CNIT 58100 CFM: CYBERFORENSICS OF MALWARE – LAB 10

Ibrahim Waziri Jr
PhD in Information Security (CERIAS)
Lab 10

Instructor: **Associate Prof Sam Liles**
Purdue University
2014

Abstract

This lab covers the skills discussed in chapter 10 of the text. The practice covered in these labs is all based on malware analysis. The malware files used are provided as an extension of the text for practical purposes.

Each of the labs consists of multiple questions that require short answers. Depending on the question, certain special tools might be required to fully analyze the malware and find answers to the question.
This paper provides answers to Chapter 10 labs. The lab uses 3 different files which are: *Lab10-01.exe, Lab10-02.exe and Lab10-03.exe.* These files are malwares are therefore could be harmful if used for non-training purposes.

The lab is analyzed using the kernel debugging tool WinDbg

Lab 10-1

1. Running the program shows that the only call to write to the registry is to RegSetValue for the value HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed. As shown in figure 1 below, some indirect changes are made by the calls to CreateServiceA, but this program also makes direct changes to the registry from the kernel that go undetected by procmon.

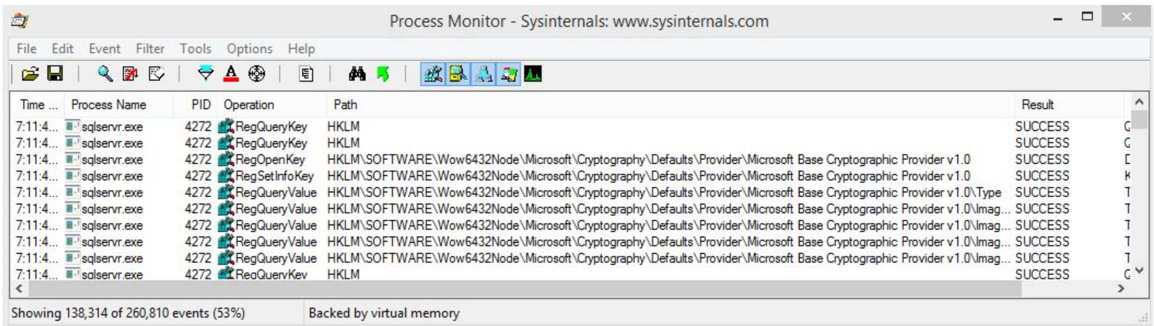
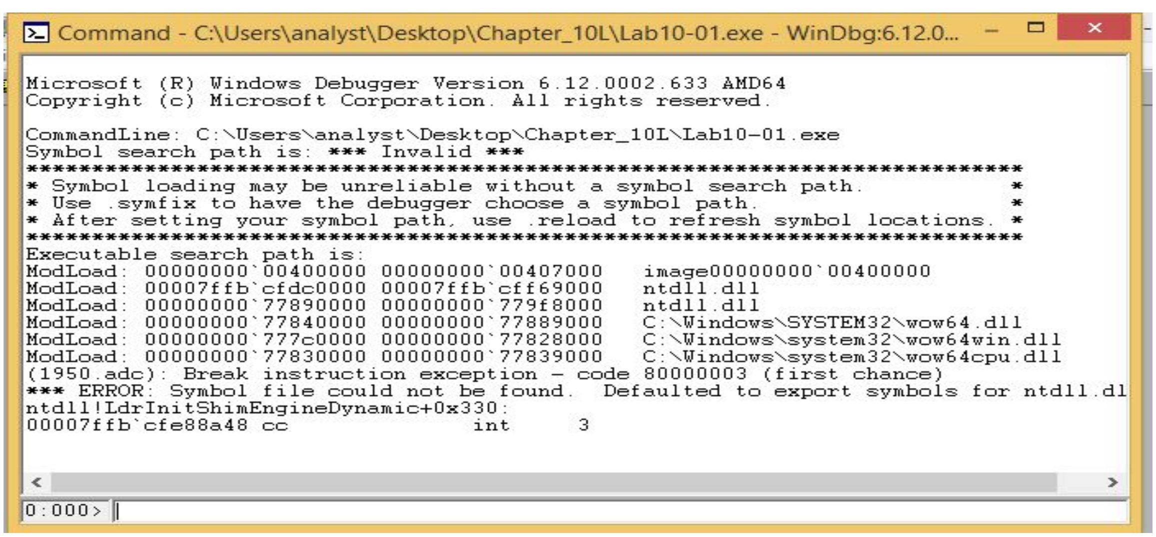


Figure 1: Procmon

Further analysis shows that the strings look a lot like registry keys, except that they all start with \Registry\Machine, instead of one of the usual registry root keys, such as HKLM. When accessing the registry from the kernel, the prefix \Registry\Machine is equivalent to accessing HKEY_LOCAL_MACHINE from another program.

2. When we try to open this program using WinDbg, an error was encountered as shown in figure 2 below:



3. To overcome this problem, we had to open the executable within another instance of WinDbg running on a virtual machine while also debugging the kernel with another instance of WinDbg in the host machine. Also using PEExplorer we can see the result below:

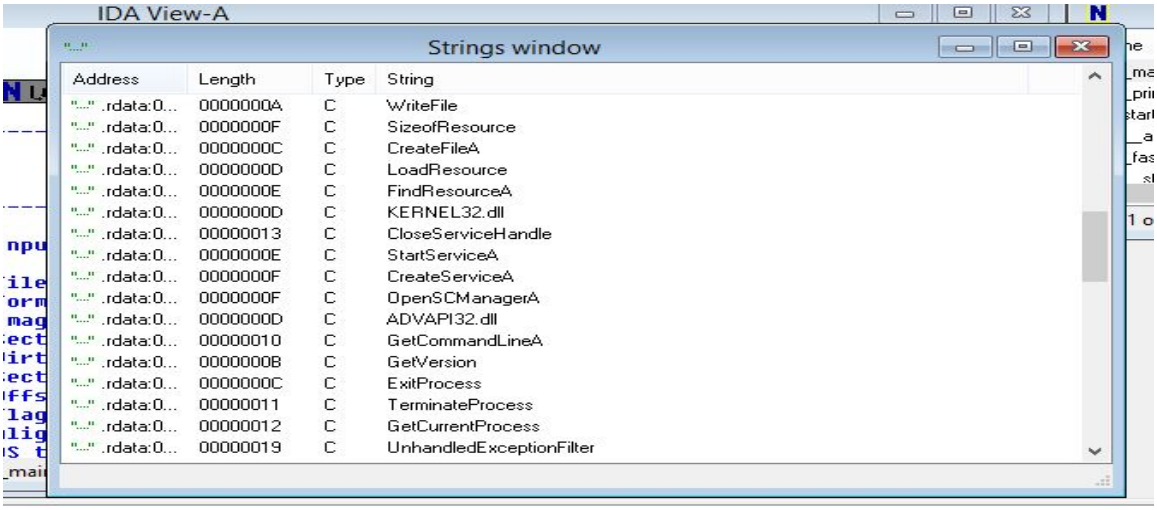
```
dt _DRIVER_OBJECT 89736f38
_DRIVER_OBJECT
+0x000 Type             : 0n4
+0x002 Size             : 0n168
+0x004 DeviceObject     : (null)
+0x008 Flags            : 0x12
+0x00c DriverStart      : 0xba72d000 Void
+0x010 DriverSize       : 0xe80
+0x014 DriverSection    : 0x89914bf8 Void
+0x018 DriverExtension  : 0x89736fe0 _DRIVER_EXTENSION
+0x01c DriverName       : _UNICODE_STRING "\Driver\Lab10-01"
+0x024 HardwareDatabase : 0x80671a60 _UNICODE_STRING "\REGISTRY\MACHINE\HARDWARE\DESCRIPTION\SYSTEM"
+0x028 FastIoDispatch   : (null)
+0x02c DriverInit       : 0xba72d959     long  +0
+0x030 DriverStartIo    : (null)
+0x034 DriverUnload     : 0xba72d486     void  +0
+0x038 MajorFunction    : [28] 0x804f35a4     long  nt!IopInvalidDeviceRequest+0
bp 0xba72d486
g
```

Using the information in figure above tells us which driver will be loaded and using that as a break point we can step over the code. Doing so reveals 3 key registry's related to firewall settings.

4. From the strings and registry's seen above, we can tell that the program creates a service to load a driver. The driver code then creates the registry keys \Registry\Machine\SOFTWARE\Policies\Microsoft\WindowsFirewall\StandardProfile and \Registry\Machine\SOFTWARE\Policies\Microsoft\WindowsFirewall\DomainProfile. These registry keys doesn't seem to change anything to the Windows Firewall even though that is what is designed to. (This could be as a result of Windows Version we are using, which Windows 8.1).

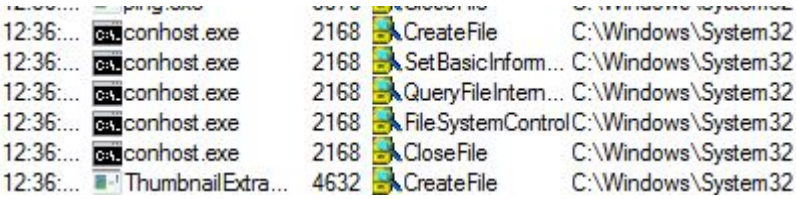
Lab 10-2

1. Running at this program, looking at the imports section in IDAPro, as shown in Figure below:



We see imports like: CloseServiceHandle, CreateServiceA, OpenSCManagerA and StartServiceA which tells us the program will create and start a service. Considering the program also calls CreateFile and WriteFile, we know that it will write to a file at some point.

2. Running the program, and using procmon, we see that the program creates a file in C:\Windows\System32 as shown in Figure



Checking the directory we couldn't find the file, however we believe that the file is a kernel as shown in the imports, which is KERNEL32.dll, the file could be use as the executable and be loaded by the OS. From procmon we can also see that another driver "486 WS Driver" is loaded.

3. We attempted to see what the program does, however, using cmd to query the drivers the file creates doesn't show any result as shown in Figure below:

It really doesn't make any sense, because the driver is running but we cant find it on the computer. (This could be as a result of the OS version we are using).

Lab 10-3

1. The program seems to do the same function as that of Lab10-02
2. Nothing seems to stop the program from running, not process explorer or anything. However a restart seems to work, because it deletes the file.
3. When we run the program we don't see any changes. Like that of Lab10-02, the kernel component doesn't create any files or change anything.

Conclusion

These labs aim to teach how to use the kernel debugging tool Windbg. Upon completion of these labs we see that the tool Windbg can show if the files are infected or packed, the tool also showed the resources on the system that is being utilized.