Malware Reverse Engineering
Practical Assignment 3
Spring 2022

J.N. Wilson

March 20, 2022

Unknown Executable

You are given a piece of malware in the file *sample3.7z* in the Desktop directory of each VM. The 7zip password for that file is *infected*.

This sample is a DLL. This sample is much more complex and obfuscated than previous samples you have seen. In this assignment, your task, as a junior malware analyst, is to find out as much as you can to be able to communicate it to a senior analyst who will take over from there.

Static analysis will identify which of the DLL entry points is probably the most interesting to pursue. (Yes, it's obvious.) Do as much static analysis as you can, try to identify what sample this might be, and figure out what you can from other analyses of related software.

For dynamic analysis, you should do the following:

- Use `rundll32` to execute the promimising entry point in x32dbg. You can find out more about this from this web page about Analyzing a DLL in x64DBG.

- Recall that one way to see when code outside the main program (i.e., library code) is being executed in the debugger is to *trace into* with a *Break Condition* which is `eip <` *addr1* `|| eip >=` *addr2*, where *addr1* is the start address of the executable's text segment (which can be found from the memory map) and *addr2* is the address just 1 byte above the last byte of the text segment.

  You can use this to find out what library functions are being called when there is extreme obfuscation being employed in a program

Several things to pay attention to are any memory or heap segments that are allocated and how they might be used by the program.

You should also make a list of all functions resolved with `GetProcAddress`. If you pause just after a call to `GetProcAddress`, you will see the *function name* argument in ESP+8 (its second argument). ESP contains the return address and ESP+4 contains the DLL from which this function is being resolved. You can follow those addresses in the dump in order to see what function name is being resolved.

It may take several debugging sessions to find everything you want, but when you do, you can set breakpoints in all the interesting functions and even see what arguments are being presented to them.

Your report should take the following form:

1. Report Title Block

The report title block should include

(a) Report title

(b) Date of preparation

(c) Your name

(d) Your email address

(e) Assignment and Class

2. Executive Summary

In this case, your executive summary is for the senior analyst, not a pointy-haired boss, so you can be a tiny bit more technical. But still, save the details for the sections.

3. Static Analysis

Check for at least the following:

(a) Identify the apparent compilation date of the program.

(b) Identify any suspicious properties of the program's Imports.

(c) Identify any suspicious or relevant strings (IP addresses, urls, process names, file names, etc.).

(d) If you find anti-disassembly techniques, report them.

(e) Describe the obfuscation methods you find. You will surely be impacted by them, but identifying any interesting patterns might be helpful in developing a tool to combat them.

4. Dynamic Analysis

Check for at least the following:

(a) Interesting behaviors that occur after the malware has executed.

(b) Identify whether or not there is any networking behavior exhibited by the program and, if so, record it.

(c) Identify any registry keys created/modified by the malware.

(d) Identify any files created/modified by the malware.

(e) Identify any processes started by the malware.

(f) Identify any persistence mechanisms employed by the malware.

(g) Describe what you did to overcome the obfuscation methods the program uses.

5. Indicators of Compromise

Present host- and/or network-based indicators of infection that could be used to detect the presence of this program. Be as specific as you can. Also include a Yara rule that you have tested against all the Practical Malware Analysis .exe and .dll files to insure none of them are false positives. Provide this as text, not an image!

Rubric

Each section of your report is assigned a point total. The point totals are as follows:

(5) Report Title Block

(10) Executive Summary

(25) Static Analysis

(45) Dynamic Analysis

(10) Indicators of Compromise

(5) Excellent Reporting

Suggestions to guide your thinking

- After you look at the disassembled/decompiled entry points of the function, you will realize that there is significant obfuscation being employed. Use the tips above to support your dynamic analysis. You may be able to associate some addresses with well-defined functions. If you can, that can potentially help a senior analyst. Provide that analyst with information like this.
- If you don't list *every* function that is being resolved by GetProcAddress, you're doing it wrong. Your senior analyst wants this information.
- If you can figure out what is being stored in the any heap structures that are allocated by the program, that may be helpful. Don't let such investigations interfere with your thorough analysis of the rest of the program, though.
- Although I have no idea if it is important in this case (seriously), you want to make sure to run `x32dbg` as Administrator if you want it to have that privilege. Some malware works *much* better as administrator than a normal user.