CYBER CHALLENGE.IT

UNIVERSITÀ DI TRENTO

Forensics

Matteo Golinelli

Download Challenges.7z

File is on the Discord

Forensics Challenges

UNIVERSITÀ DI TRENTO

Base 64

- Encoding system that allows the translation of **binary data** into **ASCII text** strings
  - Data is represented using 64 different ASCII characters
  - Frequently end in some = characters
  - Might contain / characters
    - `aHR0cHM6Ly93d3cueW91dHViZS5jb20vd2F0Y2g/dj1kUXc0dzlXZ1hjUQ==`
- `echo “base64 encoded string” | base64 -d`

Useful Links

- https://gchq.github.io/CyberChef/
- https://trailofbits.github.io/ctf/forensics/

Suggestions

- Try **simple** stuff first
- Try everything
- Don’t be scared of **brute-forcing**
  - (when done on a local file)
- Think outside of the box
- Always try to know what **type** of files you are working with

⚠️

- Don’t cheat
- There is no right solution

First Steps

1. One flag for each file (recommended order: **1. use_strings, 2. find_key**)

First Steps: Hints

- Use file to **`find`** the type of the files

First Steps: Hints

- Use **`file`** to find the type of the files
- Make the files executable **`chmod +x <file>`**

First Steps: Solution

use_strings

```
~/Documents/cc2022/forensics/Challenges/firsts_steps
strings -n 45 use_strings/use_strings
Your flag is: CCITTN{Gotta_find_those_flags_GG_WP}
```

First Steps: Solution

find_key

```
~/Documents/cc2022/forensics/Challenges/firsts_steps
strings find_key/find_key
Hi, please insert the correct password:
%14s
solarwinds1234
Your flag is: %s
```

```
~/Documents/cc2022/forensics/Challenges/firsts_steps
./find_key/find_key
Hi, please insert the correct password: solarwinds1234
Your flag is: CCITTN{_no_more_free_flags:}
```

Cool Catto

Find an image on the cyberchallenge.disi.unitn.it platform

The image hides a flag

Cool Catto: Solution

https://gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true,false)Reverse('Character')From_Morse_Code('Space','Line%20feed')ROT13(true,true,false,13)&input=ZlNBdUxTNHVJQzRnTFMwdUxpNGdMUzR1SUM0dExpMGdMUzR1TGlBdUxTMHVJRjhnTFMwdUxpNGdMUzB1TFNBdUxTNHRJQzR1TFMwZ0xTMHRMUzRnTGkwdUxpQmZJQzB1TFMwZ0xTNGdMUzR1TGk0Z1h5QXVMU0F1TFMwZ0xpMGdMUzR0TFNBdExpNHVMaUF1TFMwZ0xTMHVMaTRnTGk0dExTQjdJQzB1SUM0dExTQXRMaTR1SUMwdUlDNHVMaTQ9

Sleepy Toni

Find an image on the cyberchallenge.disi.unitn.it platform and a Python script that was used to embed a flag into the image

Understand how that was done and find the flag

Sleepy Toni: Solution

From the code, we can see that the message in its binary form is embedded one bit at a time into the least significant bit of the RGB channels of each pixel

- Write a Python script that extracts the flag
- **zsteg catto.png**
- https://stylesuxx.github.io/steganography

Diskimage

Hints:

- Use **`zsteg -a`** to navigate the file
- Find an interesting file
- Once you get an image file: use stegoveritas (you might need to run stegoveritas_install_deps)

Diskimage: Hints

Hints:

- Use **`zsteg -a`** to navigate the file
- Find an interesting file (a DOS/MBR disk image)
- Once extracted the disk image, use **`testdisk`** to analyse it
- Once you get an image file (_FLAG.ICO): use stegoveritas (you might need to run stegoveritas_install_deps)

Diskimage: Solution 1

```
~/Documents/cc2022/forensics/Challenges/diskimage
zsteg -a diskimage.png
imagedata           .. text: ["w" repeated 10 times]
b1,b,msb,xy         .. file: MPEG-4 LOAS, 8 or more streams
b2,r,lsb,xy         .. text: "UUUUUUUUUUUUUUT"
b2,b,lsb,xy         .. text: "7wwwwwwwwwwwwwwwwwwwwwp"
b3,g,lsb,xy         .. file: OpenPGP Secret Key
b4,g,lsb,xy         .. text: "\"\"\"\"3333DDDDUUUUffffwwww"
b8,r,lsb,xy         .. text: "iin btldk eenraoaelpa\rrsnk  ygn.\r"
b8,g,lsb,xy         .. text: "/NNE A2 "
b8,g,msb,xy         .. text: ["@" repeated 8 times]
b8,b,lsb,xy         .. text: ")WOA  T "
b8,rgb,lsb,xy       .. file: DOS/MBR boot sector, code offset 0x3c+2, OEM-ID "~mitsumi",
7e572f0f, unlabeled, FAT (12 bit)
```

Diskimage: Solution 2

```
zsteg -a diskimage.png -e 'b8,rgb,lsb,xy' > outfile
```

```
~/Documents/cc2022/forensics/Challenges/diskimage
file outfile
outfile: DOS/MBR boot sector, code offset 0x3c+2, OEM-ID "~mitsumi
 FAT (12 bit), followed by FAT
```

Diskimage: Solution 3

- Use **`testdisk`** to navigate the disk image
  - “TestDisk is a free and open source data recovery software tool designed to recover lost partition and unerase deleted files” https://www.cgsecurity.org/wiki/TestDisk
- Find the file **_LAG.ico**
- Use **`stegoveritas`** to find the flag

Network Challenges

packet1.pcap

packet1.pcap: Solution

Analyse the pcap using **Wireshark** or **tshark**

- One packet is different than the others: it contains a base64-encoded string

```
❯ tshark -nr packet1.pcap
  1   0.000000 172.24.20.31 → 171.64.20.62 ICMP 98 Echo (ping) request  id=0x7a93, seq=1/256, ttl=63
  2   0.000020 171.64.20.62 → 172.24.20.31 ICMP 98 Echo (ping) reply    id=0x7a93, seq=1/256, ttl=64 (request in 1)
  3   1.000980 172.24.20.31 → 171.64.20.62 ICMP 98 Echo (ping) request  id=0x7a93, seq=2/512, ttl=63
```

```
 36  17.023696 171.64.20.62 → 172.24.20.31 ICMP 98 Echo (ping) reply    id=0x7a93, seq=18/4608, ttl=64 (request in 35)
 37  17.313236 172.24.20.31 → 171.64.20.62 ICMP 70 Echo (ping) reply    id=0x0001, seq=0/0, ttl=63
 38  18.025681 172.24.20.31 → 171.64.20.62 ICMP 98 Echo (ping) request  id=0x7a93, seq=19/4864, ttl=63
```

packet1.pcap: Solution

We can extract the data (in hexadecimal representation)

Only show the 37th frame

tshark -nr packet1.pcap -e data -Tjson frame.number==37

-n: Disable network object name resolution
-r: Read packet data from a file

Add a field to the list of fields to display

Set the format of the output when viewing decoded packet data to JSON


```
[
  {
    "_index": "packets-2023-08-27",
    "_type": "doc",
    "_score": null,
    "_source": {
      "layers": {
        "data": [
          "553156445645597 94d44497a6532467058326c7a58324e7662327839"
        ]
      }
    }
  }
]
```

packet1.pcap: Alternative Solution

- Extract the string using **strings** (*kind of pointless: we are learning Wireshark*)

- your solutions?

packet2.pcap

The goal is to find a tinyurl URL, we don’t care about the last step (but you can try to do it if you want)

packet2.pcap: Solution

1. We can open the pcap in Wireshark
2. Filter away ICMP messages: **!icmp**
3. We see TCP and FTP packets:
   a. Right click on an FTP packet and go **Follow > TCP Stream**
   b. We can see that FTP is used to transfer a file named **global_thermonuclear_war.gamerules.txt**
4. We know we are interested in a file transferred using FTP: we can extract it doing **File > Export Objects > FTP-DATA**
   a. We can save the file somewhere and read its content

packet3.pcap

The goal is to find a zip archive, we don’t care about cracking it (but you can try to do it if you want)

packet3.pcap: Solution

1. We can open the pcap in Wireshark
2. We can see some HTTP packets. The response includes some HTML
3. We can extract the HTTP files doing **File > Export Objects > HTTP**
4. There is a file named suctf2023, we can save it
5. To understand the type of file, we can use file
6. It's a zip archive, but it is password protected

7. We could crack it using **zip2john** and **john** (*left as a homework to the reader*)

packet4.pcap

The goal is to find some fake flags and a ciphered real flag

- We don’t care about deciphering it
  (but you can try to do it if you want)

packet4.pcap: Solution

1. We can analyse the packets using tshark or Wireshark
2. We again see ICMP packets, but some are different than the others
3. We can extract the interesting ones by excluding the others
   a. **tshark -nr packet4.pcap -Y '!icmp.ident eq 0xc9bc'**
4. We can show the data included in the interesting packets
   a. **tshark -nr packet4.pcap -Y '!icmp.ident eq 0xc9bc' -T fields -e data.text -o data.show_as_text:TRUE**

Add a field to the list of fields to display if -T ek|fields|json|pdml is selected

-o <preference>:<value>
Set a preference value, overriding the default value and any value read from a preference file

Show the data as text

Set the format of the output when viewing decoded packet data

packet4.pcap: Solution

You can read the next steps of the solution here:
https://andrewroderos.com/how-to-solve-my-pcap-ctf-challenges/

Even reading it I could not understand how to solve it

But the point was to extract the data with tshark or Wireshark, and we don't care about the final steps here

References

All the network challenges have been ~~stolen~~borrowed from

https://andrewroderos.com/how-to-solve-my-pcap-ctf-challenges/