Cole Daley

November 4, 2023

Static vs Dynamic Malware Analysis

What is static analysis?

Static analysis is a fundamental approach towards studying and understanding malware. It serves as a more basic outlook on the malware rather than looking through it with execution. That being said, there is zero execution or interaction with the malware itself. It more so observes what the possible threats are from that particular file without running it. Examples of static analysis are analyzing the file signature, finding the hash, and running it through different AV tools, finding different strings/imports/exports from within the code.

What is dynamic analysis?

Similar to static dynamic is used in its own way to monitor how that particular malware is reacting to the system. This is done by intentionally executing the file within a controlled environment to run features like IDAPro, Ghidra, Procmon, Process Explorer, Process Hacker, and more. It is important to make sure the sandbox or VM you are doing this analysis in is safe. The malware used in this lab will not harm the host system if used in the virtual machine.

CONTENT –

In this lab, you will need:

* A controlled environment (VM or Sandbox)
* Example malware files, the author used ([GitHub - Cyber366MalwareAnalysis/DDR](https://github.com/Cyber366MalwareAnalysis/DDR))
* Tools such as Process Monitor, Process Explorer, PEStudio, Regshot and more

Part 1:

Step 1: Build a simple lab environment to download the malware.

Online resources provide various different methods around building a lab environment and common ones seen are VirtualBox, or VMware. Since this lab focuses on the malware aspect, I will not head into detail on how to set up an environment. Once the environment is ready simple copy and paste the GitHub address above and download the zip file. Enter the newly downloaded file and unzip the malware you want to analyze for this scenario we are using Lab2.zip.

Step 2: Static Analysis on Lab2.exe.

The first step I will use in analyzing this malware is finding the hash value. Doing this can provide us with more information on the malware itself that we might not have known before. By going to the cmd, we can change directory to where the unzipped .exe file is located and run the command “md5sum Lab2.exe”. This will present us with the photo below.

A screenshot of a computer

Description automatically generated

Figure 1.1 – Static Hash Value

The next step is copying this hash into VirusTotal or other antivirus detection websites to see what vendors think about this file. Below I tagged the image of my results where we can see VirusTotal linking the hash correctly to “Lab2.exe”. This allows me to know I am on the right track. It also informs us of 38 potential incidents of malicious code, which is what we were looking for. Types like “Trojan” and “Riskware” are seen allowing us to confirm that this file is not safe.

A screenshot of a computer

Description automatically generated

Figure 1.2 – Antivirus Scanner Results

The following cmd command we can run is the “Strings” command. Using the command “Strings Lab2.exe” will result us with the strings found from within the code. Strings can be very beneficial to malware analysis by telling us important information about this file. For example, in the photo below with little research I found that “VirtualProtect” is used to change permissions and is commonly seen in malware to create havoc for the user. We can also get information about possible Dynamic Link Libraries like Kernal32 seen below. We know this means it is related to the Kernal operating system and has something to do with memory allocation.

A screenshot of a computer

Description automatically generated

Next, we can move on to PEiD, which is used to check if this malware is potentially obfuscated or is hiding anything that they do not want us seeing. In the scenario below we can see I add the file to the top and it is indeed packed. The code we see is “UPX”. This does not always mean malware, but most of the time it is. We can then use the UPX code with the -d command to unpack and find the hidden text. The use of PEID is shown below.

A screenshot of a computer

Description automatically generated

Listed above was how we found the use of “UPX” but now we will use the -d command to unpack the file. Once we do this we should locate if the hash value changed if the strings change and find all the other hidden information we could not see before. Below is the command of unpacking the file. Instantly I can see the hash value changed. It used to start at 32 but now it starts with 45. Plugging the new hash value into VirusTotal gives us only 33 possible incidents of malicious code which is also different from before. The peid command we ran before is now telling us “Nothing found” because there is no longer packing associated with this file. Lastly, I reran the strings command with the new unpacked file, and it unleashed so much more information.

A screenshot of a computer

Description automatically generated

These are just a few of the common practices found when utilizing static analysis. While these could come across as simple, they are effective in the initial detection for malware. Next, we will move onto dynamic analysis.

Step 3: Dynamic analysis on Lab3.exe

The first application I will mention is process explorer. In Process Explorer we want to see the process taking place in green to know it is being executed. It will then turn red as the process has been terminated. Sometimes this process can be very fast so it is hard to see inside the application. Below are examples of Lab3.exe running and then ending very quickly.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Next is, process monitor which is a tool used to help figure out what the process itself is doing once we open the application. When we run this, we get a ton of results. We can filter these results first to registry data as seen below in the first figure. Then we can sort it by files activity and once we do that, we can see the executable creating different files when it is run. It also is shown closing files as well and this can be seen in picture 2 below. There is actually zero network connections so this malware focuses more on file creation and deletion as well as registry actions.

A screenshot of a computer

Description automatically generated

Figure 1.3 – Registry Data

A screenshot of a computer

Description automatically generated

Figure 1.4 – File Creation and Deletion

Next, we will check out PeStudio. PeStudio allows us to find a lot of information about this specific file that we are looking at. When I import Lab3.exe to PeStudio and let it boot all the information it showed me two libraries being used. The first is common which was Kernal32.dll which we have seen in the past but the other was ADVAPI32.dll. I was not sure what this was but after some research it is used to store a collection of processes which allows for interaction. So, while the DLL itself is not suspicious, how it could be manipulated is suspicious.

A screenshot of a computer

Description automatically generated

The final tool we used was the Regshot application. With this application we attempt to take a clean shot first and then a shot where we run the malware and see the differences. In the picture below we can see the shot after comparison found that the malware added 12 keys and 28 values. This can be seen below. This might indicate the files it was already creating as the malware is creating different keys on the system.

A screenshot of a computer code

Description automatically generated

Step 4: Lab Summary

Throughout this lab we have gone through basic examples of static analysis and dynamic analysis. Both serve a purpose in figuring out the details of malware. Often, we will use static analysis as predecessor for dynamic analysis. This way we can work through all the above analysis to understand the malware and start disassembling.