VIRUS ANALYSIS 3

Stealth Survival

Péter Ször
Symantec Security Response, USA

Virus writers have always attempted to challenge users, virus researchers and virus scanners alike. In fact, some of today's viral techniques, such as anti-heuristics and anti-emulation, were invented by virus writers in response to virus scanners becoming increasingly powerful. However, stealth viruses appeared very early on in the history of malware.

In fact, one of the first known PC viruses, Brain (a boot virus), was a stealth virus. Brain showed the original boot sector whenever an infected sector was accessed and the virus was active in memory, hooking the disk interrupt handling. This was in the golden days when Dr Alan Solomon was almost driven to distraction in his attempt to figure out what exactly was going on.

It was not long before stealth techniques appeared in DOS file infectors too – this method enabled the virus to remain unnoticed for long enough to replicate. In the DOS days, users would memorize the sizes of system files in an attempt to apply their very own form of integrity checking. Knowing the original size of files such as command.com was half-way to the success of detecting an ongoing infection.

There are a number of different names for the stealth technique, according to how difficult it is to detect the virus and what kind of method it uses. 'Semi-stealth' viruses conceal the changes in file size, but the original content of the infected object remains visible via regular access. 'Read stealth' is a technique where the original content of the object is simulated, usually by altering seek and read functions. The technique is described as 'full stealth' when all possible access is virtualized to a certain infected object. Finally, the highly sophisticated 'hardware-level stealth' was used by the Russian virus Strange, which hooks INT 0D which corresponds to IRQ 5. In all of the stealth techniques the virus code must be resident in memory.

Whatever Happened to Win32 Stealth Viruses?

I don't know of any *Windows* users who would bother to memorize the size of notepad.exe or other such files. Who would pay attention to this information these days, when applications are typically so huge that they barely fit on a diskette? Evidently this is the primary reason why there have been only a few attempts so far to develop stealth viruses on 32-bit *Windows* systems.

Nevertheless, one of the first known Win32 viruses, Win32/Cabanas, used a semi-stealth technique (or so-called directory stealth). Cabanas tried to hook the host's access to APIs which returned file size information in order to hide file size changes. Cabanas certainly did not use a very advanced technique, but we can say that this was the first step in stealth on *Windows*.

There have been no other major attempts to write semi-stealth viruses since Cabanas was released. The fact is that semi-stealth does not make sense from the point of view of virus replication on Win32.

Read Stealth in W95/Sma

A few weeks ago a new virus, W95/Sma, was sent to me by a fellow anti-virus researcher. The virus was marked as 'interesting polymorphic', so I was very keen to look into it immediately. After a while I figured out why it seemed impossible to replicate the virus.

At first I believed that I had replicated the virus. I knew this because the size of my goat files changed on the hard disk. Next, I copied the infected files to a diskette in order to move them over to my virus research machine. To my surprise I found I had copied clean files! I repeated the procedure twice more until I started to suspect that something was just not right with W95/Sma.

Using my Windows Commander tool I looked into the file on the infection machine. Sure enough, there was nothing in the file. In fact, the file was larger, but there appeared to be nothing appended to it. Then I accessed the file on the diskette one more time. Sure enough, the size of the file changed on the diskette too. I quickly moved to my virus research system and, finally, found W95/Sma in there. Gotcha!

W95/Sma is the first known working *Windows 95* stealth virus. Previously W95/Zerg attempted a stealth technique, but the virus crashed so quickly that it did not prove difficult to detect at all.

W95/Sma does work, but there is a minor bug in its technique. The virus attempts to set the second field of the infected PE files to 4 in order to hide its size in specially marked infected files. However, the virus clears the bit that it wants to detect before it compares and thus it will always fail to hide the size change. Infected files will appear 4 KB longer.

Decrypt Slowly

W95/Sma is an oligomorphic virus. The virus does change the main entry point in infected applications. However, it places its first decryptor into a cave of the code section itself. Such a cave usually exists in PE files and the first decryptor is only a few bytes.

The first decryptor will decrypt the main virus body in the last section and jump to that point. However, there is another decryption layer in there, which will decrypt the virus body little by little. Finally, the decrypted virus body is executed. On returning from this call the original main entry point will be executed.

Sma in Kernel Land

The virus uses a call gate mechanism. It modifies the GDT to create its own descriptor entry 1F0h. From there on, Sma can execute its code in kernel mode. This will be vital, since the virus wants to hook the file system.

First, W95/Sma allocates kernel heap memory, then it copies itself there and jumps to that location. It hooks the file system so it will be able to see file access functions (SEARCH, OPEN, SEEK, etc.). In addition, the virus hooks the TCPIP service, which it uses maliciously. Eventually, control is returned to the original host.

The virus will ensure that it hooks the file system only once. Should the file system already be hooked by Sma, the previously reserved heap block will be freed properly.

Fast Infection of PE Files

On each file open the virus will check the file content and attempt to infect PE files regardless of their attributes (it clears attributes) or their extension. It modifies the pointer to the symbol table entry in the PE header, giving it a non-zero value, in order to mark the infection internally. The entry point will be modified to point to the first decryptor in the code section.

The decryptor of the virus is oligomorphic (it does not change very much, but enough to break pattern matches). The size of the image field is also altered and the virus body is placed into the last section. The last section is marked as writeable and the encrypted virus is placed into it with the additional decryption layer on top. Infected files will grow 4 KB in size. (The change remains visible in this release of the virus.)

The virus structure looks very advanced internally. This is due to the modularity of the code. The virus does not patch the usual CD 20 xx xx xx xx patterns all the time as first-generation kernel mode viruses do on *Windows 9x*. Instead, it uses a single function and calls that with passed function IDs according to its needs. This will simplify the coding and make analysis of the virus code more difficult.

Stealth

When an infected PE file that has been marked as infected is opened, the virus virtualizes the file content. In fact, it hides the changes so well that it is very difficult to see any at all.

The virus assumes zeroes for all the places where unknown data was placed. Otherwise, original content is returned for all previously modified fields of the PE headers and section headers.

Evidently, if there were no bug in the code the virus would be totally hidden from the eye. Is it? Yes and no. The virus code remains hidden from regular file _open() and _read() functions. Consequently, when an infected file is copied via such functions, the copy will, at first, appear clean from the virus.

However, we should note that W95/Sma does not hook memory mapping at all. This means that a sequence of memory mapping APIs can lead us to the proper file content! This is good news. (Although it would be relatively simple for the virus to convert to full stealth and hook such events even further.)

TCPIP

The virus hooks TCPIP in the initialization code. I had many discussions with Peter Ferrie and other researchers at *Symantec* on this matter until all the pieces fell into place and started to make sense. The virus attempts to open PORT 53357, specifying UDP, and sets up a notification request using TdiOpenAddress() and an event handler for TDI_EVENT_RECEIVE_DATAGRAM with TdiSetEvent() using kernel mode functions.

The idea is to execute code that is received via such broadcast in kernel mode. Memory is allocated for the incoming data, then _VWIN32_CreateRing0Thread() is used to run the content that is received.

Such a technique could be used for many malicious reasons, including (but not limited to) DDoS attacks as well as backdoor features. The 'NetSt0rm 1.0, G.7 (c) Smash Inc.' string in the virus suggests that this is a version 1.0 release of the code. 'G.7' could be the sub version number of the virus – it definitely appears to be an early release of the code.

Conclusion

The stealth technique has finally arrived on *Windows 9x*. The next step for the dark side will be stealth on *Windows NT/2000/XP*. We can only hope that this happens later rather than sooner.

It seems that the merging of the malicious hacker and virus writer knowledge base is continuing to produce increasingly sophisticated attacks.

W95/Sma

Type:	Fast PE infector, oligomorphic, stealth.
Size:	4096 bytes.
Detection string:	Not possible, the virus is oligomorphic.