ISSN 0956-9979 FEBRUARY 1995

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **Infectious reading.** This month sees the last instalment of *Virus Bulletin's* four-part series on infection techniques currently used by virus authors.

• **The wheels of justice…** Both news stories this month concern lawsuits: the man alleged to be the 'Black Baron' has been charged in the UK with distributing viruses. On the corporate side, *Carmel Software Engineering* has lodged a suit against *Central Point Software* in the US Federal Court.

• **Look out, polymorphics!** *AVAST!*, an anti-virus product still relatively unknown outside its home territory, obtained stunning detection rates against polymorphic viruses in last month's Comparative Review. The product is examined in detail on page 21.

CONTENTS

EDITORIAL

NT - New Threats?

Just like every month, this edition of *Virus Bulletin* contains a long list of new viruses. The majority of these are either minor variants of existing viruses, or ones produced by a virus creation tool such as PS-MPC or VCL. Of those remaining, only one or two contain innovations or new ideas. However, every now and then a truly interesting virus is discovered, which changes the way anti-virus software must operate in order to protect the PC.

In the category of 'interesting' viruses are those which do not spread under DOS, and require operating systems like *Windows* or *OS/2*. As computing gradually shifts away from DOS towards such point-and-click user interfaces, the next battleground between virus writers and anti-virus software developers is being laid out.

"DOS security is something of an oxymoron"

The most obvious reason for the glut of PC viruses is the large number of *IBM PCs* or compatibles which are in use. Just as important a consideration is the fact that DOS is an extremely easy operating system for which to write viruses. There is little or no file protection (the read-only flag being more of a reminder than a directive), and any process has control of the entire machine. DOS security is something of an oxymoron.

Another factor to take into account is the large amount of information about the innards of the *IBM PC* and its operating system. Books such as *Undocumented DOS* and *Undocumented PC* provide an excellent source for writing code which pushes the platform to its limits... techniques which can be used for destruction as well as for construction. Programming tools are freely available - DOS even comes with a primitive debugger and assembler.

Few would dispute that writing viruses for other PC operating systems is more difficult. Development tools are harder to obtain: they are frequently shipped on CD-ROM, and sometimes actively restricted. Even if one manages to get hold of a development system, details of the internal workings of the operating system are often undocumented or difficult to obtain. The user-friendly graphical front-end hides mind-boggling layers of complexity just below the surface, making low-level patches to the kernel rather a tricky procedure.

The practical effect of this shift in operating system design is the reduction in the number of programmers who have the necessary resources to create viruses for *OS/2* and *Windows NT*, which should in turn reduce the number of viruses written. This will be a good thing for everyone, as the main squeeze on resources currently is the sheer number of viruses which arrive every month. The downside is that viruses which will turn up will almost certainly have been written (at least in part) in a high-level language, making analysis difficult.

The existing crop of boot sector viruses is extremely unlikely to spread on *OS/2* or *NT* machines. However, the majority will attempt to infect the fixed disk, causing a great deal of unintentional damage in the process. Virus authors tend to make assumptions about how the hard disk of a PC is laid out, and what parts of the disk hold critical information; experience has shown that it is well worth taking a copy of *all* partition boot sectors and the Master Boot Sector, in case the unthinkable should occur.

While switching to a new operating system changes the threat from computer viruses, it will not render them impotent, regardless of what the operating system vendors claim. No matter how clever the software running on the PC, the machine is vulnerable to attack by viruses at boot time. Only by changing the BIOS of the PC can this loophole be fixed - it is surprising that even on those machines which provide this additional functionality, few people use it. Furthermore, many viruses are more dangerous under operating systems like *NT*, where changes to the MBS can render the system unbootable. Suddenly even a comparatively innocuous virus such as Stoned can cause unrecoverable damage. Even though new operating systems may go some way to reducing the number of viruses in circulation, it is sadly a safe bet that they will not put an end to the problem.

NEWS

Man Charged with Virus Distribution

A man alleged to have written the viruses Pathogen and Queeg was charged under the *Computer Misuse Act* in Plymouth on 16 January 1995. Christopher Pile, of Plymouth, has been released on bail to appear at Plymouth Magistrate's Court on 14 February 1995.

The unemployed 26-year-old faces ten offences related to the distribution of Queeg and Pathogen: these viruses were written by a person calling himself the 'Black Baron', and have spread widely throughout the UK. They are now also in the wild elsewhere in the world.

DS Simon Janes, of *New Scotland Yard's Computer Crime Unit* (*CCU*), told *Virus Bulletin* that over one hundred instances of damage caused by the virus had been reported.

Pile's appearance in Plymouth next month, according to DS Janes, is likely to be little more than an exchange of paperwork between the Counsel for the Defence and the *Crown Prosecution Service*, with another date for commencement of the trial to be set then. It is expected that Pile will face his charges four to six weeks after his next court appearance.

This case follows hot on the heels of the recently-publicized story in Norway, where a young man was charged with unauthorized distribution of viruses, after uploading samples onto a public Bulletin Board System (see *VB*, December 1994, p.3).

Anyone with information likely to assist in the prosecution of the 'Black Baron' (or indeed any virus author) is urged to contact the *CCU* on 0171 230 1177 (UK only) ▮

Carmel Alleges Breach of Contract

Carmel Software Engineering, a computer software company based in Haifa, Israel, has filed a lawsuit in US Federal Court against *Central Point Software* (now part of the gigantic *Symantec Corporation*), alleging breach of contract.

Jay Zinns, *Carmel's* lawyer (of the New York-based legal firm *Jacobs, Zinns & Braff*), has let it be known that the case was brought only after months of unsuccessful negotiation between the two companies.

The suit has been filed because *Central Point*, under its new owners *Symantec*, 'has failed to live up to its obligations to market *Carmel's* product and has marketed and sold anti-virus software that competes with the software developed, enhanced and maintained by *Carmel*.'

Carmel Software Engineering claims that it has had an agreement since 1990: *Central Point* bought the rights to *Carmel's* anti-virus products, and the latter was to maintain and enhance the software. In return, *Central Point* would market and sell the packages. Both sides were happy with the way the contract proceeded, and renewed it in late 1993. However, it is alleged that unknown to *Carmel*, *Central Point* was at that time negotiating its merger with *Symantec*, the producer of *Norton Anti-Virus*.

Despite the pending court battle, *Carmel* insists that it will continue to maintain and update the products as it has for the past four years. *Carmel* is seeking damages in excess of US$6 million.

Michael Sweeny, Director of Public Relations for the *Symantec Corporation* commented 'We were surprised and disappointed by *Carmel's* announcement. The suit is entirely without merit, we believe, and we intend to vigorously defend our position. *Symantec* and its *Central Point* division remain committed to selling and supporting all-out current anti-virus products.' ▮

Virus Prevalence Table - December 1994

Virus	Incidents	(%) Reports
Form	20	23.5%
Anti-CMOS	7	8.2%
CMOS4	7	8.2%
Parity_Boot	7	8.2%
JackRipper	5	5.9%
Monkey_1	4	4.7%
Anti-EXE	3	3.5%
Cascade	3	3.5%
Natas	3	3.5%
Spanish_Telecom	3	3.5%
Viresc	3	3.5%
Angelina	2	2.4%
Monkey_2	2	2.4%
Ontario	2	2.4%
Tequila	2	2.4%
V-Sign	2	2.4%
1575-b	1	1.2%
AMSE	1	1.2%
B1	1	1.2%
Dinamo	1	1.2%
Diskwasher	1	1.2%
Keypress-1216	1	1.2%
PS-MPC-5	1	1.2%
Sayha	1	1.2%
Stoned	1	1.2%
Stone-p	1	1.2%
Total	85	100%

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 21 January 1995. 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

Adin.3026
CER: A polymorphic virus, probably of Russian origin. When decrypted, the strings 'AIDSTEST.EXE' and 'ADIN' are revealed. No simple search string is possible.

Australian_Parasite.Split
CER: This group of four viruses should perhaps be classified as a separate family; however, as they seem to be written by the same author as the other Australian_Parasite viruses, they have been put into that group. The viruses are 1033, 1035, 1135 and 1149 bytes long and are slightly polymorphic, preventing the use of simple search strings.

Better_World (D and F)
EN: Minor variants, detected with the Better_World (Fellowship) pattern.

Bootexe.443
ER: Detected with the BFD pattern.

Burger.560.AV
CN: Minor variant, detected with the Burger pattern.

Cascade
CR: Three new minor variants (1701.AB, 1701.AC and 1704.AA) detected with the Cascade_(1) pattern.

Chaos.1181
ER: There are now three new minor variants (J, K and L) of this virus. They are all detected with the Chaos (formerly Spyer) pattern.

DSME
DSME is a polymorphic 'engine', like the MtE, which can be used to create variable-length polymorphic viruses. There are currently four viruses known which use DSME: Apex, Connie, Demo and Teacher. As other polymorphic viruses, they cannot be detected with a simple search string.

Datalock.920.L
ER: Minor variant, detected with the Datalock pattern.

E-Morph
CR: A 1696-byte virus containing the text strings 'E-Morph' and '[1993 por JMC Ver 1.1]'.

```
E-Morph             80FC CA75 04B4 AD9D CF3D 004B 7406 9D2E FF2E 0F01 5053 5152
```

Father_Mac.1470
CR: An encrypted virus containing the text 'Poner aca el texto deseado'. There is another related variant, 1496 bytes long.

```
Father_Mac.1470     BE26 01B9 9606 89FF 80C5 0081 E926 0188 C089 F626 8A02 80C4
Father_Mac.1496     88C9 B9A6 0681 E928 0188 E488 D226 8A02 88E4 80EF 0034 2089
```

Fax_Free
ER: Quite a few variants of this virus have appeared recently, including Mosquito.B, Mosquito.C (both detected with the Mosquito pattern) and Topo.B (detected with the Topo pattern).

FFFF
CR: There are two viruses in this family, 432 and 440 bytes long. Both are detected with the Attention pattern, but are considered sufficiently different to justify placing them in a separate family.

Flash.688.D
CER: A minor variant, detected with the Flash pattern.

Gonads
CN: This 1781-byte virus contains a long message which seems to be written by one VxBBS SysOp as an attack against another.

```
Gonads              B440 B9F5 068D 9403 01CD 21B8 0042 9933 C9CD 218B 8438 082D
```

Granada
CR: A 2765-byte encrypted virus.

```
Granada             B9B6 0ABB 0501 32C0 3007 43FE C0E2 F9E9 40FF
```

HLLC
P: Several new 'companion' viruses: 4768.A and 4678.B, 9261, 17690.B and Captain (which is a variable-length virus). Also reported are 14880, Unvisible.A and Unvisible.B, 'renaming companions', which rename and replace the original program.

HLLO
CEN, EN: Three new overwriting viruses, written in Pascal or C. They are 4240, Hepatitus (variable size), Joker.B (10000) and Tyst (3424).

HLLP

CEN, CN: There are several new parasitic HLL viruses this month: 3779, 4075, 4568, 4942, 4984, 5000, Birthday.5824, Birthday.7808, LouLou, Vova.8896 and Vova.9904.

Hungarian_Andromeda

CER, CR: The first virus reported in this family came from Hungary and contained the word 'Andromeda'; hence the name (Andromeda alone was already in use as a virus name). There are now seven variants with infective sizes of 725, 758, 800, 1024 (two variants) and 1536 (two variants) bytes. The three smallest variants only infect COM files, but the others infect EXE files as well.

```
H_Andromeda.725     FA80 FC4B 7503 E9F8 0080 FC30 7509 81FE FDCD 7503 BFCD ABFB
H_Andromeda.758     FA80 FC4B 7503 E9E8 0080 FC30 7509 81FE 3412 7503 BFDD FFFB
H_Andromeda.800     FA80 FC4B 7503 E94F FE80 FC30 7509 81FE FECD 7503 BF3D 1BFB
H_Andromeda.1024.B  FA80 FC4B 7503 E97D FE80 FC30 7509 81FE B4A3 7503 BFA3 A3FB
H_Andromeda.1024.C  FA80 FC4B 7503 E932 FE80 FC30 7509 81FE B4A3 7503 BFA3 A3FB
H_Andromeda.1536.A  FC4B 7503 E9B3 FD80 FC30 7509 81FE A3B4 7503 BFCA BDFB 2EFF
H_Andromeda.1536.B  FA80 FC4B 7503 E98D FD80 FC30 7509 81FE B4A3 7503 BFA3 A3FB
```

Hymn.Sverdlov.C

CER: A minor variant, detected with the Sverdlov pattern.

I_Am

CN: Polymorphic, variable-length virus. No simple search pattern is possible.

IVP

CEN, CN: This month brings some new IVP-generated viruses: 705 (CEN), Angry_Samoans.B (668, CN) and Bad_Friday (986, CEN).

Jerusalem

CER: There are four new variants: 1808.Dashes, 1808.Zeros (detected with the Jerusalem-1 pattern), AntiCad.2900.ABT.B (detected with Acad.2576) and Tarapa.D (detected with Jeru-1735).

Kela.823

CR: Contains the text 'KELA lives Don Kr. 1992'.

```
Kela.823            B996 008B FE2E 8BA4 1303 6A00 078B D683 C70C 2E8B A413 032E
```

KeyKap

P: A family of four 'companion' viruses; 685, 923, 1074 and 1077 bytes long. The 1077-byte variant contains the text 'KeyKapture Virus v0.90 [Hellspawn-II] (c) 1994 by Stormbringer [P/S]'. The others contain similar messages.

```
KeyKap.685          1E8C D848 8ED8 C606 0000 5A81 2E03 0020 0181 2E12 0020 0133
KeyKap.923          BB79 00B4 4ACD 21B4 48BB C000 CD21 2D10 008E C026 C706 F100
KeyKap.1074/1077    E91A 02AC 0AC0 75FB 817C FC45 5874 4481 7CFC 434F 7532 807C
```

Keypress.1232.N

CER: A minor variant, detected with the Keypress pattern.

La_La

CEN: This is a 1895-byte encrypted virus which contains the texts: 'Ohhhh La La!', 'Mommmmy, they're teasing me again', and 'Shut up you little sonsuvbitches'.

```
La_La               BB00 002E 8A04 2E30 813F 002E 8A81 3F00 89FE 29C6 434E E2EB
```

Leprosy

CEN: Two new 666-byte variants of this virus have appeared, Leprosy.666.M and Leprosy.666.N. Both are detected with the Leprosy-B pattern. There are also a few other variants which require new patterns.

```
Leprosy.570         8B0E 0B02 51E8 0F00 5BB9 3A02 BA00 01B4 40CD 21E8 0100 C3BB
Leprosy.664.A       535B BA00 01B9 9802 B440 CD21 E813 02C3 BB2E 018A 2732 0606
Leprosy.666.K       E81A 00E9 5801 8B1E 5F02 53E8 0F00 5BB9 9A02 BA00 01B4 40CD
Leprosy.5120        BB30 01B9 3015 8A27 3226 0601 8827 43E2 F5C3 8B1E EE01 53E8
```

Little_Red.B

CER: A minor variant, detected with the Little_Red pattern.

Monte_Carlo

EN: There are two variants, 1483 and 1541 bytes long. They are encrypted and slightly polymorphic. Although detection with a single pattern containing wildcards is possible, it is not recommended.

Murphy.Pest.B

CER: Almost identical to the original Murphy.Pest, which has now been renamed as Murphy.Pest.A. Detected with the HIV pattern.

PDV

CN: An unremarkable virus containing the text '/ PDV /'.

```
PDV                 0AC9 7408 81BE EE03 4D5A 7503 B137 C3B8 0242 33C9 33D2 CD21
```

Phunnie

CN: A simple, non-resident, 311-byte virus.

```
Phunnie             B440 8B9C 3502 B906 00BA F301 03D6 CD21 B442 B000 8B9C 3502
```

Pit

CN: A 611-byte virus, containing the text 'The Pit v1.20'.

```
Pit                 BF17 02B9 0300 8A24 3825 7505 4647 E2F6 C333 F6BF 0603 8B05
```

Pixel.200

CN: The shortest Pixel variant known. Detected with the Pixel.273 pattern.

Proto-T.599.B

CN: This is a minor variant, detected with the Proto-T.599 pattern, but the text message it contains has been changed to 'Cosmo Virus V1.0 (c) 1994 Written By [Mad Satan] in Taipei. TAIWAN.'. The virus author who calls himself 'Mad Satan' seems to be one of the least talented programmers in the entire virus-writing community: all of his viruses are merely minor variants of viruses written by other people, with his name inserted.

PS-MPC There is a fairly large number of new PS-MPC-generated viruses this month, which is surprising, considering that practically every virus scanner should be able to detect them all. The new variants are: 338.D (CN), 430 (CR), 510 (CEN damages COM files), 520 (EN), 564.B (CEN), 565.E (CEN), 565.F (CEN), 565.H (CEN), 569.E (CEN), 569.G (CEN), 570.E (CEN), 570.F (CEN), 570.G (CEN), 573.J (CEN), 573.K (CEN), 578.I (CEN), 578.J (CEN), 578.K (CEN), 578.L (CEN), 578.M (CEN), 579.D (CEN) and Weak (983, EN).

Rajaat.700 **ER:** A 700-byte virus which contains the word 'Rajaat'.

```
Rajaat.700          FF36 8E03 8F06 6B03 FF36 8C03 8F06 6D03 FF36 8603 8F06 6F03
```

Scorpio **CER:** This is a 1000-byte Bulgarian virus, which contains the words 'Plovdiv' and 'Scorpio'.

```
Scorpio             3DAA FF74 663D 004B 74DB 80FC 3D74 1480 FC43 740F 80FC 1174
```

Semtex.1000.D **CR:** A minor variant, detected with the Semtex pattern.

Shirley.C **ER:** A minor variant, detected with the Shirley pattern.

SillyC **CN:** The viruses in the SillyC family are not necessarily related, but they are all small, non-resident viruses with no obvious names, which only infect COM files.

```
SillyC.128          8BD6 8BCD B440 CD21 B43E CD21 03F5 B44F CD21 73BE C32B D22B
SillyC.166          CD21 89C3 B803 E988 2699 0028 069A 00B4 3FB9 0300 5ACD 21B8
SillyC.169.B        B9A9 008D 9600 01CD 21B8 0042 33C9 33D2 CD21 B440 B904 008D
SillyC.264          B905 008D 9607 02CD 21B8 0042 33C9 8BD1 CD21 81BE 0702 5A4D
SillyC.320          B907 00B4 4ECD 21EB 04B4 4FCD 2173 123C 1275 0B83 FDFF 7406
SillyC.331          8CC3 81C3 1010 068E C3BF 0200 BE00 01B9 4B01 F2A4 B98A 0007
SillyC.343          BAB1 00B9 0300 CD21 B800 4233 C92E 8B16 9A00 83C2 29CD 21B4
SillyC.498          8D1E D402 B001 8407 7410 B801 438D 16DD 0230 ED8A 0F80 F101
SillyC.563          4747 C705 2A2E 4747 C705 636F 4747 B86D 0089 05B8 CFB0 A318
SillyC.626          E850 01B4 4EBA 6C03 B9E7 00CD 21E8 4301 7347 EB33 908B 3E62
```

SillyCR **CR:** The only significant difference between this family and the SillyC viruses is that the SillyCR viruses are all memory-resident. Two of the viruses, 261 and 264 bytes long, are very closely related.

```
SillyCR.131         5053 521E 80EC 4B75 43B8 023D CD21 723C 93B9 0300 0E1F 33D2
SillyCR.200         601E 063D 004B 7569 6A60 07B4 43CD 2180 E1FE B801 43CD 21B8
SillyCR.239         9C1E 60B8 023D CD61 7274 8BD8 33C0 8ED8 B43F BAEF 02B9 EF00
SillyCR.261/264     3D19 9274 0D80 FC4B 7503 E808 00EA ???? ???? 86C4 CF50 5351
SillyCR.330         3D00 4B74 0B80 FCFF 7402 EBEF B8F0 F0CF 9C50 5351 5256 5706
SillyCR.357         80FC 4B74 03E9 AA00 5053 511E 5206 8BDA B902 00B0 2E38 0774
SillyCR.563         3D00 3D74 0A3D 004B 7405 EA?? ???? ??53 5106 5756 508B DAB9
```

SillyCER **CER:** Another 'pseudo-family', containing resident viruses which infect EXE as well as COM files.

```
SillyCER.307        3D00 4B75 3D50 5351 5256 571E 06B8 023D CD21 7226 8BD8 0E1F
```

SillyER **ER:** A group of resident viruses with no obvious name, or special effects, which infect only EXE files.

```
SillyER.323         80FC 4B74 0580 FC3D 75F1 6006 1EB8 0043 CD90 B801 4360 2BC9
```

Sterculius.440.B **CER:** A minor variant, detected with the Sterculius.440 pattern.

Stranger **CN:** A 734-byte overwriting virus, with a rather strange structure. The virus is encrypted and contains the text '*Stranger*'.

```
Stranger            8CD0 4850 BE03 008B C681 C600 018B DE56 81C6 2100 8BFE B9BD
```

Sum **CER:** A 1441-byte virus which infects EXE files in the normal manner, but overwrites code in the middle of COM files.

```
Sum                 21FC B900 01B8 0000 8BD8 AC30 C700 C380 D700 E2F6 81FB BD08
```

Suriv-1.April_1st.F **CR:** A minor variant, detected with the Suriv_1.01 pattern.

SVC.1064 **CER:** There are two new minor variants (B and C) this month, both detected with the SVC 3.1 pattern.

Timid.313 **CN:** Detected with the pattern published for Timid.306.

Tokyo.1060 **EN:** Detected with the Tokyo pattern.

Troi.F **CR:** A minor variant, detected with the Troi pattern.

VCL **CN:** This month brings the following variants: 420 (not encrypted, detected with the VCL.VoCo pattern), 551 and Renegade.5738. There is also one companion virus: Rat (665, detected as VCL.VoCo).

VCS.Standard.Test2 **CN:** An insignificant virus, created with the primitive VCS tool. Detected with the VCS 1.0 pattern.

Yankee_Doodle **CER:** There are two new Yankee_Doodle variants this month, 2433 and 2561. Both are detected with the Yankee pattern.

INSIGHT

Low Temperature Viruses

Megan Palfrey

Igor Muttik is a physicist. In fact, his speciality is precision physics: this was his first area of research, in the Low Temperature Laboratory at *Moscow State University*. Since he gained his honours degree at that institution in 1985, his work there has continued, but his areas of interest and expertise have broadened considerably. Now, they consist primarily of the usage of computers in physics experiments, data collection, education, and networking. In addition to all this, he lectures at the university.

So where do viruses come in?

'For me, anti-virus research was just a hobby; it was not my original job. I'm mostly interested in exchanging opinions and ideas with researchers all over the world. As far as I know, there is no official virus research at the University - the closest would probably be the Mathematics Department, or the Department of Calculations and Cybernetics.'

From Father to Son to PC Virus

Although Muttik views himself now as 'definitely mostly PC-based', this was not always the case: his introduction to computing in general was via the huge mainframes with which his father works, also at *Moscow University*, in the Chemical Department.

The next step was taken when he became supervisor of a minicomputer, a machine for which he found a strong affinity. 'Ten years ago it was a powerful enough machine, but I think now the company which built it, *Norsk Data* of Norway, only produces software. Perhaps this is because of heavy competition.'

The route led finally to PCs. The first *IBM-PC* compatible appeared in Moscow in 1986, a clone made by US *Columbia Data Products*. 'Our laboratory was one of the first, maybe in the USSR, but definitely at *Moscow University*, to get a PC. It was a relatively slow machine with no hard drive, but it meant the beginning of a new era for us.'

It was not until two years after this, in 1988, that Muttik first encountered a computer virus. A professor in the Low Temperature Laboratory was more than surprised to see letters falling down to the bottom of his monitor screen, and called Muttik for help.

'I had heard reports of the existence of computer viruses, so I took an executable file and did a disassembly; I was interested in what I would find' - the virus, of course, was Cascade, and proved to be only the first of many which Muttik has disassembled since then.

He sees his workplace as an ideal breeding ground for viruses: '*Moscow University* is actually a very interesting place; it has many departments and a great many students - around 15,000. Almost all of these students and the staff have frequent contact with people all over the world. That, of course, is a very good environment for viruses! Many have been found here, and most of them have made their way into my collection.'

Use and Development

In the course of his research, Muttik has tested many products, including one written by fellow Russian, Eugene Kaspersky (*AV Professional*: see *VB*, April 1994, p.6; August 1994, p.18), which Muttik uses for maintaining his virus collections. His hobby has led him all over Europe in search of 'new and interesting concepts' in the field.

Muttik is particularly impressed with the research of Christoph Fischer (of *Karlsruhe University*), who has created a virus analysis environment on a UNIX machine: 'Using this emulated platform, Fischer tries to understand whether there is a malicious problem, and how to deal with it. I believe this is very useful, and if we begin to add artificial intelligence to this system, it may give very vluable results.'

> *" I believe that the future is in the use of artificial intellect in heuristic scanning"*

It was perhaps not entirely unexpected that Muttik himself would also develop a product: his efforts have resulted in a tool for anti-virus researchers, one which is especially dedicated to support the problem of generation of 'goat' files for a virus.

'A goat file,' he explained, 'is a "do-nothing file"; a file which simply prints a short message containing information about its contents - for example, file type (i.e. COM, EXE or SYS) and size (in decimal and in hexadecimal). The file can then be infected by the virus. The tool can generate many thousands of identical files, a factor which is particularly useful in the case of polymorphic viruses. Basically, what I have produced is something to help researchers create a test-bed of viruses.'

Writer versus Researcher

Muttik regards the investigation and dissection of an unusual new computer virus as more interesting than the creation of the virus itself. At the moment, he is studying viruses which are targeted against particular anti-virus programs; specifically, he mentioned 3APA3A (see *VB* November 1994 p.9), a virus

which appears to have its origins in Russia. '3APA3A is designed to defeat a Russian integrity checker; one which, unfortunately, doesn't check the internal DOS files,' said Muttik.

'This technique, targeting specific products, has become very popular recently - in fact, I see it all as a game between the virus writers and the anti-virus researchers, such as myself: the virus writers are constantly trying to create something new which will defeat us. We view it as an intellectual challenge to get the better of them.'

'Unfortunately, even in a laboratory, you are only likely to see one or two really different specimens in the course of a year. For myself, I think it would be sad if there were no interesting new viruses. This research is very stimulating. For the users, of course, quite the opposite is true.'

Although he does not know any virus writers personally, he has been in contact with a few via E-mail. It is fortunate, he said, that most of these people are not very professional: 'A huge amount of the viruses in any collection are badly written: they account for as much as 99% of the time which developers of commercial products spend on research and disassembly. If a more innovative virus would appear, it would at least be more challenging for us.'

Muttik sees virus-writing as something which has become almost an established part of the computer world, and firmly believes that most virus authors attempt to create a virus only to prove that they have the ability to produce something which actually works.

'If they had more knowledge,' he asserted, 'they could make much more useful things; even commercial programs - they would certainly gain more satisfaction this way. I don't see an easy solution to this problem, and I'm not sure what is the best way to deal with someone who is caught writing viruses. Some people require punishment; for others, it suffices just to be taught why it is wrong to write a virus. I think it depends on the person.'

The New Wave

Polymorphic viruses have heralded the advent of heuristic scanning, a new method which Muttik views as a far more flexible type of anti-virus protection than traditional virus-specific software: 'It is not possible to assemble or identify polymorphic viruses using scan strings. Also, heuristic scanning serves two important purposes: it can predict the behaviour of viruses, and even detect the not-yet-existent samples. I believe that the future is in the use of artificial intellect in heuristic scanning.'

Muttik sees polymorphic viruses becoming an even greater problem than they are now, with more difficult and more sophisticated viruses being written, and with virus creation toolkits adding yet another weapon to the virus author's arsenal. To balance the issue, Muttik believes that anti-virus software will also evolve, becoming equally more sophisticated and 'intelligent', and able to overcome the challenge thrown down by virus authors.

Igor Muttik and Björg Ólafsdóttir, taking a break at the *1994 EICAR Conference* in St Albans.

Looking Forward

Anti-virus research has gradually become a prominent part of Muttik's work, and now occupies some 80% of his time, including that spent in the Low Temperature Laboratory. As his job is directly computer-related, however, such research is not perceived as far removed from the core of his work: 'Our headquarters are satisfied with the situation, especially since there are no plans to start a new department for anti-virus research.'

Unsurprisingly, with the knowledge he has acquired along the way, Muttik has become known at the University as their resident anti-virus expert. He sees this development continuing and growing: 'I believe that the main field of my work in the future will be anti-virus research, although I don't know how long this will last. Things can change so quickly - remember, the first *IBM-PC* compatible appeared only ten years ago.'

Muttik predicts much more powerful computers with inbuilt security in the next ten years. It is likely, he thinks, that most of them will run DOS applications in an emulation mode, and that viruses could find it difficult to work in such environments. As most systems will be compatible, however, he does not forecast their demise.

'I expect that the virus problem will be around for at least the next ten years, and that it will be countered by many different types of software. Heuristic analysis, integrity checking, authentication, and also data encryption systems, data security systems - these are the type of things we can expect to see leading the field in anti-virus protection.'

DOS viruses in the wild have a definite future for the next ten to fifteen years, in Muttik's view - and this makes his career into the next millennium a certain one.

VIRUS ANALYSIS 1

Anarchy Reigns

Eugene Kaspersky
KAMI Associates

The latest newcomer to the Anarchy family is Anarchy.9594 (its predecessor is Anarchy.2048). It is 'in the wild' in Russia, and is unusual in several respects.

Firstly, it is 9594 bytes long - nine times longer than the 'average' virus, which normally occupies no more than 1000 bytes. This is rather a nasty surprise for the researcher, as the time which is spent on analysis grows with the code length. Secondly, on execution of the infected file, it can be seen that the virus decreases system memory by 83 kilobytes. The 'average virus' would usually occupy one or two kilobytes of system memory.

Installation

Anarchy is polymorphic: when a file infected with this virus is executed, control passes to the decryption routine at the beginning of the virus code. The decryption loop then recovers the code, passing control to the installation routine.

The virus first checks to see which version of DOS is present, returning control to the host program if it is lower than DOS 3.0. Then it calls the undocumented Int 2Fh Installation Check function (AX=1200h) to ensure that other undocumented DOS functions are also available.

Next, the virus makes an 'Are you there?' call, performing Int 21h with AX=ABCDh. If Anarchy finds itself already memory-resident, the resident portion returns FFFFh in AX register, and the virus returns control to the host program. It then executes a second 'Are you there?' call, using Int 21h and AX=0CC0h. The resident portion returns C00Ch in AX register. I see no reason for this duplicate 'Are you there?' call, but both are present in the virus code.

The virus decreases system memory by 83K, using the standard manipulation of Memory Control Blocks and PSP fields, and copies its code there. Before doing so, it checks that the size of system memory is sufficient. It then hooks Int 09h (the keyboard handler), DOS idle interrupt Int 28h, and Int 21h, stores the pointer to the DOS critical address with an Int 21h/AH=34h call (that value is used in the pop-up trigger routine) and returns control to the host program.

Int 21h Handler

The virus' Int 21h handler intercepts several functions: 'Are you there?' calls (AX=0CC0h, ABCDh), FindFirst and FindNext calls (AH=11h, 12h, 4Eh, 4Fh), Close File (AH=3Eh), Read (AH=3Fh), Lseek End (AX=4202h) and Execute (AX=4B00h).

During FindFirst/Next calls, the virus adjusts the length of the infected file, so that it appears unchanged from the original. The virus identifies infected files by increasing the year in the date stamp by 100.

When reading (Int 21h, AH=3Fh) from an infected file, the virus carries out standard stealth operations, with the result that the file appears uninfected. On LSeek End calls (Int 21h, AX=4202h), the virus adjusts the position of the file pointer, as if the file was not infected. These actions do not suffice for a virus with full stealth abilities to hide itself completely, because such major functions as Write and FCB access functions are not intercepted by the virus; however they do for Anarchy, since its author has only attempted to make it a 'semi-stealth' virus.

"after the forty-eighth infection … the virus displays one of four messages and halts the computer"

The virus pays special attention to the CHKDSK utility, and to the archivers ARJ and PKZIP. If the Int 21h call is performed by one of these, the virus disables its stealth routines on FindFirst/Next, Read and LSeek calls. Had this not been done, CHKDSK would display a warning message, and ARJ and PKZIP would compress the files incorrectly.

When the DOS call Close File (AH=3Eh) is executed, control passes to the infection routine. The virus does not call the infection routine directly on file execution, but opens the file, then closes it immediately with the Close File function. The virus' Int 21h handler thus intercepts control twice on file execution (Execute, and then Close File calls). When the second call is made, the virus 'hits' the file; infecting it on execution as well as on closing.

On infection and on accessing infected files with Read and LSeek functions, the virus separates infected and uninfected files, by using the text 'UNFORGIVON' which is placed at the end of infected files.

Infection

The infection procedure is standard, bringing nothing new to the palette of parasitic infectors. The virus hooks Int 24h to prevent DOS error messages on writing to write-protected disks, and stores and restores the file date and time stamp (but adds 100 to the year value).

Anarchy checks for file name and extension and infects all COM and EXE files, excepting COMMAND.COM. It then reads the file header, separates COM and EXE files by noting the MZ stamp at the beginning of EXE files, and encrypts itself with its polymorphic routine. The virus writes the result of the

encryption to the end of the file and corrects the file header, writing JMP VIRUS to COM file headers, and setting corresponding values in the EXE header fields.

The polymorphic routine is quite simple: the virus XORs itself with a key which does not change in the encryption/decryption loop, and stores several constant instructions diluted with simple junk bytes such as NOP, CLC, STI, SAHF, DEC/INC BX/DX:

```
PUSH  ES
PUSH  CS
POP  ES
PUSH  CS
POP  DS
MOV  DI,  offset  encrypted_code
MOV  CX,  246Dh
PUSH  SI
POP  DI
LODSB
XOR  AL,  key
STOSB
LOOP  loop_offset
```

In COM files, the JMP VIRUS code has two instructions, DEC BP and JMP NEAR VIRUS. An unencrypted string 'JAN FAKOVSKIJ,USSR,1994' follows the JMP VIRUS code in COM files. On infection, the virus uses undocumented Int 2Fh calls to access the System File Table, changing the file open mode from Read-Only to Read/Write, and checking whether the files are placed on remote disks - if so, they are not infected.

Trigger Routine

After the forty-eighth infection in the same sequence, the virus displays one of four messages and halts the computer. Two of the messages are in Russian; the other two are:

```
DIS  IS  DI  END,
BEAUTIFUL  FRIEND...
DIS  IS  DI  END,
MY  ONLY  FRIEND-
DI  END.
IT  HURTS  TO  SET  U  FREE,
BUT  U'LL  NEVOR  FOLLOW  ME.
DI  END-OF  LAUGHTER  &  SOFT  LIES,
DI  END  OF  NIGHTS...WE  TIRED  TO  DIE...
DIS  IS  DI  END

I  WANNA  DESTROY  DA  PASSORS-BY
'CAUSE  I  WANNA  BE,-
YEAH,-    ANARCHY
```

When Alt-Minus (on the numeric keypad) is pressed, the virus calls a trigger routine emulating the file shell *à la Norton Commander*, allowing it to create, copy, move, and/or delete files and subdirectories, and edit text files.

The function keys also allow manipulation of files and subdirectories: F3, change current drive/directory; F4, edit file; F5, copy file/directory; F6, move file/directory; F7, create a file; F8, delete file/directory; F10, exit; Ins, select/deselect file/directory. Unfortunately, the virus has no help files to enable detailed explanation of its functions.

The virus contains other internal text strings which are used in infection and trigger routines:

```
R/O Arc Sel <  DIR  >
SELECTED  FILE(S)
COMMAND  COMARJ.EXE  PKZIP.EXE  CHKDSK.COM
OFFSET  LINE  COL  SIZE
EOL  EOF  INS
YESNO
ERROR  DISK  FULL
```

Conclusions

Although this is not a new type of virus, it is much longer than the 'average' virus, and more time is required to disassemble it. It is to be hoped that this does not indicate a new trend.

Anarchy.9594

Aliases: Unforgivon.

Type: Memory-resident parasitic, polymorphic file infector with stealth capabilities.

Infection: COM and EXE files.

Self-recognition in Files: At the end of infected files, the string 'UNFORGIVON' is found. The virus also adds 100 years to the date stamp of infected files.

Self-recognition in Memory: Two 'Are you there?' calls from Int 21h. Input AX values are ABCDh and 0CC0h; the memory-resident virus returns FFFFh and C00Ch in AX.

Hex Pattern in Files: No search pattern possible in files. The text string 'UNFORGIVON' is located at end of infected files.

Hex Pattern in Memory:

```
80FC  4E74  0580  FC4F  750F  E8D8
FF72  E6E8  4BFF  74E1  E88D  FFEB
```

Intercepts: Int 21h for infection, stealth and trigger routine, Int 28h and Int 09h for main trigger routine.

Trigger: The memory-resident copy of the virus keeps count of the number of files it has infected since becoming active. When this count reaches forty-eight, the virus displays one of four different messages. The virus also emulates the file shell of *Norton Commander*.

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

VIRUS ANALYSIS 2

Parity_Boot

Benjamin Sidle
Sophos Plc

Parity_Boot.A and Parity_Boot.B are two rather unremarkable boot sector viruses which are becoming increasingly prevalent in the wild. Both variants are very similar in function and code; indeed, it is possible to detect them both using the same search pattern.

The main difference between them is that variant 'A' stores a copy of the original Master Boot Sector (MBS) in sector 14, side 0, cylinder 0 of the fixed disk, whereas variant 'B' copies it to sector 9, side 0, cylinder 0. Although this is a small difference, it is vital to distinguish between the two for the purpose of disinfection. The remainder of this article will discuss the Parity_Boot.B virus.

On Start-up

When a machine is booted from an infected floppy disk, the virus first examines the contents of the fixed disk's MBS. After checking for the marker 55AAh at the end of the sector, the virus checks whether the byte at offset 01BCh is C9h. If that is not the case, the hard disk is deemed suitable for infection, and the virus copies its own code to the MBS of the disk.

"the processor is halted with the HLT instruction - the only option is to turn the machine off"

When an infected system is booted, the virus stores part of the 24-hour ticker timer count for later use. It also stores the address of the current Int 13h handler and reduces the amount of DOS memory by 1K, reserving it for installing the virus code. Parity_Boot then hooks both Int 13h (BIOS Disk Services) and Int 09h (BIOS keyboard).

Finally, Parity_Boot issues an Int 19h (soft reboot), the purpose of which is to start the booting procedure again, but this time using the virus' Int 13h and Int 09h functions. As Parity_Boot is a stealth virus, this loads the original boot sector into memory, and passes control to it.

Resident Operation

When an attempt is next made to read the MBS, or the boot sector of a diskette, the virus checks to see whether or not the sector is already infected: if it is not, the infection process begins. This doubles as a stealth routine: every call to read or write to the Master Boot Sector of the fixed disk will be redirected.

The virus clearly shows its age: it does not even attempt to handle 1.44M floppy disks (the values default to those for 1.2M diskettes). For hard disks, the MBS is copied to the area mentioned above. On diskettes, the virus uses part of the BIOS Parameter Boot block area (the number of sectors per FAT) to calculate where to store the original boot sector.

Trigger

The activation and execution of the virus' payload is handled by the new Int 09h handler. If Ctrl-Alt-Del is pressed, the virus will simulate a memory parity error and perform a warm reboot.

During every call to Int 09h, there is a chance that the virus will enter its second trigger routine. If the clock count byte stored during boot-up is less than the current value, the message 'PARITY CHECK' (disguised within the virus by XORing the text with the value 55h) is displayed, and the processor is halted with the HLT instruction - the only option is to turn the machine off. If neither condition is met, the virus' Int 09h handler jumps to an absolute offset within the BIOS (which is a fixed entry point for all BIOSes), and makes no effort to use the original Int 09h handler.

Parity_Boot

Aliases:	None known.
Type:	Memory-resident boot sector virus with stealth capabilities.
Infection:	Master Boot Sector on hard disk, boot sector on diskette.
Self-recognition on Disk:	Checks offset 01BCh for the value C9h.

Hex Pattern:	Parity_Boot.A
	`FA0E 1FA1 4C00 A3E1 7CA1 4E00`
	`A3E3 7CA1 2400 A3BA 7DA1 2600`
	Parity_Boot.B
	`FA0E 1FA1 4C00 A3D3 7CA1 4E00`
	`A3D5 7CA0 6E04 A2BD 7DA1 1304`

Intercepts:	Int 13h for infection, and Int 09h for the trigger routine.
Trigger:	Displays the message 'PARITY CHECK' and halts the processor.
Removal:	Boot from a clean system disk using the same version of the operating system as installed on the hard disk, and use the DOS command FDISK /MBR.

FEATURE

Testing Protocol: The DOS Comparative Review

One of the most contentious issues which *Virus Bulletin* has to tackle is setting out the correct way in which to evaluate a large number of DOS-based scanners. The original *VB* testing protocol (*VB*, April 1991, pp.6-7) is now extremely long in the tooth, and many of the issues which influenced it have changed. This new draft of the testing protocol is not intended as a final statement; it is a starting point for discussion and development.

The Scope of the Review

Much as one would like it to be the case, the *Virus Bulletin* comparative reviews of scanners cannot possibly consider all aspects of their operation. Several important issues, listed below, are not addressed.

The most obvious omission is that of testing the products in an infected environment. Several products (in particular, those which have only a *Windows* component) cannot be executed without running software stored on the fixed disk. More importantly, the EXEBug virus is capable of preventing a clean boot on machines with certain types of BIOS, making it impossible to guarantee clean system conditions when the product is run.

It is therefore imperative that such products can detect the presence of this virus when run with their default options set. Time constraints prevent this test from being carried out, although it is hoped that it can be added at some later date. However, in order to test this properly, the test machine would need to be equipped with a BIOS which is fooled by the virus.

The second obvious omission is that of tests on file disinfection. *Virus Bulletin* has carried out this type of test once before (*VB*, September 1994), examining only the most widely-used products, or those which scored particularly well in comparative reviews. Disinfection tests are time-consuming and, although important, cannot realistically be included in a regular scanner comparative review.

A Reason for Change

Perhaps the most important issue to be addressed by the new testing protocol is that of polymorphic virus detection. As the number of polymorphic viruses continues to rise, one very telling measure of a scanner's overall performance is the vendor's ability to keep the scanner up to date. For this reason, the way in which the total detection score is calculated for the polymorphic test has been altered. In order to differentiate more clearly between those scanners which detect 99% of all replications of a particular polymorphic virus, and those which can detect every incidence, 25% of the total marks on this test-set will now be awarded for detecting all samples of each virus type.

Thorny Issues

A number of problems still remain with the current test procedure. The most difficult to address are those products which have a 'review mode'.

Consider a scanner which changes the way it operates when it has detected more than (for example) three different viruses on any one scan, switching over from 'turbo' mode to 'full'. Such a scanner would do very well in both speed and detection tests: on a clean system, the product would run in its fastest mode of operation. On an infected system, the scanner would operate in its most secure mode.

Although there is a good argument that this is a perfectly legitimate way in which to implement a scanning engine, it may give rise to somewhat misleading results. If one were to run such a scanner on an incoming disk containing just a single virus, the scanner would operate in its 'quick' mode, and potentially miss a virus which it was able to detect in the *Virus Bulletin* test. Furthermore, it would be trivial for a vendor to design a scanner which, if it detected a review situation, identified *any* file which looked remotely like a virus as infected, upping its detection results accordingly.

There are a number of possible solutions to this problem. The simplest is to run each scanner against only one virus sample at a time. This is impractical from the point of view of testing time. Another solution is to intersperse infected files with clean ones. Although this addresses part of the problem, it also has the potential of making tests confusing, requiring lengthy searches through log files. This would be an acceptable workload for testing one scanner, but very difficult for twenty-seven.

The most practical solution is for all products to be subjected to 'spot checks' on single infected files. Although this will not detect all possible 'review modes', it should limit the problem.

Conclusions

The current *VB* testing procedure used in comparative reviews tests only certain aspects of scanner operation. However, by defining which areas are to be tested, it is hoped that the tests can be discussed and improved, by accepting input from both industry and product users.

Anyone wishing to comment upon the testing protocol outlined here is urged to contact the Editor, either by fax (+44 (0)1235 531889) or by Email (virusbtn@vax.ox.ac.uk).

VB Comparative Evaluation of DOS Scanners

Product Category: Non-resident virus-scanning software running under DOS.

Objective: To provide the essential criteria by which to judge the relative speed and accuracy of virus scanning programs, on both clean and infected files.

Component Tested: Only the non-resident scanner is tested. No tests are carried out on TSRs or checksummers provided with the package. All tests are carried out in a clean DOS environment - no tests are conducted with viruses active in memory[1]. No specific tests are made on the product's virus removal capabilities[2].

Hardware: The hardware used is specified. All comparative testing is carried out on the same machine with the same basic configuration. Changes made by the product to system start-up files are left in for the duration of the tests made on that product. Full details of the hardware used in the tests are supplied. The operating system used is the latest version of *MS-DOS* currently shipping.

Virus Test-sets: The virus test-sets are supplied by *Virus Bulletin*. They consist of genuine infections of computer viruses stored in a form which can replicate under normal conditions. No first generation samples or droppers are included, unless they themselves are created by a virus which is included in the test-set.

File viruses are attached to a standard goat-file whenever possible. In the 'Standard' and 'In the Wild' test-sets, a sample of both COM and EXE files is included, where appropriate. The 'Polymorphic' test-set is made up of multiple samples of a small selection of polymorphic viruses, all of which can replicate.

Boot sector viruses are supplied individually on a genuinely infected floppy diskette. Details of the test-sets used are given at the end of each comparative review.

Tests: Each product included in the review undergoes six tests. These are:

- Scan time for a clean diskette
- Scan time for an infected diskette
- Scan time for a clean *Bernoulli* drive[3]
- Scan time for an infected *Bernoulli* drive
- Scanner detection of file viruses
- Scanner detection of boot sector viruses

Testing methodology: All detection and speed tests are carried out with the scanner run in its default mode of operation. The only exception to this is when the virus scanner has no well-defined default mode, or when it does not produce a log file for further reference. Where the default mode is not defined, the exact configuration of the scanner is stated.

Calculation of overall results: Analysis of detection results for the 'Boot Sector', 'Standard' and 'In the Wild' test-sets is as follows: a percentage detection rate will be calculated by considering the total number of infected objects identified. However, calculation of the overall score for the 'Polymorphic' virus test-set has been weighted to favour those vendors who identify all samples of each virus type included in the test-set. Thus, of the overall percentage detection score, 75% is made up by looking at the percentage of the total number of infected objects identified. The remaining 25% of the total score is made up in the same way, except that virus samples are only deemed detected if the product has detected every sample of that virus type in the test-set. Thus, a product which detected 98/100 of virus type A, 100/100 of virus type B and 75/100 of virus type C would score an overall percentage of 75*((98+100+75)/300)+25*(100/300) = 76.58%.

Notes:

[1] There is little doubt that testing the virus scanner against memory-resident stealth viruses or fast infectors provides further information about the functionalities of the product. However, the practical difficulties associated with this test make it impossible to carry out on a large number of products.

[2] Disinfection will be treated in a separate review. Due to the many different ways in which products offer disinfection, this is likely to be a difficult aspect to examine.

[3] This also doubles as a false-positive test.

TUTORIAL

Virus Infection Techniques: Part 4

This article is the final instalment in a series examining the various infection techniques currently employed by virus writers. The series (published in *VB*, November 1994, December 1994 and January 1995) is intended as a source of reference for anyone involved in virus prevention.

This month, five more infection strategies are considered: Source code viruses, Object code viruses, SYS file viruses, Kernel viruses and Multi-partite viruses.

Source Code Viruses

Possibly the least exploited (and most obvious) infection target under *MS-DOS* is the source code file. Such a file contains the program (in text form) which is compiled and linked to form the binary file which is actually executed on the PC.

The principle behind a source code virus involves adding virus code to source code files. When that infected source code file is compiled and linked, the newly created executable will contain a binary image of the virus, which when run could infect other source code files in turn.

Obviously, the addition of a large block of code to a program would be easy to spot, and virus authors have developed ways to add a minimum amount of code to an infected file. However, rather than look at source code viruses in general, let us consider a specific example of such a virus, as published by *American Eagle Inc.* When an infected program is executed, the virus will search for the string 'INCLUDE=' in the system environment. This variable normally points to the subdirectory which contains commonly used header files for the *Microsoft C* compiler. If this environment variable is defined, the virus will create a new header file, which it calls VIRUS.H. It will then write itself to this file, once as a binary data array, and once as its C source code.

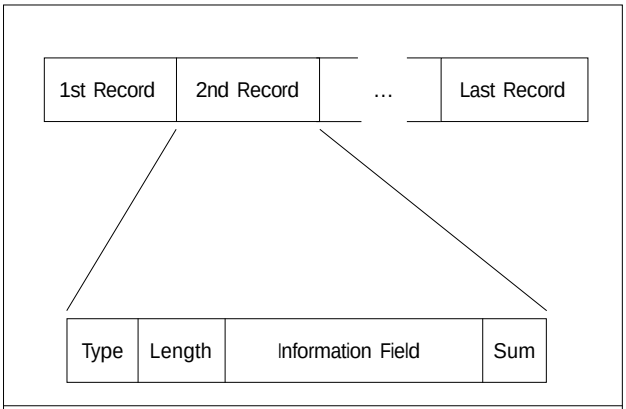


Figure 1: Structure of an Object Module. Each Object file consists of a number of variable length object records, shown in the top part of the figure. The internal structure of each of these records is shown in the lower half.

After this file has been successfully created, the virus searches for files with the file extension 'C'. If any is located, and deemed suitable for infection, it is opened and the strings '#include <virus.h>' and 'sc_virus();' are added to it. Thus before infection a source file like this:

```
main()    {
          printf("Hello  world!");
}
```

becomes

```
#include  "virus.h"
main()    {
          printf("Hello  world!");
          sc_virus();
}
```

When this file is compiled, linked and executed, processing passes to the virus code, which searches for other C files and infects them, allowing the entire process to be repeated.

This example quoted above is not the only sample of a source code virus. There are several other C and Pascal source code infectors. Equally, it is possible to infect assembler files.

In terms of detection and removal, source code viruses pose a number of problems. The most serious of these is that it is not possible to fix the binary image or the length of the virus code in an infected file, as the exact form will depend on the compiler and the command line switches used. One advantage of this infection technique for researchers, however, is that every one of these viruses carries its own source code with it - therefore they do not require disassembly!

Object Code Viruses

Although the threat posed to most users by object code viruses is extremely low, this technique has been employed by virus writers (the Shifting_Objectives virus infects object files - see *VB*, March 1994, p.11). The technique is therefore is included here, to complete the overall picture.

An object code virus will infect an object module. Although these modules are not themselves executable, the virus becomes active when an infected object module is linked

and run. In order to understand object file infectors more completely, it is necessary to consider the properties of object modules.

Most end-users would have no need to keep object files on their PC, as DOS programs are usually distributed in executable form (EXE, COM, SYS etc.). However, an object file represents an intermediate point between source code (the language in which the programmer writes) and executable code (the binary instructions executed by the processor). On almost all development machines, object files will exist, providing a possible (but limited) target for infection.

When a program is written, it is first compiled into a number of object modules. Next, the modules are linked together to form the completed executable - this allows any external routines to be linked in, and helps to make programs more modular in their structure.

Each object module contains five basic types of information: a binary image of executable code and data, a list of external functions called by the file, a list of address references, debugging information, and other miscellaneous pieces of information (for example, comments or link information).

Within each object module are a number of variable length object records. Different types of object record contain different kinds of information. Each object record begins with a one-byte file header which identifies the object record type. This is followed by a word containing the length of the record in bytes.

Next comes a variable length information field, which contains the main body of the record. Finally, the object record is terminated by a single-byte checksum. A schematic diagram of an object module is shown in Figure 1.

Now that the structure of an object module is known, it is easy to see that it is a potential infection target. As individual object records contain a binary image which is essentially executable code, it is possible for a virus to add an extra object record to an object module and make this the entry point for the linked object module. Thus, when an infected file is executed, the virus code gains control and another OBJ file can be infected.

Object file infectors pose a number of problems in terms of detection and removal. As these files are an intermediate stage in program construction, it is not practical to detect the presence of an object file infector with an integrity checker. Infection (or to be more precise, Trojanising) of the object files is difficult to prevent with a behaviour blocker, as the user will require read/write access to such files when they are created.

Detection of the infected executable files with a scanner is no more difficult than for an ordinary file-infecting virus, although the virus code's offset may vary depending on link options. However, detection of the virus code within the OBJ file is less trivial, as every object record would need to be searched in order to locate the virus code.

Fortunately, object file viruses pose only a small threat, as they are not able to replicate on most users' machines. Nevertheless, those who are involved in writing and developing software should be aware of their existence: the presence of such a virus on a development machine could be little short of catastrophic!

SYS File Infectors

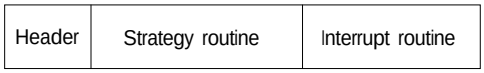
Although relatively rare, there are several viruses which attempt to spread by infecting SYS files. A SYS file is generally a device driver which is loaded at boot time through CONFIG.SYS, and used as an extension to the operating system kernel, which, for example, manages a controller, an adapter or a peripheral device.

Virus authors are well aware of the possibilities of SYS file infection. The following extract from *40-Hex* (Volume 2 Number 9 Issue 5) puts things into perspective:

```
The advantages of using SYS files are
manifold. There is no load high routine
involved apart from the strategy/interrupt
routines. This saves space. SYS files also
generally load before TSR virus checkers.
TSR checkers also can't detect the
residency routine of the virus, since it
is a normal part of the DOS loading
process. The routine for the infection of
the SYS file is ridiculously easy to
implement and takes remarkably little
space, so there is no reason not to
include SYS support in viruses. Finally,
the memory "loss" reported by CHKDSK
usually associated with memory resident
viruses is not a problem with SYS files.

A SYS file infector, when combined with a
COM and EXE general infector, can lead to
a powerful virus. Once the first SYS file
is infected, the infected system becomes
extremely vulnerable to the virus: there
is little the user can do to prevent the
virus from running, short of a clean boot.
```

Although *MS-DOS versions 3.0* and later are capable of loading device drivers in EXE format, prior versions of DOS require a precisely-defined structure for SYS files, which must be a memory image of the installed driver. Each SYS file has the following structure:



The device header always lies at the beginning of a device driver, and contains a link to the next driver in the chain, a word of attribute flags, and offsets to the executable Strategy and Interrupt routines for the device.

The Strategy routine (Strat) is called by DOS whenever any operation is attempted, passing it a pointer to a data structure called a request header. Following this call, control is passed immediately to the Interrupt routine (Intr) which carries out each

of the various functions which are supported by the driver. Lying between the kernel and the physical device, a device driver may be regarded as the 'perfect' host program for a virus to infect.

Due to the well-defined nature of device drivers, infection is a relatively trivial task. SYS files have several advantages over traditional file infectors, as they load early in the boot process, and while they do not present any particular problems in terms of detection, they do require that both checksummers and scanners carry out checks on SYS files.

Unfortunately, this is more complicated than it sounds, as *MS-DOS* allows the loading of any file with the right structure as a device driver; the file extension is ignored. Thus, for full protection, one would need to parse the contents of CONFIG.SYS and protect all the drivers which are loaded, regardless of extension.

Kernel Infectors

As has become normal in the anti-virus industry, the number of infection techniques continues to grow. One of the latest techniques to be seen, predicted by virus researchers for some time, is that of attacking the system file IO.SYS. However, rather than infecting the file by altering the contents of its executable code, the 3APA3A virus (pronounced Zaraza) changes the location of the file on the disk so that its own code is loaded, and not that of the host file.

On an uninfected DOS drive, the first entry in the root directory is the file IO.SYS. 3APA3A infects the drive by making a copy of the contents of the file, and changing the third root directory entry to point to it. It then copies its own code over the original IO.SYS file. The contents of the root directory before and after infection on a typical DOS drive are as follows:

(1) Root directory before infection

```
                size   cluster  attributes
IO        SYS   40470  2        Arc R/O Sys Hid
MSDOS     SYS   38138  22       Arc R/O Sys Hid
COMMAND   COM   52928  41       Arc
DOS             0      67       DIR
AUTOEXEC  BAT   100    68       Arc
CONFIG    SYS   150    69       Arc
```

(2) Root directory after infection

```
                size   cluster  attributes
IO        SYS   40470  2        Arc R/O Sys Hid
Vol
MSDOS     SYS   38138  22       Arc R/O Sys Hid
IO        SYS   40470  16108    Arc
COMMAND   COM   52928  41       Arc
DOS             0      67       DIR
AUTOEXEC  BAT   100    68       Arc
CONFIG    SYS   150    69       Arc
```

When the hard drive is next booted, the virus code located at cluster 2 is loaded, rather than the uninfected IO.SYS file. Once the virus has become memory-resident, the original copy of IO.SYS (pointed to by the third root directory entry) is loaded and executed, and the boot process continues as normal.

Once DOS is loaded, attempts to load the file IO.SYS will result in the original contents of IO.SYS being loaded, as the Volume attribute set on the first directory entry hides this file from the usual 'Find first' call.

The most interesting feature of Kernel infectors is that they allow the operating system to carry out the stealthing process for them: even after a clean boot, the infected copy of IO.SYS will not be seen without a specific search with the correct attributes set.

This means that checksummers need to be able to open files which are marked as the volume label, in order to detect the changes made by the virus. Similarly, virus-specific software must check the contents of the root directory with particular care.

At the time of the discovery of 3APA3A, many checksummers were unable to protect against this form of attack - it will be interesting to see which vendors will be able to deal with the problem in several months' time.

Multi-partite Viruses

A multi-partite virus can be any infector which infects a variety of objects, often (but not always) the boot sector and EXE of COM files. It contains the characteristics both of boot sector and of parasitic viruses. This is done to augment the probability of the virus spreading from one machine to another by increasing the number of different objects which can be infected. In terms of detection and removal, multi-partite viruses pose similar problems as each individual infection technique.

Conclusions

This article completes the series on virus infection techniques. While it is not a review of every single aspect of the different methods in which viruses can infect various objects under DOS, it does cover the majority of the important infection techniques.

Certain methods of infection (such as those viruses which prepend their own code, and append part of the original host file) have been omitted. In these cases, the infection algorithm was deemed sufficiently close to those described elsewhere. Another omission is the infection mechanism used by the Dichotomy virus: this is covered in the December 1994 edition of *VB*.

Windows, *OS/2* and *Windows NT* will doubtless bring with them their own particular brand of trouble, and the virus writers will not be idle. It is therefore a reasonably certain prediction that a fifth part will soon need to be added to this series. The only question which remains is when.

FEATURE

Hong Kong: A Virus Picture

Allan George Dyer
Yui Kee Co. Ltd.

The Territory of Hong Kong is a unique mixture of cultures and fast-moving environment: this is as true for viruses as anything else. Many viruses known worldwide are prevalent there, as are also the 'home-grown' variety. The most common viruses overall in Hong Kong are: Amoeba using CLME (no *CARO* name as yet); Anti-CMOS; Jerusalem.J; Jerusalem.Vtech 2358, 2513, 2886, and 2880; Mange_tout.1091; Mange_tout.1099; Ming.491; and Shutdown.644.

Many of these appear to have been written locally: Amoeba, Jerusalem.J, Jerusalem.Vtech, Ming, and Shutdown seem to have been produced by an active virus-writing group which is diverging from the relatively simple techniques it has used until now (e.g. overwriting file viruses) to writing polymorphic infectors. Typically, the group distributes new viruses by infecting software and uploading it to BBSs - *McAfee Scan* and *Microsoft* mouse drivers have both been targets.

The viruses state the authors' 'handles' and are in addition frequently intentionally damaging. The motivation of the group is unclear, but their method of distribution is chosen by the authors to cause maximum havoc.

International Connections

Some of the other viruses listed, such as Mange_tout and Anti-CMOS, might have originated in China: certainly, these were detected early in companies with connections in China, and are common there. A recent visit to China showed clearly that there are still viruses there which have never been seen by Western researchers.

Hong Kong's position as the 'gateway to China' and as a major trading centre gives it significance in the spread of viruses internationally. Large numbers of pre-formatted diskettes are exported from China, and motherboards, add-in cards and machines are assembled or packaged here. The following two incidents demonstrate the risks:

- The Nice virus, written in Hong Kong (an early offering by the writer of Ming, and of Amoeba using CLME), was first found circulating on local BBSs in January 1994. Two weeks later, a minor variant was found in Lapland and traced to a set of video driver diskettes which had been duplicated in Hong Kong from an infected master.
- Mange_tout.1099 was first found in January 1994 in a Hong Kong company with a factory in China. At the end of August 1994, it was detected in Norway on VGA display driver diskettes from Hong Kong.

These two viruses are not particularly virulent (Nice is an overwriting virus), but they have achieved international distribution by infecting an exporter.

Jerusalem.J is an interesting virus for a completely different reason: it is a rare example of a virus intended to display text in Chinese. It contains bitmaps for two Chinese characters, 'Death God', though this fails to display on some hardware. Chinese language shells are frequently loaded on top of DOS, and could be exploited by a virus to display messages in Chinese. No known virus does this - perhaps writers recognise the compatibility problems.

The Law

In early 1992, the *Hong Kong Computer Crimes Bill*, one of the first of its kind in the Far East Region, became law (see '*Information Systems Security: Legal Aspects*', by Dr Matthew K. O. Lee, *Hong Kong Computer Journal*, vol.9, no.11 pp.19-22). The bill amends existing criminal law, introducing four new types of offence. Pertaining to computer viruses, two key changes have been made:

- The term 'property' now includes any computer program or data held in any form and by any medium.
- Property damage now includes causing a computer not to function normally; altering or erasing any program or data held in any form and by any medium; and adding any program or data to the contents of a computer or any computer storage medium.

Significantly, recklessness may also suffice as intention. Thus it would appear that writing a virus with no intent to distribute it is not an offence, but that distribution is. Possession of an infected diskette with the intent of using it on a system might also be an offence, even where the accused was unaware of the virus at the time. The maximum penalty is ten years' imprisonment.

There are two specific defences: the accused is not guilty if, at the time concerned, he believed he was permitted to carry out the alleged activities, or if he believed he would have been, had the person entitled to give consent known the circumstances. Despite the existence of this legal framework, there have been no prosecutions; nor have detailed studies of virus prevalence and damage been undertaken.

The Future

Any article about Hong Kong is incomplete without a mention of the return of the Territory to China in 1997, but this factor has little apparent relevance to viruses. More important is the region's rapid economic growth, and associated explosion in all types of communication. Great efforts in user education will be required to prevent a commensurate increase in virus problems yet to arise.

PRODUCT REVIEW 1

IBM Anti-Virus for NetWare

Jonathan Burchell

IBM Anti-Virus for NetWare is the *Novell* server component in *IBM*'s anti-virus product range, designed to offer virus protection via real-time and scheduled scanning. It is strictly a server-based product, including a workstation-based alert TSR, but with no workstation protection or integration software, nor workstation-based administration, configuration or monitoring tools. It also lacks a *Mac* virus detection capability. However, the installation and user guide seems clear, well-written and concise.

Installation

IBM does not supply any installation routines; the software must be installed by copying it manually from the installation disk to a chosen server directory. The manual mentions that it may be necessary to unzip the files first to a local directory: this was the case with the version supplied.

To start the process, the data was copied from two directories on the local disk into one specified directory on the server. Users who have an earlier version of the product also have the option of an 'update only' installation. Considering that a complete version of the software fits into around 500k of disk file, it's a shame that *IBM* chose not to supply an installation routine.

IBM Anti-Virus for NetWare is supplied for use with *NetWare 3.1x* and *4.0x*. Under *NetWare 3.11* and *3.12*, CLIB versions 3.12f or later are required for real-time scanning to be supported.

The manual details how users of older versions can carry out the 'upgrade only' installation. One useful feature is that the procedure to start the software running on the server is identical, whichever version of *NetWare* is used. A small NLM is loaded which interrogates the server to discover what version and level of CLIB is running, before loading the appropriate anti-virus NLM.

A word of warning: the READ.ME file mentions that on many v3.0 installations, the NLM may stop working, as it runs out of short-term memory (*IBM* claims that the default *Novell* max.alloc parameter is simply too small).

Like most things in a READ.ME, I ignored it on the principle that it could not apply to me. Sure enough, halfway through scanning, the NLM paused, complaining that it had run out of memory. Increasing the *NetWare* limit via the system console allowed the program to continue working from where it had left off, without any apparent ill effects. The non-existent installation routines could tackle this: an ideal configuration issue.

Configuration and Administration

The NLM produces a single console screen which acts as a combined configuration screen and activity monitor - configuration must be done at the console, or at a workstation via RCONSOLE. The user interface is *IBM*'s own, developed 'in-house'; it is not the familiar and well-known *NetWare* character-based menuing system.

Control of options is achieved by typing commands at a prompt line which can be invoked via the F2 key. The options from which to choose are identical to those which would have been placed on the command line when the software was loaded, or in the preferences file. So, for instance, to specify that a user ADMIN is to be notified in the case of a virus being discovered, any of the following routes could be taken:

- add -NOTIFY ADMIN to the command line
- include -NOTIFY ADMIN in the preferences file (which can be altered with any text editor)
- load the NLM, type F2, enter -NOTIFY ADMIN

Scheduled Scanning

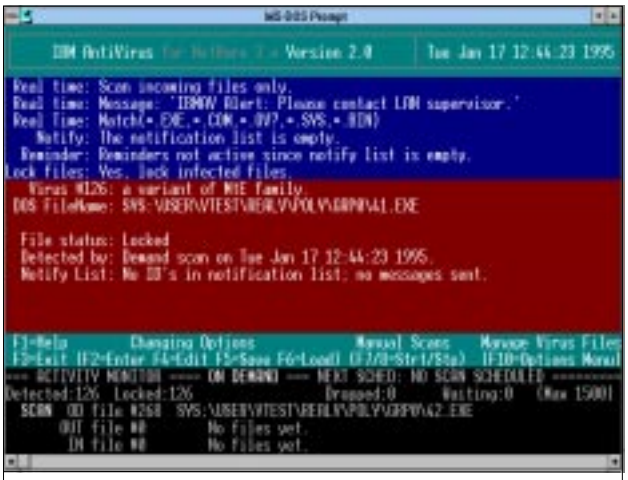
IBM Anti-Virus for NetWare supports two types of scheduled scanning: periodic, and those which start automatically at some point after loading the software. Up to five periodic scans can be defined, scheduled to start either daily, weekly, on Saturdays, Sundays, weekdays only, or at weekends. The calendar date, as well as the time at which the scan is to commence, may also be specified. The user may choose the maximum time to be spent scanning, and a list of volumes and directories to be scanned, together with a file-matching criteria. Entering the command

```
-SCHED1  Saturdays  1995/01/01  1:00:00:  FOR
2:00:00  SYS:  DEPT\953\  *.EXE  *.COM
```

will start a scan of all EXE and COM files on volume SYS, and on volume DEPT, directory 953 and its subdirectories; to begin at 01:00 each Saturday after 1 January 1995.

'FOR 2:00' tells the program to terminate the scan after two hours, even if it has not completed. The manual states that the next time this scan is invoked, an attempt will be made to start where it left off and thus complete the scan. Whilst this appears to be an attractive feature, I suspect that in a realistic environment, with large numbers of files appearing and disappearing between scans, the joins between the sessions can never be quite complete.

The second type of scheduled scan occurs at a given delay after the NLM has started, and is actually a variant of the periodic scan, and can also have up to five presets. The delay can be specified in HH:MM:SS format.



IBM Anti-virus features 'Full-time access', which prevents users receiving irritating 'sharing violation' or 'file in use' errors while the NLM is scanning files.

The scan specification allows the user to define the type of file to be checked, as well as the volumes and directories to be scanned. In addition, there is a global 'ignore list' which is common to scheduled, on-demand and real-time scanning, allowing specific files, directories and/or volumes to be ignored by the scanner.

The -DECOMPRESS option specifies whether the scanning engine should try to decompress PKZIP- and LZEXE-compressed files before scanning them. Like the 'ignore list', it is a global option and affects all types of scanning.

The scanning engine uses an *IBM*-developed technology called Full-time access, which allows users always to gain access to their files, even if they are being scanned. The product monitors all requests for file access: if a request is made to access a file being scanned, the NLM checks whether the user needs exclusive access. If so, the file is released, and the user granted exclusive access. If not, both scanner and user share access. Should a file be prematurely released to a user, the NLM notes this and continues scanning it once it has been released by the user program.

The advantage is that users should not experience 'file in use' or sharing violation messages, which are very annoying and can break user programs and environments. As far as I could see, this technique works well: I could not generate any access clash whatsoever when the scanner was working.

Real-time Scanning

Real-time scanning is controlled via two options: -CHECK allows specification of what combination of incoming and/or outgoing file access should be checked, and -MATCH allows specification of the file match criteria. File matching includes not just the file pattern (with wild cards) but also paths, directories and volumes.

The program does not check files immediately on access; rather, the access attempt is noted in a 'to do' list, and the file is scanned by a separate task as soon as time allows. The incorporation of the 'Full-time' access technology (see Scheduled Scanning) ensures that file access conflicts between users and the anti-virus software cannot arise.

The advantage of delayed scanning is that the delay caused to file access by enabling real-time checking is minimised. Additionally, it allows the use of a single scanning engine for scheduled and real-time scanning. The major disadvantage is that during periods of heavy server loading, the list of files waiting to be checked can grow considerably (to several hundred entries), yet full user access to these files is still permitted. An infected file could propagate to several machines before being checked and an alert generated.

The product addresses this problem in two ways. Firstly, a log is kept of all file access between the arrival of a file to be checked and its being scanned: if an infected file is discovered, the log provides a list of all users who have accessed it. Secondly, the size of the pending list may be decreased by increasing the priority of real-time scanning. Priorities may be specified for normal operation, with an extra, higher priority to be invoked if the list gets dangerously full.

The 'delayed list' approach has many advantages, but the thought of what would happen if a heavily-accessed file, e.g. LOGIN.EXE, became infected just before a period of extreme usage worries me. If a read request is received for a file on the checklist because the file has just been written to, the NLM should abandon this delayed processing and scan the file immediately. Such an approach would combine the advantages of both types of real-time scanning.

Processing Infected Files

On detecting an infected file, only three actions are permitted. Firstly, notification can be sent to the entries in a NOTIFY list. This can be sent using *Novell* broadcast messages, or a format understood by the Alert TSR/Utility running on designated workstations. To receive a *Novell* broadcast, a user must be logged in at the time it is sent. *IBM* has added a REMIND feature which continues to send messages at a specified interval until the problem is fixed. However, if a user was logged out when the original infection was detected, he should receive a message when he next logs in, if there is still a problem.

When a virus is detected by the real-time scanner, a configurable message can be sent to the user accessing the file. The only other (optional) action taken by the NLM on discovering a virus is to 'Lock' the file, which prevents further user access until remedial action has been taken.

Further processing of infected files is unique to this product, and perhaps a little cumbersome. Details of each infected file are placed into a log list, which is brought up by pressing F10 at the main console scene. Each entry is accessed separately. The user may choose one of several actions to take: locking or unlocking further access, erasing the file, copying it to a quarantine directory, or attempting to repair it by disinfection. Each infection must be dealt with individually. There are no global operations available, such as 'Copy all infected files to

the quarantine directory and erase them from the source directory'.

If dealing with a 'real' outbreak rather having run the product against some test-sets, however, it is probably wise to deal with infections singly. The list of infected files cannot exceed 1500 entries - if this happens, only the most recent infections will be stored. This is an improvement over version 1.06, which had a much smaller limit, but it could still mean a loss of information in a real-life situation.

An evaluation option eases the task for evaluators (who, like reviewers, put large test-sets through scanners) by forcing the NLM to produce separate log files of clean and infected files. The NLM keeps a log file which contains only information on infected files, and is the source of the data used during the 'processing' of infected files. No formal documentation of this file exists, nor facilities to filter or generate reports based on the information in the file.

The only workstation utility provided is an alert program to display 'virus alerts' on a workstation. DOS, *Windows* and *OS/2* versions are provided, and, unlike the standard *Novell* utilities, will function even if the workstation is not currently logged in to a server. The *Windows* and *OS/2* versions are more sophisticated than the DOS version: they show the last ten messages received and can be programmed to carry out another task on reception of an alert.

Conclusions

As can be seen from the results at the end of the article, real-time and scheduled scanning have identical detection ability, indicating that the same engine is being used in both cases. The score on the polymorphic samples is excellent, with the exception of the Uruguay and the DIET-compressed Cruncher infections. However, it is disappointing not to see a 100% detection rate on the 'In the Wild' test-set.

IBM Anti-Virus for NetWare seems destined to occupy the middle ground between the 'really basic' products and the more sophisticated ones. Much is missing: server-to-server facilities, group administration and workstation tools, and integration of server and workstation protection. It also has a user interface which you will either love or hate.

On the plus side, the product has a good solid 'big blue feel' to it, and polymorphic detection is sound. There is also absolutely no reason why detection of 'Standard' and 'In the Wild' viruses should not increase to 100%.

The lack of reporting and general flashiness means there is little hope of directly impressing your boss with it, but people rarely get fired for choosing *IBM*. The configuration options, which can all be specified in a control file, are flexible enough to produce a 'fit-and-forget' type of operation. If you only need to protect a single-server network this could well be a product to consider - if the 'In the Wild' detection figure of future releases improves to 100%.

IBM Anti-Virus for NetWare

Detection Results:

NLM:

Standard Test-Set[1]	227/229	99.1%
In the Wild Test-Set[2]	104/109	95.4%
Polymorphic Test-Set[3]	553/600	92.2%

DOS Scanner:

Although a workstation-based Alert TSR is provided, no DOS scanner is included.

NB: Results for real-time and scheduled scanning are identical - see text for details. Full comparative speed results and overheads for all current NLMs will be printed in a forthcoming *VB* review.

Technical Details

Product: *IBM Anti-Virus for NetWare, version 2.00.*

Developer: *IBM Corporation*, M D 223 Long Meadow Road, Sterling Forest, New York 10979 USA. Tel. +1 914 759 4570, Fax +1 914 759 4690.

UK Vendor: *IBM PC Company, Normandy House, Alencon Link, Basingstoke, Hants RG21 1EJ, UK.* Tel. +44 1256 344558, Fax +44 1256 332319.

Price: These include access to all *IBM* anti-virus programs via their Bulletin Board System, quarterly updates (also via BBS), support desk, virus alert service, and reverse engineering of new viruses. For 1-250 users, the subscription is £1000.00; 251-500 users, £2000.00; 501-1000 users, £4000.00; 1001-2000 users, £6500.00; 2001-3000 users, £9500.00; 3001-5000 users, £12,500.00; 5000+ users, price on application.

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*, January 1995, p.19 (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:** 600 genuine samples of: Coffeeshop (250), Groove (250), Cruncher (25), Uruguay.4 (75).

PRODUCT REVIEW 2

AVAST! Virus Ahoy?

Dr Keith Jackson

Although *AVAST!* has already made two appearances in *Virus Bulletin*, this is the first time the product has been given a stand-alone review. Given its excellent results in the last Comparative Review, I was curious to see if the rest of the product would live up to my expectations.

Component Parts

The product is made up of a selection of virus scanners (see below) and five separate programs, which are described in the accompanying manual as 'General anti-virus programs', and which perform different but closely-related tasks. *AVAST!*'s developers seem to apply this description to programs which use checksum verification techniques, behaviour monitoring, and/or detection of any unwanted alteration - in fact, almost everything but scanning.

One of the five programs checks a specified file, the system area of a disk, or memory. A second verifies an entire disk (creating its own checksum database when first executed). Two more general anti-virus programs are both memory-resident file behaviour blockers and modification detectors. One works under DOS; the other requires *Windows*. The fifth program of the set is used only once; to take a copy of the partition table and the boot sectors of a hard disk.

AVAST! also has three scanner programs: like the behaviour blockers, one is a DOS program, and one runs under *Windows* - the third is memory-resident. Curiously, *AVAST!* uses the adjective 'locating' to refer to these three programs. Finally, a 'Specific Anti-Virus Program' is provided which will only detect (and/or eradicate) a few named viruses.

I was surprised by the plethora of individual programs; however, an 'Integrated Environment' is provided which ties their actions together. This is easy to use (drop-down menus, etc), but relies on users knowing the meanings of the names of individual programs. The fact that they are all called 'something GUARD' could easily lead to confusion.

Documentation

Two manuals were provided: a 113-page, A5 book describing version 6, and a six-page addendum booklet, updating the documentation to version 7. Neither is indexed, possible error messages are not documented, and some of the description is to say the least perfunctory. The documentation details how each component part of *AVAST!* operates, but due to the similarity of the programs quickly becomes rather repetitive. However, the chapter entitled 'The Problem of Computer Viruses' is well written, and detailed descriptions of 21 common viruses can be found in another.

Notwithstanding the problems and omissions outlined above, I have seen far worse documentation accompanying anti-virus products. The manuals provided in this package should prove adequate unless problems are encountered, when more information may be needed.

Installation

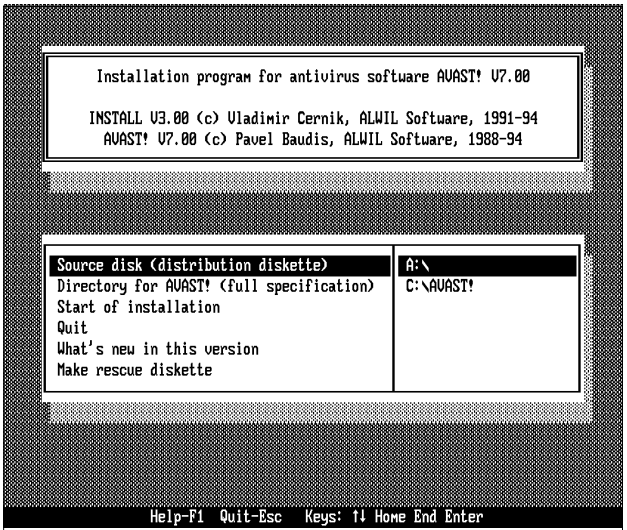
Installation is straightforward: a subdirectory in which to place *AVAST!* files is selected, a scan for extant viruses is made, and the *AVAST!* files are copied (1.08 Mbytes in 37 files). *Windows* operation requires execution of a special-purpose installation program which alters the *Windows* files WIN.INI and SYSTEM.INI, and creates its own Program Manager group.

The file AUTOEXEC.BAT is altered to add *AVAST!* to the PATH, and the altered file is stored as AVAST!.NEW, for renaming to AUTOEXEC.BAT as and when required. Perhaps such alteration should be automatic?

A Rescue Diskette may be made during installation, but this option must be selected from the installation menu, and is not offered to the user explicitly. During installation, *AVAST!* does not write back to its own diskette; in fact, it refuses to continue installation if the floppy is not write-protected. *AVAST*'s developers suggest that installation is made from a backup copy, not the original: sensible advice.

Scanning

Using the default settings, *AVAST!* took 1 minute 55 seconds to scan the hard disk of my test PC. All COM, EXE, SYS and OV* files were scanned for 3103 viruses. When the initial memory



Installation is simple and uncluttered. During this process, the program aborts if it discovers that the master diskette is write-enabled - a nice touch.

scan was disabled, scan time reduced to 1 minute 43 seconds: when all parts of all files were scanned, it rose to 5 minutes 10 seconds. For comparison purposes, when scanning the same hard disk, *Dr Solomon's Anti-Virus Toolkit* took 1 minute 23 seconds; *Sophos' Sweep* took 2 minutes 28 seconds in 'Quick' mode and exactly 7 minutes in 'Full' mode.

The documentation states that background execution of the *Windows* scanner is disabled by default. This is not strictly correct, as when first executed, the *Windows* scanner starts scanning, though permitting other operations. My initial reaction was to regard this flurry of activity as program initialisation, and it was only after watching it operate for several minutes that I realised scanning had actually begun.

Under *Windows*, *AVAST!* took 6 minutes 50 seconds to scan the hard disk of my test PC, of which 1 minute 11 seconds were spent scanning through memory - I've no idea why. Scanning under *Windows* is designed as a repetitive task, which can, at the user's choice, be carried out in the background. After one scan is complete, another starts. I do not see this sort of repetitive scanning as an ideal option for most users. Surely the *Windows* TSR should prevent the execution of any infected files? I must be honest, and say that I am baffled by this option.

Accuracy

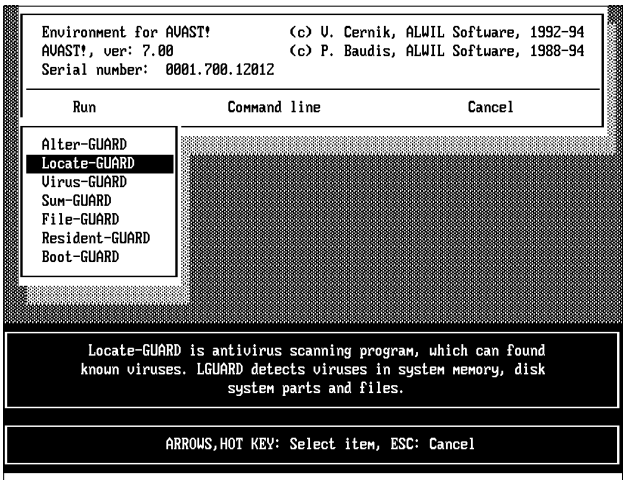
AVAST! detected all virus-infected test samples described in the Technical Details - a perfect detection rate. What more can one ask? All 500 Mutation Engine-infected test samples were also detected correctly. *AVAST!* has already proven its excellence in this area: it was the only scanner to detect all of a far more detailed polymorphic test-set in the Comparative Review published last month in *VB*.

The memory-resident scanner states clearly that it uses the same database as the scanner. However, the developers have taken a decision to scan for viruses only when programs are executed, rather than when they are read from disk. This means that infected files can be copied, but not executed. It also means that the large overhead incurred when every file read from disk is scanned is neatly sidestepped.

Whether the lack of on-access scanning (as opposed to on-execution) is a feature or an omission is a 'swings and roundabouts' choice. However, *Alwil's* solution is probably correct: any memory-resident program which incurs a noticeable performance overhead soon gets ditched.

General Anti-Virus Programs

Checksum verification of the integrity of the same hard disk used for the scanner tests described above takes 1 minute 23 seconds under *MS-DOS*. When the original checksums are derived, *AVAST!* places its database file in the hard disk's root directory. This is untidy: *AVAST!* should either maintain its database in its own subdirectory, or where the user asks it to be placed.



AVAST! provides a menu-driven front-end for each program. Unfortunately, this is somewhat confusing due to the way in which the programs are named.

Unfortunately, verifying checksums under *Windows* defeated me: after many hours of trying I still could not make it operate. All that happened was a cryptic error message saying 'Failed to open document'. Which document? This is perhaps a little unfair, as the *Windows* help files have only recently been added to the product, and were not available to us when the review was carried out.

The General Anti-Virus programs which monitored user behaviour seemed to operate correctly. Such beasts are difficult to test exhaustively, though apart from the 15 second load time (why so long?), I have nothing untoward to report. I also have no quibbles about the program which can be used to verify a particular file, or a specific area of disk or memory.

Finally, the menu-driven program which takes a copy of a hard disk's partition table and boot sectors was particularly easy to use: the user simply makes a selection from one of four proffered menu options, and the software does the rest.

Specific Anti-Virus Program

AVAST! contains one executable program which stands out from the rest, as it claims to be able to detect only a very small list of viruses (21 in total, four of which appear to be boot sector viruses). It executes more quickly than the main scanner, taking only 1 minute 7 seconds to scan the hard disk of my test PC.

This 'Specific Anti-Virus program' exhibited problems when inspecting disks in my Magneto-Optical drive. When executed, it first tested memory, then displayed a message stating 'Testing system area of the disk' for 43 seconds. Only then did it start to look at disk content. The delays were so long that I thought *AVAST!* had hung the computer, and that the machine would have to be rebooted.

Curiously, when files were inspected, the software claimed to have immunized several virus-infected files, even though the write-protect tag of the Magneto-Optical disk was enabled at all times. I could not ascertain why. The main scanner program had

exhibited no problems at all when accessing files stored on Magneto-Optical disks.

When a virus-infected program has been detected, either the file is 'immunized' (their word - the manual seems to use this phrase to mean the removal of the virus' self-recognition features), or the user is offered a chance to delete it. My advice is that users should not play games with virus removal options. Infected executable files should be replaced by a copy of the original non-infected file, not by a 'cleaned up', perhaps corrupted, version. Being fair, the *AVAST!* documentation recommends against the use of 'immunization' wherever possible.

Without knowing the precise variants which the Specific Anti-Virus program is attempting to detect, it is impossible to test the accuracy of this scanner. However, when run against the current test-set, thirty files were deemed to be infected. Of these thirty files, twenty were immunized, and the rest were deleted.

Alwil explains the presence of VGUARD by stating that it is partly obsolete, and has been replaced by AGUARD, the integrity checker. This attempts to perform perfect disinfection by using information stored about the file before infection. VGUARD is therefore a last-resort, to be used when no initial checksums of infected files are available. However, I personally found the placement of the products confusing, and would find it helpful if the proposed use of each component could be emphasized in the manual.

Conclusions

AVAST! is undoubtedly one of the best anti-virus packages around at detecting viruses. It successfully detected all the viruses against which I tested it, and performed its scans at quite a reasonable speed. The product was just as successful in last month's Comparative Review: there are very few anti-virus packages available currently which can come close to such a high detection rate.

The developers should perhaps think carefully about the structure of *AVAST!* as a set of individual programs. These programs are so numerous that users will be unsure which one to use in a particular situation. I would also humbly suggest that the *Windows* components of *AVAST!* need further development and optimization. It would be a shame for these to be reasons for rejecting *AVAST!*, as its core technology gives truly excellent results.

This product is one of the best-performing anti-virus programs I have reviewed in a long while. Given revision of the user interface structure, and extra documentation, its developers could have a world-wide winner on their hands.

After a sequence of reviews of scanners which were so poor that mercy killing could be the only real solution, it is a pleasure to write about a scanner which works exceedingly well. Interestingly enough, the literature provided with *AVAST!* does not make grandiose claims about its virus detection capability, but its performance is excellent. Why is it that a scanner's performance seems inversely correlated to the extravagance of the claims made in its marketing literature? Do vendors believe we fall for such hyperbole?

Technical Details

Product: *AVAST!*

Developer/Vendor: *ALWIL Software*, Prubezna 76, 100 31 Praha 10, Czech Republic. Tel. +42 2 782 20 50, Fax +42 2 782 25 53, BBS +42 2 782 25 50

Availability: *AVAST!* requires an IBM PC/XT/AT with at least 256 Kbytes of RAM, using *MS-DOS* or *PC-DOS v3.10* or above. A hard disk is useful but not essential.

Version evaluated: 7.00

Serial number: 0001.700.12012

Price: US$79.00

Hardware used: A *Toshiba* 3100SX laptop PC (16MHz 386) with one 3.5-inch (1.4 Mbyte) floppy disk drive, 5 Megabytes of RAM, and a 40 Megabyte hard disk, running under *MS-DOS version 5.00*.

Viruses used for testing purposes: This suite of 157 unique viruses (according to the virus-naming convention employed by *VB*), spread across 246 individual virus samples, is the current standard test-set. A specific test is also made against 500 viruses generated by the Mutation Engine (which are particularly difficult to detect with certainty).

The test-set contains 8 boot sector viruses (Brain, Form, Italian, Michelangelo, Monkey, New_Zealand_2, Quox, Spanish_Telecom), and 239 samples of 150 parasitic viruses (Spanish_Telecom appears in both lists). There is more than one example of many of the viruses, ranging up to 12 different variants in the case of the Tiny virus. The parasitic viruses used for testing are listed below. Where more than one variant of a virus is available, the number of examples of each virus is shown in brackets. For a complete explanation of each virus, and the nomenclature used, please refer to the list of PC viruses published regularly in *VB*:

1049, 1260, 12_Tricks, 1575, 1600, 2100 (2), 2144 (2), 405, 417, 492, 4K (2), 5120, 516, 600, 696, 707, 777, 800, 8888, 8 TUNES, 905, 948, AIDS, AIDS_II, Alabama, Ambulance, Amoeba (2), Amstrad (2), Anthrax (2), AntiCAD (2), Anti-Pascal (5), Armagedon, Attention, Bebe, Blood, Burger (3), Butterfly, Captain_Trips (2), Cascade (2), Casper, Coffeeshop, Dark_Avenger, Darth_Vader (3), Datacrime, Datacrime_II (2), Datalock (2), December_24th, Destructor, Diamond (2), Dir, Diskjeb, DOShunter, Dot_Killer, Durban, Eddie, Eddie_2, Fellowship, Fish_1100, Fish_6 (2), Flash, Flip (2), Fu_Manchu (2), Halley, Hallochen, Helloween (2), Hide_Nowt, Hymn (2), Icelandic (3), Internal, Invisible_Man (2), Itavir, Jerusalem (2), Jocker, Jo-Jo, July_13th, Kamikaze, Kemerovo, Kennedy, Keypress (2), Lehigh, Liberty (5), LoveChild, Lozinsky, Macho (2), Maltese_Amoeba, MIX1 (2), MLTI, Monxla, Murphy (2), Necropolis, Nina, Nomenklatura (2), Nuke_Hard, Number_of_the_Beast (5), Oropax, Parity, PcVrsDs(2), Perfume, Pitch, Piter, Polish_217, Power_Pump, Pretoria, Prudents, Rat, Satan_Bug (2), Shake, Sibel_Sheep (2), Slow, Spanish_Telecom (2), Spanz, Starship (2), Subliminal, Sunday (2), Suomi, Suriv_1.01, Suriv_2.01, SVC (2), Sverdlov (2), Svir, Sylvia, Syslock, Taiwan (2), Tequila, Terror, Tiny (12), Todor, Traceback (2), Tremor, TUQ, Turbo_488, Typo, V2P6, Vacsina (8), Vcomm (2), VFSI, Victor, Vienna (8), Violator, Virdem, Virus-101 (2), Virus-90, Voronezh (2), VP, V-1, W13 (2), Willow, WinVirus_14, Whale, Yankee (7), Zero_Bug.

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 531889, International Fax +44 1235 531889
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

The next round of **Anti-Virus Workshops** organised by *Sophos Plc* will be held at the training suite in Abingdon on 29/30 March 1995. The first day consists of the Introduction to Computer Viruses; the second is the advanced Anti-Virus Workshop. Further details available from Karen Richardson at *Sophos*. Tel. +44 (0)1235 559933, Fax +44 (0)1235 559935.

Network Security Management Limited will be holding a one-day seminar on the **forensic examination of computers** at *Hambros Bank Conference Centre* (London, UK), on 7 April 1995. The course lecturer is Edward Wilding, a consultant in the company's Investigation Division and consulting editor to *VB*.

The *Computer Security Institute* (*CSI*) has announced the publication of its *Manager's Guide to Internet Security*, the fourth in the *Manager's Guide* series. Other volumes are ***Telecommunications Fraud, Computer Viruses,*** and ***Computer Security Awareness***. These guides, along with other related benefits, are available with *CSI* membership. Contact *CSI* directly on Tel. +1 415 905 2626, Fax +1 415 905 2218 for more information.

Norman Data Defense Systems has announced the release of a white paper entitled **'Virus Policies That Work'** (written by Dr David Stang); a research report which illustrates flaws in current virus prevention policies, showing how they can best be replaced. At the same time, *Norman* has made a major upgrade to its workstation protection software, *Norman Armour*. For information, contact the company on Tel. +1 703 573 8802, Fax +1 703 573 3919.

An introduction to the audit and security of *Novell NetWare* will be held at the *Marriot Hotel*, London, UK, 3-5 April 1995. For further information, contact Mandy Moore at *MIS/Euromoney Publications plc*. Tel. +44 (0)171 779 8944, Fax +44 (0)171 779 8975.

A new strategy from developers *McAfee* in the UK is hoped by the company to increase its market presence. **Fifteen thousand fully-operational CD-ROMS of its products** are being distributed throughout the UK, featuring eleven products, including *NetShield* (server virus protection) and *VirusScan* (workstation virus protection).

An 8K ROM chip which purports to be able to detect viruses as soon as they enter a system has been released by *MIS Europe*. The chip plugs into existing PROM sockets, and **uses a heuristic algorithm to detect boot sector viruses** as they try to gain access to a system. Details from Kate Holland, *MIS Europe Ltd*.
Tel. +44 (0)1622 817808, Fax +44 (0)1622 817857.

The first issue of *EICAR News*, from the *European Institute for Computer Anti-Virus Research*, appeared in January 1995. Volume 1 Number 1 contained, amongst other articles, industry news, virus analyses, and reports from three working groups (technology, practice, and legal aspects). The organisation is dedicated to the control and prevention of the spread of viruses. Membership details are available from *EICAR e.V.*, Etzwiesenstrasse 6, D-81735 München, Germany.

PC Week reports (17 January) that *Intel* and *Novell* have launched an integrated system called *ManageWise*. The package features five separate products covering **management, remote control, analysis, inventory, and virus protection**, and is expected to compete directly with *Microsoft's Systems Management Server*. *Novell* can be contacted on Tel. +44 (0)1344 724000; *Intel* on +44 (0)1293 696000.

Data Fellows Ltd and *Command Software* announced this month that they have formed an alliance to facilitate greater penetration of the world market, with special emphasis on the UK. Both companies develop and market the world-renowned *F-Prot* anti-virus software written by *Frisk Software International* of Iceland.