CYBR473
Malware and Reverse Engineering

Malware & Reverse Engineering T1 (2023) 1–7

Assignment 2: Mysterious 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 analysis refers to any type of investigation methods used in dissecting the behaviours and abilities of a malicious programs execution within a system. Malware refers to any type of software that's core design is to harm, disrupt and intercept access to a victims computer system or network architecture. As technology advances, more vulnerabilities arise where malware developers are able to create more advanced and stealthy software for many different intended purposes. It has become more increasingly important to be able to analyze the affects of many different types of malware to understand the behavior and goals in order to mitigate or prevent future attacks.

1. Introduction

There are three primary approaches to malware analysis: static analysis, dynamic analysis and network analysis. Static analysis refers to the techniques used when examining the code of a malicious program without actually executing it. This involves tools that allow for the ability to examine the binary file, disassemble the code, manipulate functions for analysis and analyzing embedded strings or windows resources used. Performing static analysis by implementing the use of many different tools can provide valuable insights into the structure, behaviour and goals of the malware. However, due to some malware authors employing obfuscation methods such as loading the code structures with packers it can be difficult to perform a full analysis and we may not be able to uncover the malware's full capabilities.

Dynamic analysis refers to the monitoring techniques and tools used when actually executing the malware on a system. The execution involves implementing a controlled environment such as a virtual machine, sandbox to execute the malware in order to monitor its full behaviours without infecting the host machine. The monitoring tools implemented can range from process monitoring, registry change logging, tracking system calls and file system activity. Preforming dynamic analysis by implementing the use of many different tools can provide valuable insights to the routes taken and resources used on a system. However, only performing dynamic analysis may not provide enough information as some malware developers employ obfuscation methods where the malware can hide or sleep its processes if virus detection methods are in place.

Network traffic analysis, closely followed with dynamic analysis involves monitoring the network traffic and dissecting the communication methods the malware takes with a command-and-control center and what the packets contain if applicable.

Today, we will be analyzing the file MysteriousMalware2023 by combining these three methods to be able to sufficiently assess and understand this malware's type, abilities, behaviours and overall goals.

2. Network Design

For the purpose of this analysis, I have implemented a SOP INetSim network as I believe it will provide a conducive environment for sufficient dynamic and network analysis. INetSim is an open-source tool designed to simulate various internet services within a local network or environment. Implementing a INetSim will enable us to analyze the malware's behaviour on the network in a controlled and secure manner. With this setup I will be able to recreate realistic network conditions in order to dissect what communications the malware attempts through the command-and-control center.

The SOP INetSim network is particularly valuable for malware analysis due to its ability to replicate essential internet services such as email, DNS, HTTP and FTP connections. This will essentially 'trick' the malware into believing an active connection has been made which in-turn allows for concise monitoring and logging of the network traffic generated by the malware. Additionally, the flexibility of a SOP INetSim will allow for me to set up multiple virtual machines running on different operating systems. For this analysis I will be employing the previous two VMs used in Assignment 1,

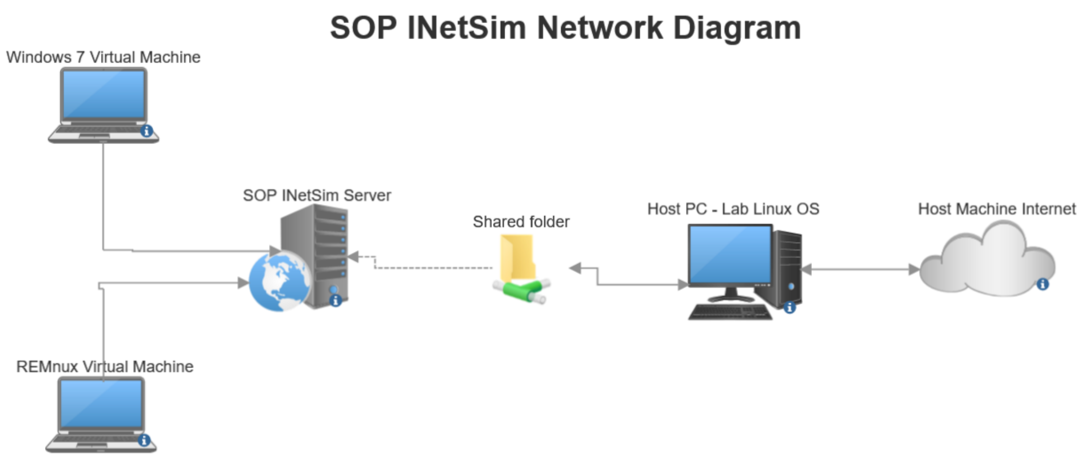


Figure 1: SOP INetSim Network Diagram

Windows 7 and REMnux devices. The above diagram displays my network implementation.

To establish the SOP INetSim network, I began by re-configuring and resetting the virtual machines used in Assignment 1. This process ensures a clean and standardized environment for accurate analysis.

2.1. Command and Control Centre

Command and Control centers play a crucial role in malware network analysis. These infrastructures serve as the communication hub between the victims OS and the attackers server. After an OS has been infected with malware the Command and Control center enables direct communication where the attacker can issue further commands, gather sensitive data, deliver payloads and even gain full administrative control. By analyzing the traffic made to and from the Command and Control center we will be able to decipher the capabilities and intentions behind the malware execution.

Due to the ambiguity behind the purpose of various types of malware it is important to truly discerning the goals set by the malware's developer. When a successful connection has been made from an attacker to an infected OS the purpose could range from quietly hiding in system processes, logging key-events and stealing sensitive data or similarly hiding within a system but slowly gathering up processing in order to launch a system-wide encryption for a ransom.

By analysing the packets sent through the Command and Control center we will be able to somewhat discern the goal and for this analysis I will employ the use of Wireshark to further delve into this traffic as seen in Section [8] - Network Indicators.

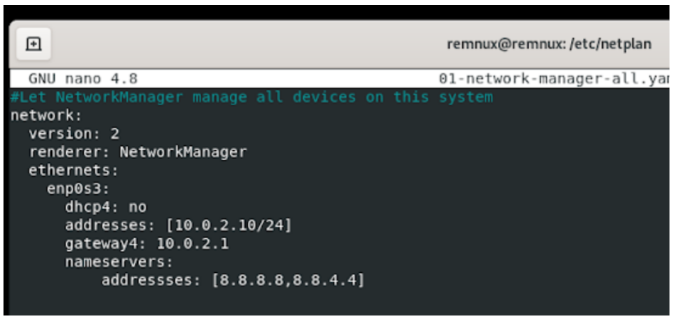
2.2. Network Deployment

The first step in Network Deployment was to create the network between the virtual machines themselves. Within the virtual machines settings I switched from a Nat-Network to an "internal network" which then requires further configuring within the machines themselves.

The next step was to configure the network so that both machines can communicate to one another. To do this I first launched the REMnux device and navigated to /etc/inetsim and made the following changes to inetsim.conf file using administrative privileges:

- Un-comment start_service dns within the first function calls
- Un-comment and add ip address under the service_bind_address function: service_bind_address 10.0.2.10
- Un-comment and add ip address under the dns_default_address function: dns_default_address 10.0.2.10

The next step is to navigate to etc/plan and create a .yaml file called '01-network-manager-all.yaml' and add the following text as seen in fig-1.



```
remnux@remnux: /etc/netplan
 GNU nano 4.8                                        01-network-manager-all.yam
#Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
      dhcp4: no
      addresses: [10.0.2.10/24]
      gateway4: 10.0.2.1
      nameservers:
          addressses: [8.8.8.8,8.8.4.4]
```

Figure 2: 01-network-manager-all.yaml

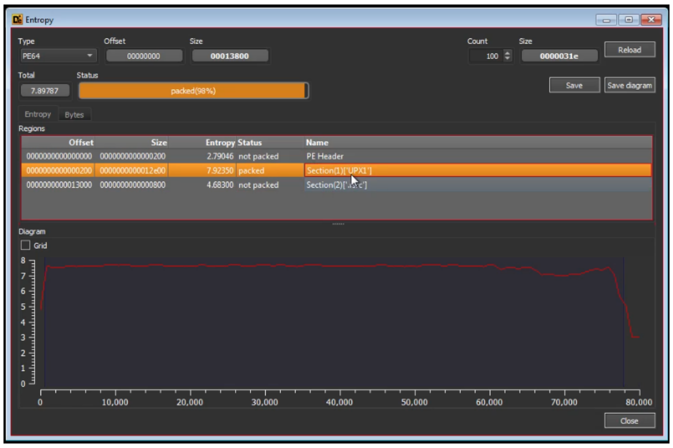
After saving the files the following commands must be executed to start up the network and apply the new INetSim functionality.

- sudo netplan try
- sudo netplan apply

Now with the REMnux device complete, we will move onto correlating the connection to the Windows device. The windows requirement was to simply edit the IP within the settings of the adapter available.

3. Static Analysis

To begin performing the static analysis I will start with uploading the file to software that can detect whether or not a packer was used and if this may hinder our following analysis. The packer identifying tool of choice I have implemented is the Detect it Easy (DiE) software [4] that has many helpful capabilities however the main functionality is packer detection. Upon uploading the malware we get a lot of interesting information layed out for us. The first thing we are going to assess is the packer information itself, within the advanced tab we can take a closer look at the entropy value which shows us that a whopping 98% of the malware is packed is loaded with a UPX01 packer. This high packer ratio identified by Detect it Easy suggests that the malware developers have employed packing techniques as a means of extensive obfuscation. The reason developers do this is for a few reasons, one of which is to evade already in-place security tools such as Antivirus and another is to make it challenging for static analysis due to intentionally including obfuscation methods within the code.


Type: PE64 Offset: 00000000 Size: 00013800 Count: 100 Size: 0000031e
Total: 7.89787 Status: packed(98%)

Regions

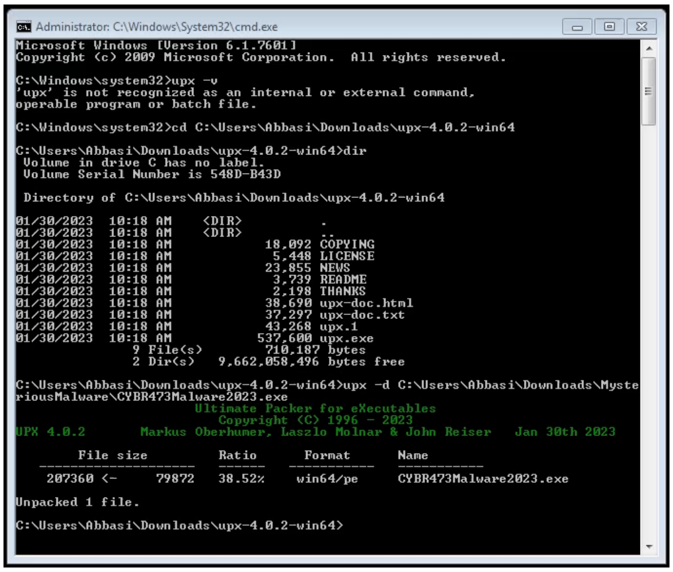
Offset	Size	Entropy	Status	Name
0000000000000000	0000000000000200	2.79046	not packed	PE Header
0000000000000200	0000000000012e00	7.92350	packed	Section(1)['UPX1']
0000000000013000	0000000000000800	4.68300	not packed	Section(2)['.rsrc']

Figure 3: Detect it Easy - Entropy Value

The presence of this packer will make it more difficult for us to extract meaningful data without employing other techniques such as dynamic analysis [4] or implementing the use of unpacking tools. As a means of testing this I first loaded up the packed version of the malware to IDA Freeware [3] to analyze the currently available code constructs. Upon uploading to IDA Freeware we get the following error: "Some imported modules will not be visible because the IAT is located outside the memory range of the input file" and upon clicking OK the only available functions are the following:

- start
- sub_14003CB12
- sub_14003CB50

With this, due to the very limited data available for analysis I will implement the use of an UPX un-packer [9] to sufficiently decompile the malware file.


```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>upx -v
'upx' is not recognized as an internal or external command,
operable program or batch file.

C:\Windows\system32>cd C:\Users\Abbasi\Downloads\upx-4.0.2-win64

C:\Users\Abbasi\Downloads\upx-4.0.2-win64>dir
 Volume in drive C has no label.
 Volume Serial Number is 548D-B43D

 Directory of C:\Users\Abbasi\Downloads\upx-4.0.2-win64

01/30/2023  10:18 AM    <DIR>          .
01/30/2023  10:18 AM    <DIR>          ..
01/30/2023  10:18 AM            18,092 COPYING
01/30/2023  10:18 AM             5,448 LICENSE
01/30/2023  10:18 AM            23,855 NEWS
01/30/2023  10:18 AM             3,739 README
01/30/2023  10:18 AM             2,198 THANKS
01/30/2023  10:18 AM            38,690 upx-doc.html
01/30/2023  10:18 AM            37,297 upx-doc.txt
01/30/2023  10:18 AM            43,268 upx.1
01/30/2023  10:18 AM           537,600 upx.exe
               9 File(s)        710,187 bytes
               2 Dir(s)   9,662,058,496 bytes free

C:\Users\Abbasi\Downloads\upx-4.0.2-win64>upx -d C:\Users\Abbasi\Downloads\Myste
riousMalware\CYBR473Malware2023.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2023
UPX 4.0.2       Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 30th 2023

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    207360 <-     79872   38.52%    win64/pe     CYBR473Malware2023.exe

Unpacked 1 file.

C:\Users\Abbasi\Downloads\upx-4.0.2-win64>
```

Figure 4: UPX Unpack

3.1. Code Decompiling

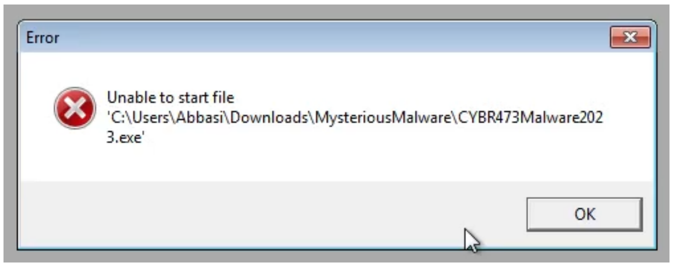
Now we have sufficiently unpacked the UPX contained within the malware we are able to decompile the code major constructs to further analyze the behaviours and properties implemented. First location of interest was within the imports tab which now shows a full breakdown of up to 125 windows API functions being used by the malware. I initially took notice to a amount of functions being implemented in regards to memory allocation, thread information, registry editing, file creating and file writing. Below are some noteworthy functions used:

- **ADVAPI32** - RegCreateKeyExA, RegOpenKeyExA, RegSetValueExA.
- **KERNAL32** - CreateEventA, CreateFileA, EnterCriticalSection, DeleteCriticalSection, InitializeCriticalSection, IsDebuggerPresent, IsProcessorFeaturePresent, TerminateProcess, VirtualProtect, WriteFile.
- **msvcrt** - _beginthreadex, _fmode, _read, _write, calloc, fopen, fprintf, free, malloc, memcpy, memmove, printf, realloc.
- **SHELL32** - SHGetFolderPathA.
- **USER32** - GetAsyncKeyState, GetKeyState, GetForegroundWindow, GetWindowTextA.

By reviewing the functions implemented from kernal32 and msvcrt I have taken notice that they heavily correlate to each other and are within similar locations within their function trees. I believe from analyzing these code constructs that the malware creates/manipulates core system files while also employing obfuscation techniques such as the use of the 'IsDebuggerPresent' from Kernal32.dll.

The functions implemented from shell32 and user32 as mentioned above can be manipulated or used in a couple different ways. The malware could be using the 'GetForegroundWindow' and 'GetWindowTextA' as another obfuscation technique in assessing the current applications that the user is viewing to verify if any antivirus/debuggers are currently being used or they could be used as a means of retrieving sensitive data that is being displayed on windows. These functions can also be used to monitor user activity by capturing window titles, url addresses, file names or visited directories.

Another function of interest was the use of 'IsDebuggerPresent' from the Kernel32.dll library. With some further investigation this functions behaviour shows this is used to check that if the file is being loaded into a debugger such as OllyDBG [5] and to block its ability to decompile the code. What I should have done here was to change/update this functionality within IDA Freeware to "unblock" this system call which would then allow me to upload and step through the malwares steps in OllyDBG. When attempting to upload to OllyDBG with the current configuration it returns the following error message as seen below (which I now believe is due to this process used).



Error

Unable to start file 'C:\Users\Abbasi\Downloads\MysteriousMalware\CYBR473Malware2023.exe'

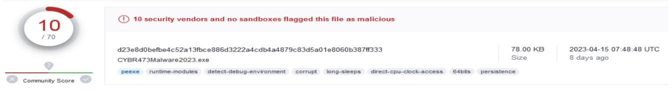
Figure 5: OllyDBG Error Message

3.2. *VirusTotal Report*

VirusTotal is a powerful tool based from many different antivirus engines and tested sandbox environments to deliver a full detailed report on the malware's capabilities, behaviours and overall threat level. Upon uploading the malware on VirusTotal the report shows that the malware has been marked as a 'Dorifel' variant of Trojan malware of which we will delve into further within the section Malware's Purpose 6.

I have uploaded this file twice, one on April the 23rd and another on May the 7th. The general information remains the same however what was noteworthy was the difference of security vendors output. My initial assessment showed that VirusTotal reported back that only 10 of the available 70 engines marked it as malicious however the second, more recent upload showed that 31 of 71 engines marked it as malicious. VirusTotal frequently updates their database with new malware signatures and this shows that it is important to revisit signature databases if any new information has been released.

Reviewing the updated VirusTotal report heavily coincides with our findings from the above static analysis [3] in regards to imports used and general functionality implemented from these imports. However, due to the nature of the report VirusTotal only provides availible information from the sections not



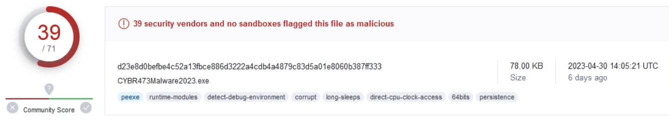
10 security vendors and no sandboxes flagged this file as malicious

d23e8d0befbe4c52a13fbce886d3222a4cdb4a4879c83d5a01e8060b387ff333
CYBR473Malware2023.exe

78.00 KB Size

2023-04-15 07:48:48 UTC 8 days ago

Figure 6: VirusTotal - April 23rd



39 security vendors and no sandboxes flagged this file as malicious

d23e8d0befbe4c52a13fbce886d3222a4cdb4a4879c83d5a01e8060b387ff333
CYBR473Malware2023.exe

78.00 KB Size

2023-04-30 14:05:21 UTC 6 days ago

Figure 7: VirusTotal - May 7th

containing a packer. Before un-packing the UPX loaded into the malware I could only review the following function imports:

- **ADVAPI32** - RegCloseKey
- **SHELL32** - SHGetFolderPathA
- **KERNEL32** - ExitProcess, GetProcAddress, LoadLibraryA, VirtualProtect
- **msvcrt** - exit
- **USER32** - GetKeyState

However, as mentioned above in section - Code Decompiling [3.1] there are actually 125 import functions implemented by the malware. With that I believe VirusTotal was great indicator of base functionality however does not provide a full report due to the packer used.

4. Dynamic Analysis

My plan for performing dynamic analysis involves utilizing system, registry and network monitoring tool such as ProcMon [7], Process Hacker [6], Regshot [8] and Wireshark [1]. The combined use of these monitoring tools can greatly enhance the effectiveness of this dynamic malware analysis. The first step is to to set up my environment ready for the malware's execution, this involves setting up the ProcMon, ProcessHacker, Wireshark and taking the first shot of the registry using Regshot.

ProcMon and Process Hacker both provide a comprehensive view of the system's processes, threads, modules, and other vital information. It will provide us valuable insights into the malware's behavior, memory usage, and resource allocation. On the other hand, ProcMon captures real-time system events, such as file accesses, registry operations, network activity, and process/thread creation. It provides detailed logs of these events, which will allow for us to track the interactions between the malware and the OS upon execution.

The will delve into the results from Regshot within section - System Indicators [7] and the Wireshark, Command-and-Control results within Section - Network Indicators [8].

Upon execution immediately a few noteworthy processes arise - ProcessHacker shows that the malware executable takes up 97.04% of CPU resources causing significant drops in speeds and in-turn made it difficult to use other tools such as performing the second registry key snapshot on RegShot. Navigating to the properties tab of the executable within Process Hacker it shows that within the Modules and Memory tabs that those same Windows API functions that we discovered during our static analysis are currently being processed such as; advapi32.dll with 876KB, kernel32.dll with 1.12MB, msvcrt.dll with 636KB, user32.dll with 0.98MB and shell32.dll with 13.53MB.

Within the malware properteis ’Token’ tab we can see some interesting abilities such as the ’SeChangeNotifyPrivilege’ which discription is to bypass traverse security checking. The ’Enviroment’ tab shows the exeuction enters system directories such as AppData/Local/Temp, Programdata, ProgramFiles, ProgramFiles(86x), ProgramFiles/CommonFiles and at the bottom with the name ’windows_tracing_logfile’ under ’BVTBin/Tests/installpackage/csilogfile.log which are all of interest. Overall we found some pretty interesting processes taken place after the malware’s execution.

5. Malware’s Information

Compilation time:

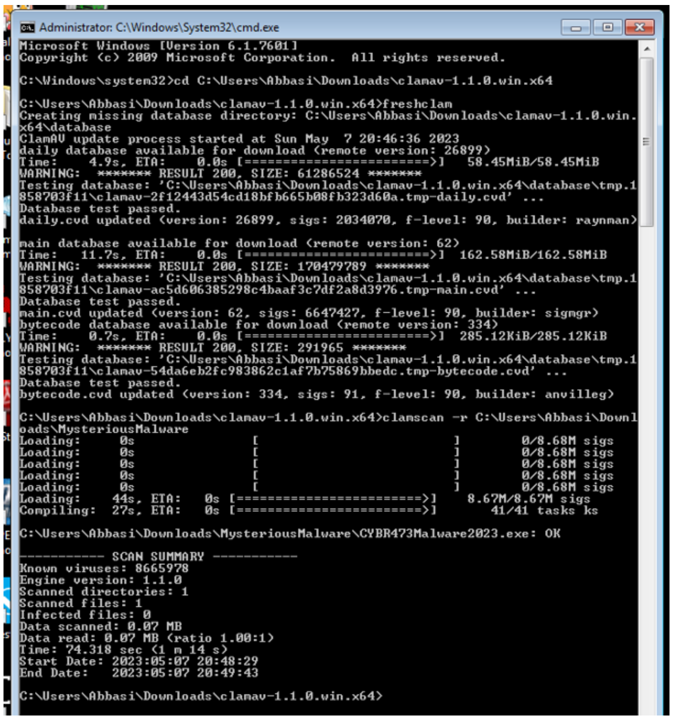
- VirusTotal reports a compilation time of April the 13th 2023 at 10:23am UTC.
- CFF Explorer reports a compilation time of April 13th 2023 at 10.23pm NZT.
- Detect it Easy (DiE) reports a compilation time of April the 13th 10.23pm.

Knowing the compilation time/data of malware is useful information and as we can see above the general timeline was that this malware’s creation date was April the 12th-13th 2023 which is very recent. This is noteworthy because it communicates to us that there may not be sufficient data/information available about this variant of malware and that security tools like VirusTotal may not yet have sufficient data or discovered this signature yet. From our above analysis within the subsection - VirusTotal Report [3.2] when we initially uploaded the file there were not many antivirus engines that have picked it up.

5.1. Signatures

VirusTotal hash results
MD5 - 2b43df664e060883e597fea656d7ba6f
SHA1 - 0af07cf0604e8b2cfa3649c8b4f5c9b3e18a2e96
SHA256 - d23e8d0befbe4c52a13fbce886d3222-a4cdb4a4879c83d5a01e8060b387ff333

ClamAV report shows that it has yet to pick up on this malware’s signature and does not contain a match in their database. I also moved this file onto a VM with an active internet connection, updated the ClamScan signature database locally and still came back with no results.



```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>cd C:\Users\Abbasi\Downloads\clamav-1.1.0.win.x64

C:\Users\Abbasi\Downloads\clamav-1.1.0.win.x64>freshclam
Creating missing database directory: C:\Users\Abbasi\Downloads\clamav-1.1.0.win.
x64\database
ClamAV update process started at Sun May  7 20:46:36 2023
daily database available for download (remote version: 26899)
Time:    4.9s, ETA:    0.0s [========================>]   58.45MiB/58.45MiB
WARNING:  ******** RESULT 200, SIZE: 61286524 ********
Testing database: 'C:\Users\Abbasi\Downloads\clamav-1.1.0.win.x64\database\tmp.1
858703f11\clamav-2f12443d54cd18bfb665b08fb323d60a.tmp-daily.cvd' ...
Database test passed.
daily.cvd updated (version: 26899, sigs: 2034070, f-level: 90, builder: raynman)

main database available for download (remote version: 62)
Time:   11.7s, ETA:    0.0s [========================>]  162.58MiB/162.58MiB
WARNING:  ******** RESULT 200, SIZE: 170479789 ********
Testing database: 'C:\Users\Abbasi\Downloads\clamav-1.1.0.win.x64\database\tmp.1
858703f11\clamav-ac5d606385298c4baaf3c7df2a8d3976.tmp-main.cvd' ...
Database test passed.
main.cvd updated (version: 62, sigs: 6647427, f-level: 90, builder: sigmgr)
bytecode database available for download (remote version: 334)
Time:    0.7s, ETA:    0.0s [========================>]  285.12KiB/285.12KiB
WARNING:  ******** RESULT 200, SIZE: 291965 ********
Testing database: 'C:\Users\Abbasi\Downloads\clamav-1.1.0.win.x64\database\tmp.1
858703f11\clamav-54da6eb2fc983862c1af7b75869bbedc.tmp-bytecode.cvd' ...
Database test passed.
bytecode.cvd updated (version: 334, sigs: 91, f-level: 90, builder: anvilleg)

C:\Users\Abbasi\Downloads\clamav-1.1.0.win.x64>clamscan -r C:\Users\Abbasi\Downl
oads\MysteriousMalware
Loading:     0s               [                         ]       0/8.68M sigs
Loading:     0s               [                         ]       0/8.68M sigs
Loading:     0s               [                         ]       0/8.68M sigs
Loading:     0s               [                         ]       0/8.68M sigs
Loading:     0s               [                         ]       0/8.68M sigs
Loading:    44s, ETA:   0s [========================>]    8.67M/8.67M sigs
Compiling:  27s, ETA:   0s [========================>]       41/41 tasks ks

C:\Users\Abbasi\Downloads\MysteriousMalware\CYBR473Malware2023.exe: OK

----------- SCAN SUMMARY -----------
Known viruses: 8665978
Engine version: 1.1.0
Scanned directories: 1
Scanned files: 1
Infected files: 0
Data scanned: 0.07 MB
Data read: 0.07 MB (ratio 1.00:1)
Time: 74.318 sec (1 m 14 s)
Start Date: 2023:05:07 20:48:29
End Date:   2023:05:07 20:49:43

C:\Users\Abbasi\Downloads\clamav-1.1.0.win.x64>
```

Figure 8: ClamScan - Output

6. Malware’s Purpose

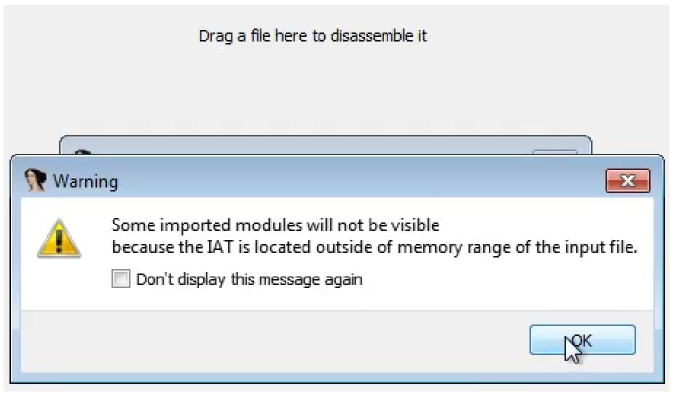
With the VirusTotal output heavily suggesting that this malware is of a Dorifel Trojan variant, I would also agree with this assumption due to our findings within sections - Static [3] and Dynamic Analysis [4].

A Dorifel trojan has been known to be able to execute quite a few commands depending on the goals set by the developer themselves. In our case, the Dorifel made quite a few system changes and appears to be hiding it’s processes within the system but other than taking significant system resources (CPU) it wouldn’t be obvious. Dorifel can/has been used as a means of gathering ransom off a victim after encrypting sensitive data or has been used as a keylogger; quietly taking the victims private keystrokes such as passwords, card information, private communications.

6.1. Obfuscation Methods

As per section - Static Analysis [3] we uncovered on Detect-it-Easy that the malware was loaded with a UPX packer that hid 98% of it’s major code constructs. As per section - System Indicators [7] we can see the changes made to the registry where it attempts to masquerade itself as other legitimate user files within the Windows-Explorer sections.

As per sections - Static Analysis [3] and Code Decompiling [3.1] we can see that by implementing certain Windows API functions can block the ability to decompile the file. Upon uploading the malware to IDA Freeware to disassemble we get a pop up warning us that ”some imported modules will not be visible because the IAT is located outside of memory range of the input file” as seen below in fig 2. However, utilizing the UPX unpacker tool fixes this issue. Building onto the above point, the use of employing the Windows API fucntion ’IsDebuggerPresent’ blocks the ability to use decompiling software such as OllyDBG. Next time, I will actually edit this functionality within IDA Freeware to remove this block before then performing a


Drag a file here to disassemble it

Warning

Some imported modules will not be visible
because the IAT is located outside of memory range of the input file.

Figure 9: IDA Freeware Warning

full analysis through OllyDBG as I believe it would've provided some really valuable insights.

7. System Indicators

For assessing system indicators I will utilize Regshot registry key snapshot to compare the registry event changes from after the malware's execution. RegShot is a powerful and useful tool for analyzing the behavior of malware on a system because it allows for snapshots of the Windows Registry before and after executing a program or malware sample. By comparing the two snapshots, RegShot helps us identify any changes made to the system's registry during the program's execution.

After executing the malware and taking my second snapshot I can the directly view the results through Notepad.

Regshot organises the output by sections as shown; Keys Deleted, Keys Added, Values Deleted, Values Added and Values Modified.

Noteworthy key deletion:
`'Microsoft-Windows-CurrentVersion-Internet Settings-5.0-Cache-Extensible Cache-MSHIST'`.

Noteworthy key additions:
(All contained within Software/Windows/CurrentVersion/Explorer)

- -comDl32-OpenSvePidMRU-.hiv
- -FileExts-.hiv
- -FileExts-.hiv-OpenWithList
- -RecentDocs-.hiv

With a few more references to -Shell-BagMRU and ComDlg.

Noteworthy value deletion:
-CachePath: "-%USERPROFILE%-AppData-Local-Microsoft -Windows-History-History, followed by related Cache removals.

Noteworthy value additions:

- -CurrentVersion-Explorer-ComDlg32-FirstFolder
- -CurrentVersion-Explorer-ComDlg32-LastVisitedPidMRU
- -Windows-Shell-Bags-272-Shell-KnownFolderDerivedType
- -Windows-Shell-Bags-272-Shell-SniffedFolderType: "Generic"

This section also show quite a few more references to the same ComDlg and BagMRU.

Noteworthy values modified:

- -Microsoft-Direct3D-MostRecentApplication-Name: "die.exe"
- -Microsoft-Direct3D-MostRecentApplication-Name: "wireshark.exe"

with also quite a few more references to RecentDocs within this section.

And at the bottom of the document were the following:
(all contained within Windows/Shell/Bags/AllFolders/Shell)

- WinPos1920x962x96(1).left [2x entries]
- WinPos1920x962x96(1).top [2x entries]
- WinPos1920x962x96(1).right [2x entries]
- WinPos1920x962x96(1).bottom [2x entries]

I believe the input references from 'Explorer' are all modifications made by the malware where one example, we can see some of those windows api functions being used such as the 'MostRecentApplicationName' which could be related to the GetForegroundWindow, GetWindowTextA implemented from the user32.dll library which is how the malware gathered that we currently had an instance of Wireshark and Detect it Easy in the foreground. I also think it is quite interesting within the values deleted section we can see that the app data within the windows history was removed, I believe this is an obfuscation technique used by the malware. Overall, some pretty interesting changes were made here.

8. Network Indicators

To perform my network analysis I will utilize WireShark for it's traffic monitoring abilities. Wireshark is a powerful network protocol analyzer and will is helpful for malware analysis due to it's ability to capture and analyze network traffic. We will be using this as out gateway in assessing the possible communications the malware employs with the Command-and-Control Center. Wireshark provides a detailed view of the communication between infected systems and the command-and-control server employed by the attacker. By capturing and inspecting packets, Wireshark enables us to identify the network protocols, data exchanged, and communication patterns used by the malware to interact with its C&C center. This information is crucial for understanding the malware's behavior, identifying command and control instructions, and uncovering potential indicators of compromise (IOCs).

Upon executing the malware on the Windows machine, the following packets initially come through: a packet from the windows OS sending a network wide broadcast to discover the other devices on the network (src 10.0.2.12, dest 10.255.255.255). The following 19 packets are then from the REMnux devices IP address (10.0.2.10), destination 239.255.255.250, SSDP protocol with the info containing 'NOTIFY * HTTP/1.1" where each packet has unique data entries. First packet entry contains the following: NOTIFY * HTTP/1.1 Host: 239.255.255.255:1900 Cache Control: max-age=1800 Location: http://10.0.2.10:39257/59558e33-2bb7-408d-b0d0-96bbb19b1307.xml Server: Linux/5.4.0-122 -generic service:X_MS_Media Receiver Registrar. The next packet is from the windows 7 10.0.2.12 to the same broadcast address 10.255.255.255 but this time containing random alphabet keys with a brief mention of 'MAILSLOT/BROWSER ......ABBASI-PC". Then the following packets have a source address that looks like 'fe80::348d:3887:1318:28ec' and a destination 'ff02::1:2' with the info containing 'Solicit XID: 0xea13c5 CID: 0010001260961550800276764l2. Letting the packets come in abit more I get some more interesting entries such as communication between the REMnux and the Windows OS where the following data is shown; 'root xmlnx= ”urn:schemas-upn p-org:device ¡friendlyName¿REMnux User’s media on remnux¡/friendlyName¿¡manufacturer¿ Ryge 1 Developers¡/manufacturer¿¡manufacturerURL¿¡http://www.rygel-project.org¿¡/manufacturerURL¿ and another being; ¡soap:Envelope xmlns:soap= ”http://www.w3.org/2003/05/soap-envelope” xmlns:wsa= ”http://schemas.xmlsoap.org/ws/2004/08/addressing” xmlns:wsd= ”http://schemas.xmlsoap.org/ws/2005/04/discovery” xmlns:wsdp=”http://schemas.xmlsoap.org/ws/2006/02/devprof” with many more additions to similar activity as above packets. What I believe we could be seeing here is that the malware could have sent that initial broadcast packet to the network as a means to discover any other active devices and then with a connection being made with the REMnux device it could have then been establishing the connection with the above dummy URLs as a means to deliver further payload/instructions from the Command-and-Control Center. However when reviewing the static analysis with the unpacked malware I did not notice any commands that may be causing these behaviours.

9. Conclusions

In conclusion, we were able to uncover valuable insights into the malware’s behaviour, abilities and route taken through the system. We leveraged many different various tools and techniques, including disassemblers like IDA Freeware, unpackers like UPX unpacker, network sniffers like Wireshark, and system monitoring like through Regshot and Process Hacker. This multidimensional approach allowed us to obtain a comprehensive view of the malware’s behavior and showcased threat-level. This malware has proved to be quite difficult to analyze due to its in-depth obfuscation methods used however we managed to bypass most of these barriers put in place. Overall, I enjoyed this assignment and found that I have learned a lot of new techniques in terms of analysis and have learned to take initiative in utilizing further tools (such as the UPX unpack functionality).

10. Video Demonstration

The video demonstrations for Sections Part 1 and Part 2 can be found under:

- fletcholiv-473-a2-demo1
  https://vstream.au.panopto.com/Panopto/Pages/Viewer.aspx?id=a6570b21-4d57-4c20-9a70-aff900b9ca4f
- fletcholiv-473-a2-demo2
  https://vstream.au.panopto.com/Panopto/Pages/Viewer.aspx?id=66cc08dc-997e-402d-b876-aff900ba773b
- fletcholiv-473-a2-demo3
  https://vstream.au.panopto.com/Panopto/Pages/Viewer.aspx?id=46212dbb-55cf-4f9a-9d2c-affb00c30a52

11. 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
Detect it Easy (DiE) [4]	Portable open-source packer identifier utility for quickly defining file types and more.
IDA Freeware [4]	Powerful tool used to quickly analyze binary code samples within files.
OllyDBG [5]	Disassembler level analysing debugger.
VirusTotal [10]	Free online public API that provides virus scanning and analysis on an uploaded file or URL.
WireShark [1]	A network-based packet analyzer which is used to capture packets from any connections within your network.
CFF Explorer [2]	Similar use to PEview, which offers the ability to analyze the contents of a PE file in a different format.
Regshot [8]	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 [6]	A multi purpose tool used for monitoring system activity and resources which can provide helpful in malware analysis.
ProcMon [7]	ProcMon is a process monitor tool which provides an overview of processes, registry changes and network changes on a system.

References

[1] Combs, G., 1997. Wireshark. https://www.wireshark.org/, accessed: 12/03/2023.

[2] Explorer, C., 2016. Cff explorer. https://github.com/cybertechniques/site/blob/master/analysis_tools/cff-explorer/index.md, accessed: 10/03/2023.

[3] HexRays, 1997. Ida freeware. https://hex-rays.com/ida-free/, accessed: 05/04/2023.

[4] majorgeeks.com, 1997. Detect it easy. https://www.majorgeeks.com/files/details/detect_it_easy.html, accessed: 05/04/2023.

[5] OlehYuschuk, 1997. Ollydbg. https://www.ollydbg.de/, accessed: 15/04/2023.

[6] ProcessHacker, 2008. Processhacker. https://processhacker.sourceforge.io/, accessed: 20/03/2023.

[7] ProcMon, 2023. Procmon. https://learn.microsoft.com/en-us/sysinternals/downloads/procmon, accessed: 20/03/2023.

[8] Regshot, 1992. Regshot. https://code.google.com/archive/p/regshot/, accessed: 10/03/2023.

[9] UPX, 1997. Upx executable packer. https://upx.github.io/, accessed: 20/04/2023.

[10] VirusTotal, 2004. Virustotal. https://www.virustotal.com/gui/home/upload, accessed: 01/03/2023.