February 1990 ISSN 0956-9979

VIRUS BULLETIN

THE AUTHORITATIVE INTERNATIONAL PUBLICATION
ON COMPUTER VIRUS PREVENTION,
RECOGNITION AND REMOVAL

CONTENTS

EDITORIAL

US Prosecution While Britain Prepares To Legislate

A federal jury in Syracuse, New York, has convicted Robert Morris of a felony charge for creating and unleashing a disabling computer program, namely the Internet Worm. Morris, 24, was the first person to be charged and convicted under the US *Computer Fraud & Abuse Act 1986*. Morris faces a maximum jail sentence of five years and a maximum fine of $250,000. He may also be required to make restitution for users whose work or operations were adversely affected by the worm program. Charges levelled at Morris were: gaining unauthorised access; preventing authorised access to federal interest computers; causing losses exceeding $1,000 in violation of the *Computer Fraud & Abuse Act.*

The Internet worm was released on the 2nd November 1988 and caused crashes in some 6,200 machines on the network. The worm was a set of programs written to take advantage of several holes in the UNIX operating systems and common utilities such as the *Sendmail* program. It was targeted at workstations from Sun Microsystems and DEC VAX minicomputers (see the *Internet Worm One Year On, Virus Bulletin, November 1989*).

Morris claimed in court that he had devised a program "that would spread as widely as possible on the Internet". He said that he had miscalculated the speed of both the program's propagation and movement through the network. However, Justice Department prosecution lawyer Ellen Meltzer concluded that Morris had intentionally incorporated features in the program to evade detection.

In the wake of the Internet worm, the *Computer Emergency Response Team* (CERT) was founded, its headquarters based at Carnegie-Mellon Universty, Pittsburgh, USA. A *Computer Incident Advisory Team* (CIAT) has also been created at Lawrence Livermore Laboratories in California. Both teams are engaged in monitoring and advising on network security and will react to future program attacks. However, there is still widespread concern that known bugs and trapdoors in wide area networks remained unfixed, that systems administrators are not applying 'patches' released by suppliers, and that manufacturers are shirking their security responsibilities.

The successful prosecution of Morris in the United States coincides with the introduction of the *Computer Misuse Bill* by Michael Colvin MP, here in the United Kingdom. Three new categories of offence are envisaged: **unauthorised access** (penalties of a six month imprisonment or maximum £2000 fine); **unauthorised access with intent to commit a serious crime** (maximum five year prison sentence and unlimited fine); **unauthorised modification of computer data** (maximum five year prison sentence and fine). The Bill has all-party support and will receive its second reading in the House of Commons on the 9th February.

The third offence, that of modification to data, is specifically aimed at the writers of viruses, worms or malicious programs. There are many arguments both for and against legislating against hacking. The unauthorised destruction, deletion or modification of data, programs or operating systems on the other hand, is indefensible and criminalising this activity is long overdue. The fact that a successful prosecution for exactly this offence has been secured in the United States is most encouraging.

Virus Reports

These reports are **unconfirmed**. Investigation of these reports is continuing.

Novell Virus (Reported only, unconfirmed)

This virus appears to be aimed specifically at Novell networks. It is reported as destroying the Novell-specific file allocation table rendering all files on the server disk useless. The only known remedy to such an attack is to base-level format the hard disk using COMPSURF and restore from backups.

The virus is said to be capable of penetrating a file server from a workstation even if it is not logged on to the network. This might be possible by altering the NET$DOS.SYS program, perhaps using the C libraries released last year by Novell.

Novell distributors *Persona* have advised that setting all files accessed by NET$DOS.SYS, or other users/programs in the same group, to Read- Only may provide some protection against such an infection. Comparing NET$DOS.SYS with a known version from backups should indicate whether or not a system has been infected. Novell have not encountered this attack program nor have they received reports of it.

IBM Virus - 'Gates of Hades' (Reported only, unconfirmed)

The 'Gates of Hades' virus probably originates from the United States. It is understood to destroy, or possibly encrypt the hard disk. In at least one case it has damaged a disk by sending the disk head beyond limits. It may be triggered by the date reaching 1st January 1990 and therefore may have remained dormant on systems for some time.

AS/400 attack - (Reported only, unconfirmed)

A virus attack on IBM AS/400 minicomputers running PC support was reported in the January edition of the *Computer Fraud & Security Bulletin*. The Jerusalem virus was reported as having attacked AS/400 DOS files on October 13th 1989.

Systems managers should take note that DOS parasitic viruses have the potential to infect minicomputers offering PC compatibility.

PROCEDURES

Edward Wilding

Guidelines to Assist Virus Prevention and Post-Attack Recovery

Prevention is better than cure. The following techniques and recommendations have been devised to help DP managers minimise exposure to the threat of microcomputer virus infection and other PC attack programs. They will also assist post-attack recovery. The proliferation of IBM PC viruses underlines the need for preventative measures. However, a balance must be struck between reasonable security and business efficiency.

1. Backups. Data files **must** be backed up regularly to minimise the extent of data loss which might occur after infection. Monitoring of, and adherence to backup procedures should be enforced. Critical data, such as mailing lists held on a database, should be backed up twice and stored separately. Without backups there is no means to restore unique data. **Backups are the most effective protection against viruses and other malicious programs.** They also provide protection against a range of other threats such as power failure, crashes and human error.

2. Master disks of all software (including the operating system) should be write-protected and stored in a safe place. No virus, or any other software, can modify a write- protected floppy disk.

3. Data can be destroyed or corrupted by computer viruses but data files do **not** spread viruses. Reduce ignorance and fear by teaching relevant staff that viruses can only spread by infecting program disks or tape and executables within the machine. In simpler terms; viruses have to run and spread to do damage.

4. Designating responsibility for taking backups should be decided at senior management level. It is important that system access and user privileges are tightly controlled and monitored. Otherwise, the system is open to abuse by staff copying sensitive or confidential information to disk - information to which they should not have access.

5. Test backup procedures periodically by performing complete restorations of the system.

6. Write-protected system disks should be stored separately from other backups.

7. PCs with hard disks should not be booted from a floppy disk except when the floppy is known to be virus-free. Boot only from a write-protected disk. PCs should only be booted from write-protected system floppy disks when using virus scanning anti-virus software. *(See box, page 4 and Virus Dissection, pages 6-7).*

8. Establish a software quality assurance unit. Program disks should be inspected and approved by this unit before use. Software evaluation should include scanning for known viruses which may reside on disks. This unit should also be responsible for copying and write-protecting master disks. Testing should be undertaken on a standalone PC containing no critical files. Incoming software can be swept with a virus-specific anti-virus software package while a checksumming program is used to report file or system discrepancies. The entire hard disk of the machine can be reformatted at the end of each evaluation. Maintain a log detailing the introduction of software for testing, use of anti-viral software, reformats, and unusual activity.

9. Bulletin Boards. Many BBSs offer programs including anti-virus software. There are conscientious BBS System Operators who regularly screen executable files offered, equally there are many BBSs which are badly maintained and controlled. **All bulletin boards should be treated with appropriate caution**. A number of Trojanised programs have been uploaded to bulletin boards and there have been numerous reports from the United States and elsewhere of viruses residing on BBSs. Some companies have forbidden the use of programs other than shrink-wrapped software from reputable manufacturers. A more flexible approach is to use programs which come from trusted sources and which have undergone 'in-house' testing and validation. Reputable manufacturers, developers and bulletin board System Operators are at pains to avoid distributing viruses, Trojan horses or other malicious code.

10. A shrink-wrapped program is not 'virus-free' by definition. Shrink-wraps can be broken and resealed. Original software can also be virus infected. However, due to manufacturers' quality control, the likelihood of encountering a 'shrink-wrapped virus' is slight.

11. Shareware presents certain added risks over other forms of software provided on disk. A shareware program may have been used on ten, twenty or even a hundred different PCs before you receive it - increasing the likelihood of viral infection. Shareware can be obtained directly by writing to its original author thus reducing associated risks. (*See Public Domain Software pp 18-19, Virus Bulletin, January 1990*).

12. Restrict software to what is necessary for the current project and make sure that its source is definitely known.

13. On receipt of a new program write-protect the master (original) disk before inserting it into a drive.

14. Make a backup copy of each write-protected program disk. Write-protect the backup copy and store in a secure and separate location. [Remember that a virus-infected master disk will result in a virus infected backup]. The original program disk is least likely to be infected and cannot be further corrupted because it is write-protected.

15. Use a compare utility to examine files on the master disk and its backup. Any differences, particularly in the size of files, should be treated as suspicious. Re-examine the PC, master and backup disk for viral code.

16. Ensure that **all** disks and back-ups are accurately labelled (date and time created) and recorded in an inventory. Back-ups must be stored safely. Fire-resistant safes are often used for critical backup program and data disk storage.

17. The movement of software between departments 'in house' and intra-company should be controlled. This system is analogous to the use of bulkheads in a ship and will help to control the spread of a virus attack and makes its path more predictable.

18. Warn staff of the potential dangers in using bootleg or pirated software and programs of unknown origin, including unsolicited software and free samples provided with magazines. Some companies have introduced disciplinary measures for staff using unauthorised software. Decisions of this kind should be taken at a senior level. A consideration is the fact that copying PC programs, for purposes other than backups, is both bad anti-virus practice and usually illegal. Ideally, forbid the use of personally acquired software on company machines. If software games and other unauthorised programs cannot be forbidden consider the use of a 'dirty PC' - an isolated machine specifically to be used for unvalidated programs. All programs run on this machine should be categorised as suspect and absolutely no company applications or data stored on it. Strict controls over the storage and movement of software on 'dirty PCs' need to be enforced.

19. Do **not** rely on clever fixes or 'patches'. Adjusting the system clock, date-setting or tampering with files does not provide protection. Viruses use different methods of infection and have varying trigger catalysts and mechanisms. 'Patches' promote confusion, intefere with legitimate applications and engender a false sense of security.

20. Educate everyone in your department about computer viruses, Trojan horses and logic bombs. Explain the level of damage which these programs can cause and tell people, in simple language, how they infect systems and how they can be prevented. Try not to engender 'virus paranoia' which will prove counter-productive. Describe company policies about the exchange of software, reporting of incidents and responsibility for investigation and removal of computer viruses. Most people welcome rules and feel more comfortable if instructions are clearly spelled out in black and white.

21. Prepare a contingency plan. Designate an individual to be responsible for dealing with a virus attack, and define procedures for isolating infected disks and PCs. Collate information about external sources of help (consultancy and software) in the event of an attack. Consider the implications to the organisation of bad publicity and establish public relations procedures to prevent information about an attack leaking to journalists.

WRITE-PROTECTED SYSTEM FLOPPY DISK

A write-protected system floppy disk should be prepared in advance and contain all system files plus AUTOEXEC.BAT, CONFIG.SYS and any other system files or device drivers such as ANSI.SYS. CONFIG.SYS normally refers to other files which are loaded into memory before the system is started, using statement such as 'DEVICE=filename'. All these files should be copied onto the floppy disk, and CONFIG.SYS on the floppy disk should be modified, if necessary, to ensure that it refers to files on the floppy disk, rather than the original copies on the hard disk

If a system becomes infected, this disk will be used to bootstrap a computer. This will ensure that various items on the computer can be examined through a 'clean' operating system, not giving a virus the chance to employ hiding techniques such as interrupt interception.

IBM PC VIRUS PATTERNS

The following are hexadecimal patterns of known viruses affecting IBM PCs and compatibles. This can be used to detect the presence of the virus by the ''search'' routine of disk utility programs such as *The Norton Utilities* or your favourite disk scanning program.

```
405               26A2 4902 26A2 4B02 26A2 8B02 50B4 19CD ; Offset 00A
4K                E808 0BE8 D00A E89A 0AE8 F60A E8B4 0A53 ; Offset 239
Agiplan           E9CC 0390 9090 9090 9C50 31C0 2E38 26DA ; Offset 0 (?)
Alabama           8CDD 33DB 8EDB 8B07 0B47 0274 7489 1F89 ; Offset 109
Amstrad           C706 0E01 0000 2E8C 0610 012E FF2E 0E01 ; Offset 114
Brain             A006 7CA2 097C 8B0E 077C 890E 0A7C E857 ; Offset 158
Cascade (1) 01    0F8D B74D 01BC 8206 3134 3124 464C 75F8 ; Offset 012, 1701 bytes, Falling characters
Cascade (1) 04    0F8D B74D 01BC 8506 3134 3124 464C 75F8 ; Offset 012, 1704 bytes, Falling characters
Cascade (1) Y4    FA8B CDE8 0000 5B81 EB31 012E F687 2A01 ; Offset 000, 1704 bytes, Falling characters
Cascade format    0F8D B74D 01BC 8506 3134 3124 464C 77F8 ; Offset 012, 1704 bytes, Formats hard disk
Dark Avenger      740E FA8B E681 C408 08FB 3B26 0600 73CD ; Offset 068, 1800 bytes
Datacrime (1)     3601 0183 EE03 8BC6 3D00 0075 03E9 0201 ; Offset 002, 1168 bytes
Datacrime (2)     3601 0183 EE03 8BC6 3D00 0075 03E9 FE00 ; Offset 002, 1280 bytes
Datacrime II      2E8A 072E C605 2232 C2D0 CA2E 8807 432E ; Offset 022, 1514 bytes
dBASE             50B8 0AFB CD21 3DFB 0A74 02EB 8A56 E800 ; Offset 636, 1864 bytes
dBASE destroy     B900 01BA 0000 8EDA 33DB 50CD 2658 403C ; Offset 735, 1864 bytes
December 24th     C606 7E03 FEB4 5290 CD21 2E8C 0645 0326 ; Offset 044
Den Zuk           FA8C C88E D88E D0BC 00F0 FBB8 787C 50C3 ; Offset 0
Disk Killer       2EA1 1304 2D08 002E A313 04B1 06D3 E08E ; Offset 0C3
Do nothing        8CCA 8EDA BA00 988E C2F3 A41E B800 008E ; Offset 020
Fu Manchu         FCB4 E1CD 2180 FCE1 7316 80FC 0472 11B4 ; Offset 1EE, 2086 bytes COM files, 2080 bytes EXE files
GhostBalls        AE75 EDE2 FA5E 0789 BC16 008B FE81 C71F ; Offset 051
Icelandic (1)     2EC6 0687 020A 9050 5351 5256 1E8B DA43 ; Offset 0C6, 656 bytes
Icelandic (2)     2EC6 0679 0202 9050 5351 5256 1E8B DA43 ; Offset 0B8, 642 bytes
Icelandic (3)     2EC6 066F 020A 9050 5351 5256 1E8B DA43 ; Offset 106, 632 bytes
Italian-Gen       B106 D3E0 2DC0 078E C0BE 007C 8BFE B900 ; Offset 030
Italian           32E4 CD1A F6C6 7F75 0AF6 C2F0 7505 52E8 ; Offset 0F0
Jerusalem         03F7 2E8B 8D11 00CD 218C C805 1000 8ED0 ; Offset 0AC, 1813 bytes COM files, 1808 bytes EXE files
Lehigh            8B54 FC8B 44FE 8ED8 B844 25CD 2106 1F33 ; Offset 1EF
MachoSoft         5051 56BE 5900 B926 0890 D1E9 8AE1 8AC1 ; Offset ?
Mistake           32E4 CD1A 80FE 0376 0A90 9090 9090 52E8 ; Offset 0F0
MIX1              B800 008E C026 803E 3C03 7775 095F 5E59 ; Offset 02E
MIX1-2            B800 008E C0BE 7103 268B 3E84 0083 C70A ; Offset 02A
New Zealand (1)   0400 B801 020E 07BB 0002 B901 0033 D29C ; Offset 043
New Zealand (2)   0400 B801 020E 07BB 0002 33C9 8BD1 419C ; Offset 041
Oropax            06B8 E033 CD21 3CFF 7423 8CCE 8EC6 8B36
Palette           EB2B 905A 45CD 602E C606 2506 0190 2E80 ; Offset ?, 1538 bytes
Pentagon          8CC8 8ED0 BC00 F08E D8FB BD44 7C81 7606 ; Offset 037
Perfume           FCBF 0000 F3A4 81EC 0004 06BF BA00 57CB ; Offset 0AA
South African 1   1E8B ECC7 4610 0001 E800 0058 2DD7 00B1 ; Offset 158
South African 2   1E8B ECC7 4610 0001 E800 0058 2D63 00B1 ; Offset 158
Suriv 1.01        0E1F B42A CD21 81F9 C407 721B 81FA 0104 ; Offset 304, 897 bytes
Suriv 2.01        81F9 C407 7228 81FA 0104 7222 3C03 751E ; Offset 05E, 1488 bytes
Suriv 3.00        03F7 2E8B 8D15 00CD 218C C805 1000 8ED0 ; Offset 0B0, 1813 COM files, 1808 EXE files
Swap              31C0 CD13 B802 02B9 0627 BA00 01BB 0020 ; Offset ?
Sylvia            CD21 EBFE C3A1 7002 A378 0233 C0A3 9E02 ; Offset 229
Syslock           8AE1 8AC1 3306 1400 3104 4646 E2F2 5E59 ; Offset 0, 3551 bytes
Traceback         B419 CD21 89B4 5101 8184 5101 8408 8C8C ; Offset 104, 3066 bytes
Spanish           E829 06E8 E005 B419 CD21 8884 E300 E8CE ; Offset ?
Typo              5351 521E 0656 0E1F E800 005E 83EE 24FF ; Offset 01D, 867 bytes
Vacsina           E800 005B 2E89 47FB B800 008E C026 A1C5 ; Offset 3AC
Vienna (1)        8BF2 83C6 0A90 BF00 01B9                  ; Offset 005, 648 bytes
Vienna (2)        FC8B F281 C60A 00BF 0001 B903 00F3 A48B ; Offset 004, 648 bytes
Vienna (3)        FC89 D683 C60A 90BF 0001 B903 00F3 A489 ; Offset 004, Virus awaiting disassembly
Virus-90          558B 2E01 0181 C503 0133 C033 BBB9 0900 ; Offset 01E
Yale              BB40 008E DBA1 1300 F7E3 2DE0 078E C00E ; Offset 009
Yankee            E800 005B 81EB D407 2EC6 875C 00FF FC2E ; Offset 0
Zero Bug          81C9 1F00 CD21 B43E CD21 5A1F 59B4 43B0 ; Offset 100
```

AIDS disk patterns - **not** a virus

```
REM$.EXE          4D5A 0C01 1E01 0515 6005 0D03 FFFF 3D21 ; Offset 0
AIDS.EXE          4D5A 1200 5201 411B E006 780C FFFF 992F ; Offset 0
```

VIRUS DISSECTION

Fridrik Skulason

Dark Avenger

This is a parasitic resident virus. It will infect both .EXE and .COM files containing 1775 bytes or greater. The virus' infective length is 1800 bytes. As with most other viruses, the author's identity is not known but there are some indications that Dark Avenger originated in Bulgaria. One of the first reports of it came from there and inside the virus is a text string which reads:

```
This program was written in the city of Sofia (C)
1988-89 Dark Avenger
```

According to recent reports the virus is fairly common in Bulgaria and several other countries in eastern Europe. It has also appeared in West Germany and Moscow. No cases have been confirmed in the UK.

The virus produces no visible symptoms, but there is a 1-in-16 chance that it will overwrite a data sector when an infected program is run.

Operation

Like other parasitic viruses, Dark Avenger will not be activated unless an infected program is run on the machine.

To determine whether it is already installed in memory, the virus checks the offset part of the INT 21H vector. If it contains some value other than 02EE, the virus will infect memory. If 02EE is found, the virus will verify that it is already present by scanning for itself in memory. One might believe that this method would not work, since another memory resident program run later might hook into INT 21H and make it point to itself. The Dark Avenger attempts to avoid this.

If Dark Avenger determines that it is not already present in memory, it will release the current memory block. It will then request two blocks instead, one for the current program, and another 3680 bytes in length located at the top of available memory for holding the virus code. This block is hidden by marking it as a block belonging to the operating system.

The virus then copies itself into the block it previously created and hooks into INT 21H and INT 27H. Before the virus transfers control to the original program it may cause damage.

Every time an infected program is run, a counter is incremented by one. This counter is stored at offset 0A in logical sector 1 (the boot sector) of the drive from which the program was loaded. In order to determine which drive to use, the environment area is searched for the full path name of the current program. Under DOS 3.x and 4.x the first character of the name contains the drive identifier. On a computer using an older version of DOS, the path name is not stored making the behaviour of the virus difficult to predict.

In 15 cases out of every 16, the virus will only increment the counter, but there is a 1-in-16 chance that the information stored in the boot sector will be processed further. The boot sector contains information about the disk including the number and size of FATs, size of the root directory and the number of hidden sectors. Dark Avenger uses this information to locate a data sector and stores its location in the boot sector in positions 08-09. Bytes 08-0A normally contain an OEM (Original Equipment Manufacturer) label or a string such as 'MSDOS3.3', so it is generally safe to use them for storing variables.

In addition, if the counter contains 0F, it will be reset to 0 and the destructive part of the virus will be activated. The virus will proceed to overwrite the sector it just selected. This process will eventually cause all data sectors to be selected for overwriting except those sectors containing the boot record, FATs and the root directory which remain unchanged.

INT 27H Routine

The function of INT 27H is to keep the calling program resident in memory when it terminates. The Dark Avenger hooks into this interrupt in order to make sure it is in control at all times. Whenever a program calls INT 27H, the virus checks the current value of the INT 21H vector to see if it still points to the virus code. If not, Dark Avenger redirects it to the virus code. This redirection method is similar in some respects to the operation of Borland's *Sidekick* utility.

INT 21H Routine

The virus' INT 21H routine is complex because, unlike most viruses, it interecpts numerous functions. They are:

```
AH=25
(Set interrupt vector)
```

If a program attempts to change INT 21H or INT 27H, the virus will simply store the new address in a variable, but will

not modify the vector. As described before Dark Avenger attempts to control INT 21H at all times.

```
AH=35
(Get interrupt vector)
```

If this function is used to obtain information about INT 21H or INT 27H, Dark Avenger will return the stored address, which may have been set by function AH=25.

```
AH=4B00
(Load/execute program)
```

Dark Avenger attempts to infect any program being executed.

```
AH=3C or AH=5B (Create file)
```

If the file being created is a .COM or .EXE file, the virus will set a flag and store the file handle.

```
AH=3E (Close file)
```

If the handle number matches the number stored in the create operation, the virus will attempt to infect the file.

```
AH=3D (Open file), AH=43 (Chmod) and AH=56 (Rename)
```

These three functions will cause an infection if the file is either a .COM or .EXE file. Paradoxically, the fact that an 'open' function may cause an infection means that running a virus scanning program on an infected machine may cause each .EXE and .COM file checked by the scanning program to become infected. This can be prevented by ensuring that the virus is not present in memory when the program is run.

This reinforces the lessons that (a)*no software, anti-virus or otherwise, should be run if you know a computer virus is active in memory* and (b) *any anti-viral scanning or checksumming program should be run from a write-protected clean system boot disk to ensure that the executable path is uncontaminated by a virus.* (See page 4).

If you are uncertain of the integrity of your bootstrap process (eg if you do not have a write-protected system boot disk) a prudent approach would be to use a scanning program which initially searches only in memory (not on disk) and which can recognise the Dark Avenger TSR pattern. The hexadecimal pattern produced on page 3 of this edition of*Virus Bulletin* can also be used for searching memory for this virus.

Any other INT 21H function will simply result in a jump to the stored address mentioned above.

Infection

Before the virus attempts to infect a file, it will redirect the INT 24H (Critical error) vector to a routine which always returns 3. This is done in order to prevent suspicious 'Abort, Retry, Fail?' messages if the virus attempt to infect a write-protected disk.

A return code of 3 is not valid under DOS 2.X, but for later versions of DOS it is equivalent to 'Fail'.

Like most parasitic viruses, Dark Avenger will turn off the Read-Only attribute and open the file in a Read/Write mode. The original attribute settings will be restored when this file has been infected, as will the creation date.

If the length of the file is less than 1775 bytes, it will not be infected. This is probably done to reduce the likelihood of detection.

The virus then determines if it has encountered a .EXE or .COM file by checking the first two bytes of the file. If they contain 4D 5A the virus will locate the start of the executable code within the program. Dark Avenger then checks for an existing infection by scanning for a part of itself within the file. If the file is not infected the virus appends itself to the program and modifies the header as necessary. The original information in the header is stored near the end of the file.

If the file being infected does not commence with the 'magic number' 4D 5A, Dark Avenger assumes it to be a .COM file. It will then check whether the program starts with JMP instruction (E9). If so, the virus will search for its own presence by the process of self-recognition described above. If no infection is found, and the .COM file is smaller than 63149 bytes, the virus will append itself to the file and overwrite the first three bytes with a JMP instruction. The original three bytes are stored at the end of the virus code.

Final Notes

In addition to the text string mentioned at the beginning of this article, two other strings are contained within the virus.

```
Eddie lives...somewhere in time!
```

and

```
Diana P.
```

'Eddie' is understood to refer to the mascot skeleton of the English heavy-metal group*Iron Maiden* who released an album entitled 'Somewhere in Time' in July 1986. It is not known who Diana P. is.

PROGRAM ANALYSIS

Jim Bates

Disassembly of High-Level Programs and the AIDS Trojan

The recent AIDS Trojan revealed an alarming lack of understanding among some specialists, concerning the power available in modern high-level languages - in this case, *QuickBASIC 3.0.* Even more worrying were the rumours circulating which suggested that the program code (INSTALL.EXE) contained a virus, would take months or even years to disassemble, and contained complex routines that even professional BASIC programmers were at a loss to duplicate.

Many computer professionals and users appreciate the significant differences between viruses, Trojans and other malicious programs. *It is therefore incumbent upon anyone offering expert evaluation of malicious code to be as accurate as possible in the information that they supply.* To suggest that a Trojan 'may' contain a virus or that a virus 'may' simply be a Trojan, without being reasonably sure of the facts is highly irresponsible.

In the case of the AIDS incident, there was an urgent need for information in order to limit the damage which non-technical users might sustain. *The balance between the requirements of accuracy and speed presents a tightrope which all researchers into malicious code must walk.* During early investigation of the AIDS Trojan I surmised that it was written in QuickBASIC 3.0. Later research indicates that although the program was certainly compiled using this language, the original source code did not necessarily use this programming environment.

A number of specialists spent many, many hours attempting disassembly of the INSTALL program at machine code level. On a high-level, compiled language program this is an immense and complex task, if only because around 50 percent of the code (and around 95 percent of the processing) is concerned with the compiled library routines. Thus normal disassembly to machine code level involves disassembling the actual library of language routines. For reference purposes I ran the INSTALL.EXE file through a standard disassembler program and this produced around 46,000 lines of assembler code. By contrast the primary code segment is equivalent to approximately 4,610 BASIC statements.

What follows is intended to provide a small insight into how a suspected malicious program, written in a high-level language, can be dissected to produce accurate information concerning its functions relatively quickly. Remember that tests had revealed what the program did during both the installation and triggering phases on a number of different machines under differing conditions. It should also be noted that "examination" of code (as referred to here) also implies actual execution on a step by step basis where required.

The priority was to establish whether the Trojan permanently modified the machine operating system, either directly or by modification of the disk or existing files. Eliminating this possibility would enable me to state with a fair degree of certainty that no virus would be introduced during the installation phase.

Disassembling virus or Trojan code at assembler level is a relatively simple (if somewhat lengthy) process. It usually involves following the chain of instructions from a known starting point, noting the inclusion of specific routines which may be referenced by a variety of methods during processing. These will include the insertion of Interrupt handling and interception routines into the machine operating system. Such specific system "enhancement" is virtually impossible when dealing with high-level languages although the language environment itself may well make such changes on a temporary basis until the program completes and exits (QB3 certainly makes substantial changes of this type). Note that I say "almost" impossible, because code to introduce permanent changes can be introduced in a number of different ways. *However, an important factor when examining high-level programs is the observed skill of the programmer as this may indicate at what level malicious code could be introduced.*

Identifying the Program Elements

Turning specifically to QuickBASIC 3.0 and the AIDS 'INSTALL.EXE' program - it must first be understood that the QB3 compiler generates a processing stream which corresponds closely to the original source code but only at a subroutine level. A simple statement such as "PRINT X" will cause the compiler to first generate an instruction to load the address of the variable X into a particular register, a further instruction is then generated to CALL the routine corresponding to the PRINT statement. The important point here is that the PRINT routine is not generated by the compiler **or the source code**, it is collected from the language's standard library of routines. Thus the compiler output is limited solely to the assembler equivalent of the source code together with a collection of the routines required by the source and a formalised collection of the literal data that the program uses. A complication is introduced where some high-level languages allow the introduction of new (or EXTERNAL) subroutines but this can be

dealt with in a slightly different way. Once the foregoing principal is grasped, the process of dissecting a QB3 program is made much simpler. The INSTALL.EXE program was some 146,188 bytes long and this was distributed into available memory according to the contents of the relocation table at the beginning of the program. In this case, the program elements were distributed as follows:

- The primary segment containing the **compiler output** contained 65,040 bytes.
- The secondary segment contained the **library routines** and totalled 42,448 bytes.
- The data segment contained the program's **literal data** (print strings), **numeric data** etc. and amounted to 11,296 bytes.
- The tertiary segment containing **QB3's initialisation code** which amounted to some 5,376 bytes.
- The **relocation data**, together with various **constant items** and **signatures** accounted for a further 22,028 bytes giving the grand total of 146,188 bytes referred to above.

Identifying segments takes only a few minutes on a program compiled with QB3 and the size of the dissection task is thereby reduced to an inspection of the **literal data area** and a disassembly of the **primary code segment**. A large proportion of the literal data area within INSTALL.EXE had been encrypted to prevent inspection and this increased the work necessary to discover the program's secrets. Decryption was left for later since it was hoped that the assembler listing of the primary segment would provide a clue to its construction.

Referring back to the inclusion of external libraries, since the QB3 compiler collects the required subroutines from specifically named library modules, routines external to the standard library will invariably be located either at the beginning or the end of the secondary code segment. These areas were examined and a single external module was identified as **not** belonging to the standard QB3 collection of library routines. This was disassembled and diagnosed as a slightly modified version of the USERLIB library provided with the QB3 package. This add-on library contains a group of routines which can be extremely powerful because they allow a programmer to secure direct access to the higher DOS functions through direct Interrupt manipulation. Considering the power of these routines and being reasonably conversant with the capabilities of QB3 at normal code level, I identified and catalogued all calls to any routines in the group from the primary code segment. Identifying the calls was relatively straightforward considering there was only 64K of code to search. Only one routine was accessed, and this was called at three distinct points in the primary code. They were categorised as calls to GET a file attribute, SET a file attribute (using INT 21H function 43H) and a call to INT 17H which returned the PRINTER status. These were entirely consistent with what the program was observed to do during all tests.

Bearing in mind the apparent skill of the programmer, it now seemed extremely unlikely that this program contained effective virus code. The only areas left where virus code might be concealed were in direct function code in the primary code segment, or in the encrypted data. Either possibility would require a far higher degree of programming skill than the program had so far displayed.

The chances of virus code being present were now so small that I felt confident of producing a preliminary program to help users to remove the initial files (thus disabling the trigger). The AIDS program installation phase created certain hidden files and directories on the C: drive of the machine. I decided to write the first of the "cure" programs, *AIDSOUT* to help non technical users clear these hidden installation files from their machines with the minimum of fuss. *AIDSOUT was completed and passed to experienced users for testing within twelves hours of receiving the AIDS disk.*

Examination proceeded on the primary code segment for immediate data manipulation which might be consistent with generating virus code. *Boot sector viruses were ruled out since they require direct access to the disk drive using track and sector addressing, and QB3 does not support this without external library routines.* As a general rule, effective code for a parasitic virus will require several hundred bytes. This might show up as a very long chain of immediate string data instructions or possibly a number of smaller chains if the programmer was trying to conceal the code. Two main chains which matched these criteria were noted but they were identified as the filename and extension encryption routines used during the trigger phase. Dissection of these chains provided complete tables of the encryption details. The final search for potential virus code was the encrypted literal data area. This contained some 341 identifiable strings together with 643 direct DATA statements. The data statements were **not** encrypted but consisted mainly of zeros and were subsequently identified as creating a fallback file called CYBORG.EXE used after the trigger phase. The remaining identifiable strings occupied some 9K of memory space, more than enough for virus code, but around 1364 bytes of this was taken up by pointers and length indicators - leaving around 7700 bytes of actual string data. Program output during tests accounted for over 6000 bytes of this and

a further 360 bytes were unencrypted filenames.*The chance of virus code being concealed within the remaining small data strings was minimal.* I concluded the initial analysis of the program file and produced a short progress report which was distributed to interested parties.

I was aware that many people in the UK and Europe were examining this program and I hoped that someone might break the encryption of the string data and publish the resulting plain text. I therefore opted for disassembly of the main elements of the program and began a systematic classification of the normal library routines used by QB3. These were then matched up with the routines included with the INSTALL.EXE program. This is not a difficult process but it **is** very time consuming. I was able to identify the general program structure including the location of GOTO, GOSUB and RETURN statements, but the nature of such a method of disassembly is that it commences extremely slowly and only speeds up as more routines become identified. I was interested particularly in the long immediate data processing routines connected with the filename and extension encryptions. Close disassembly of these revealed the exact nature of the relative encryption tables and cleared up one or two minor questions concerning them. Such things as screen colour changes can be noted during testing as an aid to identifying the various sections of the program with some accuracy and these in turn will help in locating the functions of relevant sections of code.

I was receiving information from various sources concerning the widespread distribution of the program but I was receiving no details of other researchers' efforts. Unfortunately, although the virus conference on CIX was buzzing with messages about the AIDS disk (as reported in*Virus Bulletin, January 1990*), very few gave hard information and some were actively promoting the 'virus' theory from a supposedly 'expert' viewpoint. Several silly arguments had been advanced in an attempt to support the 'virus' theory displaying an alarming ignorance about BASIC in general and QuickBASIC in particular.

Literal Data Encryption

It appeared increasingly unlikely that someone would break the literal data encryption, so I concentrated in that area in an attempt to short-circuit the overall dissection process. I discovered much later that had I continued with the primary code disassembly I would have located the encryption routine within hours. However, I enlisted the help of my colleague and friend, John Sutcliffe, a QuickBASIC expert.

Working separately but communicating by telephone, we extracted the encrypted strings and were then able to crack the data encryption during a single marathon 36 hour session. Credit for breaking the code is entirely John's.

Work started on an enhanced version of the AIDSOUT program (called CLEARAID), which would recover data from a machine where the program had triggered and locked up the hard disk. *Now that we had the complete list of decrypted strings I could state categorically that there was no virus code in the INSTALL program (there was quite simply no room for any!).* I was thus able to complete CLEARAID and send it out for testing which was successful. I then set about producing a detailed report about the AIDS disk (containing a full list of the decrypted strings) and sent it to various interested parties including the police.

The final stage was to complete the subroutine identification process and to recreate the actual source code on a statement by statement basis. The results of this process have caused me to modify some earlier assumptions about both the program and the programmer, although these changes are minor and do not affect the projected performance of either the INSTALL program or the CLEARAID program. There are also some minor technical inaccuracies in my original report which will be corrected in due corse. The encryption algorithm remains unpublished for legal reasons, as do certain aspects of the program's construction.

Conclusion

Thus the process of dissecting a 146K program is much less daunting than it first appears.*Primary analysis identifies particular areas while experience of the language enables each area to be subjected to appropriate scrutiny.* The suggestions that such disassembly might take months or even years and contained a virus were a measure of the expertise of the people who made such statements. Similarly, the assertion that this program contained "extremely clever" routines depends upon the viewpoint of the observer. Happily, the more responsible sections of the computer press provided a constant stream of accurate, relevant information about the actual effects of the AIDS disk.

PRODUCT EVALUATION

Phil Crewe

Virex Version 2.3

Virex is a well known and respected anti-virus tool kit supplied by HJC Software based in Durham, USA. Until very recently the version number was 2.1 (actually 2.12 as reviewed in *Virus Bulletin, December 1989*). It has recently been updated, however, to cope with two new viruses. The first being J-nVIR (previously called JUDE) and WDEF. The update to version 2.2 happened in the last few days of November 1989 and targeted the J-nVIR virus, and the update 2.3 followed two weeks later after a succession of WDEF virus reports. Beside the additional functionality against the new viruses, the major new advertised benefits of the upgrade to Virex are:

1. The ability to password protect the INIT portion of the Virex package (Virex INIT) so that only authorised users can change it.
2. Compatibility with the A/UX operating system.

It has also been modified to ensure compatibility with other INITs.

The Package

The floppy disk containing Virex 2.3 comes complete with it own system folder enabling immediate boot up from the floppy disk and virus diagnostics. It contains two other files, the Virex 2.3 application and the Virex INIT which is currently version 1.3.

Installation of the package is easy involving the copying of the INIT into the system folder, restarting, and then copying the application to wherever you feel it will be most useful on your hard disk (for example, the applications folder).

Configuration of the INIT is through the Control Panel. I must admit the configuration of the *Symantec* package SAM involves many more options than Virex. Whether this is an advantage or a disadvantage is a mute point. Certainly Virex is not as configurable as SAM in the customisation of levels of protection.

The options available concern diagnosis of floppy disks on insertion, diagnosis of files when open, the ability to lock the INIT in the system folder and the ability to password protect the configuration on the INIT. It is not possible to by-pass the floppy disk check when the floppy disk is inserted, by holding down a particular key combination such as shift, option or command. This means that with Virex the floppy disk is either scanned or not scanned, whereas with SAM you can override the scanning by use of the key combination. Normally this is not necessary (and in some situations having no override is beneficial). I personally like to override the scanning sometimes when I am swapping known good disks several times in the disk drive. This can save some time if implemented. There is the option to set the INIT always to ask before scanning a floppy disk, however this is time consuming. In a corporate environment the System Manager would want the INIT to be set up always to scan a floppy disk without asking. The override selection would naturally only be used by the System Manager but then again it would probably only be known to him anyway.

Regarding configuration of this INIT, it is notable that not all of the options are immediately obvious within the Control Panel. There is a button labelled "**More...**" which hides a multitude of sins. The first screen which makes itself available to you when you click on the "**More...**" button is one of the most redundant screens that I have ever seen. It provides the ability to change the maximum video depth on the Macintosh display. The selections are *default, 1 bit, 2 bit, 4 bit, 8 bit, 16 bit,* or *32 bit.* Why anyone would want to set the video depth of a single INIT to anything other than a default setting (which should check the screen available and display correspondingly) is beyond me.

The naming of the Virex INIT also causes me some concern. In every package I have so far examined which contains an INIT for virus protection it is named so that it has to be first or second INIT which loads itself on system start-up. This is to counter the possibility of other INIT based viruses loading before it if the strict alphabetical sequence is maintained. The Virex INIT starts with a 'v' - making it virtually the last, rather than nearly the first, INIT which loads into my system. Although it is very easy for a virus writer to ensure that his virus will load before a virus protection program (by using invisible characters within the name, for example), I would have expected that some attempt be made by the manufacturers to ensure that Virex is at least one of the first INITs loaded.

In Action

When the INIT has been copied into the system folder the Macintosh should be restarted. In action the INIT actually checks files before they are launched, and is almost invisible apart from the flash of the Virex icon in the centre of the screen while it is working. This is an option which can be turned off (and in my case was).

When a disk is inserted a dialogue box appears stating the fact that the disk is being scanned. Without the flashing Virex icon or the scanning dialogue box, the INIT is virtually invisible. I found that a floppy disk containing only data files is scanned much faster by Virex than by SAM, not because the scanning itself is faster but because the SAM dialogue box seems to remain on the screen much longer after the scan has finished.

There is, however, no way of stopping the scan in mid-stream. This can be frustrating if you are occasionally inserting a nearly-full floppy disk in the drive and do not wish to spend time scanning it, simply because it was scanned 10 minutes previously, and the disk has been locked in the interim. Symantec provided the option to hit *Command-Full Point* to abort a scan within SAM, and this is sorely missed within Virex. The only way to prevent scanning is to configure the Virex INIT to prompt before scanning, but again there are times (particularly when configuring software for users' ease of use and maximum protection) when this is far from ideal.

Inserting a floppy disk infected with nVIR B caused a scan followed by:

```
The disk . . . is infected with a known strain of
the nVIR virus. Please use the Virex application to
repair the infected files, or replace them with
fresh copies.
```

There is then no option but to eject the disk. On insertion of a floppy disk containing several varying viruses, the scan stops at the first virus and then again only allows ejection of the disk.

An extremely good feature of the Virex package, in particular Virex INIT, is its ability to scan files as they are opened and not merely as they are run. Whenever a new file is opened, whichever application you are running, the Virex INIT first checks to ensure that no viral resources are resident in the file. This is not however true when the Virex application is run, as in this case the scanning of files on opening seems to be disabled. This is only true with the Virex application however, and Disinfectant does not cause the Virex INIT to turn off in the "scan on opening file" option. I would not expect the action to be any different.

Few problems were observed while using the Virex INIT. Other INITs in use were - *Boomerang, Pyro, QuicKeys, SuperClock, Shiva Dial-in, Aask, HD Partition, On Cue, Remember?, Shield* (from the package SUM), and *Suitcase II*. The testing was carried out on a 2Mb Apple Macintosh SE running System Version 6.0.2, and Finder Version 6.1.

The Virex Application

When the application is opened, it is obvious it is written to Apple Guide-lines from all points of view. The major functionality of the program (ie diagnosis, repair and help) is done through buttons, and disk selection is done through pointing and clicking on icons. The menu selections are again to Apple Guide-lines with the configuration section of Virex on an options menu. On this menu is the ability to enter expert mode which gives a fourth standard option known as "Record/Scan". This gives the ability to scan internal (and indeed any external) hard disks in order to fingerprint all files resident. **Thus the Virex package has the ability to detect future virus strains.**

A potential problem is the fact that any floppy disk inserted while the Virex application is running will not be scanned by Virex INIT, and therefore, when Virex is quit, that floppy disk will actually be mounted on the desktop. Should the user insert the disk while Virex is running, simply because a previous disk had to be run through the application, the user could then inadvertently quit Virex and that floppy disk will not be scanned. Virex INIT checks by default an application before it runs. However, the basic problem comes from the fact that the Virex INIT will not allow an infected floppy disk to be mounted on the desktop. Therefore if you wish to scan the floppy disk with the Virex application then the INIT must be over-ridden.

This should be compared with the approach of SAM, where the INIT will scan a floppy and allow it to be mounted on the desktop (after a suitable warning has been posted) which therefore means that the running of

the Symantec application does not necessarily have to by-pass the running of the INIT. Therefore a floppy disk inserted while the application is running is still necessarily scanned by the INIT (if that is what the user has selected to do).

In practice this is not a problem for most users, and I appreciate the advantage of restricting a virus infected floppy from being mounted. It merely demonstrates slight differences of approach between the vendors*HJC Software* and *Symantec Corp.*

When the Virex application has finished the scanning for diagnosis, the disk is ejected.

Virex Functionality

Virex correctly detects the following viruses:

Aids, Anti, INIT 29, Mev#, nFLU, nVIR (both A and B strains), *Peace* (the DR variant only, not the RR variant), *Scores*, the *nVIR inhibitor* (flagged as having nVIR resources but not being infected), *Hpat* (flagged as being an nVIR strain which Virex doesn't recognise), and a manufactured nVIR pseudo infection with a renamed resource (flagged as an nVIR strain Virex doesn't recognise). It also detects *cross-infections of various viruses, WDEF* and *J-nVIR.*

Note that no actual J-nVIR virus was available for testing, however, a local version of the virus was made using nVIR B as a base.

Documentation

The documentation is competent. It contains sections on virus basics, as well as information on all common Macintosh viruses. Its description of operating the package is good with plentiful illustrations throughout. Moreover, the manual is consistent and can be used as a reference work. As with most Macintosh software, the documentation is often the last things referred to if you are an experienced Mac user! My one criticism is that the documentation supplied is from Virex 2.0, and the package has changed considerably since then.

Conclusion

The Virex package as reviewed (ie version 2.3) is an extremely professional and competent system of virus protection and removal (if this proves necessary). It is totally up-to-date. Although the latest version of the Symantec package SAM is also reportedly able to detect and combat all known Macintosh viruses (including the recently discovered WDEF), this version has not yet been made available to *Virus Bulletin* for evaluation, and cannot therefore be compared at this stage.

The update service offered by HJC is extremely efficient. Updates are posted at regular intervals (approximately once per month). This service costs $75.00 per year, or $15.00 for a single update. The package itself costs $99.95.

Macintosh Anti-Viral Software

Virex is available from HJC Software Inc., PO BOX 51816, Durham, North Carolina NC 27717, USA.

SAM, Symantec UK, 36 King Street, Maidenhead, SL6 1ES,UK. Commercial utility 69.00. Upgraded regularly. Like Virex, this utility has been updated to combat the WDEF virus and J-nVIR.

Disinfectant, John Norstad, Northwestern University, 2129 Sheridan Road, Evanston, Illinois 60208, USA. Shareware, Free. Version 1.5, released December 14, 1989 will combat WDEF. Available from many user groups and major bulletin boards.

VirusRX, MacUser Userware, PO Box 320, London N21 2NB, UK. Shareware, Free. Distributed by Apple Computer and available from retailers and bulletin boards.

Interferon, MacUser Userware, PO Box 320, London N21 2NB, UK. Shareware, Charity Donation.

Virus Detective, Jeffrey Schulman, PO Box 50,Ridgefield, CT 06877, USA. Shareware $25.00. Available from bulletin boards and regularly updated.

PRODUCT EVALUATION

Dr. Keith Jackson

Iris Anti-Virus Software Version 1.22

Iris Anti-Virus software detects and eradicates computer viruses, its functions are contained within two utilities : 'IMMUNE' and 'CURE'.

Documentation provided with Anti-Virus (four pages of A5) states that IMMUNE "executes various tests to search for various types of virus". On completion of these tests, a memory resident program is loaded to protect against future virus attacks. CURE is described as a program which "destroys all viruses which are present in your computer". As for the IMMUNE program, little detail is given, apart from a statement that CURE scans files to detect viruses. The documentation says:

> While the system is operating, several detailed messages are displayed. These messages are intended to warn you against an attempt by a virus to infiltrate your system. As such you should heed these warnings and act accordingly.

That's it. There is nothing else in the documentation to help users with errors, queries issued by the software, or problems which might occur. I find such a lack of explanation alarming.

A perplexing feature of IMMUNE is that it loads a "special" program into memory. There was no explanation about how much memory this program would take, what interrupt vectors it might commandeer, what other memory resident programs would be compatible, or even what it was going to do.

During installation of Anti-Virus you must leave the floppy disk write protect label off - a bad practice. Although not stated anywhere explicitly, the Anti-Virus software is copy protected. On examining the Anti-Virus floppy disk, I found that it contained 55 Kbytes of bad sectors, and two Read Only, Hidden, System files (IMMUNE2.INF and CURE.INF). Neither the copy protection scheme, nor the presence of these bad sectors is mentioned in the documentation. Copy protection prevents proper backups being taken. If such schemes are being imposed they should be explained in full. Users will be justifiably suspicious that the software might contain a virus when there are unexplained bad sectors on the floppy disk.

I was reluctant to install this software on a hard disk. Problems can and do occur with copy protection schemes. Therefore I first tested Anti-Virus on a dual floppy disk computer borrowed for the purpose. All proceeded satisfactorily. 'IMMUNE' installed a memory resident program, and this co-existed happily with my usual memory resident programs (NDOSEDIT and Sidekick). If this had not been the case, the documentation provides no trouble-shooting advice to sort out clashes between memory resident programs. CURE scanned through a floppy disk containing 28 user files (256 Kbytes) in 33 seconds and said "No virus detected".

Problems occurred when I braved the copy protection scheme and installed Anti-Virus on a hard disk. In the first instance I had to use another computer, as my own PC has a 3.5 inch disk drive as drive A. Anti-Virus came on a 5.25 inch (360K) floppy disk. Being copy protected it was impossible to make a 3.5 inch copy, so I had to borrow a computer which had a 5.25 inch floppy disk as drive A, and a hard disk.

When hard disk installation commenced the message "*IRILOCKH software protection program Ver 1.3, Unable to create directory*" appeared twice in succession. I eventually surmised that this was after all correct, and the copy protection scheme was complaining that I had created the Anti-Virus subdirectory for it in advance, when it wished to create it for itself. I did not test whether the copy protection scheme worked.

After installation was complete, Anti-Virus commenced checking. I had borrowed the computer and did not dare scatter virus infected files across someone else's hard disk. Only floppy disks were tested. Anti-Virus states that it is looking first for "Class I" viruses, then for "Class II viruses". It omits to state what these classes are, and the documentation says nothing about them. I surmise that they refer to boot sector viruses and parasitic viruses, but which way round?

A test-bed of viruses appears in the Technical Information section (see below). With the exception of the 405 virus, CURE detected all of the viruses in the list, and removed them. The only error message which occurred was "*4051* (sic) *error in removing virus*". CURE detected 405 but could not remove it. There was no indication by CURE of which virus was being removed (except if an error occurred). Each time a virus was detected, the message "*infected with a virus*" appeared, followed by a message stating that the affected file was being 'cleaned'. I fully accept that users may not want to know the technical ins and outs of viruses but they will certainly want to identify which virus has infected the PC.

The Italian virus and Brain virus (boot sector viruses) were detected when a directory listing of the relevant disk was

produced. The Anti-Virus memory resident program popped up a window on the screen, and warned that an infected disk was being used. Again there was no indication of what the disk was infected with. The memory resident program must know which virus had been detected, so why not inform the user?

Operation of the Anti-Virus software proceeded smoothly and it removed viruses from files extremely efficiently. I noticed that the default mode for CURE is to remove viruses. It is possible for CURE to operate in detection mode only. Automatic removal of viruses is far too dangerous for ordinary users. What happens when a file is wrongly diagnosed as a virus, as by chance it contains a virus pattern? The default mode of operation should be to detect viruses, not to remove them.

A virus scanning program was provided along with Anti-Virus. This is a shareware program. The help switch of the executable program provides brief list of the switches available on VIRUSEARCH.

There were two files on the VIRUSEARCH disk: VIRSRCH.EXE, and IMSGS.TXT. The former is obviously the executable file which scans for viruses, and the latter was found to contain the text messages used by this scanning program. The text messages contained no virus patterns, so I surmise that these are contained within VIRSRCH.EXE itself. The virus patterns must be stored in encrypted form, as neither VIRUSEARCH itself, nor any of the scanning programs which I used for comparison purposes, reported the presence of a virus on this disk.

A list of viruses which VIRUSEARCH is familiar with is shown in the technical details below. This list was found within the file IMSGS.TXT. It comprises 20 individual viruses, with variations taking the total up to 25. The executable file VIRSRCH.EXE was dated 29th Nov. 89.

VIRUSEARCH took 23 seconds to check a 3.5 inch (720K) floppy disk containing 24 files (49 Kbytes). VIRUSEARCH correctly found viruses in 18 out of 20 infected files. VIRUSEARCH took 69 seconds to check my hard disk containing 515 files in 9.05 Mbytes. I used SCAN50 from McAfee Associates, and Sweep from Sophos Ltd. for comparison purposes. Both of these search for just over 50 viruses, and the times taken to scan a disk were proportionately longer (3 to 4 minutes for the hard disk). Given the number of viruses searched for, the times of the three programs were roughly comparable.

In conclusion VIRUSEARCH operated correctly, but I was unimpressed by the number of viruses known to it and would have appreciated documentation. The Anti-Virus software readily detected viruses, and removed them very efficiently. The resident immunisation and cure programs were extremely effective. However, the version tested was let down by inadequate documentation and the use of a copy protection scheme which inhibits backups and reduces user confidence. If these two problems are rectified then Anti-Virus software would be worth considering as it has great technical merit. (*See below, Ed.*)

Technical Details

Product: Iris Anti-Virus Software

Developer: Iris Software & Computers, 6 Hamavo St., Givataim 53303, Israel, Tel. +972 (3) 5715319, Fax +972 (3) 318731.

Availability: IBM PC/XT/AT, PS/2, or any close compatible running MS-DOS version 2.1 or above.

Price: $99.00

Version evaluated: 1.22

Serial number: IAV11688

List of viruses known to Anti-Virus and VIRUSEARCH, decoded from internal inspection of file(s). Some of the names are unknown, but this is probably just a nomenclature problem: Black Friday 1,2/ April 1st/Israeli/Alabama/Mix1/Trace Back 1,2/Cascade 1,2/ Data Crime 1,2 Saratoga 1,2 /Suriv 1,2,3/Fu Manchu/Invisable/ Vienna/405/Ping Pong/Pakistan/Brain/Mispeller/Marijuana

Hardware used:

a) ITT XTRA (a PC compatible) with a 4.77MHz 8088 processor, 640K of RAM, one 3.5 inch (720K) drive, two 5.25 inch (360K) drives, and a 30 Mbyte Western Digital Hardcard, running under MS-DOS v3.30.

b) COM-PRO, a dual 5.25 inch floppy disk PC compatible with an 8088 processor (unknown clock speed), 640K of RAM, running under MS-DOS v3.30.

c) As b) but with only one floppy disk, and a 20 Mbyte hard disk.

List of viruses used for testing purposes: Brain/Italian/Vienna/ Jerusalem/Datacrime I/Vienna (1)/Cascade (1),(2)/ Datacrime II/ 405/Fu Manchu/Jerusalem/Traceback/Suriv 1.01/Suriv 2.01/ Suriv 3.00/South African (1),(2).

Iris Software & Computers have informed us that the latest release of Anti-Virus Software (Version 2.4) includes a substantial expansion of both physical and program documentation, removal of the copy protection scheme, no file removal by default, and on screen virus identification to assist the user. (Ed.)

NEWS

A one day seminar '**Computer Viruses, the Threat Intensifies**' is being held by PC Business World on February 29 in London. Speakers include Det Sgt Barry Donovan of the Computer Crime Unit, New Scotland Yard and Jim Bates. Details from Clare Peiser at Quadrilect, UK. Tel 01 242 4141.

ADAPSO, the Computer Software & Services Industry Association have just published a comprehensive and up-to-date book entitled **Computer Viruses: Dealing with Electronic Vandalism and Programmed Threats.** It is available from ADAPSO, 1300 North Seventeenth Street, Suite 300, Arlington, Virginia 22209, USA. Tel USA 703 522 5055.

The promised review of the Computers & Security Computer Virus Handbook has been delayed and will be published in the March edition of Virus Bulletin.

The Symantec Corporation have released **version 1.4 of SAM** (VB, October 1989). The version is said to detect current or mutated strains of the WDEF virus which VB reported in January. Symantec (UK) Ltd, Tel 0628 776343.

Dr Alan Solomon's S & S Entreprises has launched a newsletter devoted to computer viruses, worms and Trojans. Information is delivered to subscribers fax machines. Details from Ray O'Connell, Virus Fax International. Tel 0494 791900.

The Computer Threat Research Association (**CoTRA**) is to hold its first annual conference '**Conquering the Threats to Computers**' at the London Press Centre, February 21, 1990. The conference will address software piracy, hacking, threats to Macintosh systems, computer law and backup techniques. Details from Anabelle Simpson, IBC technical Services. Tel 01 236 4080.

The Allstate Insurance Co., in the United States is to offer general insurance against destruction of programs and data caused by viruses. The company announced that, with immediate effect, its home and business computer insurance policies are extended to cover virus damage to PCs. The company's standard data protection policy has been amended at no extra cost to the customer.

The UK firm International Data Security (IDS) has launched a **virus protection and recovery consultancy** as part of its PC security services. Tel 01 631 0548.

VIRUS BULLETIN

Subscription price for 1 year (12 issues) including delivery:

US$ for USA (first class airmail) $350, Rest of the World (first class airmail) £195

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, Haddenham, Aylesbury, HP17 8JD, England

Tel (0844) 290396, International Tel (+44) 844 290396
Fax (0844) 291409, International Fax (+44) 844 291409

US subscriptions only:

June Jordan, Virus Bulletin, PO Box 875, 454 Main Street, Ridgefield, CT 06877

Tel 203 431 8720, Fax 203 431 8165