ISSN 0956-9979 JUNE 1995

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **Guilty as charged?** On Friday, 26 May 1995, Plymouth resident Christopher Pile pleaded guilty to eleven charges under the *Computer Misuse Act 1990*. *VB* covered the case at *Plymouth Crown Court*, and reports on p.3.

• **Windows 95.** Exactly how much of a threat are viruses to *Microsoft's* latest operating system? For an assessment and an analysis, turn to page 15.

• **More on Macs.** A report from a university campus in the USA on how users and administrators there are coping with the threat posed by viruses on the *Apple Macintosh* - see p.6.

CONTENTS

EDITORIAL
Oh No, *Windows*! 2

VIRUS PREVALENCE TABLE 3

NEWS
1. No Good Times Here! 3
2. Fake *CERT* Advisory 3
3. 'Black Baron' Pleads Guilty 3

IBM PC VIRUSES (UPDATE) 4

TUTORIAL
On the Mac Front 6

VIRUS ANALYSES
1. WINSTART - Son of a Batch 9
2. Bua: Dangerously Obscene 11
3. Fancy a Quicky? 13

TECHNICAL NOTE
New Technical Editor for *VB* 14

FEATURE
Viruses on *Windows 95* 15

PRODUCT REVIEWS
SWEEP for Windows NT 18
ALERT to the Risk 21

END NOTES & NEWS 24

EDITORIAL

Oh No, Windows!

Windows 95. Can there be anyone in the computing industry who has not heard these fateful words? For so long now, the beautifully portrayed Siren-like figure of *Windows 95* has sung to users, luring them away from *OS/2* with promises of future glories. And, like Odysseus, users have found it hard to resist. *Microsoft* sings loud, and very, very sweetly.

'It's completely different from DOS,' they said. 'Yippee,' we cried; grateful at the prospect of getting rid of the dreaded 'C:\>' prompt after all these years. There are unspoken promises here. Completely different from DOS? Fabulous! No need to worry about those pesky viruses any more, boss, this OS is *different*! Well, take all this with a pinch of salt, because a lot of it is hype. Sure, it boots straight into *Windows*. Big deal, we could do that years ago by putting 'WIN' at the end of our AUTOEXEC.BAT. It has fast disk access, but that's been around since *Windows 3.1*. It also has installable file systems for easy networking, but *WfW* had that too.

"Windows 95 caught virtually every boot sector virus I threw at it"

Think back now, to when *Windows 3.11* and *WfW* were the latest thing, and *Windows 95* was little more than a gleam in *Microsoft's* research budget. Can you hear the cries of the marketroids? 'Uses technology from the Chicago project,' they proclaimed - and it seems that they were right. But does this mean that we've been using (and paying for) pre-beta code for two years?

Don't get me wrong, *Windows 95* has much in its favour. The new interface, friendly descriptions (like 'My computer'), support for Plug and Play, and 'networkability' are all terrific. But something is missing. Yes, it's happened. *Microsoft Anti-Virus* (MSAV) is no more. Whether *Microsoft* noticed how out of date it was, or whether they did not want to adapt it, we will perhaps never know. Whatever the reason, it's gone and (one hopes) forgotten.

However, there is someone in the *Windows 95* development team who has heard the phrase 'boot sector virus'. Indeed, *Windows 95* can tell when your computer has such a virus! Alas, it does not seem keen to tell you. For details on how to find out, read the investigation on pp 15-18, but suffice it to say it's not somewhere the average user will be looking daily.

This is surely a missed opportunity. *Windows 95* caught virtually every boot sector virus I threw at it. But whereas *NT* prints a catastrophe screen and collapses in an undignified heap when it discovers that it has such a virus (which probably counts as telling the user), *95* does the electronic equivalent of pinning up a very small notice on a dusty old bulletin board somewhere, and muddles along.

Why? Search me. Granted, the online help does state that the message may be due to some form of disk encryption software, but to my mind there are better solutions. Picture this - whilst booting up one day, the OS notices that the interrupt vectors have changed. At this point, it should scream bloody murder. If the user has just installed disk encryption software, he's going to know about it, and can then select the tick box labelled 'I don't have a virus, I've checked, so don't tell me this again unless the interrupt vectors change again'. On the other hand, if he has not, he is told to bootstrap his system from the rescue diskette (or even from a DOS system diskette), and check it. In fact, on the rescue diskette there could be a utility which copies back all the original boot sectors, so the user can be told to 'boot from the rescue diskette, and execute BRESTORE', or some such.

Quite apart from the *necessity* of warning the user, imagine the fun they would have had designing a 'You might have a virus' dialogue box! The rest of the OS is full of imagery - drums bang as it examines your hardware, torches shine around as it pings the network for machines with which to communicate, and sheaves of paper fly from folder to folder (yes, *Macintosh* fans, they're called 'folders' now!) as files are copied. They would have had a great time animating images to represent viruses, and having them gobble up the paper, smash the torches and break the skins on the drums.

There is time yet though, *Microsoft*, if you're listening. *Windows 95* has not yet shipped. Think about it - you have an unparalleled chance to do something towards eliminating boot sector viruses. After all, where do you want to go today? Make a nice slogan that, wouldn't it?

NEWS

No Good Times Here!

The latest issue of *VLAD* magazine (*VLAD* #4) contains the source code to a virus named 'Good Times' by its author, Qark. *VLAD* (*Virus Labs And Dist* magazine) is an electronic virus underground newsletter.

The major anti-virus producers have named the virus GT-Spoof in an effort to avoid Qark's attempt to muddy the waters of the Good Times affair further. A line in the source file states:

```
Remember to email all your friends, warning
them about Good Times!
```

The virus uses a polymorphic engine called 'RHINCE', which stands for 'Rickety and Hardly Insidious yet New Chaos Engine', and is derived from its author's nickname, 'Rhincewind'. This in turn is probably derived from 'Rincewind', who is a character in the fantasy novels of Terry Pratchett.

GT-Spoof only infects COM and EXE files, and *VB* stresses that it is not connected with the mythical Good Times virus. Its effects and techniques are in no way similar to those described in warnings about the Good Times virus ▮

Fake CERT Advisory

An article falsely claiming to be an advisory from *CERT* (*Computer Emergency Response Team*) has been posted to various *Internet* newsgroups, including comp.security.misc and alt.comp.virus. The message describes a fictional *World Wide Web* (*WWW*) virus, and states:

```
In the current attack pattern, the virus is
attaching and hiding its presence by infecting
the HTML file of a URL using a rare and
undocumented feature of the HTML language. Its
presence is very well hidden by dynamically
modifying the kernel… Kernel infections of
LINUX, MSDOS 6.2, MAC 7.5 and OS/2 Warp have
already been reported. THE WEB VIRUS IS VERY
DANGEROUS SECURITY THREAT!
```

Such behaviour is unlikely to be possible in reality, and *Virus Bulletin* knows of no *WWW* browser which allows such behaviour.

The author of the spoof was clearly only able to post his message to unmoderated newsgroups, and not, for example, to the *Computer Emergency Response Team* advisory mailing list or on to the *CERT* ftp site. He omitted to change the reference number (CA-95:07) from the genuine advisory which he had used as a template for the message. The original advisory using this number was issued on 10 April 1995, and concerns a vulnerability in SATAN (Security Administrator Tool for Analysing Networks) ▮

Virus Prevalence Table - April 1995

Virus	Incidents	(%) Reports
Form	17	14.7%
AntiEXE.A	14	12.1%
Parity_Boot	13	11.2%
JackRipper	8	6.9%
Jumper	8	6.9%
Monkey2	6	5.2%
NYB	5	4.3%
AntiCMOS	3	2.6%
Bupt	3	2.6%
NoInt	3	2.6%
One_Half	3	2.6%
Sampo	3	2.6%
V-Sign	3	2.6%
Cascade	2	1.7%
Junkie	2	1.7%
Stoned.Standard	2	1.7%
Other *	21	18.1%
Total	116	100%

* All reports of only one infection have now been combined into a single entry: a complete listing is available on request.

'Black Baron' Pleads Guilty

A new legal precedent may soon be set in the UK: on Friday, 26 May 1995, a man was charged with eleven offences pertaining to sections 2 and 3 of the *Computer Misuse Act 1990*. Christopher Pile, aged 26, unemployed and resident in Plymouth, pleaded guilty in *Plymouth Crown Court* to five charges of unauthorised access, and five of unauthorised modification, to computers. In a late addition to the charges, he was also accused of knowingly inciting other people to cause unauthorised modification to computer programs and data: i.e. writing viruses.

The last charge relates to a file, SMEG03.ZIP, which Pile allegedly uploaded to the *Abbey BBS*. In it, instructions are given on writing viruses using SMEG (Simulated Metamorphic Encryption Engine): the author purports to have tried to make SMEG 'as easy as possible', and hopes that the user will 'Have fun with SMEG' and 'pass it on to your friends'.

Testimony from the *CPS's* expert witness, Jim Bates (of *Computer Forensics Ltd*), was admitted to the court and used in evidence against Pile. The case has now been adjourned for a maximum of eight weeks, in order to allow the Defence to apply for *Legal Aid* to call an expert witness of their own, who was not named in court ▮

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 May 1995. Each entry consists of the virus name, its aliases (if any) and the virus type. This is followed by a short description (if available) and a 24-byte hexadecimal search pattern to detect the presence of the virus with a disk utility or a dedicated scanner which contains a user-updatable pattern library.

Type Codes

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

Antipode

CR: An encrypted, appending, 802-byte virus which contains the hidden text: '[Antipode 1.0] by Automag/VLAD'.

```
Antipode        C704 E803 00EB 1190 BE2D 0003 F28B FEB9 F502 313C 46E2 FBC3
```

Blava

CR: An encrypted, appending 787-byte virus from the Czech Republic. Contains the hidden text 'BLAVA 3.0 by RGB#'.

```
Blava           CF9D 81F9 FEFA 7510 81FA FAFE 750A FA9C 2EFF 1E9E 00B9 EEEE
```

Chuckcha

CN: Two Russian prepending, direct infectors.

```
Chukcha.554     B440 8B1E 8401 8B0E 8001 8D16 2A03 CD21 B43E 8B1E 8401 CD21
Chukcha.838     B440 8B1E 7401 8B0E 7201 8D16 D303 CD21 B43E 8B1E 7401 CD21
```

Dark_Avenger.1800.O

CER: Detected with the DA-related pattern.

DARV

EN: 1024-byte fast direct infector, named after the string 'DARV' placed at the end of infected files.

```
DARV            AC0A C075 FA49 BB01 00B4 40CD 21E8 FBFE B43E 8B1E 7900 CD21
```

Dragon

ER: 400-byte virus containing the plain-text message: 'DRAGON-2 Anti' It doesn't increase the length of infected files but inserts its code in the unused space of the EXE file header.

```
Dragon          8CC8 2E01 0691 000E 0606 8CC0 488E C026 8B1E 0300 83EB 1A07
```

Emmie.3097

CR: Encrypted, stealth, polymorphic, 3097-byte variant containing these messages: 'It'll tire you too much', 'My name is Emmie+, I am Eddie's sister'. The pattern below detects it in memory.

```
Emmie.3097      2E8E 06DE 0BCF 3D01 2574 E53D 0135 74EB E8F7 002E 803E D80B
```

Estonia

CN: Encrypted, 400-byte direct infector with a payload which triggers on 27 and 28 September. It contains the text: 'Your drives were on the Estonia ... They DIDN'T survive!!!'

```
Estonia         0151 8BCA 8BFB 03FA 8BF7 AC28 07AD 2AE0 8AC4 AA46 E2F7 59E2
```

GT-Spoof

CER: GT-Spoof is a polymorphic virus (it uses an encryption engine called RHINCE), about 1300 bytes long. When infecting COM files, it does not affect the initial jump/call instruction. Because of this bug, COM files which do not start with jump or call will not run and usually make the system crash. The pattern below detects the first generation of the virus and finds it in memory.

```
GT-Spoof        80FC 7675 0344 44CF 5351 5256 5755 1E06 89D6 FCAC 3C2E 75FB
```

Hemlock

CERM: A multipartite, stealth, polymorphic, 3183-byte virus with the hidden message: 'Hemlock by [quark/VLAD]'. It infects MBS on hard disks and the DOS boot sector on floppies. The template below detects the first generation of the virus and finds it in memory.

```
Hemlock         3DEF BE75 01CF E94E FF9C 0EE8 49FF 86C4 C350 1E51 E460 3C53
```

Holiday.3000

CR: Another variant of the Holiday virus.

```
Holiday.3000    3D00 4B74 03E9 B702 E815 0352 B42A CD21 80FE 0375 0880 FA03
```

John

CN: An unusual virus, to say the least. It is 1962 bytes long, but the code is short - the rest is a long 'hate' message directed at John Buchanan (alias Aristotle), a well-known member of the virus underground.

```
John            5D81 ED03 01BF 0001 578D B68A 08B9 1000 F2A4 B41A 8D96 AA08
```

Kaczor

ER: A polymorphic, 4444-byte virus from Poland containing the text: 'Zrobione', 'Wersja', 'Kodowanie', 'Licznik.....HD' 'k%a.c.z.o.r.t.e.s.t'. Two minor variants are known.

Kakashka

ER: 928-byte appending virus from Russia.

```
Kakashka        F6C6 0474 4FB4 FFBB FFFF CD21 0BDB 7444 1EBF 0200 8CD8 488E
```

KRAD **PN:** A 4658-byte direct infector.

```
KRAD            174C 3451 4844 5955 0108 4E51 8459 0A0C 0A2A 0DCF 0265 240F
```

Micro.128 **CR:** A short virus which installs itself in the Interrupt Vector Table.

```
Micro.128       80FC 4B75 3B50 5352 1EB8 023D CDE0 7228 8BD8 0E1F E8D7 FFB4
```

Monami **CR:** A 1059-byte virus which contains the following message in plain text: '[Mon ami la pendule] - Metabolis/VLAD'.

```
Monami          754B BB01 009D CF50 5680 FC2C 7518 E839 0351 52E8 C6FF 2E89
```

MOutOfCtrl **CR:** 288-byte, prepending virus containing the text: 'Monkeys out of Control'. Short files (<1000 bytes) will not run after an infection.

```
MOutOfCtrl      B990 00FD AD50 E2FC 5458 0514 0050 FAC3 4C4C 5DFB FC8C 8E80
```

Nigeb **CR:** A trivially-encrypted, 890-byte virus containing the text: 'AntiPascal-1 (c) Godzilla Corp'. The strange payload includes a procedure changing the typed-in string 'nigeb' to 'begin'.

```
Nigeb           56E8 0000 5B83 C30C 8BF3 83EB 12E8 5A03 A7E3 4269 47CB 6F6F
```

Notfound **ER:** A 6176-byte virus containing the text: 'Borland Virus (C) 1994 Processor Intel PentiumTM not found. 17GB disk free space not found. 512 Mb extended memory not found. 16Mb XGA Video card not found. Sound Blaster not found. Sorry, your configuration doesn't match to run this…'

```
Notfound        6869 732E 2E2E 0A0D 0120 9A00 007C 00C8 0202 008D BE00 FF16
```

Oops **CR:** 600-byte virus that only goes memory-resident after 31 May. It does not infect files with names matching the patterns *ST.COM, *NF.COM, -V.COM, *00.COM and *ND.COM. It contains the plain-text messages: 'Bad command or filename', 'C:\COMMAND.COM C:\COMMAND.BAD', 'Oops! Sorry for BAD virus!'.

```
Oops.600        BCCF 02CD 21CD 209C 80FC 9675 04B4 699D CF3D 004B 7503 E806
```

Puppet **CR:** An encrypted 487-byte virus, extremely difficult to replicate. Infects files 50000-60000 bytes long, which are shorter than initial jump offset + 490 bytes. It contains the text: '[PS/G] eMpIrE-X [G The Puppet Master 3 Virus]'.

```
Puppet (in files)   BB?? ??B9 ED00 8107 ???? 4343 E2F8
Puppet (in memory)  CC8B 6EFA 81ED 1100 061E B841 44CD 213D 5350 7456 B44A BBFF
```

Republic **CER:** An encrypted, 1206-byte virus which contains the text: 'Go the Republic!', 'Fuck off Royal Family', 'Qark/VLAD of the Republic of Australia'. The template detects the first generation and finds the virus in memory.

```
Republic        7416 583D EDFE 7403 E944 01B8 DEFA CFE9 A402 E973 02E9 9D01
```

Small.59 **CR:** A short, prepending infector that installs itself in the Interrupt Vector Table and infects files starting with a 'jump near' instruction.

```
Small.59        CEF3 A4EB DA60 8BF2 AC3D E940 750A 1E0E 1F99 B93B 00CD 211F
```

Small.263 **CER:** Another virus installing itself in the Interrupt Vector Table.

```
Small.263       BE84 0056 26A5 26A5 5FB8 4E02 AB91 AB5E 0781 C64A 0058 0BE4
```

Trakia **CER:** There are three variants of this virus; 561, 570 and 1070 bytes long respectively.

```
Trakia.561      B802 3DCD 2193 0E1F B43F BA2D 02B9 1800 CD21 33C9 33D2 B802
Trakia.570      B802 3DCD 2193 0E1F B43F BA36 02B9 1800 CD21 33C9 33D2 B802
Trakia.1070     9C3D 8842 7505 B888 429D CF3D 9942 751C 2EC6 061A 040B EB3F
```

Vlad-Dir **ER:** 653-byte virus containing the text: '[VLAD-DIR]' '[Darkman/VLAD]'. Infected files contain the signature 'VD' at offset 0012h.

```
Vlad-Dir        CD20 3EC7 8613 003E C7B8 0163 CD21 3BC3 7443 8CC0 488E D880
```

Vlad **CER:** Polymorphic, 1221-byte virus containing the text: '[VIP v0.01]'. It may delete the files MSAV.CHK, ANTI-VIR.DAT, CHKLIST.CPS, CHKLIST.MS. The template below finds it in memory and detects the first generation.

```
Vlad            7411 3C4F 740D 583D FECA 750A B80D F0CF E9CC 01E9 9B01 E989
```

Vlad.Daddy **CER:** An encrypted, 1093-byte virus which infects COMMAND.COM and contains the text: '[Incest Daddy]'. It infects the same files as the Vlad virus.

```
Vlad.Daddy      89E5 8B76 0083 EE03 33ED C351 B9E0 03D0 C8F6 D82E 3004 46E2
```

Vlad.Mummy **EN:** An encrypted, 471-byte companion virus containing the text: '[Mummy Incest] by VLAD of Brisbane.', 'Breed baby breed!'.

```
Vlad.Mummy      2EA0 D602 51B9 B001 3004 F6D8 D0C0 46E2 F759 C3
```

Vlad.Sister **ER:** An encrypted, 792-byte virus containing the text: '[Incest Sister] by VLAD -Brisbane, OZ'.

```
Vlad.Sister     E809 032E 89A4 0803 2E8C 940A 032E C784 1400 CD20 FA8C CA8E
```

TUTORIAL

On the Mac Front

Judy Edwards

Recently, more and more *Macintoshes* have become connected to the Internet, and the risk of virus infection has increased proportionately. By its very nature, the freedom engendered by the *Internet* makes it virtually impossible to police effectively. This article, making special reference to university environments, intends to clarify the risks created by this connectivity and associated increase in file transfers.

Internet Access

The easily-installable and freely available *Macintosh* telnet clients have made accessing *Internet* resources easier than ever. *POPmail* clients simplify mail transfers (and attached application and text files) between the *Macintosh* desktop and the *Internet*.

Ftp (File transfer protocol) clients, such as *Fetch*, eliminate the need to learn the command-line ftp process. 'Gopher' clients not only locate information on the *Internet*, but will automatically ftp and then install shareware on to your *Macintosh* as you work; the basic configuration takes place with initial use. The temptation to move files and applications across the *Internet* becomes almost irresistible. After all, the essence of computing is sharing information!

Should you resist? Consider this: added to the already seemingly uncontrollable melée is the temptation to distribute and accept files freely, without heeding where those files have been, or to which viruses or Trojan horses they may have been exposed.

University end-users are often unaware of the innate vulnerabilities involved in passing files to each other. An end-user who carelessly accepts an infected file could infect an entire *Macintosh* network, without even using the application. Infected applications do not need to be run to propagate viruses such as ZUC and MDEF.

Also, it is common practice for staff to transport email, and other personal configuration files, between their office and home computers. Indeed, students who must rely on public workstations have never had a choice in the matter. Keeping up to date with virus protection upgrades, and ensuring other means of protection is in place and operating appropriately, is an ongoing problem for a large university. Educating end-users to deal with a virus presence is crucial.

Much research has been done on *MS-DOS* viruses. Experts currently estimate the number of such viruses at about 5500 and growing; over the past few years, we have witnessed an enormous increase. While the same cannot be said of viruses for the *Macintosh*, the situation is potentially risky.

File Exchange

Files are passed across the *Internet* in numerous ways, with the help of telnet clients. The clients and INITs necessary for *Internet* access are freely exchanged between end-users. For example, where once a user may have had only one shareware file, they may now easily pass through thousands a month, increasing the probability of exposure to a virus.

Files are compressed, usually with a product (such as *BinHex*, *StuffIt* or *StuffIt Lite*) which has been downloaded from an anonymous ftp site. Compressed files are then stored at the local university's anonymous ftp site or on various *Macintosh* fileservers or public workstations where end-users are encouraged to take copies for personal use.

Users store client, or email, bookmarks and hotlists of *Internet* addresses for favourite sites for downloading, and subscriptions to UseNet newsgroups they read regularly on a non-write protected diskette. Further, trading hotlists and even clients with friends and colleagues has become a fad. By saving these files to disk, the end-user can easily access information resources they use frequently without leaving personal files behind when moving to another *Macintosh*.

Text files, which can contract INIT 29 or the MDEF A, B, or C strain, are often exchanged electronically as end-users collaborate on projects. They can easily be transmitted as *POPmail* attachments, in their original file format.

File Transfer Security

Transfer of files, which is only possible on a diskette which is not write-protected, is a risky process, and creates an environment ripe for passing viruses. Running an infected client from a diskette could infect other applications on that workstation, or the diskette could become infected if a virus is present on the *Macintosh* or on a server.

An ftp or *WWW* (*World Wide Web*) server may reside on a *Power Mac*, which could have *Macintosh* infections on it and, by association, pass those viruses on. The temptation to use a *Macintosh* for file transfers is based on the fact that files can be transferred via ftp or *AppleShare*, reducing the number of IP connections in use for downloads. The question remains as to how clean an application transferred via the *Internet*, and the *Mac* on which the application resides may be.

Take as an example the virus INIT 29, which although not contagious to other files, can infect a file which is passed on to another end-user. The virus would need to be destroyed just as though the original virus had been contracted on the recipient's machine. Also, one *Macintosh* Trojan horse, CPro 1.41, masquerades as *Compact Pro 1.41* and erases the System and floppy diskettes.

Accepting anything over the *Web* means that the end-user is ultimately responsible for maintaining the integrity of files they accept and use, as well as those they pass on. Staff and students, failing to understand the risks, seldom install even shareware virus protection on their own personal computers. They further assume that the departmental network manager or lab manager will maintain current server protection.

Virus Protection

The three most commonly-used *Macintosh* anti-virus products are *Disinfectant*, *Gatekeeper* and *Virex*. Each requires that an INIT (or Extension) be stored in the System folder. It is also necessary to open the application and configure the software. Many end-users skip this step, causing the virus protection to fail to scan disks in some cases, or to fail to notify the end-user when a virus has been detected or removed.

The ease of installing most *Macintosh* applications is contrasted by the fact that one of the more effective tools, *Gatekeeper*, does require some technical expertise to install properly. Even *Disinfectant* requires some manual installation and, if not configured properly, can fail to scan floppies.

Some products, such as *Virex*, must also be disabled when installing other applications, and then reactivated. *Virex* will reactivate automatically after a specified period of time, but until that time, protection is disabled. Additionally, all products require that the end-user properly understand and respond to a notification that a virus has been detected on their diskette.

It is possible to configure virus protection to omit the scanning of diskettes. Generally, people who choose this option assume that the applications are residing on the hard drive. With telnet clients, this may not be the case. Older *Eudora POPmail* versions, for instance, are small enough for the entire application, along with the end-user's mail, to reside on a diskette, leaving such users at risk.

The documentation included with the *Gatekeeper* package stipulates that certain files such as 'communications programs, compression utilities, and electronic mail packages require File(Other) privileges when decoding downloaded applications and system files'. Not assigning these privileges within *Gatekeeper* can cause problems with various telnet clients and helper applications, such as *StuffIt*.

Software in labs or within departments is often held on a network, and can be restricted by the network administrator. Applications cannot be multi-launched on *System 6* machines over a network while the *Virex* INIT is running. Universities often run multiple system versions, and many *System 6* machines can still be found. Some virus protection software may conflict with network users.

Storing applications centrally on a server is a wonderful way to ensure that every workstation has the latest version of an application. It also means that users can be unaware of upgrades which have been installed transparently for them. They may not realize that a new version of their virus protection is available, and that they should upgrade protection at home. If they are moving the data back and forth on a diskette, they move a virus between systems.

HyperCard

Disinfectant is one of the more commonly used, and more effective, anti-virus tools. However, it does not address such problems as Trojan horses, nor viruses which infect *HyperCard* hypermedia stacks.

While telnet clients themselves are not *HyperCard*-based, some helper applications for creating hypertext markup language documents for the *Web*, such as Simple HTML Editor, are *HyperCard* add-ons. Some products operate so transparently that the end-user may not realize they are using *HyperCard*. *Disinfectant* is not a fully effective solution for *HyperCard* users: this group will need additional protection.

> *"an end-user who unwittingly accepts an infected file could infect an entire Macintosh network"*

Virex protects against some Trojan horses as well as viruses. However, a *QuickMail* server will not transmit mail messages over a modem while the *Virex* INIT is running. The same is true for *Apple's PowerBook Express Modem* software for fax transmission. The solution is to disable the feature which diagnoses *HyperCard* stacks.

Risky Business

An ongoing problem for everyone is keeping current with the latest product releases. The problem is compounded by the fact that commercial vendors are often unprepared for listing multiple contacts and departments, as is the case in many American universities, as 'the' official contact point.

Through other means (such as trade publications), managers can discover that a new upgrade is available even before receiving an announcement. Universities sometimes also keep users informed of new upgrades via a university-based listserver or mailing list. John Norstad (author of *Disinfectant*) makes the following recommendations, among others:

- Join a user group, such as *BMUG* (*Berkeley Macintosh User Group*), *BCS* (*Boston Computer Society*), or a local user group
- Subscribe to the BITNET distribution lists VIRUS-L and INFO-Mac
- Read the UseNet newsgroups comp.sys.mac.announce and comp.virus

How to Protect Yourself

The old adage 'keep your disks locked' is no longer appropriate. It prevents clients operating. Users of public computer labs have no alternative but to store files on a diskette, or risk leaving their work behind on their workstation.

The first step in protecting yourself against the threat is to install current virus protection. Configure this to scan new files immediately. As new files or applications are downloaded and decompressed, they should be scanned before being used. *StuffIt Lite 3.07* allows the user to specify a virus scanner with which to check files. As a file is decompressed and installed, it is scanned, even if the virus protection is not currently set to scan every new file automatically.

It is a good idea to run a second scan, to ensure that you have eliminated the virus. End-users commonly run multiple applications simultaneously: if a virus such as INIT 29 is currently infecting one file, it is not unreasonable to assume that it might infect others as well.

Administering virus protection to networked users with a variety of needs and using a variety of System versions can become quite a task. It is crucial to know what your end-users are running, and to accept that they may not be fully aware of this, given the transparency of some applications.

After the Attack

It is necessary not only to disinfect a virus, but to know of its existence: this is as important to users as to system administrators. Protection should be configured to notify the user that a virus has been detected and removed, and which file or application was affected. If an error is encountered when reading or printing a file, a virus may be the cause.

The desktop should be rebuilt every time a new application is installed: a side benefit is that the WDEF and CDEF viruses can be removed by this simple process alone.

Protecting yourself also entails knowing how people work. Are they carrying telnet clients between workstations? Are they aware that there is a new upgrade available? Do they know how to install it? Do they have it configured to scan diskettes? Do they allow student workers to use telnet clients on diskettes to read their email from departmental computers, after using those diskettes at a public computer lab or from their home computer?

Other Methods of Protection

Checksum programs are available for the *Macintosh*, such as *Checksum* by Geoff Walsh. In addition, *Virex* has built-in checksums. If the checksum on a given file changes, it is possible it has become infected with a virus.

Checksum is compatible with most checksum programs on other systems, such as the *Unix* 'sum' program and the *POSIX*.2 'cksum', and can treat 'TEXT' file end-of-line characters to be compatible with *Unix* or *MS-DOS* systems. For example, if one of the files is on a *Unix* system while the other is on a *Macintosh*, the 'sum' command can be used to calculate a checksum for one of the files. This is an advantage, especially in view of the number of files which are transferred between platforms via the *World Wide Web*.

When using shared workstations, download a clean copy of virus protection and scan the machine before starting to work. Scan disks every time they have been used at a shared workstation to ensure that a virus infection did not occur.

Ftp clean copies of telnet clients from an anonymous ftp site. Do not accept shareware which someone is already using on a diskette or on multiple workstations. Scan software which has been pre-configured for your site.

Macintosh fileshare administrators should not allow Program Link fileshare access to the clients on a server. Running an infected client from the server can infect other files and applications. Storing a clean, compressed copy of the clients for download access only should reduce the likelihood of a virus infecting the server or being passed on to others. Be especially careful that viruses do not infect clients whilst being pre-configured for other end-users.

Concluding Notes

Many works were used as reference for this article, the most important being:

- Gordon, Sarah, *Technologically Enabled Crime: Shifting Paradigms for the Year 2000* (Proceedings, Sec 94, IFIP TC11). Curacao, Netherlands Antilles, 1994
- Harris, Kevin, *Virus Reference 2.1.3*, Software Perspective, 1994.
- Norstad, John, *Disinfectant 2.5.1, Northwestern University*, 1991.
- Schneier, Bruce, *Virus Killers: Macworld Lab tests Virus Software and Survives, MacWorld* July 1994, pp116-119.
- Spafford, Eugene, *New Macintosh Virus Discovered (INIT-29-B)*, April 1994, Purdue University, 1994.
- Virus-L, ftp.informatik.uni-hamburg.de

I would like to thank Sarah Gordon for countless hours of assistance in exploring similarities between DOS and *Macintosh* threats. It is my hope that collaborative work between researchers will lead to solutions to this ever-growing problem.

Judy Edwards is a Microcomputer Software Specialist at *Illinois State University*. She is responsible for providing *Internet* training and *Macintosh* help desk support to staff and open student computer labs, and is also an independent *Internet* consultant. She holds a Master's degree in Instructional Systems Technology from *Indiana University* and a Personal Computer Coordinator's certificate from the *University of Southern Maine*.

She may be contacted at jedwards@ilstu.edu, and her *WWW* homepage is http://www.ilstu.edu/~jedwards/Welcome.html

VIRUS ANALYSIS 1

WINSTART - Son of a Batch

Eugene Kaspersky
KAMI Associates

Whilst reading recent issues of *Virus Bulletin*, I have realised that the number of different methods of virus infection has grown several times over during the past few years. To list just a few, we have seen cavity viruses, link viruses, and source and object code viruses.

All these techniques are new, and could not have been imagined in the mid-1980s. Many variants of each method, and the techniques used to implement them, exist. The new memory-resident BAT file companion infector known as WINSTART is a good illustration of what may be done with a DOS-based system.

Installation

This virus is named after the WINSTART.BAT file in which the body of the virus resides. That particular file is only 297 bytes long, and contains the following four lines of text, followed by binary data:

```
@ECHO  OFF
:s%r#
COPY %0.BAT C:\Q.COM>NUL
C:\Q
```

On execution of this BAT file, the virus copies itself into the COM dropper, which it executes as a COM file. The dropper then installs itself into High Memory, hooks Int 2Fh, and creates WINSTART.BAT on floppy drives. Finally, the virus body is copied into the newly-created BAT file.

The BAT File

WINSTART's installation routine uses ideas in common with the first memory-resident BAT virus, known as BATMAN [*VB, March 1993, p.12*]. Just like BATMAN, WINSTART when executed receives control as a batch file containing a sequence of four DOS commands.

During execution, the virus disables echoing, and copies itself (i.e. the host BAT file) into the COM dropper, which is called Q.COM and placed in the root directory of the C: drive. It contains the same data as the source BAT file, but when it is run, it executes as a COM file.

The transformation of the source BAT file into a COM dropper, and execution of this dropper, is performed by execution of the batch commands:

```
@ECHO  OFF                    disable echoing
:s%r#                         label, no effect
COPY %0.BAT C:\Q.COM>NUL      copy host file to COM
C:\Q                          execute C:\Q.COM
```

The ASCII text part of the virus ends here: the next byte of the file is 1Ah, which signifies 'end-of-file'. This has the effect that execution of the BAT file stops after it has run Q.COM, and control returns to the parent program - almost certainly DOS, if the BAT file was run from the DOS prompt. A useful side-effect (for the virus) is that only the four plain text lines are seen if WINSTART.BAT is viewed with a standard DOS text editor.

The COM File

As the file Q.COM is a copy of WINSTART.BAT, it contains identical data. However, when it is executed it receives control as a COM file; i.e. the text strings are executed as *Intel* instruction codes, in fact as 'do-nothing' commands, such as:

```
INC AX          ; '@'
INC BP          ; 'E'
INC BX          ; 'C'
DEC AX          ; 'H'
DEC DI          ; 'O'
AND [BX+SI],AH  ; '  '
```

This useless code is terminated with the 'label' (that is, the second line of the BAT file), which is then executed as two separate instructions:

```
JNC Install     ; 's%' - jmp carry not set
JC  Install     ; 'r#' - jmp if carry set
```

Whatever the state of the carry flag, these instructions will always pass control to the binary part of the virus, which installs the memory-resident portion of WINSTART into system memory.

Going Resident

The virus' first action is to perform an 'Are you there?' call, using Int 2Fh with B700h in the AX register. The memory-resident part of the virus returns the value FFh in the AL register. This call is exactly the same as the one performed by the DOS utility APPEND on installation, which means that the two will confuse each other. However, it also shows that one way to protect yourself against the WINSTART virus would be simply to install the APPEND utility.

If the virus is not already resident, it will allocate a block of memory from the High Memory Area (HMA), using Int 2Fh, function AX=4A02h. The request is for a mere 173 bytes - even a hole of this size is large enough for the virus to install itself into memory.

It copies 168 bytes of its code into this memory block, and also stores the address of the current Int 2Fh handler (which occupies four bytes) there. The copied code contains only the virus' Int 2Fh handler and infection routine: the installation routine is not placed in the TSR segment of the virus.

The address of the Int 2Fh vector is taken from an undocumented DOS data area - in fact, the address is new to me. This address is used by DOS (versions 6.0 and higher) when its internal Int 2Fh routines pass control from one part of code to another.

The virus then performs the last part of the installation routine. This segment of the virus is executed even if the virus has failed to install itself into memory for some reason (for example, if there is no space available in the High Memory Area).

It renames the file C:\Q.COM to C:\WINSTART.BAT, and then deletes the file C:\Q.COM. If the file C:\WINSTART.BAT already exists, the rename command fails, and the delete command then removes Q.COM. After performing the deletion, the virus sets the attributes of C:\WINSTART.BAT to 'read-only', and terminates itself with an Int 20h call.

Int 2Fh Handler

The resident part of the virus intercepts two Int 2Fh functions. The first is an 'Are you there?' call (which clashes with the APPEND installation check) with AX=B700h. If the appropriate call is made, the virus immediately returns the value FFh in the AL register.

The second function is AX=AE00h, which is used by the file COMMAND.COM before every command is executed to see whether the command line is to be handled by a TSR other than COMMAND.COM itself. Whenever this is called, the virus receives control.

It then decides whether or not to 'drop' a copy of itself on one of the floppy drives. This it will only do if certain conditions are met. These are:

- if the current drive is A or B
- if the disk is more than 50% full (perhaps in an attempt to hide the new WINSTART.BAT file in the directory listing)

If it decides to infect, the virus hooks Int 24h to prevent the appearance of the DOS error message in the event that it tries to write to a write-protected disk. It then creates a new WINSTART.BAT file on the current drive, and copies C:\WINSTART.BAT there.

While making the copy, the virus uses the address of the Int 18h handler in the Interrupt Table (address 0000:0060-0063) as a read-write buffer. This interrupt is not used under DOS on any *IBM PC*-compatible (a technique allowing the virus to save four bytes of memory), but may cause problems on *NEC* clones.

After copying, the virus sets the file date/time stamp of the destination file to that of the source file, closes both the files and passes control to the original Int 2Fh handler. Infection is complete: the file WINSTART.BAT, containing the virus code, has been created on the floppy disk.

The Explosion of BAT Viruses

It seems that virus writers have started to turn their efforts to BAT infectors: until this year, such viruses have not been in the mainstream. WINSTART, I feel, is only the first of a flood of its type. Shortly after it was discovered, I received seven new BAT viruses, including a full stealth, multipartite (Master Boot Sector and BAT file) infector called BLAH.

One of these viruses, called BAT2EXE, uses the simplest technique for a BAT file virus - it utilises a file containing the following BAT commands:

```
FOR %%i IN (*.com) DO copy %0 %%i > nul
FOR %%i IN (*.exe) DO copy %0 %%i > nul
```

The file is converted into an executable by using the utility BAT2EXEC (by Douglas Boling). When an infected EXE file is executed, it thus overwrites COM and EXE files.

Clearly, the time has now come for anti-virus researchers to include the string '*.BAT' into the list of default file extensions used by their scanners.

WINSTART

Aliases: None known.

Type: Memory-resident, 297-byte companion BAT infector, uses 'worm' infection method.

Infection: Batch files only.

Self-recognition in Files:

Not necessary; see analysis for details.

Self-recognition in Memory:

'Are you there?' call using Int 2Fh with AX=B700h. TSR segment of the virus returns FFh in AL register.

Hex Pattern in Files:

in Hex:

```
4045 4348 4F20 204F 4646 0D0A
3A73 2572 230D 0A43 4F50 5920
```

in Memory (HMA):

```
3D00 B775 03B0 FFCF 3D00 AE74
03E9 9700 83FA FF75 F83A EE75
F460 1E0E 1FB4 19CD
```

ASCII Pattern in BAT Files:

```
@ECHO  OFF
:s%r#
COPY %0.BAT C:\Q.COM>NUL
C:\Q
```

Trigger: No trigger routine.

Removal: Delete the WINSTART.BAT files which contain the ASCII text strings listed above.

VIRUS ANALYSIS 2

Bua: Dangerously Obscene

Matt Brown
Sophos Plc

Bua is an example of how rapidly a virus can spread in the modern, highly-connected, electronic world - in this case, through an infected screensaver, COOLSAVR.COM, which was distributed in a collection of such programs, called BESTSSVR.ZIP. This archive found its way onto a number of very large BBSs in North America, which ensured its wide distribution - especially since many smaller BBSs automatically mirror areas of larger ones.

Once infected with Bua, the original file was compressed with PKLITE, and the PKLITE signature bytes were removed. This seems to indicate that the virus was placed there deliberately. It also makes it less likely that scanners will detect the virus within the PKLITE file: even if they are capable of checking inside files compressed like this, the removal of signature bytes means that scanners might not realise the file was compressed.

In a demonstration of the type of rapid communication made possible by the *Internet*, the virus was (largely thanks to Norman Hirsch, nhirsch@nha.com) in the hands of a good proportion of the world's anti-virus researchers within a day or so of its initial discovery.

Within a few days, these researchers had produced new recognition files for their products. Norman Hirsch then posted information to many security and virus-relating newsgroups and mailing lists about how to update a wide variety of anti-virus products, enabling them to detect the virus. One hopes this rapid response to the arrival of a new virus in the wild is the way of the future.

Infection Procedure

The code within the Bua virus which concerns infection is not very original at all; the virus is at heart simply a Vienna variant. However, it is wrapped in constant encryption and has a large payload, which accounts for about 1.6Kb of the overall 2.2Kb size of the virus.

Infection follows the usual path of a Vienna variant: first, it stores the original DTA (Disk Transfer Area), which it will restore later, and installs a new DTA so as not to wipe out the command line arguments destined for the host program.

Then it searches through the environment segment for the string 'PATH=', which precedes the search path. When this has been found, it calls FindFirst/FindNext (Int 21h, functions AH=4Eh and AH=4Fh) to find a file matching the pattern '*.COM' to infect, either in the current directory or on the path.

When it has located a matching file, the virus checks that the file is not already infected, by testing that the seconds part of the time stamp is 56 seconds or greater. Then, the virus ensures that the file is not COMMAND.COM. Once Bua has found a suitable file, it tests that the file is not an EXE file masquerading as a COM file, by checking the first two bytes of the file for the 'MZ' signature characteristic of most EXE files. If any of these tests shows that the file should not be infected, the code jumps back and attempts to find another candidate file.

If the prospective host passes all these checks, the virus moves to the end of the file, using Int 21h, function AX=4202h (Lseek). It then increments the generation count, re-encrypts itself in memory, writes its body to the end of the executable, and decrypts itself again. After this, it writes a jump to the new virus code over the first three bytes of the file, restores the file date and time (after changing the seconds field to 56), and closes the file. It then resets the file attributes, and restores the host program's original DTA.

The infection routine is called twice by the virus code, so Bua infects a maximum of two COM files each time it is run. The first infection will be one byte longer than its parent; the second infection, two bytes longer. Whilst this is not strictly a generation counter, it serves the same purpose for this virus.

"testing code which will destroy the data on your hard disk is not particularly easy"

Trigger

After infection, the virus' next action is to obtain the current date, which it does by using Int 21h, function AH=2Ah. If the date returned is before 5 May 1995, the virus will only replicate - that is, none of the trigger effects will be activated. This technique is used so that the virus has a chance to spread somewhat before its effects reveal that something is amiss.

Otherwise, the virus then 'fetches' the time (Int 21h, function AH=2Ch). Before 3pm or after 7pm, the virus takes no further action. Between those times, however, Bua may carry out any of a number of tasks.

The first routine it calls checks whether the size of the virus, the value of which is stored inside the virus body, has reached 2296 (as described above, this is incremented by 1 or 2 each time the virus replicates). If it has not yet reached this size, it skips straight to the visual effect routine described below.

The sample in the file COOLSAVR.COM which I received (believed to be the original executable from North American BBSs) had a length of 2273 bytes, meaning that the virus would have to go through a maximum of 23 (and a minimum of 12) 'generations' before it reached 2296 bytes in length. When it finally does reach this size, it creates four directories in the directory from which it was run, the names of which are all rather infantile insults.

Then, it retrieves the time again. The virus author seems to have intended his creation to format track 0 of drives C to Z, if the time given is within the first six seconds of the hour. However, the code is badly written, and in practice appears not to do much at all. It attempts to format the first track of disks 02-1Ah (the third and subsequent removable drives), with arbitrary parameters, and consequently arbitrary results.

After this attempt at formatting, it jumps to a routine which, after running the animation described below, writes junk to the first 800 physical sectors on the first fixed disk. This can destroy the boot sector, both copies of the FAT, the root directory and many of the system files on the first partition on this drive - on most systems, the boot partition. It then halts the machine by putting it into an infinite loop.

If, however, the current time is not in the first 6 seconds of the hour, the virus then checks to see whether the time stamp gives the time as the first 11 seconds of any minute, in which case it deletes all the files in the current directory.

Animation

Even if it carries out none of these effects, a text-based animation is displayed when an infected file is run between 3pm and 7pm, any day after 5 May 1995. The display consists of a crudely-drawn penis which moves in from the left of the screen, pulses to an accompaniment of frantic beeping, ejaculates, then moves off-screen. The words 'Big Caibua' flash beneath.

The words 'Big Caibua' also appear in a long text string within the virus, which is never displayed. Its usage in the text makes it quite plain that the term is offensive; much of the text is the standard juvenile rant about Patricia Hoffman and John McAfee (and why couldn't the virus author be a bit more imaginative in his choice of targets? Just about every American virus picks on them). This is followed by 'O.J. IS GUILTY!!', and then 'Tempest, Live for yourself... You do not know what love is yet... wait, it will come... No worries…'.

Such an obvious routine makes it unlikely that the virus will get far in the wild, now that its trigger date has passed. Users could hardly miss this display.

Conclusion

Bua shows how quickly a virus can acquire a whole cluster of names. Some anti-virus products have followed the virus author and identify the virus as 'Big Caibua', while others have used a cut-down form, 'Bua'. Still others have followed a suggestion made on the *Internet* by one early discoverer of the virus, and called it 'Butthead', in an attempt to describe its author.

That part of Bua which qualifies it as a virus, the infection routine, is old and uninteresting: the only difference from a standard Vienna variant is that the self-recognition date which the virus stamps on to infected files is a valid file date, meaning that infected files are not quite as obvious as would otherwise be the case.

The trigger routines, in particular the animated display, appear to have had a fair amount of effort put into them. However, some have obviously not been well tested - but then, testing code which will destroy the data on your hard disk is not particularly easy.

The author, like too many of his peers, displays a positive delight in destroying other people's work. The accompanying text messages serve only to reinforce the impression we get of its creator being a very immature individual.

However, it must be said that he proved very successful in distributing his virus far and wide before the trigger date was reached. Now that this date has passed, the effects of Bua are so obvious that the virus will probably vanish from the lists of common viruses in the wild almost as quickly as it arrived.

Bua

Aliases:	Big Caibua, Bua, Vienna.Bua, Butthead.
Type:	Parasitic, not memory-resident, encrypted but not polymorphic.
Infection:	COM files only.
Self-recognition in Files:	Seconds field of timestamp is 56 or greater.
Hex Pattern in Files:	`5053 5156 528B 849C 038B DE81 EBDE 048B D3EB 05?? ???? ????`
Trigger:	No trigger before 5 May 1995, or before 3 pm and after 7pm. Otherwise, always displays penis animation, and after about twenty generations may do any of the following: create directories in current DIR with dubious names, delete *.* in current directory, attempt to format first track of all drives (however, the code for this is broken), or destroy first 800 sectors of first fixed disk.
Removal:	Under clean system conditions, identify and replace infected files.

VIRUS ANALYSIS 3

Fancy a Quicky?

Dr David Aubrey-Jones
Reflex Magnetics Ltd

A few weeks ago, a virus landed on my desk, which appeared to be at large both in the UK and in Europe, and which many anti-virus scanners were unable to detect. Although reports such as this are becoming more common, they certainly do not happen every day.

In the week following this incident, a computer engineer came into our office with what proved to be another copy of the same virus. His whole hard disk had become infected, and he had been visiting client sites…

Mediocrity Rules

Most unknown viruses reported 'in the wild' prove to be boot sector viruses, but a quick check showed this to be a parasitic EXE infector. The virus, known as Quicky (although it contains no name internally) is a classic example of inept programming, and was not even fully debugged!

If the programmer who wrote Quicky had known what he was doing, it might have caused real trouble. However, it is flawed, and appears to contain code used solely for the purpose of debugging. An attempt has been made to incorporate a destructive trigger routine which would gradually corrupt writes to the hard disk. However, there is a stupid mistake within this code, so the author has rendered it inactive by bypassing it with a jump.

Let's take a closer look. Quicky is 1376 bytes long and only infects EXE files. The increase in file length can be spotted readily, as the virus uses no stealth techniques. It begins by modifying its own decryption routine: this has probably been added in an attempt to avoid heuristic detection.

After decrypting its code in two halves using a very simple byte-swapping XOR routine (it is not polymorphic), it re-modifies its decryption routine and patches its addressing to take account of its location in memory.

Going Memory-resident

Our writer then makes his first major error. He checks to see if the virus is already resident in memory, calling Int 21h with AX=C000h. This conflicts with some *NetWare* interrupt calls, so may result in an error message, and the infected program aborting: this is sure to give the virus' presence away.

At first glance it also appears that Quicky's 'Are you there?' call requires a certain value in the BX register, but closer analysis shows that this is only used to set an internal variable value, which is to be used by the fatally flawed destruction routine. If not already active (in which case the virus will quit and execute the host program), the virus now moves its code down in memory to overwrite the host program, then resets the stack prior to going memory-resident. Next, it hooks Int 13h and Int 21h, scans the environment to obtain the name of the host program, and executes it, prior to going memory-resident.

Infection

Quicky's method of infecting other programs is somewhat unusual. It monitors Int 21h for program execution. When detected, it uses the standard method of removing any read-only attributes, and opens the file. So far so good. But now, instead of the infection routine one would normally expect, it closes the file immediately, resets the attribute and allows the file to execute normally. So how does it infect?

This virus monitors and infects during a file close: the very act of closing the file in the above sequence will infect it. This is because the virus is re-entrant, and calls itself a second time when issuing DOS function calls.

> *"by the time the virus is found, considerable amounts of data may be corrupt"*

Since read-only attributes are not switched off by the virus when it intercepts a Close call [*No pun intended. Ed.*], the net effect is that a write-protected file will not be infected if it is just open, read and closed, but *will* be infected if executed. Write-enabled files will be infected in both cases.

The infection routine following a file close contains more examples of poor programming. During this process, it unlinks itself from both Int 13h and Int 21h. Presumably, the author was unable to pass calls to the original Int 21h handler using any other method. He also fails to trap the DOS critical error handler, so DOS error messages will be displayed if the infection process fails.

Checksummer Attack

Apart from file infection and restoration of the date and time stamp, Quicky contains another interesting section of code which deletes various checksum data files used by a number of anti-virus products, presumably to escape detection. However, again, the programming was bungled!

The checksum data files are only deleted from the current directory, which will of course not necessarily be the directory containing the program which has just been

infected. That is, the wrong checksum file may be deleted! Under these circumstances, the virus will still be detected by the checksummer.

Disabled Destruction

The only reason for hooking the low-level disk interrupt, Int 13h, would appear to be to install its destructive trigger routine, which, as already stated, contains a major programming error, that the author was presumably unable to fix.

If it functioned correctly, the trigger would corrupt a byte on a random basis during sector writes to the hard disk. By the time the virus is found, considerable amounts of data may be corrupt, including backups made when the virus was active.

The only part of the Int 13h hook which is operational is a check for a sector read of track 0, cylinder 1, sector 1 of the first or second hard disk - this area usually contains the partition boot sector. When such a read occurs, the virus stores byte 35h of the boot sector. This was probably used as a switching flag by the disabled destruction routine, inserted by the author so that he didn't destroy his own hard disk!

Conclusions

Luckily, the author's programming ability did not match up to his destructive tendencies. I have no idea how widespread this virus is, but it would be wise to ensure that your scanner is up to date, or to use an alternative form of protection. Use the scanner string given in this article, or contact your anti-virus supplier to make sure this virus is detected with the current release of their software.

Quicky

Aliases:	Quicksilver.1376, V.1376.
Type:	Memory-resident parasitic, encrypted file infector.
Infection:	EXE files.
Self-recognition in Memory:	'Are you there?' calls with Int 21h, AX=C000h. The memory-resident code returns AX=76F3h.
Hex Pattern in Files and Memory:	`812E 1500 0714 812E 1700 1305 8A26 0C00 BE60 008B FEB9 E603`
Intercepts:	Int 13h and Int 21h. Infects on program execution and file close.
Trigger:	None.
Removal:	Under clean system conditions, identify and replace infected files. Can be de-activated in memory by calling Int 21h with AX=C001h.

TECHNICAL NEWS

New Technical Editor for VB

After the departure of Fridrik Skulason from the post of Technical Editor, the chair has been filled by Jakub Kaminski. *VB* has kept to its tradition of keeping the position an international one: Kaminski works for the Australian company, *Cybec Pty*, well-known for its anti-virus products. He has these words of introduction:

'I graduated and received an MSc in Electronics (Computer Measurement Techniques) from *Warsaw Technical University* in 1986. I went on to work for the *Institute of Fundamental Technological Research*, at the *Polish Academy of Sciences*. I spent six years in the *Laboratory of Aeroacoustics* working on a range of projects, such as developing new measurement techniques and applying noise control engineering methods in practice. My most unusual project included two days on a research farm spent recording the sound effects of the social behaviour of domestic geese.

'At the *Institute*, my main tasks were to program systems; collecting, processing and visualising measurement results. My university project, a program to support the IEC-625 interface, was very useful. I rewrote it a few times early in my career for different systems, in different assemblers.

'During this time, I also had a short but unforgettable romance with transputers and transputer assembly programming. Although the work was not a commercial success, we had months of exciting evenings in the basement.

'In 1992 I moved to Australia, secured a job at *Cybec* and had my first serious taste of viruses. Triggering a destructive virus and having to recover my system taught me an important lesson and introduced me to data recovery.

'In addition to programming and disassembling viruses, I spent over a year helping out in technical support, which has made me more aware of virus problems from the user's viewpoint. Gathering knowledge and expertise, and applying it in a manner that helps someone, is a rewarding task.

'Now I spend the majority of my time analysing new viruses, incorporating them into *Cybec's* program and maintaining our virus collection and database. I provide virus information when necessary and I am still taking a small part in our data recovery service.

'Becoming Technical Editor is an honour, but also a huge responsibility; first because of the reputation of the magazine; second, because of Fridrik Skulason, whose expertise and reputation I truly respect. This makes my work more difficult, but presents an enormous challenge, which I hope to meet. I look forward to working with *VB*, and hope also to receive suggestions and comments from you, the readers.'

FEATURE

Viruses on Windows 95

Watching viruses at work in their native environment is all very well, and it is that which forms the bulk of work in the field. However, it can be considered more interesting to change that environment - to move the goal-posts, as it were.

Following *VB's* recent voyage of discovery into viruses on *Windows NT* [*see March 1995, p.10*], a pre-release version of the latest offering from those nice people at *Microsoft - Windows 95* - was obtained. Dusting off the trusty *Opus 386/25* (the very same one which groaned under the weight of *NT* a few months ago), it was duly installed.

The Fun Begins

We had a general idea of what might happen when we started playing with boot sector viruses, so two start-up configurations were created: one perfectly standard, with both the hard and floppy disk drives using 32-bit drivers, and another with 32-bit access disabled for both devices.

The viruses used for testing were all boot sector viruses, as these are the most prevalent in the wild today. The test-set was selected not only because the viruses used are fairly common, but also because of the varied (or at least as varied as is possible for this type of virus) techniques they employ.

In order to keep this study simple, it will be divided into three main parts - what happened, why that happened, and how to clean the machine afterwards. In this case, the 'why' will be much lengthier than the 'what' and the 'how'.

Part 1: What Happened

The viruses used for testing were AntiEXE.A, Form, JackRipper, Jumper, NYB, Parity_Boot.A, Parity_Boot.B, Peanut, Sampo, Stoned.Empire.Monkey.A, Stoned.Empire.Monkey.B, Telefonica.A, and V-Sign. All except one produced the same results, as follows:

- If the virus has stealth capabilities, they work perfectly well, as seen by a DOS scanner
- In standard 32-bit mode, there is no infection of floppies
- In 16-bit disk access mode, floppy infection takes place
- If only a command prompt is started (i.e. no graphical user interface), the viruses will infect floppy disks, whichever level of disk access is chosen

It should be noted that even the Stoned.Empire.Monkey variants, which are notable because they encrypt the Master Boot Sector (MBS), allow the system to function perfectly well when they are resident. The odd virus out, so to speak, is Jumper - under test conditions, it was not possible to make this virus infect floppy disks on a *Windows 95* system.

Part 2: The Reasons

In order to work out what is happening deep within *Windows 95* which causes the viruses to function as they do, it seemed appropriate to start with the exception - Jumper. One thing is glaringly different about this virus: it intercepts Int 21h. All the others use Int 13h to infect the boot sectors of diskettes [*see VB, March 1995, p.11*].

Why is this significant? Int 21h is a DOS interrupt (despite the name 'DOS interrupt', Int 21h is still with us in *Windows 95*), whereas Int 13h is a BIOS interrupt. Int 13h is the lowest level interrupt by which the disk drives can be accessed, and is present before DOS loads, which explains why most boot sector viruses hook it to carry out infection.

"if a virus has hooked Int 13h away from the BIOS, it believes it cannot use 32-bit disk access, and reverts to DOS real-mode drivers: that is, it uses Int 13h"

Windows 95, like *Windows 3.11*, uses 32-bit disk access by default [*For a full explanation of the significance of 32-bit disk access, see VB, May 1995, p.18*]. For the moment, suffice it to say that it bypasses Int 13h and accesses the disk by talking directly to the disk controller. This should mean that viruses which have hooked Int 13h will not see any calls to it after *Windows 95* has loaded, as all these calls will be routed through the 32-bit disk access subsystem.

But this is not the case. If it were, the virus' stealth capabilities would not work - the traps they install to redirect disk reads would also be bypassed. They are not.

As with *Windows 3.1* or *3.11*, the presence of a virus hooking Int 13h prevents 32-bit disk access from loading correctly. At some point in the boot process, *Windows 95* traces the Int 13h vector to find out if it can access the hardware of the disk drives directly. If a virus has hooked Int 13h away from the BIOS, it believes it cannot use 32-bit disk access, and reverts to DOS real-mode drivers: that is, it uses Int 13h.

It is the presence of some form of 32-bit disk access which causes the effects we see. But what about Jumper? As stated above, Jumper hooks Int 21h, an interrupt which is almost

completely bypassed by *Windows 95*. Calls to such interrupts are intercepted by the kernel, and 'bounced' away from real mode (DOS) code, up into protected mode code. Here, they can be processed more quickly, without the processor having to switch modes. It is evident that those functions of Int 21h intercepted by Jumper are never reaching the virus.

Int 13h in Windows 95

If a DOS application makes an Int 13h call, a complex series of operations is performed by *Windows 95*. Andrew Schulman's excellent book 'Unauthorised *Windows 95*' gives this illustration of the route taken by such a call:

```
Int 13h in DOS program
 >  VMM         (Virtual Machine Monitor)
 >  VFD         (Virtual Floppy Device)
 >  BLOCKDEV/IOS
 >  Any Int 13h hook in DOS -> ...
 >  Handler installed with Int 2Fh, AH=13h
 >  BLOCKDEV/IOS
```

It should be noted that Int 2Fh function AH=13h is an undocumented method used to locate the bottom of the Int 13h chain - beneath IO.SYS, just before the chain enters the BIOS.

However, if Int 13h has been hooked away from the BIOS *before* IO.SYS has loaded (for example, by a boot sector virus), then Int 2Fh, function 13h, will return the address of the virus' Int 13h handler. There is nothing magic about Int 2Fh, function 13h: when IO.SYS loads, before adjusting the Int 13h vector, it stores the current vector at 0070:00B4.

Now we must look at this *Windows 95* boot process. When the MBS code loads the partition boot sector into memory and passes control to it, this then loads IO.SYS. This file serves the same purpose as (and indeed is almost the same as) IO.SYS and MSDOS.SYS in *MS-DOS*.

IO.SYS loads some memory drivers and processes the registry, loads the Installable File System Helper Driver (IFSHLP.SYS, driver name IFSHLP), and runs WIN.COM. This loads VMM32.VXD (containing the 'guts' of *Windows*), which then starts a huge number of drivers in .386 and .VXD files, amongst them the 32-bit disk drivers.

Somewhere in the *Windows 95* boot process, Int 2Fh, function 13h, is used to locate the BIOS Int 13h handler. But, as we know, the virus has hooked Int 13h away from the BIOS, so *Windows 95* obtains the address of the virus. It will then go on to examine the handler, believe it has a very obscure system, and refuse to load the 32-bit disk drivers.

It is reasonable to assume that calling Int 2Fh function 13h from a DOS box within a *Windows 95* GUI would return an address of the 'lowest' level Int 13h handler: in the case of an infected machine, this should be the virus. In fact, the system crashes, as the VMM (Virtual Machine Monitor - see below) does not appear to instance this piece of data in the VMs (Virtual Machines) which it creates and controls.

Virtual Machine Monitor

The VMM (Virtual Machine Monitor) acts as a sort of overseer for the system - in a sense, it should be called the operating system. It runs VMs, in which the applications run. The VMM takes care of multi-tasking, and provides low-level services to applications and what we know as the operating system (i.e. *Windows 95*) - a similar concept to *NT's* microkernel.

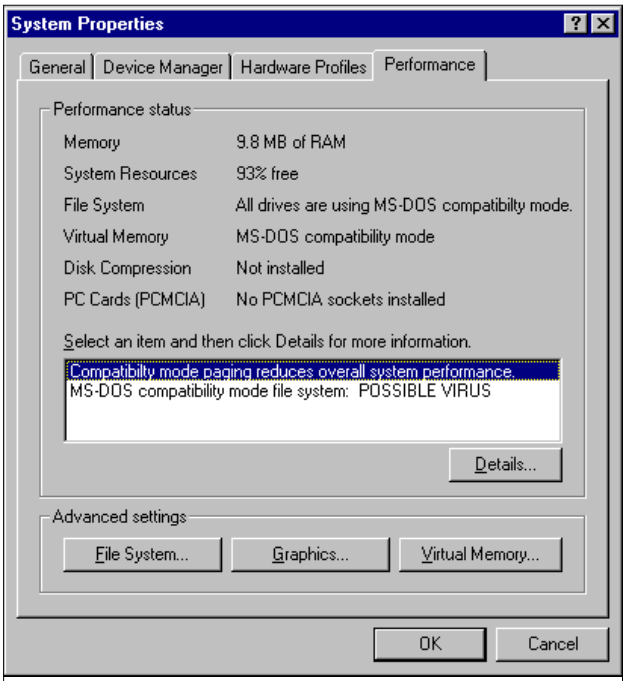
What happens when an application (specifically, a DOS virus scanner) is run is important. When we open a DOS box, it is the job of the VMM to create a façade for it - the pretence that it is the only thing running, and that it is a DOS session, controlling the machine. One hopes that this is not actually the case, but applications to run in that box must not become confused by anything else.

When the DOS scanner calls Int 13h function AH=02h to read the boot sector, it is unaware of all the monkey business beneath that call (which it thinks is going straight to the BIOS). The VMM has 'pulled together' all the oddities going on inside the OS for the application to see.

Here, the application is completely unaware of whether its Int 13h call goes to the BIOS, to the 32-bit disk driver, or to a virus. It is not told, and it does not need to know.

Part 3: Cleaning Up

After all this, the precise difference between the 32-bit drivers failing to load, and being told not to, is not completely clear. To recap, this is important because the viruses



How *Windows 95* tells the user that the machine has become infected. This display is accessed from the Control Panel, under the 'System' icon.

behave differently in each situation. It seems that some part of the 32-bit drivers is remaining active, even if unable to perform a full role.

The Hidden Truth

Perhaps even more peculiar than the behaviour of the viruses is that of *Windows 95*. Unlike *Windows 3.1* or *3.11*, *Windows 95* (at least the pre-release version on which the tests were done; version 4.00.347) gives no warning that any of the 32-bit drivers have failed to load. A little odd, one might think - an important part of the system fails to load, and *Windows 95* keeps quiet about it?

To discover that these drivers have failed to load, one must look at the 'System' section within the Control Panel. This brings up a tab dialogue with four property pages, labelled General, Device Manager, Hardware Profiles, and Performance. If the page headed Device Manager is selected, and the section of the tree concerning disk drivers is examined, it will say: 'The device is not functioning correctly' (referring there to the driver rather than to the physical drive).

But this is not all. If the Performance page is examined, exactly what *Windows 95* knows is revealed. 'All drives are using *MS-DOS* compatibility mode,' it says. But here comes the real killer: '*MS-DOS* compatibility mode file system. POSSIBLE VIRUS'.

It says this with all of the viruses used, bar one - Jumper again. This is because, as discussed before, Jumper is a virus which hooks Int 21h. It does not sit in the Int 13h chain and interfere with 32-bit disk access.

Disinfection

Of significant interest to the user is clearly the subject of how to remove a boot sector virus from a *Windows 95* system. On the subject of removal, once again, *Windows 95* differs markedly from *Windows NT*, and bears a very close similarity to DOS.

When you install *Windows 95*, you are offered the chance to create a 'Rescue diskette'. Do not pass up this opportunity. The Rescue diskette is similar to a DOS system disk. With it, you can boot the operating system past an infected boot sector to a DOS prompt. If a 'VER' command is executed, the reply '*Windows 95*' comes back; but for all intents and purposes, it is a DOS prompt.

In the event of an infection, this Rescue diskette should be used to boot the machine to the command prompt. At that point, a DOS anti-virus product may be used to clear the infection, or standard techniques can be used to eradicate it (for example, replacing boot sectors with clean copies using a disk editor).

Under some circumstances, the user may run across problems with the system complaining about direct disk access. It was not possible to identify precisely when these occur. If such a message does appear, the internal *Windows 95* command 'LOCK' should be used. This allows an increased level of direct access to the disks, and it may well allow the problem to be fixed.

However, as is always the case with boot sector infections on 'other operating systems' (referring here to operating systems other than DOS), there is a another, probably more flexible, option. Keep a clean DOS boot diskette: in the event of disaster, this can be used to boot your machine. Then DOS tools can be used in their native environment to fix the problem.

A word of warning: it may not be a good idea to use the DOS 'SYS' command to clean partition boot sector viruses. Partition boot sectors for *Windows 95* system drives are different from their DOS counterparts. There is a reference to 'WINBOOT.SYS' at the end of the sector, which is strange, as this file no longer appears to exist. For some previous releases (more accurately, previous pre-releases) of *Windows 95*, it seems to have been another name for IO.SYS. It may be that for the full release, *Microsoft* will go back to using it.

"users will be able to go on using Windows 95 systems which have become infected with boot sector viruses"

After the infection has been successfully removed, it is just as important as with a purely DOS system to boot cleanly again, then check that the infection has really gone. If the user boots *Windows 95* from the fixed disk to check, and the virus is still present, its stealth abilities will hide it.

Conclusion

One thing is clear - users will be able to go on using *Windows 95* systems which have become infected with boot sector viruses. This is, of course, not recommended, but is nevertheless possible. However, they should notice a degradation in system performance, and will probably not pass the infection on to floppy disks under normal circumstances. It is by no means impossible, however, that they will not notice.

For example, what if the users have Telefonica.A? Four hundred reboots later, the virus activates, and wipes their hard disks. In some respects, users' chances of discovering that they have this virus are seriously reduced - no longer are their colleagues in other companies receiving infected diskettes from them, which these colleagues might scan and report the infection back to its source.

It seems likely that once a boot sector virus has infected a *Windows 95* machine, it will stop there. This simple fact alone will seriously limit the spread of boot sector viruses.

PRODUCT REVIEW 1

SWEEP for Windows NT

Jonathan Burchell
JC Designs

This month we look at *SWEEP Virus Detection for Windows NT*, from *Sophos*, whose *SWEEP for DOS, OS/2* and *NetWare* products are already well known to readers cognizant with packages on the market today.

Despite initial scepticism from industry pundits, it seems that *Windows NT* is here to stay. It is available in two versions, *Windows NT Server* and *Windows NT Workstation*, targeted respectively at server and high-end client markets.

The operating system is rapidly gaining acceptance in many corporate environments as the platform of choice. It is seen not only as an upgrade path from DOS and *Windows* to a powerful 32-bit multi-tasking environment, but also as a more stable platform on which to run existing applications.

NT offers not only the traditional server approach to networking, but also built-in peer-to-peer networking - 'workgroup computing', as *Microsoft* likes to refer to it. This makes it extremely easy for users to share disks and files - attendant to this is the risk of virus infections passing rapidly through the network.

Windows NT itself blocks direct access to disk hardware (and most low level file I/O), so in a pure *NT* environment many, although not all, viruses would have trouble replicating. In reality, however, most *NT*-based networks will still involve DOS and *Windows* workstations using the *NT* systems as file servers; and for these machines the situation is no different from any other DOS/*Windows* environment.

If undetected, infected files loaded from the server will be able to execute and release any accompanying infection or payload unhindered. *SWEEP for NT* attempts to counter this by offering native 32-bit virus scanning on the server. As a native *NT* program running on the server, the scanner is protected against conventional viral stealth techniques and cannot be fooled by attempts to prevent file inspection.

Product Presentation and Installation

The package includes the scanner for *NT* systems, together with a copy of *SWEEP for DOS* and the *InterCheck* client, which provides real-time protection for DOS and *Windows* workstations.

The documentation for the product is extremely good. In addition to the user manuals for these products, a copy of the *Sophos Data Security Reference Guide 1994/95* is supplied. [*Since this review was written, the 1995/6 Data Ref. Guide has been printed, and is being shipped with the product. Ed.*] This 380-page book includes a substantial tutorial on computer viruses, and provides a great deal of useful information which should help administrators to understand the virus problems, deal with outbreaks and, most importantly, set up a good anti-virus strategy.

SWEEP for NT is supplied on a single 3.5-inch high-density diskette. Installation consists of logging into the *NT* server as an administrator and running the set-up program from the floppy disk - this invokes a fairly standard *Windows*-type install. Should anything go wrong, the documentation describes how to carry out a manual installation.

During installation, a program group is created with a single icon to invoke *SWEEP*. *Windows NT* is, of course, available on several platforms other than *Intel*. The version reviewed was that for *Intel* - *Sophos* is soon to release binaries for the *DEC Alpha*, and claims that supporting any platform is simply a case of recompiling and commercial justification.

Operation

SWEEP for NT provides on-demand, scheduled and real-time scanning. The real-time scanning is different from most products in that it relies on the workstation having loaded the *InterCheck* component of the *Sophos* package. *SWEEP* and *InterCheck* then co-operate with each other to provide the clients with real-time protection.

SWEEP for NT is a native *NT* application, but it runs at the command line prompt (all the icon in Program Manager does is to invoke a CLI session and pass into it the specified *SWEEP* command line). There is no configuration utility, and all parameters are passed to *SWEEP for NT* via the command line, whilst complex area definitions can be specified in an external file; SWEEP.ARE.

Most output from (and interaction with) *SWEEP for NT* is via the CLI session, but on some occasions it will choose to display a *Windows*-type dialogue box. This mixture of some command-line and some GUI-based interaction is not hard to work with, but does produce a slightly strange feel to the software, reminiscent of early *OS/2* programs.

On-demand Scanning

On-demand scanning consists simply of invoking *SWEEP for NT* with chosen command-line options. Several customising options are available: which areas of the file system to sweep, automatic disinfection, virus removal, quick sweeping, when to run, and compressed files, among others.

As well as parameters allowing the user to specify which files and directories to include or exclude from the scan, options exist on all versions of *SWEEP* to specify logical and physical areas of the disk to be checked. Although it is

cumbersome to specify complex scanning instructions via the command line, the options provided are comprehensive and should enable even most complex scans to be specified.

Automatic disinfection is activated by using the -DI option, and is sensibly limited to repairing the boot sectors of infected hard and floppy disks, when it is known that a repair can be 100% successfully achieved.

The -REMOVE and -REMOVEF options cause *SWEEP for NT* to delete infected items. The -NOC option controls whether the product may do so without asking the user first. The only difference between these two options is that -REMOVEF limits itself to files, whilst the more powerful -REMOVE option will also attempt to disable any viruses found in boot sectors which *SWEEP for NT* cannot disinfect.

By default, the product only checks those parts of a file likely to contain an infection. The -F option makes it check the entire file. The test results were obtained with *SWEEP for NT* in the default quick mode - and somewhat beg the question as to the benefit of enabling full mode.

Some users will want to invoke a scan from their login script. Two options can be used to optimise this further. The -DE option ensures that *SWEEP for NT* is only executed once per day, whilst the -D option allows execution to be limited to specific days, or alternatively assigns a given percentage chance that the scan will occur. For instance, specifying -D=10 tells *SWEEP for NT* to run on average 10 times out of every 100 opportunities for execution.

It is possible to specify the priority of the scan as either the same as all applications, higher than any application or lower than all applications. The 'lower than all applications' setting can be used if *SWEEP for NT* is set to perform continual background scanning.

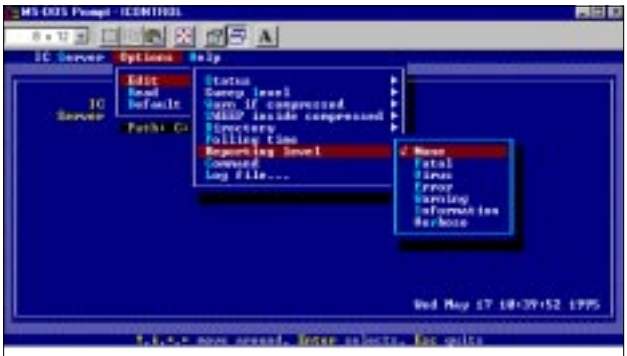
The product is now able to scan inside files compressed with the Diet, PKLITE and LZEXE dynamic compression utilities. Alternatively, a warning can be generated whenever such files are found.

Messaging and Reporting

Whenever the product detects a virus, a standard message is displayed on-screen. If a file called SWEEP.MSG is present, its contents will be used instead of the standard message. SWEEP.MSG is a plain text file and can thus easily be customised. In addition, the -FM parameter allows a different filename for the message file to be specified.

SWEEP for NT sends some events (such as virus detection) to the *Windows NT* application log. This is a neat trick, as the built-in *Windows NT* event log viewer provides sophisticated viewing, filtering and selection techniques.

Additionally, events may be sent to the application log on a different machine, allowing one machine to act as the message store for all copies of *SWEEP for NT*. The product is able to mail copies of the 'Scan report', using *MS Mail*.



SWEEP for NT offers various options, including specification of when to scan, and the type of report message to be issued.

New Patterns and Scheduled Scanning

Between monthly updates, new patterns can be specified via a SWEEP.VDL (binary) or SWEEP.IDE (7-bit ASCII) file: these are generated by *Sophos*. It is also possible to specify a search pattern in ASCII hex and to place it in the file SWEEP.PAT. A missing option is the ability to move suspect files to a quarantine directory, or leave them where discovered but rename them.

Scheduled scanning options are identical to those described above: this is because scheduled scanning is achieved using the *Windows NT* built-in 'AT' command which runs a program at a specified time. The *Windows NT* scheduler copes with multiple outstanding jobs, so almost any conceivable pattern of scheduled scanning can be created.

Real-time Protection

The *Sophos SWEEP* system provides real-time protection for DOS and *Windows* workstations by using client/server architecture. A workstation to be protected loads the *InterCheck* client, and thereafter any file which is loaded at the workstation is checked by the *InterCheck* client against a list of 'verified' files. This check uses cryptographic fingerprinting of the file and as such is extremely reliable.

If the requested file is in the database of verified files, loading proceeds normally. If not, the file is transferred to the server where *SWEEP* is located. *SWEEP* then checks the file and instructs the *InterCheck* client either to proceed, or to abort the load because the file is infected with a virus.

This system is extremely reliable and of course always gives the same results for real-time and scheduled scanning, and results in minimal overhead for checked files.

All versions of *SWEEP* can act as the server, by using the -ICS option. That the DOS version can act as an *InterCheck* server means that even networks with servers running unsupported operating systems (e.g. *Banyan Vines*, or UNIX hosts) can include full real-time checking, simply by running a copy of *SWEEP for DOS* on the server within a DOS emulation session or on a DOS workstation. The only requirement is for *InterCheck* clients and the *SWEEP* scanner to see at least one file area in common.

The client/server relationship of *SWEEP* and *InterCheck* is based on swapping information at a filing system level. This makes the process wonderfully independent of network protocols, and able to work in any environment which provides DOS level filing systems between the server and client (which is usually a given, with a network file server!).

It also means that in environments where a native version of *SWEEP* is not available for the server (UNIX or *Banyan Vines*), it is still possible to provide full execute-time protection by running a copy of *SWEEP* on either a *Soft-PC* box on the UNIX server or on a dedicated workstation. The only requirement is that *SWEEP for NT* and the *InterCheck* clients can see each other via a shared network directory.

New with this release is *ICONTROL*, a program which permits remote administration and control of the *InterCheck* server process, allowing control of various options, e.g. scan level, and whether compressed files should be searched.

ICONTROL allows editing of the current setting for *SWEEP* and client interaction. It sets them up not on a 'per client' basis but rather on a global basis for all *InterCheck* clients using the services of *SWEEP for NT* on the server. From within the *ICONTROL* shell it is possible to administer all the servers visible to the workstation, allowing a type of administrative domain to be established.

Most of the standard command-line options to *SWEEP* can be dynamically altered from within *ICONTROL*, including search type (Quick or Full), whether or not to search inside compressed files (or to warn if compressed files are encountered) and reporting level (Fatal, Virus, Error, Warning, Information or Verbose). It is also possible to set the location of a quarantine directory and to specify a DOS command to be executed when a virus is detected.

InterCheck offers complete real-time protection for DOS and *Windows* users; however, no *InterCheck* client is available for *Windows NT* workstation users. *NT* users are also protected to some degree from DOS and *Windows* viruses (which could be executed in an *NT* DOS session) by *NT*'s blocking of low level file and hardware access. In addition, some measure of *NT* workstation protection can be provided by scheduling frequent *SWEEP* scans of the workstation.

Conclusions

SWEEP for NT offers the same excellent level of virus protection as *SWEEP for NetWare*. Used in conjunction with *InterCheck*, workstations should have complete protection against infection. However, there are issues concerning real-time protection for clients running operating systems other than DOS which must be resolved, and the interface to *SWEEP* should be changed to a GUI-based system.

These considerations aside, it can be seen from the table at the end of this article that *SWEEP for NT* scored 100% in every test-set. Such results are not to be sneered at, and once *Sophos* has polished the product to its own satisfaction, it has the potential to remain a market leader.

Sweep for Windows NT

Detection Results:

NT Scanner:

Standard Test-set[1]	230/230	100%
In the Wild Test-set[2]	126/126	100%
Polymorphic Test-set[3]	4796/4796	100%

DOS Protection:

Workstation protection is provided by means of a TSR component which passes new files and boot sectors to the server for checking. If they are considered clean of viruses, a checksum is added to a database of known clean files.

Technical Details

Product: *Sweep for Windows NT.*

Version Number: *2.72b.*

Developer: *Sophos Plc*, 21 The Quadrant, Abingdon Science Park, Abingdon, Oxfordshire, OX14 3YS, England.

Contact Numbers: Tel +44 1235 559933; fax +44 1235 559935; BBS +44 1235 559936.

Price: Single user licence, £295.00; single server licence (up to 25 PCs), £495.00; single server licence (25+ PCs), £895.00. Site licence, and other prices, available on request. Server price includes DOS executables and *InterCheck* client-server checking for DOS workstations. All prices include 12 monthly updates and 24-hour technical support line.

Hardware used: Client machine - 33 MHz 486, 200 Mbyte IDE drive, 16 Mbytes RAM. File server - 50 MHz 486, SCSI hard disk, *NT 3.51*, 16 Mbytes RAM.

Each test-set contains genuine infections (in both COM and EXE format where appropriate) of the following viruses:

[1] **Standard Test-set:** As printed in *VB*, January 1995, p.19.

[2] **In the Wild Test-set:** 4K (Frodo.Frodo.A), Barrotes.1310.A, BFD-451, Butterfly, Captain_Trips, Cascade.1701, Cascade.1704, CMOS1-T1, CMOS1-T2, Coffeeshop, Dark_Avenger.1800.A, Dark_Avenger.2100.DI.A, Dark_Avenger.Father, Datalock.920.A, Dir-II.A, DOSHunter, Eddie-2.A, Fax_Free.Topo, Fichv.2.1, Flip.2153.E, Green_Caterpillar.1575.A, Halloechen.A, Helloween.1376, Hidenowt, HLLC.Even_Beeper.A, Jerusalem.1808.Standard, Jerusalem.Anticad, Jerusalem.PcVrsDs, Jerusalem.Zerotime.Australian.A, Keypress.1232.A, Liberty.2857.D, Maltese_Amoeba, Necros, No_Frills.843, No_Frills.Dudley, Nomenklatura, Nothing, Nov_17th.855.A, Npox.963.A, Old_Yankee.1, Old_Yankee.2, Pitch, Piter.A, Power_Pump.1, Revenge, Screaming_Fist.II.696, Satanbug, SBC, Sibel_Sheep, Spanish_Telecom, Spanz, Starship, SVC.3103.A, Syslock.Macho, Tequila, Todor, Tremor (5), Vacsina.Penza.700, Vacsina.TP.5.A, Vienna.627.A, Vienna.648.A, Vienna.W-13.534.A, Vienna.W-13.507.B, Virdem.1336.English, Warrior, Whale, XPEH.4928.

[3] **Polymorphic Test-set:** 4796 genuine samples of: Cruncher (25), Uruguay (75), Satanbug (100), MutationEngine (500), Girafe (1024), One_Half (1024), Pathogen (1024), Smeg (1024).

PRODUCT REVIEW 2

ALERT to the Risk

Dr Keith Jackson

Virus ALERT is a DOS anti-virus program marketing itself as 'a comprehensive package which offers quality high-tech features'. It operates under *Windows* and/or *OS/2*, and describes itself as 'completely compatible with all known networks' - I did not test the *OS/2* or networking claims.

Media

The product was provided on one 3.5-inch (1.44 Mbyte) diskette. I was stumped by finding two seemingly identical floppies in the package, but this was explained by the vendor as an oversight - two versions had accidentally been included. It transpired (for reasons which are beyond me) that the correct disk was not the one in the sealed envelope!

The diskette is not labelled with a version number for the product - the vendors would be well advised to alter this. Such an omission may make their life easier, and make it possible for retailers to unload old copies on unsuspecting purchasers, but makes life a pain for everyone else.

Installation

When installation commences, an initial menu containing six choices is offered. The user may install the complete package or just the memory-resident program component. Alternatively, he may choose to install a scan which takes place whenever the computer is rebooted.

From the initial menu, it is possible to make a recovery disk and/or install icons for *Windows* and *OS/2*. *Virus ALERT* is a DOS program: all that happens if *Windows* is selected is that icons are placed in the Program Manager. As yet, there is no *Windows* (or *OS/2*) -specific version of *Virus ALERT*.

To assist me with installation, I requested the Preview help section. This was a bad move: I was confronted by a blank screen. Much experimentation, and trying *Virus ALERT* on a computer with a colour screen, showed that this was caused by *Virus ALERT* using royal blue characters on a cyan background. On my *Toshiba* laptop, with its 16 shades of orange, these colours are indistinguishable. The problem occurred in many places throughout the installation, and there does not seem to be any way of tailoring the onscreen colours used by the installation program.

Continuing with the installation, I was asked to choose between TURBO 2000 (described as a 'no fault, no brainer, type of installation'!), or SAGA 3000 installation: the difference between the two is defined only later in the process. I selected TURBO 2000, which told me (for the first time - note that now I'm in the thick of things) to scan my hard disk. Ninety files were then copied to drive C, into a subdirectory chosen by *Virus ALERT* itself. Curiously, it copied one less than the claimed total number of files. This number included 40 (yes, forty) documentation files.

The parts of the installation which introduce both a boot-time scan and a memory-resident program operated on the whole without problems. The file AUTOEXEC.BAT was amended correctly (if requested). The program did not validate data entries, but simply used them blindly. I accidentally specified a drive name as 'C.' (rather than 'C:'), which the install program then inserted into my AUTOEXEC.BAT: only when a reboot occurred was any error message visible.

After de-installing the product, I tried re-installing it using SAGA 3000: as promised, it was more cautious (scanning both source and destination), and did allow more flexibility with regard to subdirectories and drives. The integrity check performed during this installation comprised a count of the number of files, and a visual inspection that the available space on the floppy disk was correct: this is close on useless.

For some reason, the installation program decided that I had files from an old version of *Virus ALERT* installed (this was not the case), offered to delete them, and unsurprisingly, as the assumption was mistaken, failed to find them. This caused umpteen 'File not found' messages to whizz past onscreen. The last straw came when the message 'NEW.EXE is not on the disk' was displayed, at the end of the installation: the computer then locked up. This is poorly tested territory. Stick to the other installation method.

Finally, I cannot conclude this section without making it crystal clear that many of the batch files used by the installation process assume that ANSI.SYS is installed on the computer on which *Virus ALERT* is being installed. This is a very unwise assumption to make nowadays, as few people use ANSI.SYS. Running the install program without this file present leaves strange character sequences visible onscreen: this renders installation impossible, and no help is offered on how to make the install program work properly.

Documentation

The main part of the printed documentation provided with *Virus ALERT* comprises one sheet of paper, printed in a very small font and folded twice to make a little booklet called a 'Quick Reference Handbook'. To be blunt, there isn't much in the way of documentation.

Accompanying the booklet were two A4 sheets covered in testimonials from 'contented users', a few sheets describing 'Work in Progress', and an A4 sheet headed 'Virus Alert, Product Information'. The latter is full of marketing hype coupled with a table of data purporting to show that *Virus*

ALERT is incredibly fast compared to other scanners (which it is - see below). Also included was a Product Information sheet for a golfing program called 'Handicap Manager', but I doubt I was meant to receive that. Golf never was my forte.

An on-line index, containing a Glossary, and Index, and several 'Quick-read' sections, replaces the manual. The colour scheme used meant that, similar to the problems experienced during the installation, the on-line index was invisible on my laptop. However, scanning through the various documentation files manually, they seemed to provide sensible advice.

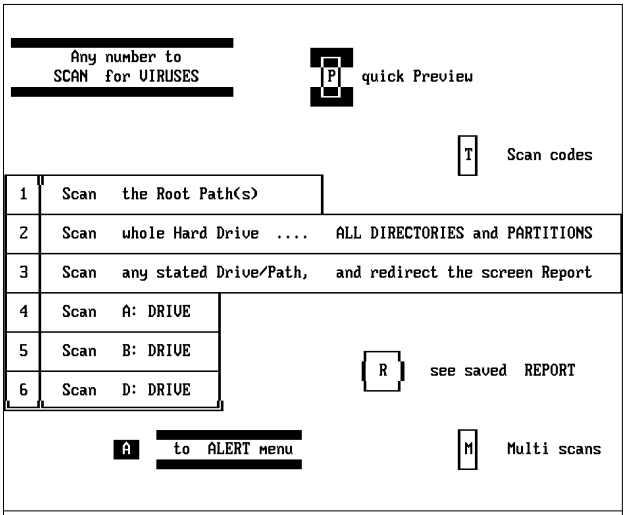
The Product

If you dig around inside the *Virus ALERT* files, there are several references to *ThunderBYTE* (a well known, fast, accurate, anti-virus program manufactured by the Dutch company *ESaSS*). One small file containing such a reference, called VALERT.KEY, is dated 30 July 1995 (?!). On examination, the interior of VASCAN.EXE (the main program) is seen to contain the message: 'This program is the property of *ThunderBYTE BV'*.

This situation raises an intriguing question; one which crops up ever more frequently as the amount of effort involved in maintaining an anti-virus scanner becomes steadily more onerous. However, given the amount of work needed to keep a scanner up to date, I was neither surprised nor worried to discover that *Look! Software* has badged a scanner - many companies do.

Recovery Disk

The installation option to make a Recovery Disk copies six files across to the floppy disk. This can be used as 'a quick and easy method of removing Boot and Partition table virus infections, or for recovering your computer's Boot Record and CMOS values after a hard disk crash…'.



Virus Alert allows several choices as to which area to scan. The scanner's options have otherwise been preset.

The diskette used for the Recovery Disk is not formatted before the files are copied, and the copying process fails (albeit gracefully) if not enough space is available. If the disk is not inserted or is write-protected, the time-honoured DOS error message 'Retry, Ignore, Fail, Abort' appears: this should be trapped and a more user-friendly message displayed. On leaving this section of the installation, the error message 'Missing ENDTEXT' always appeared. I know not why.

Scanning

All scanning options are fixed: the company espouses a philosophy that their product has been preset with the 'best' options, and that the user should not have to make any 'choices'. Exactly how the developers of *Virus ALERT* can know which are the best options is beyond me. I certainly would not know how to set up a scanner so that its modes of operation suited everybody.

The opening screen of the scanner offers a numbered menu, the options being to scan the root path, the whole hard drive, any stated path, or a floppy disk drive.

The colour scheme used by *Look! Software* also merits a mention: I have never commented on the colours used by an anti-virus scanner before; however, those used by this product certainly are unusual. *Virus ALERT* uses combinations such as pink, lime green and purple. The vendors defend this 'striking' colour scheme as one of the product's main features - *à chacun son goût*, I suppose.

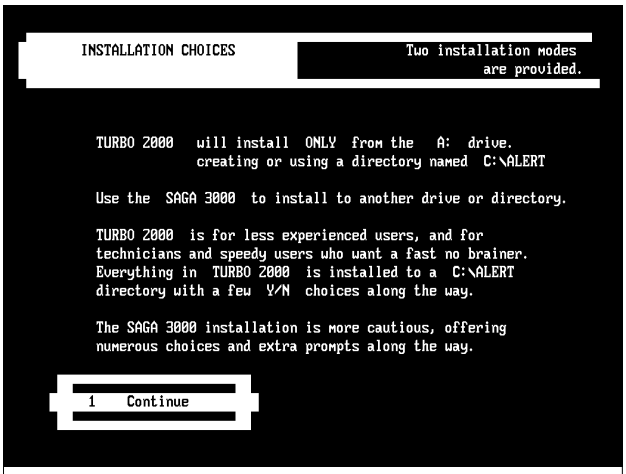
The product scanned the hard disk of my test computer in 40 seconds (708 files occupying 11.1 Mbytes). When executed under *Windows*, the scan time only increased to 42 seconds, an impressively small increase. In comparison, *Dr. Solomon's AVTK* took 1 minute 36 seconds, and *Sophos' SWEEP* 2 minutes 54 seconds, to perform the same scan.

There is no doubt that *Virus ALERT* is seriously fast at scanning: as the scanning engine is in reality *ThunderBYTE*, I would expect nothing less. The scan times quoted above were unaffected by the presence of *Virus ALERT's* memory-resident monitor program, which is to be expected.

Accuracy

When run against the *VB* test-set, *Virus ALERT* detected 247 of the 248 virus infected test samples, a detection rate of 99.6%. It missed the EXE version of Invisible_Man, even though it correctly detected the COM version of the same virus. When tested against the 500 Mutation Engine (MtE) samples, all 100% were detected correctly.

By any standards, these are excellent results. There are a few minor nomenclature differences between what *Virus ALERT* calls a particular detected virus, and what I would expect it to be called, but overall not only does *Virus ALERT* do a good job of detecting virus infected files, it also identifies nearly all the viruses correctly. Impressive.



The product offers two types of installation; TURBO 2000, and SAGA 3000: all scanning options, however, are fixed.

Memory-resident Program

The memory-resident component of *Virus ALERT* is called ONGUARD. The documentation claims that it occupies 31 Kilobytes of memory when installed: I measured memory usage at 43 Kilobytes (40 Kilobytes for the main ONGUARD file, and 2.9 Kilobytes for an associated driver). The documentation states that ONGUARD is a TSR which 'monitors all standard DOS activities including floppy disks for viruses'. Executable programs are checked before loading and during copying.

The overhead introduced by ONGUARD was measured by timing how long it took to copy 40 files (1.25 MB) from one subdirectory to another. Without ONGUARD, they could be copied in 23.4 seconds, rising to 52.4 seconds under ONGUARD: such an increase (over 120%) is significant.

ONGUARD provides an onscreen message stating that it contains 2380 signatures. I tested detection by copying a test-set containing one sample of each of the viruses listed in the Technical Details. I was fooled by a zero detection rate until I realised that ONGUARD only tests a file being copied if its file name carries an executable extension. Once the infected files were renamed, *Virus ALERT* identified 130 out of the 149 test viruses - a detection rate of 87%.

The viruses missed (WinVir14, Todor, Coffeeshop, Sibel_Sheep, Invisible_Man, Nuke_Hard, Starship, Maltese_Amoeba, VFSI, Tequila, Kennedy, V2P6, 1260, Casper, Virus_101, Slow, MtE, Tremor, Satanbug) are mainly polymorphic and/or encrypting viruses; 'difficult-to-detect' viruses which would further increase the overhead imposed by ONGUARD if a check were made for them.

Other Comments

Using *Virus ALERT*, a virus can also be 'cleaned' from an infected file. I did not review this, as I am convinced that the safest way to remove a virus from an infected executable is to replace the file with a known clean copy.

The boot-time scan added by the installation program checks memory, boot sector, partition table and root path after a soft or hard reboot. The documentation states that this takes '3 or 4 seconds'. On my test computer it took 6.1 seconds.

A program called EXTRACTOR is provided which can make a *Virus ALERT* system disk on smaller diskettes than the 1.44 Mbyte disk provided. As this is not an anti-virus feature, I will not comment on it. Likewise, a program called TESTER is mentioned in the *Virus ALERT* documentation, but was not present on the floppy disk provided.

Conclusions

Whilst *Virus ALERT* does use the scanning technology of a much better-known product, *ThunderBYTE* - it would be wrong to write it off just because of that. *Look! Software* claims that the cheerful interface they have added to the underlying technology is exactly what novice users want.

Indeed, it is easy to see why this would be so. The configuration options on many modern scanners are bewildering in their complexity, and unless the system is configured carefully, it is very easy for the results to be unexpected. *Virus ALERT's* main function is to provide a simple entry level anti-virus system.

By their own admission, specifically targeting this end of the market will alienate the more 'expert' user. Users who are reasonably well-versed in DOS or in the mechanics of anti-virus products will probably want to look elsewhere.

Look! Software needs to debug system installation, which seems to have several problems (especially the SAGA 3000 option). It is likely, however, that their target market will opt for the simpler TURBO 2000 installation.

On the other hand, if you want a very fast scanner with an excellent detection rate, and the interface appeals to you, then *Virus ALERT* may well be the right choice.

Technical Details

Product: *Virus ALERT v 3.34.*

Vendor: *Look! Software*, PO Box 78072, Cityview, Nepean, Ontario, Canada K2G-5W2, Tel +1 613 822 2250, fax +1 613 822 2160, BBS: +1 613 837 2159, Email: info@look.com

Availability: Nothing is mentioned about specific PC requirements. The *Virus ALERT* DOS programs require 300 Kbytes of RAM; the ONGUARD TSR requires 44 Kbytes of RAM. The install program requires 425 Kbytes of RAM.

Serial number: None visible.

Price: US$69.95 (single user licence), US$199.95 (5 users), US$299.95 (10+ users). Includes updates every two months.

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

Viruses used for testing purposes: For complete details of the test-sets used, see *Virus Bulletin*, May 1995, p.23. For a complete explanation of each virus, please refer to the list of PC viruses published regularly in *VB*.

SUBSCRIPTION RATES

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

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

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, 21 The Quadrant, Abingdon, Oxfordshire, OX14 3YS, England

Tel 01235 555139; International Tel +44 1235 555139
Fax 01235 531889; International Fax +44 1235 531889
Email virusbtn@vax.ox.ac.uk
CompuServe 100070,1340@compuserve.com

US subscriptions only:

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

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

END NOTES AND NEWS

The 5th Annual Network Security in the Open Environment conference (*NetSec 95*) will be held from 12-14 June 1995 in New Orleans, Louisiana, USA. Tutorials, vendor exhibits, and seminars will be included. Further information is available from the Computer Security Institute on Tel +1 415 905 2626; fax +1 415 905 2218.

The fifth annual *Virus Bulletin Conference*, ***VB 95*, will be held at the *Park Plaza Hotel* in Boston, Massachusetts, USA**, from 20-22 September 1995. Internationally-renowned virus and security experts will address the problems of virus protection in the 1990s. For more information, contact Petra Duffield, Conference Manager, on Tel +44 1235 555139; fax +44 1235 531889.

S&S International* will be holding a *Live Virus Workshop on 12/13 June (venue to be confirmed). Cost for the two-day course is £680 + VAT. Contact *S&S International* on Tel +44 1296 318700; fax +44 1296 318777 for further information.

Precise Publishing Ltd has announced three further dates for their ***Live Computer Virus 'Hands On' Workshops***. They will take place on site at the company's training centre, on 14, 22, and 28 June. For more information, contact Kevin Powys on Tel +44 1384 560527; fax +44 1384 413689.

***Compsec 95* will take place in London**, UK, from 25-27 October 1995. For details on the conference, contact Jill Spear at *Elsevier Science Ltd* on Tel +44 1865 843643; fax +44 1865 843971.

RG Software has announced the launch of a new generation of software aimed at **eradicating the boot sector virus**. The product, *'No More #*!$ Viruses'*, uses the company's own *'PC Thermometer ™'* technology to ascertain whether or not the PC is already infected, and refuses to install on an already-infected PC. The heuristically-based software, said to have a one-time installation taking only minutes per PC, operates transparently. It will be reviewed in a forthcoming issue of *VB*. For further details, contact Phoenix, Arizona-based *RG Software* on Tel +1 602 423 8000; fax +1 602 423 8383.

The ***First Cologne IT Security Forum*** (*1. Kölner IT-Sicherheitsforum*) will be sponsored by *datakontext tagungen GmbH* in Cologne, Germany on 12/13 July 1995. The main aim of the conference is to pinpoint areas of weakness within companies' structures, and the risks posed by these weak areas. Price for both days is DM 1850, inclusive of VAT. Further details are available by contacting either the company on Tel +49 2234 65633; fax +49 2234 65635, or Ralf Herweg or Thomas Müthlein on Tel +49 2234 691961.

The next round of **anti-virus workshops from *Sophos Plc*** will be held on 26/27 July, at the training suite in Abingdon. Day one is an introduction to computer viruses; day two, an advanced virus workshop. One day costs £325.00; both, £595.00 + VAT. Contact Julia Line on Tel +44 1235 544028; fax +44 1235 559935 for details.

A two-day workshop has been scheduled by *Reflex Magnetics*, at their premises in London on 21/22 June 1995. Titled ***'Live Virus Experience - Introduction and Advanced'***, the two-day course costs £345 + VAT. Details are available from Rae Sutton on Tel 44 171 372 6666; fax +44 171 372 2507.

Three computer security-related conferences have been scheduled in London by *IBC Technical Services Ltd*: *Computer Investigations* - a one-day seminar to be held at the *Britannia Intercontinental Hotel* on 6 July 1995; *Theft from Electronic Systems* - Friday 7 July 1995 (also at the *Britannia*) and *New Security Issues 1995*, 12/13 September 1995, to take place at the *London Hilton Hotel*. Information on the seminars is available from Lisa Minoprio on Tel +44 171 637 4383; fax +44 171 631 3214.