December 1989 ISSN 0956-9979

# VIRUS BULLETIN

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



## CONTENTS

# EDITORIAL

## Towards a Common Language

Currently there is no agreed terminology for defining different types of computer virus, and no standard nomenclature to identify specific viruses. People involved in research or anti-virus product development can quickly identify a known computer virus from a clear description. However, the situation is creating difficulties for computing staff responsible for data integrity. Questions are being asked; is, for example, the 1701 virus the same as the Cascade virus, or is Cascade the same as Fall or Hailstorm or 1704? Are all these different viruses or variants? Standardisation would prove a major advance in the battle against the virus writers but there are a number of reasons why this seems unlikely.

A new virus is likely to be named or coded or classified by the first person to examine it. Since there is no consensus within the anti-virus community it is probable that other names, codes or classifications will arise. Some people believe that the use of names glamorises computer viruses. To name a virus from a message contained within the program is to do as the virus-writer intended. Effectively, it is playing his 'game'. Names such as 'Datacrime' or 'Dark Avenger' also promote a sinister aura which plays into the hands of the media. This group has promoted the use of the virus' infective length as a means to its identification. In this way Datacrime is known as 1168 or 1280, Jerusalem is called 1813 and Traceback becomes 3066 and so on.

Others say that names provide the quickest means of identification. If a virus says 'Datacrime' on screen it has gone some way to naming itself and people will describe such a symptom before ascertaining an infective length. Names, they say, are easier to remember and avoid the need for a complex catalogue. An argument against using the infective length for identification is that two different viruses might share a common infective length. However, the use of names can be just as confusing. Is the Spanish computer virus the same as the Lisbon virus? MIX-1, Swap, Typo and Mistake have names which imply certain similarities. Only by analysis can we distinguish between them or prove their commonality.

There seems to be a movement towards the use of names rather than numbers. Terminology is also causing confusion. 'File', parasitic' and 'shell' are three common expressions which are used to describe viruses which append themselves to programs, while 'overwriting' and embedded' are two expressions for viruses which alter code within an executable module. Failure clearly to identify separate viruses is compounded by a disparate vocabulary to describe them. The ideal position would be for the anti-virus community to speak the same language but this would only be possible once a universally accepted glossary of terms had been established. Any suggestions to speed this process are most welcome.

## Threats: Real or Imagined?

There are two distinct categories of computer virus - 'lab' viruses and 'wild' viruses. The former is an experimental program written in a controlled environment for research purposes, the latter, a genuine threat which has struck unsuspecting users. The dBASE virus which is analysed in this month's edition (pages 10 -12) is said to have been a 'wild' virus which was captured and confined before it could spread. It was first demonstrated on 26th October at a UK seminar entitled *Computer Viruses: Combat & Cure* by Ross Greenberg of the United States. No concise, validated account of the circumstances surrounding its discovery exists and until the situation is clarified the virus must be treated as a potential menace. Upon examination, this virus displays numerous anomalies to earlier reports and published descriptions of it.

## The Elusive Disk-Killer

Neither a copy of, nor technical details about, the Disk-Killer computer virus have been made available to *Virus Bulletin*. The only new information which may be of interest is that Dr. Alan Solomon of S & S Enterprises claims to have encountered this virus after if destroyed a PC's hard disk in Ealing, London, UK, in October of this year. The virus is said to trigger after an infected PC has been running for more than 47 hours. It appears in the *Reported Only* section of the table of *Known IBM PC Viruses*.

## Welcome

We welcome Dr. Fred Cohen and Mr. Yisrael Radai to the editorial advisory board.

Dr. Fred Cohen, whose thesis "Computer Viruses: Theory and Experiments" is widely accredited as the scientific basis for computer viruses, is acknowledged for his pioneering work in this field including the first in-depth mathematical analysis of virus propagation and the development of protection mechanisms. Dr. Cohen has held professorships at both Lehigh University and the University of Cincinnati in the United States. He is the author of the acclaimed ASP anti-virus system.

Yisrael Radai received his M.Sc. in Computer Science from the Hebrew University of Jerusalem, Israel, in 1975. He became active in the fight against computer viruses at the time of the Israeli PC virus outbreak in Jerusalem in January 1988. He has written many articles on the subject and is currently undertaking research into reliable checksumming methods for defence against viral infection.

# KNOWN IBM PC VIRUSES

The following is a list of the known viruses affecting IMB PCs and compatibles, including XTs, ATs and PS/2s. The first part of the list gives aliases and brief descriptions of viruses which have been seen, while the second part lists viruses which have been reported.

Each entry consists of the virus group name, its aliases and the virus type (See "Types codes" table). This is followed by a short description (if available) and a 10 to 16 byte hexadecimal pattern which 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 (See *VB Nov 89*). Offset normally means the number of bytes from the virus entry point. For parasitic viruses, the infective length (the amount by which the length of an infected file has increased) is also given.

# MACINTOSH VIRUSES

***David Ferbrache***

## Swiss nVIR B Clone Detected

The Universities of Basel and Zurich have reported a new clone of the common nVIR Macintosh virus. This clone has been named **Jude** after the characteristic four character resource names used for the nVIR resources.

Initial reports of resource code sizes and numbering indicate that this is an nVIR B clone. To date there have been 5 clones of nVIR B namely: Anti,Hpat (not a simple clone - code resource renumbered), MEV#, nFLU and now Jude.

A new version of Disinfectant (1.3) was released on November 29th to deal with the virus. John Norstad (the author) is actively seeking to extend the functionality of this program to perform generic clone detection (such as VirusRx performs) and will be incorporating this into version 2.0.

Users of Virus detective need not alter their detection strings to recognise this nVIR B clone.

It can be expected that commercial anti-virus software will be upgraded shortly as part of most companies' continuing commitment to monitor new virus developments. Intercept utilities such as Gatekeeper, Vaccine and the SAM intercept will of course detect Jude's attempts to replicate (although the SAM intercept will not if operating in basic protection mode).

## INIT 29 - Infectious, But Your Data is Safe

The INIT 29 virus was discovered in late 1988, and represents one of the most infectious Mac viruses known to date. The virus itself is, as the name suggests, a short (712 bytes) block of code inserted as an INTI (initialisation code) resource into system and data files. An important point is that such infected data files are not contagious, and represent dormant viral code. It would, however, be possible for a rogue user to activate such code by moving it from a data file into the system file.

### Inside an Infected Application

When an infected application is run on a Mac, the INIT 29 viral code is executed prior to the host's code. Application programs on the Mac comprises two areas, the data and the resource fork. The resource fork contains a number of executable CODE resources. One of the CODE resources is the CODE 0 resource which contains details of the entry points to all other CODE resources in the application and acts as a jump table (*see Fig. 1*).

```
CODE   0   [XXXX]---->CODE 1   Code
Jump       [XXXX]---->CODE 5   Resources
Table      [XXXX]---->CODE 9
           [XXXX]---->CODE 2
```

*Fig. 1*

When a program is invoked the segment of code in the first code resource (here CODE 1) is loaded and called via the jump table pointer. The segment can make local subroutine calls and jumps within its own code without reference to the CODE 0 jump table, however it may also invoke code in other segments by indirection through the jump table.

When an INIT 29 virus infects a code file it modifies the resource fork to add its code by creating a new CODE resource (using the lowest unused CODE resource number). It also modifies the first jump table entry (storing the original) to point to the viral code segment. Thus the table in Fig.1 would change to that in *Fig. 2*.

```
[XXXX]---           CODE 1   Code <---------
[XXXX]---|------->CODE 5   Resources         |
[XXXX]---|------->CODE 9                     |
[XXXX]---|------->CODE 2                     |
          -------->NEW CODE 3 VIRAL CODE    |
                   [SAVED CODE 1 JUMP]----
```

*Fig. 2*

### From Application to System File

The virus copies itself into the system file whenever an infected application is invoked. The copy operation adds the virus as a block of system initialisation code (INIT 29). When the infected machine is next rebooted the virus will install itself in main memory.

Infected system files will show timestamp changes and will contain the characteristic 712 bytes INIT 29 resource.

### Capturing Control

The virus INIT code patches the resource manager trap OpenResFile using the trap manager to update the table

of system trap pointers. All operation system calls on the Mac are made by indirection through this table providing a facility comparable, but with finer control possible, to the IBM PC interrupt vector table.

The call that the virus patches is important, as not only is it invoked when an application is launched, but also when any form of resource fork is opened such as occurs when the desktop is scanned on insertion of a new floppy disk.

Whenever the trap is invoked the normal operating system code is invoked, followed by the viral code, as shown in *Fig.3.*

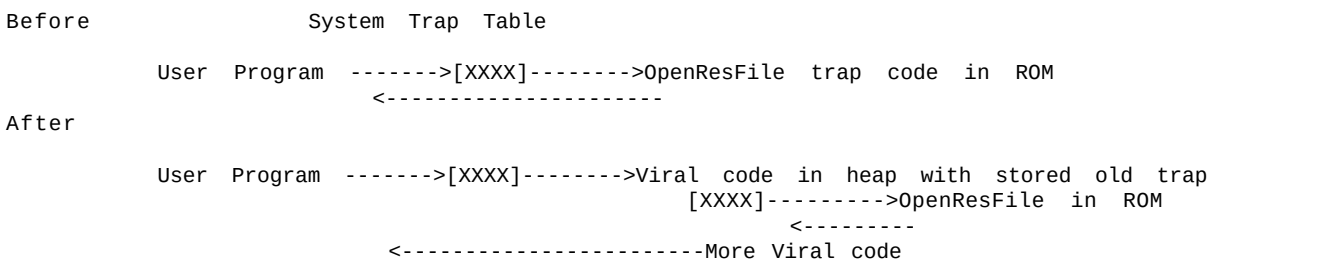


*Fig. 3*

The virus code causes the application file or data file which is the subject of the OpenResFile call to be infected. The form of the infection is:

```
If No Code Resources
     add the virus as INIT 29
    overwriting any existing INIT 29 resources
Else
    add a new CODE resource using the lowest unused
     resource number
    link it via the jump table to be executed first
```

**Symptoms of Infection**

**Odd resources**

*Look for 712 bytes INIT 29s in the system file

*Look for changes in modification times on application files

*Look for CODE resources of size 712 bytes

**Desktop errors**

A major symptom of INIT 29 is the error message produced when a locked (read only) disk is inserted. The virus will attempt to add an INIT 29 to the desktop file causing the message:

```
"the disk needs minor repairs"
"do you wish to repair it?"
```

If the response is affirmative then the Mac will eject the diskett with a message that the desktop could not be rebuilt.

**Time Critical Software**

Software which relies on the speed of response of interrupt service routines which use the OpenResFile trap may fail. Reports include failure of Laserwriter printing and network communications in the presence on INIT 29.

**Disinfection**

Use a public domain or commercial anti virus package. It is possible to disinfect using ResEdit by removing INIT 29s from the system file, and patching the infected CODE jump table by replacing words 9-12 in CODE 0 with 16-19 of the viral code segment. The simplest approach is to use an anti-viral product.

**The virus is highly infectious and cannot be reliably disinfected unless the system is booted from a clean disk.**

**RE-BOOTING FROM A CLEAN DISK**

*Shutdown your Mac

*Power down for at least 20 seconds

*Power up, inserting a clean, write protected system disk (preferably with the virus sweep and disinfection utility included)

*Immediately run the disinfection utility from the system or another clean write-protected disk.

**Even commercial anti-viral utilities cannot guarantee disinfection of the system if the virus is active in memory. Always boot from a clean system disk before attempting disinfection.**

# VIRUS DISSECTION

*David Ferbrache*

## MIX-1 - Garbled Text and Bouncing Balls

The MIX-1 IBM PC virus was discovered in September 1989 in Israel and is an .EXE infector with a characteristic file extension on infection of 1618 to 1633 bytes. The virus randomly substitutes text characters when files are printed or sent via communication lines from the IBM PC.

### Infecting Memory

The virus, carried at the end of .EXE files, infects memory when executed. The virus checks for a flag at segment 0 offset 33CH which is set to 77h when the virus is active. If a flag is not set the virus copies itself into a specially constructed memory control block (MCB) and then gains control of the PC by redirecting five interrupt vectors - Int 8h (system clock tick), Int 9h (keyboard event interrupt), Int 14h (asynchronous communications service interrupt), Int 17h (printer service interrupt) and Int 21h (main DOS service interrupt).

Redirection of Int 8h and Int 9h is dependent upon an infecting counter mechanism incorporated in the virus' EXE file infection sub-routine. The code to increment this counter is omitted from the virus version disassembled and thus dependent on the initial contents of the count variable. The count controls the activation of a bouncing ball and caps lock change features described later.

### Infecting programs

The Int 21h interrupt subfunction 4Bh (execute program) is trapped by the virus when resident in memory. Subsequently each executed .EXE program of 8K or over in size is infected by:

1. Removing the Read-only attribute from the code file which is not restored if originally present

2. Storing the file modification timestamp

3. Checking for a 4 byte signature at end of file comprising the characters 'MIXI'.

4. Padding file to a paragraph boundary

5. Appending virus to end of file

6. Modifying .EXE file header to invoke viral code

7. Restoring original file modification timestamp

### Asynchronous Line and Printer Output Corruption

The virus intercepts attempts to write characters via asynchronous comms or printer interrupts, translating each character according to the following table:

```
Original   LF  VT  ( )+-
<>ABEFIOPUY[]acefiopsu{}
Modified   VT  LP  ) (-
+><BEAPYUFOI][esapyufco}{
```

Additional foreign and graphics characters are also modified. The phrase the "Quick Brown Fox" becomes "Qoysk Eruwn Pux" when the virus is active.

### Other symptoms

Fifty minutes after the virus has become active in memory it will begin to modify numbers/caps lock settings each time character is entered. The caps lock is released and the numbers lock applied. Sixty minutes after the virus is active the system clock interrupt is used to drive a bouncing ball display similar to the Italian virus display. This comprises a lower case 'o' bouncing of each screen boundary. The character moves through text with replacement of the original text at the rate of 18 character positions per second.

### Source and Commentary

The virus incorporates code from the Icelandic or Saratoga virus including the majority of the .EXE infection routine. The incorporation of the file timestamp and a check to prevent infection of small files are new and may indicate that this is a later strain. Sufficient changes have been made, including the use of MOV instructions in place of XOR, and PUSH/POP pairs instead of MOV to invalidate search hexadecimal patterns taken form the Icelandic virus.

Segments of redundant code and code replaced by NOP instructions are present which indicate that this virus is partially completed or that the author did not have the technical knowledge or access to detailed disassemblies required to product an optimal virus.

The bouncing ball code, although algorithmically similar, shows significant differences to the Italian boot sector virus.

The current trend for computer viruses to be modified to change side effects, while retaining a tested infection core from a previous virus, seems to be increasing. We can expect a number of other such hybrid viruses to appear.

***Acknowledgements:*** *My thanks to Fridrik Skulason of the University of Iceland for providing details of this virus.*

# VIRUS DISSECTION

***Jim Bates***

## dBASE Virus

The dBASE virus was first reported in March of this year by one of our U.S. correspondents. It is a Parasitic Virus, infecting .COM files by appending an additional 1864 bytes of code. As the name suggests, this virus is associated with the dBASE programming environment or, more accurately with files having an extension of .DBF. No indication is given that the virus is resident, all systems appear to function normally but after a period of between three and four months, the virus enters its secondary phase and will eventually cause spurious errors to occur before freezing the machine during the access of DBF files. There is a bug in the copy of the virus that I have, which might cause it to generate errors during use, but without much more detailed knowledge of the inner workings of dBASE than I currently have, I cannot be absolutely certain that such errors **will** occur.

The virus installs itself as TSR within the PC operating system and hooks into the DOS Services Interrupt 21H, monitoring the Create, Open, Read, Write and Close functions. Once installed the virus will attempt to infect other program files with a .COM extension during a Load and Execute function, service 4BH. Since this service accepts a complete file specification including drive and path specifiers, the virus is capable of infecting across directories and drives although these will usually be those of the DOS path.

### Infection Method

The target program file name is first examined for a .COM extension. If this condition is met, the file is opened in READ ONLY mode and the first four bytes are read into memory. The second and third bytes are then assumed to be part of an initial jump instruction and the offset is calculated. The file pointer is then moved to the calculated offset position minus two - and two bytes at that position are read into memory. These two bytes will be E5H E5H if the file is already infected. If these recognition bytes are found, then the infection routine is aborted and processing continues with the original Load and Execute Service call. If they are not found, the file is closed and then re-opened for Write access so that the virus code can be appended to it. At this time, the original four bytes of the host program are within the virus code segment, as are the Date/Time stamp and Attribute settings. Once the virus code has been appended, a new initial jump is calculated and inserted at the head of the program file. Finally, the Date/Time stamp and Attribute settings are reset before processing continues with the original Load and Execute Service call.

It should be noted that the actual code that the virus looks for to recognise an infected program is the word E5E5H at offset 631H into the virus code (=117H offset from the end of code).

### Installation Method

When invoked, the virus code first issues a "Virus recognition call" to determine whether a copy of the virus is already in memory. This call is generated by putting a value of FB0AH into the AX register and then issuing an INT 21H instruction. If the virus is resident, processing returns from the interrupt with the AH and AL sections of AX transposed so that the AX register contains 0AFBH. If the virus is resident, processing continues by replacing the three bytes which were originally at the beginning of the host program and then jumping to offset 100H to transfer control to the host. If the virus is not resident, it is installed by first collecting the current INT 21H vector addresses (Segment and Offset) and inserting these into two places within the virus code. Then the virus code is moved up to the highest point in free memory and the DOS MCB byte is modified to protect the code thus moved. Next, the address of the new INT 21H handler is inserted into the interrupt vector table and finally, processing continues into the host program.

### Operation

This is the first virus that I have seen which actually targets particular data files and maintains a separate file which acts as a log of corruptive activity. The actual DOS services monitored are the major file services accessible via INT 21H - **Create, Open, Close, Read** and **Write**. In describing the activity of the interception code, I shall consider these in turn and collate the implications afterwards.

### Create File Services - 6CH, 5BH and 3CH

A call to any of these file creation services is treated in the same way once minor differences of buffer placement have been rationalised. The virus maintains a "log" file of its activity and the name of this file is hard coded as "C:\BUGS.DAT". When the first Create of Open call is received, the virus checks for the existence of this file and creates it (with the Hidden attribute set) if it does not exist. During a Create call, if the name of the file to be created has

an extension of .DBF then its name is added to the BUGS.DAT file. It should be noted that the original date of the BUGS.DAT file is maintained, since this is used by the virus to determine whether it is time to execute the trigger function. If is also important to note that the actual check to initiate the trigger is only conducted during a file create call. Once the BUGS.DAT file has been updated, the file is created as requested and its file handle is added to a table maintained in memory by the virus. Processing then returns to the caller.

**Open File Service - 3DH**

When an Open File request is received, the file is first opened and then the contents of BUGS.DAT are checked to see if they contain the requested file name. If they do, the file handle is placed in the virus' table of active file handles, otherwise the handle and all subsequent calls involving it are ignored. Processing then returns to the caller.

**Close File Service - 3EH**

A Close File request causes the virus to check whether the file handle is in its table. If found, the handle is deleted.

**Read File Service - 3FH**

When intercepting a Read File request, the virus checks the file handle to see if it exists in the table. If not, the request is allowed to continue unmodified. If the handle is in the table, the file is read as requested but then adjacent bytes in the buffer are transposed. Thus bytes 1 and 2 are reversed, bytes 3 and 4 and so on. Provision is made in the code to ensure that bytes are always transposed on an Odd/Even basis. This transposition process actually mirrors the activity of the Write Service interception and ensures that the calling program receives "uncorrupted" data during the life of the virus.

**Write File Service - 40H**

Once the target file handle has been identified as "belonging" to the virus list (by the presence of its name in the BUGS.DAT file), the write process performs a similar transposition of Odd/Even bytes within the write buffer. Allowance is made for the buffer starting and/or finishing on an odd or even byte boundary so that the transposition remains consistent. Once transposition is completed, the buffer is written to the files as requested. Then the buffer is re-transposed to undo the corruption, thus maintaining data integrity for the calling program. Within the routines which handle the transposition, I discovered a bug which "loses" a single character when the buffer is written to the file if the file pointer is on an odd byte boundary. This will produce errors when the data is read back from the file, but it is impossible to forecast when these errors may occur without detailed knowledge of the particular data file structure and the methods used by dBASE to access files.

It will, therefore, be obvious that the results of being infected by this virus will only be apparent upon direct, uninfected inspection of the contents of the data file. No other manifestation of the virus is apparent during file access. This seemingly pointless exercise suddenly gains meaning when we examine the trigger mechanism. As mentioned, this only occurs during a Create File call and consists of a routine which checks the date of the BUGS.DAT file against the current system date. Only the month portion is checked and the trigger conditions are met if the difference between the two month readings is negative (ie: next year) or equal to or greater than three. Thus if the month portion of the date on the BUGSDAT file is January, then the virus will trigger when April arrives. This trigger point will therefore arrive at some time after two months and a day have elapsed since the creation of the BUGS.DAT file. The trigger routine itself is something of an enigma since it issues two successive INT 3 calls within a count loop contained in the AL register, and then goes into an infinite loop to "hang" the machine (*but see box at end of article - Ed.*). INT 3 is the breakpoint interrupt usually used by monitor and debug programs. Within normal DOS operations this interrupt routine only contains an IRET instruction and thus returns immediately to the caller without accomplishing any function. I am not sufficiently familiar with the internal workings of the dBASE environment to forecast with any accuracy exactly what INT 3 is used for, by I would hazard a guess that it is probably a service routing for error reporting and recovery. If so, the result of the trigger being invoked would be to force dBASE to display a succession of totally misleading error messages before freezing.

**Observations**

There are several disturbing aspects to this virus, not least of which is the length of time between infection and final trigger. Remembering that the trigger will only operate during file creation, the time period could well be much greater. **This is important because all backups of affected data files would become unreadable once the virus had been removed.** The BUGS.DAT file is easily removed but since this contains a list of filespecs which have been corrupted, its removal will cause immediate errors to appear when corrupted files are next read. It is also worth noting that the DOS File Rename service is **not** intercepted so that if a corrupted file is renamed, the virus will no longer be

able to apply the de-transposition to it and its corruption will become apparent to the accessing program. This would affect programs which used the .DBF extension and which used a write - delete old - rename new to old, type of processing for database management since the new file would be listed in the BUGS.DAT log, while the old file may not be. Under these circumstances creating and writing to the new file would introduce corruption while updating the same file under its new (old) name would not. It is also quite possible that since the virus only infects .COM files and at least one version of dBASE runs from a .EXE file, a file could have been created on an infected system and then updated later on an uninfected system. Both of these situations would produce the presence of the virus.

### Conclusions

**The fact that this virus corrupts data files makes it extremely dangerous and its construction means that even long term backups could be corrupted without the user being aware of it**. It is therefore paramount that the virus is detected and removed as quickly as possible. A recognition pattern has been extracted and added to the *Virus Bulletin* list of patterns. To date, only **one** affected site (which was in the United States) has been reported.

### Detection on Disk

An infected file may be recognised (using *The Norton Utilities* or a similar disk sector examination utility) by an initial jump to an offset of 276 (decimal) (114H) from the end of the file. A somewhat easier thing to look for is the two INT 3 opcodes at offsets of 10 and 11 (decimal) from the end of the file - these will show up as CC CC.

Alternatively a scan program which has the search signature for the dBASE virus incorporated into it will probably be a much faster method. You should also look for the hidden file BUGS.DAT in the root directory of the C: drive. If this file exists, an examination of its contents will indicate which data files have been affective by the virus.

### Detection in the System

If the virus is resident in the system, its own recognition call may be used to detect its presence. This is executed by placing a value of FBOAH into the AX register and issuing an INT 21H call. Upon returning, AX will contain OAFBH if the virus is resident or FB00H if is not.

### Removal

If the virus is found, you should first note the contents of the BUGS.DAT file. Probably the best course of action would be to remove its "hidden" attribute but leave it there for the time being. If there are any affected data files, you should remember that they will have been corrupted (either continually or intermittently) since their creation. It may be possible to recover some clean data from a file corrupted by this virus, but the process would be difficult and complex and should only be attempted by someone with intimate knowledge of how the virus works. The best course would probably be to generate a printed listing of all the data while the machine was still infected, and then to reboot the machine to clear the virus from the system. Then all infected program files should be replaced and corrupted data files deleted. Finally, new files could be created by re-entering the printed data via the keyboard.

### Addendum

Dr Jan Hruska has pointed out that the apparent enigma surrounding the virus payload utilising two INT 3H in succession could be, in fact, a pre-release (test) version of a very much more destructive virus. If the two INT 3H instructions, **which assemble as 2 bytes** are replaced with one INT 26H instruction **which also assembles as 2 bytes**, the virus payload becomes highly destructive. On delivering the payload, the virus will overwrite the first 256 sectors of each drive from D: to Z: before disabling the interrupts and hanging the machine. Some BIOSes, however, will reject the INT 26H request to write more than 64K, returning the boundary error. IBM XT, for example, rejects the call, while the Comcen XT clone accepts it and writes the first 64K to the disk.

The above points to the fact that the virus writer had released a virus which could be quickly and easily modified into a destructive one by either changing two bytes or by producing a simple co-program which sets INT 3H address to INT 26H address in the interrupt table.

**Note: We have published TWO hexadecimal patterns for the dBASE virus: one for the virus sample we have and one for the potentially destructive version which may also be in circulation.**

# PRODUCT UPGRADE

***Phil Crewe***

## Virex 2.12

This article is not meant as a full evaluation of the Virex virus guard package for the Macintosh. It is meant, instead, to indicate some of the features of the latest version (2.12) which subscribers to the Virex upgrade system should have received recently.

The package comes as an INIT called VirexINIT (version 1.12), which was previously called VirexGuard (version 1.11), and an application called Virex 2.12. Both are dated Sunday September 17th 1989.

The INIT should be placed in the System Folder, and is actually an INIT within a within a Control Panel document. Of note is the fact that is that there is no special character at the beginning of the name. Such a feature would ensure that the VirexINIT loads first, or is at least one of the first INITs to load during system power-up. Its absense is worrying in a much as an infected startup document could load before Virex has a chance to inform you of the fact, and indeed could be coded in such a way as to render Virex useless. This shortcoming, noticeable in the previous version, has not been corrected.

Control of the INIT is provided through the Control Panel, though is should be said that this control is limited, and certainly people that have seen the Symantec Antivirus intercept in action will feel somewhat constrained at the level of customisation available.

On restarting the Macintosh the INIT loads up (in my case virtually last) and the Virex icon flashes mid-screen until the desktop and finder appears. Also when you are doing something else on the Macintosh (like opening the Control Panel) this Virex icon starts flashing again midscreen. This appears to be an information device to let the user know that diagnostic scanning of something is taking place, and this is a feature which is an addition on this upgrade. There is an item in the Control Panel to enable or disable this "flashing icon" feature. I must confess that it was one of the first things which I turned off.

When you insert a disk into the Macintosh the VirexINIT then intercepts the mount procedure and scans the floppy. This can be configured to either always scan or scan on request (the default). The user is then informed if any viruses have been found. When a virus is located, the user must eject the floppy, and this (as with VirexGuard 1.11) activates the Shield INIT part of Symantec Utilities for Macintosh. The floppy is ejected and then Shield appears to do a volume save on the internal hard disk. This still occurs with this version, and I am not currently sure why it should do it at all. I will do some further tests with later versions of SUM, and hope to provide an explanation in a forthcoming full evaluation of Virex.

On opening the virex 2.12 application you are then presented with the normal run-down information about author, company, limitations of Virex, and a good note of what all of the icon buttons do. It remains very much a button-based application. There are normally three available buttons, which are Diagnose, Repair and Help. Invoking Expert Mode (which is an option) gives you a fourth button called Record/Scan. This can be used to snapshot disk file and to check whether or not the file changes. This is the route for "not known at this time" or non virus specification detection.

When the Virex application is running, the VirexINIT is disabled (so that an infected floppy can be mounted on the desktop). As with version 2.1 all available disks for scanning appear as icons along the bottom of the screen, and are normally black outlined. Clicking on the outlined disk icon causes it to go grey (and vice versa) thereby deselecting it for scanning during diagnose or repair.

Initial testing has indicated that Virex 2.12 and VirexINIT 1.12 will successfully detect AIDS, ANTI, MEV#, INIT29, HPAT, nVIR A and B, nFLU and Scores, but not local variations on nVIR strains or either variation of Peace (RR or DR). There is very little change between this release and the previous 2.1 release (which added detection for nFLU).

This upgrade places some emphasis on bug-fixing. The new version is described as containing "minor changes which will make Virex easier to use a well as corrections to minor INIT conflicts which have been reported". I presume the flashing icon (which I immediately disabled) is intended as a minor change for ease of use. I hope the person who asked for it is happy!

**Product:** Virex

**Manufacturer:** HJC software Inc., PO Box 51816, Durham, North Carolina, NC 27717,USA.

**Price:** Annual subscription with updates: $99.

*A thorough evaluation of Virex is currently underway, the results of which will appear in the Virus Bulletin in the new year.*

# PRODUCT REVIEW

***Dr. Keith Jackson***

## Mace Vaccine

Mace Vaccine is a software package that tries to prevent a virus from replication, and prevent viruses (or any other form of malicious program) from altering disk files.

Mace Vaccine is a memory resident utility which monitors disk system activity, and triggers when 'illegal' operations are detected. Different levels of operation provide various definitions of 'illegal' activity (*see below*). The MS-DOS operation system is required. When Mace Vaccine is triggered it asks the user whether a particular action should be allowed or not (or if the number of warnings has got so annoying that protection should be removed altogether).

### Documentation

Mace Vaccine comes in a slim 15 page manual, and a singe 5 ¼ or 3 ½ inch disk. The manual does not contain and index, but at this size, such an omission is not pertinent. The manual uses terms such as *Trojan Horse* and *Time Bomb* but it does not define them. If you are completely unfamiliar with such jargon, then some background reading will be necessary. Apart from these minor quibbles the manual is well written, if rather terse. The manual clearly states that "The only infallible protection is to have a copy of data stored somewhere else". I could not agree more. This statement is an honest recognition by the manufacturer that Mace Vaccine (like any other package) is not a cure-all.

Mace Vaccine

can be used in three distinct ways, described within the manual as **Level 1, Level 2** and **Level 3**.

**Level 1** operation prevents any attempt to alter the boot record, and/or the partition table of the hard disk. It also prevents any attempt to update the File Allocation Table, and/or root directories, which has not been made via an MS-DOS function call. For this reason, Mace Vaccine must be turned off before disk test programs, or low level programs such as *The Norton Utilities* or *PC-Tools* are executed. Level 1 operation also prevents floppy disks from being formatted, or their boot record amended, and it detects alterations to MS-DOS system files.

**Level 2** operation encompasses all of the protection offered at Level 1. It also prevents any attempt to writer directly (not via an MS-DOS function call) to the hard disk. Note that this prevents the MS-DOS program CHKDSK from clearing up lost disk clusters unless Mace Vaccine is temporarily disabled.

**Level 3** operation encompasses all Level 1 and Level 2 protection. In addition it checks all programs for alteration before they are executed. A program that has been altered cannot be executed while Mace Vaccine is active. Level 3 operation requires that checksums are calculated for every executable file. This requires a program called 'Survey' to be executed.

Survey checks all files with the extension EXE, COM, BIN or OVL. My hard disk currently contains 811 files in 14.7 Mbytes, and out of all these files, 127 (comprising 5.43 Mbytes of executable files) were checked by Survey. It took 6 mins 50 seconds for Survey to complete its work, corresponding to a checking rate of only 13Kbytes per second. A back of the envelope calculation will reveal some daunting execution times for anyone with a large full, hard disk. I'm fully aware that a big hard disk will probably be associated with a fast processor, but I would not be surprised if Survey routinely took over 10 minutes to calculate the checksums for a large hard disk.

I had no problem in getting Mace Vaccine to work at any of the three levels of protection. Installation was simply a matter of executing the MS-DOS command "VACCINE", whereupon the default Level 1 protection is installed. Vaccine occupies about 6 Kbytes of computer memory. This is in spite of the fact that the executable file VACCINE.EXE is 12 Kbytes long. Obviously, onlt part of this file becomes memory resident. Switching between the various levels is simply a matter of executing the MS-DOS command used to initiate Mace Vaccine operation, with the intended protection level added as a parameter. Mace Vaccine spots that it is already resident in memory, does not load itself again, and switches to the requested level.

### User-Interface

Using Mace Vaccine at Level 3 noticeably slows down loading programs from disk. This is because the checksum for the program has to be recalculated and verified. I measured the increase in load time with WordStar and Turbo Pascal. WordStar version 5.01 takes 6.6 seconds to load normally, this rose to 12.6 seconds when Level 3 Mace Vaccine was active. Under similar circumstances, Turbo Pascal version 5.5 normally takes 4.7 seconds to load, this figure rose to 12.6 seconds. In broad terms, Level 3 operation doubles the program load time. If you regularly transfer between programs, then you'll find this delay irritating. However, if you use one program such as a word processor all day, then it becomes irrelevant. Of course you can always operate Mace Vaccine at a lower level of protection (see

above for details).

I encountered very few problems while using Mace Vaccine. However, at the highest level of operation, Mace Vaccine appeared quite often, enquiring whether I wished to proceed with certain operations. This is more a reflection of what I use my PC for than a reflection on Mace Vaccine. Operation via standard MS-DOS function calls does not cause a problem. If the program becomes irritating while a specific software package is being run (one that is known to be 'safe'), then Mace Vaccine can be switched off temporarily. Similarly, you can instruct Mace Vaccine to permit any request that applies to floppy disk, while still protecting the hard disk. This is very useful if a batch of floppy disks is being prepared by first formatting each floppy disk, and then writing files to the floppy disk.

Some software packages (even some virus detection programs) make configuration changes by directly amending the executable file. Of the more commonly used programs, WordStar is probably the best known example of this style of configuration. This can be a nuisance if you've just installed a program, and the configuration keeps changing as you learn how to use the product. Software packages that use a specific configuration file are not affected as Mace Vaccine only holds checksums for executable files which have one of four specific extension types (see description of 'Survey' above). Recalculating the checksums is time-consuming - Survey is very slow, takes many minutes to recheck a disk, and always checks the entire disk. An operation to recalculate the checksum for specified file(s) would make life somewhat easier.

All this means that Level 3 protection, and installing software package which updates executable files directly, are incompatible. This is a shame, because it is when a new software package is installed that a virus can be introduced (attached to the new software).

### Security

My main bone of contention with Mace Vaccine is with the algorithm used to calculate checksums. I've complained before about other products, and I'll say it again, there is no mention in the documentation of the algorithm used by Mace Vaccine to calculate its checksum. It is important that such an algorithm is cryptographically strong (see *Checksum Methods Used to Detect Virus Attacks, VB Sep 89.* for explanation).

I spent a couple of hours investigating how the algorithm used by Mace Vaccine appears to work. I have no inside knowledge of any details of its operation, and did not disassemble any program, but within the aforementioned couple of hours, I succeeded in developing a Pascal program that could mimic the checksum calculation process. It would be irresponsible to publish details, as this could be detrimental to the users of Mace Vaccine. Suffice it to say that initial investigation has produced a program capable of calculating checksums for small files, and more work should permit the algorithm to be reverse engineered completely.

If I can reverse engineer the checksumming process and figure out how it works, then so can the author of a virus. In the principle there is then nothing to prevent a virus amending a file, and adding extra data to make the checksum revert to the original value. Other algorithms have sadly also proven easy to reverse engineer (see *Technical Review of Dr. Solomon's Anti-Virus Toolkit, VB Jul 89*). Copies of the programs developed to facilitate reverse engineering are currently in the possession of the Editor.

### Summary

In common with last months review of FLUSHOT+, Mace Vaccine offers protection against a virus, a Trojan Horse, or a potentially destructive software bug. It stops all of them from corrupting or destroying data held on disk. From the user's point of view, it is probably one of the easiest anti-virus packages to install, and changing between the various levels of protection is very straightforward. Mace Vaccine is altogether a good product, it certainly spots malicious actions, but I have reservations about the algorithm which it uses to calculate checksums.

**Technical Details**

**Product:** Mace Vaccine

**Developer:** Paul Mace Software Inc., 400 Williamson Way, Ashland, OR 97520, U.S.A., Tel. +1 (503) 488-0224.

**Vendor in the US:** Fifth Generation Systems Inc., 11200 Industriplex Blvd., Baton Rouge, LA 70809-4112, U.S.A., Tel +1 (504) 291-7221.

**Availability:** IBM PC/XT/AT, PS/2, or any close compatible running MS-DOS version 2.0 or above.

**Version evaluated:** 3.0

**Price:** $99.00

**Hardware used:** ITT XTRA (a PC compatible) with a 4.77MHz 8088 processor, 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.

# BOOK REVIEW

***Anthony Naggs***



**The Complete Computer Virus Handbook**
ISBN: 0 273 03255 0, Price $14.95
Available at bookshops or from the publisher - Pitman Publishing, 128 Long Acre, London, UK, WC2E 9AN

# EVENTS

**MVS Audit, Control and Security Workshop,** 7-8 December, London. Details from MIS, The Netherlands, Fax + 31 70 64 99 16

**Hacking - a New Law?** Barbican Centre, London, UK, 11 December. Emma Nicholson MP continues her crusade against those naughty hackers. Details from Chapman Beggs, 37 Lambs Conduit Street, London WC1N 3NG, UK

**Risk Management - the Practical Approach**, London Press Centre, 12 December. One day conference followed by in-depth workshop on 13 December. Details from IBC Technical Services, UK Tel 01 236 4080

Sophos Ltd continue a series of **Virus Workshops**. The next available workshops are on 24-26 January 1990 in London and 27-28 March 1990 in Oxford. Management and Technical streams are available. Details from Karen Richardson at Sophos, UK, Tel 0844 292392

**Corporate Computer Security '90**, 13-15 February 1990, Novotel, London, UK. Details from PLF Ltd, UK, Tel 0733 558571

**Network Security: Managing the Risk**, Kensington, London, 20-21 February, 1990. Details from the Informatics Resource Centre, UK Tel 0871 2546

**Disaster Recovery and Contingency Planning**, 1 March 1990, Kensington, London. Details from Informatics, UK, Tel 01 871 2546

**SECURICOM '90,** Computer and communications security conference, La Defense, Paris, France, 13 March 1990. Details from SEDEP, France, Tel +33 1 4742 4100

**SICUR '90**, Computer security conference/exhibition. Madrid, Spain, March 13-16, 1990. Details from IFEMA, Spain, Tel +34 91 470 10 14

**COMPACS '90,** Hilton Hotel, London, UK, 20-23 March 1990. International conference on Computer Audit, Control and Security. Details from the IIA, UK, Tel 01 498 0101.

**Enigma Variations**, Wembley Conference Centre, London, May 2-3 1990, Conference and exhibition. Details from Gill Spear, Elsevier, UK, Tel 0865 512242.

**IFIP/SEC '90**, The sixth international conference and exhibition on information security, Espoo, Finland, 23-25 May 1990. For details contact Congrex, Finland, Tel +35 80 175355, or Jugani Saari, Finland, Tel +358 0 177901


## 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