Why won't my sample run?

nakedsecurity.sophos.com/2010/07/24/sample-run/

By James Wyke 24 Jul 2010

Here at SophosLabs we have recently been seeing samples of Zbot (also known as the Zeus crimeware kit) that refuse to execute on any of our testing machines.

Often when this happens it is because the sample is corrupt or will only execute on specific versions of Windows, or maybe because the file will only run on a specific date, or because a certain payload is only activated on a certain date (e.g. CIH).

However, these Zbot samples have been crafted to ensure that they only work when executed on one specific machine and from one specific path. Any attempt to execute the sample on a different machine or from a different path will result in early termination of the malware and no impact on the target system.

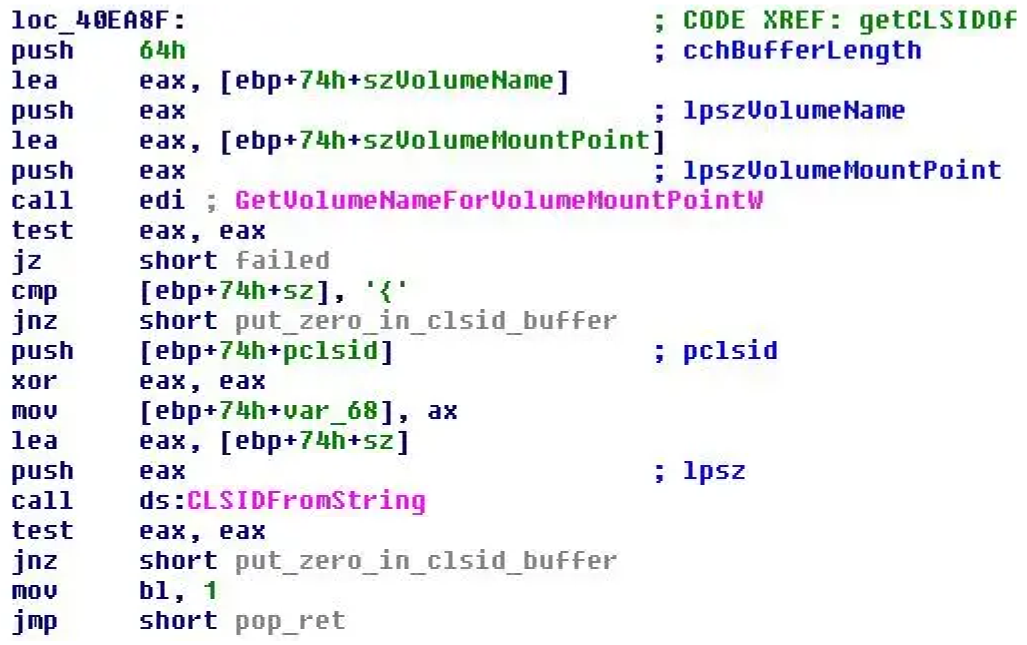
This is achieved through a form of hardware based digital watermarking that makes dymanic analysis of the sample effectively impossible for AV researchers.

Older versions of Zbot (pre version 2.0), when first installed would copy their executable to a fixed location (%SYSTEM%\sdra64.exe), sometimes appending random amounts of data to the end of the file to avoid checksum based detections. Version 2 creates a new file with a random file name inside a new folder under the user's %APPDATA% directory. It then deletes the original file with a batch script.

The new file is almost identical to the original file except for a small block of encrypted data at the start of the ".data" section. This block contains the hardware and pathname information that ties the sample's successful execution to one location on one machine.

The block contains several key pieces of information including:

- A string that includes information from the Computer Name and DWORD values generated using the OS install date and product key.
- A GUID generated using GetVolumeNameForVolumeMountPoint and CLSIDFromString.
- The randomly named directory and exe file that the new file will be dropped to.



```
loc_40EA8F:                              ; CODE XREF: getCLSIDOf
push    64h                              ; cchBufferLength
lea     eax, [ebp+74h+szVolumeName]
push    eax                              ; lpszVolumeName
lea     eax, [ebp+74h+szVolumeMountPoint]
push    eax                              ; lpszVolumeMountPoint
call    edi ; GetVolumeNameForVolumeMountPointW
test    eax, eax
jz      short failed
cmp     [ebp+74h+sz], '{'
jnz     short put_zero_in_clsid_buffer
push    [ebp+74h+pclsid]                 ; pclsid
xor     eax, eax
mov     [ebp+74h+var_68], ax
lea     eax, [ebp+74h+sz]
push    eax                              ; lpsz
call    ds:CLSIDFromString
test    eax, eax
jnz     short put_zero_in_clsid_buffer
mov     bl, 1
jmp     short pop_ret
```

The block is then encrypted using RC4 and embedded into the new file which is written to disk and executed. When the new file is executed it decrypts the block, re-computes the GUID based on the information from the machine it is now running on, compares it to the decrypted value and exits if they differ. The current path of the executable is then also checked against the decrypted path information from the block.

So when the malware sample is discovered on the machine and sent off for analysis it will be executed on a new machine and generate a new GUID based on different hardware and OS information, which will fail the comparison and result in a sample that does nothing, causing AV researchers to scratch their heads and wonder what’s going on.

This sophisticated technique is very similar to hardware based licensing systems employed by major software companies to protect their products from piracy. But until now I had not seen the technique used to protect malware binaries from analysis.

Fortunately Sophos customers are protected from Mal/Zbot-U.