# CMSC 491/691 Malware Analysis HW 1

Name: Daniel Roh
Assigned: 1/29/2020
Due: 2/5/2020 by ~~5:00pm~~ 11:59pm
*Note to self: Submit to royce8@umbc.edu as pdf*

Download and extract hw1.zip on a virtual machine. The password to the zip file is "infected". It contains hw1_1.infected and hw1_2.infected, which are malware samples. **Do NOT run them!**

Hint: Chapter 1 and Appendix A of your Practical Malware Analysis textbook are very useful references! Other parts of your textbook may be helpful as well!

**Part 1: hw1_1.infected (50 pts)**

**1) What is the MD5 of hw1_1.infected? (3 pts)**

- MD5 HASH: 0d07363187dcda999e1a6e750ed7a57a

**2) Which section in hw1_1.infected contains executable code? What is the virtual size of this section? How did you find this information? (8 pts)**

- The executable code is contained within the .text section.

- The virtual size of the .text is 65,821 bytes.

- I found this information by taking a look at the entry point given by virus total. The entry point showed a location of 16,083 which after taking a look at the virtual addresses of the executable section was after the .text address of 4096 and before the address of .rdata. This would mean that the code is being run within the .text section.

**3) What is the exact date and time (to the second) that hw1_1.infected was compiled? How did you find this information? (5 pts)**

- The compiled time is: 2017-07-04 18:19:53

- This information was found from looking under the details tab of virustotal

**4) Investigate the Windows API functions that hw1_1.infected imports. List two functions that suggest that hw1_1.infected can check whether it is being debugged. How can a malware sample use each of these functions to check whether it is being debugged? (14 pts)**

1. Windows API function: GetEnvironmentStringsW
    - This function gets the environment variables for the process. In a virtual machine, the environment variables are slightly different compared to a native running OS. So the

malware can use this function to see if its being run under a virtual machine and change its action accordingly.

**2.** Windows API function: IsDebuggerPresent

- This function checks if the program is being run by a debugger. In this case, the malware can check if it is being analyzed by a debugger and change accordingly.

**5) One of the strings in hw1_1.infected is a registry key that is commonly used to give malware persistence. What is this string? (10 pts)**

- Registry Key: "Software\Microsoft\Windows\CurrentVersion\Run"

**6) Do any of hw1_1.infected's imports suggest that it is able to connect to the internet? Do any of the strings in hw1_1.infected suggest that it is able to connect to the internet? Why is this suspicious? (10 pts)**

- None of the imports suggest that the malware is able to connect to the internet

- In the strings, the malware calls CHROME.DLL, WSOCK32.DLL, and OPERA.exe which suggests that it is trying to connect to the internet.

- This is suspicious because normally a program will import the needed functions to connect to the internet. By calling the .DLL of other programs and system functions, its a suspicion that the attacker is trying to hide any indication that the file is trying to contact the outside world. Also, the strings output seems to show that the program is trying to open a connection to upload a file containing the privileges that the malware was able to gain. Could be that if the malware was able to gain access to a system under admin rights, the attacker can later go and gain access.

<u>**Part 2: hw1_2.infected (50 pts)**</u>

**1) What is the MD5 of hw1_2.infected? (3 pts)**

- MD5 Hash: 70a2fd5bd44482de36790309079fd9ac

**2) List three indicators that hw1_2.infected is packed. What packer was used to pack hw1_2.infected? (15 pts)**

*Flare VM has a PEID program which can be used to look up the packer used for the malware. A packer is used to compress a file, but makes it unable to read.

**1.** Virustotal gives no .text, .rdata, and .data in the header section but instead gives a section UPX0 and UPX1. This gives a potential indicator that the file has been packed due to the missing/odd sections names.

**2.** Also in the UPX0 section, the virtual size shows 90112 bytes but the raw size shows 0 bytes which is an indicator that the file is packed.

3. In the UPX1 section, the entropy of the section is greater than 7 which is a indicator that the file is packed

**Name of packer:** <u>UPX 0.89.6 - 1.02 / 1.05 - 2.90 -> Markus & Laszlo</u>

**3) Unpack hw1_2.infected and describe how you unpacked it. What is the md5 of the unpacked file? (10 pts)**

- To unpack the file, I first started with the PEiD tool to find the packer type which was a upx packer. After a few hours trying to unpack the file using the PEiD/DIE tools and then seeing the demo by RJ; found out that there is an easier way by opening the cmd in the file location and running the command "upx -d hw1_2.infected"

- MD5 Hash: b6d5449653396a74b9bcffd00b28a9fe

**4) Use Resource Hacker to investigate the first resource in the unpacked file. What are the first 4 bytes of the resource? Based on these bytes, what is the type of this file? (10 pts)**

- First 4 bytes: MSCF

- The type of this file is a cabinet file.

**5) Perform additional analysis on the resource file from question 2.4. What do you think this file does and why is it malicious? Describe how you got your answer in a few sentences. (12 pts)**

- I think that this file is trying to perform a man in the middle attack so that the attacker can potentially listen in when a user is using their computer to do something that requires a secure connection.

- I got this answer due to the resource file showing an import of CryptBase.dll. In this dll, it allows applications to encode and decode data sent from or received from windows applications. By gaining access to this dll, the attacker could be trying to look into the secure information that is being sent.