



CDAC-TVM  
DCSF

# "INVESTIGATING MALWARE IN THE CONTEXT OF NETWORK FORENSICS"

MINI PROJECT

SUBMITTED BY:

SURAJ TALKATKAR  
ATHARVA JOSHI  
VIRAJ PADEKAR  
PRISCILLA A.  
NIKITA THORAT

**MINI PROJECT REPORT**  
**On**  
**“Investigating Malware in the Context of Network  
Forensics”**

**Submitted to**



**For Partial Fulfilment  
Of  
Post Graduation Diploma In  
CYBER SECURITY AND FORENSICS  
(September 2023)**

**Submitted By:**

<b>NAMES</b>	<b>PRN</b>
<b>SURAJ TALKATKAR</b>	<b>230960940051</b>
<b>ATHARVA JOSHI</b>	<b>230960940008</b>
<b>VIRAJ PADEKAR</b>	<b>230960940058</b>
<b>PRISCILLA A</b>	<b>230960940041</b>
<b>NIKITA THORAT</b>	<b>230960940053</b>

**UNDER THE GUIDANCE OF  
MR JAYARAM P.  
(Project Guide & Centre Co-ordinator)**

## **ACKNOWLEDGEMENT**

We take the opportunity to express our gratitude to Prof. Jayaram P Who was the driving force during our project work. His guidance, encouragement and inspiration throughout the various phases of this project kept us on our toes; throughout the job without his advice, guidance and cooperation we would not have succeeded so far. He has helped us as a friend, philosopher and in all possible ways.

Suraj Talkatkar

Atharva Joshi

Priscilla A

Viraj Padekar

Nikita Thorat

# INDEX

<b>Chapter No.</b>	<b>Title</b>	<b>Page No.</b>
01.	Abstract	1
02.	Objectives	2
03.	Technical Requirements	3
04.	Working	4
05.	Finding Network Patterns	8
06.	Pylocky Ransomware Decryption Using Pcap Data	14
07.	Decrypting Hidden Tear Ransomware	15
08.	Behaviour Patterns And Analysis	18
09.	Conclusion	23
10.	References	24

## **CHAPTER 1 - ABSTRACT**

This is a small-scale project investigating malware in the context of Network Forensics. Most of the incidents requiring network forensics will be based on malware-oriented events, such as network breaches, financial crime, data theft, and command and control. Most attackers will deploy command and control malware to enslave the compromised machine and gain leverage over the internal network for lateral movement. Generally, network forensics and computer forensics go hand in hand in case of investigating malware. The computer forensics investigator will find all that has changed on the system and where the malware resides. Then, they will find the executable causing the issues and upload them to a site, as mentioned as follows <https://www.virustotal.com> or <http://www.hybrid-analysis.com> to find more about the malware and its behaviour on the system and the network.

## **CHAPTER 2 - OBJECTIVES**

- Dissecting malware on the network
- PyLocky ransomware decryption using PCAP data
- Decrypting hidden tear ransomware
- Behavior patterns and analysis

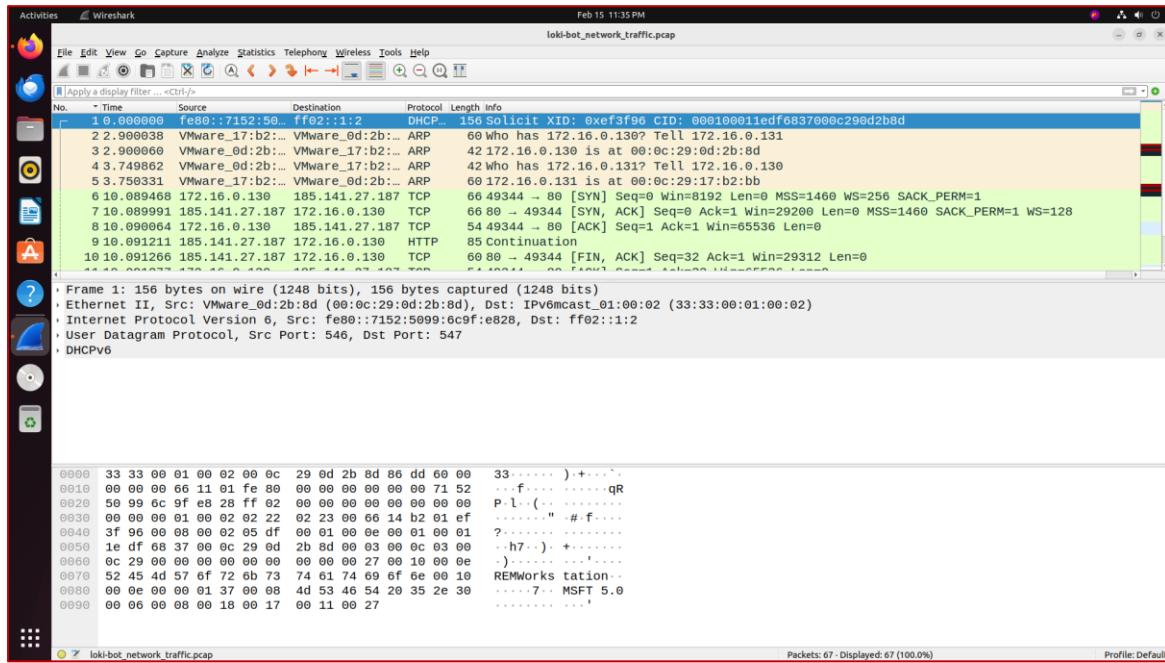
## **CHAPTER 3 - TECHNICAL REQUIREMENTS**

- Wireshark v3.0.0 installed on
- Windows 10 OS and Ubuntu 14.04
- PCAP Files for the exercises
- NetworkMiner installed on Windows 10

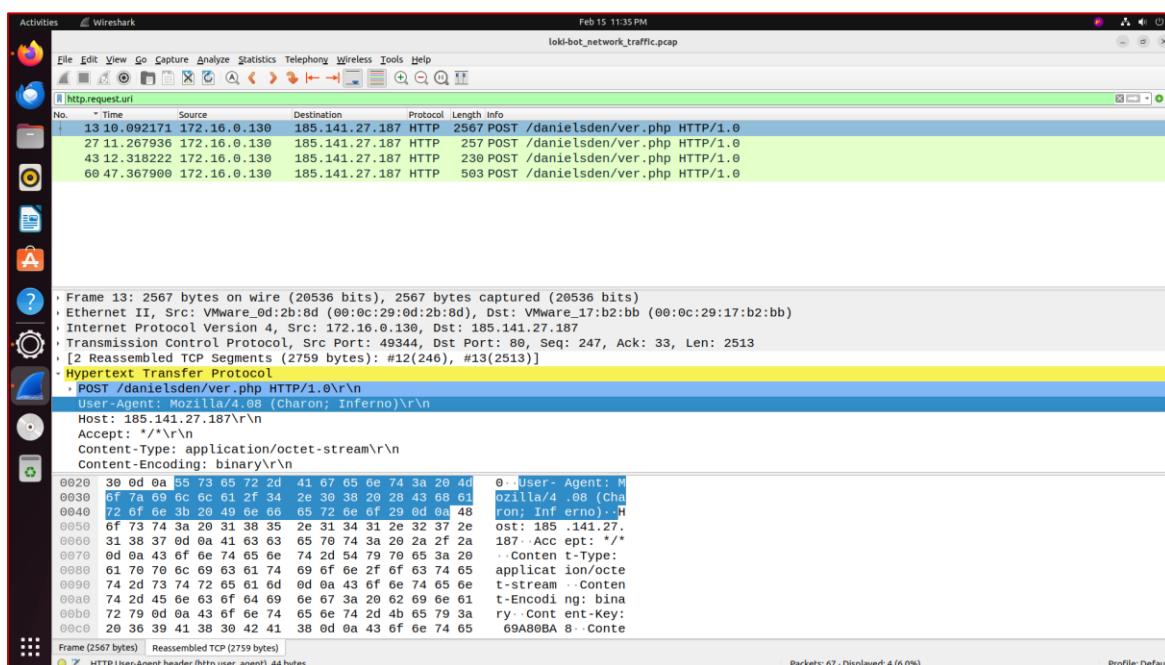
# CHAPTER 4 - WORKING

## Dissecting Malware on Network

Initially let's load the PCAP in Wireshark as follows:



We can see in the above screenshot that there is a lot of HTTP data present in the We can see there is a lot of HTTP data present in the PCAP file. Let's add columns to display the full URI and User-Agent entries, and also filter the requests using the http.request.uri filter as follows:



The user agent is quite important in malware communications since they might not be the standard user agents used by popular browsers. We can see we have Mozilla/4.08 (Charon; Inferno) as the user agent, and the URI contains a single user, as shown in the previous screenshot

It seems that the HTTP requests are generated by the nefarious LokiBot. LokiBot is a popular malware that infiltrates data on infected systems. So as we can see in the recent result open the third link from the preceding results which is from <https://lab.dynamite.ai/pcaps> and analyze similar samples:

This screenshot shows the DynamiteLab interface for analyzing the file `lokibot_network_traffic.pcap`. The main header displays "Suspicious Traffic" with 2 IP Addresses and 11 Alerts.

The timeline at the top shows activity from April 28, 2017, between 12:32:43.919 AM and 12:34:18.664 AM. Below the timeline are two tables: "Top Alert Categories by Alert Count" and "Top Rules by Alert Count".

**Top Alert Categories by Alert Count:**

Category	Count
A Network Trojan was detected	6
Malware Command and Control Activity Detected	4
Generic Protocol Command Decode	1

**Top Rules by Alert Count:**

Rule	Count
ET MALWARE LokiBot User-Agent (Charon/Inferno)	4
ET MALWARE LokiBot Checkin	4
SURICATA HTTP unable to match response to request	1
ET MALWARE LokiBot Keylogger Data Exfiltration Detected M1	1
ET MALWARE LokiBot Keylogger Data Exfiltration Detected M2	1

This screenshot continues the analysis of the `lokibot_network_traffic.pcap` file. It features two main sections: "Top Alert-Generating Hosts" and "Top Alert-Receiving Hosts".

**Top Alert-Generating Hosts:** A chart showing alert counts for hosts 172.16.0.130 and 185.141.27.187. Host 172.16.0.130 has an alert count of approximately 10, while host 185.141.27.187 has a count of approximately 1.

**Top Alert-Receiving Hosts:** A table showing hosts receiving alerts. The top host is 185.141.27.187 with a count of 10, and the second host is 172.16.0.130 with a count of 1.

**Top Alert-Generating Protocols:** A donut chart indicating that TCP accounts for 100.00% of the alerts.

**Top Rules by Unique Host Count:** A table listing rules categorized by unique host count. All rules listed have a count of 1.

Timestamp	Source IP	Destination IP	Source Port	Destination Port	Transport Protocol	Signature Name	Alert Description	Severity	Signature ID
4/28/2017 12:33:20 AM	185.141.27.187	172.16.0.130	80	49344	TCP	SURICATA HTTP unable to match response to request	Generic Protocol Command Decode	LOW	2221010
4/28/2017 12:33:22 AM	172.16.0.130	185.141.27.187	49345	80	TCP	ET MALWARE LokiBot User-Agent (Charon;Inferno)	A Network Trojan was detected	HIGH	2021641
4/28/2017 12:33:22 AM	172.16.0.130	185.141.27.187	49345	80	TCP	ET MALWARE LokiBot Checkin	Malware Command and Control Activity Detected	HIGH	2025381
4/28/2017 12:33:58 AM	172.16.0.130	185.141.27.187	49347	80	TCP	ET MALWARE LokiBot User-Agent (Charon;Inferno)	A Network Trojan was detected	HIGH	2021641
4/28/2017 12:33:58 AM	172.16.0.130	185.141.27.187	49347	80	TCP	ET MALWARE LokiBot Checkin	Malware Command and Control Activity Detected	HIGH	2025381
4/28/2017 12:33:58 AM	172.16.0.130	185.141.27.187	49347	80	TCP	ET MALWARE LokiBot Keylogger Data Exfiltration Detected M1	A Network Trojan was detected	HIGH	2024315
4/28/2017 12:33:58 AM	172.16.0.130	185.141.27.187	49347	80	TCP	ET MALWARE LokiBot Keylogger Data Exfiltration Detected M2	A Network Trojan was detected	HIGH	2024319
4/28/2017 12:33:23 AM	172.16.0.130	185.141.27.187	49346	80	TCP	ET MALWARE LokiBot User-Agent (Charon;Inferno)	A Network Trojan was detected	HIGH	2021641
4/28/2017 12:33:10 AM	172.16.0.130	185.141.27.187	49344	80	TCP	ET MALWARE LokiBot User-Agent (Charon;Inferno)	A Network Trojan was detected	HIGH	2021641
4/28/2017 12:33:10 AM	172.16.0.130	185.141.27.187	49344	80	TCP	ET MALWARE LokiBot Checkin	Malware Command and Control Activity Detected	HIGH	2025381

As we can see there have been numerous entries with similar behavior. The important items from the preceding list are the HTTP Method and the User-Agent columns. We can see that there is plenty of information about LokiBot analysis. The takeaway for us is the first-byte word of the HTTP payload is the LokiBot Version. Let's see what it is by making use of “tshark -r /home/deadlist/Desktop/loki\_bot\_network\_traffic.pcap -2 -R http.request.uri -Tfields -e ip.dst -e http.request.full\_uri -e http.user\_agent -e data -E separator=, | cut -c1-91” command. The command will read the PCAP file defined using the X switch and it will display all packets having the URI using http.request.uri filter. The command will print comma-separated values (-E separator=,) of fields like destination IP, full URI, User-Agent and Data (-Tfields). Since the last value is of the data field, the use of cut -c1-91 will print the first two bytes (Byte Word) of the data only as shown in the following screenshot:

```

Activities Terminal Feb 16 12:00 AM suraj@Suraj: ~/Desktop
suraj@Suraj:~/Desktop$ tshark -r loki-bot_network_traffic.pcap -2 -R http.request.uri
1 10.092171 172.16.0.130 --> 185.141.27.187 HTTP 2567 POST /danielsden/ver.php HTTP/1.0
2 11.267936 172.16.0.130 --> 185.141.27.187 HTTP 257 POST /danielsden/ver.php HTTP/1.0
3 12.318222 172.16.0.130 --> 185.141.27.187 HTTP 230 POST /danielsden/ver.php HTTP/1.0
4 47.367900 172.16.0.130 --> 185.141.27.187 HTTP 503 POST /danielsden/ver.php HTTP/1.0
suraj@Suraj:~/Desktop$ tshark -r loki-bot_network_traffic.pcap -2 -R http.request.uri -T fields -e ip.dst -e http.request.full_uri -e http.user_agent -e data | cut -c1-91
185.141.27.187 http://185.141.27.187/danielsden/ver.php Mozilla/4.08 (Charon; Inferno) 1200
suraj@Suraj:~/Desktop$ ^C
suraj@Suraj:~/Desktop$ 

```

We can see the first-byte word is 1200, which implies 00 12(18) being divided by 10, which means that we have the LokiBot version 1.8. We can see that, in the next word (the next two bytes), we have hexadecimal values of 27, 28, and 2b, and, according to the information that we have read, this value defines the functionality of the packet and a value 27 implies Exfiltrate Application/Credential Data, 28 implies Get C2 commands, and 2b implies Exfiltrate Keylogger Data.

This means that the LokiBot has done the following activities in order:

- Exfiltrated an application's credential data twice
- Made the new command, which was to exfiltrate keylogger data
- Sent keylogger data

Finally, let's have a look at the data we have got so far:

- **The infected system:** 172.16.0.130
- **The command-and-control server:** 185.141.27.187
- **Malware used:** LokiBot
- **Malware detection:** User-Agent, HTTP Method (POST)
- **Malware activities:** Application data exfiltration and keylogging

Having basic information about the malware, let's dive deep into finding more information about the exfiltrated data by understanding its patterns in the next section.

## CHAPTER 5 - FINDING NETWORK PATTERNS

We know that the malware is stealing some application data, but we don't know which application it is and what data was stolen. To find this out by viewing the HTTP payload in the packet bytes (lowest pane) pane of the standard Wireshark display as follows:

0000	00 0c 29 17 b2 bb 00 0c	29 0d 2b 8d 08 00 45 00	) ..... ) + .. E ..
0010	00 00 3c a7 40 00 80 06	00 00 ac 10 00 82 b9 8d	< @ .. P .. _P ..
0020	1b bb c0 c0 00 50 d2 7f	db 05 f3 5f 3a 50 18	..... ' ..
0030	01 00 81 e1 00 00 12 00	27 00 00 00 0a 00 00 00	XXXXX111 11 ..
0040	58 58 58 58 58 31 31 31	31 31 01 00 06 00 00 00	R-E-M... .... R-E..
0050	52 00 45 00 4d 00 01 00	1c 00 00 00 52 00 45 00	M-W-O-R.. K-S-T-A..
0060	4d 00 57 00 4f 00 52 00	4b 00 53 00 54 00 41 00	T-I-O-N.. .... R..
0070	54 00 49 00 4f 00 4e 00	01 00 1c 00 00 00 52 00	E-M-W-o.. r-k-s-t..
0080	45 00 4d 00 57 00 6f 00	72 00 6b 00 73 00 74 00	a-t-i-o.. n-p.....
0090	61 00 74 00 69 00 6f 00	6e 00 70 0d 00 00 a0 05	..... .... k..
00a0	00 00 01 00 01 00 00 00	06 00 03 00 01 00 6b 00	..... .... a!....
00b0	00 00 01 00 00 00 00 00	00 00 61 21 00 00 01 00	

1. LokiBot Version – 1.8
2. Exfiltrate Data: 27
3. Wide:0 Length:10
4. Binary ID: XXXXX11111
5. Wide:1 Length:6
6. Username: REM
7. Wide:1 Length: 1c = 28
8. Computer Name: REMWORKSTATION

We can see from the preceding screenshot that the payload started with LokiBot version 18 in Decimal (12 in Hexadecimal), and we need to divide that by 10 to get the exact version. Also, we had 27 as the identifier for data exfiltration on application credentials. Next, the first word denotes a zero width, denoting that the payload value will be unpacked as a normal string. Next, we have a word value that denotes a length of 0a, 10 in decimal. We can see that we have a length of 10 bytes denoting the binary ID, which is XXXXX11111. Again, we have the next width and length, which will denote the system username; we can see we have a width of one and a length of six. Since we have a width of one, we will unpack this data as hex. Therefore, at two bytes each, we have the REM username. Next, we have the system name, and again

width is 1 and the length is 1c, denoting 28. The next 28 bytes indicate that the infected system name is REM WORKSTATION. Following the same notation for the values, the next value shows the domain, which is, again REM WORKSTATION. Below we can look at the next hex section as follows:

0090	61 00 74 00	69 00 6f 00	6e 00	70 0d 00 00	a0 05	a.t.i.o. n.p.....
00a0	00 00 01 00	01 00 01 00	00 00	06 00 03 00	01 00 6b 00	..... .....k.
00b0	00 00 01 00	00 00 00 00	00 00	61 21 00 00	01 00	..... ..a!....
00c0	30 00 00 00	42 00 37 00	45 00	31 00 43 00	32 00	0...B.7. E.1.C.2.
00d0	43 00 43 00	39 00 38 00	30 00	36 00 36 00	42 00	C.C.9.8. 0.6.6.B.
00e0	32 00 35 00	30 00 44 00	44 00	42 00 32 00	31 00	2.5.0.D. D.B.2.1.
00f0	32 00 33 00	05 00 00 00	67 35	63 79 32 06	09 00	2.3..... g5cy2...
0100	00 01 e1 48	01 6c d9 09	18 36	13 68 83 74	38 05	...H.l.. .6.h.t8.
0110	70 38 73 38	3a 22 2f 70	61 44	63 6f e0 75	e0 6e	p8s8:"/p aDco.u.n
0120	d9 31 2c 2e	c1 67 b9 1d	6c 0d	68 65 d1 1c	1c 32	.1,..g.. 1.he...2
0130	6d 79 1c 26	6e 19 e9 99	2d 40	3d 6d 9a 75	69 48	my.&n... -@=m.uiH
0140	16 22 44 08	74 cc 3d 73	0d 26	a6 78 44 d0	3c 00	."D.t.=s .&.xD.<.
0150	3f 78 6d 6c	20 76 65 72	00 73	69 6f 6e 3d	22 31	?xml ver .sion="1
0160	2e 74 30 fd	f7 cb 63 e3	64 ff	e6 67 1e 06	55 54	.t0...c. d...g..UT

1. Screen Width: 07d0 (3440), Screen Height: 05a0 (1440)
2. IsLocalAdmin: 1(Yes), IsBuild In Admin (Yes)
3. is64Bit: No (0)
4. Major Version: 6, Minor Version: 3, Product Type: 1, OS\_Bug Patch: 6b (107)

We have the next four bytes as the **Screen Width** and the following four as the **Screen Height**. We have a check on local admin and built-in admin, and the preceding screenshot shows that, in the next four bytes, both are showing a one, indicating a yes. The next two bytes are set to one if the OS is 64-bit, which is not the case, so it's set to zero. The next eight bytes define the OS major and OS minor products and the os\_bug patch variables, which are 6,3,1,107 respectively. This means that we can denote the OS as 6.3.1.107, which is Windows 8. Additionally, the values stored here are in the little-endian format which means the last significant byte is the first. In the next section, we have the following:

00b0	00 00 01 00	00 00 00 00	00 00	61 21 00 00	01 00	..... ..a!....
00c0	30 00 00 42	00 37 00	45 00	31 00 43 00	32 00	0...B.7. E.1.C.2.
00d0	43 00 43 00	39 00 38 00	30 00	36 00 36 00	42 00	C.C.9.8. 0.6.6.B.
00e0	32 00 35 00	30 00 44 00	44 00	42 00 32 00	31 00	2.5.0.D. D.B.2.1.
00f0	32 00 33 00	05 00 00 00	67 35	63 79 32 06	09 00	2.3..... g5cy2...

1      2      3      4      5

1. Reported: 0
2. Compressed: 1
3. Encoded: 0
4. Encoding: 0
5. Original Stolen Data Length: 8545 (Hex:2161)

We can see the next two bytes as the value denoting the first-time connection as a zero. This means that the victim has connected for the first time. Next, two bytes denote that the data stolen is compressed, while the following two bytes define whether the stolen data is encoded or not, and following these two bytes are another two bytes defining the encoding type. The next four bytes denote the original stolen data's length, which is 8,545 bytes. A separator is in between, and we again have the width and length for the string:

00b0	00	00	01	00	00	00	00	00	00	00	61	21	00	00	01	00	.....	..a!....
00c0	30	00	00	00	42	00	37	00	45	00	31	00	43	00	32	00	0...B·7·	E·1·C·2·
00d0	43	00	43	00	39	00	38	00	30	00	36	00	36	00	42	00	C.C·9·8·	0·6·6·B·
00e0	32	00	35	00	30	00	44	00	44	00	42	00	32	00	31	00	2·5·0·D·	D·B·2·1·
00f0	32	00	33	00	05	00	00	00	67	35	63	79	32	06	09	00	2·3···	g5cy2···
0100	00	01	e1	48	01	6c	d9	09	18	36	13	68	83	74	38	05	...H·l··	.6·h·t8·
0110	70	38	73	38	3a	22	2f	70	61	44	63	6f	e0	75	e0	6e	p8s8:"/p	aDco·u·n

1. Width:1 Length: 30 (48)
2. MUTEX

As shown in the preceding screenshot, we have a 48-byte-long mutex value used by the LokiBot. Next, LokiBot uses this mutex as follows:

- Mutex: B7E1C2CC98066B250DDB2123

Based on this value, the LokiBot's files will be located in the following locations:

- Hash Database: "%APPDATA%\C98066\6B250D.hdb"
- Keylogger Database: "%APPDATA%\C98066\6B250D.kdb"
- Lock File: "%APPDATA%\C98066\6B250D.lck"
- Malware Exe: "%APPDATA%\C98066\6B250D.exe"

If we observe closely, we can see that the directory name starts from the 8th character to the 13th character of the Mutex while the file name starts from the 13th character to the 18th character.

Well! That was too much information traveling on the network. Let's see what's next:

1	00f0 32 00 33 00 05 00 00 00	2	67 35 63 79 32 06 09 00	3	2.3..... g5cy2... ...H.l... .6.h+t8. p8s8:"/p aDco-u.n .1,...g.. l.he...2 my.&n... -@=m.uIh ."D.t.=s .&.xD.<. ?xml ver .sion="1 .t0...c. d..g..UT F-8".?>.. <Np...P
0100 00 01 e1 48 01 6c d9 09	18 36 13 68 83 74 38 05				
0110 70 38 73 38 3a 22 2f 70	61 44 63 6f e0 75 e0 6e				
0120 d9 31 2c 2e c1 67 b9 1d	6c 0d 68 65 d1 1c 1c 32				
0130 6d 79 1c 26 6e 19 e9 99	2d 40 3d 6d 9a 75 69 48				
0140 16 22 44 08 74 cc 3d 73	0d 26 a6 78 44 d0 3c 00				
0150 3f 78 6d 6c 20 76 65 72	00 73 69 6f 6e 3d 22 31				
0160 2e 74 30 fd f7 cb 63 e3	64 ff e6 67 1e 06 55 54				
0170 46 2d 38 22 01 3f 3e 0d	0a 3c 4e 70 c7 ef f7 50				

1. Key Length: 5
2. Key
3. Compressed Data: 2310

We have the key length, the key itself, and the length of compressed data. We now know that the length of the compressed data is 2,310 bytes, which looks like this:

The screenshot shows a Wireshark session titled "Follow TCP Stream (tcp.stream eq 0) · loki-bot\_network\_traffic.pcap". The packet list pane shows several TCP segments, with the last one being reassembled. The details pane displays the XML and HTML content of the compressed data. The bytes pane shows the raw hex and ASCII data. A status bar at the bottom indicates the total length is 2,513 bytes.

We can see some of the values as XML and HTML. But, we still need to decompress this data. On researching the malware executable file (Run strings command on the executable), we will discover that one of the strings in the binary executable contains LZSS, which is a popular data-compression encoding scheme. Using the library, we can copy the bytes from Wireshark

capture and feed it as input to the decompress function defined in the library. Let's decompress the data as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<FileZilla3>
  <Settings>
    <Setting name="Use Pasv mode">1</Setting>
    <Setting name="Limit local ports">0</Setting>
    <Setting name="Limit ports low">6000</Setting>
    <Setting name="Limit ports high">7000</Setting>
    <Setting name="External IP mode">0</Setting>
    <Setting name="External IP"></Setting>
    <Setting name="External address resolver">http://ip.filezilla-project.org/ip.php</Setting>
    <Setting name="Last resolved IP"></Setting>
    <Setting name="No external ip on local conn">1</Setting>
    <Setting name="Pasv reply fallback mode">0</Setting>
    <Setting name="Timeout">20</Setting>
    <Setting name="Logging Debug Level">0</Setting>
    <Setting name="Logging Raw Listing">0</Setting>
    <Setting name="fzsftp executable"></Setting>
    <Setting name="Allow transfermode fallback">1</Setting>
```

The stolen data is from FileZilla, and it looks like a config file. On repeating the analysis for other packets, such as one with the value 2B (keylogger) type, we will have similar data, and on decompression, it will look similar to the following:

```
n
Window: Search Pane
otepad
Window: new 1 - Notepad++
i
Window: *new 1 - Notepad++
thdshfhasdlf jas jdflahsldfh ashflhsklf asjf lahshl ashflahsflhhfl ashasdl
fhlfshdf hasklfhls hfahflasf
s
fas fashfdl ahshglhas lkjaslkhf lahsghalsjlasdflhalshf hasglha sldfhlhaslhg as
```

We have successfully gathered the following **Indicators of Compromise (IOC)** details by working on the preceding sample:

- **The infected system:** 172.16.0.130
- **The infected user:** REM
- **The infected system hostname:** REM WORKSTATION
- **Domain infected:** REMWorkstation
- **OS architecture:** 32 Bit
- **Screen resolution:** 3440 x 1440
- **Windows OS NT version:** 6.3.1 (Windows 8)
- **The command and control server:** 185.141.27.187
- **Malware used:** LokiBot
- **Malware detection:** User-Agent, HTTP method (POST)
- **Malware activities:** Application Data Exfiltration on FileZilla, Keylogging
- **Malware version:** 1.8
- **Malware compression:** LZSS
- **Malware encoding:** None
- **Malware files names:** %APPDATA%\C98066\6B250D.\*

## CHAPTER 6 - PYLOCKY RANSOMWARE DECRYPTION USING PCAP DATA

Cisco has launched the PyLocky decryptor ([https://github.com/Cisco-Talos/pylocky\\_decryptor](https://github.com/Cisco-Talos/pylocky_decryptor) ), which searches through the PCAP to decrypt files on the system. PyLocky sends a single POST request to the control server containing the following parameters:

```
PCNAME=NAME&IV=KXyiJnifKQQ%3D%0A&GC=VGA+3D&PASSWORD=CVxAfel9ojCYJ9So&CPU=Intel%28R%29+Xeon%28R%29+CPU+E5-1660+v4+%40+3.20GHz&LANG=en_US&INSERT=1&UID=XXXXXXXXXXXXXX&RAM=4&OSV=10.0.16299+16299&MAC=00%3A00%3A00%3A00%3A45%3A6B& OS=Microsoft+Windows+10+Pro
```

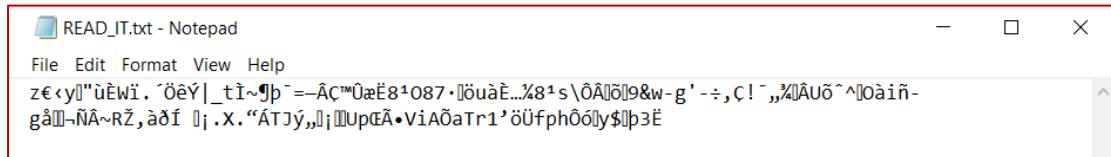
We can see that we have iv, the initialization vector, and the password as the parameters. In case the network was being logged at the time of the system infection, we could use this information to decrypt the files with ease. Let's look at PyLocky's code for decryption, as follows:

```
if "lockedfile" in fname:
    global counter
    fname_w_e = os.path.splitext(fname)[0]
    if debug:
        print("Opening fname: "+fname)
    fd = open(fname, "rb")
    data = fd.read()
    fd.close()
    if debug:
        print("Closed fname: "+fname)
    ddata = des3_decrypt(password, iv, data, debug)
    rdata = ddata.decode("base64")
    if debug:
        print("Opening fname_w_e: "+fname_w_e)
    fd = open(fname_w_e, "wb")
    fd.write(rdata)
    fd.close()
    if debug:
        print("Closed fname_w_e: "+fname_w_e)
    if debug:
        print("File processed correctly: "+fname)
    if remove:
        os.remove(fname)
        if debug:
            print("File removed correctly: "+fname)
    counter += 1
```

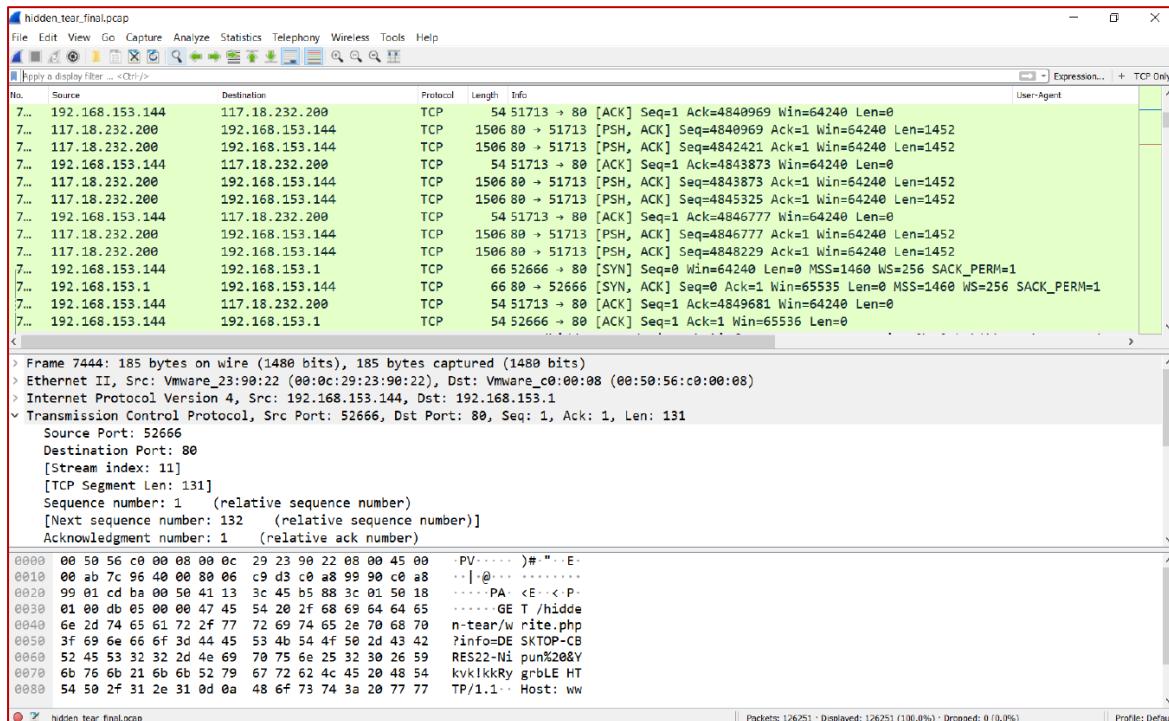
We can see that PyLocky decrypter makes use of IV and passwords to decrypt the files encrypted with the PyLocky ransomware, and generally, this way works for several ransomware types out there. PyLocky makes use of DES3 to encrypt the files that can be decrypted back.

CHAPTER 7 - DECRYPTING HIDDEN TEAR RANSOMWARE

Let's see another example with hidden tear ransomware. Consider a scenario where hidden tear ransomware has locked files on a Windows 10 system, and the situation is pretty bad. It looks like the files are encrypted.



Yes—the contents of the file are encrypted. Luckily for us, we have a PCAP of the fully captured data with us. Let's start our analysis:



We can see we have a fairly large PCAP file, containing a good amount of HTTP data. Since we know that malware have issues with user-agents, display the full user-agent and URI data in Wireshark as we did in the earlier examples:

No.	Source	Destination	Protocol	Length	Info	User-Agent	URL
7...	192.168.153.144	117.18.232.240	HTTP	539	GET /filestreamingservice/files/5cbc6... Microsoft-Delivery-Optimization/10.0		http://11.tlu.dl.delivery.mp.microsoft.com
7...	192.168.153.144	13.107.4.50	HTTP	542	GET /filestreamingservice/files/ffec... Microsoft-Delivery-Optimization/10.0		http://7.tlu.dl.delivery.mp.microsoft.com
7...	192.168.153.144	13.107.4.50	HTTP	542	GET /filestreamingservice/files/ffec... Microsoft-Delivery-Optimization/10.0		http://7.tlu.dl.delivery.mp.microsoft.com
7...	192.168.153.144	117.18.232.240	HTTP	539	GET /filestreamingservice/files/5cbc6... Microsoft-Delivery-Optimization/10.0		http://11.tlu.dl.delivery.mp.microsoft.com
7...	192.168.153.131	239.255.255.258	SSDP	175	M-SEARCH * HTTP/1.1		http://239.255.255.258:1900*
7...	192.168.153.144	117.18.232.240	HTTP	539	GET /filestreamingservice/files/5cbc6... Microsoft-Delivery-Optimization/10.0		http://11.tlu.dl.delivery.mp.microsoft.com
7...	192.168.153.144	13.107.4.50	HTTP	542	GET /filestreamingservice/files/ffec... Microsoft-Delivery-Optimization/10.0		http://7.tlu.dl.delivery.mp.microsoft.com
7...	192.168.153.144	13.107.4.50	HTTP	542	GET /filestreamingservice/files/ffec... Microsoft-Delivery-Optimization/10.0		http://7.tlu.dl.delivery.mp.microsoft.com
7...	192.168.153.144	117.18.232.240	HTTP	539	GET /filestreamingservice/files/5cbc6... Microsoft-Delivery-Optimization/10.0		http://11.tlu.dl.delivery.mp.microsoft.com
7...	192.168.153.144	117.18.232.240	HTTP	539	GET /filestreamingservice/files/5cbc6... Microsoft-Delivery-Optimization/10.0		http://11.tlu.dl.delivery.mp.microsoft.com
7...	192.168.153.144	13.107.4.50	HTTP	542	GET /filestreamingservice/files/ffec... Microsoft-Delivery-Optimization/10.0		http://7.tlu.dl.delivery.mp.microsoft.com
7...	192.168.153.144	13.107.4.50	HTTP	542	GET /filestreamingservice/files/ffec... Microsoft-Delivery-Optimization/10.0		http://7.tlu.dl.delivery.mp.microsoft.com
7...	192.168.153.144	117.18.232.240	HTTP	539	GET /filestreamingservice/files/5cbc6... Microsoft-Delivery-Optimization/10.0		http://11.tlu.dl.delivery.mp.microsoft.com

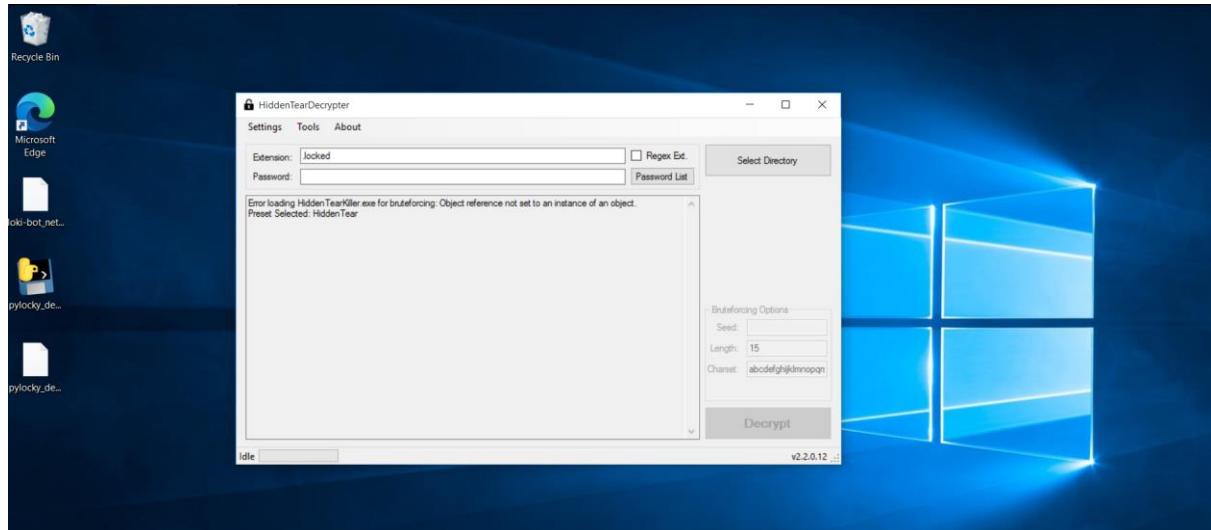
We can see that most of the data *is* being fetched from Microsoft domains, and probably looks like it is used by Windows update. Let's unselect this user-agent and see what we are left with:

No.	Source	Destination	Protocol	Length	Info	User-Agent
5...	192.168.153.144	184.85.125.248	HTTP	274	GET /static/mws-new/WeatherImages/210... Microsoft BITS/7.8	
7...	192.168.153.144	192.168.153.1	HTTP	185	GET /hidden-tear/write.php?info=DESKTOP-CBRES22-Nipun%20ajroR8/v0t/?/5& HTTP/1.1	
...	192.168.153.144	192.168.153.1	HTTP	185	GET /hidden-tear/write.php?info=DESKTOP-CBRES22-Nipun%20ajroR8/v0t/?/5& HTTP/1.1	

We can see that by using the `!(http.user_agent=="Microsoft-Delivery-Optimization/10.0")&&http.request.full_uri&&ssdp` filter, we are left with only a few packets. Let's investigate the packets as follows:

No.	Time	Source	Destination	Protocol	Length	Info
1042	154.50646	192.168.153.144	192.168.153.1	HTTP	185	GET /hidden-tear/write.php?info=DESKTOP-CBRES22-Nipun%20ajroR8/v0t/?/5& HTTP/1.1

We can see that a GET request containing our machine name and some string is sent to a domain. Could this be the password? We'll have to check. Let's download the decrypter from <https://github.com/goliate/hidden-tear>:

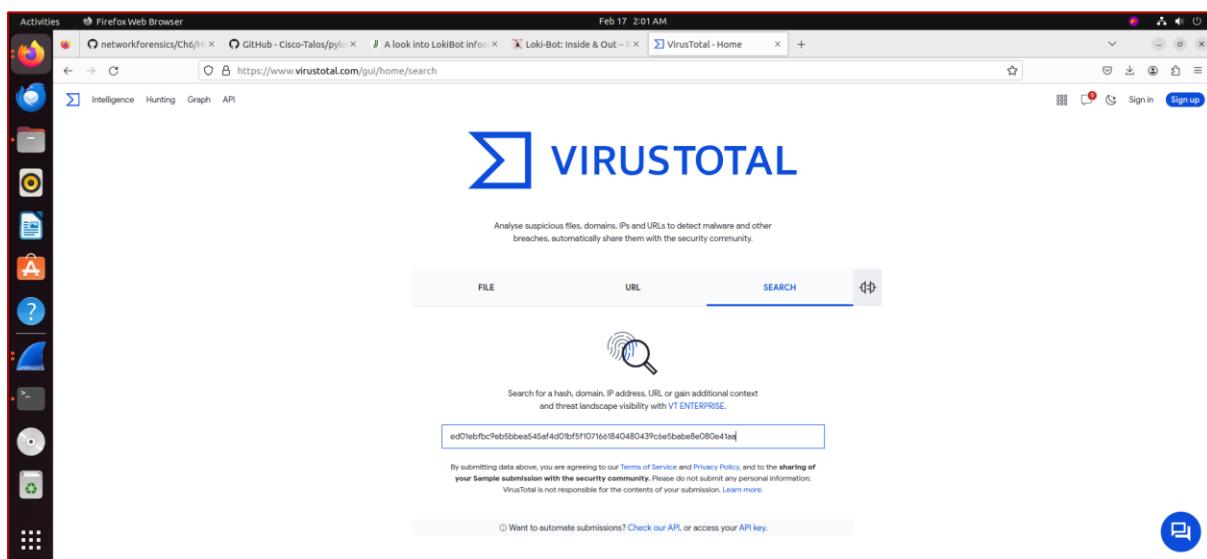


## CHAPTER 8 - BEHAVIOUR PATTERNS AND ANALYSIS

For a forensic network investigator, it is important to find the behaviour and network patterns of malware. Consider that you have received a few binaries (executable) and their hashes (signature) from the incident response team that are likely to be carrying malware. However, the analysis of PE/COFF executables is generally done by malware analysts and reverse engineers. What can you do with the PE executable? You don't have to study reverse engineering and malware analysis overnight to analyze the sample. Consider that you have received the file hash as

ed01ebfb9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.

You can use websites such as <https://www.virustotal.com/gui/home/upload> and <https://www.hybrid-analysis.com/> to analyze your sample without analyzing it on your system. The following screenshot shows the VirusTotal website:



Let's search the hash of the file at VirusTotal. The results should show up if the file has previously been analyzed:

Activities

Firefox Web Browser

Feb 17 2:02 AM

networkforensics/Ch6/ | GitHub - Cisco-Talos/pydrox | A look into LokiBot info... | Loki-Bot: Inside & Out - | VirusTotal - File - ed01eb...

https://www.virustotal.com/gui/file/ed01ebfbcb7eb5bbea545af4d0fbf5f1071661840480439c5e5babbe080e41aa/details

ed01ebfbcb7eb5bbea545af4d0fbf5f1071661840480439c5e5babbe080e41aa

64 security vendors and 6 sandboxes flagged this file as malicious

ed01ebfbcb7eb5bbea545af4d0fbf5f1071661840480439c5e5babbe080e41aa  
diskpart.exe

Community Score

Size 3.35 MB | Last Analysis Date 46 minutes ago | G EXE

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY 30+

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Basic properties

MDS 84c82836a5d7fbcc775e1706dbd5d49  
SHA-1 5f1445feafabcf010501a3ba2c2e413fa4424467  
SHA-256 ed01ebfbcb7eb5bbea545af4d0fbf5f1071661840480439c5e5babbe080e41aa  
Vhash 0360466568d70a823a3371ff  
Authentihash 4b2c4c70f0ff6feaddfecf33710ea6680a0c0c7ccdf7979c8dcf1f96002b99fd  
ImpHash 68f013d473743653ba98b058079febf1  
Rich PE header hash 417a0d0f1984fbc0cecd06546c882482  
SSEDEP 98304c90pb0hz1kxecf5UD36SAEdhwWm995938byNp3gXQqPeCxxk32AUeladzb9y4cgB  
TLSH T1P9D9G4Z42ZB1P2B1Z93M9JL92742BIEBFIE2D0A8001A0D044F7FB0C491  
File type Win32 Executable - windows - win32 - pe - peers  
Magic PE32 executable (GUI) Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz  
TxD Win32 Executable MS Visual C++ (generic) (37.8%) Microsoft Visual C++ compiled executable (generic) (20%) Win32 Executable (generic) (12.7%) Win32 Dynamic...  
DetectEasy PE32 Compiler: EP-Microsoft Visual C/C++ (6.0 (1720-7972)) [EXE32] Compiler: Microsoft Visual C/C++ (12.00.9752) [C++] Linker: Microsoft Linker (6.00.8047) T...  
File size 3.35 MB (3514368 bytes)  
PED packer Microsoft Visual C++

Here, 64/70 antivirus engines detect the file as malicious, and it may be a WannaCry ransomware sample.

Let's see the details from the Details tab as follows:

Basic properties ⓘ						
MD5	84c82835a5d21bbcf75a61706d8ab549					
SHA-1	5ff465afaabcbf0150d1a3ab2c2e74f3a4426467					
SHA-256	ed01ebfb9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa					
Vhash	036046656d1570a8z3631zfz					
Authentihash	4b2c4c7f06f5ffaaea6efc537f0aa66b0a30c7ccd7979c86c7f4f996002b99fd					
Imphash	68f013d7437aa653a898a05807afeb1					
Rich PE header hash	417a06d07984f3bce5cd6546c98842					
SSDEEP	98304:QqPoBhz1aRcxSUDk36SAEdhvxBaP593R8yAvp2g3x:QqPe1Cxcxk3ZAEUadzR8yc4gB					
TLSH	T173F533F4E221B7ACF2550EF64855C59B6A9724B2EBEF1E26DA8001A70D44F7F8FC0491					
File type	Win32 EXE executable windows win32 pe peexe					
Magic	PE32 executable (GUI) Intel 80386, for MS Windows					
TrID	Win32 Executable MS Visual C++ (generic) (37.8%)   Microsoft Visual C++ compiled executable (generic) (20%)   Win64 Executable (gen					
DetectItEasy	PE32   Compiler: EP:Microsoft Visual C/C++ (6.0 (1720-9782)) [EXE32]   Compiler: Microsoft Visual C/C++ (12.00.9782) [C++]   Linker: M					
File size	3.35 MB (3514368 bytes)					
PEiD packer	Microsoft Visual C++					
Names ⓘ						
SuperKeyPass.exe	Signature info ⓘ	Signature Verification				
diskpart.exe						
WannaCry.EXE						
wannacry.exe						
tmpDAC1.tmp.EXE						
Bewerbung10.exe.exe						
WannaCry.bin						
ed01ebfb9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.bin (1)						
ed01ebfb9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe						
evidencia01.exe						
Portable Executable Info ⓘ						
Compiler Products						
[IMP] Windows Server 2003 SP1 DDK build 4035 count=13	Creation Time	2010-11-20 09:05:05 UTC				
[---] Unmarked objects count=163	First Seen In The Wild	2016-05-16 15:27:03 UTC				
[C++] VS98 (6.0) SP6 build 8804 count=7	First Submission	2017-05-12 07:31:10 UTC				
[RES] VS98 (6.0) SP6 cvtres build 1736 count=1	Last Submission	2024-02-16 06:35:30 UTC				
id: 0xc, version: 7291 count=2	Last Analysis	2024-02-16 06:30:47 UTC				
id: 0xb, version: 8047 count=1						
id: 0xe, version: 7299 count=4						
id: 0xa, version: 8047 count=11						
id: 0x4, version: 8047 count=4						
Imports						
+ ADVAPI32.dll	Header					
+ KERNEL32.dll	Target Machine	Intel 386 or later processors and compatible processors				
+ MSVCRT.dll	Compilation Timestamp	2010-11-20 09:05:05 UTC				
+ USER32.dll	Entry Point	30650				
	Contained Sections	4				
Sections						
Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	27056	28672	6.4	920e964050a1a5dd60dd00083fd541a2	205225.92
.rdata	32768	24432	24576	6.66	2c42611802d585e6eed68595876d1a15	421271.69
.data	57344	6488	8192	4.46	83506e37bd8b50cacabd480f8eb3849b	502756.31
.rsrc	65536	3448736	3448832	8	f99ce7dc94308f0a149a19e022e4c316	647.82
Contained Resources						
SHA-256						
5873c1b5b246c80ab88172d3294140a83d711cd64520a0c7dd7837f028146b80	File Type	Type	Language	Entropy	Chi2	
f8cbc0ddb17a85f2ba099416961fefef915f8eba926681df7cd2c1fa69f3c2b6a	ZIP	XIA	ENGLISH US	8	450.37	
590b5bae6a9c329da6d5b83e3ec9baeb9607b8ea88e7015a01e021fc416707f	unknown	RT_VERSION	ENGLISH US	3.53	68837.26	
	unknown	RT_MANIFEST	ENGLISH US	5.04	17066.64	
Overlay						
size	1572864					
filetype	data					
offset	65536					
entropy	7.999859178965702					
md5	5822a94206522fe5382d2f00acc5cadf					

Plenty of detail can be seen on the **DETAILS** tab, especially the common names of the files causing this infection. We can also see that the file has been analyzed previously with a different name. Additionally, we have the following details:

The screenshot shows the VirusTotal analysis interface for a specific file hash. The top section displays the file's history across three scans: 2023-05-31 (5/62 detections), 2023-01-26 (0/60 detections), and 2023-05-08 (0/58 detections). Below this is a 'Graph Summary' section showing various relationships. A central node labeled 'PE RESOURCE STREAMS' has several outgoing connections: '10+ pe resource parents', '3 pe resource children', '10+ execution parents', '4 contacted urls', '10+ contacted ips', '10+ dropped files', and '10+ contacted domains'. The bottom right corner of the browser window contains a blue circular icon with a white arrow pointing up-right.

We can see that there are five IP addresses contacted by the WannaCry executable. We can filter the network based on these details to check infections in the network and pinpoint the infected source. Let's also upload/search the sample on the Hybrid-Analysis website (<https://www.hybrid-analysis.com/>) as well:

The screenshot shows the 'Network Analysis' section of the Hybrid Analysis website. It includes a 'DNS Requests' section stating 'No relevant DNS requests were made.' and a 'Contacted Hosts' table. The table lists seven IP addresses, their associated ports/protocols, processes, and locations:

IP Address	Port/Protocol	Associated Process	Details
185.118.0.67	9001 TCP	taskhsvc.exe PID: 4084	Norway
128.31.0.39	9101 TCP	taskhsvc.exe PID: 4084	United States
178.33.183.251	443 TCP	taskhsvc.exe PID: 4084	France
108.197.232.51	9001 TCP	taskhsvc.exe PID: 4084	United States
86.59.21.38	443 TCP	taskhsvc.exe PID: 4084	Austria
172.107.96.70	443 TCP	taskhsvc.exe PID: 4084	United States
78.46.217.214	443	taskhsvc.exe	Germany

Below the table is a 'Contacted Countries' section featuring a world map with colored arrows indicating network traffic flow between the United States and other countries like Norway, France, and Germany.

On searching the sample on Hybrid Analysis, we can see that we have the list of connected IP addresses, and a list of ports as well. This information will help us to narrow the outbound connections down from the infected system. We can see that Hybrid-Analysis has gone ahead and executed the associated sample file of the hash we provided for analysis in a secured environment:

The screenshot shows the Hybrid Analysis interface with a red border around the main content area. At the top, there are navigation tabs: Sandbox, Quick Scans, File Collections, Resources, Request Info, and a search bar. On the right, there's a sidebar with links like Incident Response, Related Sandbox Artifacts, Indicators, File Details (selected), File Metadata, File Sections, File Resources, File Imports, Screenshots (4), Hybrid Analysis (30), Network Analysis, Extracted Strings, Extracted Files (1905), Notifications, and Community (72). A 'Back to top' link is also present.

**File Resources**

Name	RVA	Size	Type	Language
XIA	0x100f0	0x349635	Zip archive data, at least v2.0 to extract.	English
RT_VERSION	0x359728	0x388	data	English
RT_MANIFEST	0x359ab0	0x4ef	exported SGML document, ASCII text, with CRLF line terminators	English

**File Imports**

ADVAPI32.dll KERNEL32.dll MSVCRT.dll USER32.dll

CloseServiceHandle  
CreateServiceA  
CryptReleaseContext  
OpenSCManagerA  
OpenServiceA  
RegCloseKey

**Screenshots**

The four screenshots show the following sequence:  
1. A clean Windows desktop with the Hybrid Analysis logo.  
2. A dark screen with a small window showing a command-line interface.  
3. A red-themed ransomware interface with a message about the system being encrypted.  
4. Another view of the ransomware interface with a message about the system being encrypted.

We can see the state of the system before and after the execution of the malware, where we can see that the system got infected with WannaCry ransomware.

## **CHAPTER 9 - CONCLUSION**

We have executed an in-depth exploration of malware, with a focus on dissecting the LokiBot on the packet level and decrypting ransomware variants such as PyLocky and Hidden Tear. Employed automated techniques utilizing platforms like VirusTotal, Hybrid-Analysis to enhance Investigative processes.

## CHAPTER 10 - REFERENCES

- <https://community.juniper.net/blogs/elevate-member/2020/12/22/a-look-into-lokibot-infostealer>
- <https://r3mrum.wordpress.com/2017/07/13/loki-bot-inside-out/>
- [https://r3mrum.files.wordpress.com/2017/07/loki\\_bot-grem\\_gold.pdf](https://r3mrum.files.wordpress.com/2017/07/loki_bot-grem_gold.pdf)
- <https://sensorstechforum.com/use-wireshark-decrypt-ransomware-files/>
- <https://paper.bobylive.com/Security/Hands-On-Network-Forensics.pdf>
- <https://lynseygrahamnovak.medium.com/network-forensic-investigation-identifying-malware-in-network-traffic-9a6bc32116dc>
- <https://www.dionach.com/behavioural-analysis-of-malware-via-network-forensics/>