Malware Analysis

Project Archive - Final

Final Report

Ethan Berman
Ryan Bharat
Jenny Chang

Fall 2022
CS467 - Online Capstone Project

Table of Contents

Introduction

We have all had the dreaded experience of downloading an unknown file, or following a nefarious link, only to find our computer infected with malware. Some have had the experience of encountering a situation in which a virus scan from an antivirus software does not provide the antidote needed to rectify the issue. In difficult situations like this, rather than focus on solely resolving the malware issue, what if instead we aimed to analyze the malware infection and attempt to understand its intent? In addition to analyzing the procedure which the malware adheres to, what if we were to compare various malware tools with those that are more modern and robust, to find which are best suited for analysis? Given this premise, we sought to set up a secure malware sandbox and observe malware files with various analysis tools - those outdated and unsupported and others which are more modern and robust - in order to study the function of the malicious files and how each affects the operating system.

Research Goals

The goals of this research project are to set up a malware lab, perform static and dynamic analysis on malware samples, and document the process and findings.

Experimental Approaches/Setup

Note: The following are general instructions, as each step may vary depending on software preference and/or operating system type/version.

1. Download relevant VMware/VirtualBox software (pertaining to your system)
2. Create two separate virtual environments
    a. Download the .iso files of the desired operating system(s) you want to install on each virtual environment
3. At this point, you may want to either download the desired malware analysis tools to both virtual machines, or drag/drop or copy/paste them from your host machine, as this is more difficult once we begin isolating the virtual environments.
4. After the creation of each virtual environment based on their respective .iso files, and adding the desired analysis software, our purpose is to isolate each virtual environment from the host machine.
    a. If possible, remove any drivers that are not needed for testing your desired malware files (i.e. usb/bluetooth, cd/dvd drive, sound card)
    b. Disable any antivirus (i.e. windows defender) to be able to test the malware without interference
    c. Ensure that the network adapters of your virtual machines are set to ‘internal network’ or whichever setting allows network communication between your virtual machines, but isolated from the host machine.
        i. To test this, you can ping the ip address from one virtual machine to the other to ensure you get a reply, ensuring proper connection. Additionally, you can ping both of the virtual machine’s ip addresses from the host machine to ensure that the requests timeout.
    d. Ensure that isolation settings such as ‘drag and drop’ and ‘copy and paste’ are disabled to further ensure there is no communication between host and virtual machine(s).

At this point, our virtual machines should be isolated from our host machine, and we can begin dissecting malware files with no consequence to our host machine.

Experimental Results

Basic Static Analysis

The first step in analyzing a sample of malware is to examine the properties of the executable without closely examining the code. This step is commonly referred to as basic static analysis. The goal of basic static analysis is to determine whether a file is truly malicious and gain insight into its functionality without executing it. This includes but is not limited to looking at a program's strings, checking if the program's code has been obfuscated, and examining the file's function imports and exports.

PeStudio and VirusTotal are two tools that can be used to perform basic static analysis. PeStudio is a free analysis program that provides a wealth of information about an executable's properties. PeStudio will extract a file's strings, indicate if obfuscation is present, mark functions and strings as suspicious, and much more. An important property when statically analyzing a file is looking at its hash. An executable's hash can be searched online and in malware databases such as VirusTotal for more information. VirusTotal is a useful tool for static analysis as it provides just as much information as PeStudio along with the results for that executable in a variety of different antivirus engines. VirusTotal provides additional insight such as relations to other potentially malicious files and IP addresses as well as sandbox reports.

Typically, the first step when analyzing malware is to look up its file hash in VirusTotal to see if it has already been analyzed. Users can upload files to VirusTotal for analysis, however, all files uploaded to the website can potentially be seen and downloaded by others. Advanced attackers may opt to employ custom malware that is designed for a particular target, therefore, when a malware analyst uploads that sample to VirusTotal, they can potentially alert the attackers to having been discovered. Additionally, there may be instances where files uploaded are not actually malware and could contain sensitive information. If a PDF document potentially contained malware but also had sensitive medical information, it would generally be unwise to upload this to VirusTotal since it would be a breach of certain privacy laws.

Basic Dynamic Analysis

Basic dynamic analysis looks at processes, file system, registry keys, network activities during and after malware execution. The two types of analysis tools we explored are local tools and online sandbox.

Process Monitor, Regshot, and Wireshark are local tools that should only be used on an isolated virtual machine due to the risks of infection from running malware. First, Process Monitor is a tool to capture processes and operations that occur during and after malware execution. Next, Regshot is a tool to compare registry key changes using snapshots taken before and after malware execution. Lastly, Wireshark is a tool to monitor network activities. The advantage of these local tools is that malware stays on the local machine, which is good for companies who may not want malware and private company information to become public or known to third-party vendors. However, the disadvantage is the inconvenience of having to download and operate separate tools in order for a comprehensive analysis.

Any.Run is an online sandbox that can be used on any machine with internet access. A user simply uploads a URL or file and runs the analysis. It can generate a text report listing processes, registry activities, files activities, and network activities. The advantage of Any.Run is that it is an all-in-one tool that can perform a full analysis without separate tools. The process is quick and simple because a malware lab setup is not required. There is no risk to the host machine and network. One disadvantage of Any.Run is that it requires an account using a business or school email. Also, the free edition only offers Windows 7 32 bit for its environment OS, so it may not work for malware designed for another OS. Another disadvantage is that malware becomes known to the public or third-party once uploaded online, so Any.Run is not a good tool for companies who wish to keep malware or company information private.

Advanced Static Analysis

While basic static and basic dynamic analysis can provide a high level overview of the purpose of a malware file, it fails to provide in depth information about the detailed procedure which the infected file follows. At this point, we need to have a general understanding of assembly and disassembly in order to understand how malware functions at a more precise level. Advanced static analysis is the process of using a disassembler to better view the malware files assembly code. There are a plethora of tools used for advanced static analysis, but I will be highlighting Ollydbg and Malcat with the intention of showcasing the differences between the two tools. Ollydbg has been a popular malware disassembler dating back to the early 2000’s and is no longer being developed, while Malcat is in its beta phase, actively contributed to, and is a modern disassembler that is used by IT professionals.

When comparing both Ollydbg and Malcat on installation alone, Ollydbg shows its age. Ollydbg will only work on Windows machines and can work properly on machines back to Windows 95. Malcat on the other hand, is only supported on Windows 7 (64 bit) and above. Malcat also supports various Linux distributions such as Debian and Ubuntu. Overall, if the need is to test operating systems prior to Windows 7, Ollydbg would be the best, while if your systems run on anything newer than Windows 7 or Linux, then Malcat would be the choice.

Advanced Dynamic Analysis

Advanced dynamic analysis refers to the use of a debugger to step through an executable’s assembly instructions. Two debuggers to highlight are x32dbg and x64dbg. They are very similar but are intended to debug 32-bit and 64-bit executables, respectively. Debugging malware gives us more control over the behavior. Malware authors typically employ a variety of system checks to prevent certain code from executing. The Russian-speaking hacker group REvil authored their malware to avoid systems that use Russian and related languages [1]. Malware can also perform checks to see if it is being run in a virtual machine or if there is network connectivity. We typically analyze malware in an isolated virtual environment so in order to get the code to execute, we essentially need to trick the malware into executing its malicious code. Debuggers present us with the opportunity to edit register values, patch executables, and understand at a very granular level what the program is doing. By editing the values at runtime and patching executables we can get around many anti-analysis techniques employed by malware authors.

Malware Sample 1 - Lab06-01.exe from *Practical Malware Analysis*

The first malware sample is Lab06-01.exe from *Practical Malware Analysis* [2]. To begin, Pestudio shows the malware is a 32-bit Windows console program compiled on January 31, 2011. The strings “Error 1.1: No internet” and “Success: Internet Connection” suggest that the malware checks for internet connection. In addition, the library section flags wininet.dll as suspicious since this library is used to make internet connections. The imports section flags 6 functions as suspicious, including functions that check for internet connected state, write to file, terminate process, and retrieve environment information.

Figure 1. Pestudio flags libraries and imports of Lab06-01.exe.

Next, Process Monitor shows the malware manipulates the file system and registry. Also, Regshot confirms registry key changes after the malware execution.

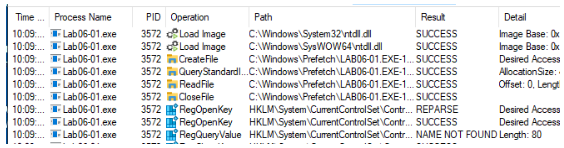


Figure 2. Process Monitor shows processes and operations from running Lab06-01.exe.

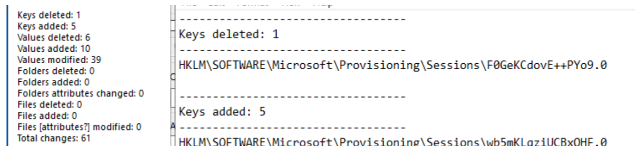


Figure 3. Regshot shows registry key changes from running Lab06-01.exe.

Lastly, Ollydbg indicates the malware calls a function at 00401000, which then calls WININET.InternetGetConnectedState. The result is stored as 0 or 1 in EAX. If the result is 0, ASCII values “Error 1.1: No Internet” are pushed and printed to the command line. If the result is 1, ASCII values “Success: Internet Connection” are pushed and printed to the command line.

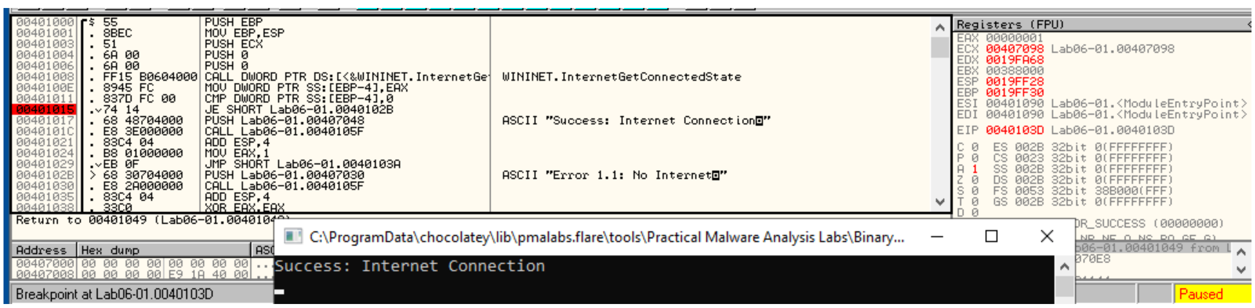


Figure 4. Ollydbg shows Lab06-01.exe calling WININET.InternetGetConnectedState and printing the result.

The findings from all the analysis tools suggest that Lab06-01.exe is a malware that checks for internet connection and prints the result to the command line.

<u>Malware Sample 2 - Lab07-03.exe from *Practical Malware Analysis*</u>

The malware sample used for advanced static analysis is Lab07-03.exe from *Practical Malware Analysis* [2]. With this sample, the purpose of the analysis will be to showcase how to use Ollydbg to disassemble the executable while utilizing some of Ollydbg’s features to gather important information about the purpose of the unknown malware file.

When opening Lab07-03.exe within the Ollydbg software, we are met with 4 separate panes that provide a plethora of information. For this example we will only focus on two panes - the CPU and Memory Dump panes on upper left and lower left, respectively.

When converting the contents in the memory dump pane to ASCII values, notice the contents (Figure 5). There exist three separate .dll ASCII strings - kernel32.dll, kerne132.dll, and Lab07-03.dll. Additionally, there are two file paths - a path to the kernel32.dll in the system32 directory, and another with the Kerne132.dll.

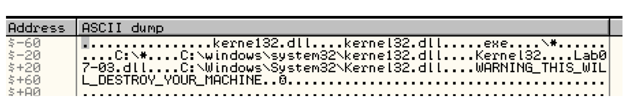


Figure 5. Contents of memory dump pane.

This causes some concern, as this can lead the one doing the analysis to believe that the purpose of this executable is to not only create a new, unknown Lab07-03.dll file, but also to replace the typical kernel32.dll file with a nefarious kerne132.dll file in the system32 directory.

When analyzing the function calls and the ASCII values in the CPU pane, it is clear that the intention of the .exe file is to replace the usual kernel32.dll file with kerne132.dll. This is shown via the string compare function _stricmp (Figure 6) to find the known kernel32.dll file.

```
0040116C  .v75 67          JNZ SHORT Lab07-03.004011D5
0040116E  . 68 20304000    PUSH Lab07-03.00403020                      s2 = "kernel32.dll"
00401173  . 53             PUSH EBX                                    s1
00401174  . FF15 64204000  CALL DWORD PTR DS:[<&MSVCRT._stricmp>]      _stricmp
0040117A  . 83C4 08        ADD ESP,8
0040117D  . 85C0           TEST EAX,EAX
0040117F  .v75 26          JNZ SHORT Lab07-03.004011A7
00401181  . 8BFB           MOV EDI,EBX
00401183  . 83C9 FF        OR ECX,FFFFFFFF
00401186  . F2:AE          REPNE SCAS BYTE PTR ES:[EDI]
00401188  . F7D1           NOT ECX
0040118A  . 8BC1           MOV EAX,ECX
0040118C  . BE 10304000    MOV ESI,Lab07-03.00403010                   ASCII "kerne132.dll"
00401191  . 8BFB           MOV EDI,EBX
00401193  . C1E9 02        SHR ECX,2
00401196  . F3:A5          REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
00401198  . 8BC8           MOV ECX,EAX
0040119A  . 83E1 03        AND ECX,3
0040119D  . F3:A4          REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
0040119F  . 8B7424 10      MOV ESI,DWORD PTR SS:[ESP+10]
004011A3  . 8B7C24 20      MOV EDI,DWORD PTR SS:[ESP+20]
004011A7  > 83C7 14        ADD EDI,14
004011AA  .^EB 96          JMP SHORT Lab07-03.00401142
```

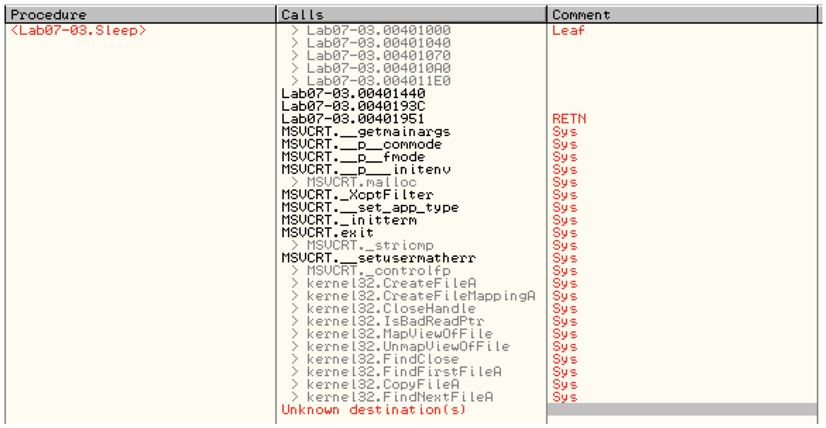
Figure 6. Disassembly showing the function call to compare strings

Further in the assembly code, we can find the creation of the file and file mapping for the kerne132.dll file. As expected, the file is created with GENERIC_ALL access, which allows full access rights (Figure 7).

```
00401495  . 8B3D 14204000  MOV EDI,DWORD PTR DS:[<&KERNEL32.CreateFileA>]          kernel32.CreateFileA
0040149B  . 50             PUSH EAX                                                hTemplateFile
0040149C  . 50             PUSH EAX                                                Attributes
0040149D  . 6A 03          PUSH 3                                                  Mode = OPEN_EXISTING
0040149F  . 50             PUSH EAX                                                pSecurity
004014A0  . 6A 01          PUSH 1                                                  ShareMode = FILE_SHARE_READ
004014A2  . 68 00000080    PUSH 80000000                                           Access = GENERIC_READ
004014A7  . 68 8C304000    PUSH Lab07-03.0040308C                                  FileName = "C:\Windows\System32\Kernel32.dll"
004014AC  . FFD7           CALL EDI                                                CreateFileA
004014AE  . 8B1D 10204000  MOV EBX,DWORD PTR DS:[<&KERNEL32.CreateFileMappingA>]   kernel32.CreateFileMappingA
004014B4  . 6A 00          PUSH 0                                                  MapName = NULL
004014B6  . 6A 00          PUSH 0                                                  MaximumSizeLow = 0
004014B8  . 6A 00          PUSH 0                                                  MaximumSizeHigh = 0
004014BA  . 6A 02          PUSH 2                                                  Protection = PAGE_READONLY
004014BC  . 6A 00          PUSH 0                                                  pSecurity = NULL
004014BE  . 50             PUSH EAX                                                hFile
004014BF  . 894424 64      MOV DWORD PTR SS:[ESP+64],EAX
004014C3  . FFD3           CALL EBX                                                CreateFileMappingA
004014C5  . 8B2D 0C204000  MOV EBP,DWORD PTR DS:[<&KERNEL32.MapViewOfFile>]        kernel32.MapViewOfFile
004014CB  . 6A 00          PUSH 0                                                  MapSize = 0
004014CD  . 6A 00          PUSH 0                                                  OffsetLow = 0
004014CF  . 6A 00          PUSH 0                                                  OffsetHigh = 0
004014D1  . 6A 04          PUSH 4                                                  AccessMode = FILE_MAP_READ
004014D3  . 50             PUSH EAX                                                hMapObject
004014D4  . FFD5           CALL EBP                                                MapViewOfFile
004014D6  . 6A 00          PUSH 0                                                  hTemplateFile = NULL
004014D8  . 6A 00          PUSH 0                                                  Attributes = 0
004014DA  . 6A 03          PUSH 3                                                  Mode = OPEN_EXISTING
004014DC  . 6A 00          PUSH 0                                                  pSecurity = NULL
004014DE  . 6A 01          PUSH 1                                                  ShareMode = FILE_SHARE_READ
004014E0  . 8BF0           MOV ESI,EAX
004014E2  . 68 00000010    PUSH 10000000                                           Access = GENERIC_ALL
004014E7  . 68 7C304000    PUSH Lab07-03.0040307C                                  FileName = "Lab07-03.dll"
004014EC  . 897424 74      MOV DWORD PTR SS:[ESP+74],ESI
004014F0  . FFD7           CALL EDI                                                CreateFileA
```

Figure 7. Disassembly showing the creation of a file and file mapping for suspicious kerne132.dll

From here, by observing the call tree, there is a reference to the procedure <Lab07-03.Sleep> (Figure 8) which gives some insight as to what the nefarious DLL file's function is. Upon inspecting the imports, with a tool like Process Hacker, of the provided Lab07-03.dll file, it is clear that its purpose is to either create a new process via exec, or to sleep.



Procedure	Calls	Comment
<Lab07-03.Sleep>	> Lab07-03.00401000	Leaf
	> Lab07-03.00401040	
	> Lab07-03.00401070	
	> Lab07-03.004010A0	
	> Lab07-03.004011E0	
	Lab07-03.00401440	
	Lab07-03.0040193C	
	Lab07-03.00401951	RETN
	MSVCRT.__getmainargs	Sys
	MSVCRT.__p__commode	Sys
	MSVCRT.__p__fmode	Sys
	MSVCRT.__p___initenv	Sys
	> MSVCRT.malloc	Sys
	MSVCRT._XcptFilter	Sys
	MSVCRT.__set_app_type	Sys
	MSVCRT._initterm	Sys
	MSVCRT.exit	Sys
	> MSVCRT._stricmp	Sys
	MSVCRT.__setusermatherr	Sys
	> MSVCRT._controlfp	Sys
	> kernel32.CreateFileA	Sys
	> kernel32.CreateFileMappingA	Sys
	> kernel32.CloseHandle	Sys
	> kernel32.IsBadReadPtr	Sys
	> kernel32.MapViewOfFile	Sys
	> kernel32.UnmapViewOfFile	Sys
	> kernel32.FindClose	Sys
	> kernel32.FindFirstFileA	Sys
	> kernel32.CopyFileA	Sys
	> kernel32.FindNextFileA	Sys
	Unknown destination(s)	

Figure 8. Reference to the Lab07-03.Sleep

Conclusion

This research project serves as an introduction to the field of malware analysis for people with little to no previous exposure. It provides an overview of malware lab setup, evaluation of analysis techniques and tools, and analysis of malware samples. Over time, more sophisticated malware, such as those that detect virtual machines, are being introduced to the Internet. More powerful modern analysis tools are being developed. Therefore, the future directions of this research project should continue to explore new analysis tools and gain insights into sophisticated malware.

References

[1] A. Zemlianichenko, “Code in huge ransomware attack written to avoid computers that use Russian, says New report,” *NBCNews.com*, 07-Jul-2021. [Online]. Available: https://www.nbcnews.com/politics/national-security/code-huge-ransomware-attack-written-avoid-computers-use-russian-says-n1273222. [Accessed: 12-Nov-2022].

[2] M. Sikorski and A. Honig, *Practical Malware Analysis*. San Francisco, CA: No Starch Press, 2012.