ISSN 0956-9979 NOVEMBER 1996

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **With a macro here, a macro there.** The *Word* macro virus phenomenon began with Concept, and is now moving apace. This edition of *VB* contains two macro virus analyses: NPad, which is spreading rapidly in the wild, and Outlaw, which features polymorphism. See p.8 and p.12.

• **Let the Word go forth.** *Word* and *Excel's* internal file formats have been, until recently, something in which few were interested – macro viruses, however, have changed all this. Andrew Krukow discusses the risks; see p.14.

• **A new Trend?** *Trend Micro Devices PC-cillin* has been completely revamped in recent years: our reviewer takes a look at the new DOS and *Windows* version on p.18.

CONTENTS

EDITORIAL

When the going gets tough, the tough play dirty

There have been rich pickings recently for the anti-virus industry watchers. US giants *McAfee* and *Symantec* have once again been indulging in what appears to be turning into a favourite pastime: the seemingly never-ending tussle between the two has produced some amazing battles of late, well worth describing here. It started with the publication by *Norman Hirsch and Associates* (Hirsch), a New York-based security software supplier, of a report on the macro detection and disinfection ability of a selection of major US anti-virus products – specifically, products from *McAfee*, *Symantec*, and *Trend*. The test showed clearly that *Norton AntiVirus* was top of the heap, in terms of both detection and removal.

“ out came the marketing equivalent of the big guns, primed and ready for action ”

Of course, this enraged the *McAfee* marketing machine – how dare Hirsch state that another product could possibly be *better* than the great *McAfee*? Scandalous! So out came the marketing equivalent of the big guns, primed and ready for action. Cannon shot included quotes from those in the industry regarded as reputable testers – namely, *Virus Bulletin* and *Secure Computing* – and big names (Peter Tippett, with whom *McAfee* is obviously familiar – even the spelling of his name is unique to *McAfee*…). Results are quoted from a *Secure Computing* macro virus test, which (surprise surprise!), showed clearly that *McAfee Scan* was top of the heap, in terms of both detection and removal…

In fairness to *McAfee*, there were errors in the Hirsch tests – the ‘viruses’ used were not all viruses (FormatC is a Trojan; hence the statement that it was replicated before the test is not particularly credible), and the macro viruses used for the test are a curious mixture of some which are in the wild and some which are not.

The press release from *McAfee* refuting these tests was a masterpiece. It had everything – ‘facts’, figures, quotes, references, and above all, the outraged air of righteous indignation of one who has been gravely wronged. It demanded that *Symantec* recall all *Norton AntiVirus* boxes bearing the offending claims from the worldwide channel (including from shop shelves!), that all *NAV* customers be notified in writing, and that *Symantec* take out large advertisements admitting and pointing out the alleged inaccuracies. None of these things were ever remotely likely to be done, of course, and *McAfee* knows that; but it will allow them to play the injured party again when nothing is done.

‘Not the first time *Symantec* has engaged in false, misleading, or libellous marketing’, the press release hollers – quite a claim, coming from the company that brought you the news that its virus researchers were working through the night to achieve a fix for Laroux; and which this very year launched a telephone help-line purely to deal with calls concerning, of all things, Michelangelo. Much of any extra volume of calls will have been due to *McAfee’s* carefully timed press release on the subject. Physician, heal thyself? Or perhaps just another case of pot and kettle…

At present, *McAfee* has on its web site a document consisting of a snapshot of a *Symantec* advertisement, which includes the Hirsch test results. This document is a clickable image map – the viewer can select different parts of the ad, and *McAfee* will tell them what they think is wrong with the claim or implied claim in that part of the ad. Interesting tactics, sure; but to top it off, the name of the image map is ‘slamsym’. Far be it from me to claim that *McAfee* is attempting to rubbish the competition, but the facts speak for themselves…

Symantec, to its credit, has been remarkably restrained in its response to this mud-slinging, stating simply that it stands behind its claims, and that the whole thing is merely a response to its release (and the alleged tremendous success) of *NAV 2.0*. This belief may well be based firmly in truth.

All this does nothing more than prove what we already knew. On one level, anti-virus companies are very cooperative – it is good to see techies cooperating with their competitors, and such cooperation undoubtedly benefits the public greatly. On another level, however, a remarkable amount of mud is hurtling around; unlikely claims and counter-claims, and downright untruths. The marketing arms of such companies can and will continue to play this game as long as the law permits it. Users can happily ignore them all, and would be well advised to do so.

VIRUS BULLETIN **©1996 Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England. Tel +44 1235 555139. /96/$0.00+2.50**

NEWS

Sophos Wins 3i Competition

On 8 October 1996, data security software developer *Sophos Plc* was awarded top place in the UK-wide search for the best medium-sized business in the biennial *Quest for Growth* award hosted by *3i*.

The company had already won the regional final in September, and faced competition from nearly 300 businesses around the UK; businesses as diverse as food services company *Pret à Manger*, the rapidly-growing UK sandwich retailer, and *Fender Care Ltd*, who make hull protection systems for supertankers.

The winner of each regional heat was presented with £5000 to be donated to the charity of their choice. At the final, *Sophos* also came away with a cheque for £50,000.

Company MD Peter Lammer professed himself delighted with the outcome: 'All the companies in the final were highly successful, so it is a great honour for us to have won. We have a tremendous team at *Sophos* and every member of the company deserves congratulations.' ▮

MicroWazzuSoft…

Following similar incidents in the past, in recent weeks *Microsoft* have once again been distributing viruses. In mid-October, a document on the *Microsoft* WWW site was discovered to be infected with the Wazzu virus.

The document, which describes the technical support conditions and telephone numbers for various *Microsoft* products in Switzerland, was located at the URL http://www.microsoft.com/switzerland/de/misc/hotl95d.doc. In addition, it was given away on a CD at the recent Orbit computer show in Basle, Switzerland.

Days before *Virus Bulletin* went to print, it was discovered that *Microsoft* has also shipped Wazzu on the September *Microsoft* Solution Provider CD (SPCD). The infected file is called \SIA\MKTOOLS\CASE\ED3905A.DOC.

Wazzu, as was discussed in *VB*, July 1996, p.3, will occasionally swap random words in an infected document, and can also insert the word 'Wazzu' at a random location. Researchers have known about Wazzu for about six months, but it has only recently become prevalent in the wild ▮

Takeover for Cheyenne

Computer Associates International (*CA*) is to acquire the software company *Cheyenne Software Inc* in a deal which has been valued at approximately US$1.2 million. The deal was approved unanimously by the Boards of Directors of both *CA* and *Cheyenne*.

Prevalence Table – September 1996

Virus	Type	Incidents	Reports
Concept	Macro	42	25.3%
AntiEXE.A	Boot	12	7.2%
Imposter	Macro	11	6.6%
AntiCMOS.A	Boot	10	6.0%
Form.	Boot	9	5.4%
Parity_Boot	Boot	9	5.4%
NYB	Boot	6	3.6%
Empire.Monkey.B	Boot	5	3.0%
V-Sign	Boot	5	3.0%
Junkie	Multi	4	2.4%
Quandary	Boot	4	2.4%
EXEBug.A	Boot	3	1.8%
Int40	Boot	3	1.8%
Sampo	Boot	3	1.8%
Telefonica	Multi	3	1.8%
Welcom	Boot	3	1.8%
AntiCMOS.B	Boot	2	1.2%
Bye	Boot	2	1.2%
Npad	Macro	2	1.2%
One_Half.3544	Multi	2	1.2%
TaiPan.438	File	2	1.2%
Other [1]		24	14.5%
Total		166	100%

[1] The Prevalence Table also includes one report of each of the following viruses: Barrotes, BlackAdder.1015, Bug70, Burglar.1150, Cruel, Defo, Falcon, Hare.?, Int7f-e9, IntAA, Jumper.B, Mono.1063, Natas.4744, Nuclear, Quadrangle, Ripper, Russian_Flag, Satria, Stat, StealthBoot.C, Stoned.?, Tentacle.10634, TPE.?, and Wazzu.

Chairman and CEO of *Computer Associates* (*CA*), Charles Wang, said of the plans: '*Cheyenne's* products, along with *CA's Unicenter* family of enterprise management products, will offer an unbeatable combination for solving the complex management problems that clients are facing today … In addition to a strong product offering, *Cheyenne's* employees are an integral part of the value in this acquisition.'

Wang continued: 'It is expected that *Cheyenne* will operate as a division of *CA*, and that it will continue to aggressively support its current distribution strategy.'

CA develops more than 500 integrated products that include enterprise computing and information management, application development, manufacturing and financial applications. Visit the *CA* Web site (http://www.cai.com/) or the *Cheyenne* Web site (http://www.cheyenne.com/) for information on the companies and their products ▮

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

Autumnal.3072 **CEMR:** A multi-partite, 3072-byte virus which infects files and the MBR on a hard disk. It contains the plain-text strings: 'C:MEM.EXE', '.SYS', 'Error in File' and 'Ver 4.00 (C)Copyright Autumnal Water Corp. 1991'. All infected files are marked with the word 0713h located at offset 0003h (COM) and at offset 001Ah (EXE).

```
Autumnal.3072       33FF BA80 00BB 0007 8BCF 83C1 03B8 0102 CD13 81EB 0002 4783
```

Avalgasil.666 **ER:** An encrypted, appending, 666-byte virus containing the text: '(Avalgasil)666'.

```
Avalgasil.666       DCB9 5E01 8D74 FCB0 ??0E 178D 6420 5A32 F032 D052 4444 E2F6
```

Champaigne.445 **CN:** An encrypted, appending, 445-byte virus which infects one file at a time. It contains the text: 'Champaigne Lives...$by KrACkBaBy'. All infected files are marked with byte 43h ('C') located at offset 0003h.

```
Champaigne.445      8DB6 1301 3E8B 96AC 02B9 CC00 3114 4646 E2FA C3
```

Champaigne.446 **CN:** An encrypted, appending, 446-byte minor variant infecting one file at a time. It contains the text: 'Champaigne Lives...$by KrACkBaBy'. Infected files are marked with byte 43h ('C') located at offset 0003h.

```
Champaigne.446      8DB6 1301 3E8B 96AD 02B9 CD00 3114 4646 E2FA C3
```

CrazyStub.831 **CN:** An encrypted, appending, 831-virus containing the text: '*.COM', 'COMSPEC=', 'Made by BVM' and 'This program requires Microsoft Windows.'.

```
CrazyStub.831       0100 C38B 3601 0181 C650 01B9 F002 8A04 32C2 8804 46E2 F7C3
```

Emas.1208 **CER:** A stealth, appending, 1208-byte virus which contains the plain-text string: '>> BL-93115 <<' and the encrypted text: '50th Indonesian Emas'. All infected files have their time-stamps set to 62 seconds.

```
Emas.1208           AA06 B821 35CD 211F 891E B209 8C06 B409 BA94 02B8 2125 CD21
```

Fruit.1623 **CER:** An encrypted, slightly polymorphic, appending, 1623-byte virus containing the text: '@AST94.COM.EXECHKLIST' and 'FRUIT'. All infected files have their time-stamps set to 62 seconds.

```
Fruit.1623          0E07 568A A456 078D B42B 018B FEB9 2B06 AC32 C?02 C?AA 4975
```

Gerli.593 **CN:** An encrypted, appending, 593-byte, fast, direct infector. It contains the text: '*.com' and 'Gerli Virus'. All infected files are marked with byte 9Eh located at offset 0003h.

```
Gerli.593           B92D 022E 8DB6 0F01 89F7 8A26 0A03 AC2E 3386 5403 AAE2 F7C3
```

Hera.1208 **CER:** An appending, 1208-byte virus containing the plain-text messages: 'HERA IS ALIVE!', 'PCBOARD.DAT', '*** PCBoard' and 'Naughty Girl!!'. All infected files are marked with the word 3A29h (the emoticon ':)') located at offset 0003h (COM) and at offset 001Ch (EXE).

```
Hera.1208           A4A5 A5B8 FF2C CD21 3DAD 2B75 03E9 0300 E810 00BC ECFF 5D5F
```

HLLP.4536 **EN:** A prepending, 4356-byte direct infector. The virus code is compressed with PGMPAK.

```
HLLP.4536           78F2 3167 E842 555A A7F0 BA0A 15C5 3B66 A088 8A1D 1095 4F0F
```

HLLP.4999 **EN:** A prepending, 4999-byte direct infector containing the text: 'Ludvi(r)ka L. Inc.'. It corrupts some COM files. The virus code is compressed with LZEXE.

```
HLLP.4999           D9D6 EE3E 7D05 4C0D E118 CC01 F67E 0575 F6C2 8FF1 F67F 0564
```

HLLP.6253 **EN:** A prepending, 6253-byte direct infector, infecting one file at time and containing the text: '*.EXE','*.*' and 'Renia's Indifference 1.0'. The virus code is compressed with LZEXE.

```
HLLP.6253           A088 B4FF 1E83 3E0E 0200 7403 E998 BBF2 3680 7DB0 3F7D F38B
```

Hooze.513 **ER:** An appending, 513-byte virus containing the plain-text message: '[Hooze Virus 1.0 (Tai-Pan clone)]'. On 5 August, the virus installs a new Int 1Ch service routine which periodically changes the contents of the Video DAC Color Registers (VGA+).

```
Hooze.513           B8CF 7B5E 83EE 03CD 213D CF7B 7517 B90A 000E 1F81 C6F7 01FC
```

Inside.1011 **ER:** An encrypted, appending, 1011-byte virus containing the text: 'exeNot enough memory' and 'INSIDE v2.0 created by ? Dedicate for M Sell'. All infected files are marked with the word CAFFh located at offset 0012h (the EXE checksum field).

```
Inside.1011         E803 00EB 3890 57BF 3A02 902E 80B5 1800 374F 87DB 75F5 5FC3
```

Jeru.Baby.1640 **ER:** A doubly-encrypted, appending, 1640-byte virus containing the text: '** BYE BABY ! **'.

```
Jeru.Baby.1640      05BF 2B01 B970 042E 8A25 30D4 30F4 30CC 2E88 2547 E2F1 C3EA
```

Jorgito.636 **ER:** An appending, 636-byte virus. Starting from January 1998, the virus displays the usually encrypted message: 'Jorgitø Was Here'.

```
Jorgito.636         BBD7 F993 CD21 3D83 7874 66BB 4154 438B C305 FE75 CD2F 9380
```

LosLobos.627 **CN:** An appending, 627-byte, fast, direct infector containing the plain-text strings: '*.666', '????????COM', 'bailos los lobos' and (at the end of the virus code) '\\DOS'. Infected files start with byte 0E8h (near call).

```
LosLobos.627        EB05 90B4 4CCD 218D B603 01BF 0001 FCA5 A4B0 2A8A E0CD 213C
```

NRLG.818 **CR:** An appending, encrypted, 818-byte virus containing the text: 'N.R.L.G. by Gehenna' and 'Dedicated to Liana. 20 May 96'. All infected files have their time stamps set to 60 or 62 seconds.

```
NRLG.818            B932 048D BE59 01BA 0100 802D 7CF6 1581 3511 3C80 3569 8035
```

NRLG.844 **CR:** An appending, encrypted, 844-byte virus containing the text: 'FORNICAT by [NRLG]'. All infected files have their time stamps set to 60 or 62 seconds.

```
NRLG.844            B94C 048D BE62 01BA 0100 8135 FBFB 8005 8781 0574 C580 05B8
```

Paulus.1804 **CER:** A polymorphic, appending, 1804-byte virus containing the text: 'Paulus de DOSkabouter lives here... (C) MeThOxY 1996'. All infected files have their time-stamps set to 56 seconds.

```
Paulus.1804         B9D5 008B DE?? ??27 0653 ???? 0786 CA?? ??86 CA2E 8807 4A??
```

PI.2048 **ER:** An appending, 2048-byte virus marking all infected files with a time-stamp set to 62 seconds. As part of its payload, the virus displays the character π in the right upper corner of the monitor screen.

```
PI.2048             B8FF FF8B D8CD 213D 911E 7406 9090 90E9 DB03 E9F9 04FB 9C57
```

Ren.2106 **ER:** A stealth, encrypted, appending, 2106-byte virus containing the text: '*.exe' and 'chklist????'. All infected files have their time-stamps set to 62 seconds.

```
Ren.2106            0C56 83C6 0730 1CD1 CB02 DFAC 03D0 E2F5 5E80 7406 8089 5402
```

Selectronics.1100 **ER:** An appending, 1100-byte virus containing the plain-text message: '(C) Selectronics Software'. The virus hooks interrupts Int 08h, Int 09h and Int 21h. On 5 August, the virus plays a tune and displays the message: '-=≡SELECTRONICS SOFTWARE≡=-'.

```
Selectronics.1100   3500 4B74 03E9 DD00 2E89 2663 0452 1EB8 2435 CD21 0E1F 891E
```

Tentacle.10496 **ER:** A polymorphic, 10496-byte virus containing the encrypted text: 'WARNING! Your system is contaminated with the Tentacle Virus. IMPORTANT: Don't open any GIF file!!' and 'GIF.gif.Exe.exe\TENTACLE.AAA'. The following template can be used to detect the virus in memory.

```
Tentacle.10496      80FC 9F75 04B8 434D CF3D 004B 740F 80FC 3D74 03E9 B804 6006
```

Trivial.60 **CN:** An overwriting, 60-byte virus with the text: '*.c*' and 'BROÅF' located at the end of code.

```
Trivial.60          83C2 1EB8 023D CD21 8BD8 B440 B93C 00BA 0001 CD21 B43E CD21
```

Uest.888 **CER:** An encrypted, appending, 888-byte virus containing the text: '92/10/12 U.E.S.T.C' and '*.COM'.

```
Uest.888            3004 463B F775 F95E C333 F6E8 E5FF BA00 01B9 7803 B440 CD21
```

UKTC.769 **CR:** An encrypted, appending, 769-byte virus containing the text: '*.COM' and 'SlowDeath is New Virus From UKTC'.

```
UKTC.769            07B8 5E03 50BE 2D01 8BFE BA00 04AC 3206 0004 AA3B F275 F6C3
```

Undying.703 **CR:** A stealth, encrypted, appending, 703-byte virus containing the text: '[Undying Lover v2.0c][by WârβläDÉ/DÇ '96]'. All infected files have their time-stamps set to 58 seconds.

```
Undying.703         3E8B 96AD 028D B612 0052 50B8 0533 CD21 585A B93E 01EB 1323
```

V.974 **CR:** An appending, 974-byte virus containing the text: 'D:\COMMAND.COM' and '.com'. Its payload includes corrupting disk sectors.

```
V.974               9C80 FCAA 7504 B4BB 9DCF 80FC 4B74 0B80 FCAB 7406 9D2E FF2E
```

V.1259 **CN:** An encrypted, appending, 1259-byte virus containing the text: '*.COM', 'C:\COMMAND.COM', 'C:\DOS*.COM', 'C:\CHKLIST.CPS', 'C:\DOS\CHKLIST.CPS', 'C:\SENTRY.LOG', 'C:\NAV_._NO'.

```
V.1259              8BD7 5F8A 05D0 C088 0547 39D7 75F5 BF22 01E8 FE02 8B05 BFF0
```

Wanderer.1209 **ER:** An appending, 1209-byte virus containing the text: 'Wanderer_M'.

```
Wanderer.1209       0E07 BB85 00FE C3FA 8C07 C747 FE12 00FB 0E1F C606 3204 0090
```

XED.2869 **CER:** A stealth, encrypted, appending, 2869-byte virus containing the text: 'COMMAND.COM', 'by: XED of Danao,Cebu' and '"Vous a eu!"'.

```
XED.2869            B88C D315 3375 72F9 D4FF 8AC4 81C3 FD0F 8ED8 BB06 0081 3F80
```

Yosha.440 **CMR:** A multi-partite, stealth, appending, 440-byte virus containing the text: 'ELDOB1X by Yosha/DC'. It infects files and MBR on a hard disk.

```
Yosha.440           BB00 7C8B E3BE 1304 FF0C FCAD C1E0 068E C0BE 037C B9B8 0133
```

CONFERENCE REPORT

VB ’96: Brighton Rock

Once again, the *Virus Bulletin* conference rolls by: every year, regular as clockwork, it comes around again. In about the middle of September, the world’s virus experts and corporate security managers gather together in a large hotel and drink beer. Oh, there are some presentations about current and future issues in the world of viruses as well, for those who make it past the bar.

Location

This year, the *VB* staff and helpers made the relatively short trip down to Brighton, on England’s south coast; definitely a relief after last year’s fairly long haul to Boston.

The hotel hosting the conference this year was *The Grand*, a remarkably beautiful building, which is in excellent condition, following total refurbishment subsequent to the IRA bombing of 1984.

In at the Deep End

Proceedings kicked off on Wednesday evening with a chance to meet delegates and speakers at the cocktail reception in the hotel’s Victoria Lounge. Needless to say, this provided a springboard to a long evening – most of the conference staff retreated to dinner in Brighton early on, enjoying a very nice meal at a local restaurant (some of them even claim to remember it, despite the wine…), but returned to the hotel in full force before midnight.

The next morning, after what was in some cases as much as three hours’ sleep, the first day of presentations dawned. After a brief introductory talk from *VB* editor, Ian Whalley, on the possibility of plugging anti-virus services into an operating system kernel, *IBM’s* Steve White (one of the comparatively few people who has attended every *VB* conference) presented his keynote address; a fascinating view of the future – computer security in an Internet-enabled world, and the problems that are bound to result from the ever-increasing connectivity, and hence vulnerability, of the systems we use daily.

Following the opening addresses, the conference split into streams. In the Technical Stream, Vesselin Bontchev and Paul Ducklin competed to see who could overrun his allotted 45 minute time-slot the farthest. Bontchev won by fifteen minutes – in a late bid to grab the limelight, Ducklin fell yelping off the back of the stage, but recovered without breaking anything more important than his flow.

While Bontchev and Ducklin were fighting it out, other discussions of an equally weighty nature were being presented in the Intermediate and Corporate streams: Martin Overton and Rob Stroud held forth on corporate issues; Steve Bailey discussed the feasibility (or otherwise) of Unix viruses, and Andy Harris spoke on the dissemination of viruses via email.

Lunch provided an invaluable opportunity to get the streams back in sync once again. The afternoon session once again divided: the highlight for the corporate delegates was the Management Briefing, chaired by Martin Smith, Paul Swarbrick, Nigel Hickson, and Rod Parkin.

Notable talks after lunch included ex-*VB* editor Richard Ford on the problems inherent in evaluating anti-virus products in today’s world and the exuberant Carey Nachenberg on Java and ActiveX risks.

After a welcome coffee break, Joe Wells (*IBM*) presented a paper on the PC-virus ‘hot zones’, after which Lujican Caric and Boris Debic provided a timely reminder of the implications of an Internet-enabled world: if malware is able to send data silently to remote sites, then the problems, unfortunately, are all too clear. The less ‘technically challenged’ among the delegates took advantage of Righard Zwienenberg’s discussion of heuristics.

Time for Tuxedos

As regular *VB* conference attendees will know, on the evening of the first day we ask as many people as possible to don formal evening wear for the Gala Dinner. Amply provided for by the remarkably proficient staff of *The Grand*, and entertained by roving table magicians and an escapologist, delegates and speakers were able to relax for a few hours in an atmosphere which (despite all the formal clothing) still managed to be less formal than the sessions. The evening’s festivities were extremely well attended; in fact, so much so, that to speak to someone at the next table,

A quorum of editors: Edward I (Wilding, now of *NSM*), Richard II (Ford, now *Command Software*), and Ian III (Whalley, present incumbent), continuing the line of *VB* succession.

it was only necessary to turn around! *VB* had become a victim of its own success. Nevertheless, the close quarters did encourage better communication.

Towards the end of the meal, diners were treated to the fascinating sight of conference manager Petra Duffield displaying suspicious expertise in securing the escapologist in his strait-jacket – a sight to remember…

The traditional after-dinner thank yous followed, and were given an unusual (not to say high-pitched) twist by Joe Wells (*IBM*) handing *VB* editor Ian Whalley a helium-filled balloon as he started to speak.

The Donald Duck voice wore off reasonably quickly, in time to allow congratulations to Jimmy Kuo (*McAfee*) on his new daughter, born (two weeks early) shortly after he left California to come to the conference.

This year's cabaret was somewhat different from those in previous years – Graham P. Jolley, a comic mind-reader, held the audience simultaneously amazed and in hysterics with his remarkable tricks and quickfire patter.

Post-prandial celebrations continued in the hotel bar, with delegates, exhibitors, staff, and invited guests enjoying a jazz band until the wee small hours of the morning.

The Last Day

Delegates often wonder why the second day does not start until ten o'clock; however, they usually only wonder this on the first day. After the gala dinner, the reason becomes more than obvious…

Friday opened with David Aubrey-Jones (*Reflex Magnetics*) in the Intermediate Stream giving a very well-researched talk on the effect of viruses on *Windows NT* – a remarkably high proportion of viruses, delegates learned, will not cause any noticeable ill effects.

Following Aubrey-Jones, Dave Chess (*IBM*) presented a clever talk entitled 'Things that go bump in the Net', a discussion of the problems which will occur in an increasingly connected society: he explained that the problems involving Internet agents ('smart' programs which scour the net for their human masters, either searching for information or performing tasks) map so closely onto the biological world that anthropomorphising was almost inevitable.

The morning session also saw *Virus Bulletin's* Technical Editor, Jakub Kaminski, discussing hidden partitions versus multi-partite viruses. The final session before lunch was dedicated to an exhibition by anti-virus software companies: although this exhibition was open throughout the conference, delegates appreciated the opportunity to visit stands at a time which did not conflict with presentations.

The afternoon kicked off with Sarah Gordon (*Command Software*) presenting a follow-up to her paper 'The Generic Virus Writer', which had been given at *VB '94*. The profile

The panel session at the end of the conference, after a 'red herring' start, was soon in full flow.

of some of today's virus writers is very different to that of two years ago, and the paper describes several disturbing tendencies in the area.

Keynote speaker Steve White, not to be let off his duties lightly, next made another appearance. His paper made elegant analogies between the way different types of computer viruses have come and gone over the past years with various aspects of the natural world. Linking Michelangelo with a lemming may seem like quite a logical jump, but that's nothing to the image of Buffalo Bill Gates killing off parasitic viruses…

The highlight of the final session before the speakers' panel was the paper given by Jason Khoury (*NCSA*) describing the legal implications of virus writing and distribution in various countries. This was a lively and confident presentation, and we will doubtless be seeing more of Khoury's work in the future.

In Conclusion

This year's conference saw well in excess of 200 delegates, in addition to dozens of representatives from anti-virus software companies exhibiting their wares. On both fronts, the numbers were higher than ever before, and both conference and exhibition were correspondingly successful.

VB '96 was the only *Virus Bulletin* conference not to have been marked either by a fire or a fire alarm – readers will recall the fire on the boat taking speakers and staff whale-watching last year (along with the alarm in the hotel), which continued the *VB* tradition. We hope to correct this unfortunate omission next year!

As always, thanks are due to the helpers (the so-called *VB* microphone girls) Jenny, Julia, Kim, Penny, and Müesli; to Alie Hothersall, who carried out much of the 'behind-the-scenes' organisation; and as ever to the remarkable talents of Petra Duffield, conference manager extraordinaire. Most of all, our gratitude goes to the speakers and delegates, without whom we could not hold a conference, let alone make it such a success.

VIRUS ANALYSIS 1

NPad: Escape from Indonesia

In the weeks before this edition of *VB* went to print, a macro virus called NPad began to be reported in the wild, particularly in the US and the UK. This virus, whilst not particularly new, seems suddenly to have made the jump into the wild.

NPad consists of one macro, AutoOpen, which is revealed on decryption. It consists of eight subroutines, five of which handle the trigger, and is protected using the standard *Word* method of setting the macro 'execute-only'. This feature of *Word* prevents users viewing/modifying a macro's source code, and causes *Word* to encrypt trivially the macro in question – the code is XORed by one byte.

Infection

When an infected document is loaded on an uninfected *Word* installation, the virus receives control via the infected document's AutoOpen macro, and the MAIN subroutine in this macro is executed. This first calls DisableInput() to prevent the user halting the macros by hitting Escape, and then ensures that auto macros are enabled.

The virus next copies itself into the Global Template (NORMAL.DOT). Interestingly, it will only do this if the source macro is encrypted – which under normal circumstances it will be, as once a macro is protected in this way, *Word* (for obvious reasons) does not provide a method to remove that protection. The only obvious reason for this check is to foil anti-virus researchers who may have decrypted the macro for experimentation purposes – in this case, it will not replicate.

If the source macro is not encrypted, or the copy fails, NPad calls the routine for infecting files other than NORMAL.DOT. Before copying the macro, NPad checks whether the document is a template. If not, it resaves it as one, checks that the source macro is encrypted, copies it, and saves the newly-infected file. In this way, the virus can handle copying into the Global Template and copying into document files in a single macro.

Trigger

On execution of the MAIN subroutine in AutoOpen, NPad uses GetProfileString$() to read a setting, 'NPad328', from the 'Compatibility' section in WIN.INI. If the number this setting contains is 23, the trigger routine is executed, and the counter reset to 0 and resaved to WIN.INI. Otherwise, the counter is incremented and resaved.

The trigger routine uses the status bar (the area at the bottom left of the window, beneath the horizontal scroll bar). The message 'D0EUNPAD94, v.2.21, (c) Maret 1996, Bandung, Indonesia' scrolls from the left, bounces back and forth several times, and exits left. Four of the trigger's five subroutines manage and perform string modification and display, whilst the fifth is a delay loop which slows the scrolling down so the message can be seen.

Detection and Removal

In the *MS Word* macro encryption system, the value with which data is encrypted (the key) does not change when the macro is copied from file to file. Thus, even encrypted macro viruses can be sought using a static pattern such as that at the end of this article. However, any decent anti-virus product should look through this and see the source directly.

Although NPad is fairly easy to remove manually, this is only advisable if the number of infected documents is very small: if a mistake is made, NORMAL.DOT will be reinfected, and it's back to square one.

First, ensure Fast Saves are off (tick box on the Save tab of the Tools/Options dialog). Then select Tools/Macro, highlight AutoOpen, and select Delete. Close the dialog, and exit *Word* – this resaves a clean NORMAL.DOT. Next, reopen *Word*, and using File/Open, highlight the file in question, and hold down the left shift key whilst clicking on 'Open'. Keep the shift key down until the document has loaded: this can take longer than expected. Return to Tools/Macro and delete AutoOpen; then go to File/Save As, change 'Save as type' to 'Word Document', and save the file.

Programming Style

NPad is an interesting virus: it appears to be in two clear sections; the first (replication code) very different from the second (trigger code). The former has variable and subroutine names in English, and the indentation style is cleaner than that in the second, where the names are in a foreign language, presumably Indonesian.

NPad	
Aliases:	None known.
Type:	*Microsoft Word* file infector.
Self-recognition:	None.
Hex Pattern:	`EAE7 8DE3 F7AA 82E7 88E4 EFFC EFE5 AA89 E78D E3F7 AA9C E290`
Trigger:	Displays scrolling message in status bar on every 23rd document open.
Removal:	See text.

VIRUS ANALYSIS 2

Batch Sketches

Eugene Kaspersky

Despite the effort virus writers put into their new creations (including the infamous macro viruses), they still do not neglect the simplest of DOS executables – batch files. Continuing their bid to do the unexpected, they create seemingly ordinary .BAT files which function as batch files, COM files, and device drivers at the same time. Impossible? Read on!

With each passing day I become more convinced that nothing is impossible in computing. A new .BAT virus, Highjaq, has now appeared, which runs as a BAT, COM and a device driver, and even writes its dropper to ARJ archives.

Virus Code and Texts

This non-dangerous virus places itself in .BAT files and device drivers. It has no TSR code to infect files, but stays memory-resident when executed as a device driver, hooking Int 21h for its 'Are you there?' call and Int 08h for the trigger routine.

Highjaq has two sections of code, the first of which is text data, executed when the virus runs as a .BAT file. The second is binary data, which takes control when the virus is executed as a COM file or a device driver. The text part of the code looks like this (the labels have another purpose – they serve as opcodes when the file is executed as binary code):

```
::pFqD
@ctty nul
copy/b %0.bat+%0 c:\q.com
dir \*.arj/s/b|c:\q.com/i
:qlpj
if errorlevel 1 goto qWpU
ren c:\q.com UMKQYGWK.5KA
echo  INSTALLHIGH=C:\UMKQYGWK.5KA>>c:\config.sys
:qWpU
for %%a in (%0 %0.bat) do if exist %%a set q=%%a
del c:\q.com
ctty con
@del %q%
```

BAT File Execution

When the virus is executed as a .BAT file, it copies itself to C:\Q.COM (third line) – this is the COM dropper used to infect other files. Highjaq accounts for the fact that the batch file may have been run in one of two ways, either by typing the full name of the batch file (e.g. VIR.BAT), or by simply typing the first part (e.g. VIR). To get around this problem when trying to copy itself, the virus uses the command:

```
COPY %0.BAT+%0
```

Normally, this appends the second file to the first in the copy (%0 refers to the name of the program being executed). In this case, only one file will exist: that file will be copied.

Next, the virus runs the DIR command to find all ARJ archives in the directory tree on the current disk, and passes their names to Q.COM (fourth line). If Q.COM exits with an exit code (errorlevel) of 1, the virus' TSR code is already in memory. If this is not so (i.e. the virus has not been loaded as a device driver), the virus renames its COM dropper to UMKQYGWK.5KA (seventh line) and writes the string:

```
INSTALLHIGH=C:\UMKQYGWK.5KA
```

to the end of CONFIG.SYS, and the virus adds its name to the list of system device drivers, then deletes host file and Q.COM.

When executed as a COM file (see third line), the virus takes the name of ARJ archives from the standard input, checks the archive and appends a block of data to the end of the archive. That block of data contains the virus code, labelled as a file, /WINSTART.BAT. The virus does not pack its code while saving it to the archive, but keeps it as 'stored' data. When Highjaq is run as a binary executable, its binary code takes control from the file header after several jumps:

```
0100 3A 3A     CMP  BH,[BP+SI]    ; ::
0102 70 46     JO   Jmp_a         ; pF
0104 71 44     JNO  Jmp_a         ; qD
....
0149 3A        DB   3Ah           ; :
014A        Jmp_a:
014A 71 6C     JNO  Jmp_b         ; ql
014C 70 6A     JO   Jmp_b         ; pj
....
01B7 3A        DB   3Ah           ; :
01B8        Jmp_b:
01B8 71 57     JNO  Main_Code     ; qW
01BA 70 55     JO   Main_Code     ; pU
....
0211        Main_Code:
....
```

The processor executes the text strings as real assembler instructions: whatever happens in this case, execution passes to the main body of binary code (one of the JO and JNO jumps will always be followed). That code detects whether the virus is being executed as a device driver (no arguments in command line) or a COM file (argument /i, see the batch file code) and switches control accordingly to one of two routines.

Intercepting the Interrupts

If being executed as a device driver, the virus hooks Ints 08h and 21h, and stays memory-resident. The virus does not conform to standard system device driver format (signature FFFFFFFFh is not found at the start of the file), so it cannot install itself in the memory with a device request; instead it uses the standard DOS Int 21h, AH=31h (Terminate and Stay Resident).

Hooking Int 08h causes the virus to be invoked on every clock tick, and at some later time it will reboot the system providing it is not running *Windows 3.x*. The virus uses its Int 21h hook to respond to its 'Are you there?' call (Int 21h,

AX=FEFEh). When this is made, the virus' handler returns SI=1994h. This handler also looks for Int 21, AX=4300h (Get File Attributes), and terminates the call if the filename begins '/W'. The only obvious reason for this is that the virus is trying to protect the dropper called /WINSTART.BAT) it places in ARJ archives. If it is executed as a device driver or a COM file without arguments, the virus also gets the name of the host file, and sets its attributes to Hidden and ReadOnly.

Trigger Routine

The virus checks the system COM ports, and reads data from the Scratch Register and the Line Status Register of each port. If the bits DataSetReady and ClearToSend are set in the Line Status Register, the virus writes the port number to the DOS kernel at some fixed address. I see no reason for this action. If the value read from the Scratch Register is 51h ('Q'), the virus launches the trigger routine, which clears the screen, reinitializes the COM ports, and writes the string

```
ATL0M0A<cr><lf>
```

to the relevant COM port by using Int 14h (Serial functions) calls (<cr><lf> are bytes 0Dh and 0Ah). This is a standard modem control sequence, understood by most modern modems, which instructs the modem to lower the speaker volume (L0), or to mute the speaker (M0). The final 'A' forces the modem into immediate answer mode, even if there is no incoming call. The virus then sends the following text string to the relevant COM port, again by using Int 14h functions:

```
HIGHJAQ on COMx:38400,N,8,1<cr><lf>
```

where COMx is the number of the port accessed. Finally the virus revectors Int 08h to point to a new handler, and executes the following command:

```
C:\COMMAND.COM C:\ COM1 /E:1024/P/F
```

The new Int 08h handler checks the COM port's Carrier Detect bit. If the bit is set, the virus reboots the computer.

Batch Sketches

Aliases: Highjaq, Winstart.

Type: BAT and device driver virus. Not memory-resident, but does leave TSR code. Adds worms (droppers) to the ARJ archives.

Self-recognition in Files: Uses 'worm' method; does not check files. Infects ARJ archives 2+ times. Detects its TSR code, and does not re-infect CONFIG.SYS.

Self-recognition in Memory: 'Are you there?' call; Int 21h, AX=FEFEh. TSR code returns 1994h in SI register.

Hex Pattern in Files and Memory:

```
9C3D FEFE 7507 3BC3 7503 BE94
193D 0043 7512 538B DA81 3F2F
575B 7508 9DF9 B802
```

Intercepts: Int 08h for trigger routines, Int 21h for 'Are you there?' call.

Trigger: Outputs to modem post, reboots PC.

Removal: Under clean system conditions, delete infected files. Check ARJ files for WINSTART.COM. Where necessary, delete the last line of the file CONFIG.SYS.

VIRUS ANALYSIS 3

Unsnared and (not so) Dangerous

Jakub Kaminski

When I have to analyse virus code, it is usually because a virus has been found in the wild, and hence needs detecting and cleaning, and because users want more detailed information; not because it is especially interesting or challenging.

Unsnared.814 is a perfect example of this. Its insignificance is reflected in its name: most anti-virus products call it either _814 or V.814 – the attempt to use the name 'Unsnared' was probably made in order to classify the virus somehow and move it into a relatively empty 'U' folder.

Unsnared is a normal, memory-resident, appending, EXE file infector, which infects programs on execution. Its simple payload will corrupt certain types of file.

Execution of Infected Files

When an infected file is executed, the virus first checks whether the system is already infected, or the virus is active in memory; checking the word at address 0000:02CCh. If this is set to 02CCh, the virus assumes it is memory-resident, and an image of the original program is restored in memory. This is allowed to execute in the usual way.

If Unsnared is not present in memory, it installs itself there. First, the virus sets to 02CCh the value of the word at address 0000:02CCh. This address is within the Interrupt Vector Table (stored at the bottom of segment zero). More precisely, this word contains the offset of the Int B3h service routine. At this stage, any system using this interrupt (such as *IBM* ROM BASIC interpreter or ZIPKEY) will crash.

The virus requires a little over 900 bytes, so it decreases the amount of available memory to other programs by lowering the current Top Of Memory (TOM) by 1KB. Calculations

are based on the BIOS variable holding the TOM value (value at offset 413h in segment zero). Next, Unsnared copies the 814 bytes of its own code into this newly-seized area. Before passing control to the original program, Int 21h is hooked, and redirected to the virus code residing at TOM.

Infection Mechanism

When a file is executed, the virus' Int 21h handler receives control. As a first step, this installs a new critical error handler by revectoring Int 24h. When invoked (i.e. when a critical error occurs), this handler abandons the infection, restores the original Int 24h vector, and allows control to pass through to the original Int 21h handler.

After installing the new Int 24h handler, the virus reads the file's attributes, modifies them so that only the 'archive' bit is set, and renames the file by changing the last character of its name to 'V'. Now, all modifications are performed on a renamed file. This can fool some anti-virus programs which are monitoring modifications to executable files.

After opening a file, the virus checks its time-stamp. If the minutes field is set to 13, Unsnared assumes that it is already infected, and aborts the infection routine.

If this is not the case, the virus checks for the EXE signature (it infects only files starting with 'MZ') and checks whether the value in the file length field in the EXE header matches the real length of the file (this eliminates most *Windows* executables). A final check rejects all files greater than 589823 bytes in length.

If a file passes all the above tests, Unsnared modifies the file's header so as to reflect all changes caused by infection. At this stage the virus executes the first of its payload (see discussion below), attaches itself at the end of the file, sets the time-stamp to indicate that the file is infected, renames it to its original name and restores the original file attributes. Immediately before passing control to the original Int 21h handler, the virus releases its Int 24h hook.

Payload

The payload consists of two parts, the first of which is performed whilst a file is being infected. Unsnared searches the last 72 bytes of a file for the six-byte sequence F0FD C5AA FFF0. If found, the last two bytes are replaced with the two bytes immediately following it. Unsnared seems to be targeting some programs compiled with *Microsoft C* – all EXE files matching the search criteria that I could find contained the *Microsoft* runtime library. However, my tests were inconclusive as far as version number was concerned.

The second payload is even less clear. Unsnared intercepts these other Int 21h functions: 40h (Write To File Or Device) and 07h/08h (Wait For Character Input Without Echo). The virus watches calls to Int 21h, AH=40h for these three strings: '????????g:', '???????r:' and '????????O:' ('?' denotes any character). If either is sent to be displayed, the virus stops the request and enters the next state. It waits for one of these strings: '??s[CR]', '??i[CR]', '?i[CR]' , '??nt', or checks for a call of the Character Input function. In the first case, the virus stops the text being displayed. In the second, it does not pass the call on to the default handler Int 21h, AH=07h/08h handler, but returns the character 'Y' to the caller.

Following examination of *Microsoft C* and some consultation (thanks Tony!), my guesses for the strings Unsnared is looking for are: '????????g:' = 'warning:', '???????r:' = 'error:', '??s[CR]' = 'yes[CR]', '??i[CR]' = 'oui[CR]', '?i[CR]' = 'si[CR]', and '??nt' = 'continue'. The underlying purpose of this behaviour is unclear to me, but it is obvious that the virus is targeting a specific environment, and its destructive function is correspondingly limited.

Conclusion

Unsnared is easy to detect, and infected files are easy to repair, so eradicating an infection should be fast and pose no great problems. However, the fact that some infected files can be further corrupted by the virus' modification of bytes in files compiled with *Microsoft C* leads one to the safest solution: replacing all infected files with clean originals.

Unsnared

Aliases: V.814, _814, SillyRE.814, Unsna-814.

Type: Parasitic, appending, memory-resident EXE infector.

Self-recognition in Memory:

Value 02CCh at address 0000:02CCh.

Self-recognition in Files:

Minutes field of time-stamp set to 13.

Hex Pattern in Files and Memory:

```
E857 0051 5789 1FFF 0E13 048B
1E13 04B1 06D3 E326 832E 0200
```

Intercepts: Int 21h (functions: 7, 8, 40h, 4B00h).

Trigger (Payload):

Corrupts EXE files containing the following byte sequence

```
F0FD C5AA FFF0
```

Removal: After booting from a clean DOS disk and finding an infected file, modify following entries in the EXE file-header: file length (two words at offset 2), SS, SP, IP, CS by replacing them with values taken from respective locations (offsets calculated from the end of infected file): 4 (two words), 2A5h, 29Ch, 298h, 2AFh. Trim 814 bytes from the end of the file.

VIRUS ANALYSIS 4

Outlaw: The Changing Face of Macro Viruses

Dr Richard Ford

Once again, the same old story: we predicted it for some time, now it has happened. Introducing Outlaw, the first 'polymorphic' macro virus. Note the quotes – although each of Outlaw's macros has a different name in every replication, the body of each macro remains the same, so most products should be able to deal easily with this virus.

Overview and Analysis

When given any macro virus of more than one page long, I usually first print out the contents of the macros so I can go over them on paper, scribbling as I read. While this may seem old-fashioned, I don't feel as safe as I once did using my word processor! It also lets me view the virus in its entirety. Outlaw is *big* – fourteen pages, when printed. Most of this, fortunately, is taken up with a long debug script. The 'code' itself is a more reasonable eight pages.

In operation, the virus is essentially identical to those already known. Outlaw contains macros which copy themselves to the global document template. Once there, they are available whenever *Word* loads, and can copy themselves into documents used on the system, converting the file type from document to template. On meeting certain conditions, the trigger is executed. In the big picture, it is much like any other macro virus; however, in the details, quite different.

Auto What?

Most *Word* macro viruses replicate in one of two ways. They may use one of the many 'auto' macros within *Word*, such as AutoOpen or AutoClose, which are executed automatically whenever a particular condition arises. Others replace frequently-used menu commands like 'FileSaveAs'.

However, the goal of the author of Outlaw seems to have been elementary polymorphism. This requires either that macro names change with each replication, or that the bodies of the macros themselves change. If the former technique is used, it precludes the usual replacement of *Word* menu items, so the virus writer has taken a more unusual route: the addition of hotkeys within *Word*.

The ability to assign tasks to keystrokes is a powerful feature of *Word*. There are two main ways to assign hotkeys. The most familiar is to open the Tools/Customize menu, and select keyboard; then, select by category an action to assign to a hotkey. Within this menu box is an option for assigning user-definable macros to a keystroke. Handy, for legitimate as well as illegitimate purposes.

In the case of an Outlaw-infected *Word* environment, every time the 'e' key or the spacebar is pressed within a *Word* session, the virus takes control. We shall examine these macros in turn.

As the virus has no fixed macro names, for the remainder of this article we refer to the macros by the name of the hotkey with which they are associated (thus, the 'e' macro is the macro run whenever a lowercase 'e' is entered). As the macro which carries the payload is not associated with any hotkey, we shall simply refer to it as the payload macro.

Hotkey 'e'

Outlaw's first action is to check whether there is a currently-active file. If not, the virus will skip its infection routine – at least, for this keypress – and control will pass to code which decides whether or not to call the trigger routine. The virus attempts to infect any open document.

Before infecting, the virus checks to see if the variable 'VirNameDoc' is defined for the infection target, then checks the contents of this variable against the names of the macros defined in the active template. If a match is found, the document is assumed infected and infection is aborted.

If the file is deemed suitable for infection, Outlaw first saves the document as a template, then calls a routine named (innovatively) 'Routine' – the heart of the virus' 'polymorphism'.

Routine generates a quasi-random name for the macro it is about to copy to the document by using *Word's* own random number generator (one of *Word's* many extremely useful features for the would-be virus writer) to generate a random number between 7369 and 9291.

It then converts this number to a string, and prepends a letter between A and X to the string. The letter chosen depends on the current time; the current hour is converted to a particular letter by means of a simple series of 'If' statements.

Once this string is set, the virus copies the 'e' macro to the document, and adds a hotkey definition so that macro is executed when the letter 'e' is pressed. A variable 'VirName' is set within the host document to the name of this macro. Routine then returns to the body of the macro code.

The next routine, Crypt, performs an analogous function for the 'spacebar' macro, setting a variable 'VirNameDoc' in the host document. Finally, the payload macro is copied across. Once all above macros are copied, the file is saved.

Only two more operations remain for the 'e' macro. First, the letter 'e' is inserted at the current insertion point, replacing the letter 'stolen' by the macro. Second, the virus checks whether to call its payload macro.

If the date is 20 January, the virus enters its trigger routine. After checking that the machine is not running *Windows 3.x* or is a *Macintosh*, the virus makes a final check that it is running *Word 7*. If so, the last of the three macros is executed. We will examine this macro in its own section.

'spacebar' Macro

The 'spacebar' macro is similar in form to the 'e' macro, but is designed to infect the global document template. Like the 'e' macro, it begins by checking that there is a file open. If not, execution aborts. This check completed, the virus checks its infection marker, in a manner analogous to the 'e' macro.

If the virus determines that the Global document template is uninfected, it copies its own macros to it, changing each macro's name as described above.

When each macro is copied to the template, the virus records the name of the macros by issuing a SetProfileString command. These strings are added to the 'intl' section of WIN.INI (yes, WIN.INI still does something under *Win 95* and *NT*…). Thus, WIN.INI in an infected system will have a section similar to:

```
[intl]
sCountry=United States
SCurrency=$
…
sTime=:
Name=O8896
Name2=O8659
Name3=O7379
…
```

These settings are used from within the 'e' macro when copying infected macros to host files. No attempt to execute the trigger routine is carried out by the 'spacebar' macro.

Payload Macro

As described above, the 'Payload' macro is called whenever the 'e' key is pressed within *Word* on 20 January, any year. Like the other two macros, the payload macro name changes from infection to infection. This macro uses the Shell command, allowing the virus to execute arbitrary commands.

The first two sections of this macro use the 'Declare Function' command, which makes available external functions to *WordBasic*. In the case of the virus, the function name GetWindowsDirectoryA and sndPlaySound are made available, from KERNEL32 and winmm.dll respectively.

The macro code breaks down into routines. The first, Install, creates the file laugh.scr. This is written to disk, and consists of a debug script, which creates a binary file, laugh.com, on the fixed disk. Next, a batch file, Sounds.bat, is created: this turns off screen echo, then executes debug, redirecting its input and output from laugh.scr and to nul, respectively. Finally, Sounds.bat is executed via a Shell command.

The macro then waits 25 seconds, and issues a Beep. At this point, the virus has created the file LAUGH.COM, using debug. It must now rename this file to the extension .WAV.

Again, this task is carried out by the creation and execution of another batch file, Rename.bat. Similiarly, this batch file is executed via the Shell command, and as above, the routine pauses, this time for five seconds, before issuing another beep.

The next routine, named insert, turns full screen mode on, attempts to maximise the document, and inserts in a large font the words 'You are infected with', followed by 'Outlaw'. More paragraph breaks follow, before the text 'A virus from Nightmare Joker'.

Finally, the virus uses the *Windows 95* calls previously registered to play the .WAV file, which consists of a series of laughs. (Who would have thought the day would come when I would need a soundcard in my replication machine to write a *VB* virus analysis? Hmmm… maybe it's time to put in for a full multi-media upgrade and joystick, just in case!) Once the WAV file has completed, the payload macro terminates.

Conclusions and Thoughts

Outlaw is an obvious next step for the virus writers, and thus is only a slight surprise. Fortunately, the payload is not deliberately damaging, neither attempting to drop a regular DOS virus nor causing corruption of work in progress. Furthermore, several errors and invalid assumptions in the code can cause the virus to reveal its presence before the trigger date.

The greatest danger is that it represents the next step in macro viruses. As virus authors gradually map out exactly what *WordBasic* can and cannot do, we can expect to see macro viruses become increasingly complex in nature, until they are comparable to some of the current crop of DOS viruses. Real polymorphism, as well as stealth, is eminently possible within *Microsoft Word*. Which next?

Outlaw

Alias:	None known.
Infects:	*MS Word* documents and templates.
Self-recognition in Files:	Checks for the presence of the variable 'VirusNameDoc', and attempts to match this to macros already present. Checks for a macro name in the Global Template which matches the setting for Name3 in WIN.INI in the 'intl' section.
Hex Pattern:	`6D65 3306 6464 1D69 1143 6865 636B 496E 7374 616C 6C65 6444`
Trigger:	20 January, any year.
Payload:	Plays WAV file; adds text to host document.
Removal:	Delete infected macros and convert file type to Document, if required.

FEATURE

‘In the Beginning was the Word...’

Andrew Krukov

The twentieth century has been one of innovation and new technology, seeing the popularization of the so-called ‘thinking machines’ we call computers. A side-effect of this development is the computer virus: today, approaching the end of 1996, viruses have been infiltrating machines for over ten years, replicating, crashing systems, and corrupting data.

Physics teaches us that every action has an equal and opposite reaction – so, following close on the heels of viruses, along came the anti-virus industry. Many hundreds of people the world over work for dozens of research companies and hundreds of sales/support sites which, with varying degrees of success, protect users against computer bugs.

Millions of dollars are lost to virus action, and millions are spent to recover data, or to buy anti-virus and other security soft- and hardware. Hackers write viruses, other hackers create new anti-virus programs, publishers print them, distributors distribute, end-users buy. And life goes on…

The End of the Beginning

What would happen if the world woke up one sunny morning, and viruses had disappeared overnight? Perhaps not much of significance. Users would be happy, the anti-virus industry would put its expertise into other fields, and Grandma would tell small children old myths about the nasty viruses which used to run rampant through the computers of the world.

Is it possible to kill viruses off forever? To do this, everyone would have to use operating systems that do not support viruses. Granted, viruses may be written for any popular OS; but to write viruses and spread them internationally, two things are necessary:

- a well-documented OS, which makes the writing easy
- many people exchanging executables for that OS

Only one OS meets both requirements: the very popular (and remarkably fully-documented) DOS. DOS viruses are the only ones, in the last ten years, to have created problems daily for users in every corner of the planet. *Windows* viruses were discovered in the wild only in 1996, and the Tentacle variants are the only ones to make any impact so far. No *Windows 95* or *OS/2* viruses are in the wild.

Moreover, compared with the circa ten thousand DOS viruses, the number of viruses for other operating systems is paltry: 100–200 *Mac* viruses, fewer than twenty *Windows* viruses, three *Windows 95* viruses (all are variants of Boza), and a handful for *OS/2*. So, there are over 100 times as many DOS viruses in existence as the total of all other viruses.

Therefore, it seems that to break the circle of virus writing, users must stop using DOS and turn to one of the plethora of new operating systems. Viruses will then die, as will the anti-virus industry.

Not so.

A Totally New Concept

WinWord.Concept overturned these beliefs. This infector was the first in the new breed of *Word* macro viruses; viruses for which the old rules do not apply. They are application-specific, multi-OS viruses; spreading only within *Word* documents, but under all OSs for which a version of *Word* is available.

They are at the same time simple and complex: simple, because they are written in a variant of Basic, so it is not necessary to look at long listings of assembler instructions to analyse them; complex, because locating the infected macro in the document, detecting the virus and disinfecting the document is a complex task. To make matters worse, *Word* macro viruses spread like wildfire – after all, *Word* documents are a standard method of data exchange.

So anti-virus researchers began to direct their considerable resources and intellect against the new ‘visitors’. To detect and disinfect these viruses, it is necessary to parse the *MS Word* format, then go through data structures, calculate pointers, follow these pointers, and examine a considerable amount of data – and all this simply to *find* the macros in a given document!

The binary format of a *Word* document is more complex than that of a conventional executable. A *Word* document looks like an entire filing system, with its own FATs, directories, blocks of data, etc. Researchers have spent a great deal of time, and used many different techniques, to reverse-engineer this format; to understand this most undocumented of file formats. Now many scanners can do this, and detect and remove viruses elegantly and quickly from *Word* documents.

Not so long ago we were still awaiting the next hit, which was bound to be an *Excel* macro virus; nevertheless, the appearance of Laroux in the wild shocked many anti-virus researchers. Detecting the Laroux virus presents a much more complex problem than detecting the *Word* viruses, as the *Excel* internal binary format is more complicated. The parsing procedures have to manipulate different tables of information, different sequences of pointers, and different data formats.

These new problems have initiated a new wave of anti-virus activity around the world: at present, the disinfection mechanisms are still under construction. At the moment, there is no standard method of disinfecting *Excel* spreadsheets.

The conclusion? Viruses will not die, nor will the anti-virus industry. Users will not be properly protected. *Word* and *Excel* viruses are just the current chapter in this never-ending story.

Languages

Two products from one company, one 'office', but each with a completely different macro language. The different development paths of *Word* and *Excel* are all too clear to those who have played extensively with the two applications' respective programming languages.

Any programming language built in to an application intended for manipulating documents and allowing automated document and data processing must clearly have access to the application's internal data. In the *Office* suite, there are two methods to access internal data: using functions and procedures, and using object-oriented programming.

Both languages discussed here (*WordBasic* and *Excel VisualBasic*) have the same parent, Basic, but use completely different methods for accessing the application's internal data. All internal objects in *WordBasic* are accessed by functions and simple statements. For example, this statement would modify the current style's font attributes:

```
FormatDefineStyleFont.Points = "12", .Bold = 1
```

Statements are extensions of normal Basic, and represent procedures with named arguments. In contrast, *Excel VisualBasic* uses an object-oriented method of access. All internal data is organized into an object hierarchy, and each object has its own methods and properties. The root object in this hierarchy is referred to as 'Application'. The following commands set the font attributes for the object 'myObject':

```
myObject.Font.Bold = True
myObject.Font.Size = 12
```

Whilst the statements given above for the two languages may appear remarkably similar, they actually function in a completely different way.

Another big difference between the two is the ability to use user-defined named constants in macros, a feature present only in *Excel VisualBasic*. Both languages can invoke external routines stored in a *Windows* dynamic-link library (DLL). This feature allows the programmer access to all system resources via the *Windows* API, offering huge flexibility and power, with similarly-proportioned risks.

Editing and Hiding

Excel offers an enhanced environment for source-code editing; it provides real-time syntax highlighting, and checks each line of code as it is typed for syntax errors. By contrast, *Word* only checks the syntax of a macro whilst it is being executed.

Both languages can make the source code for macros inaccessible to the user; *WordBasic* achieves this by setting the 'Execute Only' flag whilst the macros are being copied, whereas in *Excel* the same feat is accomplished by setting the sheet's 'Visible' property to 'xlVeryHidden'.

Documents, Templates, Sheets, Workbooks…

Only *Word* templates can contain *WordBasic* macros. A *WordBasic* macro is a set of functions and procedures – one of the procedures must be called MAIN, and will be executed when the macro is invoked. MAIN, like all other functions and procedures, can of course call functions and procedures from any macro in any loaded template. It is possible to create procedures and functions within a macro which are only accessible by other macros, not by the user.

Any *Excel* file can contain any number of macro sheets, each of which can contain any number of procedures/functions. Operations with macros from macro level are valid only for the macro sheet as a whole.

Macro Activation: Executing the Victim

Both *Word* and *Excel* have the unfortunate ability to run macros automatically on specified events. The first method by which this can be done is identical on both systems. By giving a macro a special name, the application can run it automatically when a user performs an operation such as opening/closing a document. *Word* and *Excel* recognize the following names as automatic macros; the now-infamous 'auto' macros:

Event	Word	Excel
Open a document	AutoOpen	Auto_Open
Close a document	AutoClose	Auto_Close
Application start	AutoExec	-
Application quit	AutoExit	-
Create a document	AutoNew	-
Activate a sheet	-	Auto_Activate
Deactivate a sheet	-	Auto_Deactivate

Another method of macro activation provided by *WordBasic* is the interception (or 'hooking') of built-in commands. By giving a macro the same name as a *Word* built-in command (for example, FileSave or ToolsMacro), *Word* will run it instead of the original command. For example, if a macro called FileOpen has been installed, it will be executed when the user selects the Open item from the File menu, or when he presses the Open button on the toolbar. Also, a programmer has the ability to determine the name of the command or macro assigned to a menu item or toolbar button – that is to say, he can modify the Open button to have a completely different purpose, including calling a custom macro.

The third method of activation is via the OnTime statement. For example, this command would run a macro called 'WakeUp' at 10:00:

```
OnTime "10:00", "WakeUp"
```

At any given time, only one macro can await execution – the scheduling is lost if *Word* is closed before the given time. In addition, the timer is not reactivated when *Word* is restarted.

Excel has a more complex and convenient system for processing events; it is possible to attach a macro to most *Excel* objects to allow event processing on that object. This information is accessible only at the macro level and is saved with the document.

The name of the event-processing macro is a property of many *Excel* objects, and macros can read and write to it. This table describes some properties and methods related to event processing:

Property/method	Applies to	Event description
OnAction	most visible objects	object is clicked
OnKey	Application	particular key/key combination pressed
OnTime	Application	specified future time
OnData	Application, Worksheet	DDE- or OLE-linked data arrives in *Excel*
OnDoubleClick	most visible objects	object double-clicked
OnSheetActivate	Application, Workbook, Worksheet...	object activates
OnSheetDeactivate	Application, Workbook, Worksheet...	object deactivates

Undocumented Documents: Going Inside…

Both applications save their documents in the OLE2 (Object Linking and Embedding) file format, a complex file system with directories and files (streams) which will not be described here. *Word* templates (remember, only a template can contain macros) are held in the OLE stream named 'WordDocument' within the file.

This stream contains all the information placed in the template by editing – including text, macros, toolbars, menus and styles. A pointer to the template area is stored at offset 118h from the beginning of the stream (not the beginning of the file!).

The template area consists of multiple variable-length records, each of which begin with signature bytes. A signature of 01h means that this record is a macro table. The macro table is further subdivided into records, each of which contains the offset of the macro from the beginning of the OLE stream.

If the OLE2 file contains an *Excel* file, things are more complicated: the OLE2 directory VBA_PROJECT contains all streams related to macros in an *Excel* document. It consists of one stream named 'dir' and at least one macro sheet stream.

The 'dir' stream contains references to object libraries, and objects called the 'small macro sheet table', the 'macro sheet table' and the 'global name table'. The 'macro sheet table' describes all the macro sheets: each record in this table contains the name of the OLE2 stream containing the macro sheet, and an offset to its name in the global name table.

The 'global name table' is a set of variable-size (10 or 12 bytes long) records. Each record describes one name which is used somewhere in the macros within the document, and each contains a pointer into an array of strings. Every name used in any macro is described somewhere in this name table.

Each macro sheet has a corresponding macro stream. The structure of this stream is:

```
header
static area
macro area
    line descriptor table
    macro body
```

The 'static area' consists of variable-size records. Each record can describe a declared variable, constant, function or procedure. References to the static area used in some statements (for example, Dim and Sub). The 'line descriptor table' contains each line of source code (with the line indent) and the offset to the compiled code for that line in the macro body, and a flag marking it as executable.

Code Representation

WordBasic uses a simple coding scheme to convert the macro source code into byte code by tokenizing. The usual form of a *WordBasic* token is a one-byte prefix code, which is followed by a variable amount of data relating to that prefix code.

The prefix represents Basic keywords such as 'If' or 'While', in addition to language constructs such as user-defined names, labels, internal function calls, and statements. Below is a list of some of these special prefixes:

Prefix	Optional data	Description
0x51	none	space
0x52	none	tab
0x64	none	new line
0x65	string	alphanumeric label
0x66	word	integer label
0x67	word	internal function name
0x68	8 bytes	double integer constant
0x69	string	name
0x6A	string	string constant
0x6B	string	comment (with ')
0x6C	word	integer constant
0x6E	byte	several spaces
0x6F	byte	several tabs
0x70	string	comment (with REM)
0x73	word	named argument of statement

Excel VisualBasic uses partially compiled code, which is intended for direct execution on a stack machine, in the same manner as Forth. This method is faster on execution, but significantly slower on editing, than the method

WordBasic uses. *Excel* compiles the macro code line-by-line while the macro is edited. Each line of source code is compiled into a set of micro commands for execution by the stack machine.

Each micro command consists of a two-byte command identifier followed by optional data aligned on a two-byte boundary. To illustrate this, the diagram at right reproduces the decompilation process of one line of *VisualBasic for Applications* (*VBA*) code.

Each micro command controls both the token representation and the stack machine. The stack machine is controlled through two pseudo-commands:

- push – put decoded token onto stack
- pop – get token from stack

Careful analysis obtains a strange result – the micro command 'pop' does not only get tokens from the *top* of stack! I have no words for the language creators…

Not all names in a macro are contained in the global name table. Micro commands can include Basic keywords and some internal Basic functions such as Format or Error.

```
0000003C: 00A3 0001 == push 1
     00A3 - opcode 'push integer'
     0001 - constant value
Stack: 1
00000040: 00A3 0002 == push 2
     00A3 - opcode 'push integer'
     0002 - constant value
Stack: 2
       1
00000044: 00AD 0006 == “sheet1” push
     00AD - opcode 'push string'
     0006 - constant length
Stack: "sheet1"
       2
       1
0000004E: 00AD 0005 == “book1” push
     00AD - string constant
     0005 - constant length
Stack: "book1"
       "sheet1"
       2
       1
00000058: 0024 0782 0001 == Workbooks(pop arg)push
     0024 - name(arguments)
     0782 - pointer into global name table
     0001 - number of arguments
Stack: Workbooks(“book1”)
       "sheet1"
       2
       1
0000005E: 0025 078C 0001 == pop.Worksheets(pop arg) push
     0025 - pop.name(arguments)
     078C - offset to name in global name table
     0001 - number of arguments
Stack: Workbooks(“book1”).Worksheets(“sheet1”)
       2
       1
00000064: 0025 0798 0002 = pop.Cells(pop 2 args) push
     0025 - pop.name(arguments)
     0798 - offset to name in global name table
     0002 - arguments count
Stack: Workbooks(“book1”).Worksheets(“sheet1”).Cells(1,  2)
0000006A: 0020 05A4 == push n
     0020 - name
     05A4 - pointer into global name table
Stack: n
       Workbooks(“book1”).Worksheets(“sheet1”).Cells(1,  2)
0000006E: 000B == pop + pop push
     000B - pop plus pop
Stack: Workbooks(“book1”).Worksheets(“sheet1”).Cells(1, 2) + n
00000070: 0027 0194 == a = pop
     0027 - name = stack; end decode
     0194 - pointer into global name table
Stack: none
Result: a = Workbooks(“book1”).Worksheets(“sheet1”).Cells(1, 2) + n
```

What's Next?

In version 5.0, *Excel* acquired *VisualBasic*, as well as the *Excel 4.0* macroing language. Both types of macro sheets are supported in *Excel 5.0* and later. *WordBasic* was changed in *Word 6*, and macros from previous versions must be converted (automatically or manually) before use.

These modifications did not affect the two products equally. *Excel* has a more convenient, professional and powerful language, the next version of which (*VBA5*) will be the standard application language in *Office 97*. An analysis of *PowerPoint* data files showed the presence of *VisualBasic* macros in those files.

Unfortunately, *Excel* and *Word* are not the only applications which make it possible to create macro viruses. *AmiPro* also has macros, and one virus has been written for that system [*Green_Stripe; see VB, March 1996, p.11*]; however, *AmiPro* documents are not widely exchanged.

Do other systems exist that will allow the easy creation and subsequent widespread replication of yet more brand new viruses and virus types?

PRODUCT REVIEW 1

PC-cillin 95

Dr Keith Jackson

PC-cillin 95 is a software package which describes itself as 'an anti-virus utility that provides a complete system of anti-virus features…'.

Scanners are provided for DOS, for *Windows 3.1* and for *Windows 95*. Each of these three operating systems has memory-resident software which can monitor program execution and attempts to detect virus activity. This review will be concentrating on the components for DOS and *Windows 3.1*.

The product was provided for review on four 1.44 MB floppy disks. Two of these were for the *Windows 95* version of *PC-cillin*, and two for both the DOS and the *Windows 3.1* versions. Very confusingly, all four of the disks were labelled 'PC-cillin' 95, although smaller print does give product details. I am not sure why the product is named in this manner. This review will refer only to *PC-cillin*; the universal '95' tag is just confusing. Anyway, it's 1996!

PC-cillin has been reviewed by *VB* twice before – July 1991 and December 1993. The previous version required a dongle before it would operate – I was pleased to see that this particular protection strategy has been consigned to the bin.

Documentation

The *PC-cillin* printed documentation comprised a small booklet – A5 size; 94 pages. Although DOS, *Windows 3.1*, and *Windows 95* versions are supplied, the documentation concerns itself almost entirely with the *Windows 95* version.

There is an appendix entitled 'PC-cillin for DOS/Windows 3.1' at the end of the manual, but even this is a misnomer. It is entirely about the *Windows 3.1* version. If you want to use just the DOS version of the scanner, you seem to be cast adrift on your own as far as the manual is concerned.

A printed piece of paper warns that the name of the installation program as stated in the manual for the DOS and *Windows 3.1* version is incorrect. An inauspicious start. To my amazement, when I looked back at my previous review of *PC-cillin* [*see VB, December 1993, p.20*], exactly the same fault was present – almost three years ago. This despite the fact that the name of the installation program is different in both cases!

One would think that the developers would learn from previous errors. I can think of no better way to criticise this error than by quoting verbatim from my previous review, which said: 'Getting the name of the installation file wrong in the manual just shouldn't happen'.

Installation

The documentation for *PC-cillin* states that the DOS and *Windows* installation requires 400 KB of RAM, and 4.0 MB of disk space.

Installation begins with a virus scan. When the scan is finished, the user must choose between 'Express' and 'Custom' installation. Next, the subdirectory in which *PC-cillin's* files are to be stored is specified.

The installation program can also create a 'Rescue Disk', which saves a copy of the boot and partition sectors of the hard disk: the diskette can be used to help with restoration of a corrupted hard disk at some future date.

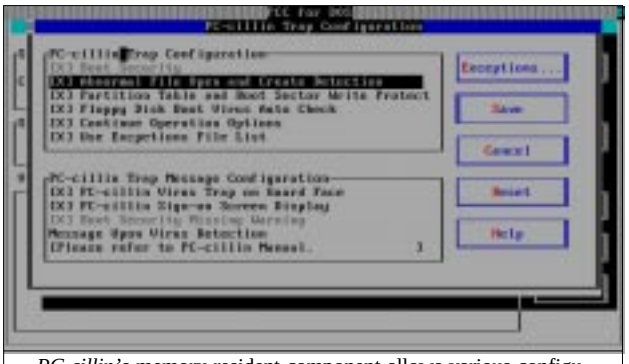
The installation program amends the file AUTOEXEC.BAT so that *PC-cillin's* memory-resident software is loaded whenever the PC is rebooted. Use of this memory-resident software is optional, and extra components must be added to enable it to function correctly under *Windows 3.1*.

The files CONFIG.SYS, WIN.INI, and SYSTEM.INI are also edited: the first is amended to include the device driver PCCDEV.SYS; the latter two to enable *PC-cillin* to operate correctly under *Windows 3.1*.

PC-cillin had created, when installation was complete, three files in the root directory of the hard disk of my test computer – MIRROR.FIL, PCCILLIN.SYS, and MIRORSAL.FIL (this last is a hidden file). Nothing explained what these files actually do. In addition to these files, *PC-cillin* installed 3.5 MB of files in its own subdirectory.

Windows Operation

When *Windows* was run after the *Windows 3.1* part of *PC-cillin* had been selected for installation, an onscreen message asked whether a *Windows* group should be created for *PC-cillin*. Nothing more was necessary, over and above the DOS installation, to complete installation of the *Windows 3.1* version.



PC-cillin's memory-resident component allows various configuration options – notably, check on copy is missing.

The *PC-cillin* program WINDISP, which allows the product's memory-resident component to communicate with the user via onscreen messages, is loaded with *Windows* – this sort of thing is necessary with the *Windows* components of most anti-virus products.

The documentation accompanying the product lays great store by the fact that *Windows* versions of the product provide large amounts of on-line help. This claim rather lost its credence when I tried to look at the *PC-cillin* help file – no matter what I tried, the response from the *Windows* help program was always: 'This file is not a Windows help file'.

I am not sure what went wrong here, but a quick check on a *Windows 95* computer revealed that the *PC-cillin* help file was also not readable by *Windows 95*. Given my gripes about the lack of documentation for non-*Windows 95* users (see above), this was doubly galling.

Scanning

When used in its default state, the DOS version of *PC-cillin* could scan the hard disk of my test PC in 1 minute 51 seconds (1729 files in total, 283 files scanned, 65.4 MB), which is reasonably fast. In comparison, *Dr Solomon's Anti-Virus Toolkit* could scan the hard disk of my test PC in 1 minute 15 seconds and *Sophos' Sweep* required 2 minutes 8 seconds for the same scan. Hidden behind these raw figures, however, *Dr Solomon's* scanner actually scans 491 files, many more than *PC-cillin*.

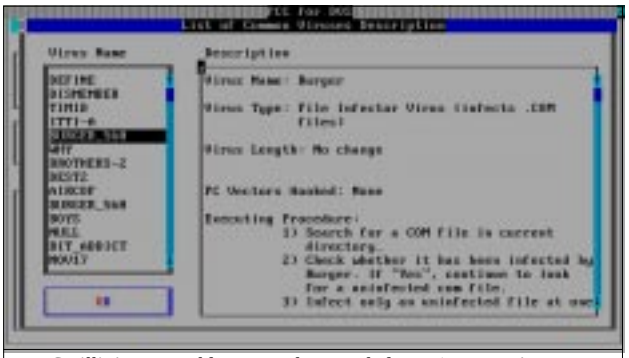
When *PC-cillin* was used to scan all files on the hard disk, the scan time increased by a factor of nearly four, to 7 minutes 1 second. This is slower than the times recorded when the scanners used for comparative purposes were asked to scan all files; for example, *Dr Solomon's Anti-Virus Toolkit* took 2 minutes 44 seconds to complete the scan.

During the tests, I noticed that whilst *PC-cillin* was scanning a disk it proved quite difficult to terminate the scan early. Although the product eventually responded to the request to terminate, it was very slow to do so; although according to *Trend*, Ctrl-C stops the scan almost immediately.

A few minor points about the *PC-cillin* scanner do stand out. First, the onscreen message 'No of files scaned' could perhaps do with being put through a 'speeling chucker'. Second, in the DOS version of *PC-cillin*, the name of a virus which has been detected is often written outside the onscreen area allocated for such texts. In short, the virus filenames are often too long and are not truncated. The end result is an unreadable mess – allowing the line to wrap would be a plus.

Detection

I tested the virus detection capability of *PC-cillin* against the test-set described in the Technical Details section below. When the product was tested against the viruses in the 'In the Wild' test-set, using its default settings, it detected



PC-cillin's reasonably comprehensive help on 'common' viruses as seen from the DOS product.

285 of the 286 test samples, missing just one sample of the Changsha.A virus. Against the viruses in the 'Standard' test-set, again using default settings, *PC-cillin* detected 263 of the 265 test samples – it missed only the two samples of Cruncher. These results are very close indeed to 100% detection, and both the DOS and *Windows 3.1* scanners gave identical detection results.

When tested against the polymorphic virus samples, *PC-cillin* detected 5353 of the 5500 test samples, a detection rate of 97%. This is very good. In fact, *PC-cillin* was 100% perfect at detecting all the polymorphic viruses apart from DSCE.Demo, MTZ.4510, Neuroquila.A, Nightfall.4559.B. Most impressive.

Boot Sector Viruses

PC-cillin claims to be able to detect boot sector viruses either by scanning a disk directly, or by relying on its memory-resident software to detect a virus when a floppy disk infected with a boot sector virus is accessed.

I tested the capabilities of the memory-resident software by using the *Windows* File Manager to inspect a directory of floppy disks infected with various boot sector viruses. When this test was tried using *Windows 3.1*, *PC-cillin* detected only fifteen out of the twenty boot sector test samples, missing Empire.Monkey.B, Form.A, Junkie, Peanut and Unashamed.

These missed viruses were re-tested under DOS by inspecting a directory of the floppy disk using the DOS command 'DIR', and every one but Unashamed was detected. Nothing specifies which virus has been found, however: the DOS memory-resident software merely states that an infected disk has been discovered.

I cannot see why this discrepancy should occur. [*I suspect perhaps a caching problem. Ed.*] In view of the fact that the DOS memory-resident software is actually present, and presumably active, when *Windows* is executing, something very curious is certainly going on here.

If the *PC-cillin* scanner is executed explicitly, then all twenty boot sector test samples are detected. This was always the case, no matter whether *Windows 3.1* or DOS

was used. The *Windows* scanner, however, has a curious habit of producing warnings about the boot sector virus infection, then stating: 'Scan completed, No file virus found'.

Strictly speaking, this is true; the viruses were boot sector viruses, not file viruses. However, I am sure that a naïve user could well infer that such a message meant that the disk was OK. Surely if *any* virus infection is found, a clear and unambiguous message should be given.

The *Windows* scanner had a further endearing habit of displaying a message saying 'Scan complete' over the top of message boxes which indicate that a boot sector virus has been detected. This could perhaps do with some alteration. *Trend* states that recent modifications fix this problem.

Note that the above results mean that when *Windows 3.1* is used, unless a specific scan of a floppy disk is performed, *PC-cillin* cannot detect the most common boot sector virus of them all – Form. This is far less than I would expect.

Memory-resident Software

PC-cillin's memory-resident software is split into two components; TSRSCAN and PCCDEV. I am not sure why this is so, and I could find nothing in the documentation to explain it. TSRSCAN uses 14.2 KB of RAM, and PCCDEV requires 6.7 KB of RAM. Both are installed whenever a PC protected by *PC-cillin* is rebooted.

With *PC-cillin's* memory-resident software installed, a small, single character 'smiling face' (their phrase) flashes unceasingly on and off in the top right-hand corner of the screen. This only happens under DOS, disappearing when *Windows* takes over. I find this irritating; fortunately, it can be disabled. *Trend* states this should not happen – presumably it is interacting with other software on the test PC.

Although several options are provided to alter the way the memory-resident software operates, no options are provided to detect viruses whilst files are merely copied. All checks are done just before a file is actually executed. *PC-cillin* was quite happy to copy any virus-infected file from one location to another without detecting any virus infection.

Virus Information

PC-cillin contains (in various ways in each of its incarnations) a section entitled 'Virus Information'. This provides a reasonable description of what a specific virus can do, though I am unsure that splitting information into specific sections entitled 'Common virus', 'Boot virus', 'File virus' and 'All viruses' is helpful or informative. Users who resort to this section are not likely to know what the individual definitions mean. The addition of a section about macro viruses here would be beneficial – not all macro infectors fall into the 'common' category.

However, all this is mere quibbling. The information provided about individual viruses is quite reasonable, and more detailed descriptions are rightly provided about 'common' viruses than about other types of virus. A definition of exactly what is meant by 'common' would not go amiss, but, given my other complaints about the documentation, this would not rate very highly on a list of problems which should be fixed.

The Rest

PC-cillin provides features which are claimed to be able to remove (disinfect) viruses from infected files. As always, I did not review these features. Use a backup. You know it makes sense.

The *Windows 3.1* version of the product includes features that permit scans to be scheduled at prearranged dates/times. This is a very good idea. One which I know makes sense.

Conclusions

This product has definitely improved by leaps and bounds since its previous reviews in *VB*. In both those *VB* reviews, *PC-cillin* came out very badly indeed, but somebody obviously listened – the current version of the product includes very good virus detection capabilities (very close indeed to 100%) and a reasonable speed of scanning. The product is also quite easy to use.

I wrote the last review of *PC-cillin*, and (if I remember correctly) my comments caused the then editor of *VB* to have a 'meaningful exchange of correspondence' with the developers of *PC-cillin*. This review of the product is much more positive – I would like to hope that this improvement is in some way due to my previous reviews.

Technical Details

Product: *PC-cillin v5.02* (no serial number visible).

Vendor: *Touchstone Software Corporation*, 2124 Main Street, Huntington Beach, CA 92648, USA. Tel +1 714 969 7746, fax +1 714 969 1555, email TechSupp@touchstone.sc.com.

Developers: *Trend Micro Europe S.r.l.*, Via Ponchielli 4, 20063 Cemusco sul Naviglio, Milano, Italy. Tel +39 292 111 847, fax +39 292 111 853, BBS +39 292 118 007.

Availability: *IBM* PC with at least 4 MB of RAM, a mouse and 4 MB of free hard disk space. The *Windows 95* version requires at least 8 MB of RAM.

Price: Single user licence US$50; 1–5 users US$225; 6–10 users US$425; 11–25 users US$950; 26–50 users US$1775; 51–75 users US$2495; 76–100 users US$3075; 101–200 users US$5700; 201–300 users US$7850. Licence includes updates; frequency at least monthly, downloadable from BBS or Internet.

Hardware used: A 33MHz 486 clone with 12 MB of RAM, one 3.5-inch (1.44 MB) floppy disk drive, one 5.25-inch (1.2 MB) floppy disk drive, 1 GB of hard disk space, running under *MS-DOS v5.0* and *Windows v3.1*.

Viruses used for testing purposes:

Where more than one variant is used, the number of examples of each virus is shown in brackets after the virus name (if the total is greater than one). For a complete explanation of each virus, and the nomenclature used, please refer to the list of PC viruses published regularly in *VB*. For a listing of the boot sector viruses see *VB*, March 1996, p.23; for the others, see January 1996, p.20.

PRODUCT REVIEW 2

ViruSafe LAN

Martyn Perry

ViruSafe LAN from *EliaShim* attempts to bridge the boundaries between *Novell* and *NT* environments: one product which detects the server type and installs appropriate components. This review examines its performance under *NetWare*.

Licensed on a per-server basis, the product allows an unlimited number of workstations to be attached to the server. Version information is inconsistent: although the console screen shows the version as 7.01, the disks are labelled 7.1.

Presentation and Installation

The product comes with three user manuals: for DOS, for *Windows*/*Windows 95*, and for networks. There are three main set-up diskettes plus a licence registration card. Version information here is also incorrect: the registration card's number [VNSE-0301 ver 1.1] did not match that on the diskettes [VLE-0300 ver 7.1]. This, according to *EliaShim*, was an error on the registration card.

The latest version allows for a single installation, irrespective of whether the workstation OS is DOS, *Windows 3.x*, *95* or *NT*. Server installation is now one procedure, whether for *Novell* or *Windows NT*. The software's default home directory under *NetWare* is SYS:VSLAN, not SYS:VS as documented. This may sound trivial, but when the NLM attempts to load, it fails, as one of the virus signature files (VSDRV1.OXN) still expects to be found in SYS:VS. To overcome this, use the /F command line option to point to the correct directory, although such action should now be unnecessary, as *EliaShim* claims to have fixed this.

Loading ViruSafe

The NLM program is loaded from the server console prompt using the standard *NetWare* Load command. Several command line options are available, including: load with a password, locate supporting files if not in default directory, launch InterServer support (more of this later) and task scanner priority.

The main NLM is loaded along with the *NetWare* C Library and *NetWare* Streams NLMs. The *ViruSafe* NLM console can be used to start/stop a scan, but configuring scan jobs is done from the management software run from a workstation.

Administration

Scanner administration is performed only from a DOS or *Windows* workstation. When running the client job scheduler (VSNLM.EXE) on a DOS workstation, F6 is supposed to deactivate the selected job. The onscreen Help file calls this Disable. Either way, it fails to stop the scanner once started, though it does remove the 'submit job' symbol. The *ViruSafe* NLM must be active on each server that needs configuring, otherwise communication errors result.

The administration is concerned with creating and managing jobs, each of which is given a name. Every job has the volume(s) to be scanned, the starting directory and the file lists to include in or exclude from the scan.

The actions for each job can be defined, and include: Delete, Check or Mark Integrity (where integrity is monitored using checksums) and Scan Programs. A scan is stopped by 'un-submitting' a job or by pressing the Esc button on the console. *ViruSafe* has two modes of scanner operation: Scheduled and On-access (Real-time) using InterServer.

On-access scanning allows checks to be performed when a file is copied to or from the server, or accessed by the workstation. This latter option, InterServer, can be disabled, and works by registering (checksumming) files. If a new file is added, or an extant file changes, it is sent to the scanner for checking before further operations are processed. Scheduled scanning provides checks on a timed basis, defined as:

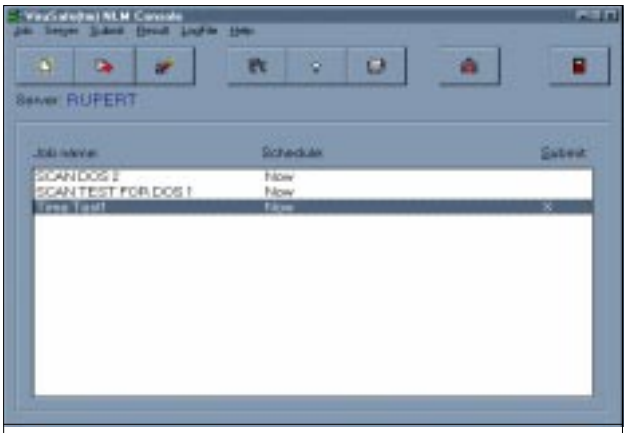
- Now: effectively an immediate scan. It has a sixty-second delay, which gives supervisors a chance to cancel (un-submit) a job if they have made an error.
- Once, Hourly, Daily, Weekly, Monthly.

The Time, Date, Day settings are activated/deactivated in context with the selection made. No facility exists to start another NLM (e.g. backup) after a scheduled scan is complete.

Configuration Options and Alert Management

For each mode of operation, various selections can be made:

- File extensions to be included. Defaults are COM, EXE, OV?, and SYS. Extra file extensions can be added.
- Files and directories to be excluded from the scan.



ViruSafe's Windows 95 configuration program is fairly basic, but is easy to use and does the job well.

- Actions to be taken on detection of a virus (separate menu option). These are: scan only, disinfect infected files, move infected files to user-defined quarantine directory, and erase infected files.

Infections found in compressed files will only be reported. They will need to be decompressed before any further action can be taken.

ViruSafe provides alert management using a predefined list of users who are notified if a virus is detected. In the test copy, the user list had the unfortunate error, each time the list was opened for editing, of continuing to multiply existing user names until the sixteen-user limit was reached. Additional alert handling makes use of *AlertTrack* (a third-party NLM which must be purchased separately) to detect any message written to the log files and to send an alert to the defined recipients. *AlertTrack* provides communication with numeric pagers, email, fax and network broadcasts.

Workstation Support; Reports and Activity Logs

The workstation component of *ViruSafe* (VS) can be run with various command options. In the evaluation copy, when the command VS /IS (necessary to enable InterServer) was executed, the message 'VS.NLM not found' appeared; however, VSNLM.EXE was able to communicate without difficulty. If VS /IS was run after VS /CF (configure), the error message did not appear.

ViruSafe keeps a record of activities in an Event log for each job. The log has the file name type VS00000x.log, where x is the sequence number. This file is (strangely) stored in the root directory of the starting path defined by the job parameters.

Updates and Detection

Updating the product entails replacing the three signature files VSDRV1.OXN, VSDRV2.OXN, and VSDRV3.OXN in the SYS:VSLAN directory. VS.NLM does not need to be unloaded for the new signatures to take effect; it checks for new signatures every fifteen minutes. This level of automation does not extend to the workstation, which must be rebooted to reload the memory-resident scanner with the new signatures once they have been installed.

The *ViruSafe* NLM console screen displays the status of the various components and allows basic operations to be performed.

The scanner was tested against the usual test-sets: In the Wild, Standard and Polymorphic (see summary for details). Viruses which went undetected were identified by allowing the scanner to delete files and then listing the files remaining on the server.

The tests were conducted using, first, default file extensions, then repeated with the file extension DO? added to the extension list. The first pass through the In the Wild set missed the macro viruses; however, these were detected when the DO? option was added, giving a good result of 98.4%. The results against the Standard (58.5%) and the Polymorphic sets (55.5%) suggest that more work is needed to keep the engine up to date.

Real-time Scanning Overhead

To determine the impact of the scanner on the server when it is running, 63 files of 4,641,722 bytes (EXE files from SYS:PUBLIC) were copied from one server directory to another using *Novell's* NCOPY, and the process timed. The directories used for the source and target were excluded from the virus scan, to avoid the risk of a file being scanned while waiting to be copied. Because of the different processes which occur within the server, the time tests were run ten times for each setting and an average taken. The tests were:

- NLM not loaded. This establishes the baseline time.
- NLM unloaded. This is run after the other tests to check how well the server is returned to its former state.
- NLM loaded, using the default setting of priority=50; InterServer off; no scan. This tests the impact of the scanner loaded in its quiescent state and without the load of the InterServer check.
- NLM loaded, priority=50, InterServer off; with manual scan. This reveals the impact of running the scanner on the server files.
- NLM loaded, priority=50, InterServer on; with manual scan. This shows the full impact of the scanner and InterServer running together.
- NLM loaded, priority=25, InterServer on; with manual scan. This shows any effect of reducing scan priority.
- NLM loaded, priority=1, InterServer on; with manual scan. This also shows any effect of reducing scan priority as low as possible.

The initial impact of loading the software is minimal, but (as expected) jumps considerably when the scan is run. The additional impact of having InterServer active with existing files is very small. Changing the priority of the scanner task across its allowable range had almost no effect on overall time taken over file copying.

Conclusion

The product is simple to install, and its ability to autodetect the underlying network type and perform the appropriate installation is convenient. However, the documentation needs to be updated to match the current version. A single sheet including

the paragraph 'The chapters in the book which discuss unique procedures for different servers or clients are irrelevant now' cannot be regarded as sufficient support for the product.

Several times in the article, attention has been drawn to inconsistencies between the operation and the documentation. *EliaShim* states that, since the product was sent for review, the documentation has been updated: as such inconsistencies give the feeling that the product was in late stages of beta testing, it is to be hoped that this is the case.

ViruSafe LAN

Detection Results

Test-set[1]	Viruses Detected	Score
In the Wild	339/342	99.1%
Standard	299/511	58.5%
Polymorphic	5553/10000	55.5%

Overhead of On-access Scanning:

Tests show time taken to copy 63 EXE files (4.6MB). Each is performed ten times, and an average is taken.

	Time	Overhead
NLM not loaded	10.7	-
NLM unloaded	10.7	-
NLM Loaded; InterServer off		
Priority = 50; no Manual scan	10.8	0.6%
Priority = 50; Manual scan	32.2	201.3%
NLM Loaded; InterServer on		
Priority = 50; Manual scan	32.4	202.9%
Priority = 25; Manual scan	32.4	202.9%
Priority = 1; Manual scan	32.1	199.7%

Technical Details

Product: *ViruSafe LAN v7.1*.

Developer/Vendor: *EliaShim Ltd*, PO Box 25333, Haifa, 31250 Israel. Tel +972 4 872 8899, fax +972 4 872 9966.

Price: Per server, including unlimited workstation connections: (UK)£599, including telephone support, quarterly updates on diskettes or via Web/FTP site for first year. Additional servers; £99 each. Discounts are available for educational institutions and charitable organizations.

Hardware Used:
Server: *Compaq Prolinea 590* with 16MB RAM and 2GB of hard disk, running *NetWare 3.12*.
Workstation: *Compaq 486/66* with 16MB RAM and 250MB of hard disk, running *Windows 95*.

[1]**Test-sets:** In the Wild, Polymorphic – see *VB*, October 1996, p.17. Standard – 511 samples, comprising: Abbas.5660 (5), AIDS (1), AIDS-II (1), Alabama (1), Alexe.1287 (2), Algerian.1400 (3), Amazon.500 (2), Ambulance (1), Amoeba (2), Anarchy.6503 (5), Andreew.932 (3), Angels.1571 (3), Annihilator.673 (2), Another World.707 (3), Anston.1960 (5), Anthrax (1), Anti-Pascal (5), AntiGus.1570 (3), Argyle (1), Armagedon.1079.A (1), Assassin.4834 (3), Attention.A (1), Auspar.990 (3), Baba.356 (2), Backfont.905 (1), Barrotes.840 (3), Bebe.1004 (1), Big_Bang.346 (1), Billy.836 (3), Black_Monday.1055 (2), BlackAdder.1015 (6), Blood (1), Blue_Nine.925.A (3), Burger (3), Burger.405.A (1), Butterfly.302.A (1), BW.Mayberry.499 (3), BW.Mayberry.604 (6), Cantando.857 (3), Cascade.1701.Jo-Jo.A (1), Casper (1), Catherine.1365 (3), CeCe.1998 (6), CLI&HLT.1345 (6), Cliff.1313 (3), Coffeeshop (2), Continua.502.B (3), Cosenza.3205 (2), Coyote.1103 (3), Crazy_Frog.1477 (3), Crazy_Lord.437 (2), Cruncher (2), Cybercide.2299 (3), Danish_Tiny.163.A (1), Danish_Tiny.333.A (1), Dark_Avenger.1449 (2), Dark_Avenger.2100.A (2), Dark_Revenge.1024 (3), Datacrime (2), Datacrime_II (2), DBF.1046 (2), Dei.1780 (4), Despair.633 (3), Destructor.A (1), Diamond.1024.B (1), Dir.691 (1), DOSHunter.483 (1), DotEater.A (1), Ear.405 (3), Eddie-2.651.A (3), Eight_Tunes.1971.A (1), Enola Gay.1883 (4), F-You.417.A (1), Fax_Free.1536.Topo.A (1), Fellowship (1), Feltan.565 (3), Fisher.1100 (1), Flash.688.A (1), Four Seasons.1534 (3), Frodo.3584.A (2), Fumble.867.A (1), Genesis.226 (1), Green.1036 (6), Greetings.297 (2), Greets.3000 (3), Halloechen.2011.A (3), Hamme.1203 (6), Happy_New_Year.1600.A (1), HDZZ.566 (3), Helga.666 (2), HLLC.Even_Beeper.A (1), HLLC.Halley (1), HLLP.5000 (5), HLLP.7000 (5), Horsa.1185 (3), Hymn.1865.A (2), Hymn.1962.A (2), Hymn.2144 (2), Hypervisor.3128 (5), Ibqqz.562 (3), Icelandic.848.A (1), Immortal.2185 (2), Internal.1381 (1), Invisible.2926 (2), Itavir.3443 (1), Jerusalem.1607 (3), Jerusalem.1808.CT.A (4), Jerusalem.Fu_Manchu.B (2), Jerusalem.PcVrsDs (4), John.1962 (3), Joker (1), July_13th.1201 (1), June_16th.879 (1), Kamikaze (1), Kela.b.2018 (3), Kemerovo.257.A (1), Keypress.1280 (6), Kranz.255 (3), Kukac.488 (1), Leapfrog.A (1), Leda.82 (3), Lehigh.555.A (1), Liberty.2857.A (5), Liberty.2857.D (2), Loren.1387 (2), LoveChild.488 (1), Lutil.591 (3), Maresme.1062 (3), Metabolis.1173 (3), Mickie.1100 (3), Necropolis.1963.A (1), Nina.A (1), November_17th.768.A (2), NRLG.1038 (3), NutCracker.3500.D (5), Omud.512 (1), On_64 (1), Oropax.A (1), Parity.A (1), Peanut (1), Perfume.765.A (1), Phantom1 (2), Phoenix.800 (1), Pitch.593 (1), Piter.A (2), Pixel.847.Hello (2), Pizelun (4), Plague.2647 (2), Poison.2436 (1), Pojer.4028 (2), Positron (2), Power_Pump.1 (1), Prudents.1205.A (1), PS-MPC.227 (3), PS-MPC.545 (6), Quark.A (1), Red_Diavolyata.830.A (1), Revenge.1127 (1), Riihi.132 (1), Rmc.1551 (4), Rogue.1208 (6), Saturday_14th.669.A (1), Screaming_Fist.927 (4), Screen+1.948.A (1), Semtex.1000.B (1), Senorita.885 (3), Shake.476.A (1), ShineAway.620 (3), SI.A (1), SillyC.226 (3), SillyCR.303 (3), SillyCR.710 (3), Sofia.432 (3), Spanz.639 (2), Stardot.789.A (6), Stardot.789.D (2), Starship (2), Subliminal (1), Suomi.1008.A (1), Suriv_1.April_1st.A (1), Suriv_2.B (1), Surprise.1318 (1), SVC.1689.A (2), Svin.252 (3), Svir.512 (1), Sylvia.1332.A (1), SysLock.3551.H (2), TenBytes.1451.A (1), Terror.1085 (1), Thanksgiving.1253 (1), The_Rat (1), Tiny.133 (1), Tiny.134 (1), Tiny.138 (1), Tiny.143 (1), Tiny.154 (1), Tiny.156 (1), Tiny.159 (1), Tiny.160 (1), Tiny.167 (1), Tiny.188 (1), Tiny.198 (1), Todor.1993 (2), Traceback.3066.A (2), TUQ.453 (2), Untimely.666 (3), V2P6 (1), V2Px.1260 (1), Vacsina.1212 (1), Vacsina.1269 (1), Vacsina.1753 (1), Vacsina.1760 (1), Vacsina.1805 (1), Vacsina.2568 (1), Vacsina.634 (1), Vacsina.700 (2), Vbasic.5120.A (1), Vcomm.637.A (2), VCS1077.M (1), VFSI (1), Victor (1), Vienna.583.A (1), Vienna.623.A (1), Vienna.648.Lisbon.A (1), Vienna.Bua (3), Vienna.Monxla.A (1), Vienna.W-13.507.B (1), Vienna.W-13.534.A (1), Vienna.W-13.600 (3), Virogen.Pinworm (6), Virus-101 (1), Virus-90 (1), Voronezh.1600.A (2), Voronezh.600.A (1), VP (1), Warchild.886 (3), Warrior.124 (1), Whale (1), Willow.1870 (1), WinVir (1), WW.217.A (1), XQG.133 (3), Yankee_Doodle.1049 (1), Yankee_Doodle.2756 (1), Yankee_Doodle.2901 (1), Yankee_Doodle.2932 (1), Yankee_Doodle.2981 (1), Yankee_Doodle.2997 (1), Zero_Bug.1536.A (1), Zherkov.1023.A (1).

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, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire, OX14 3YP, 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
World Wide Web: http://www.virusbtn.com/

US subscriptions only:

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

Tel +1 203 431 8720, fax +1 203 431 8165

END NOTES AND NEWS

Dr Solomon's Software (formerly *S&S International*) is presenting **Live Virus Workshops** at the *Hilton National* in Milton Keynes, Bucks, UK on 4/5 November and 2/3 December 1996. Details from the company: Tel +44 1296 318700, fax +44 1296 318777.

***Sophos Plc's* next rounds of anti-virus workshops** will be on 20/21 November 1996, and on 29/30 January and 19/20 March 1997 at the training suite in Abingdon, UK. Additionally, the company's training team is hosting a Practical NetWare Security course; on 26 November 1996, and 21 January and 13 March 1997 (cost £325 + VAT). Information is available from Julia Line, Tel +44 1235 544028, fax +44 1235 559935, or access the company's World Wide Web page (http://www.sophos.com/).

The *Computer Security Institute* (*CSI*) *23rd Annual Computer Security Conference* is to be held from 11–13 November in Chicago, Illinois, USA. The event will feature over 120 sessions, including presentations on **Internet security, access, email**, etc. It will also include an exhibition of computer security products – free passes for the exhibit available from the *CSI*. For details on attending the conference, contact Patrice Rapalus on Tel +1 415 905 2310; email prapalus@mfi.com.

EVE for X.400, **an email virus scanner for X.400 systems**, has been launched by UK-based *NetConnect*. The first release supports *X.400 MTA* and many popular LAN-based mail systems, and support for *Windows NT* and *OS/2 Warp* is said to be imminent. Contact the company on Tel +44 1223 423523.

Reflex Magnetics Ltd has sent out a press release announcing the launch of *Disknet v4.2*, which is said to contain such new features as **a module to trap macro viruses, and an upgrade to the Data Encryptor** enabling users to create a 'virtual' hard disk of up to 1.8GB. The software will be available for *Windows 3.x* and *NT*, for *OS/2 2.x* and *Warp*, *NetWare*, *MS LAN Manager*, and any other OS using the TCP/IP protocol. At the time this edition of *VB* went to proof, the package was not available for review. For further details, contact Phillip Benge at *Reflex*; Tel +44 171 372 6666, or visit their WWW site: http://www.reflex-magnetics.co.uk/.

Symantec Corporation has announced the launch of a new 'check-up' package, meant to identify and fix small problems on PCs. Aimed at the more inexperienced user, the software 'tunes up' the hard drive and scans for viruses. ***Healthy PC* combines in one utility features from *Norton AntiVirus* and *Speed Disk***, and its RRP is £29. Information on this and other *Symantec* products is available on the World Wide Web; http://www.symantec.com/.

In a deal with *Netscape Communications Corporation*, *Trend Micro Incorporated* has signed a licensing agreement whereby its virus detection software will be bundled with *Netscape Proxy Server*. The agreement will also allow *Netscape* to **incorporate *Trend* technology into other *Netscape* server products**. Contact Ken Millard of *Trend Micro Europe* for more information; Tel +39 292 111 847, fax +39 292 111 853, or visit the WWW site, http://www.trendmicro.it/.

Seagate Software has announced the launch of its anti-virus software, *Seagate Virus Control*. **The package, based around behaviour blocking technology, is developed in conjunction with *Norman Data Defense Systems***, and is available as an add-on to the *Seagate DMS 2.0*. For further details, contact Paul Segrave or Caroline Hammond of *Seagate*; Tel +44 1628 771299. Alternatively, email Emmanuel Vitrac at emmanuel_c_vitrac@notes.seagate.com.

The proceedings of the sixth *VB* conference are now available; price £50 + p&p. To order, contact conference coordinator Alie Hothersall; Tel +44 1235 544034, email alie@virusbtn.com.