Web Security and Malware Analysis

Assignment 7 – 19/04/2021

Goal: Solve each task by providing a brief explanation of the solution that you adopted. Answers should mainly include screenshots and some brief comments. Use the provided template to write your assignment.

What to do: Save the Assignment as a PDF and submit it to the E-Learning platform (**Section: Assignment 7**). The file should be named as follows: **"websec_report_7_name_surname.pdf".** Remember to include your name, surname, and matriculation number in your report.

Starting Notes: Please remember that the files you are going to deal with are real malicious samples, which could potentially destroy your machine. DO NOT EXECUTE OR OPEN THEM in your own system. ALWAYS USE A VIRTUAL MACHINE (VMware/VirtualBox) FOR EVERY OPERATION. Save a snapshot. Additionally, you could execute the tools in a virtualized environment with the Sandboxie tool (https://www.sandboxie.com/). You can find the executables for this assignment on the e-learning platform.

(password: **malwanalysis**)

Additional notes: if you have problems with Windows Defender, please follow these instructions to disable it:

https://www.windowscentral.com/how-permanently-disable-windows-defender-antivirus-windows-10

If you have problems at running IDA PRO, please install Microsoft Visual C++ Redistributable Package:

https://www.microsoft.com/en-us/download/details.aspx?id=5555

To use GHIDRA, you have to install the Java JDK at the following link:https://www.oracle.com/java/technologies/javase-jdk14-downloads.html

Task 1 – Static Malware Analysis

You are required to analyze the malware sample **Lab06-01.exe** by using the following strategy:

- Use IDA PRO to visualize the disassembly.
- Use GHIDRA to visualize the decompiled code.

However, the decompilation made by Ghidra is slightly imprecise for this sample. Try to rename functions, variables and variable types to make the decompiled output coherent with the IDA disassembler. In this way, you will practice with both IDA and GHIDRA.

(Note: You may also use IDA decompiler that works better for these samples, but stick to the GHIDRA-IDA practice).

Answer the following questions:

1. Describe the structure of the PE Executable, and report interesting information found in sections and strings.
2. What does the malware do in general?

Remember that the task is NOT asking you to make a precise code analysis. It is enough to show some "interesting" parts of the decompiled code and its equivalent Assembly routines.

Task 2 – More Static Analysis (Do Task 1 first)

By using the same strategy employed in Task 1, analyze the code contained in **Lab06-02.exe**. In particular, answer the following questions:

1. This file is similar to Lab06-01.exe. What is added? What is the new functionality of this malware sample? What are the differences between the two samples?
2. Describe the subroutine 0x401040. In particular:
   a. Show the decompiled code by renaming the variables and showing what it does. You have to report the entire decompiled code, by discussing each instruction (do not analyze further subroutines).

b. Analyze the Assembly blocks of the subroutine, describing what code constructs (e.g., nested ifs, for loops, arrays) that your found here.

Task 3 – Assembly X64 Analysis

Analyze the Assembly X64 code contained in the file x64_analysis.txt. You are required to answer the following questions:

1. Describe what each part of the code (starting with LOC…) does. **You are not required to analyze every single instruction.**
2. Describe the functionality of the code and write the equivalent program.
3. Describe the code constructs that you found in this analysis (refer to chapter 11 of the slides).

Hint: The code seems to be a lot, but some parts are repetitive. Analyze block by block, by following the flow of the program. The algorithm is actually very simple…

BONUS TASK – A strange function

Consider again the sample analyzed in Task 1. Analyze the code of the program, **starting from the main** function until **routine sub_40105F**. Are you able to understand what this routine represents? (**Warning: do not perform an in-depth analysis of the three subroutines that are called from this function**. Limit yourself to understanding what the routine does). Can you motivate your intuition? Why do you think the code of the next subroutines becomes so complex?