ISSN 0956-9979 DECEMBER 1996

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

Editor: **Ian Whalley**

Assistant Editor: **Megan Skinner**

Technical Editor: **Jakub Kaminski**

Consulting Editors:

Richard Ford, Command Software, USA
Edward Wilding, Network Security, UK

IN THIS ISSUE:

• **A Wordy issue.** The number of *Word* macro viruses in the wild continues to rise: this edition of *VB* sees analyses of another two; Bandung (p.9) and MDMA (p.11).

• **How much did you say?** The direct financial costs incurred in recovering from a computer virus are not inconsiderable, and when the indirect costs are added, the total can be much higher still. Shane Coursen discusses why having clear guidelines in place can save money and effort; see p.13.

• **Avast! Perfect scores ahoy!** *Alwil Software's* package for *Windows NT* and *95* detected every single virus in all the *Virus Bulletin* test-sets: read more on p.21.

CONTENTS

EDITORIAL

The Unbearable Lightness of Testing

Next month sees the publication of the biannual *VB* DOS Scanner Comparative. Despite the rise of *Windows 95* and *NT*, this is still very much our flagship review; the one which garners most attention. It has figures quoted from it left, right and centre, and, inevitably, attracts the most criticism.

The testing of anti-virus products is, as has been well documented in these and other pages in previous years, an incredibly difficult thing to do – or at least, to do well. *VB* is fortunate in being one of the few well-regarded organisations to perform comparative testing: it is not least for this reason that I take such care when carrying out these reviews.

“ it is unreasonable to expect people to have to run multiple programs to detect different types of virus ”

As I write, I have reached the stage where preliminary sets of results are sent to developers in each company – this system has proven worthwhile in the past as far as catching small errors at an early stage goes, and offers time for the results to be discussed and suggestions to be made. Having valuable ideas put forward after publication is not immediately useful, after all…

Say what you like about the people who make virus scanners, each company certainly cares greatly how its product is reviewed. This initial, very restricted, distribution of results inevitably leads to a manifold increase in the level of email coming into my computers, and a flurry of extra checking of virus samples inevitably ensues as developers compare the results of their internal tests with ours.

During this period, there can be a certain antagonism between myself and the developers; testing methodology is often a sticking point. This time, the discussion has focused on the treatment of macro viruses. When it comes to adding new features to a scanner (e.g. the ability to scan with OLE2 *Word* documents), the DOS environment offers more problems than most, the most significant of which is the 640K memory limit.

With scanners already groaning under the load of the ever-increasing number of viruses, adding complex new scan capabilities threatens to be the last straw for some. The clear stopgap solution is to provide a second executable. Simply place the macro-scanning functionality in a program on its own, and the problem is solved – indeed, several products in this January’s tests do this. However, this system has drawbacks. First, re-educating a product’s users to execute *two* programs instead of the one they had to run previously will take time – it is inconvenient for these users to have to adjust their behaviour in this way. And is a one-stop solution too much to ask?

My problem with this particular solution is on a simpler level. Should the detection rate of the product’s macro add-in be included in the product’s score? That is to say, should the macro detector be run over the macro samples and that score added to that of the main program on the more traditional parasitic and boot sector viruses? My conclusion at this stage is that it should not. It is unreasonable to expect people to have to run multiple programs to detect different types of virus – it smacks of the thin end of the wedge. To take the situation to extremes, imagine a product consisting of 9500 separate executables, one for every virus…

Needless to say, the manufacturers of products with add-ins do not agree – to some extent, they are right. It would indeed be unfair to imply that they were incapable of handling macro viruses, so the presence and functionality will be discussed in the article, but they will not be used to form part of the headline detection figures. These will remain the sole domain of the main scanner.

There must be an interesting dilemma in the minds of these companies: on one level they cannot wait for the demise of DOS and its puny memory limits and annoyingly restrictive design. The eventual end to DOS’ incredibly long-drawn-out death throes will bring all that to a close, and relegate problems of this nature to distant memory (at least until the limit of the next OS is reached…). On the other hand, DOS was where it all began for anti-virus companies; it will be a shame to see it fade away. Nonetheless, the products will live on, converted to the new generation of operating systems. Plus ça change, plus c’est la même chose?

NEWS

Floating Dr Solomon

Following the management buyout of S&S International [*see VB February 1996, p.3*], the company (which is now called Dr Solomon's Group PLC) has announced that it intends to go public.

As usual with such things, the details are extremely complex. The offering, expected to take place in the week commencing 25 November 1996, divides the company into approximately 18.4 million American Depositary Shares (ADSs), 5.7 million of which will be made available on NASDAQ and EASDAQ at an estimated price of between US$15 and US$17 apiece.

This pricing values the company at approximately US$300 million, and the company will receive around US$64 million from the sale. This money will be used to repay borrowings to fund the management buyout, expansion of North American operations, and increased R&D investment.

The remainder of the ADSs (those not sold to the public) are to be split between the management, employees, and those companies that financed the management buyout ∎

Addendum

Following publication of the analysis of the Unsnared virus [*see VB, November 1996, p.10*], *Datafellows'* Peter Szor has contacted *VB* with information on the purpose of the six-byte sequence for which the virus checks whilst infecting.

It appears that the byte sequence F0FD C5AA FFF0 is an ID string used by old versions of *McAfee Scan*. The bytes which follow this sequence comprise a 32-bit CRC of the executable, attached when *Scan* is run with the /AV (Add Validation) code.

When *Scan* is then run with the /CV (Check Validation code) switch, it looks for the byte sequence, and compares the 32-bit CRC immediately following it with the current CRC of the program, to determine whether it has changed. Corrupting the six-byte sequence prevents *Scan* locating the CRC; thus it cannot tell that it has changed.

In addition, Costin Raiu of *GeCAD s.r.l.* (Romania) suggests that the virus was written in Romania. He states that Unsnared's second payload is directed at an old version of *McAfee Scan*, which asks the user whether to continue scanning when it detects that it has been modified. The virus pushes 'Y' into the keyboard buffer to answer this question.

It follows, therefore, that when the virus awaits, and responds to, certain printed strings, it is expecting to receive messages from some part of an anti-virus product telling the user that the file's CRC has changed. It responds by forcing the answer 'yes' to the question ∎

Prevalence Table – October 1996

Virus	Type	Incidents	Reports
Form.A	Boot	60	8.4%
AntiEXE.A	Boot	52	7.3%
Concept	Macro	47	6.6%
Parity_Boot.A	Boot	44	6.2%
AntiCMOS.A	Boot	41	5.8%
MDMA	Macro	39	5.5%
Empire_Monkey.B	Boot	37	5.2%
Wazzu.?	Macro	36	5.1%
Ripper	Boot	33	4.6%
Imposter	Macro	27	3.8%
NYB	Boot	24	3.4%
Quandary	Boot	24	3.4%
Junkie	Multi	23	3.2%
EXEBug	Boot	22	3.1%
Npad	Macro	18	2.5%
Sampo	Boot	13	1.8%
Telefonica	Multi	13	1.8%
Stoned.Angelina	Boot	12	1.7%
WelcomB	Boot	12	1.7%
Assistant	Macro	10	1.4%
Hare.?	Multi	10	1.4%
Tentacle	File	10	1.4%
One_Half.?	Multi	9	1.3%
Parity_Boot.B	Boot	9	1.3%
DelCMOS.B	Boot	7	1.0%
Edwin	Boot	7	1.0%
Empire.Monkey.A	Boot	6	0.8%
Laroux	Macro	6	0.8%
Jumper.B	Boot	5	0.7%
Tequila	Multi	5	0.7%
Burglar.1150.A	File	4	0.6%
Stealth_Boot.C	Boot	4	0.6%
V-Sign	Boot	4	0.6%
Form.D	Boot	3	0.4%
Unashamed	Boot	3	0.4%
Cascade	File	2	0.3%
Defo	Boot	2	0.3%
Int40	Boot	2	0.3%
Irish	Macro	2	0.3%
Stoned.Spirit	Boot	2	0.3%
Other [1]		23	3.2%
Total		712	100%

[1] The Prevalence Table includes one report of each of the following viruses: AntiCMOS.B, Barrotes.1310, Bath, Boot.437, Chance.A, Diablo_Boot, Die_Hard, Disk_Washer, EncePhalyt, Halloween, Helper, IntAA, Leandro, Major.1691, Multiani, Natas.4744, Neuroquila, Nutcracker.?, Ornate, RDA_Fighter, She_Has, Spanish Telecom, and Stealth_Boot.H.

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 November 1996. Each entry consists of the virus name, its aliases (if any) and the virus type. This is followed by a short description (if available) and a 24-byte hexadecimal search pattern to detect the presence of the virus with a disk utility or a dedicated scanner which contains a user-updatable pattern library.

Type Codes

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

Change.663 — **CER:** An encrypted, appending 663-byte fast infector. The virus infects all files in the current directory when the ‘cd’ (‘change directory’) command is issued.

```
Change.663          81EE 0C00 560E 1F8A 8427 0081 C628 00B9 6F02 3004 46E2 FBEB
```

Chiche.1436 — **CER:** A stealth, encrypted, appending, 1436-byte virus containing the text: ‘Virus Chiche Ver. 0.99a (C)Bugs Bunny [DAN] Digital Anarchy 2/11/94Fuck! Telefonica Argentina’. The virus overwrites the MBR with its own code, from 2–14 June it displays the message: ‘Un regalito para el JUAN XXI’. All infected files have their time-stamps set to 62 seconds.

```
Chiche.1436         368B 2D83 ED07 83C4 021E 060E 0E1F 078D B629 00B9 7305 8034
```

Chkbox.892 — **ER:** An appending, 892-byte virus that looks like a prelude to the Chkbox.936 virus (see *VB*, September 1996). It contains the same encrypted text: ‘iamtheboss’, ‘checkboxports’ and ‘givegodmode’ but infects only one file: ‘c:\test\test\test\test\test.exe’. Every infected file has its time-stamp set to 10 seconds.

```
Chkbox.892          B803 63CD 213D FFFF 747F 9090 8CD8 488E D88B 1E03 0083 EB3B
```

Djifx.2372 — **CER:** An appending, 2372-byte virus containing the encrypted text: ‘DJ[I]-FX, Ver. 0.53., (c)1996 by ■ ■ ■ Just call my name and I’ll be back... Password: (D&I) Enigma:’. All infected COM files are marked with the character ‘I’ located at offset 0003h, and EXE files with characters ‘DJ’ at offset 0012h.

```
Djifx.2372          2E00 A412 072E FE45 F92E 8AA4 1207 2E30 8412 0786 C446 E2E4
```

Dnepr.377 — **CR:** An appending, 377-byte virus which hooks interrupts 21h and 1Ch. The payload shows a message in the middle of the screen. The white text on the red background reads: ‘DNEPR-CHAMPION’.

```
Dnepr.377           B979 0131 D2B4 40CD FFB8 0042 31C9 CDFF B440 BA4F 00B1 0389
```

Hidenowt.1741.B — **CER:** An appending, 1741-byte virus with stealth capabilities; a minor variant of the original Hidenowt virus. A few changed (corrupted?) bytes cause the unreliable infection of EXE files (system may hang during infection attempt). The virus infects on Find First/Next calls (i.e. files are infected while system is executing the ‘dir’ command). All infected files have the characters ‘Yx’ located near the end of the code.

```
Hidenowt.1741.B     1E06 9C58 25FF F850 9DE4 0100 2000 0000 0043 E853 01C1 F8E5
```

IR.469 — **CN:** An appending, 469-byte, direct infector infecting three files at a time. It contains the plain-text strings: ‘????????EXE’ and ‘*.EXE’. Infected files are marked with characters ‘IR’ located at offset 12h.

```
IR.469              80BE 2901 4D75 693E 81BE 3B01 4952 7462 B802 4233 C933 D2CD
```

Koder.1024 — **ER:** An encrypted, appending, 1024-byte virus containing the text: ‘KODER(96) WROCLAW’. It does not infect files on diskettes in floppy drives. All infected files have their time-stamps set to 60 seconds. When active in memory it makes a click on every keystroke.

```
Koder.1024          FF33 F6B9 6D00 F3A5 B992 01AD 33C2 D1C8 ABE2 F806 B8DA 0050
```

Mask.2389 — **CER:** An encrypted, appending, 2389-byte virus, containing a picture of a Christmas tree, and text: ‘* /|\ / /|\\ ///|\\\ ////|\\\\ /////|\\\\\ | # Happy New Year, buddy !-Yours truly. The Mask 2 System diagnostic failure: cylinder #1 of your hard disk is cracked. There is a peril of hard disk explosion. Only an immediate formatting can help. Confirm format now [Y/N]? Say GOOD BYE to all data on your hard disk ! Start formatting ... Wrong. Your hard disk NEEDS to be formatted. Format complete. Peril of explosion is not over. Next several weeks be very careful: if you will hear suspicious noise inside computer, immediately rush down to the floor, quickly crawl to phone and call for rescue rangers (911). Relax ! It’s a joke. Your hard disk is OK. Video signal lost. Some device causes strong interference. Please reorient or relocate your computer or consult experienced technician for help. Press the ENTER key. << The Mask 2. Dedicated to Mrs Marina - the Wonderfulest Woman I ever met. >>’.

```
Mask.2389           E800 005D 8D76 FDB9 4109 B0?? 3046 1145 04?? E2F8
```

Mipt.602 — **CR:** An appending, 602-byte virus containing the text: ‘logo.pic’, ‘heretic.wad’, ‘kb2.dat’, ‘COM’ and ‘MIPT(75),1995(C),TERMINATOR’. All infected files have their time-stamps set to 30 seconds.

```
Mipt.602            B440 B95A 0290 BA04 010E 1FE8 C100 7221 B800 42B9 0000 BA00
```

Mipt.2000	**CR:** An appending, 2000-byte virus containing the text: ‘\dos\count.sys’ and ‘MIPT (72) (C) EliteSoft February 1996 ****** 50 YEAR PhysTech ******’. Infected files have time-stamps set to midnight or 1am. `Mipt.2000           891E 9306 8C06 9506 B821 25BA 0A07 CD21 B813 35CD 2189 1E97`
Mmca.505	**CER:** An appending, 505-byte virus. All infected files end with the following sequence of bytes: 8Ah, 8Fh, 88h (the virus uses this ‘signature’ for self-recognition). `Mmca.505            0188 7503 E98B 0081 C600 0189 36ED 0133 D2B4 40B9 F901 CD21`
Mmca.1131	**CER:** An encrypted, inserting, 1131-byte (EXE) and 1134-byte (COM) virus. The virus keeps the Int 21h service routine encrypted and decrypts it on-the-fly while infecting files. All infected COM files end with the following sequence of bytes: 8Ah, 8Fh, 88h. All infected EXE files are one byte shorter than the file-size value taken from the EXE header. `Mmca.1131           5E46 5653 2E8A 64FF B954 042E 3024 46E2 FA5B C3??`
NRLG.795	**CR:** An appending, doubly-encrypted, 795-byte virus with stealth capabilities. It contains the text: ‘[NuKE] N.R.L.G. AZRAEL’. All infected files have their time-stamps set to 60 or 62 seconds. `NRLG.795            812D 65F9 FF05 802D 7247 47EB 0590 B44C CD21 B40B CD21 E2C2`
Onkelz.527	**CN:** An encrypted, appending, 527-byte direct infector infecting one file at a time. It contains the text: ‘*.CoM’ and ‘Hello User, You have The Boehse Onkelz-C Virus I was written in Eastgermany in Sept. 1993! If you’re an Onkelz Fan, call the following number Germany (0049): 069/445052 Wir ham’ noch lange nicht genug!’. All infected files have their time-stamps set to 12 seconds. `Onkelz.527          01E8 0400 EB12 **** 8B96 1701 8BFE B9EC 01AC 32C2 AAE2 FAC3`
Onkelz.541	**CN:** An encrypted, appending, 541-byte direct infector infecting one file at a time. It contains the text: ‘*.CoM’ and ‘Hello User, You have The Boehse Onkelz Virus 1.5! I was written in Eastgermany in Sept. 1993 ! If you’re an Onkelz Fan, call the following number! Germany (0049): 069/445052 ! Wir ham’ noch lange nicht genug...!!!’. All infected files have their time-stamps set to 12 seconds. `Onkelz.541          01E8 0400 EB12 **** 8B96 1701 B9FA 018B FEAC 32C2 AAE2 FAC3`
Penthouse.1568	**CN:** An encrypted, appending, 1568-byte direct infector containing the text: ‘*.co*’. All infected files are marked with the character ‘^’ at offset 03h. The harmless payload (a tune) triggers on Tuesdays. `Penthouse.1568      0633 C08E C0B8 3412 26A3 2202 2629 0622 0274 02FF E007 58B0`
Probe.2140	**CER:** An appending, 2140-byte virus containing the encrypted texts: ‘Imperial Probe V 1.09’, ‘.EXE.COMCOMMAND.COMMSDOS’, ‘\dos\smartdrv.exe’, ‘\dos\doskey.com’, ‘\dos\keyb.com’, and ‘\windows\smartdrv.exe’. The payload triggers on the first five days of December, and includes the displayed message: ‘Please wait, Smartdrive is checking your disk structure. Warning, interrupting this task could cause disk damage !’ and ‘Oh, you are using MS-DOS. Now you see what you have got ! Greetings from Bill Gates ...’. Infected files are marked with characters ‘UP’ located at offset 03h (COM) and 12h (EXE). `Probe.2140          FCF3 A45F 1E8B 8577 068E D8BA 4503 B821 25CD 211F E80D 0007`
QDragon.414	**CR:** An appending, 414-byte virus containing the encrypted text: ‘COMMAND.COM’ and ‘Quick Dragon v.7’. When infecting COMMAND.COM, the virus inserts its code into the usually empty (filled with zeros) area at the end of the file. All infected files are marked with byte B8h located at offset 03h. `QDragon.414         3D00 4B74 2580 FC4F 751D B42F CD21 061F 8D57 1EB0 00B9 0D00`
Scroll.600	**CR:** An appending, 600-byte virus containing the plain-text string: ‘COMcom’. The virus intercepts the Ctrl-Alt-Del combination and rewinds the contents of the screen instead of rebooting the PC. `Scroll.600          B499 CD21 8CC0 3D21 4375 03E9 8200 8CD8 488E C026 8A1E 0000`
Skull.257	**CR:** An overwriting, 257-byte virus with the encrypted text: ‘*.com’ and ‘Protovirus by Skull of Death’. `Skull.257           C3BB 0002 BEFF 018A 2732 24F6 D488 274B 4E81 FB37 017D F0C3`
Slips.1475	**ER:** A stealth, appending, 1475-byte virus containing the text: ‘!SlipS gotcha!’. `Slips.1475          B809 42CD 2172 1D8C C82E 0106 A905 2E01 06B1 05FA 2E8E 16A9`
V.643	**ER:** An appending, 643-byte virus installing itself in memory only when run from the DOS box under *Windows*. Infects files on the ‘dir’ command (Find First/Next). All infected files contain the value 38h at offset 14h (initial IP). `V.643               B983 0233 D2B4 40CD 218B 160C 008B 0E0E 0081 C22E 0083 D100`
V.738	**ER:** An appending, 738-byte virus installing itself in memory only when run from the DOS box under *Windows*. Infects files on the ‘dir’ command (Find First/Next). All infected files contain the value 38h located at offset 14h (initial IP). `V.738               B9E2 0233 D2B4 40CD 218B 160C 008B 0E0E 0081 C22E 0083 D100`
Virion.254	**CN:** An overwriting, 254-byte, direct infector containing the text: ‘Virion*.COM’. The virus infects only files with the 254-byte-long array of bytes of the same value as the byte at offset 03h. `Viriron.254         FF36 1600 FF36 1800 B440 BA2B 008B 0EB0 42CD 215A 59B8 0157`
Wasp.1312	**CN:** A prepending, 1312-byte virus containing the text: ‘W.A.S.P.’, ‘PATH=’, ‘Insufficient memory’, ‘FuckingIncorrect DOS version’, ‘*.com’ and ‘tmp.$$$’. The virus payload includes a procedure shaking the contents of the monitor screen. `Wasp.1312           B440 B920 05BA 0001 CD21 7310 E86B 02E8 7102 B441 BAEB 04CD`

VIRUS ANALYSIS 1

Arianna: Computer Games

Eugene Kaspersky

It is very important to check any software downloaded from the Internet, or indeed from any other sources, with an anti-virus program. If this is not possible, a simple visual check with a debugger is often sufficient to reveal code that is obviously viral.

If such security measures are not observed, your system could be at risk of infection by the multi-partite Arianna, which could wreak havoc with data and programs.

Having analysed more than 8000 computer viruses, I am familiar with most of what virus writers have to offer. When I looked at a file infected by Arianna, I saw data, encrypted with a simple XOR or ADD/SUB method, containing text and code. This was a strong indication that there was a virus within the file.

The Infected File

When an infected EXE file is run, DOS parses the file header, loads the image of the infected file into system memory, calculates the Entry Point address, and passes control to the virus code.

First, the virus decrypts itself using a simple loop of XOR instructions and jumps to the hard drive infection and installation routine. That routine checks the system for an already-loaded TSR copy of the virus and returns to DOS if the virus is already in the system.

To detect whether a TSR copy of the virus is present, Arianna obtains the segment address of the current Int 21h handler and looks for the word 4D5Ah, or 5A4Dh, at offset 0B13h within this segment. If the virus is already present in system memory, the data at that address contains the read/write buffer which the virus uses to store the EXE header while infecting a file. As Arianna infects only EXE files, that buffer will start with the EXE file stamp MZ or ZM: the virus uses this marker to recognize its TSR copy.

If a copy of the virus is already in memory, Arianna returns to the host program; otherwise it infects the MBR of the hard drive, and hooks Interrupts 21h and 22h (Program Termination Address). If the MBR of the hard drive is infected successfully, the virus also hooks Int 13h for its stealth routine.

To leave its TSR copy in memory, the virus patches the Memory Control Blocks chain, decreasing the size of the last block and copying itself there. Arianna accesses neither UMB nor HMA, but rather leaves its code in conventional DOS memory.

Then the virus terminates program execution using the DOS call Int 21h, AX=4C00h (Terminate with Return Code). That brings control to the Int 22h handler which has already been hooked by the virus. This handler executes the host file with the DOS call Int 21h, AX=4B00h (Load and Execute), and returns to the original Int 22h handler.

When the infected file is executed by the virus' Int 22h handler, the virus hooks that call, disinfects the file, executes it and then once more reinfects; that is, the virus temporarily disinfects the host file.

At this stage, the installation routine has completed. The MBR is infected, the virus is in memory, and the host file is executing.

Infecting the Hard Drive

When infecting the hard drive's MBR, the virus reads it into system memory, parses the Partition Table and looks for FAT12, FAT16 or BIGDOS entries there. If there is no such entry, the virus terminates the infection routine. Arianna does not reinfect an already infected MBR: to check whether or not it has already infected the MBR, the virus compares the word at offset 001Eh with FF36h, which is part of the virus code.

"the trigger routine is executed approximately one month after the MBR was infected"

Then the virus gets the parameters of the logical disk pointed to by the Partition Table entry, saves the original MBR and the virus code to the last sectors of that logical disk, and overwrites 29h bytes of the MBR with the virus bootstrap code. The virus also decreases the size of that logical disk in the Partition Table, and thus leaves its code in 'extra' sectors on the logical disk. This trick was first seen in Flip, a multi-partite virus.

The infected MBR is recoverable using the command FDISK /MBR; however, this does not restore the size of the logical disk – it stays reduced.

Loading from an Infected MBR

When loading from an infected MBR, the virus decreases the size of DOS memory (by modifying the word at address 0000:0413h), hooks Int 1Ch and returns control to the original MBR sector. The Int 1Ch handler checks the address of the DOS Int 21h handler: when it changes (the DOS loading procedure is complete), the virus releases Int 1Ch and hooks Interrupts 13h and 21h.

This procedure is quite simple, and is one of the standard methods used by multi-partite viruses to leave their code in system memory while loading from an infected disk.

Interrupt Handlers

The Int 13h handler contains only a stealth routine, which does not allow reading from, or writing to, the infected MBR. While reading from the infected MBR sector, the virus ‘shows’ the original (uninfected) one – a standard stealth trick used by boot sector and multi-partite viruses.

The virus’ Int 21h handler intercepts nine DOS functions:

- 4B00h – Load and Execute
- 3Ch, 3Dh, 3Eh – Create, Open, Close file
- 1Ah – Set DTA
- 11h, 12h, 4Eh, 4Fh – Find First/Next FCB/ASCII

Arianna infects EXE files when they are executed. When an infected file is opened, the virus disinfects it, and restores the original time- and date-stamp. When opening or creating an EXE file, the virus stores the opened file’s handle and infects the file when it is closed.

Hooking the Set DTA and FindFirst/Next functions enables the virus to realize the stealth routine while looking at the contents of directories. The virus ‘decreases’ the size of infected files to make them appear uninfected.

To infect a file, the virus encrypts and writes its code to the end of the file, and alters the EXE header. It also temporarily hooks the Int 24h vector to disable the standard DOS error message while attempting to write to write-protected disks, and gets/clears/restores the file attributes and time mask. This is a standard method, used by thousands of DOS EXE viruses.

To distinguish between infected and uninfected files, Arianna uses another standard method. While infecting a file the virus sets its time-stamp to 62 seconds: this is a ploy which has been around since Vienna.648, the non-memory-resident ‘grandfather’ of modern viruses.

Trigger Routine

While infecting the hard drive MBR, the virus gets and checks the current date, then increases the current month value by one and stores the result in its code. If the current date is the sixteenth of the month or later, the virus increases the month value by two.

Then, on each Int 21h call, the virus checks the system date: if the current month is the same as the value stored while infecting the MBR, the virus calls the trigger routine. Thus, the trigger routine is executed approximately one month after the MBR was infected.

Unfortunately, the sample of the virus in my possession appears slightly corrupted, and it is impossible to recover the effect routine. However, analysis of the uncorrupted parts brings up a video effect routine which displays this message:

```
ARIANNA is changing your computer activity
If you wish no damage do not turn it off
           ThanX for diffusion
```

Depending on the system timer, this effect routine also overwrites the MBR of the hard drive and the sectors saved when the MBR was infected. The Trojan displays this message while loading from a corrupted disk:

```
“ARIANNA VIRUS”
HAS DONE A RECOVERABLE DAMAGE
GOOD LUCK FRIEND !!
```

The virus also contains the string:

```
Coded in Bari thanX 2 DOS UNDOCUMENTED
```

Conclusion

Arianna is a relatively uninteresting virus – it contains no innovative or even new tricks, and offers no resistance to detection. Like many other such viruses, the only remarkable thing about it is that it is in the wild.

Arianna

Aliases:	None known.
Type:	Memory-resident, multi-partite virus with stealth capabilities. Encrypted but not polymorphic.
Infection:	DOS EXE files and hard drive Master Boot Record.
Self-recognition in MBR:	Compares the code at offset 001Eh with the value FF36h.
Self-recognition in Files:	File’s time-stamp is set to 62 seconds.
Self-recognition in Memory:	Checks the data in segment of Int 21h handler; see analysis.

Hex Pattern in Files:	`E800 005E 1E0E 0E07 1FB9 F60A`
	`83EE 048B FEFD AC34 ??AA E2FA`

Hex Pattern in MBS and in Memory:	`FA33 DB8E D3BC 007C FBCD 122D`
	`0600 B106 D3E0 8EC0 B807 02B9`
	`???? BA?? ??9C 36FF 1E4C 00B8`

Intercepts:	Interrupts 13h, 1Ch, and 21h for installation, infection and stealth.
Trigger:	Displays a message, overwrites the Master Boot Record.
Removal:	Under clean system conditions, identify and replace infected files; repair an infected MBR with FDISK /MBR. See analysis for details.

VIRUS ANALYSIS 2

Particularly Shifty

Matthew Brown
Sophos Plc

After several years of looking at viruses, it has become clear to me that most of them are boring, tedious beasts; rehashings of the same unoriginal ideas. It's almost as if they're trying to discourage virus researchers with their sheer tedium.

Occasionally, though, something like ShiftPart comes along. This is a small virus, only one sector long, yet it is a true box of tricks. Every dozen lines is yet another sneaky way of doing things; it seemed that, while analysing it, I found another cunning feature almost every few minutes.

Stealth, anti-heuristics, anti-behaviour blockers, port level HD access, 'Rainbow'-style recursive partitions [*see VB, September 1995, p.12*], dynamically-loaded infection routine, minuscule in-memory size – I don't see how much more could be fitted into a single sector of code.

Like many of the more fascinating viruses, this one came directly from the wild, in this case from Slovenia.

Installation

The virus' first action at boot time is to set the stack pointer to 7C00h, just beneath its own code. This is critical, as we shall shortly see. Next, it copies 56 bytes of code to the location 0000:0300h, followed by the value of the Int 13h vector from the interrupt table. These 60 bytes are the entire permanently-resident portion of the virus, making it one of the smallest memory-resident viruses I have ever seen.

ShiftPart then searches through the ROM BIOS for a call to Int 18h. Int 18h is the ROM BASIC vector, which the system calls when no boot devices can be found. Most PCs do not have a ROM BASIC, in which case Int 18h generally causes a system hang. As far as the virus is concerned, what matters is that the BIOS always has an Int 18h call in it, and the vector is never used after boot time.

It then sets the Int 13h vector to point to the Int 18h call it has just found, and the Int 18h vector to the code just copied into 0000:0300h. Thus, it intercepts Int 13h while still leaving it pointing to a location in the BIOS ROM. Many programs will thus assume that no virus is hooking Int 13h.

Next, it checks a counter to decide whether to trigger, as described below. If not, and the boot is from floppy disk, it jumps to the hard disk infection routine. If the boot is from hard disk, ShiftPart pushes the word '13CDh' onto the stack and then does an indirect jump via the stack pointer to that location. This word, of course, is the opcode for 'Int 13h', and because the stack pointer was set to 7C00h on entry, the Int 13h just pushed is immediately before that location. This Int 13h loads the real, pre-infection MBR and partition table, and since the new MBR is loaded right after the Int 13h that loads it, control flows directly into it without a jump.

Hard Disk Infection

On a floppy boot, the virus loads the MBR and checks to see whether or not it is already infected. If not, it saves the virus code and the original MBR on the first track of the hard disk, in sectors 5 and 6 respectively. Then, it edits the partition table in memory to add a recursive extended partition in a manner similar to Rainbow.

ShiftPart assumes that the disk contains a bootable partition as partition 1 at the start of the disk, and that partition 2 is empty and can be used for the virus' extended partition. No checking is done, so if this is not the case, data will be destroyed and the machine may be rendered unbootable.

"the virus overwrites randomly-chosen areas of the hard disk in an endless loop"

The virus then writes out the modified MBR by driving the IDE hardware directly through the ports, probably as an anti-heuristic measure. Next, it writes out an extended partition table at sector 4 on the first track. The looping extended partition points to this sector, 4, which in turn references itself. Finally, the virus calls Int 19h to restart the booting process.

Resident Behaviour

As stated previously, only 60 bytes of virus code are permanently resident in memory. This code intercepts Int 13h as described above. It traps read and write to the first sector either on floppy disk or on the first hard disk.

This code first saves the first 1KB of the IO buffer in sectors 7 and 8 on the first track of the hard drive. Next, it overwrites this memory by loading the virus code in from disk – dynamically loading the main body of the virus into memory that will be immediately overwritten.

Next, the virus jumps to the loaded code. For hard disks, this stealths the partition table changes and returns; otherwise, if the operation is a read, it goes ahead and tries to infect. Writes to floppy disk will not cause infection, since a read was always first and the disk is probably already infected.

ShiftPart then saves the disk's volume label, and tests the infection marker at offset 20h in the boot sector. If this reveals that the disk is already infected, stealth is performed by copying the uninfected boot sector from the C: drive and replacing the BPB (BIOS Parameter Block) with the data from the floppy disk. The original boot sector is *not* saved when a floppy disk is infected.

On an uninfected disk, the boot sector code is overwritten with the virus code; the BPB and the volume serial number are preserved, except for the infection marker. The trigger countdown is then incremented by two, putting it back by two reboots; the more disks are infected, the longer it will take before the countdown gets to zero (see 'Trigger' below).

Finally, the infection code is overwritten by the saved application buffer data, leaving no trace of it in memory.

Trigger

ShiftPart keeps a counter, which starts at 120, in an infected boot sector. One is subtracted from this counter with every reboot, while two is added with each floppy disk infected. Thus, those who are good at passing on the virus are less likely to have it trigger.

When the counter reaches zero, the virus overwrites randomly-chosen areas of the hard disk in an endless loop, which will eventually destroy practically all data.

ShiftPart

Aliases:	None known.
Type:	Memory-resident DOS boot sector infector with stealth capabilities.
Infection:	Boot sectors of floppy disks; DOS boot sector of hard disk.
Self-recognition on Diskette:	Infected floppies have the word at offset 20h (the first word of the long sector count, always zero on normal uninfected floppies) set to a non-zero value.
Intercepts:	Int 13h, redirected to point to a call to Int 18h in the BIOS. Int 18h, infection and stealth (as does redirected Int 13h). Int CEh, as unstealthed Int 13h.
Trigger:	When counter in boot sector reaches zero, hard drive sectors are erased in a random pattern until rebooted.
Removal from Hard Disk:	Repair partition table, deleting extended partition and restoring correct start to first partition. Wipe sectors 4–10 of first track of hard disk to erase virus traces.
Removal from Floppy Disk:	Replace code area of boot sector with DOS boot code.

VIRUS ANALYSIS 3

Bandung on the Run

Toby Hutton
Cybec Pty Ltd

Indonesia seems to be an emerging source of new *Word* macro viruses: NPad was the first; now there is Bandung. Researchers and developers have confused the two because they originated in the same country: they are, however, quite different in every other respect. Bandung has no unique replication methods or payloads, but is in the wild (particularly in Australia) and is destructive: *Word* users, beware.

Bandung has six macros, none of which are encrypted by *Word's* 'execute only' feature, which is the norm with most *Word* macro viruses. The macros are AutoExec, AutoOpen, FileSave, FileSaveAs, ToolsMacro and ToolsCustomize.

Bandung's replication code, like many other macro viruses, is based on that of Concept, the oldest *Word* macro virus in the wild. Concept is consistently at the top of the virus prevalence table: because of this, and because its macros are unencrypted, it is one of the easiest viruses to plagiarize.

Replication

Bandung is activated when an infected document is opened. Gaining control as *Word* runs the virus' AutoOpen macro, the infection routine copies each virus macro to the global template (NORMAL.DOT). From there, Bandung spreads to documents via FileSave and FileSaveAs. These macros, which are very similar, run as the user saves a document, and copy each macro from the global template to the document.

In neither phase of the replication procedure does Bandung do a self check; it will replicate successfully to an already-infected document or template. Infection by this virus is very straightforward, yet this simplicity is the key to widespread distribution – the primary goal of any PC virus.

Dirty Deeds

Bandung was designed to have two different payloads, but one is disabled. The ToolsMacro and ToolsCustomize macros are identical, and, due to a 'goto' statement on the first line of the macro code, do nothing.

They are triggered as the user selects Macro or Customize from the Tools menu: as the user does so, it becomes apparent that something is amiss, as the usual dialog boxes accompanying these commands do not pop up.

The intended payload in these macros checks the date and continues only if it is later than 10 March 1996. It would then display a message box entitled 'Err@#*(c)' with the message 'Fail on step 29296' and an 'OK' button. Each letter ‘a’ in the document would then be replaced with the symbols ‘#@’. But, as mentioned, this payload is disabled and never executed.

The second payload is triggered as an infected copy of *Word* is started and the AutoExec macro is automatically called. First, it checks the date and time – the virus continues only if it is on or after the 20th of the month and after 11:00 am. This, scarily, is far too often for comfort.

The virus then counts the number of subdirectories branching from the root directory on the C: drive (C:\). A message is displayed on the status bar, courteously saying ‘Reading menu...Please wait !’. Bandung then deletes every file in all those directories except ‘C:\WINDOWS’, ‘C:\WINWORD’ or C:\WINWORD6’, if they exist. Bandung is obviously trying to avoid any complications that may arise from deleting its host, *Word*, the currently-active application.

The virus then repeats the process in each subdirectory, skipping the directories ‘WINWORD’ or ‘WINWORD6’. Next, to be thorough, it deletes every file in every subdirectory branching from these first-level subdirectories. Effectively, every file in every directory three levels deep is deleted, unless they are ‘lucky’ enough to stem from ‘WINDOWS’, ‘WINWORD’ or ‘WINWORD6’ directories.

If, like many people, you use non-standard directories for your applications (I use ‘C:\WIN3’ for my *Windows 3.11* directory, and ‘C:\WORD6’ for my *Word 6.0* directory), none of your files may be safe!

A file called PESAN.TXT is created in the root directory on drive C after every other file has been deleted. It contains a message from the virus author, written in Indonesian, claiming responsibility for the missing files on drive C. The message ends with ‘Bandung’, presumably the author’s home city, and the date and time the payload triggered.

Detection and Removal

There are two obvious signs that a file is infected with Bandung. First, many directories on the C: drive may be empty, and a file called ‘PESAN.TXT’ will be present in the root directory of drive C. Second, the menu items Tools/Macro and Tools/Customize do nothing. Thus, making a positive identification of the virus is not straightforward, as the easiest way to see if macros exist in currently-active templates is through the Tools/Macro dialog box.

To check for or delete macros from the system without using Tools/Macro, create a toolbar button to launch the Template Organizer. Usually, this can be done with Tools/Customize, but on a Bandung-infected system, that menu item is disabled. In this case, right-click on any existing toolbar, and select ‘Customize’ from the pop-up menu. Select ‘All Commands’ from the Categories list, and drag ‘Organizer’ from the Commands list to your workspace. Assign it a name, close the Customize dialog box, and you will be able to launch the Template Organizer from the newly-created toolbar button.

Once you have launched the Template Organizer, you may delete any of the macros in any open template, including NORMAL.DOT. Each of Bandung’s macros have common names which may be found in many other macro viruses or in harmless custom utilities and plug-ins for *Word*.

If you are unsure whether or not a macro should be in your global template, rename it (use an ambiguous name; perhaps add ‘1’ to the end), so *Word* will not run it automatically in the future – this applies particularly to the ToolsMacro macro (if it exists). The Tools/Macro menu item may then be selected and the macro code examined. Manual deletion of macro viruses is unreliable: all reputable anti-virus vendors should be able to detect and clean Bandung in the near future.

Mutation

A variation of Bandung (Bandung.B), which corrupts versions of the AutoExec, ToolsMacro and ToolsCustomize macros, is already in existence. *Word* refuses to execute these macros, displaying an error dialog box, but since they contain only the payload code, the virus can still replicate with the still-intact remaining macros.

This type of corruption is a common source of new strains of macro viruses. When *Word* copies macros via the MacroCopy command or the Template Organizer (every macro virus to date employs one or other of these methods), it does not check the validity of the code being copied. If a file is corrupted, and the binary image of a macro within the *Word* document is subsequently invalidated, *Word* will happily copy the macro, byte for byte, from source to target.

Thus, a corrupted macro may spread with the rest of a virus without necessarily impeding the virus’ performance, especially if the macro is cosmetic (e.g. Concept’s Payload macro). Unfortunately, these mutations make a *Word* macro virus less detectable using conventional anti-virus methods, but only until the corrupted virus is identified by anti-virus researchers: it can then be detected and removed as easily as the original. This is the case for Bandung.B.

Bandung

Aliases:	Jakarta.
Type:	*MS Word* file infector.
Self-recognition:	None.
Hex Pattern:	`6174 6452 6905 4F6A 656B 240C 6A03 433A 5C07 6908 5375 6264`
Trigger:	When *Word* is started on or after the 20th of the month, later than 11:00 am.
Payload:	Files are deleted from drive C. See text.
Removal:	See text.

VIRUS ANALYSIS 4

Tales of the Anarchic

MDMA is neither particularly new – it has been on Joe Wells' *WildList* since July 1996 – nor very complicated, although it does contain some interesting trigger code. It consists of a single *Word* macro, AutoClose, which gives the virus control whenever a document is closed within *Word*.

The virus first calls its 'install' subroutine, which checks the Global Template for an AutoClose macro. If one is not found, the virus' macro is copied from the current document to NORMAL.DOT, infecting that machine's *Word* setup.

If an AutoClose macro is already present in the Global Template, the virus checks the current document for an AutoClose macro. If not found, it copies NORMAL.DOT's AutoClose into the document, and adjusts the document type so that when it is next saved, it is saved as a template.

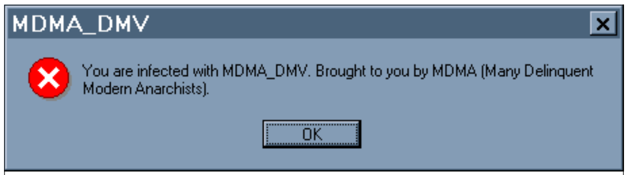
Trigger

The only interesting thing about MDMA (and the reason for its ill-deserved fame) is its trigger. This has been coded with reasonable skill; however, there are mistakes (one in particular) which prevent it functioning properly under all conditions.

After carrying out the infection routine, the virus checks the date, and only proceeds if it is the first of any month. In this case, the virus calls the subroutine 'get_platform', which attempts to determine the OS hosting this *Word* installation.

Due to some idiosyncrasies in *WordBasic*, this task is not as easy as one might expect. Certain useful calls will generate an error on the *Mac*, for example, and thus reveal the presence of the virus. MDMA works around some of these. First, it uses AppInfo$(1) to get the name of the underlying OS – checking to see if this contains the word 'Macintosh' will establish that this is *Word* for the *Macintosh*.

The next check is to call AppInfo$(9) to return the total memory size in K. This may seem peculiar initially, but all becomes clear when the Help is consulted: this reads 'In Windows 95 and Windows NT, this value returns 0' [sic]. If non-zero, the virus assumes this is a *Windows 3.x* machine. Unfortunately, the function also returns non-zero on the *Macintosh*, so the virus promptly forgets that the computer is a *Mac*, and starts thinking it's *Windows 3.x*.


MDMA_DMV

You are infected with MDMA_DMV. Brought to you by MDMA (Many Delinquent Modern Anarchists).

Figure 1: The message box displayed by MDMA.

The final check (GetSystemInfo$(21)) is another method to get a string identifying the system type: the virus looks for 'NT' in the return value. If this is found, the system is clearly *Windows NT*. This call generates an error on the *Mac*, and the execution path becomes confused, as MDMA will have requested that execution should continue on the next line whenever there is an error. On a non-*Macintosh*, this function returns with the correct OS type; on a *Mac* (as mentioned), it will most likely produce an error.

Payload

The payload executed depends on the OS the machine is running. On *Windows 3.x*, the virus appends commands to AUTOEXEC.BAT to delete all files from the C: drive, and to display the message: 'You have just been phucked over by a virus'. These commands will be executed on the next reboot. On *Windows NT*, the virus simply deletes all files in the current directory.

Perhaps the most curious payload is that reserved for *Windows 95* systems. In addition to deleting all .CPL and .HLP files from the directory 'C:\WINDOWS', MDMA uses *Word* calls to modify values within the system registry. It turns off login script processing (to inconvenience network users); it turns on StickyKeys, part of the Accessibility options in *Windows 95* (if these are not installed, this has no effect); and it turns on HighContrast (also part of Accessibility, this sets the desktop to an extraordinary colour scheme, useful to the visually impaired). In addition, on all above platforms, MDMA attempts to remove a file called 'shmk.' from the root of the C: drive.

The *Mac* payload attempts to delete all files in the current directory, but it is unlikely that execution will make it to this point: on a *Mac*, the virus will have generated several errors by now. Finally, the trigger causes the dialog box shown in Figure 1 to be displayed.

MDMA

Aliases:	MDMA-DMV.
Type:	*MS Word* file infector.
Self-recognition:	AutoClose macro in NORMAL.DOT.
Hex Pattern:	`C3CF C7C0 EAA2 A396 A8EA DCE7 89E7 E0FD FAEF E2E2 EADC 938B`
Trigger:	First day of any month.
Payload:	Various system-related payloads. See text for details.

VIRUS ANALYSIS 5

Urkel

Paul Ducklin
Sophos Plc

Remember Ping-Pong? In those days, we were often advised that if we saw the message 'Non-system disk or disk error', we should switch off, remove the floppy, and only then try to reboot. As Ping-Pong infects the hard disk only when it is first accessed by DOS, doing this can forestall infection.

This scheme does no good with most other boot sector viruses, of course, which hit the hard disk during an infective floppy bootup. By the time you see the message, it is probably too late. 'Non-system disk' does serve as a warning, however, as it usually indicates a mistaken (and potentially dangerous) attempt to boot from some arbitrary floppy disk.

Urkel has an irritating trick which hides the warning: firing up from an infected floppy causes the virus to infect the hard disk, then boot from it. The floppy in the diskette drive goes unnoticed, and the machine appears to load normally from the hard disk.

Pick a Spot

At boot time, Urkel copies itself into segment 9F80h, which is the last 2KB of base memory on a regular 640KB machine. During this copy, the bulk of the virus, which is XORed with a random byte, is decrypted into its dwelling place.

The value 7Eh is then written into the low byte of the BIOS base-memory count at 40h:13h. The value originally stored there is typically 80h (this corresponds to the low byte of the value 280h; 640 in decimal), so this completes the 'theft' of 2KB of memory by the virus. Urkel uses only the first 1KB of this stolen region, allowing it to work on machines in which the last 1KB of base memory is used by the BIOS.

The virus then looks to see if the hard disk is uninfected (even after booting from the hard disk), immediately infecting it if so. The bootstrap then continues from the hard disk, even if the infective bootup was from a floppy. This effectively conceals the moment of infection (as described above).

Stealth

Urkel intercepts Int 13h functions 02h and 03h (read and write). Whenever sector 0.0.1 is accessed, the virus looks to see if the disk is already infected (which it signals with the value 23h at offset 1FDh in the sector). If not, the virus decides on a hiding place for the original boot sector, stashes it in this hiding place, then writes a copy of itself over 0.0.1.

The hiding place is 0.0.5 on the hard disk; on floppies, the standard places (insofar as the word 'standard' can be applied to boot viruses) are used: 0.1.3 (360KB), 0.1.14 (1.2MB), 0.1.5 (720KB), and 0.1.15 (1.44MB).

When infecting, the virus leaves bytes 03h to 23h of the original boot sector intact, which maintains the BPB area when infecting floppies. It overwrites the area from 1BEh to 1FDh with its own code and data, however, so it does not preserve the partition table when infecting hard disks. Thus, after a clean boot, partitions on the hard disk are inaccessible.

To keep things looking healthy after an infective bootup, Urkel employs stealth. Attempts to read Urkel's boot sector are replaced with a read of the original boot sector, making everything seem normal when the virus is active.

To frustrate disinfection of the virus, the stashed copy of the original boot sector is scrambled with a rolling XOR loop. This scrambling is automatically undone by Urkel's stealth routines.

Urkel's own code in the boot sector is also scrambled, as previously mentioned. The encryption is a simple bytewise XOR loop; the 'key' is selected at infection time by reading the realtime clock and using the number of minutes past the hour. The encryption serves little purpose, given its simplicity and the distinctiveness of the decryption loop, but it does obscure the lone text-string in the virus: 'Urkel'.

Warhead

This string is used by the virus during the midnight hour (00h00 to 00h59), when it is printed to screen using Int 10h function 0Eh (video teletype output) every time a disk write request (Int 13h function 03h) is issued. The string is printed at the current cursor position, followed by a carriage return.

Urkel

Aliases:	None known.
Type:	MBS and FBS infector.
Self-recognition in Boot Sector:	Byte 23h at offset 1FDh.

Hex pattern:	`B880 9F8E C08A 0EFC 018A 0483`
	`FE4B 7E04 32C1 8804 2688 0446`

Intercepts:	Int 13h, functions 02h and 03h.
Trigger:	Prints 'Urkel' on every disk write during the hour of midnight (00h00 to 00h59).
Removal:	Do not use FDISK /MBR! The partition table is obliterated by the virus. The original MBS is stored at 0.0.5, but must be unscrambled before it is copied back. XOR each byte with the low byte of its offset in the sector.

FEATURE

How Much is that Virus in the Window?

Shane Coursen

Many businesses have fallen victim to the destructive nature of computer viruses since they first appeared ten years ago. Back then, while many companies did have data security measures in place, the vast majority were not prepared to deal with a computer virus threat.

Until that time, data loss had mostly come from 'undocumented features' in software they knew they were using. Suddenly, a new threat emerged: programs *designed* to endanger data integrity. By the time most people were able to employ a more adequate data defence, a computer virus had often entrenched itself, ultimately costing those businesses untold amounts of money.

Benign or destructive, most damage comes simply from the disruption of business; the loss of time. The words 'computer virus' are not usually the first that come to mind after encountering an apparent software glitch. In the minds of most, such a problem seems to scream driver conflict. Attempting to resolve the problem from that perspective ultimately results in a loss of time.

There are also indirect costs associated with computer viruses. For example, if a company roster does not include somebody familiar with computer viruses, training costs may have to be added.

Of course, one person alone cannot hope to solve the entire problem. Many others may be conscripted from their normal day-to-day functions to aid in the recovery process. Those who are able to work 'as usual', in many cases, will find that they cannot, since their computer either has no access to company servers, or has itself become a victim.

Once a computer virus strikes, it is never a matter of a simple *BandAid* fix. Merely slowing the propagation of a computer virus does not work: the solution must be its annihilation. Adding to the problem, in a business environment, the fix must be implemented in such a way that it is all-encompassing and simultaneous across infected sites. Tending to one site and neglecting another will surely allow a persistent virus to work its way back again.

This article investigates why such small entities as computer viruses make such a large impact on today's businesses. A look at the obvious costs associated with benign and destructive viruses gives testimony to their impact. A closer look at the indirect – the intangible – costs reveals why computer viruses may be more serious than may be thought at first glance.

Every operation comes with a price tag, and implementing an emergency recovery plan is no different. Accounts of what can happen when a computer virus (real or imagined) strikes will give incontrovertible proof of just how much computer viruses cost businesses.

Part 1: Infection

A common misconception is that all computer viruses intentionally delete files and overwrite hard drives; that all computer viruses are physically destructive. This is untrue. In fact, the most successful viruses do nothing but replicate.

Benign Computer Viruses

A possible reason for the success of benign computer viruses is due in part simply to staying hidden. Modifying data in a manner not apparent to a user may allow a benign virus to go undetected. If the virus code is sound enough and the computer continues to operate with few or no obvious symptoms, it is likely that the virus will quietly make its way to other computers.

This may lead to the conclusion that benign computer viruses are not really a problem. This is also not true. To clarify the term, a benign computer virus (as it has come to be known) is one which has no purposely destructive routines; whose only apparent intent is to propagate.

In this case, 'purposely' is the operative word. Most damage done by viruses – benign or otherwise – is due to bugs and incompatibilities. A computer virus that does not make itself apparent may go undetected, and undetected viruses tend to propagate, sometimes at a very high rate. Computer viruses often break the 'rules' of the operating system: for this reason something, somewhere along the way, is bound to break.

If the virus is new to anti-virus researchers, there may be a 6–48 hour waiting period before a method of detection and removal is available. The waiting period might be even longer if it is a newly-discovered class of virus (as Concept was to anti-virus producers when it first appeared), or if the virus is especially complex.

Policy and Procedures

Throughout the waiting period you have choices. Two are very basic. The first is to shut down computers suspected of being infected. Since it is usually not known which computers are infected, you must figure on temporarily shutting down at least one segment of the network. The second choice is to continue working as though nothing out of the ordinary is happening. Although this sounds ridiculous, in the case where a virus is believed to be benign, there is a tendency to do this! Just remember, even the supposedly harmless viruses are sometimes not totally harmless; you will not know until it is too late. The third choice is to coordinate and implement a virus recovery plan.

The first choice has obvious drawbacks. We may as well accept that if any real amount of work is to be accomplished, we must have our little silicon friends in operation. Once committed to shutting down computers, or taking servers off-line, one can be assured that business will not go on 'as usual'.

No surprise, the second choice also has obvious drawbacks. What is surprising is just how many people believe this to be an acceptable course of action. This may, in part, be the fault of anti-virus researchers. Emphasizing that most known viruses are benign may give people a false sense of security. The belief seems to be that, since the virus is benign, why worry about it? It seems not to have hurt anything yet…

The most prudent course of action is, of course, to implement a recovery plan. If adequate security procedures are in place, the chances of even having to deal with a virus are greatly reduced. In the event that one manages to get through and propagate, a recovery plan will at least make the ordeal a bit more manageable. A sound recovery plan can make all the difference in getting things back to normal.

"to those directly affected by a damaging computer virus, there is sometimes no other choice than to restore data from a backup"

Fortunately, virus outbreaks are not all that common. Slight variations between computers, operating system versions, or device configurations can cause the behaviour of a virus to change from machine to machine. Before the virus has a chance to propagate, it has been impeded by its own buggy code. Such incompatibilities do not always manifest themselves in like fashion. Some of the unlikeliest non-viral symptoms sometimes turn out to be viruses (and vice-versa), and much time can be wasted while troubleshooting misleading symptoms.

In some businesses – usually very large corporations with a great many computers – one system or another always seems to be locking up. Network servers seem to abend at regular intervals, newly installed software does weird things, upgraded hardware causes problems, and so forth.

With so many computers to look after, avoiding complacency can be a real challenge. In many cases, operators may just reboot the computer and accept their losses. Often the lockups go unreported, a computer operator mistaking the problem as the network server crashing or buggy software. While in most cases the answer will be that simple, what people do not realize is that in fact the software does contain a bug; a parasite that wastes valuable time and money.

Damaging Computer Viruses

Discovering that a 'benign' computer virus has just caused your computer to lock up can be annoying. You could lose from five minutes to several hours worth of work. Annoyed, however, would not even begin to describe somebody who has just lost their data to an intentionally damaging virus – pure rage would be more like it. In a matter of seconds, a virus can render data inaccessible, resulting in hours, weeks or months of lost work. In one extreme case, where backups were not a high priority, a writer lost two years of work!

It is probably difficult to imagine, especially for that unfortunate writer, but as physically and psychologically devastating as a damaging virus is, there is a positive side. When the payload activates, the virus makes its presence known. It can no longer hide behind a curtain of uncertainty. There is very little troubleshooting required; the problem is a virus – a lemming virus[1], to be specific. There is little to do except to scan all computers and diskettes immediately, then repair or restore as necessary.

Depending on the number of computers at a business, a site-wide scan may be no small undertaking. Smaller businesses (with fewer than fifty machines, or 50–100 machines on a closed LAN) could coordinate and complete a virus recovery effort in eight to twelve hours.

However, businesses any larger than that generally have multiple interconnecting LANs. Multi-national corporations with satellite offices in foreign countries are usually connected via a WAN. Based on past experience, coordinating a virus recovery plan where time zones come in to play can take eight to twelve hours alone. In many cases, the right people are not immediately accessible.

Many factors, including accessibility to infected computers and diskettes, whether implemented by sneakernet or the Net itself, contribute to the amount of time required for full recovery. A sneakernet recovery team could scan 250 machines in 12 hours, barring any unexpected interruptions. After checking the first 250 machines, the team, although considerably more tired, is more efficient [*though perhaps less careful. Ed.*], and will take less time to scan the next 250 machines.

The preferred method is to scan from a centralized location. Immediately on accessing the network, a workstation can be automatically scanned. If necessary, access will be denied until the offending computer is cleaned up. If a clean bill of health is issued, the operation continues as usual. For machines that are not immediately accessible, the scan would simply take place at a later time. Better late than never – an automatic virus scan may prevent a costly future outbreak.

Backups

To those directly affected by a damaging computer virus, there is sometimes no other choice than to restore data from a backup. If there are good backups, restoring may take between two and sixteen hours. Two hours for an individual user; sixteen for a large business. A typical user does not usually create regular backups; however, large businesses have no valid excuse not to do so. If a restore requires different backup sets because of corruption by the virus, the time required to restore will clearly multiply correspondingly.

Many people consider tape backups the best source of recovery from catastrophic events, but should we reconsider relying solely on them? For example, Ripper, a common ItW virus, slowly modifies data as it is being archived. If Ripper goes unnoticed, there is a possibility of corrupted backups dating back several months. Those not in the habit of verifying that data is properly archived might find themselves in a very bad situation. The Catch-22 is that, while the virus lies undetected, it may look as if the data is properly archived.

Part 2: Costs and Consequences

Not taking the time to compare security products may have its costs. A selection that is not suitable to your needs can be as expensive as no selection at all. Asking others what they think about two or three different products is not exactly the best way to select a product. Where can you go for information when even many ‘independent’ reviews mislead?

Implementing a Solution

Time and care must be taken to ensure that you find a data security solution which is right for your needs. In so doing, much time can be dedicated to the review and selection of a suitable security product.

Information on the different types of security software is abundant. Many of the credible computer trade publications regularly carry reviews of the latest security products. Publications related specifically to the anti-virus industry also exist. Although they are considerably more expensive, they carry information about the latest virus threats and results of exhaustive tests based on real-world viruses.

Whether for a single computer or across an entire business enterprise, if reviews are thorough, both the trade magazines and publications focused on anti-virus issues can aid in the selection of the proper security solution.

Many security vendors offer free evaluation copies. Obtaining a copy of each is rarely difficult. On-line services and the Internet have made it possible to retrieve the files in mere minutes. Each product must then be implemented one at a time. Having more than one (active TSR/VxD) anti-virus product at the same time is not a good idea: it is very likely that one will conflict with the other, causing skewed evaluation results. A small-scale implementation of each product allows for the actual use of the product and can give you a better feel of the company standing behind it.

The expenditure involved in putting a security product in place is only the beginning of the investment. Educating employees on its use is critical to the success of the product. Unfortunately, such training does not always find its way onto the corporate agenda.

In one instance, a virus report made its way to an MIS manager. After responding to the report, not one but six distinct viruses were found. A scan of all machines in the department found that thirty out of 100 were infected. Results of a full site scan yielded similarly troubling results.

Interestingly, at least two different scanners had already (albeit sporadically) been installed on workstations throughout the company. It was not until the company purchased a third scanner that it realized it had very serious problems. The computer viruses were the obvious one. The second – and more serious – was the anti-virus software in place: the two existing anti-virus scanners were three years old, and had even been made obsolete by their manufacturers!

When software-based security measures are put into place, what is the productivity impact to the operator? Although security packages are unobtrusive and can be configured to do almost anything automatically, they are not without their faults. Problems associated with such products could have a

	Employees involved	Hours/Employee	Work-hours	Productivity Adjustment	Financial Impact
Employee Time Lost	325	4	1300	455	$8749.65
Coordinate Recovery	4	3	12	-	230.76
Scan/Install Impact	325	0.25	81.25	-	1562.44
Recovery/Data Restore/ReKey	16	6	96	-	1846.08
Diskette Scan Plan	3	15	45	-	865.35
Total Cost of Virus Incident					$13,254.28

Figure 1: This diagram shows the estimated costs involved in recovering from a virus incident, based on a flat annual salary of US$40,000.00 for each employee (see p.16 for details).

negative impact on an employee's productivity: complex user interfaces can make installation, configuration, maintenance, and normal day-to-day use difficult.

Likewise, if the software is insufficiently intuitive, the same problems could arise. A scanner could employ the best technologies in the world, but will make very little difference (except to the pocketbook) if it produces too much of a negative impact on productivity.

Naturally, anti-virus software will utilize resources that were previously available for other tasks. File transfers may appear sluggish at first. Accessing diskettes may seem to take longer. The slowdown might be imperceptible, but over time and across many computers, there is a definite impact on computer performance.

"the ability to implement a plan such as this is key to the success of any recovery effort"

Leaving yourself completely open to a virus attack is not good. Configuring a product to monitor every avenue a virus may take to infiltrate your computers can be just as bad.

Recovery

For small companies, recovering from a virus requires utilizing resources which are normally dedicated elsewhere. Typically, small companies do not have the budget or the need for MIS staff. The individual who is the most computer-literate, who (maybe as an aside) maintains the company computers, will be the one to try and solve the problem, to the sacrifice of day-to-day work. Rather than managing the business, selling, or doing research and development of their product, they spend time and energy tracking down and eliminating a virus.

Businesses large enough to dedicate several people to the job of recovery are rarely in a better position than the small businesses. The typical 'strike team' consists of one or two 'computer experts' with several other 'worker bees'. The experts devise a recovery plan and direct others throughout its implementation. More people on the recovery effort may make for greater efficiency, but it also creates a situation involving the time of not just one individual, but several.

Not surprisingly, companies that can afford to employ a staff of trained MIS professionals are the best equipped to recover from a computer virus attack. Like an army guarding a border, the MIS team represents the company's front-line defence in data security – and as such, it is well-funded and has allies.

MIS departments know to invest in their company's data security. In cooperation with MIS, outside sources specifically trained to handle viral threats may help out during the recovery effort. Even if the team is armed to the teeth with professional knowledge, the job can still be daunting. A larger company means more computers and usually a very complex and far-reaching local area or wide area network. A virus which has infected computers spanned across multiple sites would tax even the best-trained MIS team.

After cleaning the virus from the computer system, the first step is complete. The term 'initial recovery' better defines recovery from a computer virus, because viruses are sticky little programs which love to outstay their welcome.

Cleaning your computer of a virus does not guarantee that it will not come by for another visit. Just one leftover infected diskette or program can have a snowball effect and start another virus outbreak. Within a matter of hours, the entire business could be under siege again. Any time spent cleaning up from the initial infection or outbreak can easily be lost in those few hours. The complete virus recovery process would have to be repeated.

Projecting Incident Costs

This section describes what the impact might be if a number of machines in a business were affected by a computer virus. Rather than cite a specific case, several typical large-company (300-1000 node) virus incidents are used to attempt to form an average dollar impact/incident.

In this composite scenario, shortly after updating their anti-virus product, MIS personnel began to get reports of a computer virus. Not just a single report isolated to one department, but several. It didn't take long for MIS to verify the reports; a multi-partite virus was on the loose!

By obtaining another scanner, and having it find the same virus, they were pretty sure they were on to something. It wasn't until they found infected files in shared server areas that MIS decide to take several network segments off-line.

For this case, assume MIS is prepared with a sound – albeit, as of yet, not implemented – virus recovery plan. In its most basic sense, the plan gives key people the ability to verify the existence of, and quickly isolate, a computer virus. It also provides a step-by-step plan of action in the event that an actual computer virus is found. The ability to implement a plan such as this is key to the success of any recovery effort.

To start, a single network workstation is scanned. After confirming the workstation is clean of viruses, the user connects to the network and proceeds to scan all network servers. Many infected files are found in shared areas, but nothing in the boot or partition area. When all is said and done, the infected files are easily replaceable, making restoration from tape backups unnecessary. After three-and-a-half hours, the servers are brought back on-line, and employees once again allowed access to the network.

On average, an employee accesses the server 35 percent of every hour to perform a job. If an adjustment for this is made, then company-wide, 455 work-hours were lost while the network servers were off-line. Undoubtedly, throughout that same period of time, more attention was focused on the recovery effort than on normal work, further reducing each employee's productivity.

With the servers once again accepting connections, a distribution process starts as each user logs in to the network. First, a memory scan. Assuming no virus is found, the distribution process continues. Those workstations found to have outdated scanners are updated. Machines found to have no scanner at all are provided for.

A few of the workstations had unique configurations and presented unexpected problems during the installation, but not enough to add to the overall impact of the incident. The average time to scan a network node, and/or install/update the product was typically fifteen work-minutes per machine.

With all servers now back in operation, and almost every workstation scanned, the time to repair the damaged machines had come. For several computers, the damage went beyond a few infected files. The computer virus did not have a purposely damaging payload, but it did have bugs. In certain situations, there would be a very subtle loss of data. Tape backups were the only hope.

"if a company implements the proper security measures, the recovery costs will be dramatically reduced"

With this virus, as with most viruses, the backups were not directly affected. Each damaged machine required an average of two hours to rebuild, and four hours to restore it to its previous state (minus whatever data had not been backed up since the previous day). Restoring and rebuilding all damaged machines resulted in the loss of 96 work-hours.

This one incident (which incidentally was handled very well), involving 325 employees over a four-hour period of time, realized a loss in excess of US$10,000.00[2] (see Figure 1). Add to that the value of non-recoverable data, the time required to scan every diskette in the company, and other miscellaneous operational charges, and the cost of the incident comes into focus.

In this case, there were very few difficulties encountered along the way, thanks in part to a well-planned and well-implemented recovery effort. The financial impact, however, still seems quite significant. How much do viruses cost businesses each year, you ask? The answer is still unknown. If you were to place yourself in the same situation as described above, how would you fare?

Conclusions

Much to my chagrin, the particulars from incident to incident always seem to vary greatly. As everyone knows, such varied data makes it extremely difficult to give accurate figures. In the case of reporting on how much computer viruses cost businesses[3] each year, rough estimates are about the best that anybody can do.

Looking at the same topic from another angle then, the question becomes: what could a computer virus possibly cost you? When it comes right down to it, it seems that there cannot be just one answer; rather, a seemingly limitless number of possible outcomes.

The final cost of an incident depends on many different things – details such as 'Is there consistent detection of a virus in the same file when scanning with multiple scanners?' If so, the chances are that it is a computer virus. If not, it may be a false positive on the part of one scanner. 'How widespread – across the company – is the virus? For that matter, how big is the company?!'

Incidents isolated to one or two users seem to be the most commonplace. If so, the problem is usually cleared up in short order. It is much more difficult to determine how widespread a virus is in a large business environment. One usually cannot be sure that a virus is isolated to just a few users. For this reason, a site scan may be prudent.

On the more technical side: 'Can a replicative virus sample that cannot be detected by any anti-virus scanner be isolated?' In order for an anti-virus producer to help rid you of a virus, a sample is required. Sound processes are then followed to identify a possible virus and isolate a working sample.

Did the virus 'incident' turn out to be a false positive or an overactive imagination? False positives cost untold amounts of money, and worse, aggravation. Psychologically, they are draining. Fortunately, it is not difficult to identify a false positive. As was described in the September 1996 issue of *Virus Bulletin* [*When Scanners Collide, p.12*], the availability of multiple scanners provides a tool to verify the existence of a virus.

Tracking the exact cost of a virus-related incident is not easy. The number of variables involved make it almost impossible to predict the outcome. If a company implements the proper security measures, recovery costs will be dramatically reduced. Without these measures, the final cost is anybody's guess.

Bibliography and Notes

[1] See White, Kephart, and Chess, 'The Changing Ecology of Computer Viruses', proceedings of the sixth *Virus Bulletin Conference*, for the exact meaning of the term 'lemming virus'.

[2] Figure 1 gives costs based on an assumed salary of US$40,000.00/annum and a 2080-hour work year; the impact per employee is $19.23 per hour. Note that many things occur simultaneously (scanning, repairing, rebuilding, distractions, etc) during a virus recovery effort. This table does not take into account such complex interactions.

[3] American businesses only

Article reprinted with permission of *Auerbach Publications*.

Shane Coursen can be reached at the following email address: shane.coursen@us.drsolomon.com.

PRODUCT REVIEW 1

VirusNet

Dr Keith Jackson

VirusNet is a software package which describes itself as ‘a complete anti-virus and disaster recovery system’. It was provided for review on three 1.44 MB, 3.5-inch floppy disks containing both the DOS and *Windows 3.1* versions of the product. The documentation states that *VirusNet* can also be used under *Windows 95* or *OS/2*, a claim I did not test.

VB reviews exactly what is provided: I was surprised to see that we had received an ‘Evaluation version’ which expired on 1 November 1996. I worked at high speed to beat this developer-imposed deadline! The *VirusNet* documentation explained that the evaluation version of *VirusNet* does not incorporate the checksummer or the ‘Rescue Disk’ features, and displays ‘For Evaluation Only’ on several screens.

Documentation

The printed documentation provided with *VirusNet* takes minimalism to another level. It comprised just a small, thin (14-page) booklet that outlined how to install the product, how to perform a scan, what is available with the *VirusNet* ‘Rescue Disk’, and details of a few other functions. There’s more detail in the glossy bumph which accompanies *VirusNet* than in the manual!

However, this does not really matter very much, as the *Windows* on-line documentation is excellent. It provides thorough details of all the *VirusNet* functions available under *Windows*, and includes a decent index, so finding something is very easy indeed.

The DOS version of *VirusNet* also proffers on-line help, but this does not seem to be as extensive, nor as easy to use, as the help available with the *Windows* version. This is reinforced by the fact that the DOS help system is a simple 31KB text file, while the *Windows* help file occupies 96KB. The difference between the two is further exacerbated by the myriad features available within the *Windows* Help system.

Installation

Either the DOS or the *Windows 3.1* version of *VirusNet* may be chosen. *Windows* installation is, as expected, performed by the SETUP program. DOS installation is even easier: it is necessary merely to copy the *VirusNet* files to a user-specified subdirectory. I chose to install the *Windows* version, as this also installs the DOS version.

When *VirusNet’s* SETUP program is executed, the ‘Setup Wizard’ takes control. It requests the location of the subdirectory to which *VirusNet’s* files should be installed, then gets on with installation. What could be easier?

When installation is complete, the user has the option of creating a Program Manager group to launch the *VirusNet* programs. Answering in the affirmative creates icons for the *VirusNet* Scanner, the Spectrum Scanner, the Macro Virus Scanner/Remover, along with notes on the Latest Updates, and help specific to macro viruses. *VirusNet* installed 33 files comprising a mere 4.24MB.

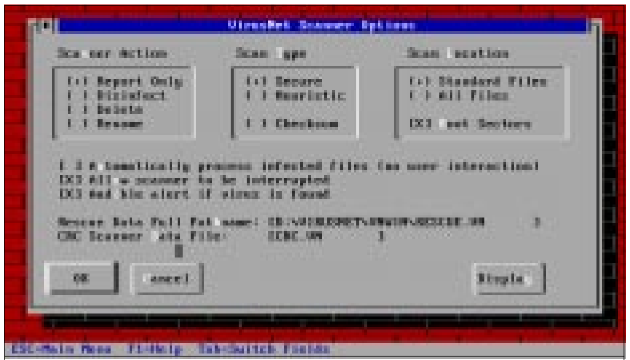
Note that *VirusNet* does not change any system files when it is installed. No memory-resident options are installed, and no amendments are made to *Windows* operation. Therefore for the DOS version of the *VirusNet* scanner to operate, either the DOS PATH must be amended, or the current subdirectory must be switched to the *VirusNet* subdirectory before execution begins.

Scanners

VirusNet provides six main options: Express Scan, Rescue Disk, Spectrum Scanner, Scan Settings, Virus Information, and Help. I have listed these explicitly because in the *Windows* version they are all available as large onscreen areas which light up when the mouse pointer moves over them. One simple mouse click and the selected function activates. All these functions are also available from drop-down menus.

The DOS version of the product offers a similar interface which can be controlled from a special scan selection screen, or from drop-down menus. However, the *Windows* interface is definitely much easier to use: it has a less confused look about it, though I doubt that the New York City skyscraper backdrop has much to do with viruses.

The phrase ‘Spectrum Scanner’ is not a well-established anti-virus concept. It appears to be the *Windows* version of the *VirusNet* scanner coupled with a very flexible scheduler. A Spectrum scan can be programmed to occur at a preset time, or when a preset event has happened – e.g. when *Windows* starts, or after a pre-defined period of system inactivity.



```
                     VirusNet Scanner Options

 Scanner Action          Scan Type            Scan Location
 (•) Report Only         (•) Secure           (•) Standard Files
 ( ) Disinfect           ( ) Heuristic        ( ) All Files
 ( ) Delete
 ( ) Rename              ( ) Checksum         [X] Boot Sectors

 [ ] Automatically process infected files (no user interaction)
 [X] Allow scanner to be interrupted
 [X] Audible alert if virus is found

 Rescue Data Full Pathname: [illegible]                          ]
 CRC Scanner Data File:     [CRC.VN       ]

   OK      Cancel                                     Single

ESC=Main Menu  F1=Help  Tab=Switch Fields
```

VirusNet offers the usual crop of scanning options.

Scanning Speed

In its default state, *VirusNet* for DOS scanned the hard disk of my test PC in 1 minute 11 seconds (1635 files, 275 files, 53.4MB). When the *Windows* version of *VirusNet* was used, this scan time, as expected, rose to 1 minute 36 seconds. This time was the same whether the 'normal' *VirusNet* scanner was invoked, or the 'Spectrum' scanner was used.

The above timings are as reported onscreen by *VirusNet*. As seems normal amongst anti-virus products, the actual scan time is quite a bit larger than the reported time, and certain features are neatly omitted from the quoted time. For instance, when measured with a stopwatch, *VirusNet's* scan time under *Windows* is actually 1 minute 51 seconds, corresponding to an under-reporting of around 10%.

I compared *VirusNet's* scanning times with two competitor products. *Dr Solomon's AVTK* scanned the hard disk of my test PC in 1 minute 1 second, and *Sophos' Sweep* took 1 minute 55 seconds for the same scan. *VirusNet's* scan rate is not really as good as it looks at first glance. Both competitor products were scanning more files than *VirusNet* – 457 and 425 respectively, compared with *VirusNet's* 275. Clearly, *VirusNet* is not including as many file extensions in its scan, but I do not know which types of file have been excluded.

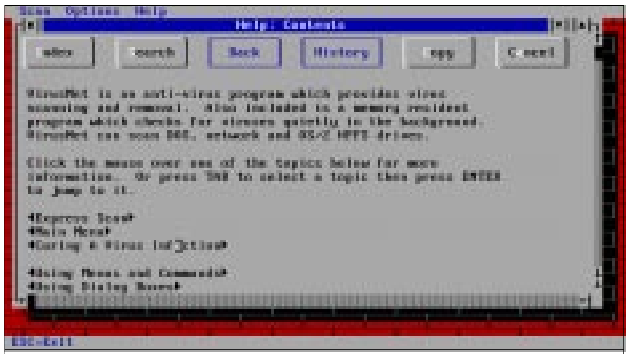
Scanner Problems

I tested the virus detection capability of *VirusNet* against the test-set described in the Technical Details section. However, I did encounter problems trying to test such a large suite of virus-infected files with *VirusNet*. When the complete test-set was scanned, the report file grew so large that *VirusNet* truncated it at about 25KB. I have no idea why.

There were also problems in terminating a scan. Whenever the 'Cancel' button provided by the *Windows* version of *VirusNet* was pressed, the button went grey indicating acceptance of the mouse click, but nothing happened. The scan continued, and a reboot was the only way to stop *VirusNet* in its tracks. The Ctrl-Alt-Del keypress did not offer to terminate *VirusNet* – it only allowed a reboot to be performed (or the scanner to continue).

Further problems occurred at the commencement of a scan using the *Windows* scanner. With the *VB* virus test-set stored on a Magneto-Optical disk, *VirusNet* took almost 4 minutes before it detected a virus. I eventually traced this to the memory scan initiated by *VirusNet* before performing a disk scan. As there is some 12MB of memory available for *Windows*, scanning memory took a *loooong* time.

The final problem was somewhat humorous. In last month's review, I described how I needed to put my mobile phone on the space bar to force a scan to continue to its conclusion. This month I had to put my stapler on the Return key to get past all the messages warning me a virus had been detected. When the scan was complete, *VirusNet* made a noise like a machine gun as it rejected presses of the return key. When



```
VirusNet is an anti-virus program which provides virus
scanning and removal.  Also included is a memory resident
program which checks for viruses quietly in the background.
VirusNet can scan DOS, network and OS/2 HPFS drives.

Click the mouse over one of the topics below for more
information.  Or press TAB to select a topic then press ENTER
to jump to it.

◄Express Scan►
◄Main Menu►
◄Curing a Virus Infection►

◄Using Menus and Commands►
◄Using Dialog Boxes►
```

Windows online help is comprehensive and well-indexed – the DOS version (above) is slightly less detailed, but still good.

the stapler was removed, *VirusNet* stopped making a noise, but as the 'Start Scan' button was then underneath the mouse pointer, it activated the scanner again and the process restarted. This loop continued ad infinitum. I could start a complete disk scan, but I couldn't terminate it! Ah well.

I'm hanged if I'm pressing a mouse key several thousand times, so the detection capabilities were measured using the DOS version of *VirusNet*.

Detection

The file README.TXT that came with *VirusNet* claimed to be able to detect 8300 viruses. The file VIRLIST.TXT, however, claimed to identify '8996 different viruses and also detects viruses belonging to 133 other families', a total of 9129. There are actually 9138 viruses listed in the file VIRLIST.TXT. Make of this variation what you will.

When *VirusNet* was tested against the viruses in the 'In the Wild' test-set, using its default settings it detected all of the 286 test samples. One cannot ask for more than 100%.

Against the viruses in the 'Standard' test-set, again using the default settings, *VirusNet* detected 259 of the 265 samples, missing both samples of the Phantom1 virus, both of Cruncher, and the single samples of Argyle and Peanut. At 97.8%, this is still close to 100% correct detection.

When heuristic detection was introduced, strange things happened. *VirusNet* did detect one more file as infected (Peanut), but removed the Cruncher test samples from the *VirusNet* report file. I have no idea why this should happen.

The *VirusNet* scanner detected nineteen of the twenty boot sector test samples as infected. The only sample not spotted was Peanut, but *VirusNet* did report that 'the DOS boot sector is suspicious'. Fair enough I suppose.

Polymorphic Viruses

Against the polymorphic virus samples, *VirusNet* detected 4410 of the 5500 test samples – a detection rate of 80%. Switching on heuristic detection made only a minimal difference: the number of polymorphic viruses detected rose to 4416 – i.e. still 80%.

VirusNet was 100% perfect at detecting five of the eleven different types of polymorphic virus: Pathogen:SMEG, Groove and Coffee_Shop, Neuroquila.A, One_Half.3544, and SatanBug.5000.A. It failed to detect any DSCE.Demo test samples, and provided variable detection rates for the other five types of polymorphic test samples.

When heuristic scanning was invoked, *VirusNet* simply noted that one test sample of DSCE.Demo, and five samples of Girafe:TPE contained code 'usually only found in a virus'. This accounts for the increase of six in the overall polymorphic virus detection total. One test sample of DSCE.Demo was also described by VirusNet as 'armoured'. Obviously, such a finding is not thought serious enough to qualify for nomination as a fully-fledged virus.

Macro Viruses

The *VirusNet* package includes VNmacro, a *Windows*-only program which is very easy to use, and is able to detect and remove macro viruses. Instructions are provided in the form of a text file, but usage is simply a matter of choosing a subdirectory, selecting either '.DOC' files or all files, and commencing a scan. A report file is created on the hard disk. The on-line help for VNmacro was always 'greyed out' in the drop-down menus.

I tested the detection capability of VNmacro and the other component parts of *VirusNet* against the *Windows*-specific viruses listed in the Technical Details. The DOS and *Windows* versions of *VirusNet* both detected eighteen of the twenty-three macro samples listed in the Technical Details section, missing Doggie, Guess, Polite, Trojan.FormatC, and W2offnen. VNmacro, on the other hand, detected all of the viruses except W2offnen.

Virus Information

One of *VirusNet's* six main options is called Virus Information: this is a database containing details of whether or not each virus known to *VirusNet* is removable, and whether each is a boot sector virus. The database is divided into sections according to the initial letter (or number) of the name of the virus.

A special section called 'Statistics', which contains overall details of the number of viruses *VirusNet* can detect, identify, and/or disinfect, is also provided in the Virus Information. The difference between detect and identify appears to be one of degree. *VirusNet* claims to be able to spot 'even a single-bit change' in the viruses it can identify. I could not test this claim, because the checksumming facilities were disabled!

The Rest

In common with many other scanners, *VirusNet* provides features which disinfect viruses from infected files. However, being (as always) consistent, I do not review such features. That's it, there is no more. As mentioned above, the evaluation version of *VirusNet* provided for review did not incorporate checksumming or Rescue Disk features, so I cannot say whether they work effectively or not.

VirusNet calls itself a 'disaster recovery system': this does not sit comfortably with the fact that no proper backup facilities are provided. All *VirusNet* can do is replace the CMOS memory, and/or boot sector and hard disk partition table. Also, the copy I was sent includes no memory-resident software.

Conclusions

My final thoughts are very straightforward: *VirusNet* contains a scanner which is reasonably quick, and is very good at detecting viruses.

When various parts of the *VirusNet* software are executed, they produce the phrase 'Portions copyright Frisk Software International'. This refers to anti-virus products developed by Fridrik Skulason, former Technical Editor of *VB*. I suppose if you're going to badge a scanner, then it should be a good one. *VirusNet* is indeed very good at scanning.

I'm not sure about the claim made by the developers as quoted in the first paragraph of this review. As I received no memory-resident software, it is hard to support *VirusNet's* claim to be a 'complete anti-virus' system. It is, however, a very good scanner wrapped up in various ways.

The company has now released a new version of the product which contains a background scanner and a macro virus remover. The latest release also comes with an extended manual (approximately eighty pages), and the ability to support *Windows 95* long filenames. Finally, the product now fully supports *Windows NT*.

Technical Details

Product: *VirusNet v2.24c*.

Developer/Vendor: *Safetynet Inc*, 140 Mountain Ave, Springfield, NJ 07081, USA. Tel +1 908 276 9641, fax +1 201 467 1611, BBS +1 201 467 1581, email safety@safe.net.

Availability: Not stated.

Serial number: None visible.

Price: *Windows 95*/*NT* US$90.00; 10-node LAN US$895.00. Site licences available on request. Price includes six months worth of updates; thereafter annual update maintenance is available; apply to the company for these rates.

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

Viruses used for testing purposes:

The macro viruses used in this test are: Atom, Boom, Colors.B, Concept, Concept.fr, Date, Divina, DMV, Doggie, Friendly, Guess, Hot, Imposter, LBYNJ, NOP, Nuclear, Nuclear.B, Pheew, Polite, Wazzu, Trojan.Formatc, XOS, and W2offnen. For a listing of the boot sector viruses see *VB*, March 1996, p.23; for the remainder, see *VB*, January 1996, p.20.

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

PRODUCT REVIEW 2

AVAST32 for Windows NT and Windows 95

Martyn Perry

AVAST32 is the latest product from *Alwil Software* and supports both *Windows 95* and *NT*. Whilst it is essentially a workstation product, it contains limited server functionality; for example, the ability to distribute anti-virus software to attached clients.

Presentation

The licence is granted on a per-PC basis, with no separate provision for home/portable use. The product comes with a single manual covering both *Windows 95* and *NT*.

The copy of the documentation shipped for the evaluation was for an earlier revision of the product and some of the screen shots did not match with the shipping software. This did not cause any difficulties, however.

Installation

As so many other products these days, *AVAST32* makes use of *InstallShield* from *Sterling Technologies* – this product is now the *de facto* standard for software installation. Two types of installation are offered, for the workstation and for administration.

The workstation installation prompts for the name of the *AVAST32* folder, in addition to user details and a serial number (which is supplied with the product). The installation will not proceed until this is entered.

In addition to these settings, the administration installation requires the following to be entered:

- a network folder to store the workstation setup
- the name of the workstation folder in which *AVAST32* will be stored
- any resident programs which are to be loaded from the *Windows 95* Startup group

The folders are now created and the files copied, progress being shown by a vertical level meter showing transfer of individual files as well as a normal progress meter for the overall installation.

AVAST32 for Windows NT

The application starts with the display of the Test Console. This provides an interface to the various available configurations. These include options for virus detection, upgrade handling, and information about the product itself.

The virus library can be viewed from a tab on the main screen. This displays a small amount of information about each individual virus in a *Windows* Explorer-type format. However, the information available is somewhat limited.

Administration

Scanner administration is managed without any special password, and system for both the *Windows NT* and the *Windows 95* versions are very similar. There are three areas of interest under *Windows NT*: the Virus Scanner, the Integrity Checker and the Simple Checker. Under *Windows 95* there are two additional options; Resident Scanner and Behaviour Blocker.

Each of the selections has a number of built-in options which are displayed in the same *Windows* Explorer style as the virus library information. For the main scanner, there is a default configuration, an interactive selection and a diskette scan.

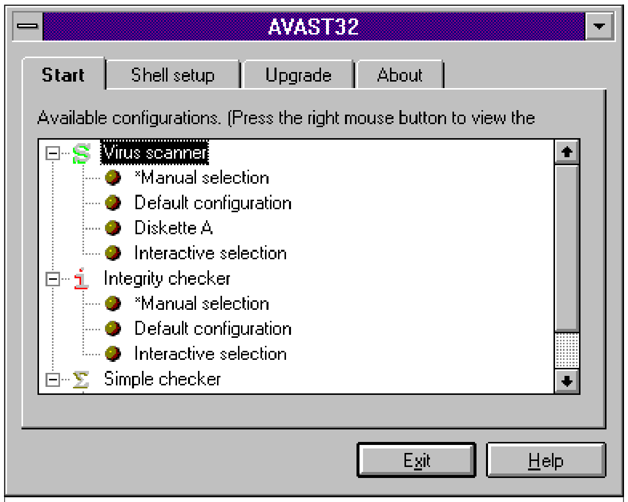
Furthermore, user-defined tests can be set up. Each test configuration needs a name (which can be up to 254 characters long). All characters can be used except asterisk (*), which is reserved for a default name.

Similar facilities are available for the Integrity Checker which is a CRC calculator, and the Simple Checker which provides CRC checking for a defined set of files.

Configuration: Scanning

The virus scanner configuration options include:

- file types to be checked: this can be set to all files, executable with manually selected, and manually selected only



Available configurations. (Press the right mouse button to view the

- Virus scanner
  - *Manual selection
  - Default configuration
  - Diskette A
  - Interactive selection
- Integrity checker
  - *Manual selection
  - Default configuration
  - Interactive selection
- Simple checker

The main *AVAST* panel allows the various components of the product to be executed from a single starting point.

- file extensions considered 'executable': the default extensions are COM, EXE, SYS, BAT, OV?, VPS, 386, BIN, DLL, VXD and DO? (VPS is the extension for the *AVAST32* virus signature files)
- areas to be scanned: memory, all accessible drives, all locally accessible drives (the default), locally accessible hard drives, and an interactive option which allows drive selection to be made at scan time

Other options available include checking hidden or system files and scanning inside compressed files. Finally, options are available to configure what happens when a virus is detected: defining the name of the report file to be used, and selecting a sound file to be played in the event of virus detection. The default WAV file issues a polite warning message that the file is infected!

Surprisingly, there are no options for managing the infected file; i.e. moving it to a quarantine folder or renaming it to prevent execution.

AVAST32 has only one mode of scanning – immediate. There is no scheduled facility. The ability to pause the scan is a useful option, however.

Configuration: Checksumming

The integrity checker uses the same default file types as the scanner. Similarly, the file areas to be checked are the same as the scanner, with the obvious exception of memory. A quick mode can be chosen which checks the content of a file only if attributes such as time, date or length have changed. The checksum databases can be stored either in a defined folder, or in the default location of a file in the root folder on each logical drive.

The simple checker extends the checksum principle to a restricted list of files. The default list consists mainly of the DLL files associated with the scanner. This list can, of course, be edited by the user to include whichever files he wishes.

Configuration: Windows 95

At the present time, both of the resident components are only available for *Windows 95* – the developers will certainly be working on the same functionality under *Windows NT*, however.

The 32-bit resident scanner installs an icon into the Windows 95 tool tray on the Start bar, which can be selected to make changes to the resident component's configuration. The options allow the user to decide what should be checked: boot sectors, executables, and libraries are the available choices.

In addition, this component can check 16-bit *Windows* and *MS-DOS* applications. Although there are options for defining the warning messages, here again no provision is made for infected file management.

The behaviour blocker utility also works in a very similar way – it installs an icon to the tool tray: clicking on this icon provides access to the configuration. Here, the user can configure which areas of the system are to be monitored: DOS file operations, *Windows* file operations, and track formatting. Specific file types can be included for monitoring, and for each file type, a list of individual files can be specified.

Reports, Activity Logs, Communication and Updates

AVAST32 keeps records of scans in a report file with the default name of ScannerReport.rpt. This report lists the files which were scanned, which files were skipped (these are mainly log and event files associated with the scanner, and are skipped because they could not be opened), and infected files. If the user wants to reduce the volume of information stored, options can be selected to ignore skipped files and uninfected files.

"a scheduled scan or even a real-time scan under Windows NT would be a realistic future facility"

Updates are administered from the product's main window – a simple mechanism for installing new signature files is provided.

Detection Rates

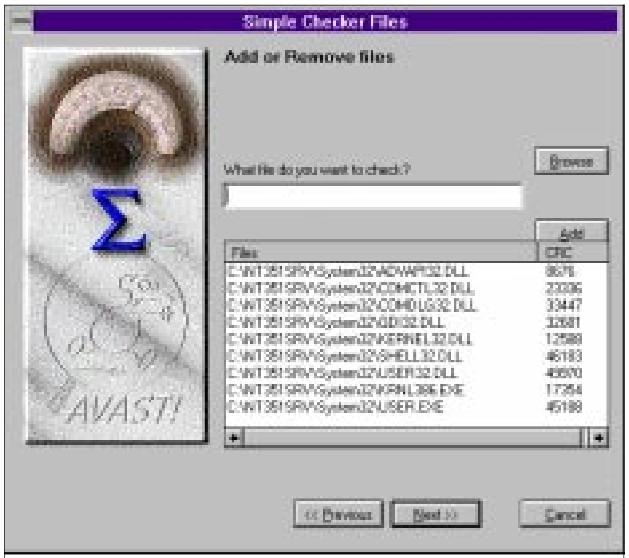
The scanner submitted was Build 356 (dated 3 October 1996), and it was checked using the usual four test-sets: In the Wild Boot Sector, In the Wild File, Standard and Polymorphic (see summary for details). The tests were conducted using the default scanner file extensions supplied with the product.

Since there is no option to remove infected files, the usual method of checking the virus folders for undeleted files could not be used. It was necessary to read the scan report and extract the number of viruses detected.

In the end, however, minimal work was required to determine how the scanner had performed. This was because the scanner delivered an impressive 100% detection rate on all the test-sets listed above – simply outstanding.

Real-time Scanning Overhead

To determine the impact of the scanner on the *Windows NT* server when it is running, 50 EXE and COM files (comprising 6,797,522 bytes) were copied from one folder to another on the server using COPY. The folders which were 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.



Add or Remove files

What file do you want to check?

Files	CRC
C:\WINT351SRV\System32\ADVAPI32.DLL	8676
C:\WINT351SRV\System32\COMCTL32.DLL	23396
C:\WINT351SRV\System32\COMDLG32.DLL	33447
C:\WINT351SRV\System32\GDI32.DLL	32681
C:\WINT351SRV\System32\KERNEL32.DLL	12588
C:\WINT351SRV\System32\SHELL32.DLL	46183
C:\WINT351SRV\System32\USER32.DLL	49870
C:\WINT351SRV\System32\KRNL386.EXE	17354
C:\WINT351SRV\System32\USER.EXE	45188

The simple checker provides CRC checking for a defined set of files, which can be customized.

The default setting of Best Foreground Application Response Time was used, as this is the default configuration for *Windows NT*.

As usual, the time tests were run ten times for each setting and an average taken. The four tests were:

- program not loaded: this establishes the baseline time for copying the files on the server
- program unloaded: this is run after the other tests to check how well the server is returned to its former state
- program loaded, but the immediate scanner not running: this tests the impact of the application in a quiescent state
- program loaded and the immediate scan running: this is the full impact of running the scanner on the server files (see the summary for the detailed results)

The extra load due to the scanner operation was down at a very creditable fifteen percent over the quiescent conditions. Since *AVAST32* performs a clean unload of all the files which were originally installed, there is effectively no residual overhead.

[*This test has much less relevance on a Windows NT system than on a server running NetWare; whereas under NetWare there are plenty of opportunities for support NLMs to be left running, under NT this is simply not the case. Ed.*]

Summary

The product's installation is, as expected these days, very easy indeed. Configuration options were comprehensive on target selection, but were lacking when it came to handling an infected file. Whilst the documentation was slightly out of date, this did not cause problems, and it will surely catch up very shortly; besides, the on-line help covers much of this deficiency.

Putting those minor problems to one side, however, the detection rate was excellent. Combined with the relatively low scan overhead, what the product offers is very impressive.

Compared to other products available, *AVAST32* really does not make the grade in terms of features. To remedy this, the product will need scheduled scan capabilities, better network administration, and an on-access component – the company states that the next version of the product (due for release in January 1997) will include a scheduler and several other features. These facilities, in tandem with the detection rate seen here, will definitely result in a first-class product.

AVAST32

Detection Results

Test-set[1]	Viruses Detected	Score
In the Wild	342/342	100%
Standard	511/511	100%
Polymorphic	10000/10000	100%

Overhead of On-access Scanning on NT:

Tests show time taken to copy 50 EXE files (6.8MB). Each is performed ten times, and an average is taken.

	Time	Overhead
Program not loaded	6.43	-
Program unloaded	6.45	0.3%
Program loaded; no manual scan	6.88	7.0%
Program loaded; manual scan	7.43	15.6%

Technical Details

Product: *AVAST32 for NT, Build 356*.

Developer/Vendor: *ALWIL Software*, Lipi 1244, 19300 Prague 9, Czech Republic. Tel +42 2 685 5961, fax +42 2 685 5624, WWW http//www.anet.cz/alwil/.

Distributor UK: *Lance Premier Services*, Britannia House, 4-24 Britannia Street, London WC1X 9JD, England. Tel +44 171 681 8610, fax +44 171 681 8615.

Price: A one-year software licence with twelve updates downloadable from the company or sent via email. Manual provided as an *MS Word* file downloaded from company. Number of workstations equal to number of ordered licences.

Number of computers	Price per licence
1–10	US$59.00
11–25	US$35.00
26–50	US$29.00
51–100	US$26.00
101–200	US$21.00
201–300	US$18.50
301–400	US$16.00

Prices for larger licences available on request.

Hardware Used:
Server: *Compaq Prolinea 590* with 16MB RAM and 2GB of hard disk, running *Windows NT v3.51*.
Workstation: *Compaq 386/20e* with 4MB RAM and 207MB of hard disk, running *MS-DOS 6.22* and *Windows 95*.

[1]**Test-sets:** In the Wild File, In the Wild Boot Sector, and Polymorphic – see *VB*, October 1996, p.17. Standard – see *VB*, November 1996, p.23.

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
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 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 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 21 January and 13 March 1997 (cost £325 + VAT). Further information is available from Julia Line, Tel +44 1235 544028, fax +44 1235 559935, or access World Wide Web page (http://www.sophos.com/).

McAfee Associates has announced the launch of its '*GroupWare*' anti-virus product line; the first products to ship are *GroupScan* and *GroupShield*, software developed for the *Lotus Notes* environment. ***GroupScan* is designed to protect *Notes* desktops; *GroupShield* to protect *Notes* servers.** Information on these or any of the company's other products is available on its WWW site; http://www.mcafee.com/.

Integralis Ltd has released MIME*sweeper v2.3*. The latest version of this email security product contains functions to enable control of email users, archiving of email contents, and prevention of unauthorized transmission of information. Additionally, the company has released **a *Windows NT*-compatible firewall on the UK market**. For information on these and other products, contact Joanne Simpson at *Integralis*; Tel +44 1734 306060 or email info@integralis.co.uk.

Symantec Corp has inaugurated an anti-virus certification designed to help **ensure virus-free content on the Internet and on intranets**. The program is built to be installed on a Web server, and automatically detects the presence and status of any of the *Symantec* anti-virus products; files are automatically scanned when downloaded, opened, created, modified or run. Information on these and other *Symantec* products is available from the company's WWW site; http://www.symantec.com/.

InfoSecurity 1997 will take place at London England's Olympia 2 from 29 April–1 May 1997. The event is planned to address all aspects of IT security in the business environment, and several anti-virus developers (including *Dr Solomon's*, *Portcullis*, *Second Sight*, *Sophos*, and *Symantec*) have already announced their participation. For further information, contact Yvonne Eskenzi on Tel +44 181 449 8292.

Trend Micro has launched a new product, *InterScan WebProtect*, designed as **virus protection for *Microsoft Proxy Server***, and to block Java applets, *ActiveX* objects, unsigned software, and most incoming software. For information, contact Ken Millard of *Trend Micro Europe*; Tel +39 2 92 111 847, email kenm@trendmicro.it.

Precise Publishing is in action again, after its recently-announced collaborative deal with US-based *SecureNet*. The latest move is the ***Toothbrush*, developed jointly with *EAST*** (IT security arm of *Ethan Adams & Associates*) – a combination of *Precise Publishing's* anti-virus package *Enforcer* with selectable time-out boot passwords facility. Information from *EAST's* Andrew Butterworth; Tel +44 1530 565900.

An integrated security solution for secure remote access has been launched: *Kasten Chase's* OP*tiva Secure+* has been combined with *Racal's SafeDial* encryption modem to facilitate **simple yet secure remote communications**. Information from Allen Mendelsohn; Tel +1 905 238 6900 ext 229, or email amen@mail.kasten.on.ca.

The ideal stocking-stuffer for the researcher who has almost everything: the proceedings of the sixth *VB* conference; price £50 + p&p. To order, contact conference coordinator Alie Hothersall; Tel +44 1235 544034, email alie@virusbtn.com.