ISSN 0956-9979 FEBRUARY 1996

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **Courtly activities.** Jim Bates was the expert witness for the Crown in the case brought against self-confessed virus writer Christopher Pile. In November, Pile was jailed for eighteen months: Bates describes the process which led to this conviction. See p.17.

• **Stop press!** A virus infecting *Windows 95* and a new *Microsoft Word* virus found their way to the *VB* offices only days before this edition went to print. For a brief description of each, see p.15.

• **Recalculating the calculations.** In the magazine's January edition, *Virus Bulletin* published its biannual comparative review. Turn to p.19 for important information on the results.

CONTENTS

EDITORIAL

Is this a virus I see before me…?

“ viruses are, by their very nature, not put through beta testing”

Have you heard the one about the door? I’d be very surprised if it works in any language other than English, but it goes like this: ‘When is a door not a door?’ ‘When it’s ajar’.

Yes, that one was old when Noah built his Ark, but it leapt unbidden to my mind when rechecking the samples from the January comparative (see p. 17). It seems strange, doesn’t it? A virus is, after all, such a definite concept – surely either a file is infected or it is not?

Alas, this is not always the case. There is a grey area between the states ‘infected’ and ‘uninfected’: for example, where the target object has been modified from its clean state, but is not viral in itself. The term ‘damaged’ is often used to describe such an object.

The reasons are simple. Viruses are, by their very nature, not put through beta testing. They are not tested in all circumstances, and many do not work under all conditions; indeed, a surprising number of the samples received by anti-virus labs do not work in any circumstances whatever. A swift perusal of *Virus Bulletin* back issues is enough to remind one that there are many infectors which contain bugs.

The more complex viruses are particularly prone to these problems. Those which use internal DOS functionality are often dependent on a specific version of a certain type of DOS, or on a particular machine configuration.

So, for whatever reason, damaged samples do exist. Exactly how is an anti-virus product supposed to deal with such a thing? It is not a virus; it doesn’t replicate. So it should not be detected, right? Technically speaking, this is so; yet the object has been affected by the virus. The chances are it won’t work, the odds are high that it will crash the computer: your mileage will certainly vary. Therefore, it should be detected… It is an interesting little paradox.

The samples of Sarampo which were used in the January comparative illustrate this point: seventeen products detected the invalid samples, and only four more detected the valid new ones. Some of these products, it transpires, detect the samples deliberately – that is to say, they have built-in detection for the broken replicants as well as for the valid ones. This discovery was somewhat reassuring, as a modern entry-point scanner should not find them accidentally. The initial JMP passes control to a point 100h bytes too far into the viral code, which is a non-trivial mistake on the part of the virus.

Some products, of course, do not detect the damaged files – this could be for one of two reasons. Firstly, they are taking the more technical viewpoint: the damaged files are not viruses, and so should not be detected as such. Secondly, it is highly likely that they do not realize that there is a problem: unless they happen to have tried to infect goats which are a multiple of 100h bytes long (the conditions under which the virus will infect incorrectly), there is no reason for them to be aware of this particular difficulty.

Whilst not detecting the samples is, as described above, technically the right course of action, it is undesirable from the users’ point of view. There is no real problem simply in identifying the broken replications as ‘Damaged by Sarampo’ or something similar – that gives the best of both worlds. In addition, it would give the user far more information about what has happened to his files.

It is all a question of the level of information provided. The more information a product can convey about an infection, the better – but the price of that information may be high in terms of speed and space. If a product reports a file as (for example) ‘Damaged by Sarampo’, it must both work harder to make that determination, and store more information to tell the user why it has done it, not to mention the additional research required to discover that the problem can occur. The compromise between precision and usability can only continue.

NEWS

Selling Dr Solomon

S&S International, long famous for its anti-virus software and data recovery service, is undergoing a major shake-up. According to a press release from the company, a buy-out of the anti-virus software side of the business has been agreed between a senior management team consisting of Geoff Leary, David Stephens and Keith Perrett, and the current shareholders, Alan and Susan Solomon, who will retain the data recovery side of the business, to be known as *Authentec*.

The transaction is said by *PC Dealer's* sources to be worth some $28 million, and is expected to be completed sometime in February. The board of directors of the anti-virus company (to retain the flagship name of *S&S*) will consist of Leary as managing director, Perrett as head of products and purchasing, and Stephens as financial director. Alan Solomon will become a non-executive director.

Although Solomon will not be actively involved in the day-to-day running of *S&S* once the buy-out has been completed, he will continue to sit on the board and hold a level of responsibility for its activities.

Solomon also expects to take a fairly behind-the-scenes role in the data recovery business. 'I haven't done any data recovery myself for some years now, so I don't expect to be one of the engineers, but I will be active on the planning side,' he explained. 'I'm planning to do a lot more writing for magazines and newsletters, and I also have some other ideas which I will develop independently of *S&S*. Some of these are computer-related, some virus-related.'

S&S has stated that, under its new management, commitment to its products remains firmly in place. No change in priorities or operations are planned, nor is a reduction in staffing levels forecast – in fact, the company is currently recruiting around twenty new staff, and operations world-wide are expanding.

Geoff Leary, of the management buy-out team, commented: 'We are poised for continued growth in all our target markets. The Directors look forward to leading the *S&S* team to even greater success in the future, with continued emphasis on high-quality products and great customer support.' ▮

Another Month, Another Macro

With the passing of the *Underground Technology Review* into the annals of history [*see VB December 1995, p.4*], *Virus Bulletin* had expected to see Mark Ludwig's impact on the virus world take a temporary dive.

The discovery that the last issue of the journal shipped with the first known *AmiPro* macro virus has dented this belief slightly. The virus, which calls itself 'Green Stripe', is reported by *Reflex Magnetics* to contain a payload which is designed to change all occurrences of 'its' in a document to 'it's', but it is not currently known when, if ever, this payload is triggered.

AmiPro presents at least one major problem for viruses – the fact that the macros are stored in a separate file (with extension .SMM) alongside the main document file (standard extension .SAM). This subject will be covered, along with a full analysis of the virus and its techniques, in the next issue of *Virus Bulletin* ▮

Prevalence Table – December 1995

Virus	Incidents	(%) Reports
AntiEXE.A	30	11.3%
Concept	28	10.5%
AntiCMOS.A	21	7.9%
Form.A	19	7.1%
Empire.Monkey.B	17	6.4%
Parity_Boot	16	6.0%
NYB	12	4.5%
Stoned.Angelina	12	4.5%
BuptBoot	10	3.8%
Ripper	10	3.8%
Manzon	9	3.4%
Jumper.B	6	2.3%
Junkie	6	2.3%
Sampo	6	2.3%
EXEBug	5	1.9%
StealthBoot.C	5	1.9%
Die_Hard	4	1.5%
Telefonica	4	1.5%
Natas.4744	3	1.1%
Bye	2	0.8%
Empire.Monkey.A	2	0.8%
Espejo	2	0.8%
Form.D	2	0.8%
J&M	2	0.8%
One_Half.3544	2	0.8%
Taipan.438	2	0.8%
Other*	29	10.9%
Total	266	100%

* The Prevalence Table includes reports of one of each of the following viruses: Ash.270, Barrotes.1301, Boot.437, Byway.A, Da'Boys, Faca.1901, Finnish_Sprayer, Flip, Fly.1769, Int7f-e9, Jerusalem.?, Ken+Desmond, NewBoot_1, Nightfall.4559.B, Nuclear, Plato, Polonaise.2402, SMEG:Queeg, Starship, Stoned.LZR, Stoned.Standard, Stoned.Stonehenge, Tequila, Teraz.2717, TPE, Trakia.1070, Unashamed, Urkel, Vienna-Vio.2262

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 1996. 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	**M**	Infects Master Boot Sector (Track 0, Head 0, Sector 1)
D	Infects DOS Boot Sector (logical sector 0 on disk)	**N**	Not memory-resident
E	Infects EXE files	**P**	Companion virus
L	Link virus	**R**	Memory-resident after infection

Alex.599 — **CN:** A simple, appending, 599-byte direct infector containing the text: 'ALEX'.

```
Alex.599            BF00 018D B64A 03B9 0300 F3A4 C686 5A03 00C6 865B 0300 E80B
```

Baby.83 — **CERP:** A companion virus with an effective length of 83 bytes. The virus renames *.COM files to *.COV and *.EXE files to *.EXV, and copies itself to the original programs. The length of programs containing the virus code differs, but is always equal to N*256+2 bytes.

```
Baby.83             B456 2688 65FE 5FCD 21B4 3CB1 02CD 210E 1F93 B440 BA00 01CD
```

Bad_Brains.554 — **CNO:** An overwriting, encrypted, 554-byte variant of the virus known as Leprosy.Bad_Brains.570. It contains the texts: 'SKISM', '*.COM' and 'Bad Brains'. Although the virus has an extremely slow infection mechanism, reports have been received of it in the wild.

```
Bad_Brains.554      E9E8 0051 B962 03BE 3801 8BFE FCAD 3306 0301 AB49 E302 EBF5
```

Claws.684 — **EN:** An appending, encrypted, 684-byte direct infector infects two files at a time. It contains the texts: '*.MS *.CPS ANT*.DAT' and 'CLAWS (c) 1994-95 WerWolf'. The virus contains a destructive procedure, but a minor bug in the code prevents the payload being triggered.

```
Claws.684           1300 2EB8 35?? ??47 4781 FFA2 0272 F3C3 2EC6 06A7 0281 EBE7
```

Combi.1106 — **CN:** A prepending, 1106-byte, direct infector with a nasty payload. When an infected file is run on a Saturday, the virus overwrites one randomly-chosen sector on the C: drive. On the first day of any month, the virus displays a message containing the text: '> Combi - KOREAN code <'. The rest of the message is unreadable and probably requires a special (Korean alphabet?) driver. Other strings included in the virus code are: 'soniccc ????????COM' and '*.COM'.

```
Combi.1106          B42C CD21 B402 B901 0032 F6BB 0301 CD26 7203 83C4 02C3 B42C
```

Dark_Avenger.1783 — **CER:** An appending, 1783-byte, modified version of the Dark_Avenger virus. It contains the texts: 'Sydney!', ' -= Poisson & Sabber Live !!!! =-' and '(c) 19988-89 Dark Avenger. Mod. by Poisson'. The virus may be detected using the template matching some of the other variants.

```
Dark_Avenger.1783   A4A5 8B26 0600 33DB 53FF 64F5 E800 005E 81EE 6300 FC2E F694
```

Deinonychus.1000 — **CN:** An appending, 1000-byte direct infector which targets files in subdirectories. On 18 January it displays the message '—Deinonychus—' and tries to overwrite the first sixteen sectors of the C: drive. (Fortunately, the virus uses the older version of Int 26h, limited to small partitions and older versions of DOS.) It contains the texts: '*.C?M' and 'chklist.ms anti-vir.dat *.DBF'.

```
Deinonychus.1000    6A02 586A 1059 FA99 CD26 FB68 02FA 5868 4559 5ACD 21C3 FCB9
```

Dick.1242 — **CEN:** An appending, slightly polymorphic, 1242-byte direct infector. It contains the texts: 'Hello, This is Dick v2.02 Virus It's me again. Nice to meet You.!) Don't Worry, be Happy. I'll just to Say..... TURTLE I'll ALWAYS..... MISS YOU & LOVE YOU'; 'Written by Dick Kennedy in 1994.12.12 R.O.C PS:[80305] Love [80324] TRUTLE, Do YOU KNOW'; and '<This virus is [Dick v2.02] Written by Dick Kennedy>'.

```
Dick.1242           BD64 02BB ???? CC2E 81?? ???? 4343 CC4D 75F5
```

Entity.1997 — **CER:** An appending, encrypted, 1997-byte virus with stealth capabilities. It contains a lengthy message beginning: 'Hello user. I am a computer virus. My name is Entity.', and ending: '[This warning comes from Entity virus (c) 1995 by The Nuker]'. The time stamp of all infected files is set to 62 seconds.

```
Entity.1997         0000 5D81 ED03 018D B619 01B9 DB03 2E81 34?? ??83 C602 E2F6
```

Exe2Win.710 — **EN:** An overwriting, 710-byte, fast, direct infector containing the plain-text messages: 'This program requires Microsoft Windows', and 'EXE2WIN (Anti Windows) by [HtTM]'.

```
Exe2Win.710         B801 3DCD 2193 B440 B9C6 02BA 0001 CD21 B43E CD21 B44F CD21
```

Genvir.1504 — **CN:** A prepending, 1504-byte member of the Genvir family. It contains the texts: 'COMMANDE ?', '*.COM', and '[NuKE]-93'. It infects one file at a time and frequently hangs the system. The virus may be detected with the template used for the whole family [*see VB August 1994, p.4*].

Httm.80

CNO: An overwriting, 80-byte, direct infector which contains the texts: ‘Divide overflow’ and ‘[HtTM] *.Com’. The first message is displayed after all files in the current subdirectory have been infected.

```
Httm.80          CD21 8BD8 B440 B950 00BA 0001 CD21 720A B43E CD21 B44F CD21
```

Httm.87

CNO: An overwriting, 87-byte, direct infector which contains the text: ‘Divide overflow’ and ‘by [HtTM] *.Com’. The first message is displayed after all files in the current subdirectory have been infected.

```
Httm.87          CD21 8BD8 B440 B957 00BA 0001 CD21 720A B43E CD21 B44F CD21
```

Httm.112

CNO: An overwriting, 112-byte direct infector containing the texts: ‘Packed program is corrupted’ (displayed after all files in the current subdirectory are infected) and ‘1992 by [HtTM] *.Co?’.

```
Httm.112         CD21 8BD8 B970 00BA 0001 B440 CD21 720A B43E CD21 B44F CD21
```

Httm.475

CR: An appending, slightly polymorphic, 475-byte virus containing the encrypted text: ‘Die you loosy Motherfucker!’ and ‘[HtTM]’. When an infected file is executed, the virus installs itself in memory and hooks Int 21h. A bug in the code ensures that, after the virus has become resident, only the first *.COM file run will be infected properly – subsequent files will be corrupted instead. The virus always uses one of two encryption procedures, and may be detected with the following strings:

```
Httm.475         BE?? ??FC B8E7 0091 812C ???? A7E2 F9
Httm.475         BE?? ??FC B8E7 0091 8104 ???? A7E2 F9
```

Immortal.2174

ER: An appending, stealth, 2174-byte virus with the plain-text message: ‘IMMORTAL (c) 1994 by MW’.

```
Immortal.2174    1200 9500 8CC0 408E C033 FFB9 7E08 0E1F F3A4 06B8 B800 50CB
```

Inquisitor.625

CR: An appending, stealth, 625-byte virus containing the text: ‘Croatia must be free ! Croatian virus V1.1 (c) 1995 by Inquisitor’ and ‘CHKLIST.MS’.

```
Inquisitor.625   3D00 4B74 113D 034B 740C 3DAA FF74 03E9 E800 B8DC ACCF 9C55
```

IVP.Flipper

CEN: An encrypted, appending, 872-byte, fast direct infector. It contains the text: ‘Flipper In a Blender’, ‘BloodLust and friends’, ‘Flipper is in a blender’, ‘Eeeekk !!!! Eeeeeeekk!!’, ‘Eeeeeeeeeeeeeeeeeeeek!!’. The virus emits dolphin-like sound effects and after a few minutes invokes the print screen routine (Int 5h).

```
IVP.Flipper      8D9E 1501 B941 032E 8A27 2E32 A66C 042E 8827 43E2 F2C3
```

Kat.623

CR: An appending, 623-byte virus from Poland. It contains the text: ‘Dzieciatka wylewaja niewinnie lzy bo czuja nieszczescie choc go nie pojmuja... - KatVir by Warlock from III LO in Olkusz’. All infected files have the signature ‘KAT’ located at the end of the code.

```
Kat.623          8D8E B001 FFD1 07BE 2701 B905 00BF 0001 F3A4 1FB9 0001 FFE1
```

Mobius.231

CN: An appending, simple, 231-byte, direct infector. It contains the text: ‘ORION Mobius’ and ‘*.COM’.

```
Mobius.231       B603 028D BEE9 01F3 A4B4 408B D581 C200 01B9 E700 CD21 7214
```

Proto.720

CR: An appending, 720-byte virus which contains the plain-text message: ‘** ProtoVirus v1.0 by Chr’92 **’. The virus’ self-recognition is based on the value C3h (RET) located in all infected files at offset 4 (i.e. the virus does not infect files which begin: ?? ?? ?? ?? C3)

```
Proto.720        7416 80FC 3074 03E9 C500 81FE 3930 7403 E9BC 00BE 31D4 9DCF
```

PS-MPC.719

CN: An appending, PS-MPC based, 719-byte, direct infector containing the text: ‘VALENTINE HAS ENTERED’ and ‘This is the VALENTINE virus, v1.0 by Black Mantra’. The virus has a dangerous payload which overwrites the contents of drive C.

```
PS-MPC.719       B403 B001 B500 B101 B600 B202 FEC5 FEC0 CD13 E2EC B49C CD21
```

SillyC.787

CN: An appending, simple, 787-byte fast, direct infector. It contains the texts: ‘COR’ and ‘*.com’.

```
SillyC.787       E898 FFE8 89FF E80C FE8B C883 F112 E335 5659 1E5A B800 708E
```

SillyCR.303

CR: A simple, appending, 303-byte virus marking infected files with byte 2Bh (‘+’) at the end of the code.

```
SillyCR.303      50E8 0000 5EB8 FE35 CD21 81FB 0110 7449 BA01 10B4 25CD 218C
```

SillyCR.416

CR: An appending, simple, 416-byte virus containing a dangerous payload which corrupts the hard disk.

```
SillyCR.416      B807 51BE 5634 CD21 0BDB 7532 E440 A807 751A 33DB 8AE8 E440
```

Tip.554

CR: An appending, stealth, 554-byte virus with the plain-text message: ‘Virus program message from TIP to YSJ’. All infected files have their date-stamp modified to 27.05.81 and their time-stamp set to 0:01:56.

```
Tip.554          3628 0381 C6BB 02BF 0001 B903 00F3 A4B4 D7CD 2180 FD77 7434
```

Turbo.846

CER: An appending, 846-byte virus containing the plain-text strings: ‘*.exe’ and ‘TURBO.EXE’. All infected files have their time-stamp reset to 0:00:00.

```
Turbo.846        891E 3E03 8C06 4003 BAB2 00B8 2125 CD21 1F2E 80BE 4C03 5A75
```

VCL.Kmee

CN: An appending, encrypted, 847- or 848-byte virus. On 27 July, the virus displays the message: ‘Happy birthday SkyIron ! From mΣ to mΣ ! Viva Croatia ! I’m harmless virus Please don’t kill me... 26.07.197x => D day to remember... When I open my little eyes and Said “Kmeeee.”... Thank you Mother Nature... Osk’95...Croatia’. Variant 847 omits the terminating 0 at the end of the text and shows some meaningless ASCII characters after the above message. The virus may use one of two slightly different decryption engines. The following templates may be used to detect both mutations of both variants.

```
VCL.Kmee         9301 8135 ???? 4747 EB01 90EB 0190 EB04 90EB 0390 EBFB EBEA
VCL.Kmee         9301 8134 ???? 4646 EB01 90EB 0190 EB04 90EB 0390 EBFB EBEA
```

INSIGHT

Roger Riordan: Thoughts from Down Under

There was once a young boy whose passion was plants: at the age of ten, he was corresponding with research botanists. Although it was not to be his career, botany retained the boy's attention. Roger Riordan's particular interest is in ferns and orchids, examples of which pepper his garden at home.

The advent of World War II changed the planned course of his life: from prep school to boarding school, on to university, and then, had his father had his way, law. 'When the Japanese entered the war,' Riordan recalled, 'we moved to Mount Dandenong, and I went to the local state school.

'From Upwey I went to Trinity College (Melbourne University) to study engineering. I was accustomed to being top of the class, and I fear I became objectionably arrogant about it, but two events in my final year meant more to me than all my honours: I shared the "Wigram Allan Essay Arts Prize", and took out the same girl several times (this should give you some idea of the inadequacy of my social skills!).'

Before Viruses

As a student, Riordan had planned to specialise in nuclear power, and duly arranged a graduate apprenticeship in the field in England; however, seeing the *Atomic Power Department* skulking in its barbed-wire cage soon changed his mind. He went back to engineering, moving to mechanical engineering labs at Whetstone, near Leicester (England), where he worked on governors for water turbines.

He returned to Australia in 1957, spending 17 years at *CSIRO*, a government research organisation. International recognition came in 1967, when he designed the first practical high-performance gyrator using op amps. This period included one year as visiting lecturer at Berkeley (University of California), where he began work on circuit analysis. Not long after this, *CSIRO* installed computers: after learning the rudiments of Fortran, Riordan started writing a program for circuit analysis.

'We punched programs on cards,' he explained, 'which were taken by courier to a site 12 miles away. The program could run on the small (32KB, 24 bit words) computer there, or be transferred to mag tape, flown to Canberra, and run on their big computer (64KB, 48 bit words; i.e. 384KB and about $2,000,000). We got one or two runs a day, with an hour or so to find and fix bugs before we sent the cards off again.'

By 1973, fast approaching the age of 40, Riordan was dissatisfied with *CSIRO*: 'I decided it was now or never, so I resigned and set up *CYBEC Electronics* – did lots of interesting things, but never made quite enough to live on.

Stoned as Destroyer

'In 1983,' he continued, 'I took a job as a part-time lecturer in instrumentation and assembly language programming at the Chisholm Institute of Technology. In 1989, the PC lab was attacked by the Stoned virus. The lab used *Olivetti M24s*, which had a non-standard hard disk layout, and this normally harmless virus killed them instantly: it was a serious problem. After much effort, I discovered how Stoned worked, and how to disinfect it. This was difficult, as few people or books knew how a PC booted, and nothing we had could trace what was happening while the PC was booting.'

This incident led to the first version of *VET*, which Riordan gave out as shareware. In June 1989, he and some colleagues gave virus seminars, and again distributed *VET*, after which people began to register as users. Within the year, he was earning enough to retire, and work on the program at home.

How VET Grew

Principles have always been important to Riordan, and have helped him to establish these company ethics:

- We aim for technical excellence
- We trust everyone we deal with, including our staff
- We provide all the support our customers need
- We conduct all dealings with honesty and integrity

'I am sure,' he asserted, 'that these principles are responsible for our success, as our most effective sales force has been our many enthusiastic and loyal customers. In six years, *CYBEC* has grown from a backyard hobby to a company employing more than twenty people, with a turnover last year of AUS$2.5 million. Today, *VET* is recognised as one of the world's best half-dozen products.'

Somewhat ruefully, Riordan admits that despite the commercial success of the company, certain government departments and major Australian firms have repeatedly ignored *VET*, opting for overseas products which in some cases were markedly inferior to the home-grown package.

However, he is happy with *VET*: 'It took six years of hard work,' he said, 'but I was in the right place at the right time. A new program would have to handle the thousands of known viruses before it could be launched, and would cost a huge amount to write. It would be difficult to fund such a program, or for a new firm to assemble a full virus collection and establish the network of contacts needed to maintain it.'

Infectious Newcomers

Despite the fact that the growth in virus numbers no longer appears to be exponential, new viruses are and remain a problem to any anti-virus researcher, not least to Riordan.

The latest ‘novelty’ in virus-writing, Word Macro infectors, he sees as a threat. ‘Normal’ viruses, he thinks, infect well-defined classes of files with a fairly well known structure. ‘So it is easy,’ he said, ‘to say which files should be checked, and where to look for any viruses. Also, they can only be activated under clearly-defined circumstances, so it is relatively easy to detect them with resident scanners.

‘Almost any file could contain a macro virus. The file structure is not well known (we have only just been able to obtain definitions of file structure, and then only under a non-disclosure agreement), so it is difficult to do an intelligent scan, and prohibitive to do regular full scans of all files.

‘It needs a resident macro scanner to handle these viruses effectively,’ he continued, ‘but again it is not obvious how to build this, as the circumstances under which the macro is activated are neither well-known nor readily determined. The other problem with macro viruses is that they can run on multiple platforms, and will shortly be able to run on multiple (i.e. any *Microsoft*) applications. This may considerably complicate the design of a resident macro scanner.’

Polymorphic viruses, on the other hand, are in Riordan’s opinion no longer the problem they once were: ‘They have not proven the “killer app” that rendered scanners obsolete, as the jeremiahs predicted. It required a lot of work, and quite a bit of “code bloat” to overcome them, but most of the scanners have been able to keep up. They are also a lot of work to write, and are beyond most of the hack virus writers, so I suspect they will go out of fashion; why put in all that work to get just another polymorphic virus?’

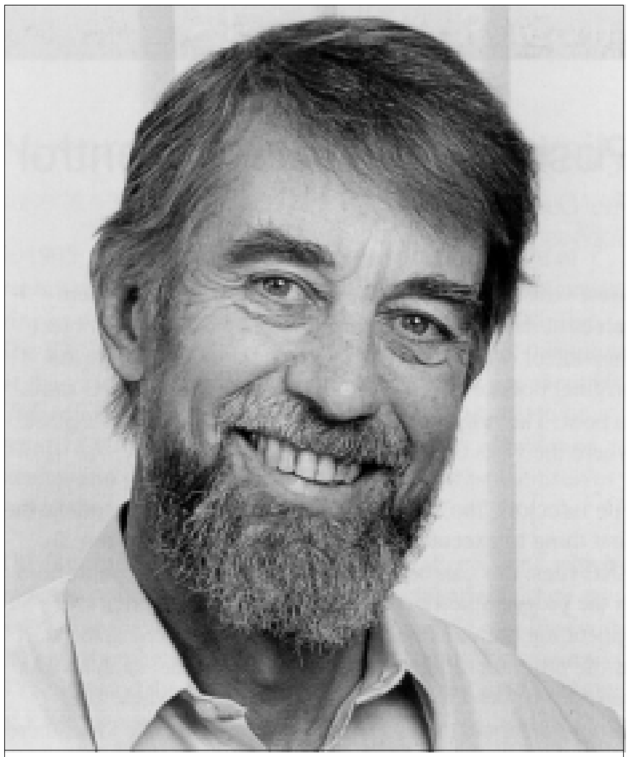
Many developers, in efforts to cope with the ever-increasing stream of new viruses, are turning to heuristic methods of detection, and Riordan is no exception: ‘I think nearly all the major programs incorporate some heuristic techniques already,’ he said. ‘Everybody will be trying to improve these, but they are only part of the answer, as they tend to generate false alarms, and false alarms almost certainly already cost users far more than actual virus incidents.

‘The most urgent problem facing the industry is to develop a resident macro scanner, which checks macros before they are executed. So far as I know, no-one has solved this yet.’

CYBEC

Riordan is proud of the fact that his company is as international as the computer scene in general. He boasts an international team: Jakub from Poland (otherwise known as *VB*’s erudite Technical Editor), David from Georgia, Oleg from Russia, Lise from the Seychelles, Peter from Malaysia, Jia Duong from China, and Frances from England. The General Manager, Andrew, is Australian, but like Riordan spent some years working overseas.

The company has offices in Sydney and in Adelaide, as well as its headquarters in Victoria. *CYBEC* has representatives in several countries outside Australia, and various avenues to build up export sales are being actively pursued.


Roger Riordan, part-time botanist – a remarkable addition to the anti-virus world.

Although no-one at *CYBEC* is interested in a takeover by one of the multi-nationals, discussions with other firms on ways in which they could work together co-operatively have taken place. Final decisions are yet to be made on joint projects. ‘We see ample scope in viruses for the moment,’ he stated, ‘but we are certainly looking for other opportunities.’

This year, the company began to increase its technical team, and establish a line of succession. Riordan plans to decrease the time he spends programming, perhaps writing more.

CYBEC, under Riordan’s guidance, also has a philanthropic side: the company recently set up two scholarships. The first, at Riordan’s *alma mater*, Trinity, will be awarded yearly to a student showing outstanding ability but needing financial assistance. The second is a fellowship in memory of his lifelong friend Jim Willis, the botanist he first met as a boy. There are plans to augment both funds and establish others.

The Other Side

Outside plants and computers, Riordan’s family occupies the rest of his time. Married for more than 30 years to Sally, an Englishwoman (naturalised Australian), they have three grown-up children – Riordan is, however, still to be found in his garden with them, playing with water rockets. What spare time remains is devoted to studying Australian history and to an interest shared with his wife; cooking – exotic dishes are a speciality.

Versatility is a word which well describes Roger Riordan, and it is certain that when he does (in the distant future) retire, his will not be an ‘old-age rest’, but another learning experience. Who knows – he might even begin a whole new venture!

VIRUS ANALYSIS 1

Positron – Grabbing Control

Paul Ducklin
Sophos Plc

Most viruses are greedy for control, and insinuate themselves at the start of the execution path through the object they infect. Most boot viruses write themselves over the original boot sector, so they get control when the PC starts to boot. The original sector is normally demoted to a place where the virus can refer to it when necessary.

File infectors, too, usually arrange things so their code is the first thing to execute when infected programs are run. In EXE files, this can be done by altering the entry point field in the program header. COM files can be subverted by appending the virus body to the file, then patching the start of the program to jump to this malicious code: such a virus can then be detected without examining the whole file. Simply compute the entry point, and look for the virus there.

Virus writers use many tricks to obfuscate entry points. Instead of patching a jump at the start of an infected COM file, for example, some viruses add redundant instructions followed by a jump. Others use an indirect jump, or a jump to a jump, or push an address on the stack and return to it.

One_Half takes the extreme approach of scattering small pieces of itself throughout an infected file, tying them together with jumps from piece to piece. Although it is easy to detect One_Half by tracing through these pieces and following the jumps, this means reading in tiny fragments from all over the file, which is time-consuming.

Confusing the Path

Positron, instead of infecting files immediately they are executed or opened, monitors their execution. It then locates an address within the program through which the flow of control has passed, and patches in a jump to itself there. Next time the program is run, control will pass to the virus – but only if the same portions of the program are executed.

As the execution path through a program may depend on many things, including such factors as available memory, date, time, command-line arguments and options selected, it may be that running an infected file will not actuate the virus. So, even tracing through an infected file or running it under a debugger will not guarantee that the body of the virus will be visited. Naturally, this frustrates detection.

Installation

Positron acquires memory in which to reside by manipulating the memory control block (MCB) chain directly. It shrinks the last MCB by 23 paragraphs, and copies itself into the gap. It creates a dummy MCB header at the start of this gap, flagging this new MCB as a Z-type block (‘Z’ denotes the end of the chain) belonging to the DOS kernel. Positron then hooks Int 21h by directly altering the interrupt vector table. Function 0Bh (Get STDIN Status) is made to do duty as an ‘Are you there?’ service: if called with 1944h in the BX register, the virus copies 1944h into AX and returns – a simple residency test.

Actuation

The virus intercepts function 4B00h (Load and Execute), so it is woken up when a program is about to run. At this point, the virus opens the file and reads the first two bytes. If the sum of these is A7h, the virus assumes this is an EXE file (A7h = ‘M’ + ‘Z’, the magic number denoting an EXE file) and gives up – it infects COM files only.

If it is a COM file, the virus saves the file handle for later use, and locates its System File Table (SFT) entry, examining it directly. First the date stamp is checked – if greater than the year 2080, the file is deemed already infected and the virus goes no further. Otherwise the time stamp is saved (to be restored later), and the file access mode manipulated within the SFT, giving the virus Read/Write access. Positron then sets a flag indicating that it is watching for an infection opportunity, and returns control to the DOS program loader.

Infection

Positron assumes the next Int 21h call is from the victim. The return address on the stack points to the instruction after the one which triggered the Int 21h call, denoting an offset in the victim program through which control is about to pass: a good point for the virus to patch in its control-grabbing jump.

The virus subtracts two from the return address on the stack, assuming the instruction just executed was two bytes long (it was probably ‘Int 21h’, which encodes to CD21h). The three bytes starting at this address are copied into the virus body, which is appended to the victim file, using the handle saved earlier during the DOS Load And Execute intercept.

At the point in the file which corresponds to the ‘INT 21h’ instruction identified above, a three-byte CALL instruction is patched, passing control to the virus body. The program file on disk is now infected, although the image executing in memory is still clean. Finally, time and date stamps are re-set, with 100 years added to the date (something usually unnoticed in directory listings, which generally omit the century).

Warhead

Despite the fact that there is no explicit warhead coded into the virus, it makes dangerous and potentially damaging assumptions. The first Int 21h call Positron intercepts after a

Load And Execute is likely to have been initiated by its intended victim, but if a memory-resident utility which hooks Int 21h (such as DOSKEY) is loaded after the virus has installed itself in memory, calls from the victim will pass through that resident program before they reach the virus.

This means the return address which the virus grabs off the stack will refer to the resident program, not to the victim file. So the file will probably be ruined during infection, as the virus will patch its CALL instruction in the wrong place.

Detection and Removal

Although entry-point-based virus scanners may have trouble detecting Positron, infected files are fairly obvious, as they include the unencrypted boast: ‘Positron (c) 1994 Evil Avatar’. Now that the Black Baron (Chris Pile, author of SMEG) is in jail, it will be interesting to see if Evil Avatar wishes to appear in an English court to defend the intellectual property right he claims above.

Disinfection of infected files is possible, but not recommended. Given the likelihood of the virus infecting incorrectly, and given that the virus infects only COM files, restoration from clean originals is the best bet.

Positron

Aliases:	None known.
Type:	Parastic COM infector. Control passes to the virus body with a call which may occur anywhere in the original file – its position cannot be determined trivially. Tracing from the program entry point is not guaranteed to bring control into the virus body.
Infection:	COM files.
Recognition:	Correctly-infected files grow by 512 bytes. The string ‘Positron (c) 1994 Evil Avatar’ appears in infected files.
Self-recognition in Files:	100 years added to the date stamp.
Self-recognition in Memory:	Int 21h called with AH=0Bh and BX=1994h returns with AX=1994h.
Hex Pattern:	`2E8F 06FE FF9C 601E 06B4 0BBB 9419 CD21 E800 005E 81EE 1300`
Intercepts:	Int 21h for ‘Are you there?’, actuation and infection. Int 24h for internal handling of critical errors.
Trigger:	None.
Removal:	Under clean system conditions, identify and replace infected files.

VIRUS ANALYSIS 2

Nutcracker – Monster Family

Eugene Kaspersky

In 1995, yet another ‘young talent’ entered the world of virus writing. He has thus far produced a family of several viruses which are difficult to analyse, detect and disinfect. The ID ‘1975’ appears in each: I would guess that this might be the year of his birth, which would make his age about 20. Unfortunately, he is diligent, and has bombarded us with a group of viruses which use different methods of infection, stealth, and polymorphism, and which manifest themselves in various ways.

The family has been named Nutcracker, after a word which appears in the internal text strings of each variant. Their most interesting features are infection of the middle of a file (with the polymorphic decryption loops at the end), trojanising SYS files and the MBR of the hard drive, and modification of EXE files in a manner similar to that of One_Half.

The last variant in the family, Nutcracker.AB7, is its most interesting member. It is multi-partite, stealth, unencrypted, exactly 2000 bytes long, and infects EXE files, the hard drive MBR, and floppy boot sectors. This analysis will be based on AB7: for an overview on other members, see p.11 (hex patterns for all variants are in Virus Summary Table).

Infecting the MBR

On execution of an infected file, control passes to the virus installation routine, which issues an ‘Are you there?’ call (Int 40h, AX=0BADh; AX=DEADh is returned). If the virus is in memory, control returns to the host program; otherwise the MBR is infected.

Before infecting the MBR, the virus checks that the addresses of the Int 40h and Int 41h handlers still point into the BIOS, then scans the BIOS to get the original Int 13h address for use during infection.

The virus reads the MBR into system memory, comparing it against 12 bytes of its own code to prevent multiple infection. It then obtains the hard drive parameter, using the Fixed Disk Parameter area (Int 41h), and saves the four sectors of its own code, as well as the original MBR, to the last sectors of the hard drive. Then, after overwriting 21h bytes of the MBR with virus-loading code, the virus saves the modified MBR (including this loader code) to the first hard drive sector.

The Virus in Memory

The virus stays resident only when loaded from an infected boot sector or MBR. The loader code (the 21h bytes placed in the boot sector by the virus) reads the rest of the virus into memory at address 0000:7C00h, stores the addresses of the Int 13h and Int 40h handlers for future use, hooks Int 1Ch, and returns control to the original boot sector.

When loading from a diskette, the virus uses the same technique as for file infection. To prevent duplicate infection of system memory, the virus checks the address of the Int 1Ch handler. If it points to segment 7C00h, installation aborts.

The virus ascertains when DOS is loading by hooking Int 1Ch (system timer tick). It checks the address of the Int 21h handler on each tick: when it changes, the virus unhooks itself from Int 1Ch and hooks Int 21h. Next, the virus waits for any program to execute, allocates a block of system memory, copies itself there, and hooks Interrupts 9h, 13h, 15h, 21h, 2Fh, and 40h.

To obtain the memory needed for its memory-resident copy, Nutcracker.AB7 first calls the XMS driver (usually HIMEM.SYS) to allocate a block of upper memory. If this fails (no XMS driver is installed, or there is insufficient upper memory), the virus allocates a block of conventional memory, then merges that with the previous block by manipulating the MCB chain.

Thus, the resident copy of the virus will be located either in upper memory or in an extension to a system driver memory block. This hides the virus well: standard memory browsers reveal nothing out of the ordinary.

Infection of Files and Diskettes

To infect files, the virus hooks Int 21h, and intercepts DOS calls Execute, Create, Close, and FindFirst/Next By Name. When the first file after the virus code has loaded is executed, the virus completes installation, and infects the hard drive MBR if it has been disinfected.

The virus decreases the length of infected files to make them appear uninfected by hooking FindFirst/FindNext calls. AB7 marks and identifies infected files by setting the seconds field of their date/time stamp to 58. Only new files are infected: when the virus detects that a file is being created, it stores the file handle and infects the file when it is closed.

When a file is infected, the virus first checks whether or not the file is on a *NetWare* drive – if it is, the infection routine aborts. Otherwise, it reads the first two bytes from the beginning of the file, and compares them with 'MZ', the EXE file marker.

Where the file is of EXE format, the virus checks the file length, infecting if it is 64KB long or less: the virus writes its body to the end of the file, overwrites the header with a three-byte JMP, stores its ID value, and returns control to the original Int 21h handler.

When the file is subsequently run, DOS will treat it as a COM file, because the 'MZ' marker has been overwritten. To let such a converted program run normally, the virus must itself carry out the operation which is usually performed by DOS when it loads an EXE file. Since COM files must be smaller than 64KB, the virus can only infect EXE files smaller than this. The first virus to use this technique was Yankee_Doodle.

It is not necessary to check files to prevent duplicate infection – the virus converts EXE to COM files, and does not infect any COM files; therefore, infected files will not be re-infected.

To infect the floppy boot sector, the virus hooks Int 40h and intercepts the Read Sectors function (AH=02h). When the boot sector is read, the virus compares 12 bytes of its own code with the contents. If the sector is clean, it writes in the 21h-byte loader routine (the same code as that used to infect the MBR), formats an additional track on the floppy, and saves its code into that area. The Int 40h handler also contains a stealth routine and an 'Are you there?' call handler (AX=0BADh).

Stealth Routines

This Nutcracker variant uses Int 13h and Int 40h to allow it to perform stealth functions – it is invisible in infected sectors whilst the virus is in memory.

The virus exhibits special behaviour with regard to disk-checking and CRC utilities: it hooks Int 9h and Int 21h, and when it spots such a utility starting, it removes itself from the MBR, reinfecting when the next program is executed or the computer rebooted from the keyboard.

Integrity checkers often retrieve the original Int 13h vector in an attempt to bypass resident viruses by calling Int 2Fh AH=13h (Get Disk Interrupt). The virus intercepts the call, disinfects the MBR, and sets an internal flag to force reinfection of the MBR when the next program is executed or on a warm reboot – the virus intercepts Ctrl-Alt-Del by hooking Int 9h.

The virus hooks two Int 15h functions (AX=9000h, 9001h), presumably to hide itself in a multi-tasking environment. Int 15h is called by the operating system when a device is busy, to allow multi-tasking programs to trap the interrupt and allow task switching whilst one task is waiting for an auxiliary device. (The subfunction values (00h and 01h) refer to the floppy and hard disk units respectively.)

While booting from an infected disk, the virus checks the system date, and on 12 January displays this message:

```
   I'm Nutcracker(AB7)!
```

Its most interesting technique is that which it uses to avoid detection by integrity checkers. AB7 is a slow infector, and hits only new files. Because a CRC is not available for a newly-created file, infected files are not detected; neither are any changes found: there are no suspicious memory blocks, existing files have the same CRCs, and the Master Boot Record is not infected. After exiting from the CRC utility, the virus reinfects the MBR, and the whole process begins once again.

The Nutcracker Family: Variants AB1 – AB6

- AB1 is polymorphic and parasitic. It infects COM and EXE files on Execute, Open, Rename, or any access to the file's attributes. It converts EXE files to COM format on infection. If video refresh is supported, the virus hooks Int 8h (timer), enabling/disabling video refresh on each call, which produces an effect not unlike slight interference. When infecting, it first copies a section (which it encrypts) of the host program to the end of the file, and places its encrypted body in the middle of the target file, over the section of code copied out of the way.

- AB1.Antarex is polymorphic and parasitic, infecting COM and EXE files on access, and trojanising SYS files. If an error occurs during installation, it erases the CMOS and reboots the PC. Before returning to the host program, the virus looks for files in the current directory of drive C and forces infection by accessing their attributes.

  It encrypts the header of EXE files which are 64KB or larger in size, and replaces the 'MZ' stamp with 'AB'. Thus the first sector of large EXE files is corrupted, causing the PC to hang when such a file is executed under clean system conditions. When the virus is resident, corrupted sectors are decrypted on-the-fly, and the file executes normally. Clean-booting may reveal a corrupted system, and removing the virus will destroy the decryption key contained within it.

- AB1.Antarex.A infects BIN, COM, and EXE files, corrupts EXE files, and trojanises SYS files. If an error occurs whilst the virus attempts to install itself, it will erase the CMOS and reboot the PC. As AB2, it plays (depending on system timer) the theme from a Russian cartoon.

- AB2 is multi-partite: it infects COM and EXE files, the hard drive MBR, and diskette boot sectors. When an infected file is executed, the MBR is infected and the virus does not stay resident. If the PC is booted from an infected diskette, the virus infects the hard drive MBR, and installs itself in memory. It waits for DOS to load by hooking Int 1Ch and monitoring the Int 21h vector on each timer tick – once DOS has loaded, it hooks Interrupts 13h and 21h.

  When the first file executes, the virus appends its code to the end of the last used memory block, and uses its Int 21h hook to intercept file access calls, allowing it to infect files and trojanise SYS files. The Int 13h hook is used to infect floppy boot sectors and corrupt EXE files. The virus will hang if the PC is *Pentium*-based, or if the virus is run under a debugger.

- AB3, AB4, and AB5 all hook Int 21h on execution, and write themselves to the end of COM and EXE files on Execute, Open or Rename. When a file is created, its handle is stored, and it is infected when closed. The virus perform stealth routines on FindFirst/Next, Seek, and Read calls. These viruses may delete *.?AS files when they are accessed (conditions differ for each variant) and other specific files when opening.

  On 12 January and 23 July, AB3 erases sectors on drive C. When an infected file is executed 23 days after its initial infection, the virus hooks Int 10h and slows down the PC by delaying each Int 10h call.

  AB4 disinfects files under a debugger, and places a Trojan in the MBR, which, on 12 January and 23 July, formats hard drive sectors. By means of a counter stored in the boot sector of an infected disk, the virus marks a random sector on the current drive as bad on every 64th program execution.

  AB5 deletes *.MS files, and trojanises the MBR to count the number of boot-ups. On the 511th boot, the Trojan reformats hard drive sectors, erases the CMOS, and displays the message: 'Gloomy Nutcracker (AB5) from the city of Brest (BY) with best wishes! Only the hope dies last!..'

- The AB6 variants are multi-partite stealth infectors. When an infected file is executed, the viruses infect the hard drive MBR, hook Ints 13h, 17h, and 21h, and search and hit COM and EXE files on drive C. They stay memory-resident, writing themselves to the end of COM and EXE files when they are accessed.

  While loading from an infected MBR, they hook Int 17h and Int 1Ch, wait for DOS to load, and then hook Int 13h and Int 21h. They do not change the size of available RAM, but instead modify the MCB chain directly. Int 13h is used to hide infected sectors, and Int 17h to change characters (occasionally) during printing. When CHKDSK is executed, they temporarily disable some stealth routines to avoid errors caused by CHKDSK receiving conflicting information about file length. They also delete *.FW and *.?AS files, and try (but fail) to delete *.MS files.

  On 12 January, while loading from an infected MBR, the viruses format hard drive sectors, erase the CMOS, and display the messages: 'Dreary Nutcracker(AB6) Lives' (AB6.a), 'Dreary Nutcracker(AB6) Lives Again' (AB6.b), 'Dreary Nutcracker(AB6)' (AB6.d), and 'Dreary Nutcracker(AB6) lives forewer !' (AB6.d).

Nutcracker.AB7

Aliases:	None known.
Type:	Multi-partite, stealth, memory-resident.
Infection:	EXE files, hard drive MBR, floppy boot sector.

Self-recognition in Files:

The seconds field in the time stamp is set to 58.

Hex Patterns – for all Variants:

[NB Where no pattern is given to locate the virus in files, this is because it is polymorphic, and no useful pattern is possible.]

AB1

```
in Memory:  8BEC 4444 816E 0007 008B 7600
            BA75 19BB 7619 B8AB 1ECD 21FA
            8CD8 488E D880 3D5A
```

AB1.Antarex.a

```
in SYS Files: 2EC7 0606 00?? ??9C 5051 52F8
              B404 CD1A 7205 80FA 1274 095A
              5958 9D2E FF26 0600

in Memory:    4444 8BEC 816E FE07 008B 76FE
              BA75 19BB 7619 B8AB 1ECD 21FA
              B97D 018C D848 8ED8
```

AB1.Antarex.b

```
in Memory:  8BEC 4444 816E 0007 008B 7600
            BA75 19BB 7619 B8AB 1ECD 21B9
            5B01 B449 CD21 F91B
```

AB2

```
in SYS Files: 2EC7 0606 00?? ??9C 501E 2BC0
              8ED8 813E 7200 007C 740E 803E
              F605 EA74 07F6 066C

in Sectors:   E800 005F 2BC0 8ED0 BC00 7C8E
              C4BA 0001 B9?? ??BB ???? CD13
              B806 02CD 138A E372

in Memory:    9CFB FC2E FF06 4101 3DAB 2075
              1081 FB75 1975 0A2E FF06 4501
              B802 AB9D CF2E 803E
```

AB3

```
in Files:   E800 005E 81EE 0300 9C50 5351
            5257 551E 061E 062B C08E D8B4
            30CD 2186 C4C4 1EB2

in Memory:  E800 005E 81EE 0300 9C50 5351
            5257 551E 061E 062B C08E D8B4
            30CD 2186 C4C4 1EB2
```

AB4

```
in Files:   E800 005E 81EE 770A 9C50 5351
            5257 1E06 8CDB 83C3 102E 019C
            1A0C 2E01 9CAB 03B4

in MBR:     33C0 8EC0 8ED8 BB00 7CFA 8ED0
            8BE3 FB50 53BF 0006 508D 7524
            9056 8BF3 B975 0090

in Memory:  E800 005E 81EE 770A 9C50 5351
            5257 1E06 8CDB 83C3 102E 019C
            1A0C 2E01 9CAB 03B4
```

AB5

```
in Files:   8BEC 4444 816E 00BD 088B 7600
            2E81 BC4D 014D 5A74 0B44 4489
            6506 8BE6 81C4 540C

in MBR:     2BC0 509D 8ED8 8EC0 BB00 7C8E
            D08B E3BF 0007 8D75 1E56 8BF3
            B93B 01F3 A4C3 99CD

in Memory:  8BEC 4444 816E 00BD 088B 7600
            2E81 BC4D 014D 5A74 0B44 4489
            6506 8BE6 81C4 540C
```

AB6.a

```
in Files:   9C56 5750 5351 521E 062B FF57
            9DE8 3500 C406 4C00 2E89 8438
            022E 8C84 3A02 C406

in MBR:     2BDB 539D 8ED3 BC00 7CBA 4000
            8EDA 836F 1304 8B47 13B1 06D3
            E08E C026 C607 5A26

in Memory:  9C56 5750 5351 521E 062B FF57
            9DE8 3500 C406 4C00 2E89 8438
            022E 8C84 3A02 C406
```

AB6.b

```
in Files:   9C56 5750 5351 521E 062B FF57
            9DE8 3500 C406 4C00 2E89 843B
            022E 8C84 3D02 C406

in MBR:     2BDB 539D 8ED3 BC00 7CBA 4000
            8EDA 836F 1304 8B47 13B1 06D3
            E08E D8C6 075A C747

in Memory:  9C56 5750 5351 521E 062B FF57
            9DE8 3500 C406 4C00 2E89 843B
            022E 8C84 3D02 C406
```

AB6.c

```
in Files:   9C56 5750 5351 521E 062B FF57
            9DE8 4300 C406 8400 2E89 843B
            0A2E 8C84 3D0A 2E89

in MBR:     33C0 FA8E D0BC 007C FB8E C48B
            D8CD 13B9 0B00 B807 02CD 1373
            02CD 1806 B83F 0050

in Memory:  9C56 5750 5351 521E 062B FF57
            9DE8 4300 C406 8400 2E89 843B
            0A2E 8C84 3D0A 2E89
```

AB7

```
in Files:   8B36 0101 8D84 0301 2D21 0050
            B104 D3E8 8CCA 03D0 52B8 3B00
            50CB 2BDB 2E88 1ECE

in Sectors: 2BDB FA8E D3BC 007C 8EC4 FBB9
            ???? BA?? ??2A E4CD 13B8 0402
            CD13 72F5 EAF5 0000

in Memory:  8B36 0101 8D84 0301 2D21 0050
            B104 D3E8 8CCA 03D0 52B8 3B00
            50CB 2BDB 2E88 1ECE
```

Intercepts:	Int 13h for stealth, Int 40h for stealth in sectors and floppy infection, Int 1Ch to detect DOS loading, Int 15h, Int 2Fh and 9h (see text), Int 21h (various functions for file infection and stealth).
Trigger:	On 12 January, while booting from an infected disk, displays message.
Removal:	Under clean system conditions, identify and replace infected files and sectors.

VIRUS ANALYSIS 3

Keeping the Peace?

Igor G. Muttik
S&S International PLC

Polymorphic viruses were created to defeat anti-virus scanners – they attempt to hide their code by continually changing it; by masking it with an ever-changing decryption loop. Most modern polymorphic viruses use this approach: they just conceal the decryption loop (or loops) in garbage instructions, and make the decryption commands variable.

However, there is a class of polymorphic viruses which tries to hide not only the decryption loop, but also the entry point to a virus or its decryptor. The first virus to use this technique was probably Commander_Bomber, followed by the One_Half family and the later members of the Uruguay family. Now the same ideas have returned, this time in the Peacekeeper viruses.

JMPing Around

Hiding a virus' entry point is, in many cases, an easier task than the creation of a true polymorphic engine. Many different methods can be used to make it extremely difficult to determine the true entry point of the file. Viruses can use a chain of jump and call instructions; they can calculate an entry point address, place this value in a register and jump there (using many different techniques – for example, RET/RETF/IRET/JMP reg/CALL reg, etc.); and so on.

A long chain of jumps is very inconvenient for an anti-virus scanner, especially if the jumps point all over the file. Normally, scanners follow the jump chain only to a very limited depth; otherwise, they would be too slow. Following the chain too deeply would require loading much more of the file from disk, even on a clean machine, and that takes time. This is why, in some cases, viruses such as Commander_Bomber are detected through scanning the file in its entirety.

Peacekeeper

The Peacekeeper virus is a parasitic resident file infector. It is encrypted and polymorphic, and the virus body and decryptor are attached to the end of the victim file.

The virus infects COM and EXE files as they are executed, but it treats the two types of file in very different ways.

COM File Infection

Peacekeeper infects COM files by creating a number of 'jump islands' – each island transfers control to the next one. The first island is at the very beginning of the file, and the others are located between there and the virus decryptor, at random locations within the victim file. The size of the islands is also random. The decryptor, however, is always located just before the main body of the virus.

The number of jump islands in an infected COM file is, as stated above, variable. This is very inconvenient for automatic analysis. Fortunately, there is a upper limit: the virus never generates more than five islands. In addition, the chain of execution from island to island can never move backwards in the file, and each island is further along the file than the one executed before it.

To run its host program, the virus needs to restore the memory image of its host file. All jump islands need to be replaced with the original code. The virus saves all the necessary information when it infects a file; and data overwritten by the jump islands is saved at the very end of the file (after the virus body and before the DEADh marker).

In addition, every infected COM file carries a marker which the virus uses for self-recognition. The last two bytes of the file are set to DEADh. This fact can also be used by scanners as a final check to determine whether or not the file is infected, but is clearly not reliable for use as the sole detection technique.

EXE File Infection

It seems that the author of Peacekeeper was too lazy to implement an island infection technique for EXE files: it is much more complex to do carry this out on EXE files than on COM files. Therefore, EXE files are infected using a much simpler (and more standard) technique.

In EXE files, the decryption loop, which is polymorphic, is placed at the file's entry point. However, the degree of polymorphism is rather low – for example, the decryption loop always contains the bytes 2Eh 81h 34h (memory modification: XOR [si],const) or 2Eh 81h 35h (XOR [di],const), and E2h (LOOP).

Infected EXE files also carry the virus marker (DEADh), but in these files it is situated in the EXE header, in the file checksum field (the word at offset 12h in the file).

The Variants

There are two known variants of this virus; Peacekeeper.a and Peacekeeper.b. Both contain the same internal strings:

```
Peace-Keeper Virus V2.10 Written by Doctor
Revenge 18-May-1994 , Italy'
[MCG v0.31 ß]
```

Careful comparison shows that the two viruses are essentially identical. The main difference between them is that Peacekeeper.b uses LEA instructions in many of the places, where the former uses MOVs. As an LEA occupies four bytes and a MOV uses only three, Peacekeeper.b is about 30 bytes longer than Peacekeeper.a.

Installation

When an infected COM file is executed, the first jump island gains control. Each island should always be terminated by a JMP instruction (E9h), which passes control to the next island in the chain. In some cases this JMP is followed by a number of polymorphic instructions. Obviously, this is a bug in the virus' 'island generator'.

There is one other, more serious, bug: when it infects short COM files, the virus frequently overwrites the decryptor with the last island. Victims corrupted in this way will hang when they are run.

Provided none of these bugs are present, the last jump island passes control to the decryptor, which uses the XOR instruction to decipher the virus body backwards. When decryption is finished, control passes to the virus body.

"although the increase in length of an infected file is random … the stealth routine shrinks the reported size by a constant amount"

The virus body first intercepts the single step interrupt (Int 01h). Although the code is rather long, its purpose is merely to prevent execution being traced with a debugger. The virus then obtains the current DOS version: Peacekeeper is much more stable under higher DOS versions. Samples usually hang under DOS 3.30, while they replicate perfectly under a newer version (e.g. DOS 5.0).

Next, the virus issues an 'Are you there?' call; Int 21h, AX=DEADh. If both the AX and the BX registers are set to DEADh on return from the interrupt, the virus decides that it is already resident, passing control to the code which restores the victim file image in memory (for COM files). Control then returns to the original entry point.

If the virus is not resident, and the version of DOS being used is 3 or higher, it issues calls to Int 16h (AX=FA00h, DX=5945h) and Int 16h (AX=FA01h, DX=5945h). These calls are intended to subvert certain memory-resident anti-virus utilities.

It then reduces the size of the last Memory Control Block (the Z-block), to create 183h paragraphs (6192 bytes) for itself, and copies itself into that area. At this point, it intercepts Int 13h and Int 21h and issues an Int 21h call (AX=DEAEh), which simply randomises the virus' random number generator. Finally, it repairs the memory image of the host file and jumps to its entry point.

Interrupt Handlers and Payloads

The virus goes resident at the top of conventional memory, above the last MCB (a technique sometimes referred to as TWIXT). The Int 21h handler intercepts the following DOS functions: DEADh ('Are you there?'), DEAEh (randomises the built-in random number generator), 4B00h (Execute), 11h and 12h (FindFirst/Next File using FCB).

The last of these functions allows the virus to perform its partial stealth functionality, which is implemented in a very peculiar way. Although the increase in length of an infected file is random (though usually between 3840 and 4450 bytes), the stealth routine shrinks the reported size by a constant amount: 3800 bytes for Peacekeeper.a; 3820 bytes for Peacekeeper.b. So, a change in the length of a file is visible even with a standard DOS 'dir' command, although it is not the true size change (the visible change is from 40 to 150 bytes). In addition, Int 21h functions 4Eh and 4Fh (FindFirst/Next File by name) are not intercepted, so many programs will report correct file sizes.

Int 13h is also hooked, but this code actually does nothing. Careful examination shows that the most likely intention of the virus writer was to use this handler for a payload. It would appear that his intention was to cause one in every 65536 disk accesses to fail silently.

However, this code is inaccessible (rather than a conditional jump, an unconditional jump is present), so the payload does not work – and this is not the virus' only inactive payload! The virus also has code to place its own copyright message into the boot sector of disks used in the A: drive. But again, it is never executed.

File Infection

The time stamp and attributes of files are preserved by the virus, and restored after infection. Also, the virus tries to avoid infecting anti-virus software. It contains an exclusions list, and all programs starting with the following two-letter combinations will not be infected: SC, CL, VI, VS, MS, CP, F-, IM, VH and TB. These stand for SCAN, CLEAN, VIRUSCAN, VIVERIFY, VSHIELD, MSAV, CPAV, F-PROT, VHUNTER and TBSCAN – these are all popular anti-virus programs.

Peacekeeper's polymorphic engine is not very powerful. One of the most noticeable peculiarities of the generated code is that it contains a large number of conditional jumps which have zero jump offset (such as JNO $+2, JB $+2 … JLE $+2, JG $+2, etc., represented by the opcodes 7000, 7100, …, 7E00, 7F00). These cause control to pass to the next instruction, whether or not the condition is met.

In addition, as with many polymorphic engines, the code generated is full of the very simple one-byte do-nothing instructions NOP, CMC, CLC, STC, CLI and CLD (NOP and CMC are twice as frequent as the others). Among the two-byte do-nothing commands, CMP, TEST and XCHG are the most common.

The virus has an internal random number generator and an associated 'randomise' function: this is very common in polymorphic viruses. The randomise function is called when the virus goes memory resident: it uses the DOS Get System Time function in addition to reading from I/O port 40h (Timer counter). It seems that the author of the virus believes that complex and obscure random number generators are the best: he uses a complex mixture of SHL, XOR, INC and LOOP instructions.

Conclusion

It seems now that the idea of hiding the entry point of viral code through the use of complex jump constructions is becoming more and more popular amongst virus writers.

Although the Peacekeeper viruses are interesting enough to merit their analysis, I do not have the impression that the writer is a talented programmer. With almost no effort, the virus code could be made shorter by some hundreds of bytes, and the bugs discussed above removed.

Peacekeeper

Aliases:	MCG-Peace.
Variants:	Peacekeeper.a, Peacekeeper.b.
Type:	Memory-resident, semi-stealth, encrypted polymorphic file infector.
Infection:	COM and EXE files.
Size:	Variable. Minimum is 3800 for Peacekeeper.a and 3830 for Peacekeeper.b.
Self-recognition in Memory:	Int 21h, AX=DEADh. Response is AX and BX set to DEADh.
Self-recognition in Files:	DEADh marker (COM: word at the end of file; EXE: word at offset 12h, file checksum field)
Hex Pattern in Memory:	`E800 005E 83EE 03EB 4790 2E80 3E23 0B00 7405 80FC 0374 062E`
Intercepts:	Int 01h (temporarily, for armouring), Int 13h (for [disabled] payloads), Int 21h (for infection and semi-stealth), Int 24h (temporary, while infecting).
Trigger:	None.
Payload:	None (but see analysis).
Removal:	Under clean system conditions, locate all infected files and replace with clean copies.

TECHNICAL NOTES

Hot 95: New Kids on the Block

It seems to be the way things work here at *Virus Bulletin* – there are often new developments in the virus world immediately before we go to print.

This month we have two interesting new viruses: the first genuine *Windows 95* virus (not believed to be in the wild at present), and another *Microsoft Word* virus, which has appeared in the wild in Russia.

95 Infections

This virus, for which there is currently no final name (those so far proposed include 'V32' and 'Boza'), is fairly simple in virus terms, but the fact that the virus is the first of its genre makes analysis that much more complex.

The virus is a simple, non-resident, direct action file infector. When an infected program is run, the virus immediately locates and infects three executables in the current directory. Occasionally, however, it produces corrupted replicants.

The virus contains an extremely simple payload, triggered when all executables in the current directory have been infected. When this has been achieved, the virus first checks whether the current directory is the root. If so, the virus randomly displays the dialog box shown in Figure 1.

In addition to the messages displayed in the dialog box, the virus also contains the following text message, which does not appear to be used:

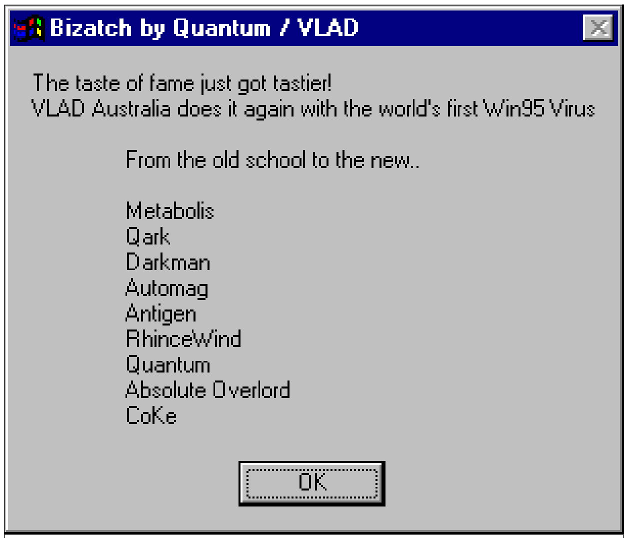
```
Please note: the name of this virus is
[Bizatch] written by Quantum of VLAD.
```

A Brief Analysis

The virus uses standard Win32 APIs throughout, and has basic knowledge of the format of both the PE (Portable Executable) file format and the DOS stub, which allows it to modify *Windows 95* executables correctly. This is achieved by adding a new section (a section in a PE file is roughly analogous to a segment in an NE, or 16-bit *Windows*, file) with the identifier '.vlad', and adjusting the PE header to account for the new section and to modify the program's entry point to it.

It is not known at this stage why this virus occasionally corrupts files it infects, but by and large it is perfectly capable of replicating.

The virus' self-recognition in files is fairly simple: it makes use of an unused field in the PE header; at offset 4Ch is a word labelled as 'Reserved1' by the *Microsoft* documenta-



Bizatch by Quantum / VLAD

The taste of fame just got tastier!
VLAD Australia does it again with the world's first Win95 Virus

From the old school to the new..

Metabolis
Qark
Darkman
Automag
Antigen
RhinceWind
Quantum
Absolute Overlord
CoKe

Figure 1: The dialog box displayed by the newly-discovered *Windows 95* virus.

tion. The virus replaces the value in this field (which always appears to be set to zero in uninfected files) with F00Dh. It uses this to check that a file it is about to infect is not already infected.

The following byte-pattern may be used to detect the virus in byte-by-byte search mode:

```
E809 4400 E800 0300 0066 81BD
200A 4400 5045 0F85 A601 0000
6681 BD6C 0A44 000D F00F 8497
```

For this type of virus, however, it will be perfectly possible for scanners to perform the same type of efficient entry point scanning as they do for other files. The entry point can be read from the header and searching can start from there.

WinWord/Hot

The second of the new viruses which arrived at the end of January was another *Microsoft Word* macro virus. Tentatively named ‘Hot’ after its internal identifier ‘QLHot’, it is notable for a few new techniques, in addition to its destructive trigger.

Basics

The virus is installed in the Global Template in much the same manner as Concept [*see VB September 1995 p.8; October 1995 p.3*] – an infected document contains an AutoOpen which copies macros into NORMAL.DOT. Unlike Concept, however, Hot also places an AutoOpen macro in the Global Template, and it is this macro which contains the trigger.

All macros used by the virus are ‘encrypted’ using standard *Word* functionality. *Word* refers to it as ‘execute-only’, because it prevents the macros being edited by *Word*, and presumably also because the encryption is achingly trivial.

The complete list of macros used by Hot, their names in NORMAL.DOT and standard infected documents, is:

NORMAL.DOT	Documents
`AutoOpen`	`DrawBringInFrOut`
`StartOfDoc`	`AutoOpen`
`InsertPageBreak`	`InsertPBreak`
`FileSave`	`ToolsRepaginat`

InsertPageBreak is used by the virus to detect whether or not NORMAL.DOT is already infected, in much the same way as Concept uses PayLoad. The macro mirrors the standard behaviour of the menu option Insert/Page Break.

From the user’s point of view, infection is standard: once the Global Template is infected, each document saved using File/Save is converted to a Template and infected.

Other Techniques

Hot is the first *Word* Macro virus to utilise functions held in an external library; by using the standard *WordBasic* syntax ‘Declare Function’. This allows macros to call any standard Windows API they wish, simply by predeclaring it – the situation is somewhat analogous to the standard programming practice of calling a DLL or shared library. It uses the standard APIs GetWindowsDirectory(), _lopen() and _lclose() (found in the library KERNEL.EXE) in this way.

Trigger

When the virus infects the Global Template, it puts an entry, ‘QLHot=<date>’, in the ‘[Microsoft Word]’ section of the WINWORD6.INI file in the *Windows* directory. ‘<date>’ is a number representing a date fourteen days after the current date; i.e. the date of initial system infection.

Every time the Global Template’s AutoOpen is called (every time a ‘document’ without an AutoOpen macro in it is opened), Hot checks the date against that stored. After the initial fourteen-day pause (to allow the virus to spread undetected), for ten days out of every seventeen there is a one in seven chance that the payload will be activated.

If the payload is triggered, the virus selects and deletes all text in the newly-opened document, resaves it with confirmation turned off, and closes it, effectively destroying its contents. There is, however, a special case: if the file C:\DOS\ega5.cpi exists, and the virus succeeds in opening it (using the _lopen() function from KERNEL.EXE discussed earlier), the file is simply closed without loss of data.

Conclusion

Hot was obtained from the wild in Russia, and only discovered because it had started to trigger and destroy documents. This means that Hot had been present on that site for at least fourteen days: in that time it could have spread widely. The following pattern may be used to identify infected files:

```
A186 9DAD 889D 8CA7 86CD E58E
0369 EC8E EE69 EC8E E868 ECEF
```

FEATURE

Regina v Christopher Pile: The Inside Story

Jim Bates

With the Black Baron case still relatively fresh in the minds of computer users, now is a good time to put the facts on record. As I became involved at an early stage, I can speak with some authority on what happened and how the case against Christopher Pile was built.

Preliminary Enquiries

On 13 July 1994, the Police obtained a warrant to search Pile's home, under Section 14 of the *Computer Misuse Act 1990*. That search revealed an old *Sinclair Spectrum* computer in a bedroom wardrobe: nothing more. Pile affected disinterest in anything to do with computers.

His bedroom did, however, contain a table boasting a newly-installed telephone extension cable. The Police were fairly sure that Pile knew they would be calling, so such a large 'computer-shaped hole' was suspicious. Pile was cautioned, arrested and taken to the Charles Cross Police Station in Plymouth (southwest England).

I then went to another address in Plymouth where the police were conducting a simultaneous search in connection with the same enquiry. Here, a quantity of computer equipment was found in the living room, which was labelled, packed, and transported to the Police station.

In a bedroom at the same address, on top of a wardrobe, a box containing a *Tandon* computer, keyboard, modem, mouse and around fifty diskettes was found. The occupant of the house, on being questioned, indicated that the *Tandon* belonged to a friend, and was merely being stored there. This too, along with the occupant, was taken to the Police Station. The occupant was subsequently cleared of involvement in Pile's activities.

At the police station, both men (each initially unaware of the presence of the other) were questioned further. Enquiries centred around a known series of telephone accesses to certain BBSs around the UK and the uploading of virus-infected programs to them. During preliminary questioning, Pile denied any recent knowledge of computers, saying he had disposed of his machine some time around the previous November (1993).

It is my understanding that the police at this stage already had sufficient evidence to charge Christopher Pile with offences under the *Computer Misuse Act* and were anxious to complete their enquiries before a complete list of formal charges was preferred.

Analysing the Equipment

Meanwhile, I was creating image copies of the machine's fixed and floppy disks before beginning initial examination of their content and structure. Preliminary analysis of the first computer showed it to be a standard machine in a state indicating normal use by someone involved in programming graphic images for computer games. The *Tandon*, however, had been completely defragmented and wiped, destroying almost all traces of previous activity.

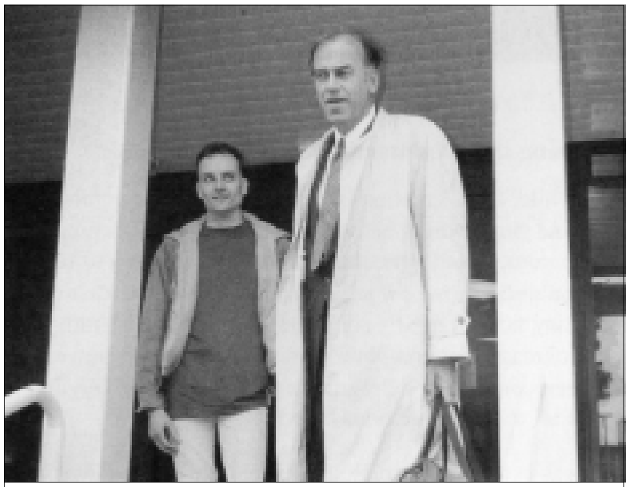
This was extremely suspicious, and a more detailed examination was begun. Fairly quickly, this revealed two document files, both of which were job applications in the name of Christopher Pile name, thus establishing that the machine was probably Pile's. The files were printed off and given to the investigating officers.

When imaging of the floppy disks was completed, they were examined, and shown to be commercial disks for *Windows*, *MS-DOS*, a modem, and mouse and printer drivers. Image analysis was then begun: the first stage highlighted a file on a manufacturer's diskette found in the mouse box. The file, MOUSE.DAT, showed sufficient indication of being unusual that it was marked for further analysis. It had the same date and time as the other files on the disk, and closer examination revealed that it contained encrypted information.

The Accusations Admitted

The immediate results of this analysis, together with the printouts, were passed to the investigating officers, who began a second interview with Pile. He denied any knowledge of the computer until he was shown the job application

Virus expert Jim Bates, whose testimony (along with police efforts) was instrumental in obtaining a conviction.


Pile with his solicitor on the steps of Plymouth Crown Court, after being charged with distribution of viruses.

documents. At that point he asked for a private word with his solicitor and subsequently admitted owning the machine, although he still denied any knowledge of viruses.

Questions were then put to him concerning telephone calls made from his address over a period of time to various BBSs in the UK. Pile eventually admitted making these calls on his computer, but although the timing of the calls was a strong indication that he was responsible for certain infected uploads, he continued to deny involvement with virus code.

Eventually, Pile was asked about the MOUSE.DAT file on the floppy, at which stage he asked to speak with his solicitor. On resuming the interview, he admitted everything, and gave the Police the password of the encrypted file. On decryption, the file was found to contain source code to a number of viruses, as well as documents by Pile and others which were directly concerned with viruses. After making his confession, Pile was released on bail while further enquiries were conducted.

This concluded the initial investigation: I returned to my office with copies of all the images to analyse. I was later asked to produce a report, together with any related material I could find, on the contents of the file MOUSE.DAT. There were small evidential traces on the *Tandon*: in the light of Pile's confession, analysis of these was not required.

I wrote a report on the evidence and sent this to the Police. Over the next few months I received a regular supply of virus-infected files sent to me by the Police from various complainants. Each of these files had to be confirmed as one of Pile's creations and analysed to reveal the value of the generation number stored within them.

At this time, a number of reports of infection by Pathogen or Queeg were reported from various quarters, then denied: some complainants were presumably being gagged by their companies. Happily, however, many specimens were received from other sources, and were analysed as the evidence mounted.

Pile was interviewed further by the Police, and ten charges were framed under the *Computer Misuse Act*: five of unauthorised access, five of unauthorised modification. The trial date was set for May 1995 at Plymouth Crown Court.

Approaching Sentencing

As the papers were passed to Counsel before the trial, a further charge of incitement was added. At the trial the defence objected to the introduction of the charge of incitement, but the judge allowed it to stand. Pile pleaded guilty to all eleven charges.

The defence then applied for permission to commission their own technical report on the viruses so that this could be presented before sentencing. This was allowed, and Pile was remanded on conditional bail pending the setting of a date for sentencing.

In the interim, efforts were made to confirm the existence overseas of the viruses and the file SMEG03.ZIP – in this task, I had some assistance from other anti-virus researchers. Vesselin Bontchev helped with a statement confirming SMEG's existence on the continent, and other enquiries confirmed that it had been spread fairly rapidly and widely amongst virus exchange BBSs in various countries.

"the judge considered that the distribution of (viruses) was certainly the most serious charge brought before him"

Meanwhile, I continued to analyse examples of infection by these viruses. Amongst these was one from a Nottinghamshire college which had suffered quite severely and exhibited the highest generation number so far found – 27. This validated my assertion that continued infections would show increasing generation numbers until the destructive payload was delivered (generation 31).

In all, I disassembled and analysed over sixty specimens in connection with this case. The viruses were relatively simple, used no new techniques and would have been easily identifiable on their own. The polymorphic code was more devious, but in concept rather than in execution.

The defence technical report was produced by a Mr John Boarder, who, though displaying an impressive academic record in various fields of computing, had no experience of virus code or its effects in the real world. I produced a fifteen-page supplementary report detailing technical analysis of additional complaints, and highlighting inconsistencies and inaccuracies in Mr Boarder's report.

A date of 17 November at Exeter Crown Court was set for the hearing, at which it was expected that Mr Boarder and I would be questioned. However, on that date, after I had been examined and cross-examined, the defence counsel rose and announced that although Mr Boarder was in court and had heard my evidence refuting his conclusions, he had nothing to add to his report and would not take the stand.

It then only remained for Judge Jeremy Griggs to adjourn the proceedings before returning to announce the sentences. On each of the ten charges, Pile was sentenced to six months imprisonment, to run concurrently. On the incitement charge, as it involved the SMEG polymorphic engine and indefinite proliferation of polymorphic virus production by other virus writers, the judge took a more serious view, sentencing Pile to twelve months, to run consecutive to the other sentences. So, Pile went to prison for eighteen months. Pile's solicitor later indicated an appeal might be considered: I am not aware that any appeal has since been filed.

The Implications

This landmark case has been interesting for many reasons; first, for the co-operation between the *Metropolitan Police Computer Crime Unit* and Devon and Cornwall Constabulary's Fraud Squad, which worked extremely effectively in co-ordinating enquiries both in the UK and overseas. Second, there was a direct link in the first nine charges between the defendant and the complainant: Pile was shown to have uploaded an infected file to a BBS; and the complainant was shown to have downloaded the same file, suffering virus infection as a result.

In the tenth charge, the situation was different. The complainant, *Microprose Limited* (a software publishing house) had been infected by Pathogen. The infection came from an outside source with no connection to any BBS known to have been accessed by Pile. The charge was the only one in which there was no direct link with the defendant other than the virus itself. This shows that, if someone writes a virus and someone else becomes infected by it, it is not essential that the link between the writer and the victim should be proven: presence and identification of the virus is enough.

Most significant of all was the sentence attracted by the incitement charge. This was concerned with the distribution of the polymorphic engine and its associated files. Even though Pile tried to suggest that this had beneficial uses, the judge considered that the distribution of such material into the world-wide computer communications network was certainly the most serious charge brought before him.

The Police can in future be expected to keep a much keener eye on the activities of the virus exchange BBSs, as well as distribution of certain books on virus writing techniques. There are those who think that the *Computer Misuse Act* is too weak to deal effectively with virus writers and distributors: this case has certainly strengthened it.

I cannot speak highly enough of the dedication and efficiency of the officers involved in this case, from initial collection of evidence, through search and seizure operations, to the series of interviews culminating in Pile's confession. I consider myself privileged to have worked with such men.

COMPARATIVE REVIEW

Testing the Testers

Following the publication of *VB's* latest six-monthly DOS scanner comparative review [*VB January 1996, pp.13-20*], both the *NCSA* and *Command Software* commented on the fact that *F-Prot* did not detect all samples in the In the Wild test-set. As the product is *NCSA*-certified to detect all viruses in the wild, they expressed surprise at our results. The comments led to two discoveries: three of the six samples of Sarampo were non-viral, incorrect replications, and the five samples used as Natas.4744 were in fact Natas.4774.

The latter would not be cause for concern, but for the fact that the samples are in the In the Wild test-set. Joe Wells' *WildList* states that the version of Natas which is in the wild is Natas.4744, so this version must be used.

The Sarampo issue, however, is a different class of problem. In the COM samples used, the virus has patched in its initial JMP incorrectly – it should transfer control to the start of the virus body, but in certain circumstances the JMP lands 100h bytes too far into the virus. This only happens if the carrier program is a multiple of 100h bytes in length. Thanks are due to Igor Muttik of *S&S International* for this information.

The Re-test

Given these problems, a re-test and recalculation were necessary. Eight new samples were created, and then scanned with the same installations of the products used for the main testing. The information which has changed from that presented in January is as follows:

Product	Score (#)	ItW (%)	Overall (%)
CPAV	238	82.6%	72.8%
F-Prot	282	99.2%	89.2%
McAfee	281	97.6%	91.4%
NAV	276	97.1%	87.2%
NVC	282	97.9%	97.2%
Virus Buster	199	73.5%	66.9%
VET	269	95.4%	85.4%

Conclusions

The figures displayed here do not have a great effect on the results: the top five products are unchanged, but, there are small order changes in the middle of the rankings. For accuracy, it was considered essential to print the corrections.

It is evidence of the level of cooperation between *VB* and the anti-virus community that the problems came to light and were resolved within a six-hour period in January. *VB* thanks those who helped with the matter, in particular Sarah Gordon (*Command Software*) and Richard Ford (*NCSA*).

PRODUCT REVIEW

Norton AntiVirus

Dr Keith Jackson

Version 3 of *Norton AntiVirus* has been around for some time, but has not been reviewed in *Virus Bulletin* since November 1993. In addition to this review, previous versions of *Norton AntiVirus* have been published in *VB* in March 1992 (v2) and January 1991 (v1).

Installation

This version of the product was supplied on three 1.44MB (3.5-inch) diskettes. Although *Norton AntiVirus* (*NAV*) is provided as DOS and *Windows* executable programs, the install program is itself purely a DOS program.

When execution commenced, the install program first scanned memory, the Master Boot Record, the boot record, and the files on drive C. It then requested the user's name and company, offered a choice between a full and custom installation, and requested the name of the subdirectory where the *Norton AntiVirus* files should be stored.

And then fell over. Consistently.

I tried manfully to select various options, but even if I accepted all the default choices, nothing made any difference. The installation program was a non-starter: in fact, after every installation attempt, my PC was locked up so thoroughly that nothing worked apart from the mouse, and a reset was required.

Eventually, I gave up and telephoned the developers for advice. After I had explained the problem, they recognised the symptoms, ascribed the problem to a shortage of memory (I did not agree with this explanation: 582KB of conventional memory, and 7MB of extended/expanded memory, are available), and dispatched a new set of disks, which arrived a couple of days later.

I recommenced installation, and received the error message 'Error on drive A:, disk may not be formatted'. This proved a transient error: retrying let me proceed. This, however, was the good news; the bad news was that the installation program still hung in exactly the same place.

At this point I decided to try another, older and slower PC, which made installation a turgid process. The program went off for a 40-second think before showing any signs of life, and after a burst of activity went off for another think. After two minutes, I gave up: it was locked up. The same as before.

I decided to don my Hercule Poirot costume, and try to solve the problem myself. *Norton AntiVirus* had added itself to the PATH incorrectly during installation, but altering this made no difference. I then instructed the scan carried out at the start of installation to scan only the root of drive C. This worked, and installation completed. Finally I switched the original scanning options back on, one by one, and found that scanning memory and/or MBS was not harmful, but requesting a 'Boot Record' scan caused the PC to lock up during installation. Why? I've no idea!

After these shenanigans, I eventually got the installation program to complete. Automated changes to the start-up files AUTOEXEC.BAT and CONFIG.SYS are offered, and, finally, users are encouraged to create a 'Rescue' disk.

If full installation (as opposed to Custom installation) is selected, all the required *Windows* component parts are installed, but a *Windows* group is not created. I commented on this omission in my last review of v3, but it has not changed. *C'est la vie*.

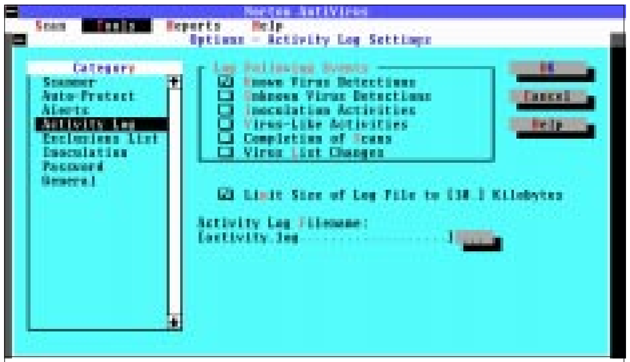
My review of v2 of *Norton AntiVirus* contained the conclusion that 'The installation program provided with *Norton AntiVirus* is excellent'. Would that it had stayed that way.

In response, *Symantec* stated that the product sent for review was a Beta of version 3.08 and that this problem was corrected in the full release.

Documentation

The two A5 manuals provided were thoroughly indexed, with a decent glossary and explanation of the error messages, but did not go into anything in great detail. Most of the manual consists of explaining what each of the various onscreen buttons do. For instance, the comments 'Repair - Allows you to repair the file', and 'Delete - Allows you to delete the file' are true and accurate, but hardly add to comprehensibility. Even though it is a bit shallow, it has to be said that the documentation is easy to follow.

Information about the viruses currently known to *Norton AntiVirus* is contained in the 'Virus List' (*Windows* version only). V1.0 knew about 115 viruses (142 variants), and v2.0



Options - Activity Log Settings

Log Following Events
[✓] Known Virus Detections
[ ] Unknown Virus Detections
[ ] Inoculation Activities
[ ] Virus-Like Activities
[ ] Completion of Scans
[ ] Virus List Changes

[✓] Limit Size of Log File to [10] Kilobytes

Activity Log Filename:
[activity.log]

Norton AntiVirus offers comprehensive configuration options, all of which are set from the user interface.

about 341 viruses (1006 variants). The original v3.0 knew about 2350 viruses, and the current v3 claims knowledge of 6401 viruses – the distinction between variant and virus seems to have been dropped.

NAV has been criticised in previous reviews for using silly or misleading nomenclature. The prime example of this is their use of 'inoculation' to refer to the process otherwise called 'checksumming'. As far as I know, other anti-virus products use 'inoculation' to mean addition of executable code to a program so the program can recognise that it has been altered when it is executed. *NAV* begs to differ, which is confusing, but the terminology is consistent within the product, so the average user will experience no problems.

Scanner Operation and Detection

The scanner part of *Norton AntiVirus* is easy to use – it always has been. When execution commences, a menu of all available drives is displayed onscreen. A multitude of other options and settings are available from drop-down menus. The product permits scanning to be carried out in any desired manner; e.g. scanning only specified file(s) or a named subdirectory.

I tested *Norton AntiVirus* against the viruses listed in the Technical Details section below. This includes large numbers of polymorphic viruses, a Standard test-set, an In the Wild test-set, and several Boot Sector viruses.

It failed to detect nine of 160 In the Wild samples: the four Goldbug, the two Lemming, and the three Maltese_Amoeba; a detection rate of 94%. It also failed to detect 12 of the 233 Standard test samples: 1260, Casper, Crazy_Lord (2), Cruncher (2), Halley, NukeHard, Phantom (2), V2P6 and WinVir_14; a detection rate of 95%. All eighteen boot sector samples were detected correctly. The last time v3 of *NAV* was reviewed, it failed to detect 1260, Casper, WinVir_14, Maltese_Amoeba, Power, and V2P6. V2 of *NAV* could not detect either Casper or 1260. This is still the case.

When it came to polymorphic viruses, *NAV* did not fare so well. It detected all of the Girafe, the One_Half and the Pathogen samples, but none of the RDA.Fighter or the Uruguay test samples. It detected 21 Groove/Coffee_Shop test samples, 100 Sat_Bug test samples, and just sixteen of the SMEG test samples. A total of 3787 of the 5000 polymorphic test samples were correctly detected, an overall detection rate of 76%.

This figure hides the fact that there are polymorphic viruses which have been around for quite a while of which *Norton AntiVirus* has no knowledge whatever. As far as I could tell, both the *Windows* and the DOS version gave identical detection results.

Scanning Speed

Comparative results of scanning speed are the only fair way to measure how fast a scanner can operate. In its default installation mode, the DOS version of *NAV* scanned the entire content of my hard disk (702 files, 25.6MB of executable files) in 2 minutes 13 seconds. If program files only were tested, 366 files were scanned, and the scan time dropped to 1 minute 33 seconds.

When these scans were repeated under *Windows*, the scan times increased to 2 minutes 34 seconds and 1 minute 47 seconds respectively. In comparison, *Dr Solomon's AVTK* scanned the same disk in 2 minutes 56 seconds, and *Sweep* from *Sophos* took 4 minutes 1 second for the same scan.

When *NAV* was instructed to scan inside compressed files, the scan times increased once again, to 2 minutes 50 seconds using the DOS version of the product, and 3 minutes 8 seconds using the *Windows* version. These increases are actually quite respectable; for instance, when *Dr Solomon's AntiVirus Toolkit* performed a scan inside compressed files, its scanning time rose to 9 minutes 1 second – a factor of more than three longer than the default DOS scan time.

The log file was always written to disk at the end of a scan. In the worst case, this took 19 minutes 20 seconds to spool itself back to the hard disk of my test PC.

"including polymorphic virus detection in the memory-resident software would greatly increase the imposed overhead"

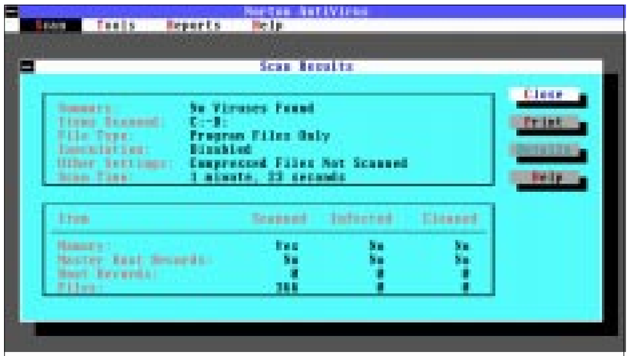
The scanner does slow down noticeably when polymorphic virus samples are scanned. The time taken to perform the first scan of the Magneto-Optical disk containing the complete virus test-set described in the Technical Details section was 16 hours 3 minutes! I started it going one night, and when I looked the next morning, it still hadn't finished.

I could not complete this scan: when the product looked inside ZIP files containing large numbers of polymorphic virus samples, it reported an 'Out of Memory' error. When scanning inside compressed files was disabled, a scan of the complete virus test set reduced to *only* 4 hours 50 minutes.

It is definitely the polymorphic virus samples that are occupying the scanner. When scanning the Magneto-Optical disk, *Norton AntiVirus* could scan the In the Wild test-set in just 51 seconds, and the Standard test-set in 55 seconds.

The times quoted above were taken using a stopwatch, as the times quoted onscreen were between 10% and 15% different. Shorter, of course! *Norton AntiVirus* is not alone in following this sharp practice: both *Dr Solomon's AntiVirus Toolkit* and *Sweep* from *Sophos* quote a scan time which is about 15% less than the actual time taken to perform the scan.

I have no doubt that the developers will claim that they are only measuring actual scanning time, not the associated setup overhead – but users are affected by total scan time.



```
Summary:           No Viruses Found
Items Scanned:     C:-D:
File Type:         Program Files Only
Inoculation:       Disabled
Other Settings:    Compressed Files Not Scanned
Scan Time:         1 minute, 33 seconds

Item                     Scanned   Infected   Cleaned
Memory:                      Yes         No        No
Master Boot Records:          No         No        No
Boot Records:                  0          0         0
Files:                       366          0         0
```

The results of a scan; displayed in the same *Windows*-style DOS interface as used by the rest of the product.

Memory-resident Software

The 'Automatic Protection' part of *Norton AntiVirus* can be set up in umpteen different ways, choosing different options from within the scanner, then rebooting the computer. *NAV* uses the term 'Automatic Protection', somewhat misleadingly, for what most other people call memory-resident software. The marketing-driven names are unnecessary.

The executable used to provide memory-resident protection was loaded as a device driver from the file CONFIG.SYS. A *Windows* program was provided (NAVTSRW), but every time I tried to execute this program, it locked up, saying it could not load SYMEVNT.386. Just like the installation program! I was unable to prevent this happening.

Even if this is a configuration problem, why does the software lock up? Because of this, all the results described below were taken using the DOS component of the memory-resident software.

The overhead introduced by this component was measured by timing how long it took to copy 40 files (1.25MB) from one subdirectory to another. Without *Norton AntiVirus*, the files could be copied in 23.4 seconds, a time which rose to 29.3 seconds when *NAV* was present and the option to scan any opened file was enabled.

This overhead of 25% is significant, but to keep things in perspective, it is a worst case measurement. If files are only scanned when they are executed, the overhead is only 1%. Almost immeasurable.

The detection rate of the memory-resident software was less than for the stand-alone scanner. Only 148 of the 160 In the Wild samples were detected (93%), and 33 of the 233 Standard test samples (14%). No polymorphic viruses could be detected by the memory-resident component. This is less than ideal, but at least the results are skewed towards the detection of viruses already known to be In the Wild.

Including polymorphic virus detection in the memory-resident software would greatly increase the imposed overhead, and most anti-virus companies omit detection of polymorphic viruses from their memory-resident software.

The Rest

Whenever Ctrl-Alt-Del was pressed to force a warm boot, and the *Norton AntiVirus* TSR was active, *Norton AntiVirus* scanned any floppy disk found in drive A before the reboot was performed. This is a nice touch, which will help prevent boot sector virus infections.

The scheduler included with the *Windows* version of *Norton AntiVirus* permits scans to be set up on any basis (once only, hourly, daily, each weekday, weekly, monthly, annually).

Conclusions

The DOS and *Windows* versions of *Norton AntiVirus* are *very* similar; a feature that has always been present in this product. Apart from the imposed *Windows* graphical style, anyone would be hard pushed to design them to be closer.

I broadly agree with the conclusion about *Norton AntiVirus* contained in *VB's* comparative scanner review last month: the product's virus detection capability is good, but its polymorphic detection rate is let down by a zero detection rate for several polymorphic viruses which have been around for quite some time now.

Given the problems I describe above with the installation program, ask yourself the question 'Was this build of *Norton AntiVirus* thoroughly tested?'. By any stretch of the imagination, the answer to this question must be no.

Norton AntiVirus compares quite well with rival packages as far as scanning speed is concerned, but it is let down by a lack of technical detail in the documentation – the same conclusion I made in my previous review of v3, over two years ago.

Technical Details

Product: *Norton AntiVirus* (no serial number visible).

Developer/Vendor: *Symantec Corporation*, 10201 Torre Avenue, Cupertino CA 95104, USA, Tel +1 541 334 6054 (US), +31 71 353 111 (Europe), fax +1 541 334 7400 (US), +31 71 353 150 (Europe), BBS +1 503 484 6699 (US), +31 71 353 169 (Europe). WWW: http://www.symantec.com/.

Availability: Not stated.

Version evaluated: v3.08, Nirvana Build 103 (I'm not making this up!).

Price (UK): for *Windows 3.1* – £89; for *Windows 95* – £149 Both versions include 12 monthly updates.

Hardware used: A *Toshiba 3100SX*; a 16 MHz 386 laptop with one 3.5-inch (1.4MB) floppy disk drive, a 40MB hard disk and 5MB of RAM, running under *MS-DOS version 5.00* and *Windows version 3.1*.

Viruses used for testing purposes:

For a complete explanation of each virus, and the nomenclature used, please refer to the list of PC viruses published regularly in *Virus Bulletin*.

For the viruses currently in the Standard, In the Wild, Polymorphic, and Boot Sector test-sets, refer to *Virus Bulletin*, December 1995 p.23.

REVIEW

1995 – Was that the Year?

The twelve months which have gone into the history books as 1995 have seen new developments on every side – systems, software, and of course viruses and virus writers. The usual round of small companies being subsumed into larger ones took place, and will doubtless continue.

Virus Bulletin saw some changes at the helm, gaining both a new editor and a new technical editor. The year also saw what amounted almost to a 'mass migration' of anti-virus researchers; from one company to another, from one country to another, and even from one continent to another.

The Mounties get their Man

One of the highest-profile events of the year was the court case of Christopher Pile, known as the Black Baron, who produced the viruses Pathogen and Queeg, and SMEG (Simulated Metamorphic Encryption Generator). Pile's court appearances reached their climax in May when he pleaded guilty to eleven counts under the *Computer Misuse Act*. The case culminated in November when Pile was sentenced to eighteen months in jail; the first successful UK prosecution for distributing, and inciting others to write, viruses.

Officers of *New Scotland Yard* were satisfied that a custodial sentence was awarded. John Samuel (Devon and Cornwall Police) stated that it should be a warning to other would-be virus authors, that they too may expect to be visited by the full force of the law.

The More Things Change...

1995 saw many alterations to company structures: *Central Point*, having as recently as two years ago acquired *Xtree*, was subsumed into the multi-national *Symantec*. *Symantec* continues its trail of acquisitions: prior to acquiring *Central Point*, they had bought out other developers such as *Fifth Generation* and *Certus*. *McAfee Associates* also gave much effort to expanding its operations, adding its former European distributors to its stables and opening a research centre in the Netherlands.

Trends and Troubles

Polymorphic viruses continued to be the main focus of attention, but the year's second half was dominated by the new Word Macro viruses. The first of these was Concept (previously called Winword.Concept, and originally referred to by *Microsoft* as the prank macro): it was shipped on a number of CD-ROMs by various companies, including *Microsoft*, allowing it to spread widely. Concept has given rise not only to other viruses in the same family, but also to Trojan horses.

Earlier in the year, *Microsoft* had already been in the embarrassing position of distributing demonstration disks infected with the Form virus. However, it was not the only company guilty of inadvertent distribution. There was the usual round of viruses distributed on PC magazine cover disks – *Ziff-Davis* managed this on two occasions. Hardware giant *DEC* also handed out infected demo disks at the *DECUS* conference in Dublin in September.

A topic on many lips throughout the year was the 'Good Times' virus, first reported by *VB* in January 1995. This was a virus purported to be attached to an email message with the subject line 'Good Times'. Although quickly discounted as a hoax, *VB* is still dealing with intermittent queries from concerned users.

There were numeric milestones last year: the count of known viruses zoomed past 6000, and mid-year approached 7000. Some counts now put the total above 8000. Growth trends continue to be linear: considering the number of viruses received by researchers every week, we must be thankful for small mercies.

New Year, New Themes

August saw the advent of the much-heralded *Windows 95*: after many delays, *Microsoft* finally released it on an unsuspecting world. The good news is that, in tests carried out by *VB* [*see VB June 1995, p.15*], it has been shown that once a boot sector virus has infected a machine running under *Windows 95*, only under unusual circumstances will it be able to replicate onto diskettes. This should severely restrict the spread of boot sector viruses, which currently account for at least 70% of viruses known to be in the wild.

1995 also saw the timely demise of *Underground Technology Review*, the magazine formerly known as *Computer Virus Developments Quarterly*. This journal was produced by Mark Ludwig's *American Eagle Publications Inc* (vendor of the infamous virus CD-ROM), but was last year relaunched as a monthly under the name *Underground Technology Review*. Ludwig cited lack of readership and the increased expenditure involved in going monthly as the reason for the shut-down.

Elsewhere, he published the *Giant Black Book of Computer Viruses*, and announced his intention to produce another book, provisionally entitled *Computer Virus Supertechnology 1996.*

What's Coming Up?

1996 will certainly continue to bring surprises in both the anti-virus and the virus worlds. The growth of polymorphic viruses has seen more and more developers turning to heuristic methods of detection: it remains to be seen where they turn next.

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 editorial@virusbtn.com
CompuServe address: 100070,1340

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

Precise Publishing Ltd, **developers of the disk authorisation product *Enforcer***, has announced that *SecureNet Inc* (formerly *Reflex Inc*) has purchased the North American rights to the product for an undisclosed sum. The product will be known as *V-Net* in the US, although its UK name remains unchanged. Details from *Precise Publishing* on Tel +44 1384 560527 or Frank Horowitz at *SecureNet* on +1 206 776 2524.

The next rounds of **anti-virus workshops presented by *Sophos Plc*** will be held on 27/28 March and 22/23 May 1996 at the training suite in Abingdon, UK. Cost for the two-day seminar is £595 + VAT. Any day (day one: Introduction to Computer Viruses; day two: Advanced Computer Viruses) can be attended at a cost of £325 + VAT. Contact Julia Line on Tel +44 1235 544028, fax +44 1235 559935, for details.

The *Third Ibero-American Seminar on Virus Protection and Computer Security Technologies* will be held in Havana, Cuba; 4–9 March 1996. Topics include legal experiences in virus control, virus control policies and strategies, and data protection. **Information from José Bidot**, President (*National Commission of Data Protection*); Tel +53 778 1987.

On 4/5 March, 15/16 April, and 13/14 May 1996, *S&S International* is presenting **Live Virus Workshops** at the *Hilton National* in Milton Keynes, Buckinghamshire, UK. The two-day courses cost £680 + VAT. Details from the company: Tel +44 1296 318700, fax +44 1296 318777.

Symantec Corporation's* anti-virus software package, *Norton AntiVirus for Windows NT, is to be made available free of charge to end-users. It can be obtained, amongst other places, from *Symantec's* BBS (+1 541 484 6669), their FTP site (ftp.symantec.com), or their WWW home page (http: //www.symantec.com/).

***Eurosec 96* will be held in Paris, France on 26/27 March 1996**. Details from Isabelle Hachin, *XP Conseil*: Tel +33 1 42 89 65 65, fax +33 1 42 89 65 66.

Reflex Magnetics has scheduled further **Live Virus Experiences** for 6/7 March, 12/13 June, and 9/10 October 1996. Information on the two-day courses is available from Rae Sutton: Tel +44 171 372 6666, fax +44 171 372 2507.

The Russian anti-virus package *AVP* (*KAMI Associates*) is now being marketed in the UK by **Cambridge-based *NEST Ltd***. The product, distributed in the US by *Central Command Inc*, is available from *NEST*: Tel +44 1223 565058, email general@nest.compulink.co.uk.

A new Special Interest Group addressing computer security has been set up by *ADS (Computer Systems)*. A bi-monthly newsletter and an IT product database are included in the subscription price (£25), and the group boasts a number of IT solution providers who offer discounts on various products. For information, email johnwalker@compulink.co.uk.

***BootShield*, anti-virus software** claimed to detect consistently 99% of all boot sector virus infections, has been launched by *McAfee*. The company, which estimates boot sector infectors to cause 70% of all infections, hails the product as a breakthrough. Further information from *McAfee*; Tel +1 408 988 3832 (in the UK, Tel +44 1344 304730).

IVPC 96, the ***NCSA's* fifth conference on virus issues**, will be held on 1/2 April 1996 in Washington DC. Further information can be obtained from the *NCSA* on conference@ncsa.com.

The *Computer Security Institute* has published *The CSI Manager's Guide to E-mail Security*, discussing **risks involved with using email**, and outlining preventative measures. For a free copy, email your mailing address to prapalus@mfi.com, or view the booklet on the *CSI* home page; http://www.gocsi.com/. ***CSI's NetSec 96***, scheduled for 3–5 June 1996, will focus on security issues, problems, and solutions in networked environments. For information, contact the *CSI* by email at csi@mfi.com, or on Tel +1 415 905 2626, fax +1 415 905 2218.