CMPSC 443 Lab 8: Dynamic Analysis

# Lab Description

Note: you are supposed to use a Windows VM to finish all these labs

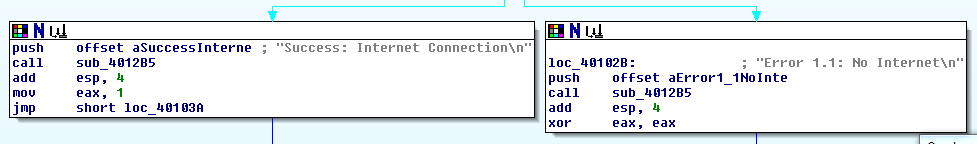
**Part I: basic dynamic analysis**

## Task 1

Analyze the malware found in the file *Lab06-04.exe*. You need to use ApateDNS to setup a DNS server that directs all DNS request to the local IP. Meanwhile, you may use python simplehttpserver (the command is “python –m SimpleHTTPServer 80”) to launch a web server that handles any HTTP request from the malware.

You need to explore the malware behavior based on the prior setup. Please provide a detailed paragraph of malware actions with screenshots for submission.

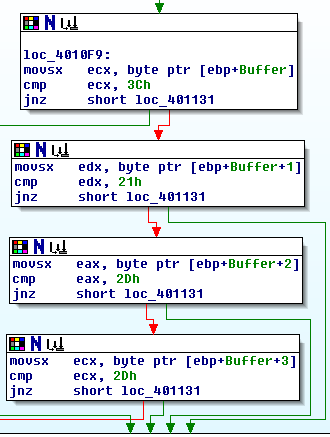
Examining *Lab06-04.exe* in IDA Pro tells us almost everything we need to know about this malware. For starters, it checks to see if it can connect to the internet:



Assuming it does, next it attempts to retrieve a file from “http://www.practicalmalwareanalysis.com/cc.htm”.

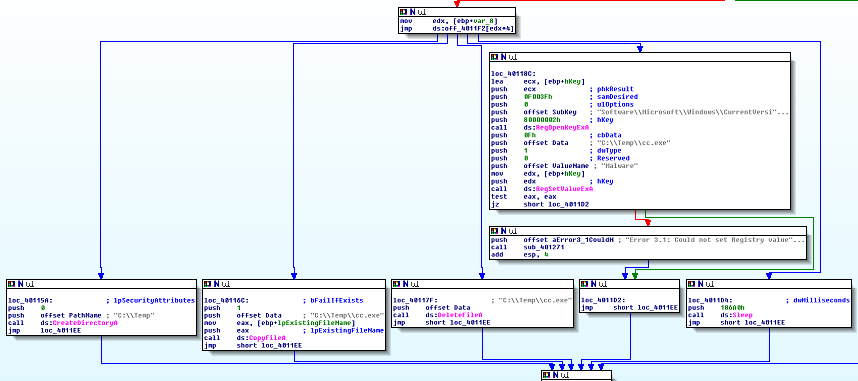
P:\CMPSC443\Git\CMPSC443\Lab08\Lab06-URL.PNG

Once again assuming that it retrieves the “.htm” file correctly, it then checks to see if the first bytes of data are “0x3C 21 2D 2D”, which correspond to “<!--", which is the beginning of an HTML document comment.



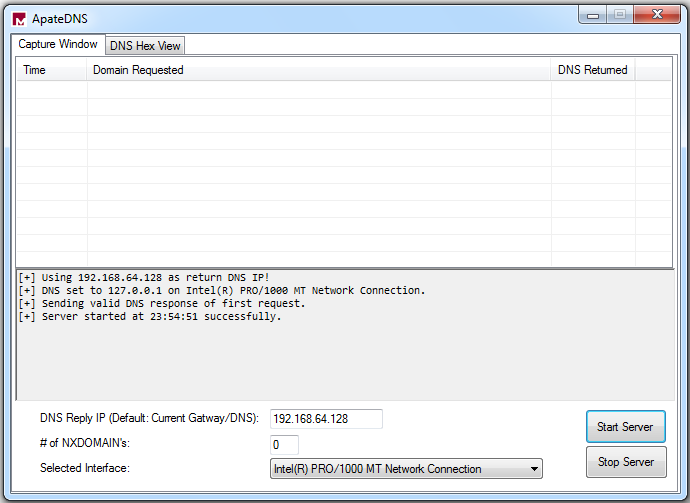
Assuming that the document starts with the above “<!--", the malware then takes the next byte of the HTML document, and subtracts 0x61 = 97 from it. 97 is coincidentally ASCII for ‘a’. As long as the resulting subtraction is less than or equal to 4 (letters that would be acceptable here are ‘a’ = 97-97 = 0, ‘b’ = 98-97 = 1, ‘c’ = 99-97 = 2, ‘d’ = 100-97 = 3, ‘e’ = 101-97 = 4), the program will jump to a memory address of that character’s respective subroutine.

Still only using IDA Pro, it seems as if ‘a’ creates a directory in “C:\Temp”, ‘b’ copies a file to “C:\Temp\cc.exe”, ‘c’ deletes the file at “C:\Temp\cc.exe”, ‘d’ sets a registry value at 'Software\Microsoft\Windows\CurrentVersion\Run', and ‘e’ sleeps for 100,000ms = 100 seconds.

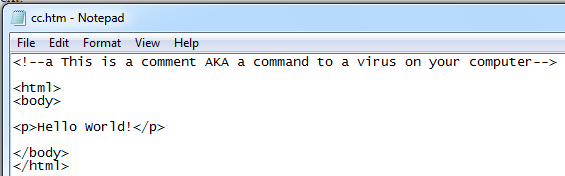


Once it’s done with the current command, it will then sleep for 0xEA60 = 60,000ms = 1 minute. It will run this loop of parsing a command and sleeping for 1 minute for 1440 times. All in total, upon execution of this program, it will run for 1440 minutes = 24 hours = 1 day.

Now, time to verify that the above hypothesis is correct. In order to do that, we will need to be able to “spoof” commands to the malware. We will do this through ApateDNS to spoof the url of “www.practicalmalwareanalysis.com” to our own computer, then by using Python to run a simple HTML server to ultimately host our own “cc.htm” file.

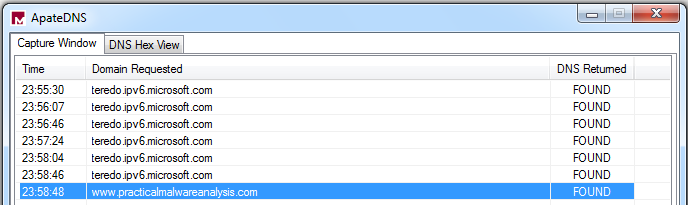


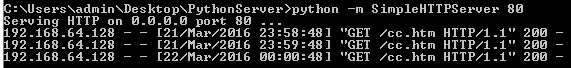
P:\CMPSC443\Git\CMPSC443\Lab08\PythonSimpleServer.PNG



Now that we have everything set up, it is time to run the malware! The results of running the malware are below:

P:\CMPSC443\Git\CMPSC443\Lab08\Lab06-04SuccessCommandA.PNG





Just as expected, “C:\Temp” now exists!



For the sake of brevity, the rest of the commands are much less in depth.

Running command ‘b’ results in “cc.exe” being copied to “C:\Temp”.



Running command ‘c’ deletes “cc.exe” in “C:\Temp”.



Lastly, command ‘d’ creates registry files and command ‘e’ sleeps for 100 seconds.

IDA Pro’s static analysis was proven correct!

**Part II: advanced dynamic analysis**

## Task 2:

Analyze the malware found in the file *Lab09-01.exe* using OllyDbg and IDA Pro to answer the following questions.

1. How can you get this malware to install itself?

The first thing this program checks for is if there is one argument. “arg\_0” represents the number of arguments and it compares this with ‘1’ to decide what to do next.

P:\CMPSC443\Git\CMPSC443\Lab08\Images\Lab09-1Arg.PNG

2. What are the command-line options for this program? What is the password requirement?

3. What are the host-based indicators of this malware?

4. What are the different actions this malware can be instructed to take via the network?

5. Are there any useful network-based signatures for this malware?

# Submission

|  |
| --- |
| * Please write a report to answer all questions. * Screenshots needed for task 1. |