VIRUS ANALYSIS 1

One_Half: The Lieutenant Commander?

Eugene Kaspersky
Kami Associates

Two years ago, a virus appeared which amazed researchers with its infection algorithm. Regular *VB* readers will remember Commander_Bomber (see *VB*, December 1992), which caused numerous problems for researchers by inserting its code into a random location within an infected file. Control does not pass from the beginning of an infected file directly to the main virus body: several blocks of polymorphic code pass control from one part to another, before the main body of the virus is executed.

This means that the standard method of calculating a virus' offset in a file cannot be used, and many anti-virus scanners still do not detect the virus correctly (at least, not when they are run in their default modes).

Now a new virus has appeared - a polymorphic, multipartite sample 'à la Commander Bomber'. Like that virus, One_Half (the name comes from its internal text string, 'Dis is one half') writes polymorphic code into random positions in the file. These 'spots' of code not only pass control to the main virus code, but also contain a loop which decrypts the main body of the virus.

Commander_Bomber is not encrypted, and can be found simply by scanning the whole file. The One_Half virus, on the other hand, is, and cannot be detected using a simple hexadecimal search string. Moreover, the decryption routine is broken up into several pieces, making decryption tricky.

Execution of Infected File

When an infected file is executed, control passes to the decryption code. The decryption loop contains ten blocks of code which are placed at random locations throughout the host file: the first five initialise registers for the decryption loop; the rest decrypts the virus body. Each block contains only one function, on completion of which there is an immediate near JMP to the next block. The last block passes control to the virus' installation routine.

The virus' first action is to issue an 'Are you there?' call (Int 21h with AX=4B53h). If a copy of the virus is already memory-resident, the value 454Bh is returned in AX. If the call is answered, the memory image of the host file is repaired and control passed to it.

If the virus is not already memory-resident, it tunnels the Int 13h vector and reads the MBS to check for the virus' presence, comparing the word at offset 0025h with value 00D3h. If this condition is met, the virus skips the infection routine and returns to the host program. A similar test is made for the value of 072Eh at offset 0180h in the MBS. This part of the boot sector does not contain viral code, and I see no reason for the virus not to infect such disks, unless to prevent conflict with another program. Another possibility is that it might have been used by the virus author to keep his own computer clean during development of the virus.

Next, the virus checks disk parameters, using function Int 13h, AH=08h, and saves the original MBS (and its own unencrypted complete code) in the last eight sectors of track 0. If the disk has been partitioned in the usual way, these are the sectors before drive C's DOS Boot Sector. It then copies 29h bytes of code (which read the virus code from the infected sectors and pass control to the virus) into the original MBS, and writes the MBS back to disk.

"the hooked Int 13h performs two functions ... the first is the trigger routine, the other, the stealth mechanism code"

After hard drive infection, One_Half modifies the Memory Control Blocks (MCBs) in the standard manner, disguises itself as a copy of COMMAND.COM (by copying the 'COMMAND' string into the MCB 'program name' field), and hooks Int 21h. This routine is somewhat unreliable - the virus did not become resident on my test computer during normal operation, functioning correctly only when executed under the control of a debugger.

Finally, the virus restores the infected host program, and passes control to it. If the file is in EXE format, the virus reads the file header and corrects the words to which the Relocation Table points, in addition to returning the decryption blocks to their original form.

The last part of this process is necessary due to the fact that, on infection, the virus overwrites randomly-selected bytes of the host program and may corrupt bytes containing information on the Relocation Table.

Loading from the Hard Drive

When the machine is booted from an infected MBS, the virus' header decreases the size of system memory (offset 0000:0413), copies the virus body into the memory area thus reserved, and passes control to the copy.

The installation routine hooks Int 13h and Int 1Ch, then reads the original MBS and passes control to it. Several other multi-partite viruses use Int 1Ch in a similar manner:

the code checks the Int 21h handler address; if changed (as it will be when DOS is loaded), it saves its current value and points the new Int 21h vector to the virus code.

The hooked Int 13h performs two functions; the first is the trigger routine, the other, the stealth mechanism code. On accessing the infected MBS through READ and WRITE functions (Int 13h, AH=02h,03h), the virus redirects the call to return either the uninfected MBS or a buffer full of zeros.

File Infection

One_Half intercepts a long list of Int 21h functions: the file infection routine is called from the Int 21h handler. On calls to FINDNEXT and FINDFIRST functions (AH=11h, 12h, 4Eh, 4Fh), the virus calls a semi-stealth routine which 'decreases' the apparent file length. On opening, renaming or execution of a file (AX=3D??h, 4B00h, 56??h), the infection routine is called. If a file is created (AH=3Ch, 5Bh), the virus stores its name and infects it when closed.

Before infection, the virus checks the file name, only infecting files with a COM or an EXE extension. Then it checks for the strings SCAN, CLEAN, FINDVIRU, GUARD, NOD, VSAFE, MSAV: if any of these is found, the file will not be infected. The virus looks particularly carefully for the CHKDSK utility and disables the semi-stealth routine during execution of that program, preventing CHKDSK from raising an alarm over lost disk space.

One_Half then checks the file's date and time stamp, which is returned in two registers, CX and DX. The CX register, contains the date stamp (year, month and day); the DX register, the time stamp (hour, minute, seconds). One_Half divides the DX register (time stamp) by 30, and if the result equals the seconds stamp, that file will not be infected. Oddly, one time in 30 the virus does not mark infected files, so it is likely that some files will be multiply-infected.

If the date/time stamp allows infection, the virus executes its polymorphic routine. This selects several random offsets in the file, copies the code from the offsets, overwrites that code with parts of the decryption loop, and encrypts and saves the virus body at the file end. The virus code is at a constant offset from the file end, so a scanner can detect the virus by checking the end of the file, rather than the file header - a useful weak point. Unfortunately, the code is encrypted with a randomly-selected key, and a special routine must be written to 'x-ray' it and catch the virus.

The Trigger Routines

There are two trigger routines: the first is complex, and many attempts to execute it failed. When this routine is called, the virus analyses the size of the DOS primary partition or the extended partition, if present, and encrypts part of the latter with an XOR instruction and a randomly-selected key. The virus decrypts partition sectors 'on-the-fly' before writing or after reading. The partition is available under an infected system, and lost after virus removal. I can just hear the telephone calls to anti-virus vendors: 'Your software disinfected the virus, but we lost all data on the hard drive!'

The second routine is called when the virus is installed in system memory. The virus checks a generation count and tests the system timer value: if these conditions are 'good', the virus displays the message:

```
Dis is one half.
Press any key to continue...
```

and awaits a keystroke. It also contains the internal string 'Did you leave the room ?'

Conclusions

One_Half poses many problems to the developers of anti-virus software. The most pressing of these is the difficulty in removing the virus from infected disks: the usual simple-minded approach of replacing the disk's Master Boot Sector is not enough. This makes it worthy of further attention, and extreme care should be taken when removing it from an infected disk. Any predictions for the next new threat?

One_Half

Aliases: Free Love.

Type: Memory-resident, multi-partite, polymorphic. 3544 bytes long.

Infection: COM and EXE files, MBS of hard drive.

Self-recognition on Disk:

Checks the word at offset 0025h for the value 00D3h.

Self-recognition in Files:

Checks the file date and time stamp.

Self-recognition in Memory:

Via 'Are you there?' call. INT 21h, AX=4B53h returns 454Bh in AX.

Hex Pattern: No search pattern possible in files. One_Half-infected MBS:

```
33DB FABC 007C 8ED3 FB8E DB83
2E13 0404 B106 CD12 D3E0 BA80
```

One_Half resident in memory:

```
9CFB 80FC 1174 0580 FC12 752F
EB?? 5306 50B4 2FE8 7FFC 58E8
```

Intercepts: Int 13h for stealth and trigger routine, Int 1Ch for installation on loading from infected MBS, Int 21h for infection.

Trigger: Encrypts sectors of the hard drive, displays message.

Removal: Can be difficult, due to encryption of sectors in the DOS partition.