CYBR473
Malware and Reverse Engineering

Malware & Reverse Engineering T1 (2023) 1–5

Assignment 1: Basic Static and Dynamic Analysis of Malware

Olivia Fletcher
ID: 300534281, Email: fletcholiv@myvuw.ac.nz

School of Engineering and Computer Science, Victoria University of Wellington, P.O. Box 600, Wellington 6140, New Zealand.

Abstract

Malware asserts itself as a constant and significant threat to any and all digital environments, to computer systems and networks alike. Malware analysis and virus detection techniques are essential to ensure a device remains secure. In this report we will be delving into various static and dynamic malware analysis techniques and provide a breakdown of our findings from the provided files. Our end goal for this is to gain a better understanding of malware's behaviors, purposes and potential impacts it can have on a system while learning a valuable set of skills to analyze and dissect its executions.

1. Introduction

Malware analysis is a critical process for detecting and analyzing malicious software to understand its behaviors, executions and potential impact on a system. We will be working through and analyzing specific files provided with a guide via the "Practical Malware Analysis" literature by Michael and Andrew[1]. Practical Malware Analysis provides a comprehensive breakdown of different monitoring and aversion techniques by implementing industry standard static and dynamic malware analysis. Using the information provided by the literature we will be performing static and dynamic analysis through a virtual environment. Static analysis involves assessing malware's behaviors and potential abilities without executing the file. We can do this multiple ways such as string analysis. By performing string analysis we can decipher some core information about the malware however, using this alone does not always reveal its full capabilities. Dynamic analysis involves the assessing of malware via executing it and monitoring the system before/during/after execution to gain a better understanding of its abilities (such as through registry key monitoring). However, only doing one or the other is not always sufficient and to combine both static and dynamic analysis is helpful in malware assessments because it provides a more comprehensive breakdown of the malware's behavior and capabilities. By combining static and dynamic analysis, we can better understand the full scope of the malware's capabilities and behavior, and use that knowledge to develop future effective detection and mitigation strategies. Analyzing malware requires creating a safe and controlled environment where the malware can be executed and monitored without harming the host system. Virtual Box Virtual Machines (VMs) provide a safe and isolated environment for malware analysis. By creating a VM, the user can install an operating system and any necessary tools or software for analysis. The VM can be configured with network settings, file systems, and other resources to simulate a real-world environment. We will be implementing the use of two VM's and connecting them via a secure NAT-network to perform the analysis. The Windows 7 OS will be used for the execution while the REMnux device will provide the network monitoring side of the analysis by checking the connected IPs to the windows machine.

2. Network Setup

The network setup for this investigation is via VirtualBox Virtual Machine running through a Windows 7 OS and REMnux OS for monitoring. VirtualBox VMs is a powerful virtualisation tool which allows a user to create and run multiple virtual machines with a range of OSs to choose from. Implementing VirtualBox VMs will be incredibly helpful in our malware analysis as the use of a virtual machine creates a safe and controlled environment where the malware can be executed and monitored without bringing harm to our (host) machine. When importing these two VMs I have created a secure NATnetwork channel for the machines to communicate to each other through. The VirtualBox UI also offers the ability to "snapshot" a moment in time at various stages of the analysis which allows for back-tracking through the malware's execution which can provide useful data in our analysis. Why use Windows 7 and not a later, more-updated version? The malware we are reviewing is upwards to 10 years old and newer versions have already got the tools in place to automatically remove the malware, however by performing this analysis it will provide useful tools and knowledge that are still being used by cybersecurity professionals today.

- VirtualBox VMs with NATnetwork secure channel
- Windows 7 and REMnux operating systems
- VirtualBox Snapshotting tool

3. Static Analysis

Static analysis is a useful technique used in computer programming and cybersecurity which is an approach of analyzing malicious code without executing it. The basis involves analyzing the structure and contents of the code to identify potential vulnerabilities or threats that could be either exploited or planted by attackers (string analysis). In our case, we will be examining the malicious files to develop more understanding of its behavior, purposes and the potential impact it may have if executed on a system. We will be doing various basic and advanced techniques such as disassembling and decompiling it into software such as PEview/PEiD, string analysis via CFF Explorer and IDA Pro for debugging purposes. However, there are some limitations when it comes to static analysis as some more advanced malware have evasion techniques or blocks of code that adapts its structure based on tools used, sometimes we can see snippets of this behavior by reviewing things such as certain imports used from legitimate Windows tools (`e.g Kernel32.dll -> VirtualAlloc, VirtualProtect & VirtualFree` can be used as an evasion technique). With that being said, combining both static and dynamic techniques will provide a more comprehensive malware evaluation.

3.1. Basic Static Analysis

To perform our basic static analysis we will be utilizing straight-forward tools such as reviewing the file properties and VirusTotal report. Evaluating the security permissions of the unzipped malware file reveals that after execution the malware has full control over the system via being able to; modify, read, write and execute for all system/admin profiles. Uploading the malware.exe to VirusTotal shows that the malware matches 57 out of the available 68 antivirus engine signatures with 1 sandboxing environment marking it as malicious - which we will delve into deeper in sect. 5.1 VirusTotal Reporting. To delve into static analysis further I will utilize software such as PEview, PEiD and CFF Explorer as these will provide breakdowns of code executions, system calls and techniques.

3.2. Advanced Static Analysis

The first tool we will delve into is PEview [6] which is software that supports the ability to analyze and examine the PE, Portable Executable file format used within Windows OSs. PEview provides a comprehensive breakdown of each section of the file such as the various headers, import/export tables, system resources used and the base assembly code. We can use PEview to perform our analysis by checking if the file contains any unusual or potentially malicious behaviors. The first noteworthy section within our PEview output is the `IMAGE_FILE_HEADER` that contains the following characteristics: `RELOCS_STRIPPED, EXECUTABLE_IMAGE, NUMS_STRIPPED, SYMS_STRIPPED, DEBUG_STRIPPED.` `RelocsStripped` indicates that the file is able to be relocated when loaded into memory and can continue to run correctly regardless of its location within the system. NumsStripped, SymsStripped and DebugStripped in conjunction indicates that the file debugging information has been removed -¿ this is noteworthy later in regards to our analysis via PEiD. Within the next section `IMAGE_OPTIONAL_HEADER` we get some file structural information such as the entrypoint memory address location, memory alignment to minimize collisions and other information such as subsystem and image size. The following header files contain data directory information which is an important section as it contains data which highly suggests malicious activity such as the use of the import and export address tables containing executions to redirect the legitimate windows process to the malicious code. PEview also indicates that this malware was compiled on the date 2014-01-20. The import name table section contains the legitimate windows functions used by the malware within the files Kernal32.dll and Msvcrt.dll where there are a few functions called we must take note to: the called Kernel32 functions; ExitProcess, which can be used to terminate the malwares process to avoid detection. FreeLibrary, which could be used to unload the memory, used DLLs to avoid detection measures. GetProcAddress and LoadLibraryA can both be used to obtain and call functions from other DLL libraries. VirtualAlloc, VirtualFree, VirtualProtect, Virtual Query all could be used in conjunction with each other to perform a variety of actions such as allocating/manipulating memory and possibly further injecting malicious code into other system processes. The msvcrt.dll functions which are noteworthy are; _onexit, where the malware has the ability to register a function call during a process exit which means it could be used either for a memory cleanup or to execute additional malicious code (e.g after malware execution has finished, delete certain registry keys or clean up files to avoid detection). calloc, malloc, free, memcpy, memmove and memset could all be used for a variety of reasons such as manipulating memory and or injecting malicious code into other system events. Final noteworthy function would be the use of signal, which could be used by the malware to register signal handlers where it could intercept system events such as exceptions or interrupts (e.g malware hides itself by terminating its processes after detecting security software). When uploading the malware to a debugger such as PEiD we would receive the following "Nothing found. [Overlay] *" which is odd considering the debugger would pick up the other lab executables but not this one [add cite for peid]. This is due to the malware containing characteristics that specifically block this capability, hiding the packer. As seen from before, our header section in our PEview indicated that the malware has specifically removed debugging tools ability via the following: `NUMS_STRIPPED, SYMS_STRIPPED and DEBUG_STRIPPED`. This may be intentional by the writer/s to make it difficult to assess the malware.

To continue my static analysis I implemented the use of CFF Explorer [3]. CFF Explorer, like PEview is a free and powerful static analysis tool that has static analysis capabilities by allowing a user to inspect the structure and contents of executable files. As we have fully delved into the sections of the malware the notable feature that CFF Explorer offers is showcasing the provided PNG image. This image is important because it shows that the malware is attempting to appear like a legitimate file by changing its icon to blend in the system.

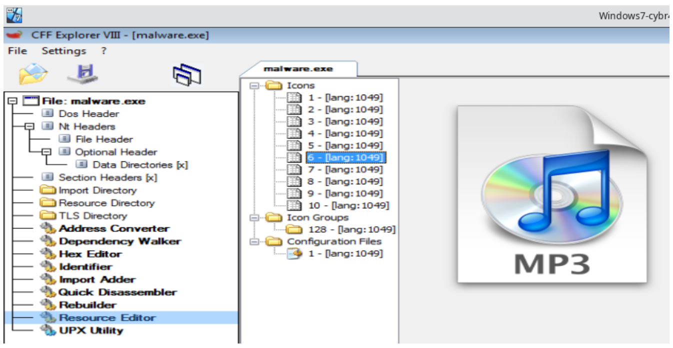

Figure 1: CFF Explorer - Malware PNG

4. Dynamic Analysis

Dynamic malware analysis involves executing the malware through a controlled environment such as in our case, VirtualBox VMs to observe its behaviors on the system. By executing the malware in the controlled environment we are able to isolate the malware to directly monitor its actions which reveals to us how it has been coded to interact with a system and what further executions it may perform. It is important to test dynamic analysis in different ways via different environments such as sandboxing because some malware writers specifically code their malware to detect if it is being analyzed or executed through a 'safe' (virtual machine) OS. More advanced dynamic analysis techniques would be directly debugging the malware using software tools which allows us to step through the code and observe its behavior and every execution it will attempt to do. Another form of more advanced analysis would be performing memory analysis which involves directly examining the system's memory to identify how the malware is operating and what resources it is using in an OS. Dynamic malware analysis techniques which we will be implementing are Wireshark, Regshot, ResourceHacker, ProcessHacker and ProcMon. We will be using Wireshark via the REMnux virtual machine to monitor the windows 7 OSs web traffic which we will delve into more in the Network Analysis section.

4.1. Basic Dynamic Analysis

To perform the dynamic analysis I will start with setting up the appropriate tools to be ready for the malware's execution such as booting up the REMnux virtual machine with Wireshark starting to capture packets from the Windows 7 IP (10.0.2.5) [2]. Next step is taking a registry key snapshot before execution using Regshot [9] which will provide a breakdown on all the registry key changes the malware makes after execution and finally start up Process Hacker [7] and ProcMon [8] which provides an overview of the current system services and programs running on the OS. By monitoring the tasks before and after we will be able to directly compare differences that arise from the malware's execution and if it overloads any system resources.

4.2. Network Analysis

To perform network analysis I have set up the REMnux OS within a virtual environment connected to the windows 7 VMs network to assess the communications during the malware's execution. I first tested the REMnux OS Wireshark connection to the Windows device by pinging two packets to see if it got picked up via the command `[ping c 2 10.0.2.5]`, the output shows that the packets were successfully received with Wireshark recording the connection. The first packet I immediately notice is from the IP address: `'23.212.169.169'` communicating with the windows machine containing encrypted information. This could signify to us that the malware is now communicating with the Windows device via a `C&C` (Command and Control) server using encryption to hide the contents of the connection. The next noteworthy packets are from an unusual source that start from an IP actually within the network at 10.0.2.2. This is a common tactic used by malware where it is able to embed itself within a network to continue malicious activity. The following two packets info contains the following `'Standard query 0x524a PTR 239.237.117.34.in-addr.arpa'` and `'Standard query response 0x524a PTR 239.237.117.34 239.237.117.34.in-addr.arpa PTR 239.237.117.34.bc...'`. This appears to be a reverse DNS lookup on the provided IP address. The next following packets are from a source of `'fe80::e887:209f:c75'` with the info tab having `'MSEARCH * HTTP1.1'` which could be the malware's activity trying to gather data of the available devices or services on the Windows network. The next packets are then from the Windows device to the above 239.237.117.34 IP with the info section containing `'52229 2302 Len 666'` which is interesting because this indicates communication between two endpoints (522229 to 2303) with the length of the packetpayload being 666. This could be the malware sending data back to its created channel. Due to the malware's ability to change registry items and make changes to system browser settings I believe the purpose of this malware is to be able to send further commands from the created HTTP service so it can not only manipulate the browser environment but also communicate back and forth, sending sensitive data. The purpose of a Loadmoney Trojan is mostly to generate income via browser adware but also potentially to steal the users sensitive information such as passwords or banking details. Details similar to the above could be used as a host based and network-based indicator on whether or not a device is infected with this variant of malware.

4.3. Advanced Dynamic Analysis

From performing our Regshot before and after it shows the following changes have been made to the registry: 191 total changes, 35 added keys, 109 added values, some noteworthy keys added: `SystemCertificates\AuthRoot\Certificates\values` Key is related to root certificates so it appears that the malware added a fake certificate to bypass security to appear safe to the user. `Microsoft\Tracing\malware_RASPI32, _RASMANCS`. The above two were also created by the malware and the use of the Tracing section indicates that it is attempting to log system information. `SYSTEM\ControlSet001\Enum\Root\LEGACY\+KPROCESSHACKER3\0000\Control`. The above registry key changes (total 12 changes, only provided one for an example)signify the following: the malware assessed the current system services that were being used and then took the name of the ProcessHacker software currently running then created a new service within the registry and gave itself system privileges under this name to avoid detection. I then tested this theory by using other software such as ProcMon with the exact same outcome, instead of ProcessHacker, the above keys were related to ProcMon. It also completely wiped my current ProcMon history as a means to conceal itself from detection. `Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\hiv`. The above keys are related to users' recent activity. In our case, the malware possibly noted that we have ProcessHacker in our usage history, thus using this name to appear legitimate when it creates new registry keys (as seen above) This is a common technique used by malware to attempt to blend into a system and appear legitimate. `Windows\CurrentVersion\Internet Settings\5.0\Cache\Extensible Cache\MSHist012023032420230235`. This key is used possibly for the malware to remove itself from the explorer browsing history to remain hidden.

Within the properties settings on the running malware on ProcessHacker shows the malware's key executions and behaviors when interacting with the system. Within the 'Token' tab it shows many important noteworthy executions such as what we discovered during the static analysis 'SeImpersonatePrivilege' which features the ability to 'Impersonate a client after authentication' and 'SeChangeNotifyPrivilege' with the ability to 'Bypass traverse checking'. The token section also shows that the malware created a new system account with execute and read privileges named `'S155059181'`. The memory tab shows all the locations of which the malware interacts with such as the kernal32 And msvcrt.dlls that we discovered in the static analysis with many more `(e.g wininet.dll, crtp32.dll etc` upwards to 30 more system32 dll interactions). Within the handles tab we get an overview which files, registrations and keys that the malware interacts with, within the keys section we can see those same key locations that have been manipulated by the malware as shown from the Regshot output above. Also something else I noticed within this were the following two keys: `'FEATURE_LOCALMACHINE_LOCKDOWN'` and `'FEATURE_HTTP_USERNAME_PASSWORD_DISABLE'` which is significant.

The next software I implemented for my analysis was the use of ProcMon[8]. ProcMon is a powerful Process Monitoring tool that works very similarly to ProcessHacker where it allows you to track file system activity, network activity, registry changes and thread activity instantaneously. First changes coming through we see are the registry key changes and elevated account privileges. We can also see that within the registry summary it is accessing the current internet settings and policies, which could indicate the malware setting up a communication channel.

5. Antivirus Signatures

Antivirus signatures work by identifying unique sets of patterns or sequences of binary code embedded within the malware. Different types of viruses (worms, malware, trojans etc) all have identifying characteristics unique to their type. Antivirus signature detection works via an antivirus engine which analyzes the code for known virus behavior patterns which can be used to identify instances of the same or similar types of malware. The programs then compare the contents of a file to its signature database to determine if it matches any already established virus or holds similar characteristics and then takes the steps to protect/repair the system from the virus's harm. This is why tools like VirusTotal and ClamAV come in handy as it utilizes many different antivirus signature databases, broadening the scope of signature checking. Even if a virus signature doesn't exactly match that within its database, if it contains a similar pattern to one already assessed it may mark it as suspicious or as malicious. This is known as a 'heuristic' based signature where the antivirus has tools in-place to detect new or previously unknown malware via reviewing and comparing the code's characteristics or patterns to the different types of malware/virus behaviors.

5.1. VirusTotal Report

In this section we will be analyzing the malware via uploading to VirusTotal[10]. VirusTotal is a free-to-use online tool, using a variety of antivirus engines and security tools which provides an in-depth malware analysis on suspicious files and URLs and gives recommendations for what next steps to take depending on the reports outcome. The file malware.exe and the initial page introduces the many different antivirus scanners and how they perceive the file. It shows that 57 out of the available 68 engines have marked this file as malicious and VirusTotal has signified that this is a variant of malware known as a Trojan-Loadmoney. Trojan malware is a type of malware most often used to gain unauthorized access to an infected system. Continuing on through the VirusTotal report are the' Details', 'Relations' and 'Behaviors' tabs which provide a further breakdown of information regarding the malware's abilities. The VirusTotal report shows a lot of the information we have already discovered this far such as the following; the malware creates its own digital signature in an attempt to appear as a legitimate file. VirusTotal report is the imports section within the details shows that the malware imports functions from Kernel32 and msvcrt. The details page also shows the use of an image icon and at the bottom provides the files Entropy value which is 7.394920349121094. This is considered quite high which in-turn signifies to us that a packer was used. So overall, the VirusTotal report is a very useful tool which provides a very detailed output of the malware's behaviors on a system, as we have been by conducting our own analysis. To fact check this, I then implemented the use of DiE (Detect it Easy)[4] to further assess whether or not the malware was packed. Reviewing the Entropy tab within the advanced settings of the file output it shows that not only does it contain packed data but 83% of the file is packed and is located in two sections: 'Section(1)['.data.'] with an Entropy of 7.99 and 'Overlay' with an Entropy value of 7.39

5.2. *ClamAV Signature*

ClamAV is an all-rounder antivirus software which provides many different tools to detect and eliminate malware. We are implementing ClamAV and utilizing its useful virus signature detection tool and will be comparing the resulting signature to our initial VirusTotal report to better profile the variant of malware. The initial setup took some time as it required some downloads and file editing to enable the software which I did struggle with as the command line scan execution was albeit confusing. Using the command line (making sure I am currently within the directory containing the ClamScan software) "clamscan C:\location\to\malware.exe" scans the malware's signature against the ClamAV signatures database. The output provides that ClamScan has marked it as a 'Trojan.Loadmoney-11912' variant of malware. VirusTotals report shows that 16 of the 57 other antivirus engines have also marked this malware as a 'Loadmoney'. This output significantly coincides with our analysis thus far as Loadmoney Trojans behave in a similar fashion.

The MD5 hash is `6c42954257ef80cc72266400236ea63c` and the SHA-1 hash is `f217d5ce69e0cb6c29889cc36ed707d2ed18e287`.

6. Conclusions

Being able to sufficiently perform both basic/advanced static and dynamic malware analysis techniques are vital to understanding the behaviors and overall goal that different viruses have. We utilized many different tools and techniques to perform our analysis on multiple files which have further developed my knowledge in being able to sufficiently identify potential threats and be able to take the appropriate actions in order to reduce harm to the system. As malware evolves, so do the techniques used for detection's and incidence responding. From our analysis, we have discovered and analyzed a Loadmoney Trojan variant from the file malware.exe which proves to be incredibly dangerous to an OS and has the ability to cause further damage if not contained properly.

7. Video Demonstration

The video demonstrations for Sections 2, 3.1 and 4.3 can be found under:

- CYBR473-a1-demo1-fletcholiv
  https://vstream.au.panopto.com/Panopto/Pages/Viewer.aspx?id=7f6113a0-d91a-4ed3-968e-afd800e3907a
- CYBR473-a1-demo2-fletcholiv
  https://vstream.au.panopto.com/Panopto/Pages/Viewer.aspx?id=f117ce61-5e92-4437-a580-afd801092b5d

8. Analysis Tools

The list of tools utilised in this assignment to perform the analysis and setup the environment are listed in Table 1 along with a brief description for each.

Table 1: The list of tools used for analysis.

Tool	Description
VirusTotal [10]	Free online public API that provides virus scanning and analysis on an uploaded file or URL.
WireShark [2]	A network-based packet analyzer which is used to capture packets from any connections within your network.
PEview [6]	A tool used to view and analyze the contents and structure within a Portable Executable (PE) file.
PEiD [5]	Used for detecting packed or encrypted code within a file.
CFF Explorer [3]	Similar use to PEview, which offers the ability to analyze the contents of a PE file in a different format.
Regshot [9]	Used as a registry-capture-compare tool which allows you to snapshot a system before and after executing malware to verify its registry key changes.
Process Hacker [7]	A multi purpose tool used for monitoring system activity and resources which can provide helpful in malware analysis.
ProcMon [8]	ProcMon is a process monitor tool which provides an overview of processes, registry changes and network changes on a system.

References

[1] Andrew.H, M., 2012. Praticle malware analysis: A hands-on guide to dissecting malicious software. http://ebookcentral.proquest.com/lib/vuw/detail.action?docID=1137570, accessed: 01/03/2023.
[2] Combs, G., 1997. Wireshark. https://www.wireshark.org/, accessed: 12/03/2023.
[3] Explorer, C., 2016. Cff explorer. https://github.com/cybertechniques/site/blob/master/analysis_tools/cff-explorer/index.md, accessed: 10/03/2023.
[4] it Easy, D., 2020. Detect it easy. https://github.com/horsicq/Detect-It-Easy, accessed: 25/03/2023.
[5] PEiD, 2017. Peid. https://www.aldeid.com/wiki/PEiD, accessed: 08/03/2023.
[6] PEView, 2013. Peview. https://www.aldeid.com/wiki/PEView, accessed: 08/03/2023.
[7] ProcessHacker, 2008. Processhacker. https://processhacker.sourceforge.io/, accessed: 20/03/2023.
[8] ProcMon, 2023. Procmon. https://learn.microsoft.com/en-us/sysinternals/downloads/procmon, accessed: 20/03/2023.
[9] Regshot, 1992. Regshot. https://code.google.com/archive/p/regshot/, accessed: 10/03/2023.
[10] VirusTotal, 2004. Virustotal. https://www.virustotal.com/gui/home/upload, accessed: 01/03/2023.