VIRUS ANALYSIS 2

Nexiv_Der: Tracing the Vixen

Peter Szor

Every year there are new developments in virus-writing; virus authors are constantly working to keep the anti-virus vendors on their toes. Occasionally, a new infection method pops up which can require a fundamental change to the way a virus scanner works: Nexiv_Der belongs in this category.

This virus uses a peculiar method of infection. It traces the execution of a victim file and randomly inserts a call to its own code in the middle of the host: this makes it difficult to use emulation or entry point scanning methods to detect it. It is multi-partite, and polymorphic in files and in boot sectors, but most interesting is its tracing method, which provides it with generic anti-goat and anti-heuristic capabilities.

The encrypted string 'Nexiv_Der takes on your files' found within the code gives the virus its name – but the message is never displayed. Indeed, the virus has no trigger; it just spreads.

Theory

'When an infected program is first executed, the virus code is executed first. Then the virus takes control and goes resident…' This definition is true for most file-infecting viruses, but Nexiv_Der is different: it traces its victim's execution and inserts the patch to its code at a random point.

Consider the DOS utility MODE.COM: MODE is a multi-purpose tool which can be used to control such things as keyboard speed, COM and LPT port settings, code pages and the number of text lines on screen. Different operations are controlled by command-line parameters.

Let's look at two cases where Nexiv_Der is resident and ready to infect this file. First, the user executes the command MODE COM1 BAUD=19200. The virus traces the program to the COM-handling routines of MODE, located mostly in the last third of MODE.COM. It then patches a call to its code in the middle of these COM port handling routines. Now the file is infected, but the virus is unable to spread unless the user runs it with a similar command-line.

In the second example, the user executes MODE CON RATE=32 DELAY=1. Nexiv_Der traces the keyboard handling routines of MODE.COM, and inserts the patch in another part of the file, probably near the beginning.

So, a Nexiv_Der-infected file is usually only able to spread further if the infected file is executed in a similar environment and with similar settings. Since most of the more advanced anti-virus programs use emulation to detect viruses, they will have problems with this sort of infection technique. How does an anti-virus program locate where the execution flow transfers from host program to virus? If the file was originally infected when used with a special command-line switch or in a special environment, how can the anti-virus program emulate the circumstances?

Infection

When an infected file is executed and Nexiv_Der takes control, it first decrypts – the first byte of the polymorphic decryptor is always a PUSHF command. Next, the virus calculates a checksum across its own code, and if the checksum has changed, returns control to the host program.

If the checksum matches, Nexiv_Der issues its 'Are You There?' call, Int 21h AX=304Eh, and expects the answer AH=30h. This is the DOS version query command: normally, the AL value is ignored. This means that many programs leave AL uninitialised when calling this interrupt and can accidentally trigger Nexiv_Der's 'Are You There?' call, getting odd results for the DOS version number.

If the call is not answered, Nexiv_Der reasons that it has not infected the DOS boot sector of the hard drive, and proceeds to do so. Unlike most other multi-partite viruses, Nexiv_Der infects the DOS boot sector, not the MBR.

The original DBR is moved to track 0 sector 6, the DBR is modified to contain a polymorphic loader (see below), and the rest of the virus is written to the twelve sectors from sector 7. After the DBR has been infected, Nexiv_Der does not stay resident, but waits for the next reboot to continue.

Going Resident

When the machine is rebooted from the hard drive, the virus is run. It first decrypts a small section of loader code which brings the virus body and uninfected boot sector into memory. Like almost all boot sector viruses, Nexiv_Der obtains the amount of DOS memory from the BIOS data area and copies the resident part of its code to just below the top of memory. Then it reduces the amount of system memory by 5KB, enough for its resident copy.

After storing the address of the original handler within its code, the virus hooks Int 13h. Finally, it re-starts the boot procedure, with the new interrupt handler in place.

Int 13h Handler

The first purpose of the Int 13h handler is to detect when DOS has been loaded so the virus knows when to hook Int 21h. To do this, the Int 13h handler checks the start of every sector read for the 'MZ' file marker. When the third such marker is seen, the virus hooks Int 21h. As modern device drivers are stored in EXE files, this usually happens when the third device driver is loaded from CONFIG.SYS.

Many multi-partite viruses detect when DOS has been loaded by hooking Int 1Ch (Timer) and waiting for the Int 21h vector to change, but Nexiv_Der has simplified this and made it more reliable. The second purpose of this handler is to infect floppies. When a request for the boot sector of drive A or B occurs, the virus checks the floppy type (it infects only 1.44MB diskettes), and compares the second byte of the sector with 45h – this will show if it is already infected. If it is not, the virus inserts JMP NEAR 47; NOP at the start of the sector and writes nine unchanging bytes to offset 47h.

Next, the virus generates a polymorphic decryptor, writing it into the boot sector after the nine constant bytes. As the polymorphic generator sometimes produces more code than can fit there, the virus changes the last word of the buffer to contain the marker bytes 55AAh. Despite this, these sectors usually function correctly. The new boot sector is written to floppy; the rest of the non-encrypted virus body to track 79.

The result is a boot infection which, whilst not completely polymorphic, contains insufficient distinctive constant code to be useful in a scan.

Int 21h Handler

In addition to watching for 'Are You There?' calls, the new Int 21h handler watches for program execution by trapping Int 21h, AX=4B00h. When such a call is made, Nexiv_Der opens the victim file, takes a copy of its first 32 bytes, and checks the file type. If it is an EXE file, infection aborts: only COM files are hit.

Next the virus checks whether the file is infected – if so, the seconds field of the time stamp is already set to 7. In this case, infection aborts. Then the virus checks the file size: a file will be infected only if 1000-55000 bytes long inclusive.

Now, tracing starts. The virus hooks Int 3h (Break Point) and re-hooks Int 13h, temporarily replacing its primary Int 3h handle with a new one. Then the victim file is closed, and the virus resets the disk system and lets execution continue.

Stepping Out

This Int 13h handler waits for a sector to be read. It will not have long to wait, as DOS will be ready to start executing the victim file, which will clearly involve loading it from the disk. When this happens, the virus compares the first 32 bytes to those saved by the virus' Int 21h handler previously. If they are the same, the first byte of the buffer is overwritten with an Int 3h instruction – as DOS is about to execute the file, the first instruction it will now see is a break point, which will trigger the virus' Int 3h handler.

This handler first hooks Int 1h (Single Step), and resets the Int 3h vector to its original state. The temporary Int 13h hook is also removed. Next, the virus generates a random number N between 256 and 2048. Finally, the trace flag of the processor is turned on, causing the Int 1h handler to be called after execution of every instruction.

The Int 1h handler allows program execution to continue for N instructions, where N is the random number previously chosen. If execution stops before N instructions are executed, the file is not infected, meaning very simple programs (such as goat files) will not be hit. If execution moves to a segment different from that of the victim file's code segment, it is simply not counted. This way, interrupt handlers do not affect the number of instructions the virus will trace.

When N instructions have been traced, the virus checks if the last executed instruction was CALL xxxx; JMP xxxx; ADD <8 bit register>, xx; or OR <8 bit register>, xx (all of which take three bytes). If so, the virus saves it, overwriting it with a call to the end of the victim file. If not, it does not infect, although it might the next time the file is run: N will be different. Such instructions occur often in normal programs.

Next the virus calls its polymorphic engine and writes a decryption routine and an encrypted copy of its body to the end of the file. Finally the seconds field of the time stamp is set to 7, marking the file as infected.

Nexiv_Der has a critical-error handler in effect during infection to prevent write protect errors, and it clears the files attributes during infection to allow it to infect read-only, system and hidden files.

Conclusions

Nexiv_Der is a complicated virus. It has some bugs which may cause problems under multi-tasking environments, and sometimes the virus crashes of its own accord. In any case, it is a complex and interesting virus which cannot be detected with searchstrings alone. It is also not easy to detect it by emulation, and disinfection is difficult. It was received from Italy in December 1995 in a collection of new viruses: thankfully, it is not thought to be in the wild.

Nexiv_Der

Aliases:	Red Vixen.
Type:	Resident COM and DBR infector.
Infection:	COM files 1000-55000 bytes long, hard drive DBRs, 1.44MB floppy boot sectors.
Self-recognition:	Seconds field set to 7 in files, value of byte at offset 2 in boot sectors is 45h.
Hex Pattern:	No simple hex pattern is possible for files or boot sectors. In memory:

```
3D4E 3074 0A3D 004B 7406 EA??
???? ??CF 5053 5152 5657 551E
```

Intercepts:	Ints 1h, 3h, 13h, 21h, and 24h.
Payload:	None.
Removal:	Replace infected files with clean copies; replace DBRs with SYS.