Web Security and Malware Analysis

Assignment 8 – 04-06/05/2020

Goal: Solve each task by providing a brief explanation of the solution that you adopted. You must use IDA PRO (for disassembling/debugging) and GHIDRA (for decompiling), or one of the indicated tools to solve the tasks (when required). Explanations should mainly include screenshots and some brief comments.

Note: Although you may find the solutions to many of the proposed tasks on the web, *try to solve them on your own and do not immediately give up*. This training will be very useful for the practical question that will be asked during the exam. Also, remember to give **brief written** answers (avoid copy-and-paste from other tutorials and try to write your own answers) and to use screenshots to describe what you do. You are also recommended not to use copy-and-paste texts from tutorials, but to use your own words.

Deadline: All assignments must be sent before taking the final exam.

What to do: Send the report in the PDF format to davide.maiorca@unica.it with the subject "[WEBSEC] - Report For Assignment 8 – NAME SURNAME" and name the file "websec_report_8_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 find the executables for this assignment on https://gofile.io/?c=3O9aWY.

(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 – Full Malware Analysis

You are required to perform a full malware analysis of the sample Lab09-01.exe. This will be the first open task of this part of the course (you will get another one in assignment 9 and one in the last assignment).

Your goal is providing as much information as you can about the piece of malware you have. This malware is much more complex in comparison to what you analysed until now and hides many "secrets". Can you find them all?

In your answer, **you must employ all the techniques learned so far, in particular**:

- Analyze the PE structure;
- Take a look at the employed strings;
- Use general static analysis with IDA/Ghidra;
- Perform dynamic analysis with off-the-shelf tools (procmon, regshot, process explorer…);
- Perform dynamic analysis with IDA (or other free debuggers as OllyDbg).

You are free to go in depth as much as you can, but be sure to motivate at least some malware behaviour by clearly showing how you obtained it.

Remember to always take snapshots of your VM before the execution. You will most likely need to restore your VM.

Some hints:

- The sample has some kind of "first installation" mechanism.
- The sample has an interesting way to hide itself after the execution.
- There is a special command-line "password" for this sample.
- The sample can interact with the network.

Good luck and have fun!

Note: if you have problems using IDA Pro 7.0, here is what you can do:

1. Switch to IDA PRO 6.8
2. Use IDA Freeware (https://www.hex-rays.com/products/ida/support/download_freeware/)
3. Used OllyDBG (http://www.ollydbg.de/ - You can find a guide for OllyDBG in Chapter 9 of the "Practical Malware Analysis" book.

Task 2 – More Static Analysis

Given the assembly X86 code contained in the file assignment_8.txt, answer the following questions:

1. Describe what each part of the code (starting with LOC…) does. As usual, you are not required to analyze every single assembly instruction.
2. Describe the functionality of the code and write the equivalent program in C or in a pseudo-language. What does the program print?

Task 3 – More Dynamic Analysis

Given the same code of the previous task, consider the stack configuration contained in the file stack_dump.txt, and the following configuration of registers (we are in a 32 bit configuration, but there registers are anyway represented as 64 bits):

```
RAX0000000000000001
RBX0000000000000001
RCX000000009CCCB056
RDX00000000007917B8
RSI0000000000792250
RDI0000000000000067
RBP000000000060FF70
RSP000000000060FEAC
RIP0000000000401630
```

Assuming that the program receives as inputs, respectively, the following values: 2, 2, 1, calculate the following registers/stack values:

1. The changes to the stack after instruction 0x401630.
2. The value of EAX after 0x401648 (hint: EAX typically stores the return value of the called function).
3. The content of ESP+0x18 after the call to 0x40165C.
4. The value of ESP at instruction 0x401680, as well as the contents of the stack at ESP+0x4 and ESP.
5. The value of ESP and the content of ESP+4 in the stack when the instruction 0x40169D is called (in both iterations). Are there any differences between the values at the two iterations? And why?
6. Consider the instruction 0x4016D6. What are the values or ESP and EAX? What is loaded inside EAX and from which address in the stack?
7. Consider the instruction 0x4016E1. When you first meet the instruction, is the jump taken? And why?

Important note: for each answer, show the complete reasoning that leads to your answer. DO NOT write the result alone.