Assignment 5

Justin Kennelly

University Of Arizona

CYBV 454 MALWARE THREATS & ANALYSIS

Windows-Based Malware

Professor Galde

April 04, 2022

LAB 7-1

* Lab07-01.exe : MD5 Checksum Value: c04fd8d9198095192e7d55345966da2e

# LAB 7-1 Static & Dynamic Analysis

Table

Description automatically generated

*Figure 1: Imports used by Lab07-01.exe in PEview*

*Table

Description automatically generated*

*Figure 2: Continued Imports used by Lab07-01.exe in PEview*

*Text

Description automatically generated*

*Figure 3: Strings returned from Lab07-01.exe*

*Graphical user interface, text

Description automatically generated*

*Figure 4: Continued Strings returned from Lab07-01.exe*

*Text

Description automatically generated*

*Figure 5: Monitoring Lab07-01.exe while running through ProcMon*

*Table

Description automatically generated*

*Figure 6: Continued Monitoring Lab07-01.exe while running through ProcMon*

*Text

Description automatically generated*

*Figure 7: FakeNet Window while running Lab07-01.exe*

*Text

Description automatically generated*

*Figure 8: RegShot data after running Lab07-01.exe*

Before anything else I begin analyzing Lab07-01.exe with static analysis. I begin by opening the file in PEView to look at its’ import and export data. There are three DLL’s being imported: ADVAPI32.dll, KERNEL32.dll, and WININET.dll. There are three ADVAPI32 functions used in this program but only two look all that interesting. CreateService creates a service that could be used at boot time allowing a piece of malware persistence and stealth. StartServiceCtrlDispatcher is used by services to connect together a service control manager to the main thread of a process. This is a fairly common functionality of a lot of programs but if it is found in malware it tells you that the malware runs as a service (Figure 1). Within KERNEL32.dll, the function CreateMutexA is one that can be used by malware to create a mutual exclusion object to make sure that only one instance of the program is running on a system. GetModuleFilename is used by malware to manipulate files on a system, paired with the function WriteFile found in the PE imports. Lastly, another interesting function is SetWaitableTimer which activates a timer, when the timer is signaled the thread that creates the timer will then run a routine (Figure 1). The WININET.dll functions: InternetOpenURL and InternetOpen tell me that this program has network functionality and potentially reaches out to a website (Figure 2).

Next, I ran the strings command on the file Lab07-01.exe. In figure 3, I find many errors related strings such as DOMAIN error, SING error, TLOSS error, runtime error, unable to initialize heap, unable to open console device, etc. I assume these errors will become clearer when I dive deeper into the programs code (Figure 3). Toward the end of the strings however, I see some strings that are interesting to me now such as MalService and HGL345. Knowing that this program has the potential to create services I will keep an eye out for these strings. Additionally, there is a string for a URL: <http://www.malwareanalysisbook.com> and Internet Explorer 8.0 which back up our theory that this file may reach out to a website and use network functionality (Figure 4).

Moving onto Dynamic analysis, I prepare my lab and start to run the program. I setup the ProcMon application to focus on the Lab07\_01.exe file. When the program begins to run I immediately get a flood of TCP sends and Disconnects (Figure 6). Verifying that this program is reaching out to the internet to 127.0.0.1 port 80 (HTTP). Closing the program, I see that a files are closed and a process occurs called ThreadCreate (Figure 5). While running the program I also had FakeNet running to simulate an internet connection. The data from FakeNet also confirms that the program is making requests to our URL found in strings as well as using Internet Explorer 8.0 as the user agent, also a string found previously (Figure 7). Lastly, I took a shot of the registry before running Lab07-01.exe and after then compared the too using RegShot. Figure 8, shows many registry keys that were deleted and added as well as values that were added during the program runtime. The keys added that stick out the most are: HKLM\SYSTEM\ControlSet\Services\MalService (Figure 8). This is the string I have found previously and it also tells me that this is most likely the service that is created by the program.

# LAB 7-1

## LAB 7-1 Question 1

How does this program ensure that it continues running (achieves persistence) when the computer is restarted?

Text

Description automatically generated

*Figure 9: The start of the main function of Lab07-01.exe*

Text

Description automatically generated

*Figure 10: Closer look at the StartServiceCtrlDispatcherA function*

*Text

Description automatically generated*

*Figure 11: Start of Sub\_401040*

*Table

Description automatically generated with low confidence*

*Figure 12: a look at loc\_401064*

Starting with the main function we see that a service table is created to hold entries then a function call to StartServiceDispatcherA is called (Figure 9). The parameters of StartServiceDispatcher are seen in Figure 10 which is a pointer to the service table. Next a call is made to sub\_401040. Here we see a familiar string, “HGL345” (Figure 11). We can conclude that HGL345 is being used as a mutex thread handle due to: the name HGL345, inheritance, and access being pushed to the stack and then called by OpenMutex (Figure 11). The data is then compared and conditionally jumped to Sub\_401064 where the mutex is created by the function call CreateMutex and then OpenSCManager is called to establish the connection to the service manager on the computer and the database (Figure 12) allowing the manager to modify services on the computer. By using this functionality of establishing the mutex handle and then testing it with eax,eax before continuing, the program is able to ensure that only one of these mutex handles is running on the system. If the there was already a handle called HGL345 running at the time of this executing then it would be caught at this point and would exit. Moving on we see a call to GetFilename which gets a path to the process running which is then passed to the function CreateService which takes the filename, size, module, pathname, and dependencies to create the service called MalService (Figure 12). By going through this process and creating a service on the target machine, the malware is able to achieve persistence and continue to run every time the computer is booted.

## LAB 7-1 Question 2

Why does this program use a mutex?

As seen in Figure 11, a mutex with a specific handle of HGL345 is compared so that it can determine if a handle of that name is already running on the target computer. If that handle is already running then the program will exit. If the handle does not yet exist then the process begins for the program to create the service MalService to persistently run on the target machine (Figure 12).

## LAB 7-1 Question 3

What is a good host-based signature to use for detecting this program?

Graphical user interface, text

Description automatically generated

*Figure 13: Detailed look at Lab07-01.exe strings containing Host-Based signatures*

The mutex HGL345 and the service MalService are both strong host-based signatures that could be used to identify this malware running on a system. MalService can be identified in the registry keys created by the malicious program as well.

## LAB 7-1 Question 4

What is a good network-based signature for detecting this malware?

Graphical user interface, text, application

Description automatically generated

*Figure 14: A detailed look at network-based strings found in Lab07-01.exe*

*Text

Description automatically generated*

*Figure 15: A detailed look at data retrieved from running the program Lab07-01.exe and FakeNet*

*Graphical user interface, text

Description automatically generated with medium confidence*

*Figure 16: Create Thread Function in IDA Pro*

*Graphical user interface, text

Description automatically generated*

*Figure 17: Start address function found inside the create thread function*

Network-based indicators of this malware would be the URL <http://www.malwareanalysisbook.com> and the user-agent Internet Explorer 8.0 (Figure 14 & 15). Looking deeper in the code using IDA pro we see that after the service is called a conditional jump leads to a loop at loc\_401126. We can tell that the esi register has the value 14h pushed onto it. Right-clicking on 14h we can convert it to the decimal number 20 (Figure 16). In this loop a function is called by the name of StartAddress, looking at that function we see that our user agent is set to Internet Explorer 8.0 and a call to InternetOpen and InternetOpenURL is called to open our discovered URL (Figure 17). This process completes 20 times, we know this because the esi is set to 20 and looking back at loc\_401126 we see that the esi is decremented after every CreateThread call but if we look at the condition in loc\_40116D we see that it does not have a condition it jumps back to loc\_40116D each time so it appears to be an infinite loop opening the web address (Figure 17). Internet Explorer 8.0 and the URL <http://www/malwareanalysisbook.com> are host-based indicators of compromise for this malware.

## LAB 7-1 Question 5

What is the purpose of this program?

Text

Description automatically generated

*Figure 18: Timer functions seen in IDA Pro*

*Text

Description automatically generated*

*Figure 19: Detailed look at the timer values*

When the program is ran on a target machine it will open the handle HGL345. If the handle already exits on the system then the program will exit. However, if the handle HGL345 does not exist then it will create a mutex with that handle name and establish a connection between the service control manager and open the database. Next, the program will create a service on the target machine called MalService which will have persistence and continue to run each time the system is started. Next the program makes several calls to the system setting a routine that is triggered by a waitable timer. The timer is set to WaitForSingleObject which appears to wait for [esp+40Ch+SystemTime.wYear], 834h (Figure 18). Right-clicking on 834h we can convert that to 2100. So it appears that the timer is set to be triggered in the year 2100. So presumably January 1, 2100 0:00:00 (Figure 19). When the timer is triggered 20 threads are created that will infintily open the web address <http://www.malwareanalysisbook.com> effectively creating a denial-of-service situation on the target machine.

## LAB 7-1 Question 6

When will this program finish executing?

This program functionality is set to never finish. The only way the program will finish is if the service, MalService is stopped and deleted manually. This also means that the victim will have to understand how to identify why this occurring on their system and look for the discovered indicators of compromise both host and network based.