

Roberto Reade

CSEC 202.03

Reverse Engineering Fundamentals

Professor Weissman

Reversing Project

INTRODUCTION:

The malware I will be analyzing and testing for this project is Lumma Stealer. This piece of malware can steal sensitive information from a user's computer. More specifically, it can target data on a user's system, browsers, and browser extensions.

Basic Static Analysis:**UPX:**

CSEC202 > Weissman Reversing Tools 2024 > Basic Static > UPX > upx-4.2.2-win64				Search upx-4.2.2-win64
Name	Date modified	Type	Size	
COPYING	3/21/2024 9:45 AM	File	18 KB	
LICENSE	3/21/2024 9:45 AM	File	6 KB	
malware.exe	4/22/2024 12:22 AM	Application	421 KB	
NEWS	3/21/2024 9:45 AM	File	25 KB	
README	3/21/2024 9:45 AM	File	4 KB	
THANKS.txt	3/21/2024 9:45 AM	Text Document	3 KB	
upx.1	3/21/2024 9:45 AM	1 File	43 KB	
upx.exe	3/21/2024 9:45 AM	Application	551 KB	
upx-doc.html	3/21/2024 9:45 AM	Microsoft Edge HTM...	38 KB	
upx-doc.txt	3/21/2024 9:45 AM	Text Document	37 KB	

The first Basic static tool I used was UPX. I am using UPX to check if the Lumma Stealer is packed by UPX. In order to do this, I first moved the malware into the UPX folder shown in the screenshot above. This would then allow me to open the command prompt from file explorer and have access to the malware.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rread\Downloads\CSEC202\Weissman Reversing Tools 2024\Basic Static\UPX\upx-4.2.2-win64>upx -o Unpacked_malware.exe -d malware.exe
          Ultimate Packer for eXecutables
          Copyright (C) 1996 - 2024
UPX 4.2.2      Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 3rd 2024

  File size      Ratio      Format      Name
  -----
upx: malware.exe: NotPackedException: not packed by UPX

Unpacked 0 files.

C:\Users\rread\Downloads\CSEC202\Weissman Reversing Tools 2024\Basic Static\UPX\upx-4.2.2-win64>
```

In the screenshot above, I entered the command:

"upx -o Unpacked_malware.exe -d malware.exe"

This command decompresses, and unpacks the malware.exe file. It then writes it to an output file. In this case I made the output file "Unpacked_malware.exe". Once this command was run, I noticed that the malware I am using is not packed by UPX. This is seen in the screenshot above in the red lettering.

Strings:

CSEC202 > Weissman Reversing Tools 2024 > Basic Static > Strings > Strings 2021				Search Strings 2021
	Name	Date modified	Type	Size
🔗	Eula.txt	3/21/2024 9:45 AM	Text Document	8 KB
🔗	Lab21-02.exe	11/21/2011 8:26 PM	Application	172 KB
🔗	malware.exe	4/22/2024 12:22 AM	Application	421 KB
🔗	strings.exe	3/21/2024 9:45 AM	Application	362 KB
🔗	strings64.exe	3/21/2024 9:45 AM	Application	467 KB
🔗	strings64a.exe	3/21/2024 9:45 AM	Application	514 KB

The next tool I used was Strings. This tool will look for strings of ASCII values in the malware.exe file and print them out for me to see. In order to use Strings, I first had to move the malware.exe file into the Strings folder as shown in the above screenshot. This allowed me to open the command prompt from the file explorer and have access to the malware file.

```
C:\Windows\System32\cmd.exe
C:\Users\rread\Downloads\CSEC202\Weissman Reversing Tools 2024\Basic Static\Strings\Strings 2021>strings64 -n 4 malware.exe | more

Strings v2.54 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

!This program cannot be run in DOS mode.
Rich
.text
`.rdata
@.data
.rsrc
uqSSS
SSSSSS
Virtf
ualP
PSH(fA
hLfA
hPfA
WPWW
h gA
_^[d
uMSSS
SSSS
SSSS
rK9w
h@gA
hPgA
@^[]
hhgA
```

In the screenshot above, I used the command: "strings64 -n 4 malware.exe | more". This command prints out all the strings in the malware.exe file that are a length of 4 or above. I did this since a lot of strings below a length of 4 are for the most part garbage.

```
C:\Windows\System32\cmd.exe
C:\Users\rread\Downloads\CSEC202\Weissman Reversing Tools 2024\Basic Static\Strings\Strings 2021>strings64 -n 4 -o malware.exe | more

Strings v2.54 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

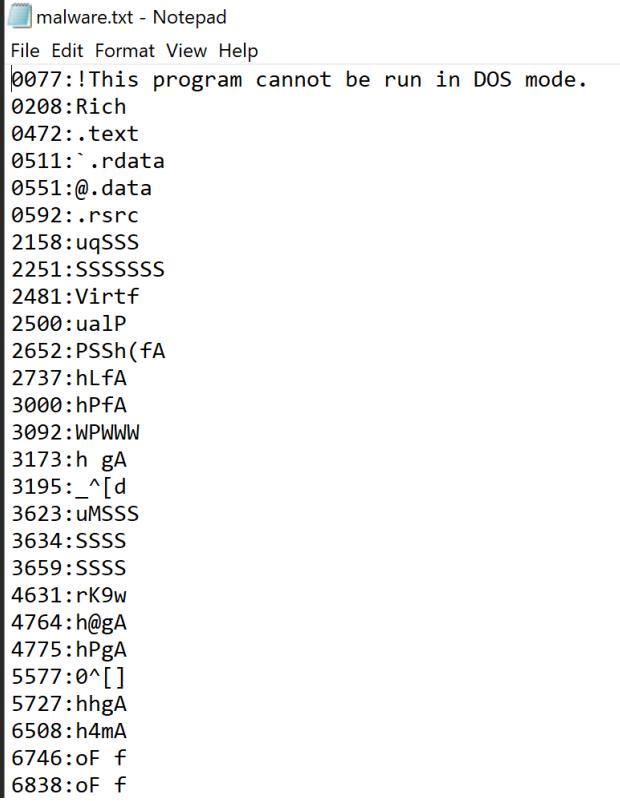
0077:!This program cannot be run in DOS mode.
0208:Rich
0472:.text
0511: .rdata
0551:@.data
0592:.rsrc
2158:uqSSS
2251:SSSSSS
2481:Virtf
2500:ualP
2652:PSSH(fA
2737:hLfA
3000:hPfA
3092:WPWWN
3173:h gA
3195:_^[_d
3623:uSSSS
3634:SSSS
3659:SSSS
4631:rK9w
4764:h@gA
4775:hPgA
5577:0^[]
E77777hba
```

In the above screenshot, I basically used the same command. However, in this command, I added “-o”. The new command I entered was: “strings64 -n 4 -o malware.exe | more”. This command and the “-o” allowed me to see the offset of the strings that were printed out. This helped me figure out which strings were used together if any.

```
C:\Users\rread\Downloads\CSEC202\Weissman Reversing Tools 2024\Basic Static\Strings\Strings 2021>strings64 -n 4 -o malware.exe > malware.txt

Strings v2.54 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\Users\rread\Downloads\CSEC202\Weissman Reversing Tools 2024\Basic Static\Strings\Strings 2021>notepad malware.txt
C:\Users\rread\Downloads\CSEC202\Weissman Reversing Tools 2024\Basic Static\Strings\Strings 2021>
```



malware.txt - Notepad
File Edit Format View Help
0077:!This program cannot be run in DOS mode.
0208:Rich
0472:.text
0511:`.rdata
0551:@.data
0592:.rsrc
2158:uqSSS
2251:SSSSSS
2481:Virtf
2500:ualP
2652:PSSh(fA
2737:hLfA
3000:hPfA
3092:WPWW
3173:h gA
3195:_^[_
3623:uMSSS
3634:SSSS
3659:SSSS
4631:rK9w
4764:h@gA
4775:hPgA
5577:0^[]
5727:hhgA
6508:h4mA
6746:oF f
6838:oF f

The last command I entered in Strings was: “strings64 -n 4 -o malware.exe > malware.txt”. This command took the output of the Strings command and made it into a .txt file. I then used the command: “notepad malware.txt” in order to open the .txt file in Notepad. I used these commands in order to save the output of Strings for this file so I could go back and look at it later for reference while using other tools.

After going through the output of Strings on my “malware.exe” file, I noticed a lot of interesting things.

I noticed that there were error messages with corresponding error numbers, as well as notification messages. These messages could either be for the threat actor, or they could show up on the user's computer when the user attempts to do something. If these messages are for the threat actor, they would most likely pop up on their computer and let them know if the malware ran into an error, or notify them that something worked correctly and they now have access to the user's information.

I also noticed strings of the days of the week and months of the year. These strings can be used for multiple things. These strings could potentially be used to manipulate timestamps, or schedule certain tasks. These strings could also be used to get the date when data was collected or exfiltrated, manipulate timestamps to evade security analysts, and possibly display dates in different formats depending on the users location.

There were many getter, setter, create and delete functions that I noticed when looking through the strings. These getter functions can be used to get information needed for the malware to run, or get personal information from the user's device. The setter functions can be used to activate different parts of malware, or set values to certain files or other important information.

The delete files can be used to delete important data or information from the user's computer. Finally, the create functions can be used to create files, create processes, create registry entries, create threads, or even create network connections.

In Lumma Stealer, I noticed a large amount of strings in the format of two letters separated by a dash and then two capital letters following that dash. Each of these strings represents a language code and country or region code. This allows the malware to be run in various countries with various languages.

Finally, at the end of the output, I noticed many long strings with the same letter, and strings with a bunch of random letters that looked like gibberish. These strings can potentially be used for buffer overflow attacks on the user's device.

PEview:

The screenshot shows the PEview interface with the file 'malware.exe' open. The left pane displays the file structure with sections like IMAGE_DOS_HEADER, IMAGE_NT_HEADERS, IMAGE_SECTION_HEADER (.text), IMAGE_SECTION_HEADER (.rdata), IMAGE_SECTION_HEADER (.data), and IMAGE_RESOURCE_DIRECTORY. The right pane shows the raw data for each section, including assembly-like mnemonics and hex values. The bottom status bar indicates 'Viewing malware.exe 0097 0418' and 'Table View'.

pFile	Raw Data	Value
00000000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....
00000010	E8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00@.....
00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030	00 00 00 00 00 00 00 00 00 00 00 E0 00 00 00 00
00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68!..L.ITh
00000050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
00000060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
00000070	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode.....\$.....
00000080	E4 88 A0 E8 A0 E9 CE BB A0 E9 CE BB A0 E9 CE BB
00000090	AD BB 11 BB B8 E9 CE BB AD BB 2E BB DC E9 CE BB
000000A0	AD BB 2F BB 8D E9 CE BB A9 91 5D BB A9 E9 CE BB	.../.....]
000000B0	A0 E9 CF BB D1 E9 CE BB 15 77 2B BB A1 E9 CE BBw+.....
000000C0	AD BB 15 BB A1 E9 CE BB 15 77 10 BB A1 E9 CE BBw.....
000000D0	52 69 63 68 A0 E9 CE BB 00 00 00 00 00 00 00 00	Rich.....
000000E0	50 45 00 00 4C 01 04 00 D7 47 71 64 00 00 00 00	PE..L..Gqd.....
000000F0	00 00 00 00 E0 03 01 0B 01 0C 00 00 F4 00 00
00000100	00 30 62 01 00 00 00 00 9F 3B 00 00 10 00 00 .Ob.....;
00000110	00 10 01 00 00 00 40 00 00 10 00 00 02 00 00 ..@.....
00000120	05 00 01 00 00 00 00 00 05 00 01 00 00 00 00 00
00000130	00 40 63 01 00 04 00 00 0F 8A 07 00 02 00 00 80	.@c.....
00000140	00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00
00000150	00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000160	44 72 01 00 64 00 00 00 00 40 61 01 78 F0 01 00	Dr..d...@.x...
00000170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000180	00 00 00 00 00 00 00 00 00 12 01 00 38 00 00 008.....
00000190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001A0	00 00 00 00 00 00 00 00 00 80 67 01 00 40 00 00 00	g..@.....
000001B0	00 00 00 00 00 00 00 00 00 10 01 00 94 01 00 00
000001C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D0	00 00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00text.....
000001E0	F5 F3 00 00 10 00 00 00 F4 00 00 00 04 00 00 00
000001F0	00 00 00 00 00 00 00 00 00 00 00 20 00 00 60
00000200	2E 72 64 61 74 61 00 00 98 6B 00 00 10 01 00 .rdata...k.....
00000210	00 6C 00 00 00 F8 00 00 00 00 00 00 00 00 00 00	l.....
00000220	00 00 00 40 00 00 40 2E 64 61 74 61 00 00 00 00@. @. data.....
00000230	20 BD 5F 01 00 80 01 00 00 3E 03 00 00 64 01 00>..d.....
00000240	00 00 00 00 00 00 00 00 00 00 00 40 00 00 C0@.....
00000250	2E 72 73 72 63 00 00 00 78 F0 01 00 40 61 01 .rsrc...x...@a.....
00000260	00 F2 01 00 00 A2 04 00 00 00 00 00 00 00 00 00
00000270	00 00 00 40 00 00 40 00 00 00 00 00 00 00 00 00@. @.....

In order to view Lumma Stealer in PEview, I had to open the malware in PEview as shown in the above screenshot. Once it opened, I noticed some differences from the binaries we examined in class.

In the IMAGE_NT_HEADERS section, there are four IMAGE_SECTION_HEADERS. These include .text, .rdata, .data, and .rsrc. A screenshot of one of these IMAGE_SECTION_HEADERS is shown below.

	pFile	Data	Description	Value
	00000250	2E 72 73 72	Name	.rsrc
	00000254	63 00 00 00		
	00000258	0001F078	Virtual Size	
	0000025C	01614000	RVA	
	00000260	0001F200	Size of Raw Data	
	00000264	0004A200	Pointer to Raw Data	
	00000268	00000000	Pointer to Relocations	
	0000026C	00000000	Pointer to Line Numbers	
	00000270	0000	Number of Relocations	
	00000272	0000	Number of Line Numbers	
	00000274	40000040	Characteristics	
		00000040		IMAGE_SCN_CNT_INITIALIZED_DATA
		40000000		IMAGE_SCN_MEM_READ

As you can see in the screenshot above, there are sections for the pFile, Data, Description, and value. Within this IMAGE_SECTION_HEADER.rsrc, it gives information on the size of the data as well as pointer information.

In the SECTION.rdata section, there are multiple IMPORT and IMAGE subsections. Below I will include a screenshot of an IMPORT and an IMAGE section.

	pFile	Data	Description	Value
	0000FA00	00000000	Characteristics	
	0000FA04	660CCFD7	Time Date Stamp	2024/04/03 Wed 03:41:11 UTC
	0000FA08	0000	Major Version	
	0000FA0A	0000	Minor Version	
	0000FA0C	00000002	Type	IMAGE_DEBUG_TYPE_CODEVIEW
	0000FA10	00000032	Size of Data	
	0000FA14	000167E0	Address of Raw Data	
	0000FA18	00014FE0	Pointer to Raw Data	
	0000FA1C	00000000	Characteristics	
	0000FA20	662498DA	Time Date Stamp	2024/04/21 Sun 04:40:58 UTC
	0000FA24	0000	Major Version	
	0000FA26	0000	Minor Version	
	0000FA28	0000000C	Type	IMAGE_DEBUG_TYPE_
	0000FA2C	00000014	Size of Data	
	0000FA30	00016844	Address of Raw Data	
	0000FA34	00015044	Pointer to Raw Data	

	pFile	Data	Description	Value
	00015A44	000172B8	Import Name Table RVA	
	00015A48	00000000	Time Date Stamp	
	00015A4C	00000000	Forwarder Chain	
	00015A50	00017724	Name RVA	KERNEL32.dll
	00015A54	00011010	Import Address Table RVA	
	00015A58	000172B0	Import Name Table RVA	
	00015A5C	00000000	Time Date Stamp	
	00015A60	00000000	Forwarder Chain	
	00015A64	0001774C	Name RVA	GDI32.dll
	00015A68	00011008	Import Address Table RVA	
	00015A6C	000172A8	Import Name Table RVA	
	00015A70	00000000	Time Date Stamp	
	00015A74	00000000	Forwarder Chain	
	00015A78	0001776E	Name RVA	ADVAPI32.dll
	00015A7C	00011000	Import Address Table RVA	
	00015A80	00017434	Import Name Table RVA	
	00015A84	00000000	Time Date Stamp	
	00015A88	00000000	Forwarder Chain	
	00015A8C	0001778E	Name RVA	WINHTTP.dll
	00015A90	0001118C	Import Address Table RVA	
	00015A94	00000000		
	00015A98	00000000		
	00015A9C	00000000		
	00015AA0	00000000		
	00015AA4	00000000		

As you can see in the screenshots above, the IMAGE and IMPORT sections also give you sections for pFile, Data, Description, and Value. In each IMAGE section it gives information regarding the characteristics, timestamp, the versions, and information on the data. In each

IMPORT section, it gives you information regarding the name RVA as well as other information regarding the imports and functions used.

In the SECTION.rsrc section, there are many subsections. These include IMAGE, ICON, GROUP, CURSOR, VERSION, and STRING. As you can tell, there is no binary in this section for me to extract like we did in labs and in class. This is because the malware I am using is not a binary. Below I will include a screenshot from each subsection in SECTION.rsrc.

	pFile	Data	Description	Value
0004A200	00000000	Characteristics		
0004A204	00000000	Time Date Stamp		
0004A208	0000	Major Version		
0004A20A	0000	Minor Version		
0004A20C	0000	Number of Named Entries		
0004A20E	0007	Number of ID Entries		
0004A210	00000001	ID		
0004A214	80000048	Offset to DIRECTORY	CURSOR	
0004A218	00000003	ID		
0004A21C	80000098	Offset to DIRECTORY	ICON	
0004A220	00000005	ID		
0004A224	80000190	Offset to DIRECTORY	DIALOG	
0004A228	00000006	ID		
0004A22C	800001A8	Offset to DIRECTORY	STRING	
0004A230	0000000C	ID		
0004A234	800001E8	Offset to DIRECTORY	GROUP_CURSOR	
0004A238	0000000E	ID		
0004A23C	80000210	Offset to DIRECTORY	GROUP_ICON	
0004A240	00000010	ID		
0004A244	80000248	Offset to DIRECTORY	VERSION	

The IMAGE section, as seen previously, includes pFile, Data, Description, and Value. It gives information about the ID and Offset to DIRECTORY for every value.

	pFile	Raw Data	Value
0004ACB0	28 00 00 00 18 00 00 00	30 00 00 00 01 00 08 00	(..... 0
0004ACC0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0004ACD0	00 00 00 00 00 00 00 00	81 7F 7F 00 83 83 7F 00
0004ACE0	81 7F 82 00 80 86 7D 00	83 83 83 00 7E 7C 80 00}..... ..
0004ACF0	7F 83 7C 00 82 86 81 00	82 7C 81 00 7C 81 83 00
0004AD00	82 82 80 00 82 7E 7D 00	82 7F 7E 00 84 7F 81 00~}...~}..
0004AD10	7F 7F 80 00 81 7F 82 00	7C 81 80 00 7C 7B 82 00
0004AD20	84 7E 7C 00 7E 7F 85 00	7E 80 7F 00 81 82 7F 00	.~ ..~..~ ..
0004AD30	80 7E 7D 00 80 83 7C 00	7D 81 83 00 85 7E 7B 00	.~}.. ..}..~{..
0004AD40	84 7F 81 00 7D 79 7E 00	86 84 7E 00 7D 84 7D 00	...}y~..~.}..
0004AD50	7C 79 7B 00 83 83 84 00	7B 84 83 00 7E 80 84 00	y{.....
0004AD60	7B 82 80 00 7F 7E 7B 00	7C 85 7F 00 7B 82 86 00	{...~
0004AD70	84 7B 7B 00 80 7C 7A 00	85 83 84 00 81 7F 7E 00	.{ .. z.....~
0004AD80	84 80 85 00 7C 81 7A 00	84 7E 82 00 83 84 7F 00z...~....
0004AD90	83 7C 7C 00 7B 7A 7C 00	7C 80 7F 00 7B 81 83 00	. .{z
0004ADA0	7B 7E 85 00 85 84 83 00	7C 7D 82 00 84 7F 84 00	{~..... }....
0004ADB0	83 7E 81 00 7D 7E 7F 00	85 83 80 00 7E 80 80 00	.~..}~.....~
0004ADC0	7C 7E 7B 00 85 80 7C 00	7E 7D 7F 00 7B 7F 80 00	{-{.. ..}- ..
0004ADD0	81 7F 7E 00 82 84 80 00	82 7B 7E 00 80 82 7F 00	.~..... ~.....
0004ADE0	84 7F 80 00 7C 83 83 00	7B 7E 83 00 7F 7C 7A 00 {~.. z ..
0004ADF0	7B 7D 7A 00 83 82 7F 00	84 7C 80 00 7C 81 82 00	{}z.....
0004AE00	7C 82 83 00 7E 83 81 00	7F 80 80 00 85 7A 7C 00~....z ..

	pFile	Raw Data	Value
0005D520	00 00 01 00 08 00 30 30	00 00 01 00 08 00 A8 0E00.....
0005D530	00 00 0E 00 20 20 00 00	01 00 08 00 A8 08 00 00
0005D540	0F 00 18 18 00 00 01 00	08 00 C8 06 00 00 10 00
0005D550	10 10 00 00 01 00 08 00	68 05 00 00 11 00 30 30h.....00
0005D560	00 00 01 00 20 00 A8 25	00 00 12 00 20 20 00 00%.....
0005D570	01 00 20 00 A8 10 00 00	13 00 18 18 00 00 01 00
0005D580	20 00 88 09 00 00 14 00	10 10 00 00 01 00 20 00
0005D590	68 04 00 00 15 00		h.....

-ICON 000B 0418
 -ICON 000C 0418
 -ICON 000D 0418
 -GROUP_ICON 0097 0418
 -ICON 000E 0418
 -ICON 000F 0418
 -ICON 0010 0418
 -ICON 0011 0418
 -ICON 0012 0418
 -ICON 0013 0418
 -ICON 0014 0418
 -ICON 0015 0418
 -GROUP_ICON 009B 0418
 -ICON 0016 0418
 -ICON 0017 0418
 -ICON 0018 0418
 -ICON 0019 0418
 -ICON 001A 0418
 -ICON 001B 0418
 -ICON 001C 0418
 -ICON 001D 0418
 -GROUP_ICON 009F 0418
 -CURSOR 001E 0000
CURSOR 001F 0000
 -CURSOR 0020 0000

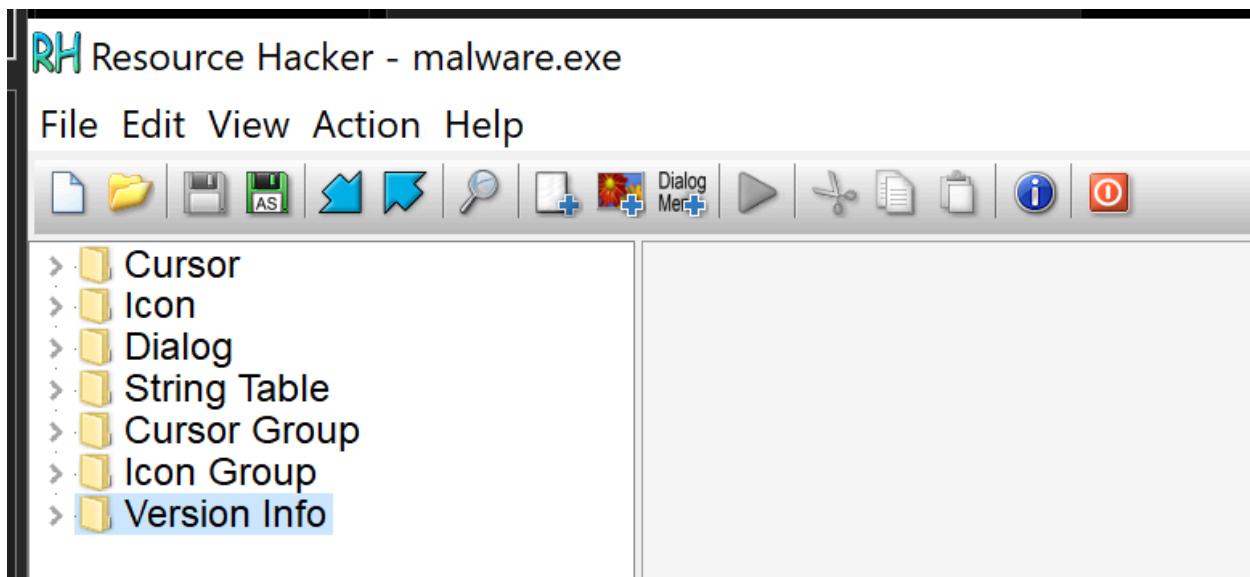
-ICON 000B 0418
 -ICON 000C 0418
 -ICON 000D 0418
 -GROUP_ICON 0097 0418
 -ICON 000E 0418
 -ICON 000F 0418
 -ICON 0010 0418
 -ICON 0011 0418
 -ICON 0012 0418
 -ICON 0013 0418
 -ICON 0014 0418
 -ICON 0015 0418
 -GROUP_ICON 009B 0418
 -ICON 0016 0418
 -ICON 0017 0418
 -ICON 0018 0418
 -ICON 0019 0418
 -ICON 001A 0418
 -ICON 001B 0418
 -ICON 001C 0418
 -ICON 001D 0418
 -GROUP_ICON 009F 0418
 -ICON 0016 0418
 -ICON 0017 0418
 -ICON 0018 0418
 -ICON 0019 0418
 -ICON 001A 0418
 -ICON 001B 0418
 -ICON 001C 0418
 -ICON 001D 0418
 -GROUP_CURSOR 094E 0000
 -CURSOR 0021 0000
 -CURSOR 0022 0000
 -GROUP_CURSOR 0953 0000
 -CURSOR 0023 0000
 -CURSOR 0024 0000
 -CURSOR 0025 0000
 -GROUP_CURSOR 0954 0000
VERSION 0001 0000

pFile	Raw Data	Value
00064C78	28 00 00 00 20 00 00 00 40 00 00 00 01 00 08 00	(.....@.....
00064C88	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00064C98	00 00 00 00 00 00 00 00 00 00 00 00 00 00 0A 0A 0A 00
00064CA8	2A 2A 2A 00 35 35 35 00 4D 4D 4D 00 8A FF 31 00	***.555.MM..1.
00064CB8	B3 B3 B3 00 D3 D3 D3 00 D4 D4 D4 00 DA DA DA 00
00064CC8	F7 F7 F7 00 FE FE FE 00 91 A5 FF 00 B1 BF FF 00
00064CD8	D1 DA FF 00 FF FF FF 00 00 00 00 00 00 1A 2F 00/.
00064CE8	00 2D 50 00 00 3F 70 00 00 51 90 00 00 63 B0 00	.-P..?p.Q..c..
00064CF8	00 76 CF 00 00 88 F0 00 11 98 FF 00 31 A6 FF 00	.v.....1..
00064D08	51 B3 FF 00 71 C1 FF 00 91 CF FF 00 B1 DD FF 00	Q...q..
00064D18	D1 EB FF 00 FF FF FF 00 00 00 00 00 00 00 2C 2F 00/.
00064D28	00 4B 50 00 00 68 70 00 00 86 90 00 00 A5 B0 00	.KP..hp.....
00064D38	00 C3 CF 00 00 E1 F0 00 11 EF FF 00 31 F1 FF 001..
00064D48	51 F3 FF 00 71 F5 FF 00 91 F7 FF 00 B1 FF FF 00	Q...q..
00064D58	D1 FB FF 00 FF FF FF 00 00 00 00 00 00 2F 21 00/..
00064D68	00 50 37 00 00 70 4C 00 00 90 63 00 00 B0 79 00	.P7..pL..c..y..
00064D78	00 CF 8F 00 00 F0 A6 00 11 FF B4 00 31 FF BE 001..
00064D88	51 FF C8 00 71 FF D3 00 91 FF DC 00 B1 FF E5 00	Q...q..
00064D98	D1 FF F0 00 FF FF FF 00 00 00 00 00 00 2F 0E 00/..
00064DA8	00 50 18 00 00 70 22 00 00 90 2C 00 00 B0 36 00	.P...p".....6.
00064DB8	00 CF 40 00 00 F0 4A 00 11 FF 5B 00 31 FF 71 00	.@...J...[.1.q..
00064DC8	51 FF 87 00 71 FF 9D 00 91 FF B2 00 B1 FF C9 00	Q...q..
00064DD8	D1 FF DF 00 FF FF FF 00 00 00 00 02 2F 00 00/..
00064DE8	04 50 00 00 06 70 00 00 08 90 00 00 0A B0 00 00	.P...p..
pFile	Raw Data	Value
000679A8	EC 01 34 00 00 00 56 00 53 00 5F 00 56 00 45 00	.4...V.S._.V.E.
000679B8	52 00 53 00 49 00 4F 00 4E 00 5F 00 49 00 4E 00	R.S.I.O.N._.I.N.
000679C8	46 00 4F 00 00 00 00 00 BD 04 EF FE 00 00 01 00	F.O.....
000679D8	00 00 2A 00 00 00 00 00 00 00 34 00 00 00 00 00	*.....4....
000679E8	0A 95 00 00 00 00 00 00 23 08 02 00 00 00 00 00#.....
000679F8	00 00 00 00 00 00 00 00 00 00 00 00 00 04 A1 00 00J..
00067A08	01 00 53 00 74 00 72 00 69 00 6E 00 67 00 46 00	.S.t.r.i.n.g.F.
00067A18	69 00 6C 00 65 00 49 00 6E 00 66 00 6F 00 00 00	i.l.e.l.n.f.o...
00067A28	26 01 00 00 01 30 00 34 00 32 00 39 00 32 00	&.....4.2.9.2.
00067A38	34 00 45 00 36 00 00 00 38 00 0C 00 01 00 46 00	4.E.6...8....F.
00067A48	69 00 6C 00 65 00 56 00 65 00 72 00 73 00 69 00	i.l.e.V.e.r.s.i..
00067A58	6F 00 6E 00 00 00 00 32 00 34 00 2E 00 33 00	o.n....2.4...3..
00067A68	34 00 2E 00 32 00 38 00 2E 00 37 00 39 00 00 00	4...2.8...7.9...
00067A78	32 00 05 00 01 00 46 00 69 00 6C 00 65 00 44 00	2....F.i.l.e.D.
00067A88	65 00 73 00 63 00 72 00 69 00 70 00 74 00 69 00	e.s.c.r.i.p.t.i..
00067A98	6F 00 6E 00 00 00 00 42 00 69 00 6C 00 6C 00	o.n....B.i.l.l.
00067AA8	00 00 00 34 00 06 00 01 00 4F 00 72 00 69 004....O.r.i..
00067AB8	67 00 69 00 6E 00 61 00 6C 00 46 00 69 00 6C 00	g.i.n.a.l.F.i.l..
00067AC8	65 00 6E 00 61 00 6D 00 65 00 00 00 46 00 69 00	e.n.a.m.e....F.i..
00067AD8	72 00 65 00 73 00 00 30 00 08 00 01 00 50 00	r.e.s...0....P.
00067AE8	72 00 6F 00 64 00 75 00 63 00 74 00 4E 00 61 00	r.o.d.u.c.t.N.A..
00067AF8	6D 00 65 00 00 00 00 53 00 6D 00 65 00 6C 00	m.e....S.m.e.l..
00067B08	66 00 69 00 65 00 00 00 3E 00 0B 00 01 00 50 00	f.i.e...>....P.
00067B18	72 00 6F 00 64 00 75 00 63 00 74 00 56 00 65 00	r.o.d.u.c.t.V.e..
00067B28	72 00 73 00 69 00 6F 00 6E 00 73 00 00 00 00	r.s.i.o.n.s....
00067B38	35 00 2E 00 33 00 30 00 2E 00 34 00 30 00 2E 00	5...3.0...4.0...
00067B48	39 00 35 00 00 00 00 44 00 00 00 01 00 56 00	9.5....D....V.
00067B58	61 00 72 00 46 00 69 00 6C 00 65 00 49 00 6E 00	a.r.F.i.l.e.l.n..
00067B68	66 00 6F 00 00 00 00 24 00 04 00 00 00 54 00	f.o....\$.T...
00067B78	72 00 61 00 6E 00 73 00 6C 00 61 00 74 00 69 00	r.a.n.s.l.a.t.i..
00067B88	6F 00 6E 00 00 00 00 00 0A 16 E3 04	o.n.....

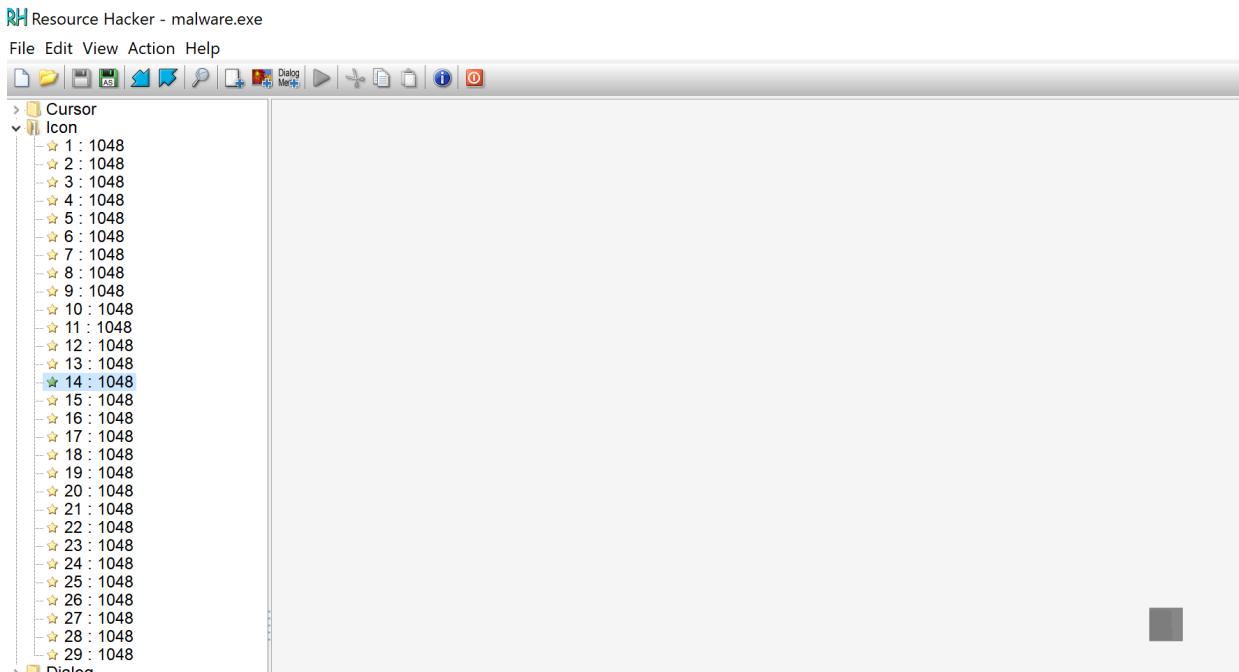
	pFile	Raw Data	Value
ICON 000B 0418	00067BF0	60 00 5A 00 75 00 79 00 61 00 7A 00 61 00 77 00	'Z.u.y.a.z.a.w.
ICON 000C 0418	00067C00	75 00 73 00 20 00 68 00 69 00 7A 00 6F 00 6C 00	u.s..h.i.z.o.l.
ICON 000D 0418	00067C10	69 00 78 00 61 00 7A 00 65 00 6B 00 61 00 20 00	i.x.a.z.e.k.a.
GROUP_ICON 0097 0418	00067C20	6E 00 6F 00 76 00 20 00 6C 00 65 00 77 00 69 00	n.o.v..l.e.w.i.
ICON 000E 0418	00067C30	76 00 75 00 72 00 69 00 74 00 61 00 6A 00 61 00	v.u.r.i.t.a.j.a.
ICON 000F 0418	00067C40	76 00 65 00 20 00 63 00 6F 00 68 00 65 00 76 00	v.e..c.o.h.e.v.
ICON 0010 0418	00067C50	61 00 63 00 69 00 76 00 69 00 20 00 67 00 61 00	a.c.i.v.i..g.a.
ICON 0011 0418	00067C60	64 00 75 00 76 00 75 00 20 00 67 00 69 00 62 00	d.u.v.u..g.i.b.
ICON 0012 0418	00067C70	69 00 66 00 69 00 76 00 61 00 70 00 61 00 20 00	i.f.i.v.a.p.a.
ICON 0013 0418	00067C80	67 00 6F 00 76 00 61 00 77 00 65 00 78 00 61 00	g.o.v.a.w.e.x.a.
ICON 0014 0418	00067C90	72 00 65 00 68 00 20 00 6E 00 75 00 66 00 69 00	r.e.h..n.u.f.i.
ICON 0015 0418	00067CA0	73 00 61 00 79 00 69 00 20 00 6D 00 61 00 68 00	s.a.y.i..m.a.h.
GROUP_ICON 009B 0418	00067CB0	6F 00 00 00 1A 00 50 00 75 00 62 00 69 00 62 00	o.....P.u.b.i.b.
ICON 0016 0418	00067CC0	6F 00 66 00 75 00 68 00 20 00 6D 00 75 00 64 00	o.f.u.h..m.u.d.
ICON 0017 0418	00067CDC	65 00 20 00 6C 00 65 00 6A 00 75 00 6A 00 6F 00	e..l.e.j.u.j.o.
ICON 0018 0418	00067CE0	78 00 6F 00 70 00 65 00 6E 00 00 40 00 57 00	x.o.p.e.n..@W.
ICON 0019 0418	00067CF0	6F 00 6C 00 65 00 6D 00 75 00 79 00 69 00 74 00	o.l.e.m.u.y.i.t.
ICON 001A 0418	00067D00	6F 00 66 00 6F 00 62 00 20 00 62 00 61 00 6A 00	o.f.o.b..b.a.j.
ICON 001B 0418	00067D10	65 00 77 00 6F 00 20 00 68 00 75 00 62 00 61 00	e.w.o..h.u.b.a.
ICON 001C 0418	00067D20	6D 00 61 00 67 00 75 00 6A 00 69 00 20 00 73 00	m.a.g.u.j.i..s.
ICON 001D 0418	00067D30	61 00 7A 00 65 00 66 00 69 00 6C 00 65 00 6E 00	a.z.e.f.i.l.e.n.
GROUP_ICON 009F 0418	00067D40	20 00 7A 00 65 00 67 00 69 00 64 00 75 00 76 00	z.e.g.i.d.u.v.
CURSOR 001E 0000	00067D50	20 00 64 00 69 00 6B 00 75 00 78 00 20 00 72 00	.d.i.k.u.x..r.
CURSOR 001F 0000	00067D60	6F 00 77 00 75 00 7A 00 6F 00 63 00 69 00 00 00	o.w.u.z.o.c.i...
CURSOR 0020 0000	00067D70	00 00 00 00 33 00 53 00 6F 00 64 00 20 00 68 00	...3.S.o.d..h.
GROUP_CURSOR 094E 0000	00067D80	69 00 6B 00 75 00 64 00 6F 00 6C 00 65 00 72 00	i.k.u.d.o.l.e.r.
CURSOR 0021 0000	00067D90	20 00 7A 00 69 00 6A 00 6F 00 77 00 75 00 66 00	.z.i.j.o.w.u.f.
CURSOR 0022 0000	00067DAO	69 00 6A 00 75 00 6A 00 6F 00 7A 00 75 00 20 00	i.j.u.j.o.z.u..
GROUP_CURSOR 0953 0000	00067DB0	74 00 75 00 76 00 65 00 20 00 76 00 61 00 70 00	t.u.v.e..v.a.p.
CURSOR 0023 0000	00067DC0	20 00 76 00 6F 00 6A 00 75 00 66 00 65 00 6D 00	.v.o.j.u.f.e.m.
CURSOR 0024 0000	00067DD0	65 00 7A 00 61 00 73 00 65 00 6B 00 34 00 5A 00	e.z.a.s.e.k.4.Z.
CURSOR 0025 0000	00067DE0	6F 00 76 00 61 00 76 00 65 00 6C 00 61 00 68 00	o.v.a.v.e.l.a.h.
GROUP_CURSOR 0954 0000	00067DF0	6F 00 68 00 6F 00 72 00 61 00 63 00 20 00 79 00	o.h.o.r.a.c..y.
VERSION 0001 0000	00067E00	6F 00 72 00 61 00 20 00 73 00 6F 00 63 00 69 00	o.r.a..s.o.c.i.
DIALOG 0134 0000	00067E10	6B 00 61 00 6D 00 65 00 79 00 75 00 68 00 69 00	k.a.m.e.y.u.h.i.
STRING 0011 0418			

The ICON, GROUP, CURSOR, VERSION, and STRING include information for the pFile, Raw Data, and Value. All of these sections are basically just hex dumps with ASCII values under the Value section. Since they are in hex, I am unable to read information and data.

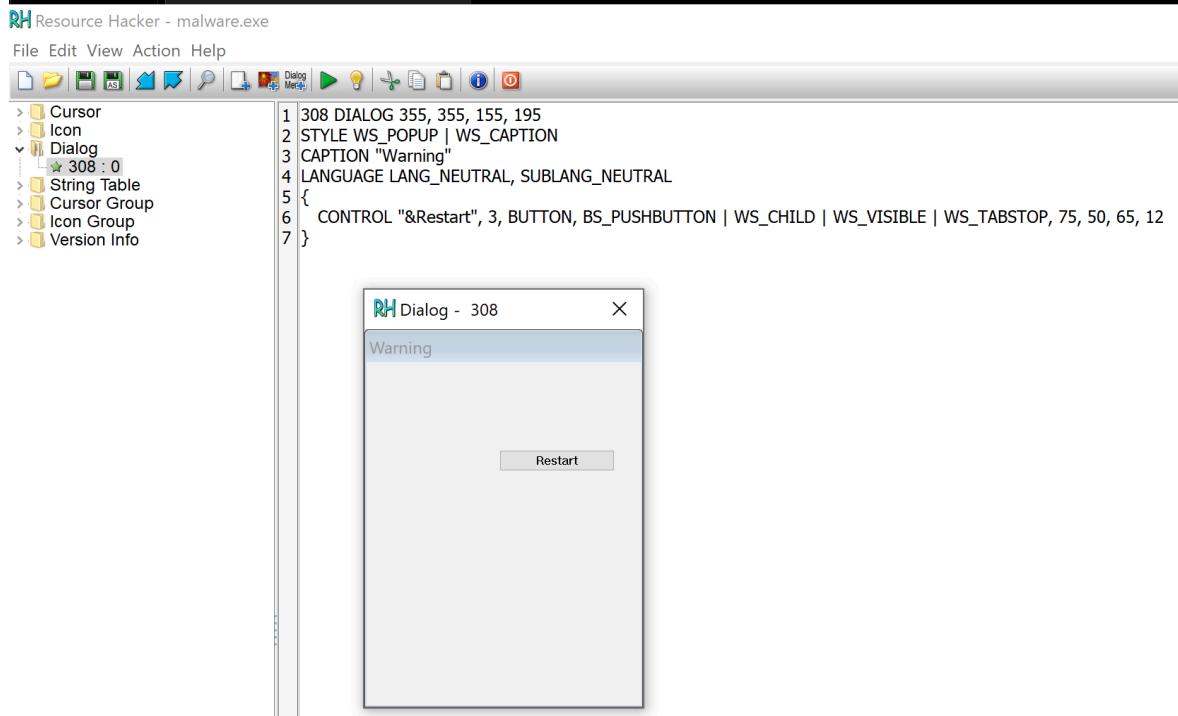
Overall, PEview showed me a lot of useful information regarding the malware I am attempting to reverse. While looking through PEview, I found a table of all of the functions used by this malware, as well as all of the dll files used. This is very helpful, as it allows me to understand more about Lumma Stealer and what exactly it does. I found icons and strings in the form of hex dumps. Due to these icons and strings being in hex, I am unable to read them and see exactly what they are or do. A majority of the information in PEview, however, was in the form of hex dumps which made it unreadable. Since this information is in hex, I can use Resource Hacker to help me make more sense of this information. I will now use Resource Hacker to view the information and dive even deeper into this malware and what it does.

Resource Hacker:

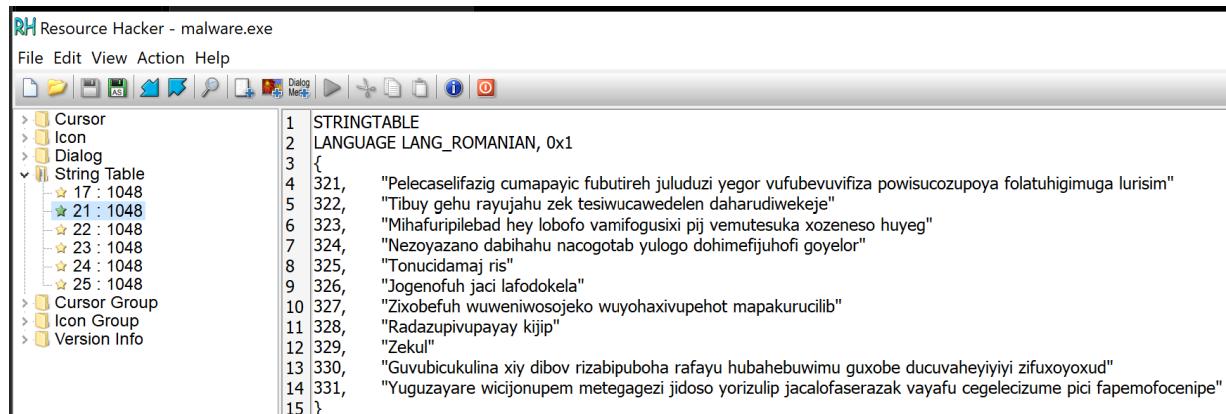
The screenshot above shows that I have opened the malware executable in Resource Hacker. Once opened, I notice that the folders it shows in Resource Hacker are the same subsections that were shown in PEview under the SECTION.rsrc section. Below I included a screenshot from each section.



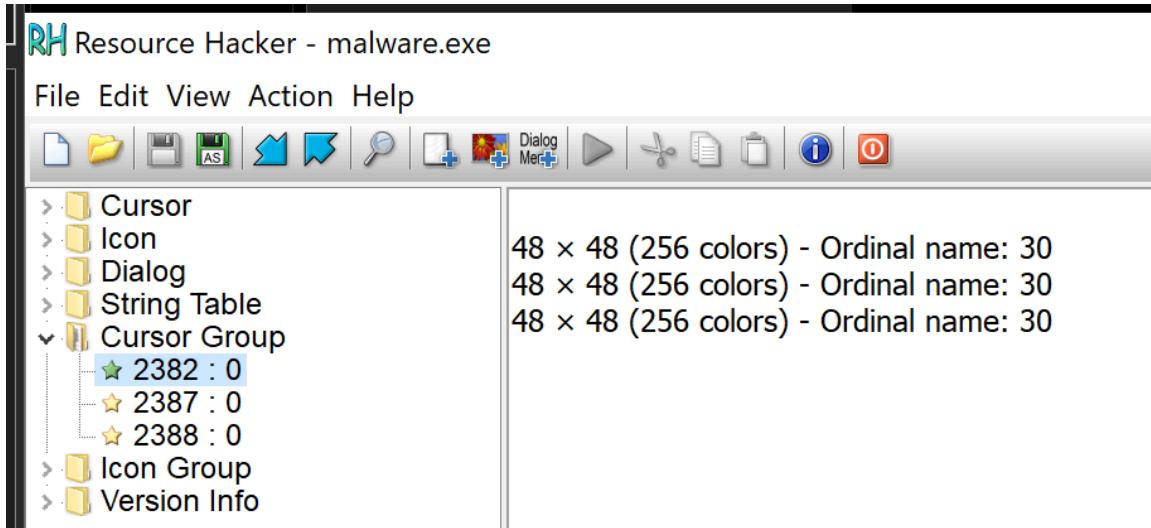
The screenshot above is a screenshot of one of the files in the Icon folder. Each file in this folder has different images of cubes that are all different shades of gray.



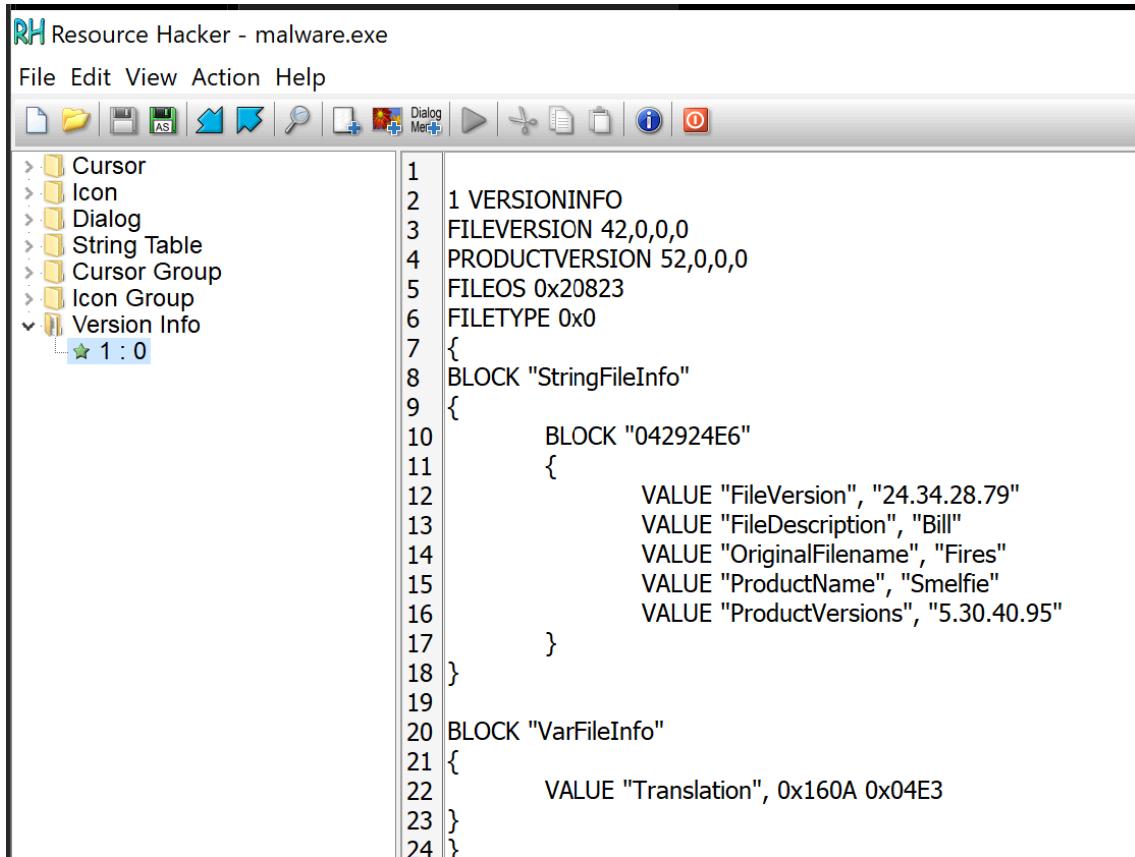
This screenshot shows the file in the Dialog folder. In this file is what seems to be code. This code creates a pop-up with a warning message and a restart button.



This screenshot is of a file in the String Table folder. All of these files in this folder seem to be in Romanian. Each line in the files is a new string, or something new written in Romanian.



The files in the Cursor Group folder seem to have a “number x number” which correlates to a certain number of colors and an ordinal name.

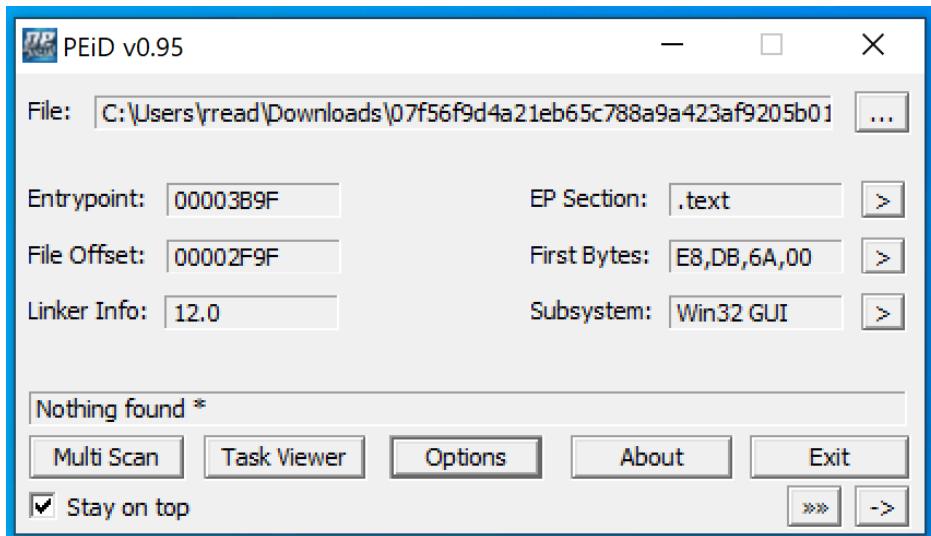


The file in the Version info as shown above gives information regarding the version of a file. It gives a value for the version, description, filename, product name, and product version.

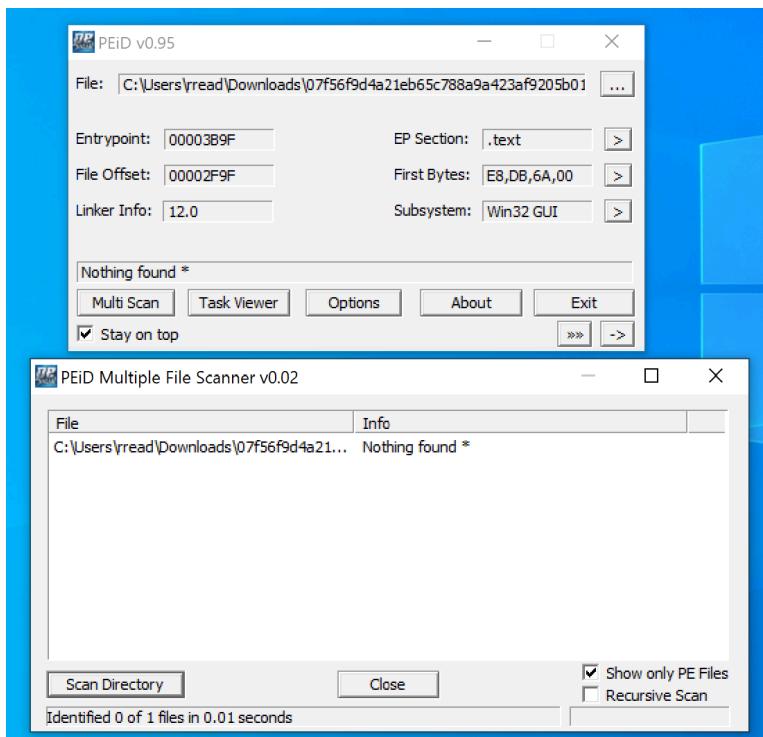
Overall when looking at what Resource Hacker provides, it seems to go more in depth about certain files that are unreadable from PEview. Some files in PEview were just hex dumps with

random ASCII values printed to the side. In Resource Hacker, you can now see those hex dumps in readable code and images. For example, you are now able to see icons instead of hex, dialog or pop-up boxes, strings in what seem to be foreign languages, and other code. This allows for a better understanding of what these files entail, and what this malware is capable of.

PEiD:



The above screenshot shows my malware uploaded to PEiD. On the main screen PEiD gives some information regarding Lumma Stealer such as file offset, first bytes, subsystem and more. PEiD also allows you to scan the file that you upload to see if anything is found regarding the compiler used.



The above screenshot shows me running a scan on the malware executable that I uploaded to PEiD. I ran the scan multiple times and every time it stated nothing was found. This could be because the threat actor that wrote this malware took precautions in hiding or burying what they didn't want to be found, or it could just be that PEiD is unsure of the compiler that was used.

The information provided from PEiD was somewhat helpful. I did get some new information I hadn't found before. One thing that was super helpful in PEiD was the additional information it provided for EP Section, First Bytes, and Subsystem. The additional information for First Bytes gave me code in assembly and also allowed me to see the strings that were used in this code and where they were used. This furthered my understanding regarding the malware.

Dependency Walker:

The screenshot shows the Dependency Walker interface with the following details:

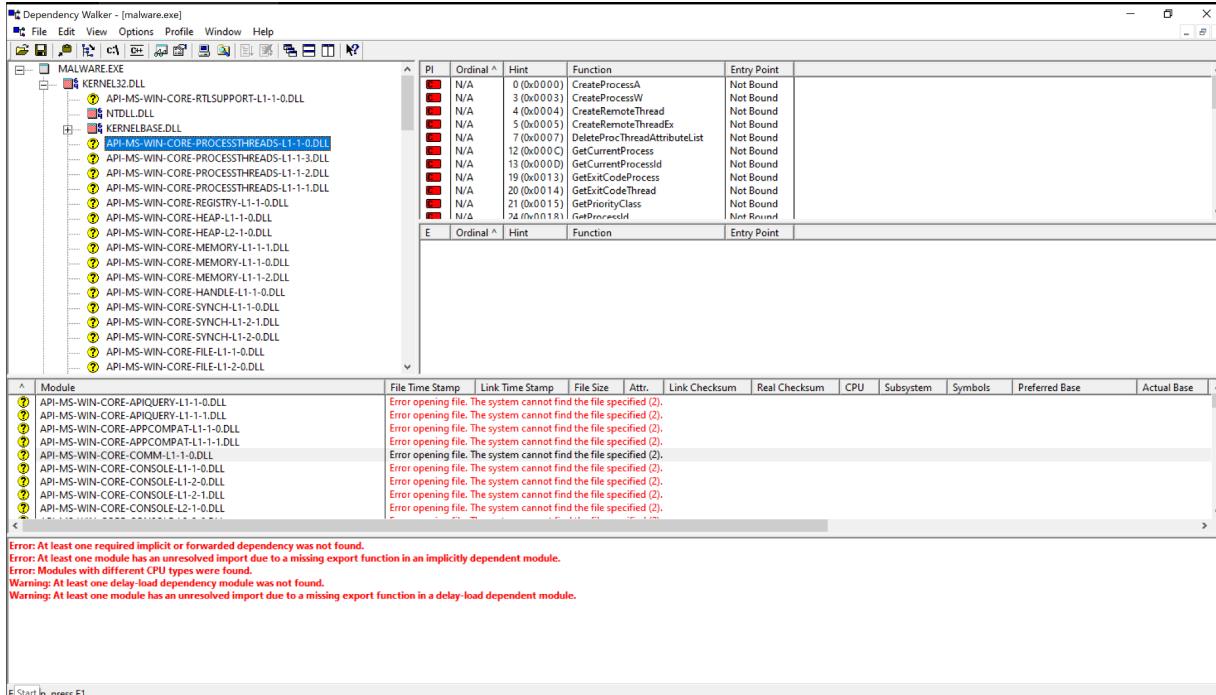
- File Menu:** File, Edit, View, Options, Profile, Window, Help.
- Toolbar:** Standard file operations like Open, Save, Print, etc.
- Left pane:** A tree view showing the structure of the executable. It includes sections for **MALWARE.EXE**, **KERNEL32.DLL**, **GDI32.DLL**, **ADVAPI32.DLL**, and **WINHTTP.DLL**.
- Table 1: Imports by Ordinal**

PI	Ordinal ^	Hint	Function	Entry Point
	N/A	3 (0x0003)	AddAtomA	Not Bound
	N/A	5 (0x0005)	AddConsoleAliasA	Not Bound
	N/A	6 (0x0006)	AddConsoleAliasW	Not Bound
	N/A	82 (0x052)	CloseHandle	Not Bound
	N/A	117 (0x075)	CopyFileW	Not Bound
	N/A	143 (0x08F)	CreateFileW	Not Bound
	N/A	144 (0x090)	CreateHardLinkA	Not Bound
	N/A	189 (0x0BD)	CreateTimerQueueTimer	Not Bound
	N/A	202 (0x0CA)	DecodePointer	Not Bound
	798 (0x0111)		UpdateVirtualSection	Not Bound
- Table 2: Imports by Hint**

E	Ordinal ^	Hint	Function	Entry Point
	1 (0x0001)	0 (0x0000)	AcquireSRWLockExclusive	NTDLL!RtlAcquireSRWLockExclusive
	2 (0x0002)	1 (0x0001)	AcquireSRWLockShared	NTDLL!RtlAcquireSRWLockShared
	3 (0x0003)	2 (0x0002)	ActivateActCtx	0x00020390
	4 (0x0004)	3 (0x0003)	ActivateActCbWorker	0x0001BA10
	5 (0x0005)	4 (0x0004)	AddAtomA	0x00059230
	6 (0x0006)	5 (0x0005)	AddAtomW	0x000128F0
	7 (0x0007)	6 (0x0006)	AddConsoleAliasA	0x00025950
	8 (0x0008)	7 (0x0007)	AddConsoleAliasW	0x00025960
	9 (0x0009)	8 (0x0008)	AddDllDirectory	api-ms-win-core-libraryloader-1-1-0.AddDllDirectory
- Table 3: Errors**

Module	File Time Stamp	Link Time Stamp	File Size	Attr.	Link Checksum	Real Checksum	CPU	Subsystem	Symbols	Preferred Base	Actual Base
API-MS-WIN-CORE-APIQUERY-L1-1-0.DLL	Error opening file. The system cannot find the file specified (2).										
API-MS-WIN-CORE-APIQUERY-L1-1-1.DLL	Error opening file. The system cannot find the file specified (2).										
API-MS-WIN-CORE-APPCOMPAT-L1-1-0.DLL	Error opening file. The system cannot find the file specified (2).										
API-MS-WIN-CORE-APPCOMPAT-L1-1-1.DLL	Error opening file. The system cannot find the file specified (2).										
API-MS-WIN-CORE-COMM-L1-1-0.DLL	Error opening file. The system cannot find the file specified (2).										
API-MS-WIN-CORE-CONSOLE-L1-1-0.DLL	Error opening file. The system cannot find the file specified (2).										
API-MS-WIN-CORE-CONSOLE-L1-2-0.DLL	Error opening file. The system cannot find the file specified (2).										
API-MS-WIN-CORE-CONSOLE-L1-2-1.DLL	Error opening file. The system cannot find the file specified (2).										
API-MS-WIN-CORE-CONSOLE-L2-1-0.DLL	Error opening file. The system cannot find the file specified (2).										
- Log Messages:**
 - Error: At least one required implicit or forwarded dependency was not found.
 - Error: At least one module has an unresolved import due to a missing export function in an implicitly dependent module.
 - Error: Modules with different CPU types were found.
 - Warning: At least one delay-load dependency module was not found.
 - Warning: At least one module has an unresolved import due to a missing export function in a delay-load dependent module.

The screenshot above shows my malware executable opened in Dependency Walker. Dependency Walker shows the .DLL files that are used in Lumma Stealer as well as the API's and functions used in those .DLL files. There is also information regarding errors where some files may be missing.



This screenshot is very similar to the last one, however, I opened an API that was in one of the .DLL files instead of just looking at the .DLL file itself.

Overall, Dependency Walker was very helpful and provided good information regarding the malware I am using. It provided me with the .DLL files that were used. These include KERNEL32.DLL, GDI32.DLL, ADVAPI32.DLL, and WINHTTP.DLL. It also goes in depth about the .DLL files used within these .DLL files, and the API's used within them as well. For each .DLL and API, it gave all the functions used within them. It also gave entry points, offsets and what seemed to be line numbers for each function. Reviewing these .DLL, API, and functions gave me a very good idea of what Lumma Stealer does and what it can be used for. There were a lot of functions including HTTP as well as getter and setter functions. Dependency Walker also listed errors when it noticed files were missing from certain API's.

VirusTotal:

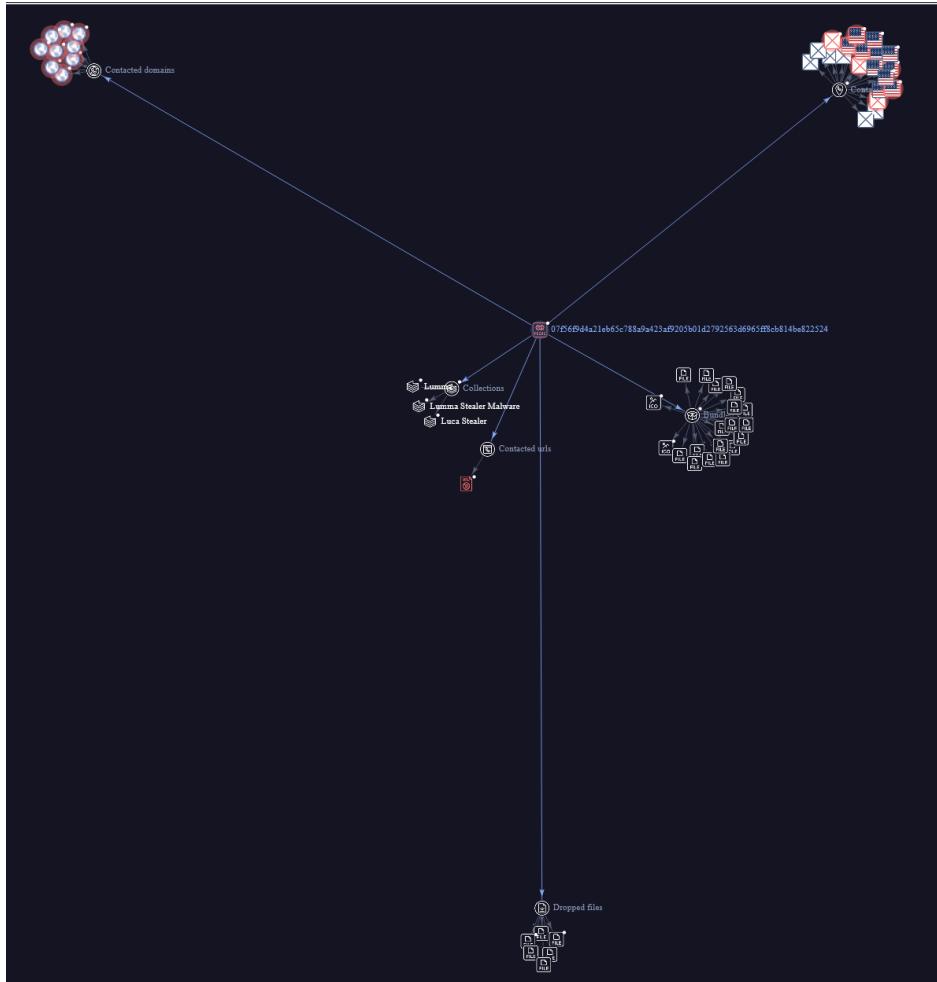
The screenshot shows the VirusTotal analysis interface for a file named "malware.exe". The main summary indicates that 60 out of 72 security vendors flagged the file as malicious. The file is a PE executable (EXE) file, 421.00 KB in size, and was last modified a moment ago. The "Community Score" is shown as 60/72. Below the summary, there are tabs for DETECTION, DETAILS, RELATIONS, BEHAVIOR, TELEMETRY, and COMMUNITY (16). The DETECTION tab is selected, showing "Dynamic Analysis Sandbox Detections" which include flags from Zenbox, VMRay, and C2AE. The DETAILS tab shows popular threat labels (trojan.tepfer/stealer), threat categories (troyan, ransomware), and family labels (tepfer, stealer, convagent). The SECURITY VENDORS' ANALYSIS section lists vendor findings: AhnLab-V3 found a Trojan/Win.CrypterX.gen.R645837; AliCloud found a Trojan[stealer]Win/Tepfer.gyf; Antiy-AVL found a Trojan/Win32.Convagent; and Avast found a Win32.CrypterX.gen[Tri]. The page also includes options to reanalyze or find similar files.

The screenshot above shows that I opened the malware I am using in VirusTotal. Once I opened it and VirusTotal checked the hash, I was able to see that a majority of security vendors and sandboxes flagged the file as malicious. To be more specific, 60 out of 72 vendors, and 3 sandboxes found the file to be malicious.

VirusTotal has multiple tabs you can click on to learn more about the file that was uploaded. When I went through the detection tab, I was able to see what vendors thought the file was malicious and what vendors thought it was safe.

In the details tab, I am able to see a bunch of information. Some of which I already know from previous tools and some new information. I am able to see things such as properties of the file which include the hashes and file type. I am also able to see the previous names, and history of the file. Lastly in this section I can see executable information such as compilers used, header, sections of data, and imports.

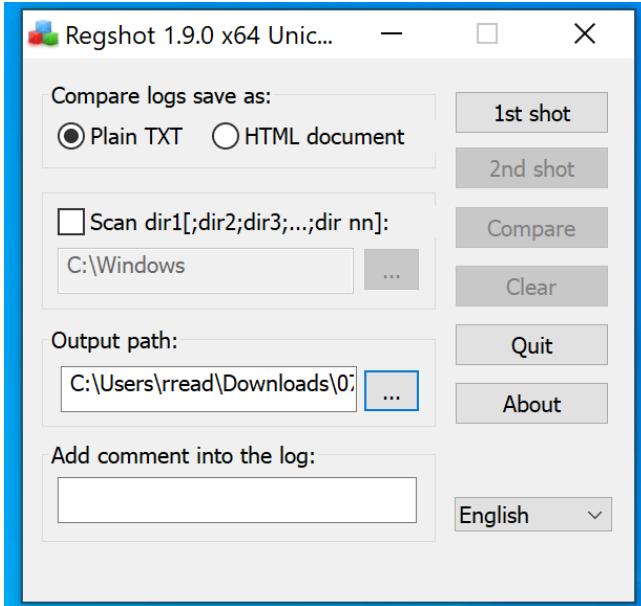
While looking into the relations tab I noticed some contacted URLs, domains and IP addresses that Lumma Stealer has probably been used on. I am also able to see a section of bundled and deleted files, however there is not a lot of information in these sections. Lastly in this section I can open a graph summary to show all the relations of this malware. The graph summary screenshot is shown below.



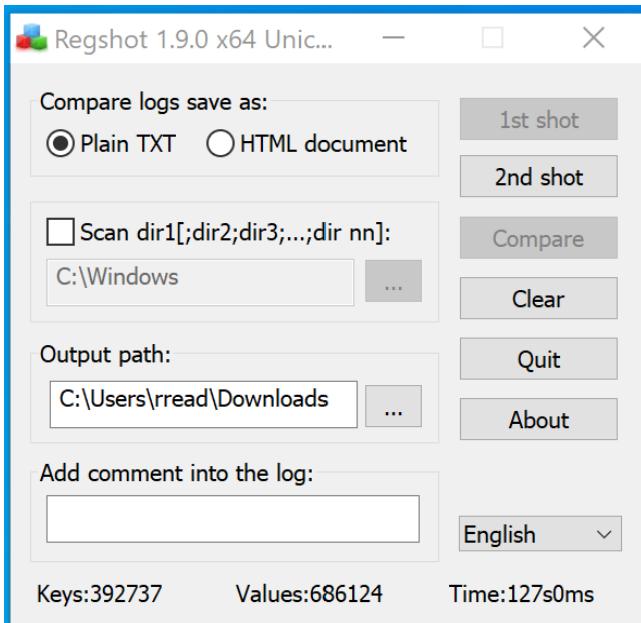
When I moved onto the behavior tab, I found some interesting information regarding the behavior of the malware. The three main things that were listed first in the behavior tab was that this malware checks user input, detects debug environments, and sleeps for long periods of time. It goes more in depth in these behaviors as well as their capabilities. After this was listed, it went on to the network communication of this malware. After the network communication, there was information regarding the file system, registry, and process actions. It was very interesting to see what Lumma Stealer can do over a network and on someone's device.

Basic Dynamic Analysis:

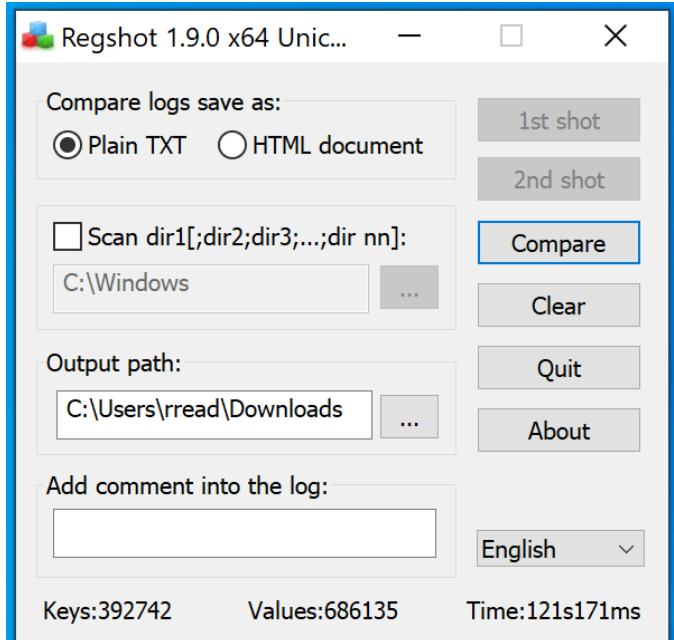
Regshot:



Above is a screenshot of me making my first shot in Regshot. I set the compare logs to plain text and I set the output path to my Downloads folder since that is where the malware file is located.



The above screenshot is me taking the second shot in Regshot. Between shots 1 and 2, I ran the malware executable file. This way when I compare them, I am able to see what exactly Lumma Stealer did.



```

~res-x64.txt - Notepad
File Edit Format View Help
Regshot: 1.9.0 x64 Unicode
Comments:
Datetime: 2024/4/27 22:45:08 , 2024/4/27 22:49:23
Computer: DESKTOP-F03RA1S , DESKTOP-F03RA1S
Username: rread , rread

-----
Keys added: 5
-----
HKLM\Software\Microsoft\SystemCertificates\AuthRoot\Certificates\DAC9024F54D8F6DF94935FB1732638CA6AD77C13
HKLM\Software\Microsoft\Windows\Windows Error Reporting\TermReason\7584
HKLM\Software\Microsoft\Windows\Windows Error Reporting\TermReason\9496
HKLM\Software\WOW6432Node\Microsoft\IdentityCRL\ClockData
HKLM\Software\WOW6432Node\Microsoft\SystemCertificates\AuthRoot\Certificates\DAC9024F54D8F6DF94935FB1732638CA6AD77C13

-----
Values deleted: 1
-----
HKLM\Software\WOW6432Node\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\AmiOverridePath: "C:\Windows\AppCompat\Programs\Amcache.hve.tmp"

-----
Values added: 12
-----
HKLM\Software\Microsoft\SystemCertificates\AuthRoot\Certificates\DAC9024F54D8F6DF94935FB1732638CA6AD77C13\Blob: 5C 00 00 00 01 00 00 00 04 00 00 00 00 00 00 08 00 00 19 00 00 00 01 00 00
42 00 00 00 30 08 08 2B 01 05 07 03 02 06 0A 2B 01 04 01 82 37 0A 03 0C 06 0A 2B 01 04 01 82 37 0A 03 04 06 08 2B 01 05 07 03 04 06 08 2B 06 01 05 07 03 01
03 13 0E 44 53 54 20 52 6F 74 20 43 41 20 58 33 30 82 01 22 30 0D 06 09 2A 86 48 86 F7 0D 01 01 05 03 02 01 0F 00 30 82 01 0A 02 82 01 01 00 DF AF E9 97 50 08 83 57 84 CC 6
E 06 03 55 1D 0F 01 FF 04 03 02 01 06 30 1D 0E 03 55 1D 0E 04 16 04 14 C4 A7 B1 A4 78 2C 71 FA DB E1 4B 90 75 FF C4 15 60 85 89 10 30 0D 06 09 2A 86 48 86 F7 0D 01 01 05 05 00
HKLM\Software\Microsoft\Windows Error Reporting\TermReason\7584\Terminator: "WerSvcKernelMsgDone"
HKLM\Software\Microsoft\Windows Error Reporting\TermReason\9496\Reason: 0x00000001
HKLM\Software\Microsoft\Windows Error Reporting\TermReason\9496\CreationTime: 30 86 46 F3 F4 98 DA 01
HKLM\Software\Microsoft\Windows Error Reporting\TermReason\9496\Terminator: "WerSvcKernelMsgDone"
HKLM\Software\Microsoft\Windows Error Reporting\TermReason\9496\Reason: 0x00000001
HKLM\Software\Microsoft\Windows Error Reporting\TermReason\9496\CreationTime: D6 D1 98 0A F5 98 DA 01
HKLM\Software\WOW6432Node\Microsoft\IdentityCRL\ClockData\ClockTimeSeconds: C4 80 2D 66 00 00 00 00
HKLM\Software\WOW6432Node\Microsoft\SystemCertificates\AuthRoot\ClockData\TickCount: 18 3E 23 00 00 00 00 00
HKLM\Software\WOW6432Node\Microsoft\SystemCertificates\AuthRoot\Certificates\DAC9024F54D8F6DF94935FB1732638CA6AD77C13\Blob: 5C 00 00 00 01 00 00 00 04 00 00 00 00 00 00 08 00 00 19 00 00
01 00 00 00 42 00 00 30 00 08 2B 01 05 07 03 02 06 0A 2B 01 04 01 82 37 0A 03 04 06 08 2B 01 05 07 03 04 06 08 2B 06 01 05
06 03 55 04 03 13 0E 44 53 54 20 52 6F 6F 74 20 43 41 20 58 33 30 82 01 22 30 0D 06 09 2A 86 48 86 F7 0D 01 01 05 03 02 01 0F 00 30 82 01 0A 02 82 01 01 00 DF AF E9 97 50 08 8
1 01 FF 30 0E 06 03 55 1D 0F 01 FF 04 03 02 01 06 30 1D 06 03 55 1D 0E 04 16 04 14 C4 A7 B1 A4 78 2C 71 FA DB E1 4B 90 75 FF C4 15 60 85 89 10 30 0D 06 09 2A 86 48 86 F7 0D 01
HKU\S-1-5-21-2042309985-658805882-381374168-1000\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\P:\Vlref\veernq\Objaybnqf\
HKU\S-1-5-21-2042309985-658805882-381374168-1000\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatibility Assistant\Store\%C:\Users\rread\Downloads\07f56f9d4a21eb65c

-----
Values modified: 43
-----
```

The two screenshots above show the comparison between the first and second shot in Regshot. While looking at what was modified, I realized that the HKLM registry was deleted, and some other interesting HKLM registry keys were added. I also realized that there were long values that seemed to be in hex that were added or modified on my system. As I looked into what these hex values went to, I realized that they were values that were added or modified for weird looking functions and weird paths to different directories. As I looked deeper into these directories, I realized that Microsoft Explorer was opened, and information is being gathered about this browser.

Overall, Regshot was very helpful because it allowed me to see what Lumma Stealer is doing. I can now see that this malware deletes and adds registry keys on the device. It also runs functions in the background that gather information about the device as well as data on browsers and browser extensions.

Process Monitor:

Time ...	Process Name	PID	Operation	Path	Result	Detail
10:11:...	Explorer.EXE	3876	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: Name
10:11:...	Explorer.EXE	3876	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: HandleTag...
10:11:...	Explorer.EXE	3876	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: HandleTag...
10:11:...	Explorer.EXE	3876	RegOpenKey	HKCU\Software\Classes\Applications\...	NAME NOT FOUND Desired Access: R...	
10:11:...	Explorer.EXE	3876	RegOpenKey	HKCR\Applications\Procmon64.exe	NAME NOT FOUND Desired Access: R...	
10:11:...	Explorer.EXE	3876	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: Name
10:11:...	Explorer.EXE	3876	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: HandleTag...
10:11:...	Explorer.EXE	3876	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: Name
10:11:...	Explorer.EXE	3876	RegOpenKey	HKCU\Software\Classes\Applications\...	NAME NOT FOUND Desired Access: R...	
10:11:...	Explorer.EXE	3876	RegOpenKey	HKCR\Applications\Procmon64.exe	NAME NOT FOUND Desired Access: R...	
10:11:...	Explorer.EXE	3876	CreateFile	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	Desired Access: R...
10:11:...	Explorer.EXE	3876	QueryBasicInfor...	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	CreationTime: 4/29...
10:11:...	Explorer.EXE	3876	CloseFile	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	
10:11:...	Explorer.EXE	3876	RegQueryKey	HKLM	SUCCESS	Query: Name
10:11:...	Explorer.EXE	3876	RegOpenKey	HKLM\Software\Microsoft\Windows\C...	SUCCESS	Desired Access: Q...
10:11:...	Explorer.EXE	3876	RegQueryValue	HKLM\SOFTWARE\Microsoft\Window...	SUCCESS	Type: REG_DWO...
10:11:...	Explorer.EXE	3876	RegCloseKey	HKLM\SOFTWARE\Microsoft\Window...	SUCCESS	
10:11:...	Explorer.EXE	3876	RegQueryKey	HKCU	SUCCESS	Query: Name
10:11:...	Explorer.EXE	3876	RegOpenKey	HKCU\Software\Microsoft\Windows\C...	SUCCESS	Desired Access: Q...
10:11:...	Explorer.EXE	3876	RegQueryValue	HKCU\SOFTWARE\Microsoft\Window...	SUCCESS	Type: REG_DWO...
10:11:...	Explorer.EXE	3876	RegCloseKey	HKCU\SOFTWARE\Microsoft\Window...	SUCCESS	
10:11:...	Explorer.EXE	3876	CreateFile	C:\Users\vread\AppData\Local\Micros...	SUCCESS	Desired Access: R...
10:11:...	Explorer.EXE	3876	QueryBasicInfor...	C:\Users\vread\AppData\Local\Micros...	SUCCESS	CreationTime: 3/19...
10:11:...	Explorer.EXE	3876	CloseFile	C:\Users\vread\AppData\Local\Micros...	SUCCESS	
10:11:...	Explorer.EXE	3876	QueryStandardI...	C:\Users\vread\AppData\Local\Micros...	SUCCESS	AllocationSize: 8,1...
10:11:...	Explorer.EXE	3876	CloseFile	C:\Users\vread\AppData\Local\Micros...	SUCCESS	
10:11:...	Explorer.EXE	3876	CreateFile	C:\Users\vread\AppData\Local\Micros...	SUCCESS	Desired Access: G...
10:11:...	Explorer.EXE	3876	<Unknown>	C:\Users\vread\AppData\Local\Micros...	SUCCESS	
10:11:...	Explorer.EXE	3876	QueryStandardI...	C:\Users\vread\AppData\Local\Micros...	SUCCESS	AllocationSize: 8,1...
10:11:...	Explorer.EXE	3876	CreateFileMapp...	C:\Users\vread\AppData\Local\Micros...	FILE LOCKED WI...	SyncType: SyncTy...
10:11:...	Explorer.EXE	3876	QueryStandardI...	C:\Users\vread\AppData\Local\Micros...	SUCCESS	AllocationSize: 8,1...
10:11:...	Explorer.EXE	3876	CreateFileMapp...	C:\Users\vread\AppData\Local\Micros...	SUCCESS	SyncType: SyncTy...
10:11:...	svchost.exe	3040	LockFile	C:\ProgramData\Microsoft\Windows\A...	SUCCESS	Desired Access: R...
10:11:...	Explorer.EXE	3876	QueryBasicInfor...	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	Exclusive: False, O...
10:11:...	Explorer.EXE	3876	CloseFile	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	CreationTime: 4/29...
10:11:...	Explorer.EXE	3876	QueryStandardI...	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	
10:11:...	svchost.exe	3040	UnlockFileSingle	C:\ProgramData\Microsoft\Windows\A...	SUCCESS	Offset: 124, Length...
10:11:...	Explorer.EXE	3876	CreateFile	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	Desired Access: R...
10:11:...	Explorer.EXE	3876	QueryBasicInfor...	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	CreationTime: 4/29...
10:11:...	Explorer.EXE	3876	CloseFile	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	
10:11:...	Explorer.EXE	3876	CreateFile	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	Desired Access: G...
10:11:...	Explorer.EXE	3876	QueryBasicInfor...	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	SyncType: SyncTy...
10:11:...	Explorer.EXE	3876	CloseFile	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	SyncType: SyncTy...
10:11:...	Explorer.EXE	3876	CreateFile	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	Desired Access: G...
10:11:...	Explorer.EXE	3876	QueryBasicInfor...	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	SyncType: SyncTy...
10:11:...	Explorer.EXE	3876	CloseFile	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	SyncType: SyncTy...
10:11:...	Explorer.EXE	3876	CreateFileMapp...	C:\Users\vread\AppData\Local\Temp\...	FILE LOCKED WI...	SyncType: SyncTy...
10:11:...	Explorer.EXE	3876	CreateFileMapp...	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	SyncType: SyncTy...
10:11:...	Explorer.EXE	3876	CloseFile	C:\Users\vread\AppData\Local\Temp\...	SUCCESS	

When I opened Process Monitor, I decided to run the malware that I am using. When I did this, I noticed a ton of Explorer.exe processes pop-up, as well as some Windows executables that are used for virus protection, and error logging. I decided to add filters to narrow down the long list of processes. I included a screenshot below showing the filters I implemented.

The screenshot shows the Process Monitor application interface. On the left, a main window displays a log of registry operations. The columns include Time, Process Name, PID, Operation, Path, Result, and Detail. Most entries show Explorer.EXE performing various registry queries and operations like RegOpenKey, RegQueryValue, CreateFile, and CloseFile on keys such as HKCU\Software\Classes, HKLM\Software\Microsoft\Windows\CurrentVersion\Run, and C:\Users\reade\AppData\Local\Temp. The 'Result' column often shows 'SUCCESS' or 'NAME NOT FOUND'. The 'Detail' column provides specific details like 'Query: Name' or 'Desired Access: R...'. On the right, a smaller window titled 'Process Monitor Filter' is open, showing a list of conditions used to filter the results. It includes filters for 'Architecture', 'Company' (Microsoft Corporation), 'User' (DESKTOP-F03RA1S\reade), 'Process Name' (Procmn.exe, Procepx.exe, Autonins.exe), and 'Result' (is not Unknown). The 'Action' column indicates whether each filter is included or excluded.

I implemented three filters. When implementing these filters I decided to make them a little more broad so I wouldn't filter out any important processes.

The first filter I implemented was if the user is read then include. I did this because this user is me, and I am the one that ran the malware, so I only wanted to see processes used under this user.

The next filter I used was if the company is Microsoft Corporation then include. I did this because every time I ran this malware, in this tool and others, Explorer.EXE always opens and there are operations done on it. Since this specific malware targets browsers, I decided to include the company that owns this browser.

The last filter I used was if the result is not unknown then include. I did this because when I was looking through the processes before I implemented the filters, I saw most of the results were success with some being name not found. I didn't notice any processes that went along with this malware that had the result of unknown. Therefore, I decided to include everything that was not a result of unknown.

After implementing all of these filters, I went back and looked through the processes. At first I noticed that Explorer.EXE was listed a majority of the time. I noticed for these processes that there were operations that queried registry keys, and a lot of create, call, and query operations used in Explorer.EXE.

As I kept scrolling down, I noticed WerFault.exe was a running process. I researched what exactly this executable is. Apparently it is used for Windows error reporting, however, hackers use this executable to deploy malware.

Looking further, I noticed that backgroundtaskhost.exe was running multiple processes. These processes all had operations on registry keys. When I looked at the properties of some of these processes, they all had weird looking command lines. It was all just strings of random letters. This seemed very odd.

After that, I noticed processes from RuntimeBroker.exe. I did some research on this executable as well and found that it manages permissions for PC apps from the Microsoft store. I looked at the properties of these processes and saw that they all had to do with embedding.

Overall, looking into Process Monitor and using filters help me gain more information regarding Lumma Stealer. It allowed me to understand what is being done when Explorer.EXE is opened, and what processes are running in the background.

Process Explorer:

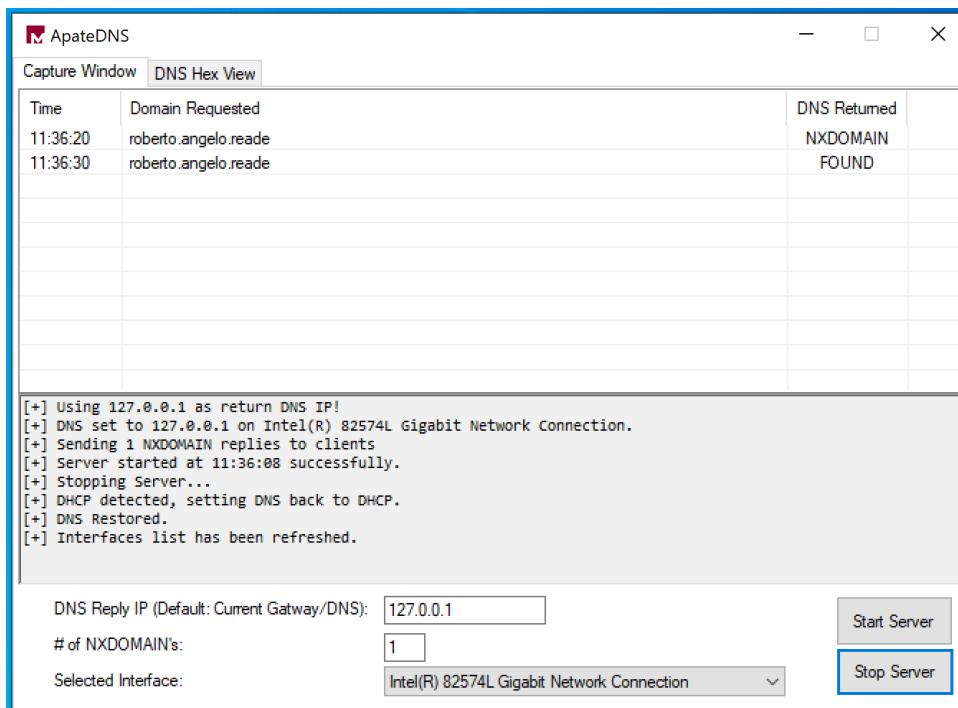
Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-F03RA1S]\read						
Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
Registry		4,552 K	69,368 K	92		
System Idle Process	97.79	60 K	8 K	0		
System	< 0.01	192 K	152 K	4		
csrss.exe		1,796 K	5,716 K	440		
wininit.exe		1,336 K	7,172 K	532		
services.exe		4,936 K	10,624 K	608		
svchost.exe		9,836 K	29,480 K	804 Host Process for Windows S... Microsoft Corporation		
WmiPrvSE.exe		11,748 K	22,700 K	3952		
dllhost.exe		3,140 K	11,232 K	5200		
StartMenuExperienceHost...		22,568 K	67,096 K	5588		
RuntimeBroker.exe		6,092 K	26,468 K	6024 Runtime Broker Microsoft Corporation		
RuntimeBroker.exe		7,308 K	30,600 K	5264 Runtime Broker Microsoft Corporation		
RuntimeBroker.exe		3,076 K	16,980 K	6712 Runtime Broker Microsoft Corporation		
SearchApp.exe	Susp...	153,132 K	226,404 K	3456 Search application Microsoft Corporation		
TextInputHost.exe		12,708 K	39,976 K	7688	Microsoft Corporation	
dllhost.exe		3,356 K	12,496 K	7928 COM Surrogate Microsoft Corporation		
ApplicationFrameHost.exe		7,212 K	40,504 K	6808 Application Frame Host Microsoft Corporation		
SecHealthUi.exe	Susp...	31,804 K	66,184 K	2528 Windows Defender application Microsoft Corporation		
SecurityHealthHost.exe		2,636 K	10,188 K	4528 Windows Security Health Host Microsoft Corporation		
MoUsCoreWorker.exe		15,784 K	31,660 K	3436		
SecurityHealthHost.exe		1,516 K	8,436 K	8308		
FileCoAuth.exe		3,672 K	18,796 K	8504 Microsoft OneDriveFile Co-A... Microsoft Corporation		
ShellExperienceHost.exe	Susp...	15,608 K	51,192 K	8704 Windows Shell Experience H... Microsoft Corporation		
RuntimeBroker.exe		2,732 K	18,604 K	8636 Runtime Broker Microsoft Corporation		
PhoneExperienceHost.exe		49,352 K	144,816 K	4940 Microsoft Phone Link Microsoft Corporation		
RuntimeBroker.exe		2,720 K	14,820 K	2660 Runtime Broker Microsoft Corporation		
SearchApp.exe	Susp...	292,512 K	331,204 K	1884 Search application Microsoft Corporation		
smartscreen.exe		9,504 K	27,664 K	6668 Windows Defender SmartScr... Microsoft Corporation		
WUDFHost.exe		2,092 K	8,256 K	888		
svchost.exe		7,920 K	17,112 K	972 Host Process for Windows S... Microsoft Corporation		
svchost.exe		2,232 K	8,524 K	276 Host Process for Windows S... Microsoft Corporation		
svchost.exe		1,292 K	5,760 K	1052 Host Process for Windows S... Microsoft Corporation		
svchost.exe		1,664 K	8,168 K	1072 Host Process for Windows S... Microsoft Corporation		
svchost.exe		1,588 K	11,488 K	1080 Host Process for Windows S... Microsoft Corporation		
svchost.exe		2,292 K	10,616 K	1088 Host Process for Windows S... Microsoft Corporation		
svchost.exe		1,792 K	12,644 K	1104 Host Process for Windows S... Microsoft Corporation		
svchost.exe		2,116 K	10,484 K	1128 Host Process for Windows S... Microsoft Corporation		
svchost.exe		1,472 K	6,448 K	1268 Host Process for Windows S... Microsoft Corporation		
svchost.exe		1,612 K	8,560 K	1380 Host Process for Windows S... Microsoft Corporation		
svchost.exe		6,292 K	16,300 K	1420 Host Process for Windows S... Microsoft Corporation		
svchost.exe		2,584 K	9,156 K	1440 Host Process for Windows S... Microsoft Corporation		
svchost.exe		15,108 K	18,908 K	1480 Host Process for Windows S... Microsoft Corporation		
svchost.exe		3,232 K	14,364 K	1496 Host Process for Windows S... Microsoft Corporation		
svchost.exe		1,892 K	8,548 K	1512 Host Process for Windows S... Microsoft Corporation		
svchost.exe		2,060 K	8,664 K	1572 Host Process for Windows S... Microsoft Corporation		
svchost.exe		4,604 K	9,216 K	1668 Host Process for Windows S... Microsoft Corporation		
csrss.exe	< U.01	18,512 K	68,8/b K	54U		
winlogon.exe		2,872 K	13,708 K	692		
explorer.exe	< 0.01	515,392 K	191,260 K	3876 Windows Explorer Microsoft Corporation		
SecurityHealthSystray.exe		1,740 K	9,608 K	4484 Windows Security notification... Microsoft Corporation		
vmtoolsd.exe	< 0.01	28,364 K	50,824 K	756 VMware Tools Core Service VMware, Inc.		
OneDrive.exe		49,540 K	105,376 K	1416 Microsoft OneDrive Microsoft Corporation		
procexp.exe		8,072 K	11,256 K	5448 Sysinternals Process Explorer Sysinternals - www.sysinte...		
procexp64.exe	1.43	31,276 K	51,888 K	1728 Sysinternals Process Explorer Sysinternals - www.sysinte...		
msedge.exe		43,924 K	97,736 K	3708 Microsoft Edge Microsoft Corporation		
msedge.exe		2,056 K	8,300 K	8864 Microsoft Edge Microsoft Corporation		
msedge.exe		10,500 K	26,228 K	1656 Microsoft Edge Microsoft Corporation		
msedge.exe		6,740 K	17,616 K	1304 Microsoft Edge Microsoft Corporation		
msedge.exe		7,688 K	21,432 K	7436 Microsoft Edge Microsoft Corporation		

After opening Process Explorer, I am able to see a majority of the processes I noticed in Process Monitor. Process Explorer helped me understand how much of the CPU these

processes are using as well as the amount of bytes. Process Explorer gives you the PID of each process and a description, however, it does not go into as much detail as Process Monitor. You are not able to see the path, the operations, or the stack for each process. Overall Process Explorer helped me understand how much memory each process was taking up. On the other hand, it did not further my knowledge of what this malware is capable of doing.

ApateDNS:

Since the Lumma Stealer does not do anything with the network or internet, I am unable to use ApateDNS on the malware. I did however start an ApateDNS server and do a couple of my own tests on it as shown below.



When I opened ApateDNS, I put in the loopback address for the DNS Reply IP, and I set the number of NXDOMAIN's to 1. I then started the server. As you can see above I tried to ping my name. The first time it returned an NXDOMAIN and the second time it found the DNS. This is because I specified the number of NXDOMAIN's to be 1.

```

Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rread>ping roberto.angelo.reade
Ping request could not find host roberto.angelo.reade. Please check the name and try again.

C:\Users\rread>ping roberto.angelo.reade

Pinging roberto.angelo.reade [127.0.0.1] with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\rread>

```

This screenshot above shows what happened when I tried to ping my name. The first time it could not find this host, and the second time I attempted to ping my name I got a response. Again, as stated above, this was due to the fact that I set the number of NXDOMAIN's to be 1.

```

C:\Users\rread>ipconfig /all

Windows IP Configuration

Host Name . . . . . : DESKTOP-F03RA1S
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : localdomain

Ethernet adapter Ethernet0:

Connection-specific DNS Suffix . : localdomain
Description . . . . . : Intel(R) 82574L Gigabit Network Connection
Physical Address. . . . . : 00-0C-29-62-55-4C
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::3fa3:5916:9746:300%3(Preferred)
IPv4 Address. . . . . : 192.168.115.128(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Monday, April 29, 2024 9:50:42 AM
Lease Expires . . . . . : Monday, April 29, 2024 12:06:08 PM
Default Gateway . . . . . : 192.168.115.2
DHCP Server . . . . . : 192.168.115.254
DHCPv6 IAID . . . . . : 100666409
DHCPv6 Client DUID. . . . . : 00-01-00-01-2D-8B-A3-2A-00-0C-29-62-55-4C
DNS Servers . . . . . : 127.0.0.1
Primary WINS Server . . . . . : 192.168.115.2

```

The screenshot above shows me entering the ipconfig /all command while the ApateDNS server is running. As you can see, the DNS server is set to the loopback address of 127.0.0.1. This is because in the server settings I set the DNS Reply IP to be the loopback address.

```
C:\Users\rread>netstat -an | more

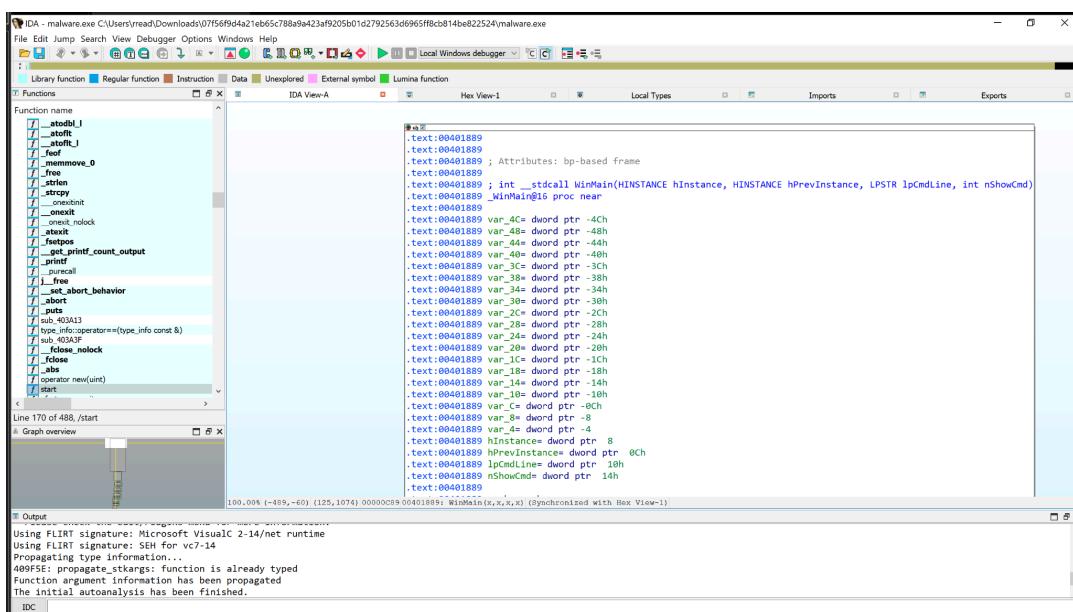
Active Connections

Proto  Local Address          Foreign Address        State
TCP    0.0.0.0:135            0.0.0.0:0             LISTENING
TCP    0.0.0.0:445            0.0.0.0:0             LISTENING
TCP    0.0.0.0:5040           0.0.0.0:0             LISTENING
TCP    0.0.0.0:7680           0.0.0.0:0             LISTENING
TCP    0.0.0.0:49664          0.0.0.0:0             LISTENING
TCP    0.0.0.0:49665          0.0.0.0:0             LISTENING
TCP    0.0.0.0:49666          0.0.0.0:0             LISTENING
TCP    0.0.0.0:49667          0.0.0.0:0             LISTENING
TCP    0.0.0.0:49668          0.0.0.0:0             LISTENING
TCP    0.0.0.0:49679          0.0.0.0:0             LISTENING
TCP    192.168.115.128:139   0.0.0.0:0             LISTENING
TCP    192.168.115.128:49687 52.159.126.152:443 ESTABLISHED
TCP    192.168.115.128:50096 184.87.60.157:80    TIME_WAIT
TCP    192.168.115.128:50099 204.79.197.200:443 ESTABLISHED
TCP    192.168.115.128:50100 20.42.73.25:443   ESTABLISHED
TCP    192.168.115.128:50101 13.107.237.254:443 ESTABLISHED
TCP    192.168.115.128:50102 4.150.240.254:443 ESTABLISHED
TCP    192.168.115.128:50103 13.107.226.40:443 CLOSE_WAIT
TCP    192.168.115.128:50104 204.79.197.222:443 ESTABLISHED
TCP    192.168.115.128:50105 184.28.190.72:443 ESTABLISHED
TCP    [::]:135              [::]:0               LISTENING
TCP    [::]:445              [::]:0               LISTENING
TCP    [::]:7680              [::]:0               LISTENING
TCP    [::]:49664             [::]:0               LISTENING
TCP    [::]:49665             [::]:0               LISTENING
TCP    [::]:49666             [::]:0               LISTENING
TCP    [::]:49667             [::]:0               LISTENING
TCP    [::]:49668             [::]:0               LISTENING
TCP    [::]:49679             [::]:0               LISTENING
TCP    UDP     0.0.0.0:53      *.*                LISTENING
```

The final step I did with ApateDNS was check to see if port 53 was present in the “netstat -an | more” command. In the screenshot above, you can see that it is in fact present. This means that this device can send traffic to the DNS server listening on port 53. If the ApateDNS server was not running, then port 53 would not be present in the output of the “netstat -an | more” command.

Advanced Static Analysis:

IDB:



The screenshot above shows me opening the malware in IDA. When I did this, I was brought right to the main function. In this function, there are many pointer variables set to hex values. The rest of the main function seems to just be a lot of add, sub, mul, and mov commands. Overall, the main function seems to be pretty basic. Below the main function are other functions that include if and for loops. All of these functions have a call to other functions. Screenshots of these functions are shown below.

```

.text:00401A29 push    ebx      ; dwNotifyFilter
.text:00401A2A push    ebx      ; bWatchSubtree
.text:00401A2B push    ebx      ; lpPathName
.text:00401A2C call    ds:FindFirstChangeNotificationW
.text:00401A32 push    ebx      ; lpCaData
.text:00401A33 push    ebx      ; CalType
.text:00401A34 push    ebx      ; Calendar
.text:00401A35 push    ebx      ; Locale
.text:00401A36 call    ds:SetCalendarInfoW
.text:00401A3C push    ebx      ; uMinFree
.text:00401A3D call    ds:LocalCompact
.text:00401A43 push    ebx      ; hHandle
.text:00401A44 push    ebx      ; nStdHandle
.text:00401A45 call    ds:SetStdHandle
.text:00401A4B push    ebx      ; dwReserved
.text:00401A4C push    ebx      ; nServerPort
.text:00401A4D push    ebx      ; pswzServerName
.text:00401A4E push    ebx      ; hSession
.text:00401A4F call    ds:WinHttpConnect
.text:00401A55 push    ebx      ; Block
.text:00401A56 call    _free
.text:00401A58 push    ebx      ; Stream
.text:00401A5C call    _fclose
.text:00401A61 push    ebx      ; Number
.text:00401A62 call    _abs
.text:00401A67 push    ebx      ; Buffer
.text:00401A68 call    _puts
.text:00401A6D push    ebx      ; Buffer
.text:00401A6E call    _putc
.text:00401A73 add     esp, 14h

.text:00401A76 loc_401A76:
.text:00401A76 mov     esi, ebx

.text:00401A78 loc_401A78:
.text:00401A78 call    ds:GetTickCount
.text:00401A7E push    ebx      ; dwErrCode
.text:00401A7F call    ds:SetLastError
.text:00401A85 call    ds:GetTickCount
.text:00401A8B cmp     esi, 207138h
.text:00401A91 jg     short loc_401A9C

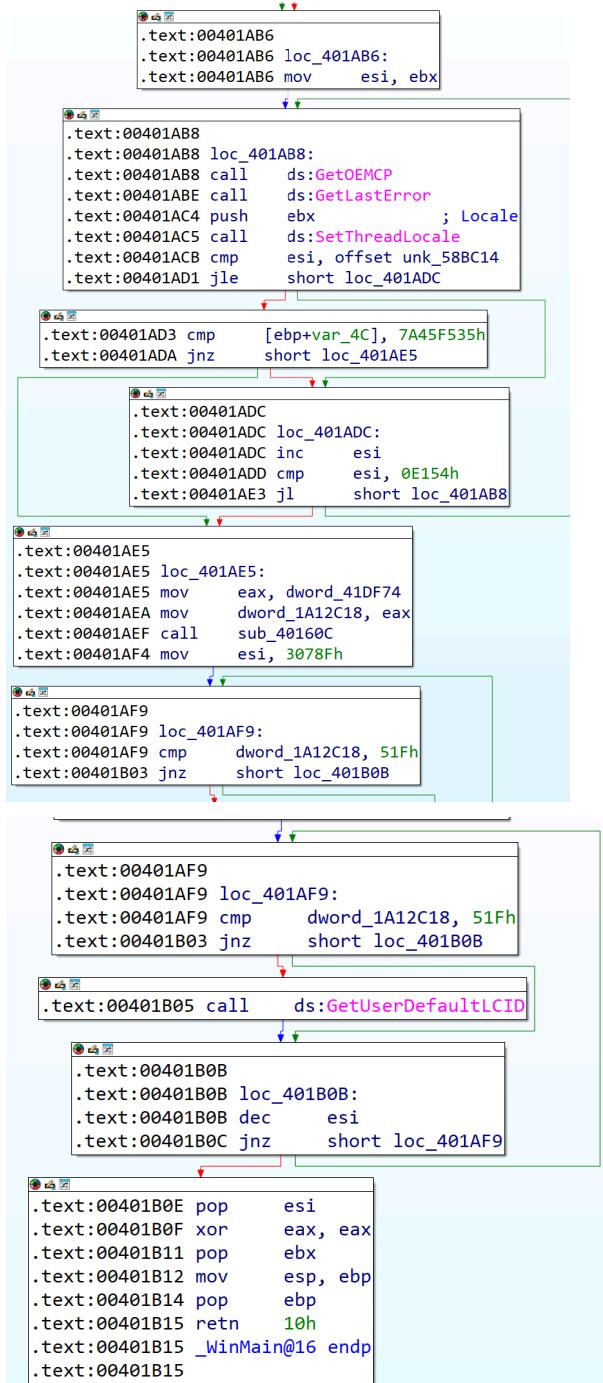
.text:00401A93 inc     esi
.text:00401A94 cmp     esi, 4F672h
.text:00401A9A jl     short loc_401A78

.text:00401A9C loc_401A9C:
.text:00401A9C mov     esi, ebx

.text:00401A9E loc_401A9E:          ; lpString
.text:00401A9E push    ebx
.text:00401A9F call    ds:GlobalFindAtomW
.text:00401AA5 cmp     esi, 2B157Bh
.text:00401AAB jg     short loc_401AB6

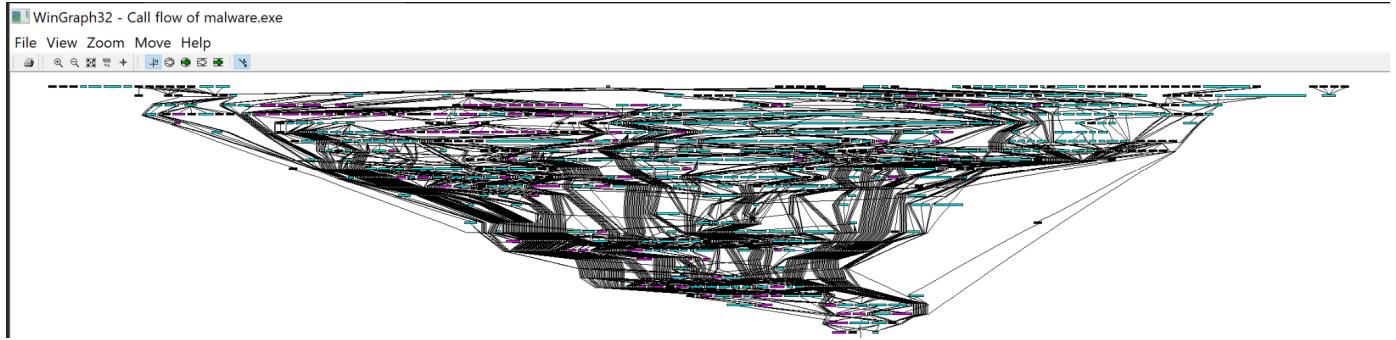
.text:00401AAD inc     esi
.text:00401AAE cmp     esi, 116BDh
.text:00401AB4 jl     short loc_401A9E

```



The four screenshots above are the functions listed under main. In these, there are calls to other functions that will execute the malware. When I click on these function calls, I am able to see the variables that are passed into each function as well as what that specific function returns.

Since I saw so many function calls, I decided to look at the graph of the call flowchart of this malware.



The above screenshot is a zoomed out picture of all of the function calls in this malware. As you can see, it is very detailed and in depth.

Since there were so many function calls, I decided to look into each function and the code that was in each one, as well as the strings used in this malware and the imports used as well.

Address	Length	Type	String
?? .rdata:00411...	00000010	C	no_buffer_space
?? .rdata:00411...	00000013	C	no_protocol_option
?? .rdata:00411...	000000E	C	not_connected
?? .rdata:00411...	000000D	C	not_a_socket
?? .rdata:00411...	00000018	C	operation_not_supported
?? .rdata:00411...	00000017	C	protocol_not_supported
?? .rdata:00411...	00000014	C	wrong_protocol_type
?? .rdata:00411...	0000000A	C	timed_out
?? .rdata:00411...	00000016	C	operation_would_block
?? .rdata:00411...	0000001D	C	address family not supported
?? .rdata:00411...	0000000F	C	address in use
?? .rdata:00411...	00000016	C	address not available
?? .rdata:00411...	00000012	C	already connected
?? .rdata:00411...	00000017	C	argument list too long
?? .rdata:00411...	00000017	C	argument out of domain
?? .rdata:00411...	0000000C	C	bad address
?? .rdata:00411...	00000014	C	bad file descriptor
?? .rdata:00411...	0000000C	C	bad message
?? .rdata:00411...	0000000C	C	broken pipe
?? .rdata:00411...	00000013	C	connection aborted
?? .rdata:00411...	0000001F	C	connection already in progress
?? .rdata:00411...	00000013	C	connection refused
?? .rdata:00411...	00000011	C	connection reset
?? .rdata:00411...	0000001D	C	destination address required
?? .rdata:00411...	00000018	C	executable format error
?? .rdata:00411...	0000000F	C	file too large
?? .rdata:00411...	00000011	C	host unreachable
?? .rdata:00411...	00000013	C	identifier removed
?? .rdata:00411...	00000016	C	illegal byte sequence
?? .rdata:00411...	00000023	C	inappropriate io control operation
?? .rdata:00411...	000000D	C	invalid seek
?? .rdata:00411...	0000000F	C	is a directory
?? .rdata:00411...	0000000D	C	message size
?? .rdata:00411...	0000000D	C	network down
?? .rdata:00411...	0000000E	C	network reset
?? .rdata:00411...	00000014	C	network unreachable
?? .rdata:00411...	00000010	C	no buffer space
?? .rdata:00411...	00000011	C	no child process
?? .rdata:00411...	00000008	C	no link
?? .rdata:00411...	00000015	C	no message available
?? .rdata:00411...	0000000B	C	no message
?? .rdata:00411...	00000013	C	no protocol option
?? .rdata:00411...	00000014	C	no stream resources
?? .rdata:00411...	0000001A	C	no such device or address

The above screenshot is just some of the strings used in this malware. Looking through these strings, I see some of the same strings from when I used the Strings tool as well as other tools. These strings along with their address helps me understand what each section of the malware is doing, and what certain functions are doing as well.

IDA View-A			Strings	Imports
Address	Ordinal	Name	Library	
> KERNEL32				
` GDI32				
0041...		GetCharacterPlacementW	GDI32	
` ADVAPI32				
0041...		DeregisterEventSource	ADVAPI32	
` WINHTTP				
0041...		WinHttpConnect	WINHTTP	

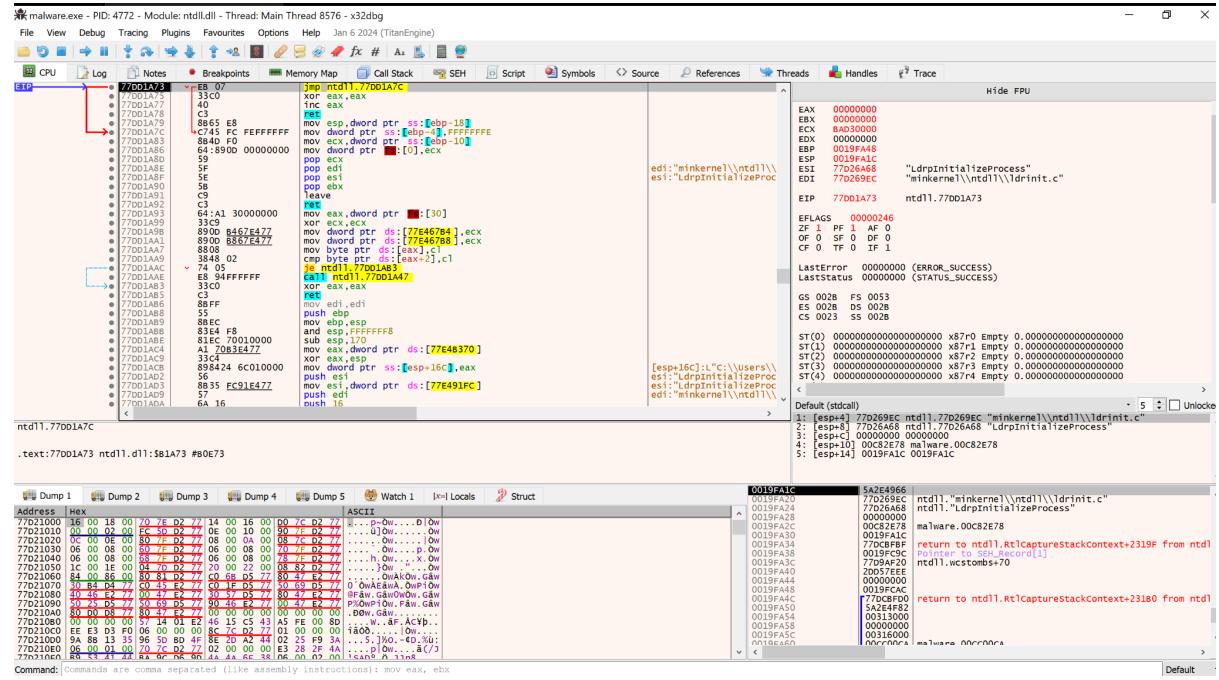
The above screenshot is a list of the imports used in Lumma Stealer. It shows me the DLL's that are imported as well as what functions are used from each DLL. Looking at the functions and their address gives me an understanding of certain roles of each section of this malware.

As I was going through IDA and getting deeper into functions, I realized that the functions listed by IDA for Lumma Stealer on the side panel are mainly for memory allocation and locking, exiting, or aborting. To me this means that this malware can detect when it is being reversed. Since it can detect when it is being reversed, the malware can possibly delete itself or hide information. The rest of the functions that are used in this malware are being called from the DLL imports.

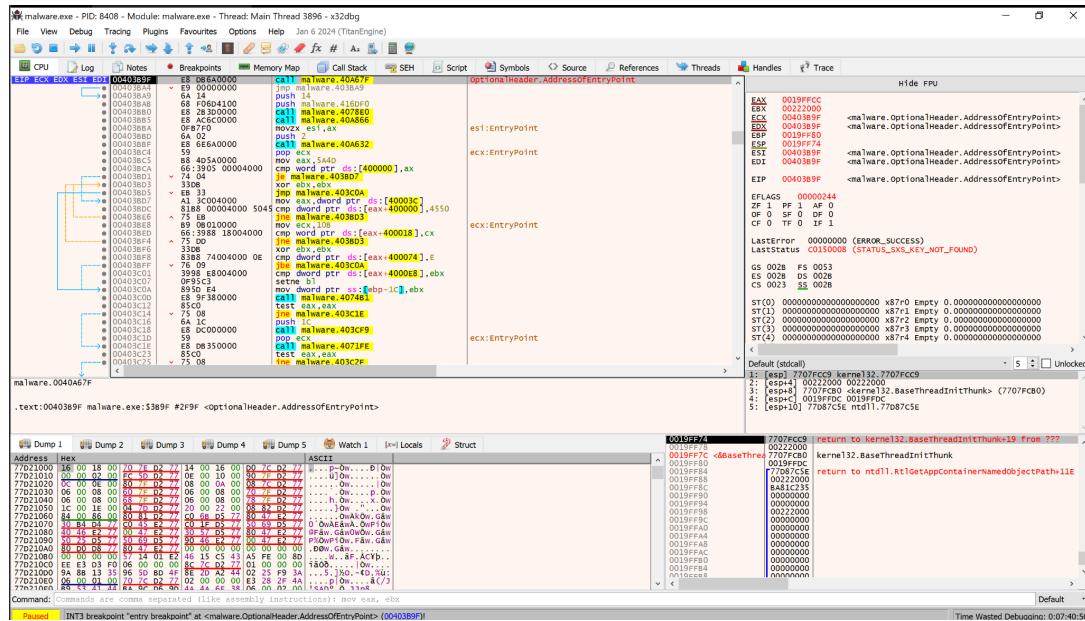
After doing a deep dive into IDA, I obtained a lot of new information about Lumma Stealer. I found out how this malware works, where specific functions are called, and what they do. This malware is able to change notifications and error messages. It also outputs errors and notifications depending on what is happening when it is run. Lumma Stealer can also detect the locale of where it is being run. Once it does this, the malware is able to customize locale specific settings. These include language, date, time, and number formats. This allows the malware to be run anywhere in the world since it can detect the locale of where it is and customize the settings depending on the locale. To go into even more depth, Lumma Stealer can obtain the OEM code page identifier which will allow it to determine the character encoding used by legacy applications. This malware can also do much more. This includes freeing space in memory, specifically the heap. This is done by removing unused blocks of memory. Finally, for the main role of this malware, it can connect to web browsers and obtain information regarding that browser as well as your personal information. Those points stated above are the main functions of Lumma Stealer and how it works. I obtained all of this information by going through IDA and looking into different blocks of code. For any function calls that I was unsure of, I did research on what those functions did. Going through IDA also allowed me to find specific locations for certain functions that I will use in x64dbg and OllyDbg.

Advanced Dynamic Analysis:

x64dbg:



The above screenshot shows the malware I am using opened up in x32dbg. I first tried to open this malware in x64dbg, however, nothing was shown. I tried again with x32dbg and it worked perfectly fine. At first glance, I am able to see the code being run, information regarding the registers, the stack, and the heap. I noticed that when I run the debugger a couple times that it freezes and I can't do anything with the code. I believe this is the code realizing that I am attempting to reverse it.



The above screenshot shows what the output of x32dbg is after I run it once. As you can see the comment next to the top line of code says “OptionalHeader.AddressOfEntryPoint”. This shows that the line of code next to the comment contains additional information about the executable, where it is loaded into memory, and where execution begins.

In order to get to specific areas in the code, I decided to use addresses I found from IDA, as well as lookup string references and use those references to put me in the correct place. Below is a screenshot of some string references found in the malware executable.

Address	Disassembly	String Address	string
004079E9	cmp dword ptr ds:[41208C],0	0041208C	"fc0"
004079F2	push malware.41208C	0041208C	"fc0"
004079F5	cmd_ecx malware.411890	00418890	"&{null}"
004079E5	movsx eax,byte ptr ds:[eax+412468]	00418890	"&{null}"
00408339	mov esi,dword ptr ds:[418894]	00418894	"&{null}"
00408426	mov esi,dword ptr ds:[418890]	00418890	"&{null}"
0040878C	mov esi,dword ptr ds:[418890]	00418890	"&{null}"
00408D40	lea ecx,dword ptr ds:[4188A0]	004188A0	"sqn"
00408D9D	lea ecx,dword ptr ds:[4188A0]	004188A0	"sqn"
0040A46C	mov eax,dword ptr ds:[eax+\$+412654]	00412654	"&L'R6002\r\n- floating point support not loaded\r\n"
0040A4D5	push malware.412FF0	00412FF0	"L"Runtime Error!\nProgram: "
0040A516	push malware.413024	00413024	"L"<program name unknown>"
0040A54E	push malware.413054	00413054	"L"\n"
0040A5AF	push malware.413068	00413068	L"Microsoft Visual C++ Runtime Library"
0040A8F9	push malware.413084	00413084	L"kernel32.dll"
0040A90C	push malware.4130D0	004130D0	"FlsAlloc"
0040A91A	push malware.4130DC	004130DC	"FlsFree"
0040A92D	push malware.4130E4	004130E4	"FlsGetValue"
0040A940	push malware.4130F0	004130F0	"FlsSetValue"
0040A953	push malware.4130FC	004130FC	"InitializeCriticalSectionEx"
0040A966	push malware.413118	00413118	"CreateEventExW"
0040A979	push malware.413120	00413120	"SetThreadStackGuarantee"
0040A98C	push malware.41313C	0041313C	"CreateThreadpoolTimer"
0040A99F	push malware.413154	00413154	"SetThreadpoolTimer"
0040A9B2	push malware.41316C	0041316C	"WaitForThreadpoolTimerCallbacks"
0040A9C5	push malware.413180	00413180	"CloseThreadpoolTimer"
0040A9D8	push malware.4131A0	004131A0	"CreateThreadpoolWait"
0040A9EB	push malware.4131B8	004131B8	"SetThreadpoolWait"
0040A9F6	push malware.4131D0	004131D0	"CloseThreadpoolWait"
0040AA13	push malware.4131E4	004131E4	"FileIOBaseBuffers"
0040AA29	push malware.413204	00413204	"FreeLibraryWhenCallReturns"
0040AA37	push malware.413214	00413214	"GetCurrentProcessorNumber"
0040AA44	push malware.413234	00413234	"GetLogicalProcessorInformation"
0040AA5D	push malware.413250	00413250	"CreatesymbolicLink"
0040AA70	push malware.413270	00413270	"SetDefaultDllDirectories"
0040AA83	push malware.413284	00413284	"EnumSystemLocalesEx"
0040AA96	push malware.4132A0	004132A0	"CompareStringEx"
0040AA99	push malware.4132B4	004132B4	"GetDateFormatter"
0040AABC	push malware.4132C4	004132C4	"GetLocalizerEx"
0040AB04	push malware.4132D4	004132D4	"GetTextFormatter"
0040AAE2	push malware.4132E4	004132E4	" GetUserDefaultLocaleName"
0040AAE5	push malware.4132F4	004132F4	"IsValidLocaleName"
0040AB08	push malware.413310	00413310	"LCMapStringEx"
0040AB18	push malware.413324	00413324	"GetCurrentPackageId"
0040AB2E	push malware.413334	00413334	"GetTickCount64"
0040AB41	push malware.413348	00413348	"GetFileInformationByHandleExW"
0040AB59	push malware.413358	00413358	"SetFileInformationByHandleW"
0040AB67	push malware.413378	00413378	"e-00"
0040AB7A	push malware.413390	00413390	"HLLA"
0040E521	push dword ptr [esi+\$+414330]	00414330	"USER32.DLL"
0040E50B	push malware.416454	00416454	"USER32.DLL"
0040E5FD	push malware.416454	00416454	"MessageBoxW"
0040E612	push malware.41646C	0041646C	"CreateEventW"

These strings above are only a few of the strings that are referenced in x32dbg.

Looking into these strings, and using the addresses found from IDA, I was able to view interesting areas of the code.

• 7113C1E2	68 24381271	<code>push winhttp.71123824 mov dword ptr ss:[ebp-10],0 call dword ptr ds:[<LoadLibraryExW>] mov esi,eax test esi,esi je winhttp.71184A89 push winhttp.71123808 push esi call dword ptr ds:[<GetProcAddress>] mov edi,eax test edi,edi je winhttp.71184AB3 push winhttp.711237EC push esi call dword ptr ds:[<GetProcAddress>] mov ebx,eax test ebx,ebx je winhttp.71184ADD push winhttp.711237D4 push esi call dword ptr ds:[<GetProcAddress>] mov dword ptr ss:[ebp-4],eax test eax,eax je winhttp.71184B07 push winhttp.711237B8 push esi call dword ptr ds:[<GetProcAddress>] mov dword ptr ss:[ebp-8],eax test eax,eax je winhttp.71184B31 push winhttp.711237A4 push esi call dword ptr ds:[<GetProcAddress>] mov dword ptr ss:[ebp-C],eax test eax,eax je winhttp.71184B58 push winhttp.71123798 push esi call dword ptr ds:[<GetProcAddress>] mov dword ptr ss:[ebp-10],eax +cc esp,cc</code>	71123824:L:"winhttp.dll" esi:"LdrpInitializeProcess" edi:"LdrpInitializeProcess" 71123808:"WinHttpCreateProxyResolver" esi:"LdrpInitializeProcess" edi:"minkernel\\ntdll\\ldrinit.c" edi:"minkernel\\ntdll\\ldrinit.c" 711237EC:"WinHttpGetProxyForUrlEx2" esi:"LdrpInitializeProcess" 711237D4:"WinHttpGetProxyResultEx" esi:"LdrpInitializeProcess" 711237B8:"WinHttpFreeProxyResultEx" esi:"LdrpInitializeProcess" 711237A4:"WinHttpCloseHandle" esi:"LdrpInitializeProcess" [ebp-0C]:wcstombs+70 71123798:"WinHttpOpen" esi:"LdrpInitializeProcess"
• 7113C280	68 7C371271	<code>je winhttp.71184B85 push winhttp.7112377C push esi call dword ptr ds:[<GetProcAddress>] mov dword ptr ss:[ebp-14],eax test eax,eax je winhttp.71184B8F push winhttp.71123764 push esi call dword ptr ds:[<GetProcAddress>] mov dword ptr ss:[ebp-18],eax test eax,eax je winhttp.71184B90 push winhttp.71123750 push esi call dword ptr ds:[<GetProcAddress>] mov dword ptr ss:[ebp-1C],eax test eax,eax je winhttp.71184C03 push winhttp.7112373C push esi call dword ptr ds:[<GetProcAddress>] mov ecx,ecx test ecx,ecx je winhttp.71184C2D mov eax,dword ptr ss:[ebp-4] mov dword ptr ds:[711D3C14],eax mov eax,dword ptr ss:[ebp-8] mov dword ptr ds:[711D3C10],eax mov eax,dword ptr ss:[ebp-C] mov dword ptr ds:[711D3C01],eax mov eax,dword ptr ss:[ebp-10] mov dword ptr ds:[711D3C08],eax mov eax,dword ptr ss:[ebp-14] mov dword ptr ds:[711D3C04],eax mov eax,dword ptr ss:[ebp-18] mov dword ptr ds:[711D3C1C],edi pop edi mov dword ptr ds:[711D3C00],eax mov eax,dword ptr ss:[ebp-1C] mov dword ptr ds:[711D3C20],esi esi:"LdrpInitializeProcess"</code>	7112377C:"WinHttpSetStatusCallback" esi:"LdrpInitializeProcess" [ebp-14]:RtlCaptureStackContext+2319F 71123764:"WinHttpResetAutoProxy" esi:"LdrpInitializeProcess" 71123750:"WinHttpSetOption" esi:"LdrpInitializeProcess" 7112373C:"WinHttpSetTimeouts" esi:"LdrpInitializeProcess" [ebp-0C]:wcstombs+70 [ebp-14]:RtlCaptureStackContext+2319F edi:"minkernel\\ntdll\\ldrinit.c" edi:"minkernel\\ntdll\\ldrinit.c" esi:"LdrpInitializeProcess"

I came across a section of the malware that caught my attention. It seems like this section of the malware is what Lumma Stealer is supposed to do. It seems to call the winhttp.dll file and make calls to other functions within this .DLL file. From what I am noticing, this section sets a proxy server, opens a browser, and gathers information from the stack. This is a majority of what Lumma Stealer is supposed to do.

71150A83	68 5C4B1271	push winhttp.71124B5C	71124B5C:"SavedLegacySettings"
71150A88	FFB424 D8000000	push dword ptr ss:[esp+D8]	
71150A8F	FF15 F4511D71	call dword ptr ds:[<RegSetValueExA>]	
71150A95	85C0	test eax,eax	
71150A97	75 30	jne winhttp.71150AC9	
71150A99	FFB424 D0000000	push dword ptr ss:[esp+D0]	
71150AA0	FFB424 DC000000	push dword ptr ss:[esp+DC]	
71150AA7	6A 03	push 3	
71150AA9	50	push eax	
71150AAA	FFB424 EC000000	push dword ptr ss:[esp+EC]	
71150ABA	FFB424 D8000000	push dword ptr ss:[esp+D8]	
71150ABE	FF15 F4511D71	call dword ptr ds:[<RegSetValueExA>]	
71150AC0	F7D8	neg eax	
71150AC2	1BC0	sbb eax,eax	
71150AC9	218424 CC000000	and dword ptr ss:[esp+CC],eax	
71150AD0	8B8424 C4000000	mov eax,dword ptr ss:[esp+C4]	
71150AD2	85C0	test eax,eax	
71150AD4	74 11	je winhttp.71150AES	
71150ADC	83BC24 E0000000 00	cmp dword ptr ss:[esp+E0],0	
71150ADE	74 07	je winhttp.71150AES	
71150ADF	50	push eax	
71150AE5	FF15 08521D71	call dword ptr ds:[<RegCloseKey>]	
71150AE7	888424 D8000000	mov eax,dword ptr ss:[esp+D8]	
71150AE9	878424 C4000000 00	mov dword ptr ss:[esp+C4],0	
71150AF7	85C0	test eax,eax	
71150AF9	74 0F	je winhttp.71150B0A	
71150AFB	50	push eax	
71150AFC	6A 00	push 0	
71150B04	FF35 DC3B1D71	push dword ptr ds:[711D3BDC]	
71150B0A	FF15 CC501D71	call dword ptr ds:[<HeapFree>]	
71150B11	8B8C24 0C050000	mov ecx,dword ptr ss:[esp+50C]	
71150B13	8BC7	edi:"minkernel\\ntdll\\ldrinit.c"	
71150B14	5F	pop edi	
71150B15	5E	pop esi	
71150B16	5B	pop ebx	
71150B18	33CC	xor ecx,esp	
71150B1B	E8 83E60100	call winhttp.7116F1AO	
71150B1D	B8E5	mov esp,ebp	
71150B1F	5D	pop ebp	
71150B20	C2 1C00	ret 1C	

The screenshot above shows the part of the code where the malware obtains the legacy settings of applications. This is used to determine the type of character encoding.

75089472	68 AC1DC975	push kernelbase.75C91DAC	75C91DAC:L"Locale"
75089477	8D4424 1C	lea eax,dword ptr ss:[esp+1C]	
7508947B	50	push eax	
7508947C	FF15 4400E075	call dword ptr ds:[<RtlInitUnicodeStr	
75089482	68 6CD3C575	push kernelbase.75C5D36C	/5C5D36C:L"Control Panel\\International"
75089487	8D4424 3C	lea eax,dword ptr ss:[esp+3C]	
75089488	50	push eax	
7508948C	FF15 4400E075	call dword ptr ds:[<RtlInitUnicodeStr	
75089492	8B4424 10	mov eax,dword ptr ss:[esp+10]	
75089496	344C24 38	lea eax,dword ptr ss:[esp+38]	
7508949A	C74424 20 18000000	mov dword ptr ss:[esp+18]	[esp+20]:wcstombs+70
750894A2	894424 24	mov dword ptr ss:[esp+24],eax	
750894A6	C74424 2C 40060000	mov dword ptr ss:[esp+2C],640	
750894AE	894C24 28	mov dword ptr ss:[esp+28],ecx	[esp+30]:RtlCaptureStackContext+231B0
750894B2	895C24 30	mov dword ptr ss:[esp+30],ebx	
750894B6	895C24 34	mov dword ptr ss:[esp+34],ebx	
750894BA	85FF	test edi,edi	
750894BC	0F84 8E000000	je kernelbase.75089550	edi:"minkernel\\ntdll\\ldrinit.c"
750894C2	8D4424 20	lea eax,dword ptr ss:[esp+20]	[esp+20]:wcstombs+70
750894C6	50	push eax	
750894C7	68 00000040	push 40000000	
750894CC	8D4424 1C	lea eax,dword ptr ss:[esp+1C]	
750894D0	50	push eax	
750894D1	FF15 EC00E075	call dword ptr ds:[<NtOpenKey>]	
750894D7	8BF0	mov esi,eax	esi:"LdrpInitializeProcess"
750894D9	85F6	test esi,esi	esi:"LdrpInitializeProcess"

75D88D9F	B9 14 21 C975	mov ecx, kernelbase.75C92114 jmp kernelbase.75D88EBF	75C92114:L"icountry"
75D88DA4	^ E9 16010000	cmp ebx, 3 je kernelbase.75D88ED7	
75D88DC4	83FB 04	push ebx	
75D88DC5	00 5F 25 F9FFFF	mov ecx, kernelbase.75C92100 jmp kernelbase.75D88EBF	75C92100:L"sLanguage"
75D88DB2	53	push esi	esi:"LdrpInitializeProcess"
75D88DB3	B9 0021C975	push eax	
75D88DB8	^ E9 02010000	call dword ptr ds:[_wtois] mov dword ptr ss:[ebp-808], 1009	
75D88DBE	FF15 F001E075	pop ecx	
75D88DC4	84 D9 F8F7FFF 0910	mov dword ptr ss:[ebp-808], 1009	
75D88DC5	59	push eax	
75D88DCF	8985 F4F7FFFF	mov dword ptr ss:[ebp-80C], eax	
75D88DD5	^ 75 2E	push 0	
75D88DD7	6A 00	lea eax, dword ptr ss:[ebp-814] mov dword ptr ss:[ebp-814], 400	
75D88DD9	8D85 ECF7FFFF	push eax	
75D88DDF	C785 ECFC7FFF 0004	call <kernelbase.NiValidateLocale>	
75D88E09	53	mov edx, dword ptr ss:[ebp-80C]	
75D88DEA	E8 61D7F8FF	mov ecx, eax	
75D88DEF	8895 F4F7FFFF	call kernelbase.75CDC44	
75D88DF5	88C8	test ax, ax	
75D88DF7	E8 484EF7FF	je kernelbase.75D88E05	
75D88DFC	66 85 C0	cmp eax, eax	
75D88E01	0F 84 D0F7FFF 7	mov ecx, dword ptr ss:[ebp-810]	
75D88E05	88B0 0F0F7FFF	call kernelbase.75D88E1AE	
75D88E08	E8 9EE3F7FF	mov cx, dword ptr ss:[ebp-80C]	
75D88E10	88B0 F4F7FFFF	call kernelbase.75D88722A	
75D88E16	E8 0F44FFF	mov ecx, eax	
75D88E17	88C8	mov eax, eax	
75D88E1D	889D F8F7FFFF	push eax	
75D88E23	88B0 ECFC7FFF	mov dword ptr ss:[ebp-808]	
75D88E29	380F	push edi	
75D88E2B	^ 75 0D	jmp kernelbase.75D88E3A	
75D88E2D	3D 09100000	cmp eax, 1009	
75D88E32	^ 74 06	je kernelbase.75D88E3A	
75D88E34	33C0	xor eax, eax	
75D88E36	33C0	xor eax, eax	
75D88E38	^ EB 11	jmp kernelbase.75D88E4B	
75D88E3A	85C9	test ecx, ecx	
75D88E3C	^ 75 2E	je kernelbase.75D88E6C	
75D88E3E	3D 09100000	cmp eax, 1009	
75D88E43	^ 0F85 8EF8FFFF	jne kernelbase.75D88E07	
75D88E49	33C0	xor eax, eax	
75D88E4B	53	push eax	
75D88E4C	51	push ecx	
75D88E4D	BA 0020C975	mov edx, kernelbase.75C920D0	75C920D0:L"Calendar"
75D88E52	33C9	xor ecx, ecx	
75D88E54	E8 94030000	call kernelbase.75D891ED	75C920E4:L"CalendarType"
75D88E59	80 E420C975	mov ecx, kernelbase.75C920E4	edi:"minkernel\\ntdll\\ldrinit.c"
75D88E61	88B0	mov edx, eax	esi:"LdrpInitializeProcess"
75D88E60	88D6	push edi	
75D88E62	53	push ebx	

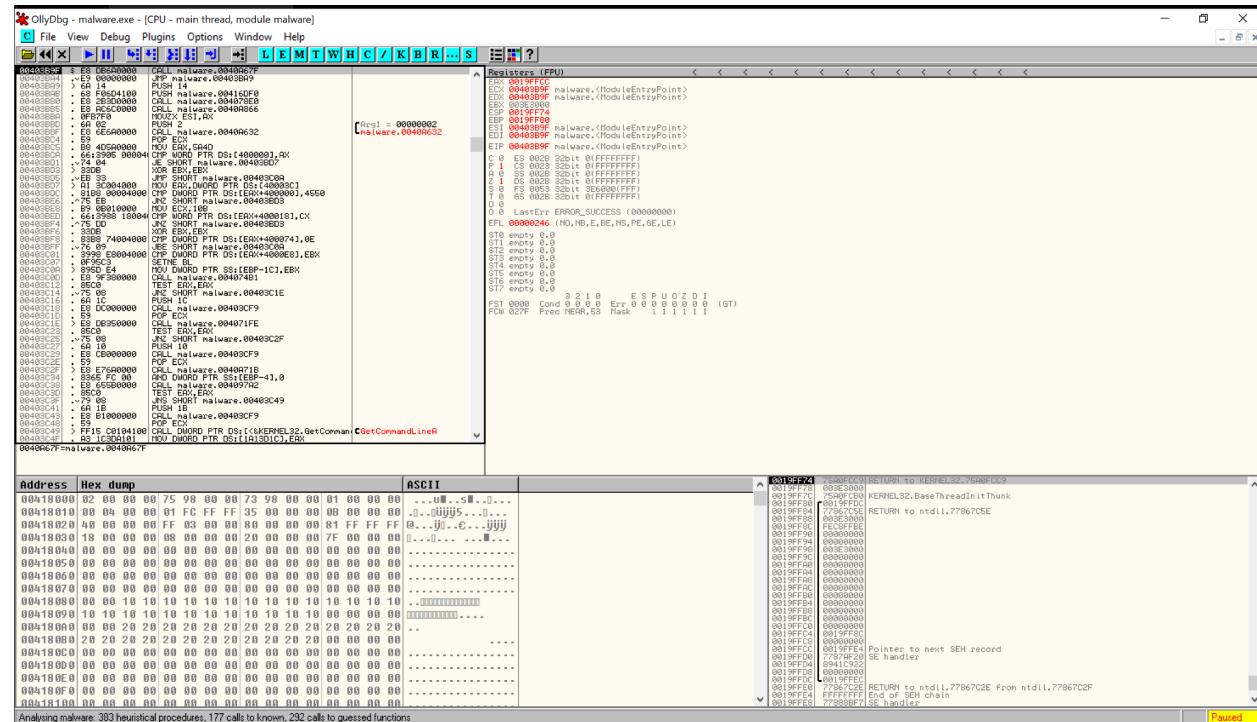
The two screenshots above show sections of code in the malware where the locale is defined based off of the country and language. This allows the malware to alter its settings such as date, time, language, and number format based on where in the world it is being run.

775F8459	68 98DC5C77	push advapi32.775CDC98	775CDC98:L"samlib.dll"
775F845E	FF15 24006377	call dword ptr ds:[_LoadLibraryExW]	
775F8464	A3 04F26277	mov dword ptr ds:[7762F204], eax	
775F8469	85C0	test eax, eax	
775F846B	^ OF84 D7000000	je advapi32.775F8548	
775F8471	68 B0DC5C77	push advapi32.775CDCB0	775CDCB0:"SamConnect"
775F8476	50	push eax	
775F8477	FF15 1C006377	call dword ptr ds:[_GetProcAddress]	
775F847D	A3 F4F16277	mov dword ptr ds:[7762F1F4], eax	
775F8482	85C0	test eax, eax	
775F8484	^ OF84 BE000000	je advapi32.775F8548	
775F8484	68 BCDC5C77	push advapi32.775CDCBC	775CDCBC:"SamOpenDomain"
775F848F	FF35 04F26277	push dword ptr ds:[7762F204]	
775F8495	FF15 1C006377	call dword ptr ds:[_GetProcAddress]	
775F8498	A3 E0F16277	mov dword ptr ds:[7762F1E0], eax	
775F84A0	85C0	test eax, eax	
775F84A2	^ OF84 A0000000	je advapi32.775F8548	
775F84A8	68 CCDC5C77	push advapi32.775CDC77	775CDC77:"SamLookupNamesInDomain"
775F84AD	FF35 04F26277	push dword ptr ds:[7762F204]	
775F84B3	FF15 1C006377	call dword ptr ds:[_GetProcAddress]	
775F84B9	A3 ECF16277	mov dword ptr ds:[7762F1FC], eax	
775F84BDE	85C0	test eax, eax	
775F84C0	^ OF84 82000000	je advapi32.775F8548	
775F84C6	68 E4DC5C77	push advapi32.775CDCE4	775CDCE4:"SamOpenUser"
775F84CB	FF35 04F26277	push dword ptr ds:[7762F204]	
775F84D1	FF15 1C006377	call dword ptr ds:[_GetProcAddress]	
775F84D7	A3 F0F16277	mov dword ptr ds:[7762F1F0], eax	
775F84DC	85C0	test eax, eax	
775F84DE	^ 74 68	je advapi32.775F8548	
775F84E0	68 F0DC5c77	push advapi32.775CDCF0	775CDCF0:"SamCloseHandle"
775F84E5	FF35 04F26277	push dword ptr ds:[7762F204]	
775F84EB	FF15 1C006377	call dword ptr ds:[_GetProcAddress]	
775F84F1	A3 E4F16277	mov dword ptr ds:[7762F1E4], eax	
775F84F6	85C0	test eax, eax	
775F84F8	^ 74 4E	je advapi32.775F8548	
775F84FA	68 00DD5C77	push advapi32.775CDD00	775CDD00:"SamFreeMemory"
775F84FF	FF35 04F26277	push dword ptr ds:[7762F204]	
775F8505	FF15 1C006377	call dword ptr ds:[_GetProcAddress]	
775F850B	A3 F8F16277	mov dword ptr ds:[7762F1F8], eax	
775F8510	85C0	test eax, eax	
775F8512	^ 74 34	je advapi32.775F8548	
775F8514	68 10DD5C77	push advapi32.775CDD10	775CDD10:"SamChangePasswordUser"
775F8519	FF35 04F26277	push dword ptr ds:[7762F204]	
775F851F	FF15 1C006377	call dword ptr ds:[_GetProcAddress]	
775F8525	A3 00F26277	mov dword ptr ds:[7762F200], eax	
775F852A	85C0	test eax, eax	
775F852C	^ 74 1A	je advapi32.775F8548	
775F852E	68 28DD5C77	push advapi32.775CDD28	775CDD28:"SamChangePasswordUser2"
775F8533	FF35 04F26277	push dword ptr ds:[7762F204]	
775F8539	FF15 1C006377	call dword ptr ds:[_GetProcAddress]	
775F853F	A3 E8F16277	mov dword ptr ds:[7762F1E8], eax	

The screenshot above shows samlib.dll being used in this malware to open a domain, free memory, and change passwords of users. This is very interesting and it is the first time I noticed that the malware is doing something like this.

Overall, when I took a deep dive into x32dbg, I noticed a lot of interesting things, some of which I hadn't noticed before. I was able to see what Lumma Stealer did when it found out that it was being reversed. I was also able to see major parts of Lumma Stealer such as code where it created a proxy and connected to a web browser, freeing memory, changing users passwords, identifying the locale and altering its settings based on that, and so much more.

OllyDbg:



The screenshot above shows the malware executable opened in OllyDbg. Once opened, I am able to see code, register information, information on the stack and heap. When I attempt to run the executable, the screen freezes and I am unable to do anything. This is the same thing that happened when I attempted to run it in x64dbg.

Since OllyDbg and x64dbg are both debuggers and ultimately do the same thing, in OllyDbg I decided to view alternative information. I first started by looking into the log, executable, and memory files.

The above screenshot shows the log file. This file is basically a log of everything that happened when the malware ran. It is very interesting to look into and see the files that were opened and .DLL files that were run. It also allows you to see the process and its ID that were created as well as the threads and their ID's.

The above screenshot shows all of the executable modules that were run. It gives information regarding where in the debugger they were run, and their size. It also gives information about the executable module itself such as the name, version of it, and its path.

The screenshot above shows the memory map of the executed malware. In the memory map, it includes the owners of the .DLL files along with the section and what it contains. It also provides the access, and initial access as well as what it is mapped to.

The three files listed above are all helpful in their own ways.

As I look through the CPU view of OllyDbg, I notice a lot of familiar function calls. I remember seeing a majority of these functions while looking through IDA and x64dbg.

Overall, since OllyDbg and x64dbg are used for the same purpose, I didn't find a lot of new information in OllyDbg. I still went through OllyDbg and used certain features that it has to help me find information. I also found some of the addresses that I was looking into from x64dbg. I think that x64dbg helped me gather more information, was more organized and had more features than OllyDbg. However, OllyDbg is still a good tool to use for reverse engineering if you need a debugger.

Summary:

By utilizing the tools provided in Basic Static, Basic Dynamic, Advanced Static, and Advanced Dynamic Analysis, I was able to learn a lot about Lumma Stealer. These tools allowed me to reverse this malware and figure out what exactly it does and how. After utilizing the reverse engineering tools I was also able to see the potential danger of Lumma Stealer. Lumma Stealer is a dangerous piece of malware that can be downloaded onto your device through phishing or social engineering tactics. Once downloaded, Lumma Stealer works in the background and is ultimately undetectable to a normal user. Lumma Stealer will aim to steal your sensitive information including usernames and passwords, and personal information. Lumma Stealer can also open a web browser and steal important information from the web browser. Once the information is gathered, the threat actors will use a proxy server to exfiltrate the data to. As you can tell this malware can be dangerous since the threat actors now have your personal and sensitive information. In order to protect against Lumma Stealer, you must have an antivirus running on your device. You should also avoid clicking on links or unknown files. You should only click on links or download files if they are from a trusted source. If Lumma Stealer is already downloaded onto your device, it can be tricky to remove. It will attempt to evade detection and removal. However you can attempt to remove it. You can do this by running an antivirus scan on your device to detect malicious files or viruses. You can then manually delete these files from your device. Once this is completed, restart your web browsers and make sure your operating system is updated. In order to make sure these steps worked, you can use trusted malware removal tools.