October 1990 ISSN 0956-9979

VIRUS BULLETIN

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

CONTENTS

EDITORIAL

Order Out of Chaos?

"Giving you order out of chaos" - the slogan of *Symantec*, the latest contender to enter the 'virus industry' war. *Symantec's* recent takeover of *Peter Norton Computing*, one of the most respected names in personal computing, guaranteed a media spotlight for the company's new product, the Norton Anti-Virus.

The announcement that Norton Anti-Virus would be launched on 17th September gave rise to great expectations. Observers hoped that this product would prove as valuable in fighting viruses as the excellent Norton Utilities and Commander packages had proved to disk inspection and data recovery. Early announcements from *Symantec's* UK chairman Mr. Michael Skok claimed that Norton Anti-Virus was the "the fastest and most complete virus protection utility on the market" while press releases assured potential customers that the PC package, when combined with the company's existing SAM anti-virus for Macintosh computers, would provide "the total data security strategy". Skok was quoted in the computer press as saying that the *Symantec* product could scan a 40 Megabyte hard disk in 26 seconds, a figure which he proceeded to use to denigrate Dr. Solomon's Anti-Virus Toolkit which appears to have been targeted as the major competitive product in the UK.

The marketing launch for the Norton Anti-Virus was unprecedented for a computer security software product - 50,000 demonstration disks were distributed with *What Micro?*, advertisements appeared in the *Sunday Times* and a tastefully designed booklet which purported to explain the computer virus problem was carried as an insert in the *Financial Times*.

The launch campaign got off to an inauspicious start. *Symantec* had planned to distribute a discount voucher in the form of a fake £10 note with its NAV demonstration disk - these had to be withdrawn following the disapproval of the *Bank of England*. However, worse was to follow. *Symantec's* attendance at the *Business Computing '90* exhibition at Earls Court, London (25-28 September), was to be the forum for a public launch of the product. Exhibition attendees were invited to submit diskettes for inspection so that they could be guaranteed "virus free" by Norton Anti-Virus and adverts to this effect were run in the British national press.

It was the technical editor of *PC Business World*, Mr. Mark Hamilton, who 'picked up the gauntlet'. His inspection of Norton Anti-Virus had revealed several shortcomings in the product, which was found to be anything but "comprehensive". Journalists and editorial staff at *PC Business World* have shown a laudable cynicism amid a deluge of virus hype during the last year. This may be explained by the fact that *PCBW's* parent company, *IDG Communications*, was subjected to the commercially pernicious AIDS Information Diskette extortion bid which *VB* reported in December 1989.

Hamilton selected 11 computer virus samples from his collection based on the criteria that each sample should be 1) *causing infections in the wild regardless of geographical region*, 2) *non-encrypting* and 3) *well known to the computer virus research community*. With an independent witness present, infected files were submitted for inspection by the Norton Anti-Virus: **all were passed as virus free**. In fact, the product had failed to identify files infected by Eddie II, Vacsina (1), Yankee, MIX1, MIX1-2, Pixel 1 and 2, Amstrad, and three variants of the Vienna virus.

The extravagant claims and idle boasts that have emanated from *Symantec* about this product may now be of interest to the *Advertising Standards Authority*. The failure of the product to detect well known computer viruses certainly demonstrated that *Symantec's* initial research and development in support of this product was lamentable. Hexadecimal patterns for all of these viruses, and many others which the Norton Anti-Virus fails to detect have been available through the *Virus Bulletin* for months. In blunt terms, this product has been shown conclusively **not** to be "the most complete virus protection utility on the market". The developers of a number of newer anti-virus products have failed to establish links with the research community in order to sustain the necessary updates to their software, and presumably this is the reason behind this product's shortcomings.

With the realisation that computer viruses have become big business, we can expect further 'big names' to muscle in, develop a market and try to 'milk' it, regardless of the complexities involved in developing anti-virus software. It is to be hoped that potential purchasers of this type of software will ignore the marketing men and demand high standards of product development, reliability and end-user support.

European Certification Schemes

The need for independent assessment of computer security products, which will help computer users tread their way through the minefield of marketing hyperbole, has been addressed, in part, by the establishment of government certification under the auspices of the UK's *Department of Trade & Industry*. The certification process is controlled by the *Communications-Electronics Security Group* at *GCHQ*, Cheltenham. Evaluation currently takes place at two *CESG* licensed evaluation facilities (CLEFs) run by *Secure Information Systems* and *Logica Space & Defence Systems*.

The purpose behind certification is to establish standards for security products used within government departments and to enable outside organisations to base their purchasing decisions on objective criteria.

Moves to introduce a harmonised evaluation protocol for France, Germany, the United Kingdom and Holland under the *Information Technology Security Evaluation Criteria (ITSEC)* are currently under way.

Information on existing schemes and about *ITSEC* proposals is available from the following addresses:

France

Service Central de la Securite des Systemes d'Information-Division Information et Systemes
18 Rue du Docteur Zamenhof
F-92131 ISSY LES MOLINEAUX

Germany

Zentralstelle fur Sicherheit der Informationstechnik
Am Nippenkreuz 19
D-5300 BONN 2

Holland

Netherlands National Comsec Agency
Bezuidenhoutseweg 67
PO Box 20061
NL-2500 EB THE HAGUE

United Kingdom

Head of UK CLEF Scheme Certification Body
CESG Room 2/0805
Fiddler's Green Lane
CHELTENHAM, GLOS GL52 5AJ

Macintosh Virus Writer Apprehended

New York State Police have apprehended the author of four Apple Macintosh viruses. The virus author, whose name has not been released, has confessed to creating the MDEF, MDEFB and CDEF viruses (*see VB, Sept 1990, p.21*) as well as a further MDEF variant which was never released. Extensive investigations by Mr. Mark Anbinder, a computer specialist at *BAKAComputers (*sic*), Inc.*, Ithaca, New York, identified the virus writer as a student at *Ithaca High School*.

The *Tompkins County District Attorney's* office is declining to prosecute, based on the author's willingness to cooperate with the police investigation and to provide source code from all four viruses for expert analysis.

By coincidence, Ithaca, New York, is where Robert Tappan Morris released the Internet Worm in November 1988 (*VB, June 1990, p. 13*).

(*Technical details about MDEF and CDEF appear in the Known Apple Macintosh Virus Table, pp.6-7*)

TECHNICAL NOTES

Whale - The Armoured Virus

The Whale computer virus (aka Motherfish) is the most complex virus program so far encountered. Analysis suggests that it is the work of a professional programmer or group of programmers, possibly sponsored by an agency. Approximately 80 percent of its code (the virus is 9K in length) is dedicated to confounding disassembly. Reports indicate that Whale is related to the Fish6 virus and that the two viruses react together when active on the same processor. Tests will be undertaken to ascertain whether this is the case. Programming techniques seen in the 4K virus are also present in Whale, suggesting that the writer(s) of Whale had access to the former virus, or wrote it. Beneath three layers of encryption a single text message has been located:

```
THE WHALE IN SEARCH OF THE 8 FISH I AM '~knzyvo}'
IN HAMBURG addr error D9EB,02
```

There has been some speculation that Whale is an intentional ploy (on the part of its creator(s)) to test the speed of response of the anti-virus community. It was hoped that a full report on this virus would be available for publication in this edition but a technical report will now be delayed until November. **The virus represents a new strategy combining 'stealth' tactics with concerted attempts to hinder disassembly and delay attempts to update scanning software.** (*See pages 12-14*). The term 'armour' is used to describe code which persistently attempts to foil debugging software.

Flip

David Chess at the *IBM Thomas R. Watson Research Center*, USA, has provided an explanation for the program targeting of the Flip virus described in last month's *VB*. He points out that the pattern which the virus searches for

```
8B16 XXXX 8916 XXXX 8B16 XXXX 8916
```

is consistent with the following code-fragment, but not the one which was published in the September edition:

```
mov     dx,[xxxx]
mov     [xxxx],dx
mov     dx,[xxxx]
mov     [xxxx],dx
```

This pattern can be found in most, if not all, versions of COMMAND.COM. Patching COMMAND.COM at this location with the INT 9FH call will have the effect of hiding the increase in size of infected files when a DIR command is given. It is thus an additional 'stealth' feature of the virus.

The report also stated incorrectly that infected COM files were incapable of transmitting infection. Subsequent checks on our

test machines show that this occurred due to the presence of specialist device drivers. Normally configured PCs will enable COM files to be infected in exactly the same way as EXE files. Early in the code, the virus checks the stack pointer and if this is below an expected value (as in the case of the test machines) the file is not infected.

Joker's Decoy

David Chess has also discovered the 'key' to make the Joker program (reported last month) activate. The program needs to be renamed from JOCKER.EXE to WABIKEXE.EXE before it is run. 'Wabik' is Polish for 'decoy'.

Pattern Changes

The hexadecimal pattern for the Joshi virus (*VB, September 1990, p. 6*) has caused an unacceptably large number of false positive indications It should be replaced by the following pattern:

```
Joshi BC00 F0FB A113 04B1 06D3 E08E C0B8 0002;
Offset 28
```

A second version of the Den Zuk virus has been received by *VB*. New patterns for both examined versions of this virus appear below.

```
Den Zuk (1) 32E4 CD13 720D 33D2 B921 28BB 007E
B806;Offset 064
```

```
Den Zuk (2) 32E4 CD13 720D 33D2 B921 28BB 007E
B809;Offset 078
```

Self-Modifying Encryption Code

A few viruses have appeared recently where the extraction of a 16-byte virus identification pattern has not been possible as the viruses use self-modifying encryption routines. The 1260, Casper, Phoenix, Suomi and Whale (Motherfish) viruses fall into this category. Some other method must therefore be used to detect infected files.

In some cases it is possible to provide a pattern incorporating 'wildcards', where some characters are replaced by question marks, but in other cases the virus may insert a variable number of 'garbage' bytes between the instructions used for decryption. The method used by the 1260 virus (which is the same as that used by the Casper virus) was described in *VB, March 1990*. We hope to describe the method(s?) used by the Whale virus in November. To assist in the detection of Suomi and Phoenix, the encryption algorithms they use will also be described next month.

Analysis of these viruses is continuing; reports on them have been delayed due to their complexity.

Self-modifying viruses are the single greatest challenge to virus-specific scanning software. Unfortunately their number is likely to increase.

Virus Mutations

If a computer virus is modified for some reason, the modification(s) will be found in all programs it infects. Modifications are usually deliberate, but there is a remote chance that a random error may occur, altering one or more bits, whenever the virus is written to disk or read back. Random changes will result in a 'mutated' variant, which often suffers from the same affliction as mutant living organisms - an inability to reproduce (or in this case replicate). When the replication ability is not impaired, the virus may spread, sometimes just as fast as the original. The mutated version may even be able to bypass some virus-scanning programs where the changed bit(s) fall within the chosen identification pattern.

There are several mutated viruses known today, for example two mutations of the Cascade (1704) virus. One, 17Y4, may cause an infected machine to crash. The other is unable to detect previous infections and will infect the same program repeatedly.

Developers of virus-scanning products can prevent a random mutation from invalidating their search patterns by using at least two different patterns from each virus

Multiple Infections

Most viruses are able to recognise exisiting infections and avoid multiple reinfection of the same file. A well known exception is the Jerusalem virus, but the 405 virus may also reinfect an infected file. An interesting sample was made available to the *Virus Bulletin* recently. When it was first examined, it seemed to contain a single 10005 byte virus, as infected COM files always grew by 10005 bytes.

A closer examination revealed an extraordinary phenomenon. Part of the added code consisted of an ordinary Jerusalem virus shrouded on either side by two 4096 byte blocks each containing the Plastique virus. By coincidence, the lengths of the viruses combined add up to a 'nice' rounded number:

4,096 + 1,808 + 4,096 = 10,000

This 10,000 byte block is located in front of the original program, but a remaining 5 bytes are appended by the Jerusalem virus.The course of events when a file becomes infected appears to be as follows:

1. The first copy of Plastique is activated and appends itself to the front of the target file.

2. Jerusalem appends itself to the beginning of the Plastique infected file but adds its 5 byte signature to the end of the file.

3. The second copy of Plastique activates, does not find its signature at the beginning of the target file, (because this area is occupied by Jerusalem virus code), and therefore infects it.

The moral of this story is that anti-virus scanning programs will have to manage combined infections by different viruses. Even if the identification pattern is found, it is no guarantee that the virus can be removed using disinfection software. **The simplest and safest strategy to eradicate infection by a parasitic virus is simply to delete the infected file**. The DOS DEL command renders the file uninvokeable by the operating system but virus code will remain in clusters accessible by Norton or other disk utilities. **For absolute security these clusters can be positively over-written.**

Virus Collisions

The methods viruses use to detect whether a file has already been infected are not necessarily unique, as two different virus writers may employ the same method. Files may be marked as infected in two ways: by adding a signature at a specific location, or by changing the information about the file in the directory entry. The latter method is used by several viruses which can lead to 'collisions'. Three viruses mark the 'seconds' field of the timestamp to 62. They are Vienna (and nearly all Vienna variants), Zero Bug and Eddie II (651). A file infected by one of them will look infected to the other two and is therefore immune to further infection by these or other viruses using this particular signature. This illustrates an important aspect of **false positive** versus **false negative** indications. To a virus, a false positive (indicating that the file is infected when it is not) is of no concern. A false negative (indicating that the virus is not present when in fact it is) is of more concern as multiple reinfection of the file may result. To an anti-virus program, on the other hand, a false positive may cause unnecessary alarm and even damage in the event that disinfection software is used on a clean file. A false negative is disastrous, as the software will have failed to detect infection.

Editorial Policy - Boot Sectors

The terminology used in *VB* to describe boot sectors has not been entirely consistent with itself or other reference sources. In order to avoid confusion in the future, the following terms will be used:

Master Boot Sector. This is the sector stored at absolute track 0, head 0, sector 1.

DOS Boot Sector. This is logical sector 0 in each DOS partition.

Partition Record (aka *partition table*, aka *boot record*, aka *master boot record*). This is the 64 byte table contained at the end of the Master Boot Sector.

IBM PC VIRUSES

Amendments and additions to the *Virus Bulletin Table of Known IBM PC Viruses* as of 28 September 1990. The full table was last published in the August 1990 edition of *VB*.

Hexadecimal patterns can be used to detect the presence of the virus with the 'search' routine of disk utility programs or, preferably, can be added to virus scanning programs which contain pattern libraries.

Filler - DR: A Hungarian virus with unknown effects.

```
Filler CD12 BB40 00F7 E32D 0010 8EC0 BA00 00EB;
Offset 074
```

Phoenix - CR: This Bulgarian virus is 1701 bytes long, but another variant with a length of 1704 bytes has been reported. Despite the identical lengths, the virus is not related to the Cascade virus. Phoenix uses encryption which renders the extraction of a single identification pattern impossible.

Plastique 5.21 - CER: A 4096 byte virus from Taiwan. An earlier version, 3012 bytes long has also been reported. Virus awaiting disassembly.

```
Plastique 5.21 C08E D8A1 1304 B106 D3E0 8ED8 33F6
8B44
```

Violator - CN: This is an unusually long variant of the Vienna virus, 1055 bytes in length. The activation date was 15 August 1990, but the virus has not been fully analysed.

```
Violator BF00 01F3 A48B F2B4 30CD 213C 0075 03E9;
Offset 00E
```

Whale, Motherfish - CER: An 'armoured' virus in which approximately eighty percent of the code is included to disrupt disassembly. Virus uses multiple layers of self-modifying encryption. The virus has been reported as 'undetectable' by virus-specific scanning methods. Virus length is 9 kilobytes. Undergoing disassembly.

Reported Only

Arema - DR: Reported in Indonesia. A variant of Den Zuk.

Freddy - CR?: Infects IBMBIO.COM

Hacker - DR: This virus from Indonesia is probably identical to the Ohio virus.

PC-Club - DR: Reported in Indonesia. Said to display a message every 30 minutes.

PC-Monster - DR: Closely related to Den Zuk.

Robert/Narvin - DR: An Indonesian virus. Displays graphics on the screen.

Semione and Keongz - DR: An Indonesian virus. Based on Den Zuk, but with sound effects.

Supernova - DR: A harmful virus from Indonesia - it will format the disk when the printer is used.

KNOWN APPLE MACINTOSH VIRUSES

Information Systems Integrity and Security Ltd

The following is a list of the known viruses affecting Apple Macintosh computers. Each entry includes the name (and aliases) for the virus; a short description of symptoms; together with the characteristic resources which can be used to detect the virus' presence.

Name	Family	Description
nVIR A	nVIR	When an infected application is executed nVIR A infects the system file (adding an INIT 32 resource), thereafter any reboot will cause the virus to become resident in memory, after which any applications launched will become infected. There is a delay period before the virus will begin to announce its presence. This announcement is made once every 16 reboots or 8 infected application launches by either beeping or using Macintalk to say "Don't Panic".
nVIR B	nVIR	Similar to nVIR A but does not utilise Macintalk if installed. Beeps once every 8 reboots or 4 application launches.
Hpat	nVIR	All clones of nVIR B produced by altering the resource names of the auxilliary ''nVIR'' resources created by the virus.
AIDS	nVIR	
MEV#	nVIR	
nFLU	nVIR	
Jude	nVIR	
Fuck	nVIR	
nVIR C	nVIR	A forerunner to the nVIR strains. This strain is believed to delete files randomly from the System folder. nVIR A or B strains will replace this strain on infection. This strain is believed extinct.
DR	Peace	Also known as the Drew or MacMag virus. The virus does not infect applications but only propagates to the System file on hard or floppy disks. The virus was designed to display a message of world peace on March 2nd, 1988 and then delete itself from the System file. It is believed to be extinct.
RR	Peace	An earlier strain with differing resource patterns.
Scores	Scores	When an infected application is executed Scores will infect the system file, note pad and scrapbook files; the icons for the last two are changed to a generic document icon. In addition two invisible files are created, named Scores and Desktop. Following this stage a reboot will cause the virus to become active in memory. Two days after infection of the system file the virus will begin to infect any application run within 2 to 3 minutes of its launch. After four days any applications run with "ERIC" or "VULT" resources will cause a system bomb (ID=12) after 25 minutes. After seven days any application with "VULT" resources will find its disk writes returning system errors after 15 minutes of runtime.
INIT29	INIT29	When an infected application is run INIT 29 will infect the system file and patch the open resource file trap. Any action which opens the resource fork of an application or data file will cause the fork to be infected. Note that this virus does not require an application to be run for it to be infected. Only infected system files or applications will spread the virus. This virus attempts to infect any newly inserted (or mounted) disk causing the message "This disk needs minor repairs" if it is write protected. Sporadic printing problems may also be encountered.
ANTI	ANTI	This was the first virus on the Mac which does not add new resources on infection, the virus instead appends its code to the CODE 1 resource of the application being infected. When an infected application is run, the virus will install itself in the system heap, and thereafter infect any application which is launched or has its resource fork opened. Unlike other Mac viruses it does not infect the system file, and will thus only become active in memory when an infected application is run. Anti does not spread under multifinder. This virus is also designed to execute automatically a code block from a floppy disk carrying a special signature marker.
WDEF A	WDEF	The code for this virus is stored in a WDEF (window definition code resource) in the invisible desktop file on each disk. When a disk is inserted all resources in the desktop resource fork are added to the search list for system resources thus displacing the standard (innocent) WDEF in the system file. When a window is opened the viral WDEF code is executed, 1 in 11 times this will cause the viral WDEF resource to be copied to the desktop of all mounted disks. During execution of the virus it bypasses most anti-viral protection INITs by patching the trap table to call resource manipulation routines direct from the system ROM.
WDEF B	WDEF	This is an early debugging version of WDEF A which will beep on infection of desktop files.
CDEF	CDEF	Using similar techniques to the WDEF virus this simpler virus spreads by adding a viral control panel definition resource (CDEF) into the desktop file. This resource will be added to the search list for system resources in the same way as WDEF. The virus infects the desktop on all active disks. Both the CDEF and WDEF strains can be removed by rebuilding the desktop file.
MDEF A	MDEF	This virus uses a viral menu definition resource (MDEF) as the carrier. When an infected application is run the virus will change the id of the standard system MDEF resource to 5378, adding its own MDEF 0 resource to the system file. Applications will become infected when the menu manager executes this viral code resource. This will cause a copy of the MDEF 0 to be added to the virus' resource fork. Vaccine will block the operation of this virus incompletely, and will cause loss of the system menu definition function, and failure of the menu manager. The name of the added MDEF 0 provides the popular name ''Garfield'' for this virus. The virus will crash the Mac128K and 512K on infection.
MDEF B	MDEF	This variant of the MDEF A virus has been modified to evade detection by the vaccine anti-viral protection INIT. Its MDEF 0 resource is named "Top Cat". A further variant of the MDEF virus with an MDEF 0 named "Heathcliff" has been reported, although this strain may not have been released.
ZUC	ZUC	Infects applications by appending its code to the CODE 1 resource of the infected file. When active the virus has a 1 in 4 chance of infecting each mounted volume. The application signatures in the desktop file for the volume are used to locate applications to be infected, the virus also has a 1 in 16 chance of scanning the disk hierarchy exhaustively to locate target applications. The virus installs a vertical blanking interrupt. After 90 seconds this interrupt will cause the mouse cursor to scan diagonally across the Mac screen. The virus carries signatures of well known anti-viral products, and will avoid infecting such products. It also attempts to bypass protection INITs by using stored ROM addresses for key functions.
Aladin (*sic*)	Aladin	Reported by the *University of Hamburg* virus catalogue project. This virus infects only Atari ST systems running the Aladin Mac emulators. The virus will add a CODE resource to the infected file. After a variable delay the virus will intercept all printing operations.

Frankie Aladin Variant strain of Aladin, also restricted to the Atari ST emulator. This strain will display a bomb and the message "Frankie says: no more software piracy" on activation, followed by a system crash. A variant strain capable of running on standard Macs has been reported by the *University of Hamburg*.

Characteristic Resources Added by Viruses

In the table below "n" refers to the resource number of the first unused CODE resource id in the application's resource fork. Resource name, number and the resource length are provided. = indicates the corresponding file type which is infected and the indicated file resource will be present.

Virus	Resource	System File	Application	Data Files	Note pad	Scrapbook	Desktop
nVIR A	INIT 32 366b	✔					
	CODE 256 372b		✔				
	nVIR 0 2b	✔					
	nVIR 1 378b	✔	✔				
	nVIR 2 8b		✔				
	nVIR 3 366b		✔				
	nVIR 4 372b	✔					
	nVIR 5 8b	✔					
	nVIR 6 868b	✔	✔				
	nVIR 7 1562b	✔	✔				
nVIR B	INIT 32 416b	✔					
	CODE 256 422b		✔				
	nVIR 0 2b	✔					
	nVIR 1 428b	✔	✔				
	nVIR 2 8b		✔				
	nVIR 3 416b		✔				
	nVIR 4 422b	✔					
	nVIR 5 8b	✔					
	nVIR 6 66b	✔	✔				
	nVIR 7 2106b	✔	✔				
Scores	INIT 6 772b	✔			✔	✔	
	INIT 10 1020b	✔					
	INIT 17 480b	✔				✔	
	atpl 128 2410b	✔					
	DATA -4001 7026b	✔					
	CODE n+1 7026b		✔				
INIT 29	INIT 29 712b	✔		✔	✔	✔	✔
	CODE n 712b		✔				
Peace RR	INIT 6 1832b	✔					
Peace DR	INIT 6 1908b	✔					
WDEF A	WDEF 0 1836b						✔
WDEF B	WDEF 0 unknown						✔
CDEF	CDEF 1 510b						✔
MDEF A	MDEF 0 314b	✔	✔				
MDEF B	MDEF 0 532b	✔	✔				

Notes on Resources

nVIR clones - have the nVIR auxiliary resources renamed to the clone name, ie "AIDS". The Hpat virus adds a CODE 255 resource in place of the CODE 256.

Anti and ZUC - both append viral code to the CODE 1 resource in the target application extending it by 1344 and 1256 bytes respectively.

Scores - "Desktop" and "Scores" files also contain copies of the INIT 10, atpl 128 and DATA -4001 resources.

MDEF A - the system MDEF 0 resource is renumbered to MDEF 5378.

FEATURE

Fridrik Skulason

Defining The Jerusalem Variants

The Jerusalem virus has been subjected to more 'hacking' than any other computer virus resulting in a series of minor variants. This has caused problems with identification and formal nomenclature - few scanning programs concur upon the names of these variants.This article is a belated attempt to end the confusion.

The Jerusalem virus is one of the oldest computer viruses. It was first reported in December 1987, but it was probably written a few months earlier, as it contains code which prevented it from activating on Friday 13th November 1987. It appears that the author wanted to allow the virus time to spread before its first activation in May 1988.

Jerusalem infects COM files (which increase in length by 1813 bytes) and EXE files (which increase in length by 1808 bytes). The virus will also infect overlay files starting with the characters MZ (4D 5A in hex notation) which will be mistaken for EXE files. (*The virus was described in VB, July 1989.*)

Jerusalem has now spread worldwide and is among the most common viruses currently in circulation. Numerous variants have appeared, most of which are described here. In some cases the differences between two variants are very minor, indicating that one may have been created by patching the other. There are other variants which have clearly been created by disassembling an infected file, modifying the assembly and reassembling. Patched variants are always the same length as the original virus (1808/1813 bytes) but variants in the second category generally have different lengths. This article will describe all the 1808/1813 byte variants as there is some confusion regarding their naming.

sURIV 3 (*VB, August 1989*) is probably the oldest variant although it was discovered later than some of the other variants described here. The virus is very rare, if not extinct, and only interesting as the ancestor of later variants, in particular, its immediate descendant, known as **sUMsDos**, in which several minor changes appear. Apart from the change of the virus' recognition signature, the time between infection and the slowing of the machine, (which is the effect of many of the Jerusalem variants) is increased from 30 seconds to 30 minutes. An error which prevented the virus from deleting infected files when it triggered has been corrected, but another error which caused EXE files to be infected repeatedly remains. This is because the string 'sUMsDos' is not present at the end of infected EXE files.

The original sUMsDos variant has served as the basis for several common variants. In **Jerusalem C**, only a single byte has been changed - byte 242H now contains EBH instead of 75H. This changes a conditional jump instruction to an unconditional, which disables the slow-down effect on processing because the delay loop is never executed. This effectively decreases the likelihood of early discovery. This variant was originally reported as *New Jerusalem*, but this name has also been used for a "reported only" variant from Holland.

A variant named **Puerto** has been reported. The virus is understood to have been isolated in Puerto Rico in June 1990. Analysis of available samples failed to show any differences between it and the standard sUMsDos variant.

Payday is closely related to sUMsDos - so closely that only a single bit has been altered. This results in a different activation date - any Friday but the 13th. This made it very easy to detect and the virus is believed to have been eradicated. The byte which has been altered is 1CEH, 75H becoming 74H.

In another variant, commonly known as **Jerusalem-B**, the signature string sUMsDos is replaced with a string of zeros. This was probably done to prevent detection. Two side effects, one trivial and the other more significant, resulted from this change. First, the virus will not infect programs ending in a string of zero bytes. Secondly, the multiple infection error is partially cured, at least when the 'garbage' area at the end of the virus happens to end in a corresponding string of zeros. Jerusalem-B variants often contain the string 'ANTIVIRUS' at the beginning of the garbage area. Jerusalem-B is probably the most common of all PC viruses.

The **A-204** variant, which is believed to have originated in Delft, the Netherlands, is also closely related to sUMsDos. Apart from the change of the signature string to '*A-204*', the only difference is a reordering of a couple of instructions. This reordering has probably been done to invalidate the search pattern used by a particular anti-virus program. The difference is that sUMsDos contains the following instructions at location 10EH, but in A-204 their order is reversed:

```
mov cs:[004BH],ax        ; 2E A3 4B 00
mov al,es:[03FEH]        ; 26 A0 FE 03
```

Another variant is called **Spanish**, as it has only been reported in Spain. The signature string has been replaced with a JMP instruction to the virus code and the activation date has been changed to the 26th of every month. Some samples of the virus seem to contain a bit-error at byte 226H, where a byte has been changed from 50H to 70H. This error prevents the virus from working properly. The Spanish variant can be identified by the following pattern starting at byte 1C7H.

```
90 90 90 90 80 FA 1A
```

The **Anarkia** virus is also based on the sUMsDos variant. The sUMsDos signature has been changed to ANARKIA.

Other changes include a modification to the method used to determine whether the virus is already active in memory. The method used by both sURIV 3 and sUMsDos is to intercept INT 21H, function E0H which is not normally used, although the Novell network print spooling mechanism uses it. If these viruses are active, a value of 3 will be returned. Anarkia modifies this process slightly, using function E1H instead. There are other minor changes - the activation date is now any Tuesday the 13th, except in 1992. The delay loop has also been increased by 25 percent above that of sUMsDos. In addition to looking for the virus signature, the presence of Anarkia may be ascertained by examining byte 1C8H, which contains 2 (Tuesday) instead of 5 (Friday).

A second variant, **Anarkia-B**, is similar in every respect except that it uses INT 21H function E4H and activates on any October 12th.

Just like the first variant of Anarkia, the **Mendoza** variant from Argentina uses INT 21H, function E1H to check whether it is already active. The virus does not check the date before it activates but only the year - it will not delete files if the year is set to 1980 or 1989. When the virus is active, one in every ten files executed is deleted.The virus does not reinfect EXE files.To determine the presence of the Mendoza variant, the following pattern will be found at offset 1C1H:

```
CMP  CX,1989      ;81 F9 C5 07
JZ   02F7         ;74 30
CMP  CX,1980      ;81 F9 BC 07
JZ   02F7         ;74 2A
```

Of the variants which seem to have been produced by patching the original sUMsDos virus, one is remarkably different from the rest. When it activates, it will play the tune 'Frere Jacques' on the system speaker, instead of slowing down the computer or producing some of the other effects of the original virus.

Several Jerusalem variants have been reported which have never been made available to computer virus researchers. These include **Jerusalem D**, in which file deletion code has been replaced by a routine to destroy both copies of the FAT on 13th September 1991 and any Friday the 13th from then on. A related variant, **Jerusalem E** is reported to activate in August 1993. The **Century virus** also belongs in the 'reported only' section - it is said to activate on 1st January 2000, erasing all data on every drive.

(*The Jerusalem virus and its variants have been observed to replicate on Novell LANs. The most recent report from the United States was of the sUMsDos variant which was tested on a NetWare 2.15C configured LAN at Novell's Paramus, New Jersey, facility in June of this year. According to the evaluation team the virus infected the server from infected nodes and wrote to the server without write-privileges. These characteristics gave rise to initial reports of a Novell-specific virus but subsequent analysis showed that this was not the case. Ed.*)

LETTERS

Sir,

The August edition of *Virus Bulletin* (page 18) has the following paragraph:

"The effect of triggering (Datacrime IIB) is that the drive is no longer recognised by the machine's POST (Power On Self Test) routines and effectively "disappears" from the machine configuration whgen it is rebooted. Restoration of this drive signature is an involved process which will probably be beyond the technical capabilities of even the best customer support departments and might even entail the drive being returned to the manufacturer for reconfiguration".

In IBM's opinion the problem is not as acute as has been implied. We feel that the virus would not require the return of any disks from the IBM product line to manufacturing. Since the hardware related information could be restored using generally available tools (e.g. reference diskette) it would not be necessary.

There are three types of hard disk drive interfaces, namely ST506, ESDI and SCSI. The ST506 interface drives have no licensed internal code built into them and are entirely controlled by IBM BIOS. These are found on (some but not all models) PS/2 models 30, 30(386), 50, 55, 60 and 80-044 machines. The drive setup information is stored on cylinder 0, head 0, sector 2. Reference disks read this sector to find out the drive type information; without it they cannot automatically configure the machine. If damaged the disk would have to be restored by running the same program, and "manually" entering the drive tape, which would then put the information on that sector. To determine the drive type the users would have to look at the drive to see if it is type 32 or 33 for example. However, it is possible to recover without requiring special tools (other than those generally available) or special skills, other than experience.

To recover the user would have to:

1. Look at the disk by opening the machine to find out the drive type number.

2. Run diagnostics or reference disk.

3. Tell diagnostics or reference disk what the drive type is.

Within large organisations the information centre or help desk is familiar with this sort of activity, i.e. when adding and removing hardware from older machine models or when the drive set-up information has been lost.

For the older types of interface that have their own licensed internal code, the data is stored in an area of the disk which is not accessible to any user programs. The stored data includes

disk type, maximum number of heads, maximum number of cylinders, sectors per track, landing zone, and control information. This data can only be read and accessed by instructions within the licensed internal code of the disk drive. Therefore, there is no requirement for tools to change this data.

Now regarding the problem of viruses damaging the partition table or the master boot record. This will cause a user to lose access to the data allocated in the partitions of his/her disk. Recovery of the data is not exactly an activity for an average user. Skilled and experienced users with commercially available products, like Norton Utilities, could in time recover the lost data. The recovery of the data may need a customer support centre, dealer, or CE organisation to provide help. This will result in either on-site support or the user taking the machine/disk in to them.

To avoid this problem the user can save the master boot record. There are a number of products (PC Tools Deluxe for example) which save the first 9 sectors of the contents of the hard disk, and allow the user to restore them later. Users who are concerned about losing their data, or not having the data to hand at all times, could then run this utility each time they back-up or end their work session. Some boot sector viruses might be backed up and restored with this procedure. In particular, the Stoned (aka New Zealand) virus will be backed up and restored by this procedure, because Stoned stores itself in the first sector and the original master boot record in the 7th sector (of a hard disk). IBM recommends using a good virus detector before doing any back-ups.

To summarise, IBM would not anticipate anyone having to return a drive to the manufacturer because a "rogue program" or "virus" had damaged the setup information.

Yours faithfully,

Mark Drew
UK Information Protection Programme Manager
IBM United Kingdom Ltd.

Jim Bates replies: I am delighted to see that IBM, at least, are concerned enough to qualify my comments on recovery from the destructive potential of the Datacrime IIB virus and I am happy to stand corrected.

My observations concerning recovery were based on enquiries of technical support departments at several dealers and users. Replies indicated a general lack of information about what configuration information is stored where on modern fixed disk drives and how it may be restored in the event of corruption.

My point was not that data could *not* be recovered, but rather that recovery might be a complex and time-consuming business (with the attendant expense).

COUNTERMEASURES

Dr. Peter Lammer

Cryptographic Checksums

Imagine buying baby-food, when you know that there are psychotics at large contaminating it with poison. Last week they were just using ground glass, so we looked for that. Yesterday they were also using arsenic and mercury, so to be on the safe side we checked for those too. Will tomorrow's list include DDT and strychnine? You can see the point; *while one could try to test for each possible poison before feeding the offspring with his apricot puree, a reliable tamper-detection wrapper on the jar would be an awful lot more satisfactory.* Knowing that somebody has fiddled with the wrapper is enough - then it doesn't matter what has been done to the contents; you just don't use that jar.

In the face of slippery self-modifying self-encrypting computer viruses, it is becoming increasingly clear that the intuitive approach of searching for known nasties is not a good long-term route. The path to Hell, as Milton says, is paved with good intentions. By contrast, the mature technique for combating viruses is to divert our attention from the details of the threat and to focus instead on the integrity of the software upon which computers rely for their normal operation. *Rather than searching for an ever-growing, permanently incomplete list of things we hope not to find, we need to check that the software structure of our systems has not been meddled with.*

Every computer virus must at some point modify or add to some element of the overall set of executable code on a PC if it is to take effect and spread. The possible attack points include all items of executable code which are not physically prevented from being modified: the bootstrap sectors, the system files, all COM and EXE files, overlay files and so on.

If we can establish with certainty that none of these items have been modified in any way, we can give our system a clean bill of health - just like the confidence of looking at a jar of baby-food with the tamper-proof wrapper intact.

While it would in theory be possible to do this by comparing the executables with copies known to be correct, there are obvious practical limitations; for each megabyte of program to be authenticated, a megabyte of write-protected code would be needed for comparison.

The obvious solution is to use some form of **checksum** which depends upon every byte in the data to be authenticated. The size of a checksum is generally independent of the volume of data it authenticates, and thus does not pose any storage problems. There are three main classes of checksum in

common use: **simple checksums**, **cyclic redundancy checks** and **cryptographic checksums**.

A **simple checksum** is typically constructed as the arithmetic sum of all the bytes in the file to be authenticated, modulo some convenient number such as 65536 (which is 2-to-the-power-16). If any one of the bytes in the file is changed, then on recalculating the checksum you should find that the resulting value is different to the previously recorded one. The weaknesses of such a checksum for authentication are not hard to see. Firstly, there is no authentication of the order of the bytes in the file: the sum of a given set of integers is the same regardless of the order in which they are added up. Secondly, it is clearly easy to modify part of the file (eg. to cause a virus infection) and yet achieve the same overall checksum simply by compensating with extra values inserted wherever convenient. Thirdly, the mapping of checksum values is very poor from a statistical point of view when small volumes of data are involved, as the resulting checksum values are also small. **Simple checksums are worse than useless as a way of detecting viruses**.

Cyclic Redundancy Checks(CRCs) are considerably more secure than simple checksums. They are checksums based on the theory of maximum-length polynomials, and work by dividing blocks of data by a predetermined polynomial. They can be described either algebraically or in terms of feedback shift registers - the latter being the easiest way to visualise a CRC. However, checksums calculated from CRCs are still vulnerable to being forged or manipulated. Some authors of checksumming software use CRCs with a random choice of 'generator' (i.e. the underlying polynomial) to prevent one user from calculating another's checksums - but the number of suitable choices is still relatively small. CRCs are therefore unsatisfactory for serious work: a 'tamper-proof' wrapper which, with care, could still be removed and reapplied without anyone noticing.

The full-blooded approach is to use a **cryptographically secure checksum**. Cryptographic checksums, as the name implies, are derived from the art of encrypting or scrambling confidential information. They can be based on either bit-manipulation or arithmetical methods and use sophisticated techniques to achieve the required properties. Given a series of bytes, such as the contents of an executable, it must be relatively easy to calculate the checksum - whereas given a particular checksum value, it must be extremely difficult to choose a series of bytes that match it. For this reason cryptographic checksums are referred to as 'one-way functions'.

A cryptographic checksum typically has a 32-bit value, i.e. a number between zero and 4,294,967,295 (inclusive). This means that it gives a virus a chance of less than one in four billion of escaping undetected on a single infection. The chance of escaping detection in two different executables would be less than one in sixteen billion billion. One of the properties of any good cryptographic checksum algorithm is that even a change to just a single bit in the 'input' bytes will lead to a completely different value of the checksum. The algorithms typically work by 'munching' their way through the stream of input bytes, one or eight bytes at a time, such that at each munch the interim value of the checksum depends in a complex and irreversible way on the values of all input bytes processed so far.

Cryptographic checksums algorithms generally allow 'seeding' of the calculation with an individually chosen value, rather like a password. In cryptographic jargon this value is an I.V. or Initialisation Vector. Good anti-virus software using these techniques often incorporates some form of password entered by the user, which is then used to form an I.V. providing additional security in the authentication process.

The two best known cryptographic checksum algorithms both come from the banking world, where they have been used for many years to check that money transfer orders are not manipulated or corrupted, for example through account numbers being modified. One is an arithmetical method known by the prepossessing epithet of 'ISO 8731-2', whereas the other, based on the DES encryption algorithm, is generally referred to as 'ANSI X9.9'. International standards do not always have instantly memorable names. However the great thing these algorithms have to offer is that they provide a tried and tested wrapper for our jar of executable code, impossible for any hacker, virus-writer or other unwanted modifier to bypass undetected. **One cannot over-emphasize the importance of using a cryptographically sound checksumming method**. The potential pitfalls of incompetent amateur attempts to design cryptographic checksumming algorithms were highlighted by the product review in *VB July 1989, pp. 13-14*, where the reviewer found the checksumming method lamentably easy to reverse.

Anti-virus software based on cryptographic checksums can still be vulnerable to attack if it is not designed with due care and skill. To ensure that such software is not circumvented, it is essential that it is run on a PC which has been bootstrapped from a write-protected, clean system floppy disk. Clearly, it is also essential to start from the correct basis; when the anti-tamper wrapper is applied to a jar one needs to be certain that there is not already poison in the puree. In the same way, the cleanliness of a PC must be assured before using an anti-virus package to calculate the checksum values which will be used as yardsticks of integrity in subsequent checks. This is usually achieved either by formatting the machine from fresh, or by performing an initial scan with a virus-specific detection program.

If future viruses continue the current trend of increasing the difficulty of virus-specific detection, then formatting a machine and rebuilding its software from good master disks will be the most reliable way of ensuring the essential initial integrity.

OVERVIEW

Jim Bates

From Brain to Whale - The Story So Far

Since computer viruses first made their appearance on IBM and compatible personal computers, several distinct phases of development have become apparent. The implication that certain dedicated individuals (or even companies/agencies) dotted around the globe are actively developing more and more complex virus code is inescapable.

Stage 1. Primitive Viruses

The early viruses were quite untidy in their internal construction and though they worked (in the main) as intended, disassembling them was a fairly simple task. The initial categorisation into Boot Sector and Parasitic types proved useful and there was either no trigger at all, or the trigger results were "benign". These early examples appeared to be written as irresponsible jokes by immature individuals who had a little skill but no real experience in programming. The impression was of a first year computer student flexing his or her intellectual muscles with new found knowledge. The implications of "joke" programs which took control of system resources in an undisciplined manner seemed to have escaped the virus writers and at least one example (the New Zealand Virus) was capable of destroying data as a result of the inexperience of the programmer. **Bugs like this still occur and are either a blessing or a curse depending upon your point of view**. The latest to be highlighted was the bug in the Disk Killer virus which effectively prevented the creation of "live" virus code on the 56,500 disks distributed by *PC Today* magazine (*VB, August 1990, p.3*).

Within a fairly short space of time, the individuals who had been writing Trojan code latched onto the possibilities of making their offspring mobile by attaching them to virus code, and the first deliberately destructive viruses began to appear. Internally, these were still extremely primitive and were mainly concerned with self-replication and triggering. Anti-virus programs were beginning to appear and they found the detection and disinfection of virus code a relatively easy matter. It is possible to write a simple pattern recognition scan program in around an hour with a reasonably powerful high-level language, as long as the operating system can be guaranteed clear of interference.

Stage 2. Encryption

New viruses were still appearing at an increasing rate but a subtle change was noticed in the emphasis of the authors as the first self-encrypting viruses made their entrance. Anti-virus software relied mainly upon searching files for recognisable segments of code which were known to belong to particular viruses. The virus writers answered this by building in pseudo-random encryption code in such a way that each new copy of the virus that was generated consisted of a different (and thereby unrecognisable) sequence of coded bytes. The first self-encrypting viruses were not very effective and since they had to have decryption routines in the early part of the code, these could be recognised by the anti-virus scanning routines. **Later viruses of this type were a little more difficult to detect as they began to use a random selection of encryption techniques such that various copies now had not only different encryption "keys" but also different encryption "locks"**.

Also about this time, the first of the "professional" viruses appeared. The differences between the early amateur efforts and these more accomplished programs are not easy to describe accurately but they are nevertheless immediately obvious once the code has been dissected down to assembler level. The first "professional" virus was the Cascade (1701/1704) virus. This had a very sophisticated trigger routine that made individual letters on a text screen appear to "cascade" down to a heap on the bottom line of the screen. The technical standard of the trigger routine is not normally a very reliable guide to the skill of the programmer since many of them are stolen from known humorous or demonstration programs. However, in the case of the Cascade virus, the standard of coding was maintained throughout the program and the hand of experience was obvious. The Cascade virus is particularly interesting since the original version was deliberately limited in both its trigger period and the range of machines which could be infected. Unfortunately, the infection routine was left out of the limitation period so that even after the virus was inactive it was still spawning replicas. My own theory on the Cascade is that it was developed "in house" by some software organisation for purposes best known to themselves. The original irresponsibility of this decision was then compounded by poor security and the virus was somehow released into the computing community at large. The result is history and other versions of this virus now exist which have had much more dangerous triggers added. **This process of modifying the trigger routines within existing viruses has become an ongoing problem which only serves to complicate the process of keeping track of differing virus types**. In most cases, such modifications do not substantially change the recognition characteristics of the basic virus type but from the point of view of anti-virus software, particularly scanning programs, the more signatures there are to search for, the more cumbersome the search becomes.

The situation was further complicated by the appearance of viruses which introduced 'garbage' bytes at random into the decryption sequence. The first virus to use self-modifying encryption, 1260, where no segments of the virus code remained static, necessitated the adaption of scanning program to search for a recognisable 'identity' as opposed to a standard

hexadecimal pattern. 1260, Casper and other viruses which employ this tactic are sometimes referred to as 'chameleon' viruses.

Stage 3. Stealth

Having complicated the issue of scanning detection with encryption, the virus writers then turned their attentions to confusing the more general anti-virus software detection capabilities. The most sophisticated example of this development was found within the 666 (Number of the Beast) virus. One of the major problems facing virus writers is where to store their code so that it is safe from being overwritten by DOS and yet is not obvious to a cursory examination of normal file directory entries. This was overcome within 666 by using redundant space within the DOS file structure in such a way that no extra space was taken up on the disk and nothing was added to the file length. This meant that not all files were suitable for infection and the potency of the virus was thereby somewhat reduced. Another difficulty was how to prevent the code itself (as stored on the disk) from being examined and recognised. 666 tackled this by installing its own DOS interrupt interception routine which "watched" for particular types of file access. If attempts were detected to read affected parts of infected files, the original program code was substituted and the searching routine was returned satisfied that the file was clean. **Viruses which adopt such active measures to avoid detection have now been christened "Stealth" viruses and they present an extra problem to both resident and non-resident anti-virus software** True, this problem can be overcome by ensuring that the target system has been booted from a clean write-protected system floppy disk but this is considered inconvenient and impracticable for most commercial users. Incidentally, the first virus to adopt such defensive measures was Brain. This substituted the absolute disk address of the uninfected copy of the boot sector whenever a BIOS request for this sector was issued.

The advent of stealth viruses forced a serious rethink amongst vendors of anti-virus software. Those marketing scanning and "fingerprint checking" programs either had to build "awareness" of stealth techniques into their products, upgrading them as new tricks were found, or they needed to increase the size of the typeface used for the warning to "Reboot Your Machine From A Clean System Disk". Many resident, monitoring-type anti-virus programs also fell foul of another technique first noticed in the 666 virus. Virus code must have unimpeded access to the disk in order that its infection can spread. Resident anti-virus programs were designed to prevent (or at least detect) such access and usually did so by detecting whenever an executable program file was opened for Write access. It is unusual for normal commercial systems to write to program files (except during copying) and this became a useful technique for detecting potential virus activity within a system. 666 got around this detection by first opening the target file for Read Only permission (thus avoiding the alarm bells) and then going through a "back door" to change that permission to Read/Write. **This obviously ruffled quite a few feathers in the anti-virus industry but worse was to come when the Flip virus was found to contain a technique which quite simply by-passed monitoring software by "stripping" back the chain of interrupt handling routines until access was available "underneath" any monitoring handlers** This was a serious problem for vendors of anti-virus programs which claimed to detect and/or prevent virus infection activity. Paradoxically however, this stripping technique has proved a boon for scanning and fingerprinting software since just as it can disable monitoring handlers, it can also disable the very same stealth techniques which some viruses use to avoid detection. The Flip virus also added a new category by providing both Boot Sector and Parasitic infection routines within it, thus becoming one of the first "multi-partite" viruses. The problem of where to store the balance of the Boot Sector code was also given a new twist when it was discovered that Flip "made" space available by reducing the size of the DOS partition and storing its code "beyond the end" of the disk!

Both encryption and stealth techniques can be classified as disinformation exercises. Each is designed to confuse and/or misinform the routines which are used within anti-virus software to detect signs of virus activity. **Since the activities of virus researchers and disassemblers were obviously keeping pace with new virus techniques, the next logical step was to extend the confusion/disinformation techniques to the researchers themselves by writing 'armoured' viruses that are deliberately intended to be difficult to dissect**.

"We are no longer dealing with amateurs, these latest viruses are written by professionals with long experience in programming. The question as always, is, why are they writing such code?"

Stage 4. Armour

This 'armoured' phase has now arrived with the Fish6 and Whale viruses. Unfortunately for the virus writers, computer programs always consist of a series of definite instructions which are presented to the micro-processor in an inviolable predetermined sequence. Thus a little experience and a lot of patience are all that is necessary to examine each step in the sequence and eventually produce an accurate analysis of

exactly what the program is capable of. Of course, powerful analytical software debuggers are an added benefit but do not change the essential requirement. Within the last year, one head of a company supplying anti-virus software in the UK made the statement that viruses might soon be encrypted using some irreversible or immensely difficult process which would make a virus virtually impossible to disassemble. This fatuous statement highlighted only the technical inadequacies of the individual who made it. For any virus to function, regardless of encryption techniques, it **must** be able to decrypt itself before processing. Thus actually cracking the encryption routine becomes a simple academic exercise before the work of analysing what the virus actually does. **Although virus encryption techniques will easily succumb in this way, the introduction of deliberate confusion code does mean that disassembly and analysis is slowed down**

While Brain was the first virus to use stealth techniques, 666 and Flip are considered the most sophisticated stealth viruses so far. Similarly, although other viruses (notably FISH6 and Flip) have used "confusion" coding, by far the most sophisticated of this type is the Whale or Motherfish virus. Little is known of this yet although a detailed report is currently in preparation for next month's *Virus Bulletin*. **What is apparent is that this is the work of a professional programmer (or group) using all the tricks at his disposal to make disassembly as difficult and complex as possible** Analysis so far has revealed routines which directly access the Programmable Interrupt Controller chip, specifically control stack growth, loop through unclosed interrupt service routines via manipulation of Trap and Interrupt flags and several other, as yet unclassified, routines - all of which are obviously designed to confuse disassembly. There are also extended sections of self-modifying, self-correcting and decrypt/recrypt code. We are no longer dealing with immature amateurs; these latest viruses are written by professionals with long experience in programming for the PC environment. The question as always, is, "why are they writing such code?".

Fighting Back

The fight against computer viruses has now become a war of attrition. Genuine virus researchers have a small advantage in that they generally have copies of most of the viruses at large today and can therefore collate all the various tricks that they use for the benefit of anti-virus software construction. It is to be hoped that few of the virus writers will have such access and are substantially on their own when trying out new ideas. However, the "pool" of existing virus code can only grow since every virus that has been let loose into the computing community will continue to exist in an active form somewhere. This is why even those who claim to create viruses for "research" purposes must be severely censured. We have now reached the stage where professional expertise is being brought to bear in producing virus code and it is time for the creation of an officially sponsored organisation which can provide similar professional expertise to fight back. There are currently only a handful of genuinely capable researchers world-wide, most of whom work on a self-funding basis. Some of these become involved in marketing anti-virus software but the immediate conflict of interests causes immense problems in such circumstances.

An industry or government funded organisation is desperately needed to provide independent, professional monitoring of both virus and anti-virus programs in such a way that the extracted information (in the form of reports, reviews etc.) can be made available globally to all interested parties within the computer industry. There are already reports of moves towards the establishment of such centres in various countries including the USA and Australia. Until such a centre is properly established in the UK we shall continue to be at the mercy of any organisation that chooses to spend money on developing this evil trade.

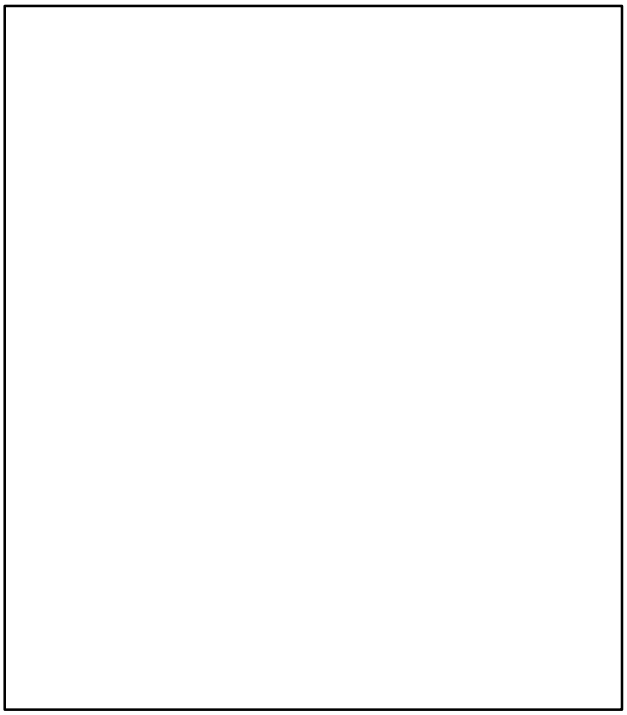
```
If you have the motherfish, you are entitled to
an explanation...when we discovered the
motherfish, the decision was made to disavow its
existence and any public comment on it was
prohibited... the file was never made available
through normal distribution based on two
findings 1. the virus can not be detected by
present methods 2. the virus is modularly
constructed to allow it to "learn" the methods
used to detect it, and then integrate this coded
thought into its arsenal of defensive
mechanisms.........the motherfish is not just a
virus, it is a virtual living, breathing entity
that is capable of teaching itself its pursuers
techniques and then increasing its code level
sophistication as its environment becomes
increasingly hostile...this characteristic made
it imperative that distribution be kept at an
absolute minimum... it would be appreciated if
you kept that in mind.
```

Red herring? This message about Motherfish (Whale) was posted anonymously to the VIRUS ECHO conference on Fidonet. Initial reports indicate that eighty percent of the virus' code is designed to prevent disassembly. The assertions made by the author(s) of this message appear fanciful and exaggerated. However, the appearance of this virus ***is*** causing concern among software developers due to the increased effort and time needed to disassemble the virus and identify search characteristics. No scanning program is yet capable of finding the virus in a consistent and reliable way. Finally, note the reference to "normal distribution" which may be a reference to the legitimate research community or, more insidiously, a reference to some underground virus distribution network.

PRODUCT REVIEW 1

Yisrael Radai

Virus Bulletin has not received permission to reproduce this article on CD from the author. Readers can obtain a paper copy of the original issue directly from *VB*.



PRODUCT REVIEW 2

Dr. Keith Jackson

Virex-PC

Virex-PC is a software package which claims to provide "virus prevention, detection and treatment".

Documentation

The manual provided with Virex-PC is 63 pages long, in A5 format, and contains ten chapters and four appendices. There is a thorough table of contents, but no index. The legal disclaimer (their word) at the front of the manual is of the 'We guarantee precisely nothing' kind; it states that the vendor "... specifically disclaims any implied warranties of merchantability or fitness for any particular purpose". The last thing most purchasers of software want when a problem occurs, is for the vendor to turn round and say that nothing will be done, because the product guaranteed nothing. This type of lawyers' weasel-wording does no-one any credit, and such sweeping disclaimers deserve only contempt.

The manual is well written and provides a reasonable explanation of how to use Virex-PC, what viruses are, and what effects they can have. Even though part of the Virex-PC software must be memory-resident to operate, the manual does not discuss the problems of interaction between various memory-resident programs. This is a critical omission, which will rightly make users who have many different TSR (Terminate and Stay Resident) programs suspicious. Clashes between memory-resident programs are an endless source of trouble, and should be discussed in the manual.

Virex-PC is provided on both 3.5 inch and 5.25 inch floppy disks. The disks came in a sealed envelope, but the 5.25 inch disk came without a protective sleeve. Given that the manual states very clearly that only a single backup copy of the software can be made, this is an oversight which needs correcting.

Scanning Speeds

The rate at which the Virex-PC scanning software can search a disk depends mainly on what mode of execution is used. By default only COM files, EXE files and overlay files are searched and these files are only checked in the areas in which a virus infection is most likely to be found. Virex-PC has an option to search entire files. This takes longer, and is recommended only for the first time that a file is checked.

The time taken by Virex-PC to check a hard disk was tested on a Toshiba 3100SX portable (*see Technical Details below*).

When used in the default (fast) mode, Virex-PC took 1 minute 31 seconds to search through 261 executable files (9.4 Mbytes). When searching each file completely, this rose to 4 minutes 31 seconds. When all files were searched (1052 files, 20.1 Mbytes) the search time rose to 11 minutes and 18 seconds. For the purposes of comparison, SCAN from *McAfee Associates* (*see VB, September 90*) took 3 minutes and 21 seconds, and SWEEP from *Sophos* took 2 minutes and 55 seconds, to test the same hard disk. Both of the latter search all parts of executable files. Therefore Virex-PC's inherent searching speed is not as great as either SCAN or SWEEP, but this is compensated for by being able to search only certain parts of each file being tested. When this is done, Virex-PC is approximately twice as fast as SCAN or SWEEP. I discussed this point in a recent review (*VB, August 90*), and it should be noted that such speed increases are at the expense of making the search process very virus-specific. **All viruses to be scanned for in this manner have to be completely understood, rather than just searched for using a search pattern.** *(The above timings for Virex-PC are disputed by the developer. See pp 4-5 on the implications of multiple infections. Ed.*)

Memory-Residence

The Virex-PC memory-resident software provides a warning when a program attempts to **format a disk**, **write directly to a disk**, or **install itself in memory**. Warnings are similarly provided if an attempt is made to execute either a program which has not previously been registered with Virex-PC, or a program whose checksum has been altered. Such warnings are an attempt to constrain the effect(s) that a virus (or any other malicious program) may have. The manual discusses in detail the action that should be taken when such a warning is encountered, and makes it clear that although some types of program may have need of such facilities, other types of program do not. In the end only the user can decide whether a particular action is sensible, or may lead to trouble.

The installation program for the Virex-PC memory-resident software does not exactly match the description in the manual. Extra features have been added, such as displaying the contents of the current README file before installation proceeds. Although such developments are laudable, it casts doubt on how up to date the manual is. The memory-resident software has a multitude of tailoring options that can be activated when the program is installed. For instance the types of file that are write-protected by Virex-PC must be named (using multiple overlapping wild-card file specifications), any exceptions to such protection must be named, a choice must be made whether or not the master boot sector should be checksummed, and the files that are allowed to execute without Virex-PC warning the user of their imminent execution must be stated. Despite this complexity, the installation process is fairly painless, and the user is led through each stage very simply. The dangers of making various choices are pointed out in both the manual and the on-line help facility. Space prohibits a detailed explanation of all the available

options. One infuriating feature of the installation program was that if a mistake occurred during the question and answer session, and the Escape key was pressed, then the installation program queried whether it should terminate. When the response 'No' was entered, the installation process started all over again. Not very friendly.

One of the final choices made is to set Virex-PC for one of four security levels, ranging from a user with very basic skills, up to the system administrator. The warnings received and the facilities available, vary with the security level chosen.

Using a PC with the Virex-PC monitoring software installed is an intriguing experience. Virex-PC detects all that it claims to, but this causes warning messages to pop up very frequently. So frequently that it drove me to distraction within just a few minutes. I know that the system can be tailored to allow certain programs to execute uninterrupted. My problem was that with the type of work that I usually do, the set of 'allowable' programs is necessarily changing on an almost daily basis. However I'm probably an atypical user, and I can well see how the protection offered by the memory-resident parts of Virex-PC are of use in an environment where PCs are used for fixed purposes. One other thing to beware of is that whoever sets up Virex-PC must spend some considerable time tailoring it for individual users.

In short, I have few complaints about how the Virex-PC memory-resident software operates, but except in environments where programs are used in a rigid manner, I have some doubts about the general philosophy behind memory-resident software monitoring program execution and attempting to second guess what is *legitimate* program activity and what is not. I concede that fixed PC environments are probably in the majority.

Detection

The complete list of viruses known to Virex-PC is contained in an appendix to the manual. This lists 48 viruses, of which 21 permit a file to be 'repaired' when the virus is removed. The list of known viruses is extended to 57 when those currently known to the executable file are displayed on screen. This is an encouraging indication that the program is actively updated to keep pace with the appearance of new viruses.

When the Virex-PC scanning program was checked against the *VB* test set of 49 viruses (*see Technical Details below*), it successfully detected all types except for Hallochen, 1260, Taiwan and VP. From the 101 virus variants tested, Virex-PC detected a virus of some type in 77 cases. Some viruses were wrongly detected; Anarkia was detected as the Jerusalem virus; Dark Avenger and Eddie II were both wrongly detected as the Murphy virus; and the Yankee virus was wrongly detected as the Fu Manchu virus. Surprising anomalies have been observed in most other scanning programs which have been submitted for review. Overall, the above results are comparable with other scanning software reviewed by *VB*.

Conclusion

In conclusion, Virex-PC detected all of the commonly found viruses, but did not have a very long list of viruses for which it tested and some of the virus variants were not detected. The Virex-PC scanning software can run in various modes ranging from a very fast search, to a thorough search of every byte of each file. The Virex-PC memory-resident software is efficient in what it claims to do, but obtrusive unless the PC is to be used for specific purposes.

Technical Details

Product: Virex-PC

Vendor: Microcom Software Division, PO Box 51816, Durham, NC 27717, USA. Tel 919 490 1277, Fax 919 490 6672.

European Office: Microcom Software Division Ltd., 2D Dukes Court, Duke Street, Woking, Surrey GU21 5BH, England, Tel 0483 740763, Fax 0483 740764.

Copyright Owners: Software Concepts Design, Microcom Systems Inc., and Softsync Inc.

Availability: IBM PC/XT/AT, PS/2, or 100% compatibles running MS- DOS v2.11 or above. A hard disk is required. Virex-PC requires 30K of memory, and 512K is recommended.

Version Evaluated: 1.10

Serial number: None visible

Price: £65.00 or £140.00 to include a one-year update service.

Hardware used: An Amstrad PPC640 with a V30 processor, and two 3.5 inch (720K) floppy disk drives, running under MS-DOS v3.30. Also a Toshiba 3100SX laptop portable with a 16MHz 80386SX processor, one 3.5 inch (1.44M) floppy disk drive, and a 40Mbyte hard disk, running under MS-DOS v4.01.

Virus Test Suite: This set of 49 unique viruses (according to the virus naming convention employed by *VB*), spread across 101 individual virus samples, is the standard *VB* test set. It comprises two boot viruses (Brain and Italian), and 99 parasitic viruses. There is more than one example of many of the viruses, ranging up to 10 different variants in the case of the Cascade and Vienna viruses. The actual viruses used for testing are listed below. Where more than one variant of a virus is available, the number of examples of each virus is shown in brackets. For a complete explanation of each virus, and the nomenclature used, please refer to the list of PC viruses published regularly in *Virus Bulletin*:

1260, 405(2), 4K(2), AIDS, Alabama, Amstrad(2), Anarkia, Brain, Cascade(10), Dark Avenger(2), Datacrime(3), dBASE, December 24th, Devils Dance, Eddie(2), Fu Manchu(3), GhostBalls, Hallochen, Icelandic(2), Italian, Jerusalem(6), Kennedy, Lehigh, Macho-Soft, MIX1(2), Number of the Beast, Oropax, Perfume, Prudents, PSQR, South African(2), Suriv(8), Sylvia, Syslock(2), Taiwan, Traceback(4), Typo, Vacsina, Valert, Vcomm, Vienna(10), Virdem, Virus-90, Virus-B(2), VP, W13(2), XA-1, Yankee(5), Zero Bug.

END-NOTES & NEWS

A *Virus Bulletin* **conference** will take place on 12-13 September 1991. The objectives of the conference are 1) *to present factual information about computer viruses*, 2) *to demonstrate defensive procedures*, 3) *to discuss probable future virus developments and countermeasures* and 4) *to attempt to harmonise research efforts*. Information about the conference venue, speakers and presentation subjects is available from Petra Duffield, *Virus Bulletin Conference*, UK. Tel 0235 531889.

VB has just received a copy of *McAfee Associates'* PROSCAN and initial tests indicate that this scanner **is** capable of detecting the Whale virus (*see pp. 2, 11-12*). However, *Bates Associates' VIS* reports that PROSCAN produces a false positive identification in DesqView's NOTE.COM file.

Microcom (developers of the Virex and Virex-PC packages) has acquired *1st Aid Development, Inc.*, the developer and vendor of a family of Macintosh recovery products including 1st Aid Kit (file and disk recovery), Complete Undelete (file undeletion), and Sector Collector (prevents the use of bad hard disk sectors). Tel (UK) 0483 740763. Tel (USA) 919 490 1277.

The Institute of Electrical Engineers is holding a one day conference "**Viruses and their impact on future computing systems**" (London, 19 October 1990). Colloquia Bookings, IEE, PO Box 96, Stevenage, Herts SG1 2SD.

S & S Consulting Group, UK, is holding a two day seminar, **the Virus Threat**, Great Missenden, 8-9 November 1990. Tel 0494 791900.

A one day **briefing on computer viruses** presented by Dr. Fred Cohen will take place in London on 13 November 1990. Details from *IBC Technical Services*, UK. Tel 071 236 4080.

Computer Viruses: A Threat for the 1990s (report IS09-250-101) is a new report published by *Datapro Research*. The report discusses different types of malicious programs, virus characteristics, risk assessment and prevention, detection and recovery from virus attacks. *Datapro Research*, Delran, NJ 08075, USA.

A new book, **Rogue Programs: Viruses, Worms and Trojan Horses**, is published this month by *Van Nostrand Reinhold*. The book is a collection of twenty seven articles and excerpts about 'vandalware' (ISBN 0-442-00454-0). Publisher's list price is $32.95 in paperback.

The first **conference on computer viruses** to be held in the Soviet Union takes place in Kiev on 12-14 November 1990. Details from Nikolaj Nikolaevich Bezrukov, Kiev, USSR. Tel +7 044 2681026.

VIRUS BULLETIN

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

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

Editorial enquiries, subscription enquiries, orders and payments:

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

Tel (0235) 555139, International Tel (+44) 235 555139
Fax (0235) 559935, International Fax (+44) 235 559935

US subscriptions only:

June Jordan, Virus Bulletin, 590 Danbury Road, Ridgefield, CT 06877, USA
Tel 203 431 8720, Fax 203 431 8165