VIRUS ANALYSIS 2

Un combate con el Kerñado

Peter Ferrie
Symantec Security Response, Australia

W32/Elkern could be considered the ‘little brother’ of W32/Klez. Even though Klez carries the Elkern virus and runs it on the machines that Klez infects, it is Klez that has received all the attention. Little mention is ever made of Elkern, and some of the details of its behaviour have remained unexplained. They are described here.

There are three variants of Elkern. The first, which is 3326 bytes long, is carried by Klez variants A to D, F and G; the second Elkern variant, which is 3587 bytes long, is carried by Klez.E, and the third, which is 4926 bytes long, is carried by Klez variants H to L.

Elkern.3326 and Elkern.3587

Both Elkern.3326 and Elkern.3587 can exist in two formats: as a DLL or as an executable file. When the viral code gains control for the first time, if it is loaded as an executable file, it will always run, but if it is loaded as a DLL the viral code will run only during the DLL_PROCESS_ATTACH event.

Windows N(o)T

If the code is run, Elkern will search memory for kernel32.dll and get the addresses of the APIs that it requires. The first major bug in the virus occurs here: the API names are converted to a 32-bit CRC value, but Elkern compares only the lower 16 bits of this value. This results in the retrieval of the wrong API addresses under *Windows NT*, where several of the calculated values differ only in the upper 16 bits.

This mistake has been made repeatedly by virus authors, including the author of W32/Kriz, and is likely to continue as the majority of computer users (including virus authors) skip *Windows NT* in favour of *Windows 2000* and *XP*.

If a debugger seems to be running, Elkern will stop running at this time.

Must Run, Back Soon

If Elkern was not loaded as a DLL, it will copy itself to %system% and alter the Registry. Under *Windows 9x/ME*, the filename will be ‘wqk.exe’ and the Registry entry ‘HKLM\Software\Microsoft\Windows\CurrentVersion\Run’ will contain a value called ‘WQK’, which points to %system%\wqk.exe.

Elkern will also call the RegisterServiceProcess() API, if it exists, in order to remove the Elkern process from the task list. Under *Windows 2000*/*XP*, the filename will be ‘wqk.dll’ and the Registry entry ‘HKLM\Software\Microsoft\WindowsNT\CurrentVersion\Windows’ will contain the value ‘AppInit_DLLs’, pointing to ‘wqk.dll’. Any previous data for this value are lost.

The AppInit_DLLs is an interesting value. It exists in *Windows NT*/*2000*/*XP*, and the files in the value data are loaded into the process memory of all processes that run after the Registry change has been made.

Furthermore, if the computer is rebooted, these files will load into critical system processes, such as Winlogon. This poses a problem for anti-virus software that terminates processes containing viral code: terminating the Winlogon process will cause *Windows* to display the dreaded blue screen of death.

Elkern calls the routine repeatedly to copy the file and alter the Registry, at random intervals from one to seven seconds, requiring much speed (or luck) in order to disable it successfully.

What are my Chances?

At this point, Elkern enters the loop that searches repeatedly for files to infect. Before each location is searched, Elkern will check whether the payload should activate. The payload will always activate on 13 March and 13 September, but there is a small chance that the payload will be activated regardless of the date.

Though small in isolation, the chance of payload activation is increased greatly by the repeated checking process.

Elkern begins searching for files to infect in %system% and in the current directory. Next it will search on drive letters, beginning with a random letter and continuing until it reaches Z, before resuming from A. Under *Windows 2000*/*XP*, or if the WQK file was the one that launched this code, Elkern will also enumerate open shares on the local network to find files to infect.

Fuel Injection

During the file search, Elkern will open every file, regardless of extension. If the payload has been activated, Elkern will overwrite the entire file with zeros. If the payload has not been activated, Elkern will examine the file for its potential to be infected. Files will be infected if they are at least 8 KB PE files and are neither WinZip self-extractors nor DLLs.

Elkern.3587 also avoids RAR self-extractors, and files protected by the System File Protection. The infection method is very similar to that used by W95/CIH. The viral code is split into a linked list of blocks that are placed in the unused space at the end of sections in the file.

Since Elkern is so large, it will increase the size of the last section if there is insufficient unused space available elsewhere in the file. The entry point is altered to point directly to the Elkern code. Elkern.3587 will recalculate the checksum if one existed before.

Elkern.4926

If the previous Elkerns were a brick wall, then Elkern.4926 would be a rock wall constructed without mortar. It looks like a hurried work, unfinished and fatally buggy. It exists only as an executable file infector. It contains some of the same bugs that exist in the previous Elkerns (for example, the 16-bit comparison of the CRC32 value).

Whenever Elkern.4926 is run, it alters its appearance slightly. Elkern has many subroutines that are encrypted individually, and whose keys are altered each time the subroutines are used.

Additionally, Elkern has several routines for altering the code of several other routines, however these alterations are limited to register replacement and alternative encodings of some instructions.

Dude, where's my Code?

Elkern.4926 will inject its code into the memory of certain processes. If the process enumeration functions are found, Elkern will open all processes under *Windows 2000*/*XP*, and any process whose name contains '\explorer' under *Windows 9x*/*ME*.

If the enumeration functions are not found, Elkern will attempt to open any accessible process, by cycling through 20,000 different process IDs. Once Elkern has opened a process, it will read from a fixed image base value of 0x400000. This is unusual behaviour because the true image base of a process can be retrieved using the enumeration functions.

Elkern will then search the import table of the process for a reference to 'user'. If this is found, Elkern will search a random number (0–63) of imports for either the DispatchMessageA function or the DispatchMessageW function.

Regardless of the success of the search, Elkern will hook an import. If the search was successful, Elkern will hook the last import that was examined; otherwise, it will hook the second last import that was examined. This routine is executed repeatedly, with a small delay between each run.

Elkern begins searching for files to infect in the current directory. Then it searches on drive letters, beginning with a random letter and continuing until Z is reached, before resuming from A. It will also enumerate open shares on the local network to find files to infect.

The file search will skip directories that contain 'rary Inter' or 'tem32\dllcac'. A misfeature of the name comparison algorithm is that files and directories are also skipped if they begin with certain characters, such as the letter 'n'.

Additionally, files are skipped if they begin with any of the following: _avp, aler, amon, anti, nod3, npss, nres, nsch, n32s, avwi, scan, f-st, f-pr, avp, nav.

Considering the number of names in this list that begin with 'n', it appears that the virus author is unaware of the comparison bug. Files will be examined if their suffix is .exe or .scr, but there is a small chance that files with other extensions will be examined too.

Elkern considers a file to be infectable if it is a PE GUI or console application that is not a DLL, does not contain the text 'irus', is not protected by the System File Checker that is present in *Windows 98*/*ME*/*2000*/*XP*, and is neither a WinZip nor RAR self-extractor.

There is also a process to check whether the file is already infected but, due to a bug in the virus, this check always fails. The result is that files are reinfected repeatedly, eventually becoming too large to execute.

The file infection procedure for Elkern.4926 is identical to that of the previous variants: the viral code is split into a linked list of blocks that are placed in the unused space at the end of sections in the file, and the size of the last section will be increased if there is insufficient unused space available elsewhere in the file.

If the file contains relocations near the entry point, the entry point will be altered to point directly to the Elkern code. Otherwise, Elkern will place a jump at the original entry point that will point to the Elkern code. If the host contained a checksum, Elkern will recalculate it now.

Conclusion

W32/Elkern shows how even a buggy virus can become widespread, by being associated with a virus that is even more prolific.

Fortunately, Elkern does not stand well on its own. For the moment, at least, this battle is half over.

W32/Elkern

Type:	Memory-resident parasitic appender/inserter.
Infects:	*Windows* Portable Executable files.
Payload:	Elkern.3326, and .3587 overwrite all files on 13 March and 13 September. Elkern.4926 has no payload.
Removal:	Delete infected files and restore them from backup.