ISSN 0956-9979 AUGUST 1996

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **Hold the front page!** Just when you thought it was safe, the macros are back. This month sees the appearance of Laroux, the first *Excel* virus, in the wild. Turn to p.9 for the low-down.

• **Ethics et cetera.** Distribution of viruses has always been a thorny problem, and the ever-increasing growth in the use of the Internet makes this problem more real than ever. Sarah Gordon presents some of her findings and thoughts on p.14.

• **Praying for salvation…** Hare Krishna is not a phrase which one would usually associate with computer viruses: this is a fact which may now have to change. Hare.7610 is a new virus, laden with interesting features… and it's in the wild. See p.11 for an analysis.

CONTENTS

EDITORIAL

What a Wonderful World

Every so often in this business, I get a strange feeling. It always follows a thought of the 'what if…?' type. Some technical details will follow, sometimes of a viral technique, often of flaws in any one of more than thirty anti-virus products, possibly even of a vulnerability in UNIX or *Windows NT*. After a few minutes of mental elaboration, that strange feeling dawns: terrible inevitability.

Take, for example, macro viruses. In August last year, along came Concept. Shortly after this, many possibilities occurred to those involved, most notably the likelihood of other applications being affected by similar creations in the future. One word kept recurring… inevitable.

"after a certain distribution of a given virus has occurred, it will be extremely difficult to eliminate that virus from the wild"

From the massed ranks of these 'other applications', one primary contender emerged: another member of the vastly-popular *Microsoft Office* suite, the most over-featured spreadsheet yet to hit the market (presumably only to be overshadowed by subsequent versions…), *Excel*. Just as *Word* is the common denominator of word processing systems, *Excel* is fast becoming the spreadsheet format of choice. As if that wasn't enough, it incorporates a macro-ing system (*Visual Basic for Applications*, or *VBA*) orders of magnitude more powerful than *Word's* (*WordBasic*). That word again: inevitable.

One year later, it has come to pass. Like an echo of Concept, Laroux raises its head above the parapet of conjecture and into the blinding light of reality; the first (well, the first to be *discovered*) *Excel* virus (see analysis, p.9).

The similarities with Concept are marked. There is no payload. The code contains some curiosities, but it is starkly functional, and by and large works well. It began to circulate in the wild before anyone noticed it, and although at this point the size of the distribution is not known, the fact that it is so firmly in the wild gives it a significant advantage. Not only is it the first of its type, it has also been placed (presumably intentionally, although conceivably by accident) into the wild by its author. We also don't know how long Laroux has been in the wild, which will have at least some bearing on how far it has spread – another significant factor determining the eventual spread of the virus is where it was introduced. Clearly, initially introducing the virus into a large multinational company with thousands of *Excel* users worldwide should result in a much wider spread than uploading one spreadsheet labelled 'Interesting numbers' to a bulletin board in Venezuela.

There is, alas, only one clear distinction here – that between a virus being in the wild, and being only found in laboratories. Once it is out there, the question of how much it is out there is secondary in importance.

It is to be expected that things will now follow the path established by Concept – we will see a number of rushed, and correspondingly careless, copycat attempts, and quick and dirty modifications by other authors; then we will see slicker, better-written follow-ups. However, by this time, anti-virus companies will have rushed around and come up with a range of 'fixes' for the problem. One hopes that they will be able to make faster headway than was possible with the Concept virus, as the basis of the file format of *Excel* spreadsheets is the same OLE format used for *Word* documents, and most major manufacturers have already built parsers for this into their scanners. However, there is still plenty of complexity to go around, as *Excel* has its own data formats hiding underneath the OLE structure. For this it's back to reverse engineering, as *Microsoft* is bound to be as recalcitrant as ever with its information.

If the *Excel*-using community is lucky, defences will be developed before the virus is able to gain the firm foothold in the real world that Concept has managed. The idea of 'critical mass' has a place: after a certain distribution of a given virus has occurred, it will be extremely difficult to eliminate that virus from the wild (by wiping it out amongst the user community), regardless of the effectiveness of the defences introduced, in much the same way as stopping a nuclear reaction becomes next to impossible once the neutron flood is too strong. Perhaps it's not such a wonderful world after all…

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

NEWS

MS Licenses AV... Again

At the beginning of July, *McAfee Associates* announced that it has licensed 'portions of its anti-virus technology' to *Microsoft Corporation* for use in *Microsoft's* Internet software products.

As readers who follow Internet trends will be aware, *Microsoft* is involved in a battle with *Netscape* for domination of the WWW browser market – *Microsoft's Internet Explorer* and *Netscape's Navigator* have been engaged in a ferocious 'features war' for several months now.

Already, anti-virus add-ins to *Navigator* are available (though see End Notes and News for further details). This move by *Microsoft* appears to be designed to remove *Netscape's* advantage in this area. It remains to be seen how *Microsoft* will provide virus information updates – *VB* is mindful of the long drawn-out and thoroughly painful *MSAV* fiasco ▮

Secure Checking?

At the end of June, *Secure Computing* (formerly *Virus News International*) announced the creation of the *Secure Computing Checkmark* scheme. Labelled as 'security product certification', it will apply initially only to anti-virus products, although *Secure Computing* intends to extend the scheme to encompass other security products 'in due course'.

The testing revolves around detecting those viruses in the wild (Joe Wells' *WildList* will be used as the primary source for information as to which viruses are out there). A product must score 100% in the tests to be awarded the *Checkmark*. Once the *Checkmark* has been obtained, the manufacturer is granted the right to use a logo on its marketing material. Additionally, a certificate is issued for the *CheckMarked* product, stating that it has been approved by *Secure Computing*: this may be included in product packaging.

The scheme is superficially similar in nature to the *NCSA* system, but will not, one hopes, be beset by problems to the same extent. The costs are difficult to calculate, but a press release from *Secure Computing* places the cost between £2200 and £9500 per product for the first year, and £1800 and £4500 per product in subsequent years.

Developers already signed up (initial testing for which will take place later this year) include *Command Software*, *DataFellows*, *ESaSS*, *Reflex Magnetics*, *S&S* and *Symantec* ▮

Corrections: In the July 1996 scanner comparative review, *VB* incorrectly listed *PCVP's* version number as 2.23: it should have been 2.33. For the same product, infected floppy scan time was listed as 31, but should have read 0:31; i.e. 31 seconds.

Further, Gregg from *Command Software* points out that FDISK /MBR will not remove Boot.437 from a hard drive – SYS C: is required (where C: is the drive in question).

Prevalence Table – June 1996

Virus	Type	Incidents	Reports
Concept	Macro	67	19.5%
Form.A	Boot	34	9.9%
Parity_Boot	Boot	28	8.1%
AntiEXE	Boot	25	7.3%
NYB	Boot	17	4.9%
Junkie	Boot	14	4.1%
AntiCMOS.A	Boot	12	3.5%
Ripper	Boot	11	3.2%
Empire.Monkey.B	Boot	9	2.6%
Sampo	Boot	9	2.6%
Quandary	Boot	8	2.3%
Imposter	Macro	5	1.5%
Jumper.B	Boot	5	1.5%
Stealth_Boot.C	Boot	5	1.5%
Telefonica	Multi	5	1.5%
Burglar.1150	File	4	1.2%
Bye	Boot	4	1.2%
Empire.Monkey.A	Boot	4	1.2%
Natas.4744	Multi	4	1.2%
Stoned.Angelina	Boot	4	1.2%
WelcomB	Boot	4	1.2%
AntiCMOS.B	Boot	3	0.9%
Feint	Boot	3	0.9%
Manzon	File	3	0.9%
Russian_Flag	File	3	0.9%
She_Has	Boot	3	0.9%
Stat	Boot	3	0.9%
Stoned.Stonehenge	Boot	3	0.9%
V-Sign	Boot	3	0.9%
Barrotes	File	2	0.6%
DieHard	File	2	0.6%
EXEBug	Boot	2	0.6%
Stoned.NoInt	Boot	2	0.6%
TaiPan.438	File	2	0.6%
Tentacle	File	2	0.6%
Wazzu	Macro	2	0.6%
Other [1]		28	8.1%
Total		344	100%

[1] The Prevalence Table includes one report of each of the following: Amoeba, Boot.437, BootEXE.451, Bug70, Cascade, Crazy_Boot, Cruel, Diablo, DMV, Fat_Avenger, Hidenowt.1747, Int40, J&M, Lozinsky.1958 Mongolian, Naughty, Neuroquila, Nomenklatura, One_Half.3544, Screaming_Fist.696, Stealth_Boot.E, Stoned.LZR, Stoned.Michelangelo, Stoned.Spirit, Trojector.1463, Unashamed, Vacsina, WBoot.

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

Type Codes

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

Alho.676

CN: An appending, 676-byte, fast, direct infector. It contains the text: 'CTRL,SHIFT & ALT keys are reserved for internal use' and, at the end of infected files, the string 'Alho'. The virus contains an internal counter: after 20 generations it hooks the interrupt Int 09h and monitors a usage of Ctrl, Shift or Alt keys.

```
Alho.676        8A24 2688 25F3 A406 1F33 D2B8 0925 CD21 C350 1E33 C08E D8F6
```

AOS.863

CER: A stealth, encrypted 863-byte virus which disables VSAFE (the memory-resident component of *Microsoft Anti-Virus*). The virus contains the text: 'M*A*D*#*C*O*W*#*D*I*S*E*A*S*E', and all infected files have their time-stamps set to 6 seconds.

```
AOS.863         5059 BA01 FAB8 4559 92CD 1692 9292 9292 9292 B9AF 01BB ????
```

Blin.1488

ER: An encrypted, polymorphic, 1488-byte virus containing the text: '[Treblinka V 2.01 by Blas Pascal] . Argentina . xx/06/1995 .' All infected files have the string 'BP' located at offset 12h (checksum in EXE header). The following template can be used to detect the virus in memory.

```
Blin.1488       3DCA B075 038B F8CF 3D00 4B74 052E FF2E 4F00 E884 0406 9C60
```

Caco.2965

CER: A stealth, appending, 2965-byte virus containing the plain-text message: 'CACO VIRUS GENE-101. COCO, ALDO, CHINO, OTTO. DOOM-TEAM &CREADORES DE VIRUS&'. All infected files have their time-stamps set to 60 seconds.

```
Caco.2965       33DB B803 FECD 215E 33D2 5681 FB45 4675 232E 3AAC 920B 771F
```

Caco.3310

CER: A stealth, appending, 3310-byte variant of the above virus. All infected files have their time-stamps set to 60 seconds.

```
Caco.3310       33DB B8FF FDCD 215E 33D2 5681 FB41 4C75 232E 3AAC EB0C 771F
```

Critter.1015

CR: An appending, 1015-byte virus containing the text: '[PGa] a critter from DC has infected U ;)' which is visible at the end of all infected files. The virus reinfects already-infected programs.

```
Critter.1015    BA34 1280 FC30 B430 7420 CD21 81FA 1234 B8?? ??74 0E8B D881
```

Deadwin.1228

CER: An encrypted, appending, 1228-byte virus containing the text: 'Dead to Windows!' and 'hard disk destroyed!'. The virus payload (triggering on 13 November, 21 June and any Friday) includes the formatting of disks and screen effects.

```
Deadwin.1228    B948 022E 8B3C F7D7 23FD F7D5 2E21 2C2E 093C F7D5 4646 E2EB
```

Delta.1163

CER: A stealth, encrypted, appending, 1163-byte virus. It contains the text: 'Good bytes from (DEL)ta Virus !!! Reset in 30 seconds !' and 'Brazil - 02/96'. This virus triggers on 4 November: its payload changes the CMOS data, disabling the hard disk and destroying the information on floppy drive types. The virus then reboots the system.

```
Delta.1163      1F0E 07BE 2300 03F5 8BFE B980 043E 8A66 04FC AC32 C4AA E2FA
```

Epsilon.513

EPR: A 513-byte (effective virus length) virus which contains the encrypted string: 'COMMAND' and the plain-text message: '<Epsilon 1.0 (C) 15.3.1995 B.T.Pir8>'. Unlike other companion viruses, it creates COM files that are not marked as hidden and have different lengths (the virus appends to its code a variable number of 'rubbish' bytes).

```
Epsilon.513     3DFC 0C75 04B8 F3F3 CF60 3D00 4B75 03E8 0600 612E FF2E BD02
```

IVP.495

CN: An appending, 495-byte, fast, direct infector. It contains the plain-text message: 'BiATcHSiQB0Y' and 'Hi, my name is Kevin S, and I live in you kompewtor! EyE yEWs 2 bE LeeT, SeW PHeAR mAH! (Fairfax, Va)'. All infected files are marked with the signature 'CA' located at offset 0003h.

```
IVP.495         A5C6 865F 03E9 899E 6003 C786 6203 4341 B905 00E9 0000 5133
```

HLLP.7000

CEN: An appending virus, 7000 bytes long, which contains the text: 'Superviced by Stork Oeba 5/1/95'. Because it was written in high level language, other plain-text messages are visible; e.g. 'Portions

Copyright (c) 1983,90 Borland', 'This program cannot be executed in a Window's shell.', and 'This program requires Windows.'

```
HLLP.7000          0343 4F4D 0345 5845 5589 E5B8 2C04 9A7C 027B 0081 EC2C 04C4
```

IVP.674 **CEN:** An appending, 674-byte, fast, direct infector containing the text: 'Hard Disk Failure Lady Seller' and '*.com *.exe ..'. The virus contains a destructive payload; its trigger is based on the system date and includes formatting disks.

```
IVP.674            CD21 7207 E862 00B4 4FEB F5B4 2ACD 2181 F9CC 0773 09B4 098D
```

Jorgito.730 **ER:** An appending, 730-byte virus. Once a year, on 14 March (beginning in March 1998) the virus displays the usually-encrypted message: 'Jorgitø Was Here Córdoba Argentina'.

```
Jorgito.730        BBD7 F993 CD21 3D83 7874 72BB 4154 438B C305 FE75 CD2F 9380
```

KorWan.1448 **CER:** A prepending (in COM files), appending (EXE files), 1448-byte (COM) and 1518-byte (EXE) virus which contains the text: '[The Wanderer, June 5th,1994 Korea]'. All infected files have their time-stamps set to 62 seconds.

```
KorWan.1448        909C 3D62 F075 0433 C09D CF80 FC11 7503 E92D 0580 FC12 74F8
```

Lazer.1000 **CN:** An encrypted, appending, 1000-byte direct infector infecting one file at a time. Amongst other text, the virus contains: '*.com', 'c:\command.com', '*.*', and '−=≡βL/\ZΣR≡=− (c)'1994'.

```
Lazer.1000         2BCF EB03 90?? ??8A A649 01AC 32C4 EB03 90?? ??AA E2F5 E9AE
```

Nado.584 **CR:** A stealth, appending, 584-byte virus which contains the text: '[RedViper (c) made by TorNado in Denmark '95]'. The virus displays a red flashing cursor. All infected files have their time-stamps set to 58 seconds.

```
Nado.584           B811 74CD 2181 FB56 5274 53B4 4ABB FFFF CD21 83EB 2690 B44A
```

Nado.602 **CR:** A stealth, encrypted, 602-byte virus containing the text: '[Undying Lover v1.01][by WarBlaDE/DC '96]'. All infected files have their time-stamps set to 58 seconds. The following is the longest possible template which can detect all infected files.

```
Nado.602           3114 4646 E2FA C33E 8B96 3A02 8DB6 1200 B910 01EB EB
```

Nado.759 **CR:** A stealth, encrypted, 759-byte virus containing the text: '[CyberBug v. 1.00][made by TorNado DK]Cyberbug.bat'. The virus creates a file 'cyberbug.bat' containing only one line: 'echo > clock$'. Executing such a file destroys current system date and time values and usually crashes the system. All infected files have their time-stamps set to 2 seconds.

```
Nado.759           E800 00CD 01E8 1600 E800 005D 81ED 0E01 E8CE 02E8 4502 E80D
```

Oktubre.1784 **CER:** A stealth, encrypted, 1784-byte virus containing the text: 'Feliz aniversario Digital Anarchy!!', 'CHKLIST.MS', 'ANTI-VIR.DAT' and 'Virus OKTUBRE Ver. 1.0a By Bugs Bunny [DAN] (c) 26/12/94 Digital Anarchy'. All infected files have their time-stamps set to 40 seconds. On 6 October, the virus overwrites the contents of the first physical hard disk.

```
Oktubre.1784       E800 00B4 FF05 5DF8 72FC 81ED 0A00 1E06 0E0E 1F07 B9A3 068D
```

Pindonga.2072 **CER:** An encrypted, slightly polymorphic, 2072-byte virus which contains several destructive payloads, including: corrupting CMOS data, overwriting the hard disk, refusing to execute programs from under *Windows*. On 16 and 18 September, the virus may also display the text: 'PINDONGA Virus V1.4. (Hecho en ARGENTINA) Programado por: OTTO (16/9/77) Saludos a: MAQ-MARIANO-SERGIO-ERNESTRO-COSTRA PD: Alguien mate a Bill Gates (El WINDOWS SE CUELGA)'. No simple template for detecting all infected files exists; the following string detects the virus in memory.

```
Pindonga.2072      B403 B102 50CD 1358 FEC6 3A36 0009 7F07 80FD 1074 08EB E932
```

Shoe.1904 **ER:** A stealth, encrypted, slightly polymorphic, 1904-byte virus armoured with some anti-debugging procedures. It contains a destructive payload. On 1 January, the virus may overwrite first 112 sectors of a hard disk and display the message: 'OOPS .. Sorry For help call now: 555-SHOE or 555-RGNE No rights reserved by M.WEINHOLD'. No simple template to detect all infected files exists; the following string can be used to find the virus in memory.

```
Shoe.1904          3DCE FA75 07B8 AFEC 9DCA 0200 9D9C 2EFF 1E7A 07CA 0200 9C2E
```

Ups.1155 **CN:** An encrypted, appending, 1155-byte, fast direct infector containing the text: '!\ oH iTs X-MAS /', '*.COM' and '\ThE_UpS-IsT_HiEr/'. From time to time, the virus displays the graphic image of a skull.

```
Ups.1155           8B94 0801 89F3 81C3 4501 B93E 0431 1743 E2FB 5BC3 5E81 EE06
```

V.514 **CR:** An appending, 514-byte virus containing the text: '*.COM' and '????????COM'. All infected files are marked with the byte 0AAh located at the end of the file.

```
V.514              B900 04F3 A406 1FBA F101 B821 25CD 210E 1F89 EBC3 3D00 4B74
```

V.699 **CER:** A prepending, 699-byte virus containing the encrypted text: '7.11.V3b'. It corrupts some infected files.

```
V.699              B8FD FFCD 2181 FB11 0775 298C 060C 00C7 060A 00B9 00B4 4CCD
```

V.768 **CN:** A prepending, 768-byte direct infector, which infects one file at a time. It contains the text: '*.com' and does not infect COMMAND.COM. All infected files have their time-stamps set to 62 seconds. The

virus payload includes a procedure which overwrites the DOS Boot Sector of the current disk.

```
V.768           B9B8 0289 0E90 00A3 9200 A104 002D 8704 7414 2B06 4C00 3D3B
```

V.1097 **EN:** An appending, 1097-byte direct infector. It contains a payload that includes deleting files with the extension 'zip'. The plain-text ASCII string 'Frvrmfsmvu2/-)v_Hitmru07Vsmsfs$' is visible at the end of infected files.

```
V.1097          E82A FF80 FCFF 740A BAA7 0403 D5B4 41CD 21C3 B4FF C3B8 0042
```

VCC.339 **CN:** An encrypted, appending, 399-byte, fast direct infector containing the text: 'Marvin the paranoid android'. The payload, which triggers randomly, installs a new Int 21h, which truncates the length of every file loaded for execution to 0 bytes. Infected files have their time-stamps set to 4 seconds.

```
VCC.339         CAE8 1600 EB26 E811 008D 9603 01B9 5301 B440 CD21 E803 00C3
```

VCC.581 **CR:** An encrypted, appending, 581-byte virus which contains the text 'Mary Reilly'. The virus does not infect files which have their time-stamps set to any of the following seconds values: 36, 38, 44, 46, 52, 54, 60 and 62.

```
VCC.581         4503 8DBE 3E01 BA01 0047 47EB 0590 B44C CD21 B40B CD21 E2F1
```

VCC.613 **CR:** An encrypted, appending, 613-byte virus containing the text 'The Grim Reaper'. It does not infect files which have their time-stamps set to one of following values: 56, 58, 60 and 62 seconds.

```
VCC.613         E5F7 1581 059E 16F7 1547 47EB 0590 B44C CD21 B40B CD21 E2E5
```

VCC.784 **CR:** A stealth, encrypted, appending, 784-byte virus containing the text: '*() Mary Mallon = Typhoid Mary)(*'. All infected files have their time-stamps set to 60 or 62 seconds, but the stealth routine ignores the latter.

```
VCC.784         35EC 8FFE 05F7 15F7 1547 47EB 0590 B44C CD21 B40B CD21 E2BA
```

Voyager.1134 **CN:** An appending, 1134-byte direct infector which does not infect programs 'WI*.*' and 'CO*.*' (e.g. win.com, command.com). The virus contains the text: '*.*', '*.vom' and 'Voyager (.com) is here'.

```
Voyager.1134    80BE 4104 E975 0780 BE44 0421 7413 80BE 4204 5A75 0780 BE41
```

FEATURE 1

Generic Decryption Scanners: The Problems

Carey Nachenberg, Alex Haddox

Anti-virus researchers strive to design their virus scanners to be as general as possible, so that the largest number of viruses can be detected without significant and continuing modifications to the engine itself. This strategy reduces the number of required changes to the anti-virus program, and diminishes the need for regression testing and frequent, expensive upgrade shipments.

This has led to a largely data-based solution to the anti-virus problem. It is quite economical to post a non-executable data file publicly, for clients to retrieve at their leisure. Furthermore, software developers need not worry about software piracy since this data file is useless without the executable portion of the anti-virus program.

Unfortunately, the very nature of computer viruses makes it impossible to design an anti-virus system that can detect current and future viruses without executable updates. New viruses are being developed constantly, and growing numbers use detection-resistant techniques to thwart existing anti-virus algorithms.

Often, anti-virus researchers develop specialized detection routines to deal with these exceptional viruses. However, when enough of these viruses exist, they invalidate the current detection paradigm, and force the development of an entirely new technology. Consequently, the anti-virus software of today is a patchwork of many detection schemes and engines.

Virus writers have already forced many shifts in anti-virus technology. For instance, when anti-virus programs first developed the capability to detect unchanging viruses, the virus authors reacted by developing polymorphic viruses. To detect these polymorphic viruses, anti-virus researchers developed the CPU emulator-based Generic Decryption (GD) scheme.

Now, with the increasingly widespread use of such emulator technology, it is only a matter of time before the virus authors design insidious new viruses to invalidate the CPU emulation technique.

What is Generic Decryption?

Current polymorphic viruses contain at least a small body of machine language instructions and data which is copied verbatim from infection to infection. For the polymorphic virus to avoid detection, this static portion of the virus is

encrypted within infected files. When a program infected with a polymorphic virus is launched by a user, the virus takes control, and launches its decryption routine to decrypt the static portion of the virus. Once this routine finishes decrypting the virus body, it transfers control to the body so the virus can replicate.

The GD scanner relies on this behaviour to detect polymorphic viruses. Each time the GD anti-virus program scans a new executable file, it loads it into a 'virtual computer' (i.e. a simulation of a PC). The program is then allowed to execute in this virtual computer as if it were running on a real machine.

During execution, if the target file is infected with a virus, it can cause no damage to the actual computer, because it executes in a completely contained, virtual environment.

If the GD scanner emulates a program infected by a polymorphic virus, the virus executes its decryption routine. This routine proceeds to decrypt the static portion of the virus within the virtual computer.

As the virus executes, the Generic Decryption anti-virus scanner monitors the progress of its execution. When the virus has decrypted enough of itself, the anti-virus scanner examines these decrypted regions and identifies the strain of the virus exactly.

"the goal of the GD scanner is to emulate as few instructions as possible, while still detecting all infectious virus samples"

The Generic Decryption scanner identifies the virus by searching for specific sequences of bytes which are certain to be present in the static (previously encrypted) portion of the virus. Of course, like other virus scanning technology, the GD scheme requires anti-virus researchers to analyse the virus, extract a virus signature and insert the signature into the scanner database.

In essence, this process is like injecting a mouse with a serum which may or may not contain a virus, and then observing the mouse for adverse effects. If the mouse becomes ill (that is, if the virus manifests itself), researchers can observe the visible symptoms, match them with known symptoms, and identify the virus. If the mouse remains healthy, researchers can select another vial of serum and repeat the process.

Generic Decryption systems provide accurate identification of polymorphic viruses and reduce dramatically the possibility of false identification or misidentification. Such extreme accuracy is possible because the scanner examines the unchanging virus body instead of the ever-changing virus decryption routine.

However, Generic Decryption anti-virus systems are not perfect: there are many ways in which viruses can and do avoid detection by GD-based scanners. The following sections describe several viruses, existing and theoretical, and discuss how they avoid detection by GD scanners.

GD-resistant Viruses

Most polymorphic viruses decrypt and transfer control to their virus body deterministically: a given infection will always decrypt and transfer control to the virus body in exactly the same manner.

As a result, if the viral sample is emulated long enough, the static body will be decrypted and executed, making GD detection possible. However, viruses do not necessarily need to gain control of the computer every time an infected sample is executed.

Consider a virus that uses polymorphic code to fetch a byte from an actively changing area of memory, such as the DOS disk buffers:

- if the value of this byte is between a certain range, then the polymorphic code continues decryption and executes the virus body
- if the value of this byte is outside the required range, the polymorphic code repairs the host program in memory and transfers control to the host program
- every time the virus infects a new file, the location from which the byte is fetched and the required range is randomly changed

This virus might gain control of the machine once in every ten executions of an infected program; however, such a program could still be quite infectious. Unfortunately, the GD scanner is simply unable to detect such a virus reliably.

The GD would emulate the infected sample until it reached the random memory test. If the emulator's virtual memory happened to contain the appropriate value in the proper memory location, the polymorphic code would continue decrypting the virus, and the sample would be detected. If the emulator's virtual memory contained a different value, however, the virus would fail to decrypt itself and the GD scanner would fail to detect the virus.

Given the number of possible memory states (well over $2^{8388608}$ for a simple 1MB PC), it is impossible to guarantee that such a virus infection would always find what it wants in the computer's memory and decrypt itself properly. The Commander_Bomber virus unknowingly employs a similar technique, making reliable GD-based detection impossible.

There exists yet another technique which thwarts GD scanners completely. Generic Decryption requires that the virus gains control and decrypts itself as soon as the host program begins executing. Why? The GD scanner must decide how long to emulate each program before it stops to report that the file is uninfected.

The goal of the GD scanner is to emulate as few instructions as possible, while still detecting all infectious virus samples. To reduce the amount of time spent emulating programs, current GD schemes emulate the suspect program and examine the instructions used by the program in an attempt to determine whether the instructions look like those used by a polymorphic virus.

If the instructions look suspicious, the GD scanner continues emulating the host in an attempt to get the (potential) virus to decrypt itself. If the instructions look like those of a 'normal' program, the GD scanner assumes the program is uninfected and ceases emulation.

Several existing viruses (such as Positron – see *VB*, February 1996, p.8) infect executable files so the virus receives control only after the host program has executed a number of its own instructions.

Thus, when an infected program is launched, the virus may or may not gain control, depending on the nature of the infection. Even if the virus does receive control, it is most likely to do so after one or more instructions of the host program have been executed.

"the virus-writing community is fully aware of GD's inherent weaknesses: it is only a matter of time before viruses which exploit these are constructed"

When scanning such an infected file, a GD scanner would initially emulate the instructions of the host program rather than those of the virus. Consequently, the GD scanner would in many cases recognize these instructions as non-viral and cease emulation almost immediately. The emulator would not emulate the file long enough to reach the virus, hence the virus would fail to decrypt itself, and the file would be reported clean.

The emulator could be set always to emulate many thousands or millions of instructions before reporting that a program is uninfected. However, even with this Draconian modification, there can be no guarantee that the emulator would emulate the host long enough to reach the virus decryption routine.

In fact, there is no guarantee that the emulator would ever execute the instructions of the virus decryption routine, even if the emulation went on indefinitely!

Imagine a program that merely waits for a key-press from the user and then terminates. This program might be infected by a virus such that the virus is given control just before the program terminates. In a typical interactive environment, such a virus would launch every time the infected program was executed by the user.

However, given that the virtual machines used in GD scanners are non-interactive, the program could execute endlessly in the virtual machine, awaiting a key-press from a non-existent user. As the program would never receive a key-press and terminate, the virus would never have a chance to execute and decrypt itself.

GD-pesky Viruses

It is a difficult task to create a fully-compatible CPU emulator. Even a simple flaw that differentiates a CPU emulator from a real machine can be located and targeted by a virus writer.

Even the 80x86 line of computers is not completely backwards compatible. Every processor is slightly different from its predecessors. For example, the pre-fetch queue on the 80x386 chip is sixteen bytes long, but for the 486, the pre-fetch queue was expanded to 32 bytes, in order to increase performance.

Consider an anti-virus product that uses the GD technology with an 80486-compatible CPU emulator. This emulator would be unable to execute properly a virus that employs polymorphic code designed to exploit the sixteen-byte pre-fetch queue of the 80386 processor.

Although it is true that this virus would also fail to execute on real 80486 machines, it might flourish on the large base of 80386 machines. Unless the Generic Decryption implementation applies several different emulators on each file, it will fail to detect this virus. Such a solution is impractical; even if it were implemented correctly, it would increase scanning time significantly.

Conclusions

The Generic Decryption scanning technique has so far proved to be the single most effective method of detecting polymorphic viruses. It allows anti-virus researchers to spend less time analysing specific polymorphic viruses, improves scanner performance, and reduces false positives.

Despite these benefits, GD technology still has significant problems. Many different classes of polymorphic viruses simply cannot be detected reliably. Currently, there are a limited number of polymorphic viruses which employ such anti-detection schemes.

However, the virus-writing community is fully aware of GD's inherent weaknesses: it is only a matter of time before viruses which exploit these are constructed. For this reason, the anti-virus community must remain ever-vigilant and never satisfied with current technology and implementation.

The authors of this article are both anti-virus specialists at *Symantec Corp*. They can be contacted as follows:

Carey Nachenberg: cnachenberg@symantec.com
Alex Haddox: ahaddox@symantec.com

VIRUS ANALYSIS 1

Excel Yourself!

Sarah Gordon

The number of macro viruses appears to increase on a weekly basis, although every new batch seems remarkably similar to the last. However, the most recent macro virus I have encountered requires special treatment: first, it was discovered in the wild; second, it infects *Excel* spreadsheets, not *Word* documents. Do I have your attention yet?

Just as we knew that many vulnerabilities existed in macro languages long before Winword.Concept reared its ugly (but persistent) head, we knew it was only a matter of time before an *Excel* virus appeared in the wild.

Excel spreadsheets have, in much the same way as *Word* documents, the potential to carry code which represents executable instructions in the *Excel* environment. And, also like *Word*, *Excel* does an excellent job of handling these macros; it usually does so flawlessly, without drawing much attention to itself.

Fortunately, Laroux is less than elegant in its design, and *Excel* is liable to notify the user under certain circumstances that the virus' copy routine has failed.

This otherwise unremarkable virus, ExcelMacro.Laroux, differs from its *Word*-based cousins in that it is written using *Visual Basic for Applications* (*VBA*). [*This is a much more powerful macro-ing language than that present in current versions of Word. Ed.*] Clearly, therefore, *Word* users need not concern themselves with this particular virus, just as *Excel* users need not worry about Concept.

The virus carries no deliberately destructive payload, and is only slightly more devious than the first *Word* macro viruses. It uses simple techniques to replicate and hide. Simple… but very effective.

Hide and Seek

As with all new viruses, I began testing cautiously, and opened the infected file I had been sent from a write-protected disk. Immediately, a design flaw in the virus became apparent. A dialog box titled 'Macro Error' popped up on my screen, telling me that a copy had failed, leading me to hope that the virus would prove ineffectual. This hope was misplaced.

[*This box is only displayed when the current drive is write-protected: if the user opens the file by typing 'A:\INFO.XLS', no error occurs. However, if he navigates to drive A, and selects INFO.XLS, the warning is displayed. Ed.*]

Examining the virus proved easy – it could be done either by using Window/Unhide to reveal the hidden Worksheet, then selecting it from the tab display, or by using Tools/Macro to select a macro to edit. Either method could easily be subverted by future such viruses to trigger the virus, just as with Tools/Macro under *Word*.

The first line of the virus code will be familiar to those who have examined *Word* viruses: 'Sub auto_open()'. Although the syntax is slightly different from the *Word* equivalent, the purpose is the same: an Auto_Open macro (*VBA* is not case sensitive) is invoked whenever a spreadsheet is opened.

This particular Auto_Open macro is very simple: it inserts a call to the second virus macro, called check_files, which is executed whenever a new Worksheet is activated. This is the virus' only other macro, and does most of the work.

Check it Out...

When activated, the check_files macro first obtains *Excel*'s start-up path; this was C:\MSOFFICE\EXCEL\XLSTART on my test computer, but will vary depending on your installation. It then looks for the file PERSONAL.XLS in this directory. (Note: users should not assume that the absence of this file indicates their systems are virus-free.)

> *"if the macros 'auto_open' and 'check_files' exist, you are likely to be infected"*

This .XLS file is akin to *Word's* NORMAL.DOT. The *MS Excel/Visual Basic for Windows 95 Programmers Guide* says:

In Microsoft Excel Version 7 you can still record your macros in a workbook that opens each time you start Microsoft Excel … this workbook is now called 'PERSONAL.XLS' or 'Personal Macro Workbook', depending on the platform (Windows or the Macintosh) ... Microsoft Excel Version 7.0 creates your new Personal Macro Workbook when you record your first macro.

Due to the way Laroux searches for the PERSONAL.XLS file, I suspect it will not replicate on *Macintosh* versions of *Excel*, although no machine was available to test this theory. [*The virus is written in VBA; thus it will also not work on versions of Excel earlier than 5.0. Ed.*]

Laroux next examines the number of Modules (*Excel*-speak for Workbook sheets that contain *VBA* code) in the currently active Workbook (referred to as 'ActiveWorkbook'). There are four possible cases:

- no PERSONAL.XLS file; the ActiveWorkbook contains no Modules. Under normal circumstances, this should not occur: if there is no PERSONAL.XLS, the virus has not infected the host machine, or an error has occurred. Given that there is no PERSONAL.XLS, the virus

should be running from a macro in the Active Workbook, which would then have to contain at least one Module.

- no PERSONAL.XLS file; the ActiveWorkbook contains Modules
- PERSONAL.XLS file present; the ActiveWorkbook contains no Modules
- PERSONAL.XLS file present; the Active Workbook contains Modules

If the first or last conditions are met, the virus will abort; this serves as an infection check. On an uninfected computer, the second condition is met, and will therefore be considered first.

If the machine does not have a PERSONAL.XLS file, it is not yet infected. It proceeds to unhide the virus Module (titled 'laroux'), and copy it into the PERSONAL.XLS file. It sets some fields in the file properties to empty strings: Title, Subject, Author, Keywords and Comments (why it does this is not clear). This done, the virus cleans up, and infection is complete.

Now, when the user opens an *Excel* spreadsheet, the virus will be activated (the third case). If PERSONAL.XLS exists, and the current ActiveWorkbook contains no Modules, then the virus knows that PERSONAL.XLS is already infected (as the macro is running from there), and it should now infect the active workbook (i.e. the one just opened).

When running from PERSONAL.XLS, the virus watches for new spreadsheets using an 'OnSheetActivate' event. This is more powerful than an Auto_Open, as it is triggered whenever a Worksheet becomes active (i.e. whenever the user clicks on a tab to view a different Sheet within a Workbook). Such routines offer both the macro programmer and the virus writer great flexibility.

Like *Word* viruses, Laroux infects the target by copying its macros there. Unlike *Word* macro viruses, this does not require the alteration of the file's type, but merely the addition of new Modules – in this case, a hidden Worksheet located at the beginning of the workbook.

Once this extra Worksheet has been created, it is tagged as 'hidden', and the user will be completely unaware that anything is amiss. However, there are some cases when this copy may fail, and the virus does not trap these errors. Under such circumstances, the virus will display the error box described above.

Detection and Removal

Determining whether or not your copy of *Excel* is infected is simple. Start the program and select the 'Macro…' option under the 'Tools' menu. If the macros 'auto_open' and 'check_files' exist, you are likely to be infected.

As a second check, select one of these macros and click the 'Edit' button (if the system states that you 'cannot edit a macro on a hidden workbook', unhide the workbook by using the Window/Unhide command). You should see the macros, and a Worksheet entitled 'laroux' should also be visible. Keep in mind that taking all these actions is only valid for this particular virus. Other viruses could render this method useless.

The hex pattern which is given below may be used in conjunction with an anti-virus program to locate the virus, but users should remember to add .XL? to the file extension list. If you suspect that you have this virus, you must check all files, as your users may not always be using the default file extensions.

Simply removing the macros from PERSONAL.XLS and all infected Workbooks clears the virus, although users should remember to remove the 'laroux' Sheet at the beginning of every Workbook. Clearly, both of the detection and removal methods mentioned here are short-term measures: it is to be hoped that anti-virus vendors will implement more efficient fixes shortly.

The Solution

Almost a year ago, when Winword.Concept appeared [*see VB, September 1995, p.8*], I wrote: 'The techniques used by this virus are so simple that any idiot could use them to construct similar viruses. If history is any indicator, we can expect to see more of this type of virus.'

We did see more. We now see that *Excel* viruses are just as trivial; it is safe to assume that there will also be more of them. They will probably be equally unremarkable.

The ease with which these macro viruses can be created and modified means that long-term solutions need to be found soon to the whole threat, rather than to individual instances.

This problem is firmly in the domain of the application developer – they should also keep an eye on the possible misuse of all this extra functionality. It is becoming more and more important as macro virus production increases. Please, designers, pay attention!

ExcelMacro.Laroux

Aliases: None known.

Infection: *MS Excel* spreadsheets, v5.0 or greater.

Self-recognition in *Excel* Spreadsheets:

Searches for a Worksheet named 'laroux'.

Hex Pattern:

```
0021 0060 0027 206A 0020 206A
00AD 0001 005C 0011
```

Trigger: None.

Removal: See text.

VIRUS ANALYSIS 2

Hare Krsna: ISKCON too far!

Ian Whalley

At the end of May, I received a virus sample. In itself, this is not unusual; however, it was not detected by any anti-virus product which the (very computer-savvy) user had, or could obtain. The sample was unpacked, and analysis started.

A swift look at the code revealed that the file was definitely unusual, and very probably infected. Disassembly was complicated by the virus' anti-debugging and -emulation techniques. Once these had been dealt with, the virus was taken apart without too much effort.

Overview

Hare is a multi-partite virus (Master Boot Sector of hard drives, floppy boot sector, COM and EXE files). It is a slow polymorphic (see below for a detailed description), and contains anti-debugging routines. It is encrypted in both files and boot sectors (viruses which encrypt themselves in the boot sector are becoming increasingly common).

The code of this sample, at 7610 bytes, is by no means the longest seen, but certainly makes it the largest non-*Windows* virus in the wild at the moment. This alone provided warning of the effort that would be involved in disassembly.

Functionality

Hare's code is, to say the least, tangled and complex. It uses many interesting techniques, including one which, in my opinion, is extremely dangerous, and could be used in the future by other viruses to greater effect. More of this later.

When the virus receives control from an infected program, it decrypts itself in memory (this involves executing three separate decryptors). Whilst doing this, it issues an Int 21h, AX=FE23h: if AX=000Dh is returned, this part of the virus is present in memory, and installation aborts.

If the virus handler is not present, Hare completes decryption, and installs itself at the end of the MCB chain (using the standard technique of walking the list looking for the Z block). It then passes control to the new resident copy.

Next, the resident component determines whether *Windows* is running, using Int 2Fh, AX=160Ah (Identify *Windows* Version and Type). If enhanced-mode *Windows* is present (including *Windows 95*), it notes the fact for future reference.

The virus then checks a sector on the track one *beyond* the end of the hard disk. This track is sometimes called the Landing Zone, Engineering Cylinder or Test Cylinder, although these terms are somewhat old-fashioned. If this starts with the identifier CCDDh, it is left alone; otherwise, the virus attempts to write a single sector of random data. The routine is flawed, however, and instead of filling a 512-byte buffer in memory with random bytes, it repeatedly places random data into the first word, which is then overwritten with the CCDDh marker anyway!

The data is used by the polymorphic encryption routines to create the header for new instances of the virus; consequently, replicants of Hare created on one PC will have very similar encryption loops. Because of the bug, the effect is not quite what the virus author had intended, but the polymorphic loops will still vary.

This polymorphic technique can foil anti-virus researchers: detectors and removers they create from one infected PC are likely to be incomplete. When an infected program or disk is taken to a clean PC, the random data which Hare writes to the sector will be different, and that PC will create samples of the virus which will look different from those samples which were seen before.

> *"attempts to remove the virus with 'FDISK /MBR' will render the disk unbootable"*

The routine that writes the random data is flawed – when setting the sector number to one (to write to the first sector on the extra track), the virus trashes the top two bits of the track number (which is stored at the top of CL to allow 10-bit track values).

If the disk in question has more than 256 tracks (very likely these days), the sector of random data will be placed somewhere in the middle of the disk, possibly over user data.

Hare then tests to see if its boot sector component is already resident, by issuing Int 13h, AX=5445h. If AX=4554h is returned, it assumes that it is. If it is not resident, it checks the MBR to see if it is already infected, and infects it if not.

MBR Infection

Whilst infecting the MBR, Hare introduces several interesting techniques. It attempts to use port-level access to the hard drive (to avoid BIOS-level boot sector protection).

If it cannot access the hardware directly, it traces Int 13h. It hooks Int 16h (Keyboard) before writing to the disk using Int 13h – it looks as if it is attempting to replace replies to BIOS questions (such as 'A program is about to write to the MBR. Do you wish this write to proceed?') with ones which allow the write to occur. It has not been possible, however, to verify this.

Hare does not leave the Partition Table intact in the infected MBR, so attempts to remove the virus with 'FDISK /MBR' will render the disk unbootable. Later, it must perform complex gyrations (see p.13; 'Loading from an Infected Boot Sector') to account for this.

It is worth noting that Hare correctly uses the *Windows 95* call Int 21h, AX=3513h, CX=084Bh to lock the disk before attempting direct writes. If this is not carried out, *Windows 95* will reject the write. The volume does not appear to be unlocked, but in normal use this should cause no ill effects.

Hare now directly modifies the IVT to revector Int 21h to its own handler – this will enable infection and stealth, of which more later. Next, it checks to see if *Windows 95* is currently running (Int 2Fh, AX=160Ah; returns BH=04h if *Windows 95* is active): if so, it installs its Int 13h hook. Interestingly, it only does this in the presence of *Windows 95*. After performing the actions described under 'Deletion of system file', it returns control to the host program, which is allowed to execute normally.

Deletion of System File

Next comes Hare's most interesting feature. It searches the *MS-DOS* environment data area for an environment variable starting 'WI', which will match either WINDOWS or WINBOOTDIR: these point to the main *Windows* directory on *Windows 95* systems. When this is located, Hare takes the value, appends to it the string '\SYSTEM\IOSUBSYS\HDFLOP.PDR' (thus obtaining a complete path to the file HDFLOP.PDR), and calls the original Int 21h handler to delete it (Int 21h, AH=41h).

Why does it do this? Documentation on the area is limited, but the file HDFLOP.PDR contains the *Windows 95* port-level driver for floppy disk drives. Readers familiar with previous discussions on the impact of viruses on *Windows 95* will be aware that this OS does not normally propagate boot sector infections: it uses direct access to floppy disks, so Int 13h hooks installed by a virus to monitor and infect floppy disks are never called.

Unfortunately, to be able to do port-level access in this way, *Windows 95* requires the file HDFLOP.PDR. If this is not present, the system uses old-style Int 13h access to floppy disks. This is a problem, as now any Int 13h handlers will be triggered, and infection can take place as before.

Worse, *Windows 95* does not warn the user of this scenario: browsing through the contents of the System applet in Control Panel does reveal that the system is not running at peak performance, but normal users do not look here every day.

Thus, after the next reboot, Hare will be able to infect the boot sectors of floppy disks. Better yet, if the virus is removed, this driver file is still missing, and any subsequent boot virus infection will be able to infect floppy disks in the same way.

In Memory: Int 21h

The Int 21h handler intercepts the functions FE23h (Are You There?), 36h (Get Disk Free Space), 4Ch (Exit), 31h (TSR), 00h (Terminate), 4B00h (Load and Exec), 11h/12h (Find First/Next by FCB), 4Eh/4Fh (Find First/Next by Name), 3Dh (Open Existing File) and 3Eh (Close File).

The Get Disk Free Space handler is rather peculiar – when this function is called, the virus checks the address of the calling process's PSP. If it is different from that of the last process which called this function, it performs a genuine Get Disk Free Space call via the original Int 21h handler, saves the number of free clusters, and returns the values unchanged. If it is the same, it still performs the genuine call, but replaces the value for the number of free clusters with the saved one, and returns to the caller.

The reasons for this are not obvious – a couple of possible explanations for this have been suggested. Firstly, one particular anti-virus product, *InVircible*, periodically calls Int 21h, AH=36h to see if the amount of free disk space is dropping. If it detects a drop, it warns that a fast-infecting virus may be in memory. Hare's technique of returning the same value every time the process asks will foil this.

"if the virus is removed … any subsequent boot virus infection will be able to infect floppy disks in the same way"

The second possible explanation is that Hare is again attempting to fool anti-virus researchers. An oft-used technique for replicating viruses is to place the virus in memory, do a DIR to note the lengths of the goat files and amount of free disk space, run the goat files, and then do another DIR.

Even if the virus has file length stealth, the change in the amount of free disk space will reveal if the virus has infected anything. Hare will not show any change, however, as each DIR command will return the same value for the free space (each call is issued by COMMAND.COM).

Exit, TSR, and Terminate calls are dealt with in the same way: the name of the currently executing program is extracted from the PSP, and that file is opened, infected, and closed.

On Load and Execute, the virus uses a much more complicated handler. After re-deleting the file HDFLOP.PDR, the virus hooks Ints 24h (Critical Error) and 1Bh (Control Break). It then gets, saves, and clears the file's attributes, before going on to examine the filename. It does not infect files whose names match the patterns TB*.*, F-*.*, IV*.*, CH*.*, or COMMAND*.*, nor those containing the letter V.

After infection, the file's time-stamp and attributes are reset (the virus modifies the time-stamp of infected files to set their seconds field to 34), and the handler is complete.

On all Find First/Next calls, the virus can do limited file length stealth – it examines the time-stamp of files encountered and subtracts 7680 bytes from the length of files tagged as infected. This results in infected files appearing to be larger than they were before, albeit not as much so as they actually are.

The Close Calls functions invoke a handler which will infect the file if it is deemed necessary – it first extracts the filename by manipulating the SFT (System File Table), performs the name checks described above, and then, where applicable, infects.

On Open Existing File requests, if Hare determines that the file is infected, it is disinfected (on disk) before the call is allowed to proceed. This will temporarily remove the virus from the file in question, which will be reinfected when the file is closed.

In Memory: Int 13h

The Int 13h handler performs stealthing of infected boot sectors, and also infects the boot sectors of floppy disks as they are used.

This latter is accomplished by first ensuring that the floppy in question is not already infected – the virus reads the boot sector (it retries three times; just what the manuals say should be done), and subtracts the word at offset 100h from that at offset 102h. If the result is CCFFh, the boot sector is deemed infected.

If the floppy is not already infected, Hare formats an extra track at the end of the floppy disk, encrypts the virus code, and writes the body to the extra track, and the loader code to the boot sector.

Loading from an Infected Boot Sector

When a computer is booted from an infected hard drive, the virus shuns the standard approach of immediately installing an Int 13h handler – this would make life much easier for it, as its own stealth features would allow DOS to see a valid partition table.

Instead, Hare copies the partition table back into the MBR whilst loading; thus, when the OS loader comes to look, the partition table is where it is supposed to be. It then knocks 9KB off base memory by the standard technique of modifying the word at 0000:0413h, intercepts Int 1Ch (System Timer Tick), and passes execution to the code of the original Master Boot Record.

Using a technique already seen in several other viruses, Hare monitors Int 1Ch to watch the operating system load. When it determines that it is safe to do so, it intercepts Ints 13h, 21h, and 28h (DOS Idle Interrupt). The first time the system issues an Int 28h (which will happen as soon as a program waits for input), Hare re-corrupts the partition table (which was fixed to allow the OS to load). It is now in a position to infect files and disks as they are accessed.

Trigger

On 22 August and 22 September, the virus' trigger routine is activated. First, it displays the message:

```
"HDEuthanasia" by Demon Emperor: Hare Krsna,
hare, hare...
```

Next, it attempts to wipe all data from all hard drives on the system with garbage.

Conclusion

Despite the many new and interesting techniques displayed by Hare.7610, the virus has several bugs. It is generally unstable, and replications will sometimes not execute properly (this includes MBR infections), and will hang the machine. The destructive trigger also sometimes fails. The fact remains, however, that Hare is in the wild across the world, and appears to be spreading. So far, it has been found in the wild in Canada, Russia, the Netherlands, Switzerland, the UK, and the USA: it appears that such a wide distribution was achieved via the Internet.

[*Note: Two variants of Hare.7610 have been discovered, Hare.7750 and Hare.7786. As well as bug fixes, they will occasionally (one in sixteen times the system is booted from an infected disk) change the random data sector. This means that the polymorphic algorithm will change periodically on any given computer. Hare.7750 was distributed via posts on the Usenet groups alt.cracks, alt.crackers, alt.sex, and alt.comp.shareware.*]

Hare.7610

Aliases:	Krsna, HDEuthanasia.
Type:	Slow, polymorphic, multi-partite virus.
Self-recognition in Files:	Seconds field of time stamp set to 34.
Self-recognition in Boot Sectors:	Word at offset 102h in BS minus word at offset 100h equals CCFFh.
Self-recognition in Memory:	Int 13h, AX=5445h, expects return of AX=4554h. Int 21h, AX=FE23h, expects return of AX=000Dh.
Hex Pattern:	None possible.
Intercepts:	Int 13h, 16h, 1Bh, 1Ch, 21h, 24h, 28h.
Trigger:	On 22 August/September, prints message and attempts to trash disks.
Removal:	Identify and replace infected files. Format infected diskettes. Replace hard disk MBR with known clean copy (FDISK /MBR must not be used).

FEATURE 2

Viruses on the Internet

Sarah Gordon

Author's note: This article explores attitudes to virus distribution facilitated by the Internet. Our increased reliance on the Internet for communication, and the retrieval of information from untrusted systems, can be expected to bring more cases of point-and-click giving users new viruses of many types, including those which take advantage of existing security holes in insecure applications.

The World Wide Web is a wonderful place. In June 1996, I decided to explore it to research this article; specifically to gauge the success of the 1995 'let's get rid of Internet virus sites!' campaign which had been sponsored by the *NCSA* and some anti-virus product developers.

My first search brought me fifty thousand matches. After regaining my composure, I realised many of these must be related to other types of virus. Fortunately, a narrowed search proved I was right. Surely we are winning the battle to encourage responsible behaviour on the Internet!

Or are we? With my refined search, I found 2000 matches to computer and virus (or virii, as virus distributors like to call them). The first site I came across was one that offered the classic 'computer virus joke' file:

```
Arnold Schwarzenegger Virus. Terminates, stays
resident. It'll be back.
Freudian Virus. Computer becomes obsessed with
marrying its own motherboard.
Star Trek Virus. Invades your system in places
where no virus has gone before.
```

What was to come was not so amusing. As I pointed and clicked, I found other 'virii' sites. Some pages were not fully operational, but many more were. Some were old pages I had run across months ago which had been taken down during the brief flurry of 'stop the virus sites'.

At that time, I predicted that the sites would come back, or reappear under other names. I hate to say it, but… *I told you so*. The sites have returned, and the methods we have tried to use to stop them have not worked.

Anatomy Lessons

What exactly can be found by following the downward spiral of the World Wide Web? More than some people would have you believe, to be sure.

I began with a site reference on university coursework. This was of particular interest to me, as I had just returned from the IFIP Conference in Samos where I heard a Swedish professor explain that making viruses was part of his curriculum. When I mentioned that two of the virus writers with whom I had spoken were students at his university, he told me he had heard about them, but he did not seem to think it noteworthy.

The following, a description of coursework from an American university, illustrates the casual attitude toward viruses which seems to prevail at many universities.

> *Computer Virus analysis*
>
> *Take a computer virus and analyse it thoroughly. You will have to isolate the virus code and disassemble it … Once you have it disassembled, you now have a program listing which IS the virus. Go through it, one assembly language statement at a time, and figure out what it does and how it works. It is best to do this on a fairly simple virus … I have a copy of the Natas virus if you want to try that one.*

This was the most responsible entry. While some would say using viruses as part of a learning exercise is 'good experience', others say it is 'poor science'. Deciding whether or not Natas is a 'fairly simple virus' remains a task for the reader. From this site, it was all downhill.

Under the banner 'Free Speech On-line Blue Ribbon Campaign', I was welcomed to 'The Virus Page: VIRUS PROGRAMMING and VIRII'. I was invited to join the Blue Ribbon Anti-Censorship Campaign and given access to all sorts of virus tutorials. There was information on disinfecting infected files, TSR, COM infections, non-overwriting COM infections, infection on closing, EXE infections, directory stealth, memory stealth, and a memorable tutorial, 'The Dangers of Thunderbyte'.

Polymorphic viruses were part of the plan as well, with 'Implementation, Detection, and Prevention'. Other instructions included infection of *Windows* executables, calling *Windows* API in assembly language from VLAD, heuristics, ANTI-AV Tricks (Tunnelling), Inbar Raz's Guide to Anti-Debugging Techniques and (from our own side), 'Anticipated trends in Virus Writing – Some ideas from the AV folks'.

There were also assembly language links, programming tools including A86 assembler v4.02, A86 debugger, a 32-bit *Windows* disassembler, *ViruScan for Windows 3.x*, *TBAV for Windows 3.x*, and, to my utter horror, *F-PROT*.

Does anyone actually get anti-virus software from sites which offer the latest and greatest virus source and executables right alongside anti-virus software? You would hope not, but I learned that some people do!

Some company employees of major firms told me that they 'trust' the virus sites because there is so much 'information' there. These are the people who are responsible, in some cases, for securing your systems. There were links to other

pages, too numerous to mention, most of them virus-related. There was even a link to Alan Solomon's hacking and virus laws page.

A trip to one of the links showed the same viewpoint, or possibly pseudo-viewpoint, one I saw repeated many times:

> Disclaimer: These files are for research and educational purposes only. I take no responsibility for any misuse of these programs which can result in ARREST OR DAMAGE TO YOUR COMPUTER. Please keep in mind that viruses are harmful and may destroy your computer: if you destroy other people's computers, you will be held responsible. Download at your own risk!

That site had files. The files were viruses, nicely catalogued. It also had generators, constructors and source code files. The warnings are nice. But who's kidding whom? Virus distribution in this manner is nothing less than irresponsible.

When I asked some of the people involved, the responses were generally that if the person who downloaded the viruses was incompetent to manage them, it would be that person's problem; that it is always the user's own choice to download. Virus sites are well and truly on the Internet, and they are here to stay.

"there are real problems in becoming the censor of user communications, both from a legal and an ethical standpoint"

A Problem with the American Legal System…

…is the outcry of some anti-virus researchers. Indeed, this is a possibility worth considering. People may take this position because some American-based public Internet Service Providers (ISPs) and on-line services hide behind the whimper 'it's not illegal'. Does this demonstrate a terrible ethnocentricity on the part of these providers? After all, the Internet is global.

An examination of one of these same providers' publicly available FTP logs shows computer viruses being siphoned to the UK just last week. Japan is another popular location on the receiving end of viruses from American ISP clients.

However, is action on the part of the service provider part of the solution? Is 'it is not illegal' adhering to the outdated paradigm 'If it's not illegal it must be OK'? Some would argue that it is, and that ISPs and on-line services should take more responsibility for the actions of their users and for the welfare of the computing public. Others recognize that there is, in fact, no viable solution.

There are real problems in becoming the censor of user communications, both from a legal and an ethical standpoint. These problems place ISPs, on-line providers and bulletin board operators in situations which may be impossible to resolve.

In 1994, representatives of several unnamed commercial ISPs and on-line services were questioned by various people regarding their policies on allowing viruses to be distributed or made available from their servers[1]. Reactions varied from 'it's legal' and 'we cannot become censors of our users', to 'we will not knowingly allow such things to be made available on our site'. It is interesting to note, however, that all the sites queried still have viruses and other 'questionable' material available from time to time.

Of course, service providers' views are based not only on the laws, but on the feelings of their customers and potential customers. 'Is it OK to make viruses available for public consumption, via the Internet?' – I have asked this question countless times, in public forums, on BBSs, at Conferences. Opinions seem to fall into two categories:

- it's nobody's business what anyone else does as long as it doesn't hurt anyone directly
- you can't do that because I don't like it

Defining 'directly' seems to vary from culture to culture; that discussion is best left for another publication.

I thought it might be interesting to query individuals in the IT field and ask the same question. The responses reflect what I have heard from the computing community in general. Only two responses stated that virus distribution should be illegal. The first said:

> *Maybe virus distribution should be illegal, but policing it will always be a problem. The Internet offers a new perspective on the 'Global Village' concept. These are issues yet to be resolved – who knows if they ever will be? A person who makes viruses available should share the responsibility, but the key word is 'should'. That opens a new arena of conflict: we must learn to be wary and learn how to avoid these problems. The ideal would be nice; people providing only helpful, useful items on the Internet. There should probably be some sort of punishment for malicious intent, but I hesitate to invite excessive government regulation to the Internet.*

A similar response:

> *I don't believe in censorship in many cases. I do believe in restricting the public market. If a person wants to write a virus, he should have the freedom to do so. If he wants to send it to his friends, still his business. If he would like to place it on his own FTP site and distribute it, as long as it is clearly marked as virus, then he should be allowed. Any distribution of the virus into the public should be illegal.*
>
> *It is the responsibility of the individual if he is on the Internet to watch out for harmful code. It should be assumed that files being downloaded may be infected.*

Then, there were those who took a more casual attitude:

> *Since I've never had a virus, and don't work on systems that most viruses infect, I'm just not that familiar with, or interested in, viruses. I find that most*

people who are very interested in viruses are those who got one and were determined to 'out' their intruder, to figure out everything they could about the creator or the processes involved.

I have a Macintosh at home. I am not very concerned about getting a virus at home, though I use Internet services daily (I don't use BBSs at all though). I run Disinfectant occasionally, but more out of a sense of duty, than fear. I don't have Word, or any other (known) macro infecting program. I think as these things go, based on my user habits and stuff, I have a low propensity for actually getting a virus. But I may be wrong.

These views seem typical of most Americans I have queried, but, despite the claim you will often hear that the USA distributes all the viruses (it used to be Bulgaria – I suspect neither deserved the amount of 'credit' bestowed upon it), I found virus distribution on the Internet to be culturally diverse. The US was there, but along with Canada, Austria, Portugal, Germany, Sweden, Norway and the UK. Viruses were available via FTP, WWW, or in casual trading centres such as IRC: they seem to have become the POGS of the Information Age.

New acquisitions are made with relative anonymity and virtually no interference. The logs of a real server, recorded 1–18 June 1996, showed various viruses, including Monkey and variants of Stealth, being retrieved by willing users. It is possible, of course, to identify users who obtain viruses via anonymous FTP or WWW should one desire to do so.

IRC BOTS dispensing viruses seem to have, at least for now, disappeared. I was pleased to hear this, but then reminded by a cynical friend that there was no need for VirusBOTS. After all, why spend the time getting limited information from a BOT when you can get all the viruses, source, and tools you want directly from the World Wide Web?

We still have the question 'How can we prevent this sort of irresponsible behaviour?' The problem seems to be that we don't really know whom we should be asking to stop it. Although, for the most part, virus download areas eventually fall into disrepair and disappear, there is a continual influx of 'young blood', keeping the number of sites in some sort of steady state.

The ISPs, companies, or universities which host these sites will not, for the most part, stop allowing such activities. For every site which acts responsibly, and does prevent such behaviour, there is a person determined to exercise his rights, oblivious to the concept of duty and responsibility…

As the college has taken this page away from me, I am searching for a new home for this information. Please, if you have any suggestions, email and tell me, I'd like to make the page available as soon as humanly possible. I'm sorry about this, but don't let it discourage your learning, because I won't let it discourage mine.

-The Demon X(a/n)^th

Supply and Demand

Who are the people commonly said to share in the Vx Internet pie? The four groups in contention for this dubious honour appear to be the virus writers and distributors themselves; the average user; the employee (who may be in charge of tech support or product evaluation); and finally, the anti-virus product developer.

The group with the most potential interest in VxWWW sites are the virus writers and distributors themselves[2]. Much of the information stored on such sites is of reasonably high quality, and can provide interesting pointers (in the form of source code or text files) to new techniques. For those who trade viruses, the attraction of such sites is obvious.

How much impact these sites have among virus writers is questionable; however, in the same way that a frisson of fear went through the industry when the VxBBSs began to appear (though the boards had little discernible effect), it is entirely possible that the impact of viruses on the WWW will not lead to vast numbers of new viruses or variants. Only time will tell.

"making viruses available via the Internet may be the 'right' of some people in some countries, but it is not responsible behaviour"

The second group, which encompasses the average user, is in the unenviable position of having the intrigue of viruses thrown at him by the media, the scare put into him by some companies, and the WWW at his disposal to get 'information' which he may think will help him protect himself.

What he does not realise is that this point-and-click could cost him his data: infected documents and Trojanised information abound on the Internet. The biggest risk which is posed to the 'average' user by these boards is that of accidental infection.

The third group with an interest in VxWWW sites comprises those interested in obtaining viruses for product testing. Although some anti-virus companies have gone so far as to recommend this, such actions are demonstrably wrong. After all, without investing a significant amount of time and expertise, it is next to impossible to verify a virus collection obtained from a third party, or to remove all Trojans, joke programs, first generation samples, simulated viruses and corrupted files.

Tests carried out on a virus collection which is not clean (i.e. does not contain real viruses) are meaningless at best, and can be completely misleading[3]. Thus, these sites are of little use as a source of scanner fodder; the problems outstrip any possible benefits.

The final group, the anti-virus product developers, are presented with a unique situation. Ever since the beginning of 'public' virus distribution, the mainstream anti-virus industry has scorned those who trawl the boards for the latest viruses. This was done initially because many VxBBSs required a user to upload new viruses to gain connect time, and also to prevent the legitimization of particular boards. However, the issues are no longer as clear.

At the recent *NCSA IVPC Conference* in Washington, one anti-virus company spokesperson publicly admitted obtaining viruses from Vx sites. I am totally against irresponsible virus distribution and joined with the majority of vendor representatives who chastised the errant company.

However, we do need to keep up with virus authors: accessing what they make available to the general public, to our customers. Knowing that people are in fact accessing and experimenting with these viruses may force a change of heart among the anti-virus community.

I believe much of the anti-virus community's reaction to the admission by the unnamed company was overreaction, based on our instinctive distaste for Vx sites in general. It is one thing to say you do not condone them while sneaking around giving or receiving viruses; unfortunately, some vendors are said to have been involved in this.

It is another matter altogether to admit that, due to the proliferation of these places, we must keep up with current trends. The only way to do that, some say, is to see what is there; to access and examine the viruses.

Unlike the VxBBSs of old, the viruses are there, free for all, only a point-and-click away… what are we supposed to do? Most anti-virus researchers do not obtain viruses from these places, claiming the mixed messages this would send outweigh the benefit of ethical behaviour related to viruses on the Internet. However, the issue is much less clear-cut than you might believe.

Clearly, the Internet is a fabulous place to obtain viruses, no matter who you are or what your intentions. Granted, you shouldn't use them to test anti-virus software. Such tests have been shown many times over to be flawed, and in some cases dangerous to the health of your company. You should not spread them to the unwilling and unknowing – even most virus writers acknowledge this. There is nothing a user can 'learn' from looking at viruses which cannot be learned from non-replicating programs.

Unless you are a product vendor or virus writer, the benefit to you from such sites is practically nil – and even if you are a vendor, the benefit is limited. The risks these sites provide to computer users in general, however, remain high. Owners and maintainers of such sites have no control over how the materials they make available are used. While this is the case with most FTPd or WWW materials, it is particularly undesirable in the case of viruses, as they are uncontrollable once released.

This leaves us with the question, again: 'What is the purpose of allowing such irresponsible behaviour?'. Maybe you believe it is an exercise in free speech, or that it is a 'right'. Making viruses available via the Internet may be the 'right' of some people in some countries, but it is not responsible behaviour. It is also, unfortunately, not showing any signs of slowing down.

Closing Thoughts

Finding a suitable conclusion to this article has been difficult, because I don't think that we are even close to finding answers. We don't know whom we should ask such simple questions as 'Why do we allow this kind of irresponsible behaviour on the Internet?'.

While it is a cliché to say that the Internet causes us to re-evaluate what we mean by censorship and freedom of speech, there is little doubt that the rapid development of the WWW has outstripped our ability as a society to control its contents.

Yes, there are viruses on the Internet, accessible via the World Wide Web, FTP, IRC, email, Usenet and other ways not discussed in this article – but we must keep our perspective. There are also infinitely more threatening problems, like child pornography, which I was unfortunate enough to encounter during my research for this article. The issues to which the Internet gives birth are much bigger than simply computer security and viruses. They envelop our communications with the fabric of cultural diversity, and force us to change the way we, in our own hometowns, think, live and do business.

There is no easy way to make us all think in the same way and magically solve the problem of irresponsible action on the Internet, be it child pornography, church-burning sound files, or computer viruses. We who work to fight computer viruses can only try to educate the public to protect itself from those who put the responsibility on the 'other guy'.

It is possible that, someday, those who view it as incumbent upon the 'other guy' to be technically competent, responsible, and ethical will realise that individual responsibility begins with not distributing or writing computer viruses in the first place.

Footnotes:

[1] *Virus-L Digest*, Fridrik Skulason. August 1994.

[2] 'Technologically Enabled Crime: Shifting Paradigms for the Year 2000.' Sarah Gordon. *Computers and Security*. November 1995.

[3] 'Analysis and Maintenance of a Clean Virus Library.' Vesselin Bontchev. *Virus Bulletin Conference Proceedings*. September 1993.

The views expressed in this article are those of the author, Sarah Gordon, a researcher at *Command Software*. Readers wishing more information on the subject may contact her via email at: sgordon@low-level.format.com.

PRODUCT REVIEW 1

CPAV for NetWare

Martyn Perry

Having recently evaluated *Norton AntiVirus*, this month we look at its stable-mate, *Central Point Anti-virus for NetWare* (*CPAVNET*). This supports both version 3.x and version 4.x of *NetWare*.

The product is licensed on a per server basis, and the software for workstation protection requires separate licensing. Although perhaps more applicable to workstation licences than to servers, the user may have the software on a single home computer, provided that the software receives at least 80% of its use on the primary computer.

CPAVNET comes with manuals for DOS, *Macintosh*, and *NetWare*. In addition, there is a manual for the alert management software, Central Alert.

Installation

Installation is performed in three stages. First, the workstation, which is used to install the network software, is checked and a version of the DOS product is copied to it. Next, the NLM is installed from DOS. Finally, the control and configuration software is installed either to the workstation (for local use) or onto the network for access (from any workstation). Both *Windows* and DOS versions of the control program can be installed into the same directory, \CPSNET. A nice feature is the display listing the files to be installed, highlighting the file currently being copied.

Multiple servers can be grouped together into one or more 'security domain'. The file servers to be grouped together into a domain can be selected individually, provided that sufficient licensed copies of the software are available. The domain name can be freely chosen.

When the NLM is installed, its files are copied from the first disk to the server. These include the directories SYS:SYSTEM, SYS:SYSTEM\CPAVNET, SYS:SYSTEM\CPS, and SYS:SYSTEM\CPS\CALERT.

The installation process next offers to add lines to AUTOEXEC.NCF to load the NLM at server boot time. There is a prompt to LOAD CPMASTER on the console, to allow the administration or configuration program to be run. The installation finally offers the chance to install the Configuration Program for DOS, *Windows*, or both.

Loading the NLM

If the automatic load option is not chosen, the *CPAVNET* NLM program is loaded from the server console prompt using the command 'LOAD CPAVNET.NLM'. This loads the main NLM plus a number of subsidiary NLMs. CPMASTER.NLM must also be loaded on at least one server in the domain, to configure the various options for the scanner and activate Central Alert.

The CPAVNET.NLM can be driven from the server console using the function keys to start or stop an immediate scan and to enable or disable the NLM. Additional function keys allow for keyboard locking and for the application of password protection.

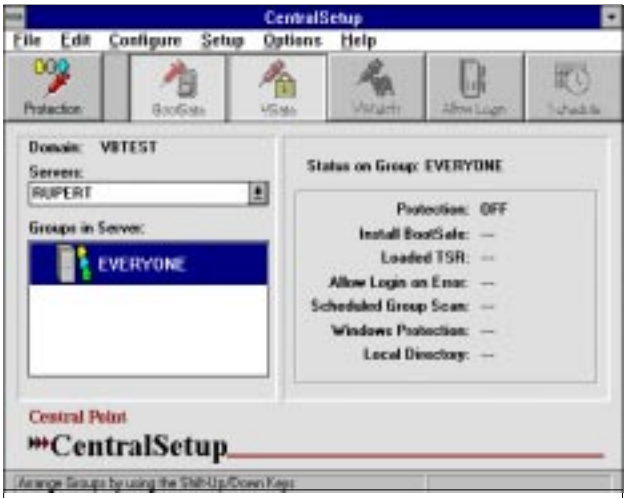
Administration

The scanner administration can be managed from the DOS or the *Windows* configuration program running on a workstation. The CPMASTER NLM must be running on each server to be configured before the configuration program is run.

The software allows servers to be added to a security domain, providing that the administrator has the necessary supervisor rights to those servers. The main administration screen provides access to view the various servers and their protection status. *CPAVNET* has the usual three modes of scanner operation: immediate, real-time and scheduled.

An immediate scan checks the server on demand, using the current immediate settings. Scanning on the server can be started either from the option on the workstation, or by using F6 on the server console.

The on-access, or real-time, scan allows scanning to be performed when a file is copied to or from the server, or when a file on the server is otherwise accessed. It is not possible to disable this option completely; scans of incoming or outgoing files, or both, must remain selected.



The setup program allows installation to be customized in various ways.

A scheduled scan provides scanning on a timed basis. An additional option is to have periodic scanning, which occurs at regular intervals; e.g. every hour between a defined start and stop time. It is possible to start another NLM after a scheduled scan is completed – for example, a backup NLM could be executed here.

Configuration Options

For each mode of operation, various selections can be made. These include the file extensions to be included in the scan: the defaults are EXE, COM, DLL, OV?, SYS, BIN, 386, FON, ICO, and CMD. Extensions may be added or removed as necessary.

As well as file selection, the product provides the ability to exclude files from the scan. This exclusion from on-access scanning is the only way in which infected files can be handled manually.

A separate menu option allows the selection of actions to be taken upon detection of a virus. There are three choices here; to delete an infected file, to move an infected file to a user-defined quarantine directory (the default directory is SYS:SYSTEM\CPAVNET\INFECTED), or to do nothing with the file.

An extra option is included, which *Central Point* defines as analysing for unknown viruses. This examines a file for 'suspicious behaviour'.

Alert Management

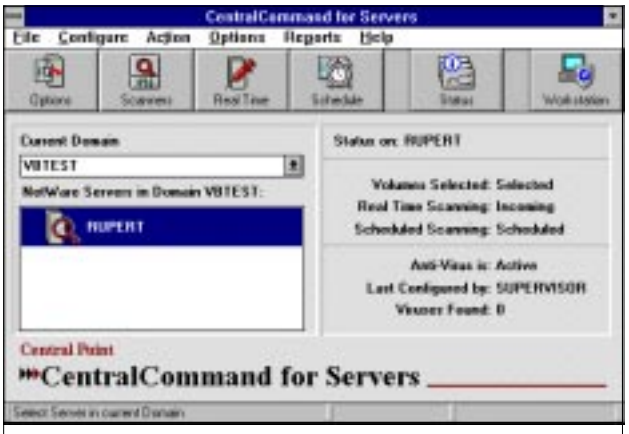
Apart from the action items which occur on virus detection, there is a separate alert program, Central Alert. This will allow modification of the current security domains as well as the sending of alerts to various alert facilities. These are:

- alphanumeric or numeric pager
- *NetWare* broadcast to the workstation
- flash Central Alert icon [*! Ed.*]
- log alerts to a file
- send MHS mail
- send SNMP traps to *NetWare* Management System workstations

Reports, Activity Logs and Updates

CPAVNET keeps a record of events in an Activity log. The events to be logged can be chosen and include:

- detection of known and unknown viruses
- scanner start and end times
- action taken
- enabling and disabling of *CPAVNET*
- loading and unloading of *CPAVNET*
- virus signature changes
- miscellaneous errors and warnings



The administrator can choose which type of scan to have running.

With this amount of data some sort of control is needed, which is supplied in two ways; first, by limiting the size of the log file, second, by filtering the events being displayed. There is a problem here – the whole log file needs to be loaded before it can be filtered. Other information available includes a virus list, and domain status (down to individual servers and their status).

Updates are performed by selecting the appropriate compressed file (for DOS, NLM, etc), and copying it to a temporary directory on the workstation. From here it can be self-extracted and resultant files copied to the correct directories.

Detection Rates

The scanner was checked using In the Wild, Standard and Polymorphic test-sets. Undetected viruses were identified by using the 'delete files' option and listing the files left in the virus directories. The tests were conducted using the default scanner file extensions supplied.

The results were generally disappointing. The tests were initially performed using the virus signatures shipped with the main product (March 96), then using the latest available (June 96). The Standard set scored 37.2% on initial scan: the updated version achieved 60.4%. The In the Wild set managed 65.7% on both passes, which implies that no detection improvement was made in the three months between the signature updates. The Polymorphic result improved slightly, from 41.4% to 43.5%, by virtue of the scanner finding additional instances of the One_Half virus.

A further scan was performed with the Virus Analyzer option selected. This made no difference to the results, however.

Real-time Scanning Overhead

To determine the impact of the scanner on the server when it was running, the usual tests were executed; copying 63 files of 4,641,722 bytes (EXE files from SYS:PUBLIC) from one server directory to another using NCOPY. The directories used for the source and target were excluded from the virus scan to avoid the risk of a file being scanned while waiting to be copied.

Because of the different processes which occur within the server, the time tests were run ten times for each setting and an average was taken. The chosen tests were executed in two groups, for two conditions.

The first group was run with on-access scanning selected first for both incoming and outgoing files, then for incoming files only. The tests were first run without the Analyzer, to establish the effect of the scanner by itself on the server performance. The four tests were:

A. NLM not loaded: this established baseline time for copying the files on the server

B. NLM unloaded: this test was run after the other tests to check how well the server returns to its former state

C. NLM loaded, using default setting of on-access scanning for incoming and outgoing files – immediate scanner not running. This tests the impact of on-access (real-time) protection.

D. NLM loaded, on-access scanning for incoming and outgoing files – immediate scan running. This shows the full impact of running the scanner on server files.

The tests were repeated with the Analyzer selected to judge its impact on performance. A separate set of tests was run with on-access scanning set for incoming files only.

At first glance, the results look a little strange. The difference in time between incoming/outgoing scan and incoming only scan were within the process variability of the server and, for practical purposes, can be viewed as the same.

The results with the Analyzer on appear to be better than those with the Analyzer off. Again, this could be attributed to server process variability; alternatively, it may indicate that separate buffering is used to process the file under analysis, leading to a slightly improved performance.

The performance overhead of checking files using the Analyzer does not appear to be significant. However, in view of the lack of additional success on the test machine, it is debatable whether or not this feature is useful. *CPAVNET* performs a clean unload of all the files which were originally installed, so there is effectively no overhead.

Conclusion

The product is easy to install and performing upgrades is straightforward. The documentation provided is clear and comprehensive.

CPAVNET's scanner detection rate is, and has been for some time, at a level unacceptable for a mature product. It is sad to see a product, which is 'feature rich' in other aspects, fail so badly in this crucial area. This product cannot be recommended as a first-time purchase due to this basic weakness. Existing users should consider biting the bullet, and take the opportunity to move to a product which is better supported; otherwise, they leave themselves seriously exposed to new virus threats.

Central Point AntiVirus for NetWare

Detection Results

Test-set[1]	Viruses Detected	Score
In the Wild	197/300	65.7%
Standard	247/409	60.4%
Polymorphic	4141/10000	41.4%

Overhead of On-access Scanning:

Tests detail the time taken to copy 63 EXE files totalling 4.6MB. Each test is carried out ten times, and an average taken.

	Time	Overhead
NLM not loaded	10.7	n/a
Incoming/Outgoing; Analyzer Off		
NLM loaded, no manual scan	16.2	51.0%
NLM loaded, manual scan	44.8	319.0%
Incoming/Outgoing; Analyzer On		
NLM loaded, no manual scan	16.6	54.0%
NLM loaded, manual scan	43.8	309.0%
Incoming only; Analyzer Off		
NLM loaded, no manual scan	16.4	53.0%
NLM loaded, manual scan	46.0	329.0%
Incoming only; Analyzer On		
NLM loaded, no manual scan	16.3	52.0%
NLM loaded, manual scan	44.5	315.0%

Technical Details

Product: *Central Point AntiVirus for NetWare*.

Developer/Vendor: *Symantec Corporation*, 10201 Torre Ave, Cupertino, CA 95014, USA. Tel +1 408 252 3570, fax +1 408 253 4992.

Distributor UK: *Symantec UK Ltd*, Sygnus Court, Market Street, Maidenhead, Berks, SL6 8AD. Tel + 44 1628 592222, fax + 44 1628 592393.

Price: The per-server price of this product in the UK is an estimated £600-£645. For site licences, apply directly to the company's corporate accounts division in the UK: Tel +44 1628 592222.

Hardware Used: Server – *Compaq Prolinea 590* with 16MB of RAM, 2 GB of hard disk, running under *NetWare 3.12*. Workstation – *Compaq 386/20e* with 4MB of RAM, 207 MB hard disk, running under *MS-DOS 6.22*, *Windows 3.1*.

[1]**Test-sets:** For a complete listing of all the viruses used in this review, see *Virus Bulletin*, July 1996, p.22. For a complete explanation of each virus, and the nomenclature used, please refer to the list of PC viruses published regularly in *VB*.

PRODUCT REVIEW 2

Survival of the Fittest?

Dr Keith Jackson

The *AllMicro Anti-Virus Survival Kit* (*AVSK*) claims to have 'four levels of defense to help keep your PC virus-free and your data safe'. Versions of *AVSK* suitable for DOS, *Windows 3.1* and *Windows 95* are included, though this review covers no *Windows 95* features.

Levels and Features

The *AVSK* manuals make great play of the fact that various 'Levels of Defense' are included. Level 1 incorporates a scanner, memory-resident software, checksumming features, and facilities for disinfecting infected files. Levels 2 and 3 are software updates, and the response of the developers to new viruses reported to them. Level 4 refers to data recovery facilities which can replace a damaged boot record and/or primary partition.

Although I cannot think of many anti-virus software developers who do not offer software upgrades and responses to new viruses, and the majority of anti-virus products incorporate 'data recovery' features, this is not meant to decry the features available within the software itself.

Included with the version provided for review are DOS, *Windows 3.x* and *Windows 95* versions of a full-featured menu-driven interface, a command-line-driven scanner, two distinct types of memory-resident software, a utility which reports on system facilities, and even a communications package which can be used to obtain software upgrades and/or virus signatures. There are too many components to discuss individually, so why dress things up by wittering on about 'Levels of Defense'?

Documentation

The printed documentation comprises a 125-page A5 *Users Guide*, and a 40-page A5 *RESCUE Users Manual*. A statement on the first page of the *Users Guide* reads that it 'avoids technical details'. This is true. Very true.

Sad to say, I found myself unimpressed by the *Users Guide*. It has a tendency to descend into trite explanations. For instance, is the explanation 'Mouse Active – activates or deactivates the mouse' really helping anybody?

The explanations of what to do if a virus is detected are sketchy, to say the least. This is somewhat offset by the on-line documentation, which provides information about individual viruses: short explanations of what the virus can do, presented as a series of boxes, reminiscent of *NAV*. It is, however, not enough. However, hardened users need more detail, and new users need more explanation.

On the plus side, the switches used by the command-line-driven version of *AVSK* are all listed in the *Users Guide*, along with an accompanying explanation. Similarly, all available options for the memory-resident programs are also thoroughly explained.

I have more time for the *RESCUE Users Manual*: although short, it provides a decent explanation of the data recovery facilities provided. Once again there is no index, but this makes less difference in a slim volume.

Installation

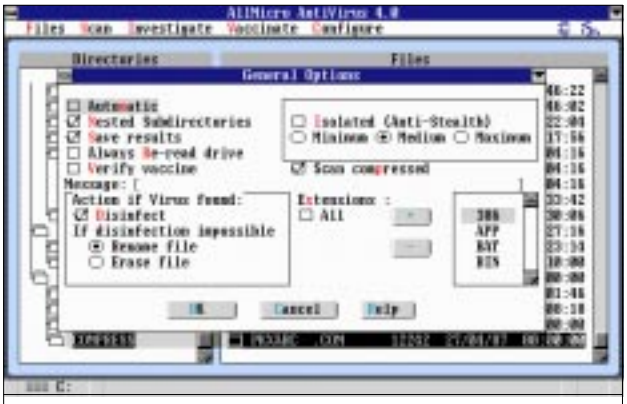
AVSK arrived on four 1.44 MB, 3.5-inch floppy disks, none of which were write-protected. Installation of the DOS version proved straightforward. After an initial warning message about viruses has been displayed, the installation offers to place the *AVSK* files in the default subdirectory, C:\AMAV – this can be altered to any desired location. The user's name and company must be entered to personalize installation; memory is then scanned: if no viruses are found, the *AVSK* files are copied across to the hard disk.

At this point, users are asked: 'Do you wish SENTINEL to be run from your AUTOEXEC?'. On-screen explanation would be more helpful – SENTINEL is a memory-resident scanner. The installation program next advises that the first action should be to create a SAFEDISK (for rescue purposes). Installation is then complete.

Installing the *Windows* program proved to be even simpler. The SETUP program offered a default subdirectory, allowed this to be altered, and then got on with things.

Scanning

As of 12 April 1996, *AVSK* claims knowledge of 8420 viruses. For reasons I could not sort out, the DOS version refused to access the virus test-sets stored on a magneto-



AVSK's general options allow for the depth of the scan to be altered at the user's discretion.

optical disk. Nothing I did could persuade it otherwise. This caused much file copying when the polymorphic test-sets were encountered. For reasons which are also beyond me, the *Windows* version was quite happy to access the drive.

The *Windows* version looks radically different from its DOS sibling, and proved very simple to use, with half a dozen on-screen buttons providing easy access to the main functions.

When used via drop-down menus, *AVSK* first scans memory, then displays the current subdirectory and its contents (in separate windows) – vaguely reminiscent of *CPAV*. Both DOS and *Windows* versions of *AVSK* offer options which can scan the entire system, an individual drive, a directory, a file, or the 'boot system'.

Scanning Speed

In its default state, the DOS version of *AVSK* scanned the hard disk of my test PC in 2 minutes 34 seconds (742 files in total, 311 files scanned, 29.9 MB).

AVSK recognises three types of compressed files (ZIP, ARJ and LZEXE). The option to scan inside compressed files is switched on by default, which of course slows down the scan. When this was deactivated, the hard disk of my test PC was scanned in 1 minute 40 seconds. With 'minimum stealth' specified, scan time fell again, to 1 minute 32 seconds. In the other direction, a scan of all files (including the contents of all compressed files) took 5 minutes 7 seconds.

Other methods of virus detection are included, and are even faster than the scanning itself. When *AVSK* searches for 'Suspicious Conditions', it inspects the entire hard disk in 37 seconds. A 'heuristic' scan takes just 2 minutes 40 seconds.

All the above timings were measured using the DOS version of the product. To provide a fair comparison, *Dr Solomon's AVTK* scanned the hard disk of my test PC in 4 minutes 21 seconds; *Sophos SWEEP* in 7 minutes 32 seconds.

One fact stands out, therefore: *AVSK* is very quick at scanning for viruses. A point worthy of note, however, is the slow-down in the other two scanners (*SWEEP* and *Dr Solomon's*) when SENTINEL is active: *SWEEP* took 13 minutes 6 seconds; *Dr Solomon's*, 15 minutes 1 second. This slow-down, imposed by the presence of SENTINEL, is severe.

Detection

I tested the virus detection capability of *AVSK* against the test-set described in the Technical Details section below.

Run against the In the Wild test-set, using default settings it detected 192 of the 286 test samples; 67%: frankly, not good enough. Curiously, the report file stated that *AVSK* had found 200 viruses, though only 192 files were infected: this was because all six samples of Jerusalem.1244 were detected as infected with (using *AVSK* nomenclature) both the Maca and the 1244 viruses, and the two samples of Keypress.1232.A were detected as doubly infected (Keypress and SamSoft).

The product showed no significant difference in detection rates despite the method of heuristic scanning chosen.

Against the Standard test-set, again using default settings, *AVSK* detected 229 of the 265 test samples (86%). Once again, samples (four in total) were detected as doubly infected (Warrier, 2 x Old_Yankee, Vienna). The DOS and *Windows* versions of *AVSK* detected the same viruses from the two main test-sets.

If the Standard and In the Wild test-sets are contained inside a ZIPped archive file, detection is slightly poorer. Only 180 files from the In the Wild test-set (63%) were found infected, and only 224 files from the Standard test-set (84%). I shall return to scanning inside archive files below.

Executing the 'Suspicious Conditions' option, rather than merely scanning, found 62 suspicious files in the In the Wild test-set (22%), and 51 (19%) in the Standard test-set. The heuristic level can be set to Minimum, Medium, or Maximum, although I could not measure a significant difference in detection when this parameter was varied. It was, however, ironic to find that 'Maximum Heuristics' found just one suspicious file: *AVSK's* own file SHIELD.COM. This was called an 'Unconventional Resident Program'. Ah well!

Polymorphic Viruses

When run against the polymorphic virus samples, *AVSK* detected 2196 of the 5500 test samples, or 40%. Three polymorphic viruses are detected only reasonably well (Girafe:TPE, Neuroquila.A and One_Half.3544), but the others are either not detected at all (three in total) or only very poorly (the remaining five).

When *AVSK* scans inside ZIP files, polymorphic detection falls off alarmingly – only 1209 (22%) of the samples are detected as infected. All but four viruses are completely undetected, and only Girafe:TPE is detected reliably. I am at a loss to see why this should be so. Surely, once a file has been extracted from a decompressed archive file, the same scanner should be used to test whether or not an infection is present? Clearly something is wrong with *AVSK's* decompression.

The product detected only fourteen of the twenty boot sector test samples, failing to detect EXEBug.A, IntAA, Peanut, Quox, She_Has and Urkel. By no stretch of the imagination can this be called an impressive result.

Vaccination

AVSK can create a database of checksum information about each executable file present on a hard disk (a process which it calls external vaccination), or it can add extra code to executable files (called internal vaccination). I am amazed that a manufacturer still considers changing executable files: use of such features is not recommended. Life is complicated enough without having to track down the inevitable problems that tampering with executable files may cause.

The product puts the two files which comprise the database for 'external' vaccination in the hard disk's root directory. I wish programs wouldn't do this. I am happy for *AVSK* to maintain a database, but it should do so in its own subdirectory.

When either creating or verifying external vaccination, the DOS version of *AVSK* took 3 minutes 50 seconds to work its way through the entire hard disk of my test PC, rising to 6 minutes 10 seconds under the *Windows* version.

Memory-resident Software

AVSK contains two distinct memory-resident anti-virus programs. One (SENTINEL) is a memory-resident scanner, the other (SHIELD) is a behaviour blocker. The documentation calls SHIELD a 'memory-resident program, whose mission is to prevent the damage that a known or unknown virus may create…': what this means is that it monitors (and prevents) certain actions; e.g. it can be set up so that any write to hard disk only takes place after user confirmation has been given.

SENTINEL can be added to AUTOEXEC.BAT by the installation program (see above), but SHIELD must be invoked by the user (either manually or as an addition to AUTOEXEC.BAT). When installed, SENTINEL uses 18.8 KB of conventional memory and 32 KB of expanded memory. SHIELD is much smaller, requiring only 3 KB.

When SENTINEL was executed with the /AE switch to ensure that all file extensions were scanned, my test PC locked up, complaining it could not load COMMAND.COM.

Testing Files before Execution

Any memory-resident monitoring program which carries out tests before allowing a file to be executed must have an impact on system performance. I measured this by copying 40 files (1.25 MB) from one subdirectory to another. With no memory-resident software present, it took 21.6 seconds, rising to only 22.1 seconds when SENTINEL was present.

This is very impressive. The result moved, however, to inducing curiosity when the file copying time went *down* to 21.3 seconds, with SHIELD added to SENTINEL.

Given the lack of overhead introduced by SENTINEL and SHIELD, it is difficult to explain why SENTINEL had such a drastic effect on the speed at which other scanners execute. Something odd is going on.

Behaviour Blocking

It is only necessary to use a PC with SHIELD active in memory for a few minutes to realise why the developers separated the two memory-resident programs. Put bluntly, SHIELD is a nuisance. If activated with all security options active, it is forever popping up and requesting confirmation. If some of its security features are turned off to prevent such intrusions (a hot-key is provided to facilitate such tailoring), effectively, SHIELD is doing nothing.

SHIELD is not alone in being intrusive or useless – all behaviour blockers tend to be like this. As a virus is merely a computer program, there is no foolproof way to decide which actions to allow and which to prevent. The only solution is to keep asking the user for confirmation as to whether a certain action should be permitted: this fails, as the average user has no idea how to answer such questions.

Therefore, although at first sight behaviour-blockers seem like a good idea, they come off the rails when the real world intrudes. SHIELD may have some use in constrained environments where users are to be allowed only a few actions, although I'm unconvinced.

Memory-resident Detection

When SENTINEL is executed, it states that it looks for only 420 viruses. Detection capabilities were measured by copying the files in the In the Wild and Standard test-sets from one drive to another. SENTINEL detected 179 and 187 infected files respectively in each set. These figures are only slightly less than the main *AVSK* scanner; surprising, given the low number of viruses about which SENTINEL claims knowledge.

Conclusions

My conclusions about *AVSK* are simple: it is very quick at scanning for viruses and/or verifying that checksums are unchanged, but simply not very good at detecting viruses. The memory-resident scanner is similarly poor, although surprisingly close to the DOS product in terms of detection. However, the behaviour-blocking memory-resident component is just plain annoying. Avoid it.

Technical Details

Product: *Anti-Virus Survival Kit v4.0* (no serial number visible).

Developer/Vendor: *AllMicro*, 18820 US Hwy 19 N, Suite 215, Clearwater FL, USA. Tel +1 813 539 7283, fax +1 813 531 0200, BBS +1 813 535 9042, email allmicro@ix.netcom.com.

Availability: *IBM PC* or *PS/2* or compatible running *MS-DOS v3.3* or higher with at least 512 KB of RAM. *Windows* components require *Windows 3.x* or higher with at least 2 MB of RAM.

Price: The base package can be downloaded from the company's Web site (http://www.allmicro.com/). Twelve months of updates cost US$79.95; six months, US$39.95.

Hardware used: *Toshiba 3100SX*; a 16 MHz 386 laptop with a 3.5-inch (1.4 MB) floppy disk drive, 40 MB hard disk and 5 MB of RAM, running under *MS-DOS v5.00* and *Windows v3.1*.

Viruses used for testing purposes: Boot sector test-set – see *VB*, March 1996, p.23. Standard, Polymorphic, and In the Wild test-sets – see *VB*, January 1996, p.20.

SUBSCRIPTION RATES

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

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

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire, OX14 3YP, England

Tel: 01235 555139, International Tel: +44 1235 555139
Fax: 01235 531889, International Fax: +44 1235 531889
Email: editorial@virusbtn.com
CompuServe address: 100070,1340
World Wide Web: http://www.virusbtn.com/

US subscriptions only:

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

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

END NOTES AND NEWS

***Sophos Plc's* next anti-virus workshops** will be on 25/26 September 1996 at the training suite in Abingdon, UK. The two-day seminar costs £595 + VAT. One single day may be attended at a cost of £325 + VAT (day one: Introduction to Computer Viruses; day two: Advanced Computer Viruses). For further information, contact Julia Line on Tel +44 1235 544028, or visit the Web site; http://www.sophos.com/.

The *NCSA* is hosting the ***Web, Internet Security and Firewall Conference***, which will be held in San José, California from 30 September to 1 October. Details on the event can be obtained from the *NCSA*; Tel +1 717 258 1816, fax +1 717 243 8642, or email fwcon96west@ncsa.com. Information is also available from their WWW site; http://www.ncsa.com/fw96west.html.

Intel Corp* has announced the launch of *LANDesk Virus Protect for NT. The product offers on-access scanning using server-based technology. Information on obtaining the product is available from the company; Tel +44 1793 403000 (UK); +1 408 765 8080 (USA).

In Dorset, UK, a man has been charged with blackmailing *Sun Alliance*: it is alleged that Keith Lamb, from Bournemouth, **threatened to infect the company's computers with 'computer bombs and polymorphic codes'** [*Oooh! Ed.*] if a claim he had made (which had been rejected) was not paid. Lamb was arrested after his calls were recorded. A verdict in the case, being held at the Old Bailey, is imminent.

Mike Hill, director of product marketing at *S&S International*, claims that **a weakness in *Netscape* leaves it vulnerable to virus attack**. Hill asserts, according to an article in the UK publication *Computer Weekly* (18 July 1996), that if the right mouse key is used to activate the shortcut menu, all calls to add-in software are bypassed, leaving files unchecked. Because of this, *S&S* has now delayed the release of its *WebGuard*, which was designed to work in tandem with *Netscape 2.0*.

Integralis has announced a working partnership with S&S: **it will license its email and scanning technology** to the anti-virus software developer, which will be marketed under the name *MailGuard*. Information on the deal is available from *Integralis*; Tel +44 1734 306060, or on the WWW; http://www.integralis.com/.

Reflex Magnetics has another **Live Virus Experience** scheduled for 9/10 October 1996. Further information is available from Rae Sutton: Tel +44 171 372 6666, fax +44 171 372 2507.

Seven Locks Software has been named as the **exclusive US distributor for Czech-based *Alwil Software's* security products**. *Alwil's AVAST!* has been a prominent front-runner in recent *VB* comparative reviews. Details on the agreement are available directly from *Seven Locks*; Tel +1 508 746 9087, or on the WWW; http://www.sevenlocks.com/.

The *NCSA* has announced the first **certification of products in its Firewall scheme**. Sixteen products have so far met the criteria for acceptance: information on the procedures involved, and the products registered, is posted on the *NCSA's* Web pages; http://www.ncsa.com/.

***CompSec 96* will take place in London, UK,** from 23-25 October 1996. For information, contact Sharron Elmsley at *Elsevier Science*; Tel +44 1865 843721, fax +44 1865 843 9458.

S&S International is presenting **Live Virus Workshops** at the *Hilton National* in Milton Keynes, Buckinghamshire, UK on 2/3 September and 7/8 October 1996. Details are available from the company: Tel +44 1296 318700, fax +44 1296 318777.

Readers are reminded that the *6th Annual Virus Bulletin Conference and Exhibition* takes place in **Brighton, UK, on 19/20 September 1996**. Contact Alie Hothersall for information; Tel +44 1235 555139, fax +44 1235 531889, WWW; http://www.virusbtn.com/.