VIRUS ANALYSIS

Attack of the Clones

Peter Ferrie
Symantec Security Response, Australia

Once again, old ideas have been given a new lease of life on the *Windows* platform. The idea used by W32/Gemini is, perhaps, all the more interesting because it came not from the era of MS-DOS and its variants, but from an operating system that existed a decade earlier. The author of Gemini has produced a series of 'one-of-a-kind' viruses; whatever the motive behind it, this is another in the collection of unusual techniques.

All the Merry Men

According to the background information, the first demonstration of this technique was seen in the mid-1970s on the Xerox CP-V timesharing system. It was described more recently by Péter Ször in his presentation at the *Virus Bulletin* conference in 1999, when he named the idea 'The Twins'.

The idea is that two co-dependent processes are run simultaneously, each one checking the state of the other as it runs. In the case of the Xerox CP-V implementation, each of those processes would, among other things, send console messages to the other process, the text being directed from 'Robin Hood' to 'Friar Tuck', or from 'Friar Tuck' to 'Robin Hood'.

That version elevated its privileges to supervisor level and then proceeded to interfere with the system by dismounting tapes and walking drives. W32/Gemini does not demonstrate any of these characteristics, remaining as a user mode application and having no noticeable effects, but what it does share is the active prevention of the termination of the malicious processes.

Anti-anti-anti-anti …

When Gemini is started for the first time, it decompresses and drops a standalone executable file that contains only the virus code, using a 'fixed' (taking into account the variable name of the *Windows* directory) filename and directory.

An occasional technique against such viruses is to create a directory or read-only file with the same fixed name. Unfortunately, Gemini contains code to work around both of these cases. If such a file exists there, then its read-only attribute (if any) will be removed, and the file will be deleted. If a directory exists there instead, then it will be renamed to a random name.

The decompressed file has an unusual structure – the MZ header, PE header, and the section table are overlapped and truncated. At a glance, it appears to be a file with a corrupted structure, perhaps as an anti-heuristic method, since corrupted files might not be scanned by some anti-virus software. Despite this apparent corruption, the file runs correctly on all current *Windows* platforms (*95*, *98*, *ME*, *NT*, *2000*, *XP*). This technique has been used by several viruses in the family. To create such a file would require a significant amount of trial and error … it seems that someone has a lot of time on his or her hands.

Are You Talking to Me?

Once the file has been dropped, it is executed. The code in the dropped file calculates the checksum of itself and stores it in a fixed location in the code. After the checksum has been stored, the virus creates two events with random names, one for each copy of the code that will eventually be running. These events are created in an unsignalled state, and requiring manual reset.

Then the virus runs a second copy of itself, passing these event names and the process identification number of the first process as command-line parameters. When the second process is created successfully, *Windows* returns to the first process the process identification number of the second process. This is how the virus establishes its inter-process communication.

The Eternal Cycle

In order to understand how the communication works, it is perhaps easiest to designate one process 'active' and the other process 'passive', where the process that has control at any instance is considered to be the 'active' process, and the other process is 'passive'. Note, however, that the designation is arbitrary because the control will be swapped continually between the two processes.

The active process checks the state of the event belonging to the active process. If the event in the active process has been reset by the passive process, then the event in the passive process will be reset by the active process. The code in the passive process will then be checksummed to detect tampering. The checksum algorithm is simply a sum of the bytes in the code. If the checksum matches the expected value, then the active process will set the event to a signalled state in the active process and wait for a short time for the event to signal in the passive process. If the event signals in time in the passive process, then the code will begin the checks again.

Rise Like the Phoenix

Thus, if the event in the active process was not reset by the passive process, then the virus considers that the passive process has been suspended or terminated, and will run another copy of it. However, if the code changes in the passive process, the virus will behave as though it had never been run before on that computer, and will start two copies of the code, with new event names and process identification numbers.

As *Windows* switches from one process to the other, so the currently active process becomes the passive process, and the currently passive process becomes the active process. The only way to disable the processes is to terminate both of them simultaneously (at least, within the period that the active process waits for the event to signal in the passive process); however this can be a difficult task to achieve in *Windows*.

Meanwhile, Back at the Ranch

In order to perform its actions, the virus uses the Structured Exception Handler list to gain access to KERNEL32.DLL. After gaining access to KERNEL32.DLL, the virus will retrieve the addresses of API functions that it requires, using the increasingly common CRC method to match the names.

Unlike the authors of some of the other viruses that use the CRC method, the author of this virus was aware that APIs must be stored in alphabetical order, so there is no need to search the CRC table repeatedly. Additionally, the virus has support for both ANSI and Unicode functions merged into a single routine, and selects the set of APIs that is appropriate to the current platform (ANSI for *Windows 9x* and *ME*; Unicode for *Windows NT*, *2000*, and *XP*).

In addition to the process synchronization, Gemini will create a thread that searches for files in all subdirectories on all fixed and mapped network drives. This thread is executed every ten minutes, and is run by all running copies of the code.

The file-searching algorithm is identical to the one used by the other viruses in this family, using a linked-list instead of a recursive function. This is important from the point of view of the virus author, because the virus will infect DLLs, whose stack size can be very small.

Filters

Files are examined for their potential to be infected, regardless of their suffix, and will be infected if they pass a very strict set of filters. The first of these filters is the support for the System File Checker that exists in *Windows 98/ME/2000/XP*.

The virus author was aware of the fact that the IsFileProtected() API requires a Unicode path, while directory searching on *Windows 9x* and *ME* require an ANSI path, so the virus transforms the path dynamically.

The remaining filters include the condition that the file being examined must be a character mode or GUI application for the Intel 386+ CPU, that the file must have no digital certificates, and that it must have no bytes outside of the image.

Touch and Go

When a file is found that meets the infection criteria, it will be infected. If relocation data exist at the end of the file, then the virus will move the data to a larger offset in the file, and place its code in the gap that has been created. If there are no relocation data at the end of the file, then the virus code will be placed here. The entry point is altered to point directly to the virus code.

Once the infection is complete, the virus will calculate a new file checksum, if one existed previously, then continue to search for files.

Once the file searching has finished, the virus will allow the application to exit by forcing an exception to occur. This technique appears a number of times in the virus code, and is an elegant way to reduce the code size, in addition to functioning as an effective anti-debugging method.

Since the virus has protected itself against errors by installing a Structured Exception Handler, the simulation of an error condition results in the execution of a common block of code to exit a routine. This avoids the need for separate handlers for successful and unsuccessful code completion.

Conclusion

While W32/Gemini offers nothing new in terms of its replication and infection methods, it does take a step forward in the implementation of ‘anti-anti-virus’ techniques.

As operating systems increasingly hide or remove the ability to produce and use a bootable floppy disk, so users rely increasingly on anti-virus software being run on a computer that is actively infected. The initial response to this was memory scanning and process suspension, termination, or alteration. Now we need a new response. We play the game but the rules keep changing.

W32/Gemini

Aliases:	W32/Chiton.
Type:	Memory-resident parasitic appender/inserter.
Infects:	*Windows* Portable Executable files.
Payload:	Actively prevents viral process termination.
Removal:	Delete infected files and restore them from backup.