CMSC 491/691 Malware Analysis HW 5

Name: Daniel Roh
Assigned: 4/6/2020
Due: 4/15/2020 by 5:00pm *(extended to 4/16/2020)*

How to turn this in for grading: You can edit your answers right into this file. Submit the homework a .docx or .pdf to RJ via email.

Hint: Chapters 9 and 11 of your Practical Malware Analysis textbook are useful references! Also, the "Malware Behavior and the Windows API" PowerPoint will be helpful.

Download hw5.7z onto your Flare VM and extract it. The password is "infected". Place your Flare VM into Internal Network mode. Open hw5.exe in Ghidra and **run Immunity Debugger as Administrator**, as the malware requires Administrator privileges to run properly. Take a snapshot of your VM when your tools are set up because you will likely need to revert multiple times.

1) **What is the value of the first argument to the call to CreateFileA at 0x4012AA? What is the desired access mode? (Expecting the name of the access mode, not an integer) (5 pts)**
   The value of the first argument is: C:\Windows\d3dx1a.dll
   The desired access mode is: Generic_Read:Generic_Write

2) **What is the value of the first argument to the call to CreateFileA at 0x40135E? What is the desired access mode? (Expecting the name of the access mode, not an integer) (5 pts)**
   The first argument is: C:\Windows\system32\cmd.exe
   The desired access mode is: Generic_Read

3) **Describe in detail what is happening from 0x40136A to 0x401395. Why is the malware doing this? (12 pts)**
   The malware is taking cmd.exe and copying the file creation times from cmd.exe to the file it just created.

   The malware is doing this to make it seem that the file it created was part of the windows installation by making it look like it was installed when cmd.exe was installed.

4) **What is the value of the second argument to the call to SetFileAttributesA at 0x4013A7? What is the name of the file that SetFileAttributesA modifies, and why is this significant? (Expecting the names of the attributes, not an integer) (10 pts)**
   The second argument that is passed is: ReadOnly:Hidden:System
   The name of of the file that this instruction modifies is: d3dx1a.dll

   This is significant because the file that was created by the malware has its attributes changed to make the file hidden from normal view and sets the file marked as a operating system file. With this change, it will be hard for a normal user to detect this file existing on the computer and if it was to be found. At first glance it would make it appear to be a legitimate windows dll.

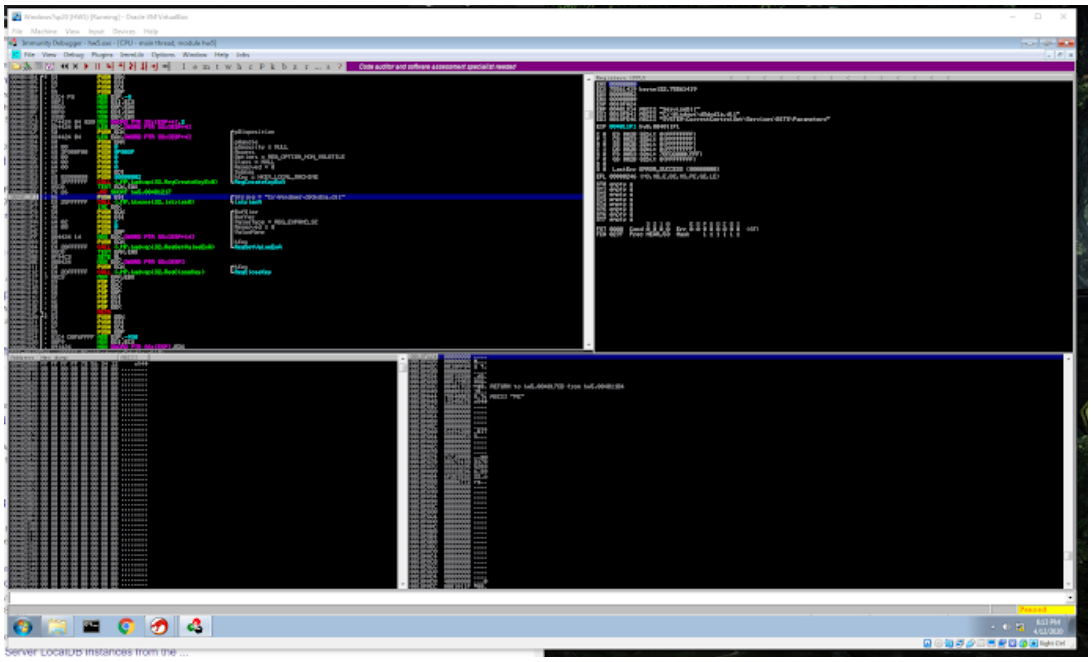
5) **In a few sentences, describe what FUN_4011B4 does. What are param_1, param_2, and param_3 used for? (12 pts)**
Fun_4011B4 creates or opens an existing registry key and with the file path of param_1. In the case that the registry key was successfully created, the registry key is modified by adding a new parameter param_2 and the data for param_2 is param_3

param_1: Path and/or name of the registry key to create
param_2: Name of the value set in the registry key
param_3: The string/data that is going to be stored in the registry key

6) **What is the value of the <u>second</u> argument to RegCreateKeyExA when FUN_4011b4 is called at 0x4017C8? (6 pts)**
The second argument is: ServiceDll

7) **The call to RegCreateKeyExA in question 6 fails and returns an integer that corresponds to an error code. What is this error code? (5 pts)**
The error code is: Access is denied

8) **Because the call to RegCreateKeyExA in question 6 fails, the JNZ at 0x4011EF will jump to 0x401217. Use Immunity Debugger to force the program not to take this jump. Describe how you did this and provide a screenshot of Immunity Debugger with execution paused at 0x4011F1 (10 pts).**
Within the immunity debugger, once I reached the TEST EAX, EAX. I double clicked on the EAX register and changed the register value to 0 to make the program think that RegCreateKeyExA successfully ran.



9) **Describe in detail what is happening from 0x401857 to 0x40188a. Why is the malware doing this? (12 pts)**
The string “cmd /c del” is being copied into a local variable then the same local variable is being appended to the front of the string “C:\Users\student\Downlo”1\hw5\hw5.exe”.

The malware is deleting the program that was used to create the changes to the registry to avoid
allowing the user to find the program that was used. This is likely to avoid an easy way to figure out what the malware did to the system and to avoid analysis of the original program.

10) **What is unusual about the assembly code from 0x4019AA to 0x4019C1? Why might the malware be doing this? (8 pts)**
The jump condition for the in this code both jumps to 0x0040198C. Normally a jump condition would go to two different locations rather than the same one. Also, this is unusual because ghidra does not show this part code in the disassembly but this is likely due to the NOP instruction causing this issue.

The malware is likely doing this to break the disassembly algorithm to avoid disassembly and allowing a person to see what the malware is actually doing.

11) **Investigate the file created in question 1 (Hint: keep in mind the attributes of this file when searching for it). List 3 malicious behaviors that this file can perform. Justify your answers and how you found them. <u>Up to 5 points of extra credit will be awarded if very good analysis is provided.</u> (15 pts)**

a) Looking at the function FUN_00405d90, this function gives the ability to create a socket connection. This can be malicious behavior as a socket connection can set open to allow an attacker access to the system at a later time. Given that virustotal gives an indication that this file is a file that creates a backdoor. I would bet that this socket connection ability is how this file creates the back door.

b) Looking further down in FUN_00405d90, there is a switch statement that is made at 00405f43. In this switch statement, the file shows that it has the ability to modify services as well as the ability to create threads, directories, and files. It also shows the ability to move files. This is likely these are the commands that are able to be made from the socket. This in my mind would mean that these commands will form a type of remote ssh/desktop to allow for the back door to work.

c) After looking around at some of the functions in FUN_00405d90, I found the function FUN_00403e80 called by FUN_0040404c from FUN_00405d90, this function calls the winbase.h functions dealing with the privileges of a process. The function here appears to handle privilege escalation to allow for gaining things like “SeDebugPrivilege” from instruction 0040411f which essentially gives admin. Given the previous idea that this code is being used for a back door, the ability to gain admin rights would make sense as an attacker essentially be able to take control of the system.