VIRUS ANALYSIS 1

Dichotomy: Double Trouble

Eugene Kaspersky
KAMI Associates

How does one define a computer virus? One possible description is of a block of code which has the property of self-replication, 'infecting' other objects on the system. When the virus spreads, that block is not divided or cut: each replication of a file-infecting virus contains functionally identical executable code. The entire body of the virus is contained in every infected file.

This generalisation may also be applied to boot sector and multipartite viruses. As a rule, all known viruses write their whole code and data on infection. However, as with any rule, there are exceptions. In this case, the exception is the Dichotomy virus.

The virus takes its name from the internal text string '[Dichotomy](c) 1994 Evil Avatar [Dichotomy]'). It is the first virus to use an algorithm which does not place all of its code in every infected file: rather, the virus copies different parts of its code into two different files. When one of these two infected files is executed, the virus becomes active.

Separated Code

The virus code is separated into two blocks, 296 and 567 bytes long respectively. The first section of the code is the virus loader: the virus writes this into files in the standard manner, appending its code to the end of the file, and replacing the first three bytes of the host file with a jump to the attached virus code.

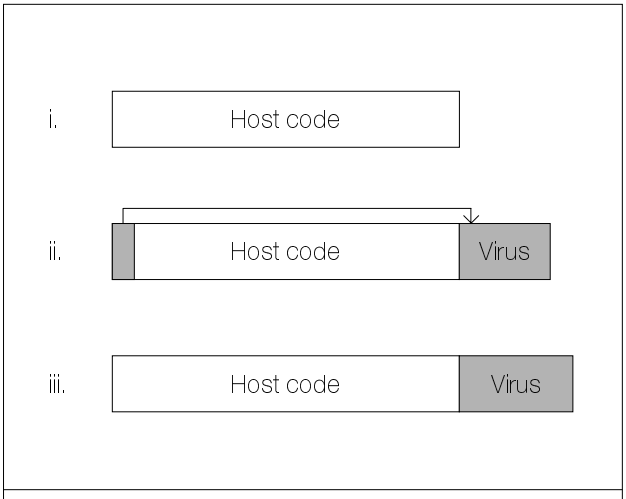

Figure 1: The appearance of the two different types of infected file. i. An uninfected host file. ii. An 'even' file, infected with the loader code. iii. An 'odd' file. Note that the entry point of the file has not been altered.

The second 567-byte block of code contains the remaining virus functions: the code which installs the virus in memory and the Int 21h handler. When files are infected with this second block of code, the virus does not modify the header of the targeted file, but appends that block to the file with no modifications in the original file body. The appearance of the two different types of infected file is shown in Figure 1.

This infection mechanism means that the virus code will not be executed when files infected with the second block of code are executed, as there is no JMP to the virus code. The virus can only be activated when a file infected with the virus' loader code is executed.

Installation

When a file infected with the virus loader is run, control is passed to the virus code appended to it. Processing then transfers to a routine that searches for a predetermined file which has the second part of the virus code attached to it. When such a file is located, the remainder of the virus code is loaded into memory, thus recreating the complete body of the virus in memory.

Dichotomy checks the second part of the code for an identification word, 445Bh or '[D' in ASCII, before continuing installation. This word, located at offset 0352h from the virus' beginning, is taken from an internal text string.

If the identification is positive, an 'Are you there?' call is made. This consists of calling Int 21h with AH=51h (Get_PSP_Address), with ES:BP pointing to the start of the virus code. If there is already a copy of the virus resident in memory, the 'Are you there?' call returns the value FFFFh in the BX register.

Should the 'Are you there?' call go unanswered, control passes to the installation routine, which is located in the second section of the virus body. This routine allocates a block of system memory, copies the virus code into it, modifies an undocumented Memory Control Block area, and hooks the Int 21h vector. The virus then restores the header of the host program, and passes control to it.

Int 21h Handler

The virus hooks Int 21h and checks three of its functions: Get_PSP_Address (AH=51h and AH=62h) and Load_and_Execute (AH=4Bh).

As stated above, Int 21h subfunction 51h is used as the virus' 'Are you there?' call. However, rather than simply checking the value of certain registers, the virus compares the bytes to which the register pair ES:BP points with its own code. If these bytes do not match, the call passes to the original Int 21h handler. This method of checking for an

already-resident copy of the virus is very effective, as it avoids the use of a non-standard Int 21h call, making the virus less likely to clash with other software.

The other two intercepted functions are used for infection. The virus has two different infection modes (which I shall label infection of 'odd' and 'even' files), and toggles between them every time a new file is infected.

If a Load_and_Execute call is intercepted, and the target file is deemed suitable for infection, the virus checks which infection mode it is in. If it is an 'odd' file, the resident code appends the second block of virus code to it, and makes no alteration to the file entry point. The name of this file is then patched into a data area in the resident copy of the first block of code.

The next file the virus attempts to infect is classified as 'even': the first virus code block (the loader) is copied to the end of the infected file, changing the first instruction of the file so that control is passed to the virus.

"Dichotomy ... copies different parts of its code into two different files"

The virus pays special attention to files located on diskette. To ensure that these files contain the complete virus code, it infects them with both the first and the second blocks. As a result, each infected file contains a complete copy of the virus, just like a standard file infector.

Such double infection, where both parts of the virus are added to the same executable, can also occur if the loader part of the virus cannot locate the filename stored inside it. In this case, the loader issues a call to the memory-resident copy of the virus via the third hooked function - Int 21h, subfunction 62h. This appends the second part of the virus code to the calling file. As with an 'Are you there?' call, the memory-resident virus compares the code of the program performing that call: only if the call was made by another copy of the Dichotomy virus is the infection procedure carried out.

After infection, the virus modifies the host file's date and time stamps. It sets the seconds value to 60 for 'odd' files (containing the 'loader' block), and to 62 for 'even' ones (which contain the 'installation' block). Infected files on floppy disks have a 62 second value in their date and time stamps. This stamp is the only form of identification used by the virus to separate infected files from clean ones.

Malfunctions

There are at least two programming bugs in the Dichotomy virus, the first of which occurs on execution of the virus loader. This results in the check of the virus' identification word being carried out incorrectly.

The second (and most important) bug is the fact that the virus infects EXE as well as COM files. On infection, the virus reads the beginning of the file and attempts to check its internal format in the standard manner, searching for the EXE stamp ('MZ' or 'ZM' word). However, there is a bug somewhere in the virus code, which appears to be a missing instruction. This results in EXE files being infected as if they were COM files. When such misidentified files are executed, they cause the system to hang.

Several other inadequacies in the virus' algorithm should also be mentioned. On accessing files, the error flags are not checked as they should be, and file length is not checked correctly: resulting in the corruption of executable files which are very short. Additionally, the virus does not hook the Int 24h vector to prevent the display of DOS error messages when an attempt is made to infect files on a write-protected diskette.

I see this as a new type of experimental virus which can never become prevalent in the wild. In my opinion, the only reason for its conception was to demonstrate just how 'smart' the virus-writers can be, and to give an illustration of the 'dichotomy' infection technique.

Dichotomy

Aliases: Evil Avatar.

Type: Memory-resident, appending parasitic file infector.

Infection: Any file executed by a Load_and_Execute function.

Self-recognition in Files:

Checks file time stamp for the value 60 or 62 in the seconds field.

Self-recognition in Memory:

An Int 21h call, with AH=51h (Get_PSP_Segment), and ES:BP pointing to the start of the virus code returns FFFFh in BX register.

Hex Pattern: There are patterns for each part of the virus; both can be used to scan system memory.

Part 1:

```
E800 008B DC8B 2F81 ED03 0044
443E 81BE 5203 5B44 B41A 8D96
```

Part 2:

```
FEC4 80FC 4C74 32FE CC80 FC51
740C 80FC 6274 052E FF2E 8C03
```

Intercepts: Int 21h for infection.

Trigger: None.

Removal: Under clean system conditions, identify and replace infected files.