a useful addition to the anti-virus market.

**Technical Details**

**Product:** *Doctor*

**Developer/Vendor:** *Thompson Network Software*, PO Box 669306, Marietta, GA 30066-0106, USA, Tel. +1 404 971 8900, Fax +1 404 971 8828.

**Availability:** A PC with 512 Kbytes of RAM and 800 Kbytes of available disk space. A hard disk is recommended.

**Version evaluated:** v94.07, July 94

**Serial number:** Nothing is written on the master disk, even though there is a space for such information on the disk label.

**Price:** US$129 per PC. Site licences available.

**Hardware used:** A 33 MHz 486 clone with 4 MBytes RAM, one 3.5-inch (1.4 Mbyte) floppy disk drive, one 5.25-inch (1.2 Mbyte) floppy disk drive, and a 120 MBytes hard disk, running under *MS-DOS v5.00* .

**Viruses used for testing purposes:** This suite of 158 unique viruses (according to the virus naming convention employed by *VB*), spread across 247 individual virus samples, is the current standard test-set. A specific test is also made against 500 viruses generated by the MtE (which are particularly difficult to detect with certainty).

For a complete list of test-set viruses, see *VB,* Feb 1994, p.23.

# PRODUCT REVIEW 2

## Norman Firebreak

*Jonathan Burchell*

*Firebreak*, another in the ever-growing range of anti-virus NLMs, is developed by *Norman Data Defense Systems*, a Norwegian company which has in recent years begun to distribute its long-established product more widely. Regular readers will remember that the DOS-based anti-virus software produced by this company, *Norman Virus Control*, has already made its *VB* debut (*Virus Bulletin*, May 94 p.17), and performed relatively well. Does the NLM match its little brother's scores?

### What You See…

The product arrives on one 3.5-inch floppy disk, together with an attractively-presented A4 manual. No 5.25-inch media is included in the package: if this is required, it must be specified when ordering. As with many companies now, the 3.5-inch disk has become standard, with the 5.25-inch floppy rarely provided as a matter of course.

*Firebreak* is designed to provide real-time and background virus detection capabilities for *NetWare 3.11* servers and above. Although the product is compatible with *NetWare 4.0*, there are no specific support features for that system. None of the documentation indicates whether or not *Firebreak* has been *Novell*-tested and approved.

No support is provided for *Macintosh* virus detection; the product is limited to scanning DOS and *OS/2* namespace. With many large companies now running mixed-platform networks, this may be a drawback for some sites.

*Firebreak*, which comes with a workstation behaviour blocker, is the NLM face of *Norman's* anti-virus software; no virus-specific workstation protection is provided. Local technical support is available only in Norway, the USA, Malaysia, the UK and Denmark, so, when a user outside these areas requires assistance, he must either make an international call or contact the company via the *Internet*, or *CompuServe*. I chose to ring their head office in Norway when a small incompatibility between the NLM and my versions of CLIB.NLM arose.

The standard of service provided was excellent: everybody who answered the telephone seemed to speak perfect English, and a new version of the product was immediately forthcoming, via the *Norman* BBS. Full marks for support here - particularly as it was a long time before I revealed that I was reviewing the product.

*Firebreak* is designed only to provide virus signature detection for *Novell* file servers; it does not provide file checksums or integrity checking. DOS and *Windows* workstation scanners from the same company integrate with *Firebreak* inasmuch as the NLM can act as a centre for collating messages, either from other copies of *Firebreak* on different servers, or from the workstation scanners.

### Installation

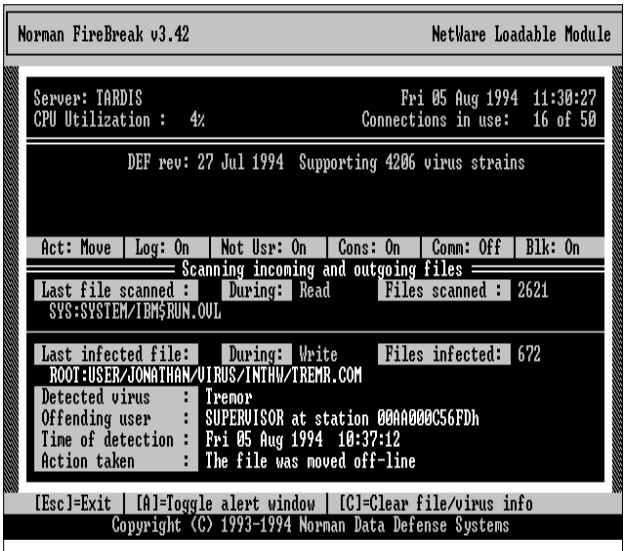
Software installation is clearly documented in the manual, and a simple install program largely automates the procedure. It is necessary to be logged in as supervisor and to map a drive letter to SYS:\ prior to starting the install program. The only question which needs to be answered about the process is which drive is mapped to SYS:\, information which is available via the network interface. Why isn't this done automatically?

*Firebreak* installs itself into a directory directly under SYS:\ and creates two subdirectories; one for log files, the other for quarantined files. This is not configurable, although if Secure Console mode is enabled, *Firebreak* will install correctly to the SYSTEM directory.

### Administration

Once installed, *Firebreak* is started by loading a single NLM, or by typing 'FB', which is the name of a *Novell* NCF file created by the install routine.

The only interface to *Firebreak* is via the system console screens, which can be accessed via RCONSOLE. *Norman Data Defense Systems* does not provide any kind of workstation-based software for configuration and administration, although, as the options to *Firebreak* are relatively limited, this is not surprising.




Real-time scanning allows specification of the type of file access to be scanned: here, incoming and outgoing files are checked.

### The Main Menu

From the main menu, the user may instigate a scan of the server, or access the *Firebreak* configuration option (described in detail below), the monitor screen, the virus library, or the logging and messaging procedures.

The server scan starts a background scan of executable files on all volumes of the server. No options are provided for time-scheduling scans, or for limiting the scan to certain volumes, directories or files - it's simply all or nothing. A particularly annoying feature is that the list of file extensions cannot be changed; it is inbuilt in the software.

This is not acceptable. Firstly, the list includes only files with the extensions APP, BIN, COM, DLL, DRV, EXE OVL, OVR SYS, VBX, and STP: I would like to see OV? and 386 included. Secondly, it is annoying not to be able to specify new names. For instance, the virus test-set is held on disk with extensions renamed to CO1, EX1, etc: a convenient way of holding infected files on disk whilst preventing their accidental execution. Not being able to specify the name means they must all be renamed; a tedious process.

It is not possible to pause scanning and then resume it, nor are any options provided to limit the amount of server power devoted to scanning.

The monitor screen displays a real-time screen showing *Firebreak* activity, and details of suspected infections or messages received from other *Norman* products.

The virus library is a reasonable on-line encyclopaedia which identifies viruses by name and gives brief information on their activity and payload.

### Configuration of Firebreak

Due to the relatively simplistic nature of *Firebreak*, setting up the product is straightforward, and limited to configuration of start-up, real-time scanning, manual scanning and action to take on virus detection.

The start-up configuration menu consists of various options, including 'enter Monitor directly', which sets *Firebreak* to start up at its monitor screen. The 'enable screen blanker' is another choice - *Firebreak* has its own screen blanker for the console; however, the manual documents a number of eccentricities of using this option and rconsole together. To be fair, these are rconsole problems, not those of *Firebreak*. Finally, the 'use 24 hour format' will set entries in the log file to the 24-hour clock, as used on the Continent.

The Real-time Scanning menu allows the user to specify the types of server file accesses which are to be scanned in real-time. Three choices are provided: incoming files, outgoing files, and files which have been renamed.

Similar to the background scanner, Real-time Scanning does not provide options to change the list of file extensions, or to specify volumes, directories, and files to be specifically included or excluded. The only other option controls whether or not virus incidents are logged to a file. The file has a preset name and location and cannot be changed.


*Firebreak's* virus library lists variants of any known virus, in addition to describing their activities and payloads.

Manual scanner configuration offers very few options: the only configuration items for the background or manual scanning are whether or not virus incidents should be logged in to a file (which, like the Real-time scanner, has a preset name and location and cannot be changed) and, if so, whether to clear the file each time, or append to it.

When a virus is detected, the user may move the affected files to the quarantine directory (which is preset as VIRUS under the *Firebreak* directory), purge them from the server or rename them. The renaming scheme ensures that even if multiple files of the same name are infected, or if they are left in place and renamed, they will not collide in the quarantine directory.

### Logging and Messaging

The user may also configure the message section to specify who will be notified in the event of an infection, and what they will be told. It is possible to select notification messages to go to the user, to a specified user group, and to the system console and/or a printer queue.

Extensive facilities are provided to edit the message to be sent in the event of a server and a workstation alert. Any message may be fully customised: although limited to a length of 56 characters by *NetWare*, it is possible to substitute dynamically into the message useful information such as the name of the infected file, the user's physical location, the user's name, the infected server name and the virus name. This should allow appropriate messages to be constructed for everyone who will receive them.

A test facility allows the user to try out the messaging system, and to check that everything is properly configured. A nominated special group can be specified to receive alert messages - the NLM is actually rather clever, and monitors

logins from this group. If a user belongs to the group, and an infection occurs whilst that user is logged out, he will receive the alert message upon logging back in.

Finally, it is possible to specify all messages to be sent to a communications hub. This would normally consist of another copy of *Firebreak* running on a central server.

Another option controls whether or not information from the last session is preserved across various loads of the NLM; others handle the saving and loading options from the configuration file, and the ability to specify a password to prevent unauthorised menu access. It is notable that the manual thoughtfully documents which file must be deleted in the event that the password is forgotten!

The NLM builds up reports in several log files. Examples of the type of entry which will be created are given in the manual: this is just as well, as no facilities to view, file, or print the log files are provided by the NLM. The documentation on file, which is far from complete, will get the user started on writing a report generator, but nothing more.

### The Results

*Firebreak* scored well on both the 'Standard' and the 'In the Wild' test-sets, but failed to reach the required 100% for the 'In the Wild' viruses. The polymorphic results are indifferent, with the product neither failing spectacularly nor achieving a detection ratio worthy of note - this places it very much in the middle ground as far as polymorphic detection is concerned. Both background and real-time scanning produced exactly the same results.

### Conclusions

*Firebreak* is rather a difficult product to place. The detection ratio shows great promise, but, because of its mediocre results in detection of polymorphic viruses, it is not good enough to be placed at the front of the pack.

It is a fairly simplistic product, and lacks certain important features; for example, the ability to set where to scan by volume or directory. In addition, the reporting of the log file is inadequate. However, those features which are implemented seem to be done well. In fact, it has the clean, elegant feel which one has come to associate with all things Scandinavian in design.

*Norman* might well argue that many configuration options are simply unnecessary, and that the product provides everything which the user needs in order to protect a network. While there is some truth in this argument (who needs a scheduled scan when the real-time scanner uses the same scanning engine?), the lack of configuration options forces the network manager into using the product in a specific way. I would feel much happier about the product if this issue was addressed. However, if the idea of a low maintenance, easy-to-configure NLM-based scanner appeals, *Firebreak* may well be worth a second look.

## Norman Firebreak

Detection Results (Secure mode):

NLM Scanner

| | | |
|---|---|---|
| Standard Test-Set[1] | 225/229 | 98.2% |
| In the Wild Test-Set[2] | 107/109 | 98.2% |
| Polymorphic Test-Set[3] | 340/450 | 75.6% |

Scanning Speed:

Speed results for an NLM product are inappropriate, due to the multi-tasking nature of the operating system. Full comparative speed results and overheads for all current NLMs will be printed in a forthcoming *VB* review.

**Technical Details**

**Product:** *Norman Firebreak version 3.42.*

**Developer:** *Norman Data Defense Systems*, PO Box 633, Tangen, N-3002 Drammen, Norway. Tel. +47 3281 3490, Fax +47 3281 3510.

**US Office:** 3028 Javier Road, Suite 201, Fairfax VA 22031, USA. Tel. +1 703 573 8802, Fax +1 703 573 3919.

**Price:** NorKr 8,900 for the server alone; NorKr 3,900 for server plus workstation software for five users. Prices in the UK are £495 per server, with unlimited users, including bi-monthly updates. Volume discounts available.

**Hardware used:** Client machine - 33 MHz 486, 200 Mbyte IDE drive, 16 Mbytes RAM. File server - 33 MHz 486, EISA bus, 32-bit caching disk controller, *NetWare 3.11*, 16 Mbytes RAM.

Each test-set contains genuine infections (in both COM and EXE format where appropriate) of the following viruses:

[1] **Standard Test-Set:** As printed in *VB*, February 1994, p.23 (file infectors only).

[2] **In the Wild Test-Set:** 4K (Frodo.Frodo.A), Barrotes.1310.A, BFD-451, Butterfly, Captain_Trips, Cascade.1701, Cascade.1704, CMOS1-T1, CMOS1-T2, Coffeeshop, Dark_Avenger.1800.A, Dark_Avenger.2100.DI.A, Dark_Avenger.Father, Datalock.920.A, Dir-II.A, DOSHunter, Eddie-2.A, Fax_Free.Topo, Fichv.2.1, Flip.2153.E, Green_Caterpillar.1575.A, Halloechen.A, Helloween.1376, Hidenowt, HLLC.Even_Beeper.A, Jerusalem.1808.Standard, Jerusalem.Anticad, Jerusalem.PcVrsDs, Jerusalem.Zerotime.Australian.A, Keypress.1232.A, Liberty.2857.D, Maltese_Amoeba, Necros, No_Frills.843, No_Frills.Dudley, Nomenklatura, Nothing, Nov_17th.855.A, Npox.963.A, Old_Yankee.1, Old_Yankee.2, Pitch, Piter.A, Power_Pump.1, Revenge, Screaming_Fist.II.696, Satanbug, SBC, Sibel_Sheep, Spanish_Telecom, Spanz, Starship, SVC.3103.A, Syslock.Macho, Tequila, Todor, Tremor (5), Vacsina.Penza.700, Vacsina.TP.5.A, Vienna.627.A, Vienna.648.A, Vienna.W-13.534.A, Vienna.W-13.507.B, Virdem.1336.English, Warrior, Whale, XPEH.4928

[3] **Polymorphic Test-Set:** The test-set consists of 450 genuine samples of: Coffeeshop (375), Cruncher (25), Uruguay.4 (50).

# BOOK REVIEW

## Solomon Says...

*Dr Solomon's PC anti-virus Book*, by Alan Solomon and Tim Kay, is another in the growing library of computer virus books. Solomon is well-known in the anti-virus world, and has spoken extensively on viruses; Kay (an ex-employee of *S&S*, now working as a freelance consultant) has been involved in the area for some years. The two have combined their expertise to produce a book which looks at viruses, anti-virus measures, and related legal issues.

### Style and Content

The book includes sections on virus definition, history, anti-virus software, virus prevention, disassemblies, post-infection cleanup, and legal considerations, a glossary and an index. The glossary is adequate, the index less so: a random check revealed that three out of five references for one entry were incorrect. A free 'virus calendar' (for 1994-1995), a chart showing dates when various viruses can be expected to trigger, is also provided.

The book was obviously not written as a reference work for the IT manager, but with the layman in mind: there is little technical detail, and what there is, is explained in a fashion which is clear and, for the most part, concise.

However, grammatical, typographical and stylistic faults abound: this is unfortunate, as the subject is dry enough without being made more turgid through what, in isolated instances, amounts to downright unreadability! Luckily, this is generally balanced by the authors' relaxed style.

### Anti-virus Software

Anti-virus products offer many features, which can confuse the user: Solomon and Kay discuss the options, and Kay tests several products for performance. The authors excluded the *S&S Anti-Virus Toolkit* from the tests on the grounds that, because it was developed by Alan Solomon, it could not be regarded as a fair and objective inclusion.

Testing took a slightly different format to that to which users have become accustomed: in addition to the obligatory testing against a virus library, Kay actively infected his PC with Stoned, Jerusalem.Standard, and Maltese Amoeba. Such a test is interesting, because it reflects what might happen on a real world machine, although it is usually avoided as it is difficult and time-consuming.

The authors view the number of viruses detected by a product as less important than removal of every instance of the virus detected, and their main criteria for testing was to see how the software performed in a situation where files have been infected by a small number of viruses.

The test results make interesting reading, although it is disappointing that there is no conclusion drawn from them. More information is also required on the testing protocol in order to understand fully quite what some of the figures given actually mean.

### Viruses

Solomon and Kay devote some 40 pages to disassemblies, describing nine viruses in detail. There is no obvious logic to the choice: for instance, why Brain *and* Ashar, two viruses from the same family? There are also certain unexpected omissions: one would expect analyses of the more widespread and/or destructive viruses (e.g. Form, Michelangelo, Spanish Telecom). That said, each virus included is well-documented and clearly explained.

### Legalities

This last chapter starts with a definition of computer crime, and describes legislation in the UK. A lengthy discussion of European law follows (much of which is dedicated to proposed and draft legislation), with America also given a cursory glance. The chapter is based on articles by Ms Wendy London, who writes on computer law for *Secure Computing*, and practises in the field.

It is interesting that the authors have explored the legal issues in such depth. These topics should concern many people, but few books have made a separate study of the area. Overall, this was the most interesting section of *The PC anti-virus Book*, and represents a useful collection of snippets and opinions on legal matters.

Surprisingly, the book ends with this chapter; with neither general recapping nor summarising comments - I rather had the feel of being left hanging.

### Conclusion

This is an acceptable tome, as such go: if you do not have a general reference work on the subject, it is a useful acquisition - of particular interest are the sections on product testing and the law. Apart from this, however, there is little which is new: if you possess one of the many books published, or most back issues of *VB*, you will already have much of the information. Although informative, it was uninspiring - not a 'must buy', but nevertheless not a waste of money.

Title: *Dr Solomon's PC anti-virus Book*
ISBN: 0-7506-1614-8
Authors: Alan Solomon and Tim Kay
Publisher: *Butterworth-Heinemann*
Price: £24.95

**SUBSCRIPTION RATES**

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

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

**Editorial enquiries, subscription enquiries, orders and payments:**

*Virus Bulletin Ltd*, 21 The Quadrant, Abingdon, Oxfordshire, OX14 3YS, England

Tel. 01235 555139, International Tel. +44 1235 555139
Fax 01235 559935, International Fax +44 1235 559935
Email virusbtn@vax.ox.ac.uk
*CompuServe* 100070,1340@compuserve.com

US subscriptions only:

June Jordan, *Virus Bulletin*, 590 Danbury Road, Ridgefield, CT 06877, USA

Tel. +1 203 431 8720, Fax +1 203 431 8165



# END NOTES AND NEWS

**A new storm is brewing regarding virus code on the *Internet*.** A service provider, *Netcom On-Line communications Inc*., is allegedly allowing one of its users to distribute virus code over its network. *Netcom* said it was investigating the claims, but had so far turned up no evidence of illegal or improper behaviour. The argument between the anti-virus researchers and *Netcom* looks set to grow, with no easy solutions on the horizon: if service providers regulate the information on their network, they will stand accused of violating the First Amendment; if they do nothing, they leave themselves open to claims that they were responsible for any damage caused by the viruses.

*LAN/SEC Europe*, a technical conference devoted entirely to local area network security, will be held in London from 27-29 September 1994. Details from Hazel Richardson, *Euromoney Publications plc*, London. Tel +44 (0)171 779 8793, Fax +44 (0)171 779 8795.

According to reports in the popular computing press, **the future of *Novell DOS 7*, which includes an anti-virus component, is in doubt**. However, Graeme Allan, the brand marketing director for *Novell UK*, refutes the gossip: '*Novell UK* is continuing to proactively promote *DOS 7* as an important product offering. We plan to announce a new channel marketing programme for *DOS 7* shortly.' The company will also be making an announcement in September, outlining corporate strategy and development plans for the future.

The *International Symposium on Computer Crime Prevention*, due to be held in Beijing, China on 25-27 October 1994 has been cancelled due to unforeseen circumstances. The organisers hope to arrange an alternative date in the near future.

***Cybec Pty* have issued another virus alert**. Lemming, a COM and EXE file infector, targets various anti-virus products, and has a destructive trigger routine; Trakia 570, _1099, and Gingerbread variant are file infectors.

The *21st Annual Computer Security Conference and Exhibition*, under the auspices of the *Computer Security Institute*, will take place in Washington DC from 14-16 November 1994. It will feature network security, client/server, open systems, and the latest technological developments, and will incorporate the *Computer Security Products and Services Exhibition*. Tel. +1 415 905 2626.

**Patricia Hoffman's *VSUM* listings for August 1994 - DOS-based scanners:** 1. *Command Software's F-Prot Professional*, 97.1% (9406), 2. *McAfee Associates' ViruScan 9.30 v117*, 96.4% (9408); 3. *Dr Solomon' AVTK v6.64*, 93.9% (9406); 4. *IBM Anti-Virus/DOS v1.05*, 86.8% (9406); 5. *Norton Anti-Virus v3.0 AO6 Upd*, 77.7% (9406). **NLM Anti-Viral Products:** 1. *McAfee NetShield 1.6 v117*, 95.2% (9408); 2. *Dr Solomon's AVTK NLM v 6.64,* 93.4% (9406); 3. *Command Software's Net-Prot v1.25*, 83.2% (9406); 4. *Norton Anti-Virus NLM v1.0AO6*, 76.4% (9406); 5. *Central Point AV/NLM v2.0*, 60.4% (9403). The number in parentheses refer to the month when that version of each product was first certified - in some cases, this indicates a product over five months old.

*Reflex Magnetics Ltd* are holding a 'Live Virus Experience - Overview' on 14 September, 7 December, and 14 March (1995). Advanced 'Live Virus Experiences' will take place on 8 December 1994 and 15 March 1995. For further information, contact Julia Vockrodt at *Reflex Magnetics*. Tel. +44 (0)171 372 6666.

*Compaq Computer Corporation*, according to an article in *PC Week* (25 July), is due to launch **a range of *Deskpro* PCs which will include *SafeStart*, built-in anti-virus protection.** The program is said to scan executable files as well as the boot sector, and to use a TSR module and virus-signature recognition to detect unknown viruses. *Compaq*, when contacted, said they could neither confirm nor deny the report, which was unauthorised by them.