

DISKO-1

By downloading we get a disko.dd.gz
by unzipping it using gunzip disko.dd.gz
we get disko.dd

A **.dd file** is basically a **raw disk image**.
It's created by the **dd** command in Linux/Unix.

dd copies data **block by block** (sector by sector) from one device/file to another.

A **.dd file** is an **exact bit-for-bit clone** of a disk, partition, or storage device.

by the hints i tried strings on the file disko.dd
we get the flag

```
teni@teni-IdeaPad-Slim-3-15IRH8:~/Downloads$ strings disko-1.dd | grep picoCTF
picoCTF{1t5_ju5t_4_5tr1n9_c63b02ef}
```

RED

by downloading the file we get a red.png
the challenge also included steganography

by search some commands to do some search in the image file i came across “zsteg”
zsteg is a **steganography analysis tool** used to detect and extract hidden data from PNG and BMP images.

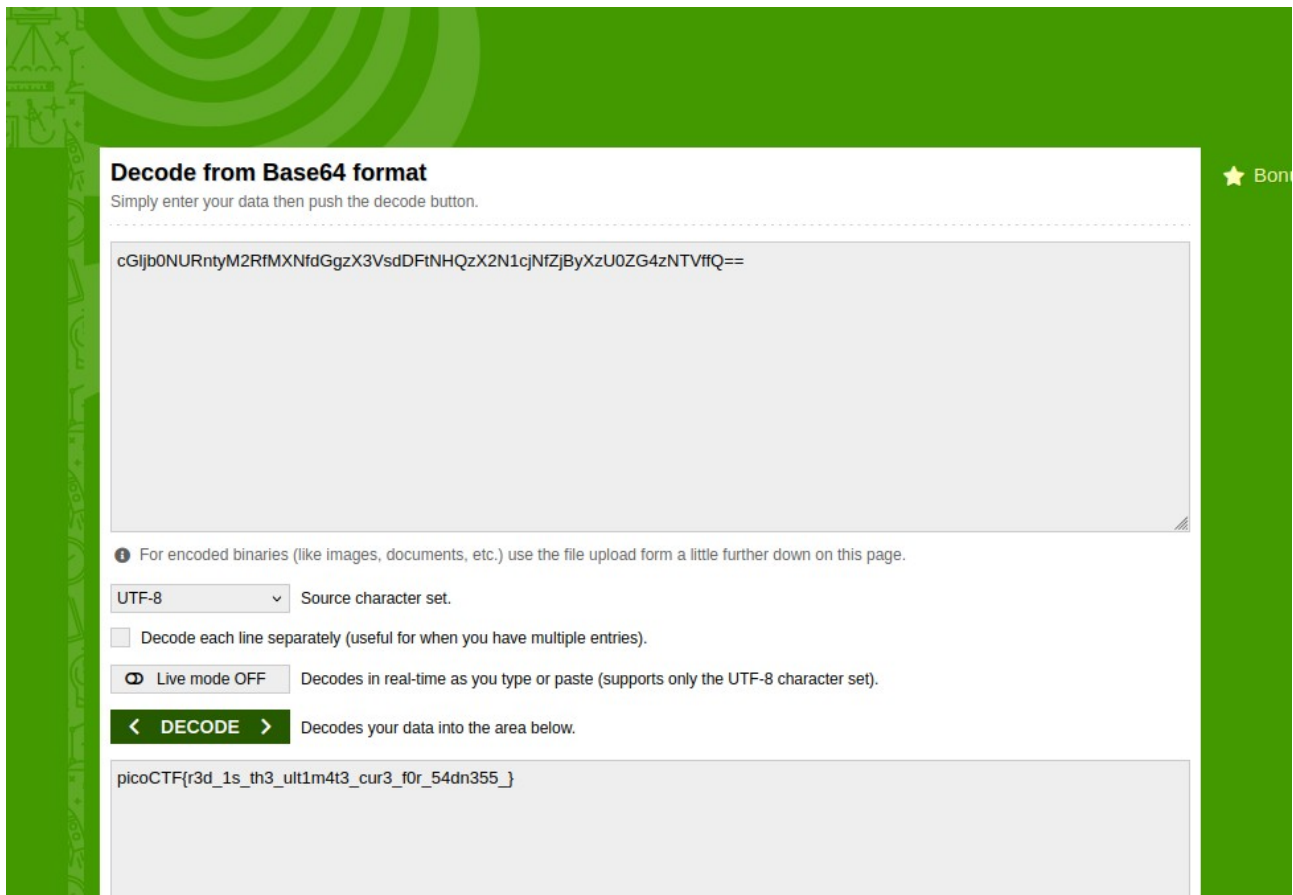
- It looks at **least significant bits (LSBs)** in image pixel data.
- Attackers or challenge authors often hide flags/messages in those bits.
- zsteg automates trying lots of encodings and outputs any hidden data it finds.

zsteg <filename>

by looking into it. We see a base64 code suspicious

```
teni@teni-IdeaPad-Slim-3-15IRH8:~/Downloads$ zsteg red.png
Crimson heart, vibrant and bold,\nHearts flutter a
t your sight.\nEvenings glow softly red,\nCherries burst with sweet life.\nKisse
s linger with your warmth.\nLove deep as merlot.\nScarlet leaves falling softly,
\nBold in every stroke."
"cGljb0NURntyM2RfMXNfdGgzX3VsdDFtNHQzX2N1cjNfZjByXz
U0ZG4zNTVffQ==cGljb0NURntyM2RfMXNfdGgzX3VsdDFtNHQzX2N1cjNfZjByXzU0ZG4zNTVffQ==cG
ljb0NURntyM2RfMXNfdGgzX3VsdDFtNHQzX2N1cjNfZjByXzU0ZG4zNTVffQ==cGljb0NURntyM2RfMX
NfdGgzX3VsdDFtNHQzX2N1cjNfZjByXzU0ZG4zNTVffQ=="
file: OpenPGP Public Key
      "ET@UETPETUUT@TUUTD@PDUDDPE"
file: OpenPGP Secret Key
file: OpenPGP Public Key
file: OpenPGP Secret Key
      "CIkiiiiI"
file: OpenPGP Secret Key
      "iiaakikk"
      "#wb#wp#7p"
      "7r'wb#7p"
file: 0421 Alliant compact executable not stripped
teni@teni-IdeaPad-Slim-3-15IRH8:~/Downloads$
```

by decoding it we get the flag



The screenshot shows a web-based Base64 decoder interface. At the top, it says "Decode from Base64 format" and "Simply enter your data then push the decode button." Below this is a large text input area containing the Base64 string: `cGlib0NURntyM2RfMXNfdGgzX3VsdDFtNHQzX2N1cjNfZjByXzU0ZG4zNTVffQ==`. Under the input area, there is a note: "For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page." Below the note are several options: a dropdown menu set to "UTF-8" with the label "Source character set.", a checkbox for "Decode each line separately (useful for when you have multiple entries).", a toggle switch for "Live mode OFF" with the description "Decodes in real-time as you type or paste (supports only the UTF-8 character set).", and a green button labeled "DECODE" with the description "Decodes your data into the area below." Below the button is a text output area displaying the decoded result: `picoCTF{r3d_1s_th3_ult1m4t3_cur3_f0r_54dn355_}`.

VERIFY

first we have to launch the instance once it's start running

we see a command

`ssh -p 59323 ctf-player@rhea.picoctf.net` Using the password `f3b61b38`.

Accept the fingerprint with `yes`, and `ls` once connected to begin. Remember, in a shell, passwords are hidden!

Then it is given that a SHA-256 hash is given and we have use decrypt script to get the password

SHA-256 stands for **Secure Hash Algorithm – 256 bit**.

- It's a **cryptographic hash function**, part of the SHA-2 family.
- It takes **any input** (text, file, password, etc.) and produces a **fixed 256-bit (32-byte) output**, usually written as a 64-character hexadecimal string.

we have three files `checksum.txt`, `decrypt.sh`, files

```

ctf-player@pico-chall$ ls
checksum.txt  decrypt.sh  files
ctf-player@pico-chall$ cat files
cat: files: Is a directory
ctf-player@pico-chall$ cat decrypt.sh

#!/bin/bash

# Check if the user provided a file name as an argument
if [ $# -eq 0 ]; then
    echo "Expected usage: decrypt.sh <filename>"
    exit 1
fi

# Store the provided filename in a variable
file_name="$1"

# Check if the provided argument is a file and not a folder
if [ ! -f "/home/ctf-player/drop-in/$file_name" ]; then
    echo "Error: '$file_name' is not a valid file. Look inside the 'files' folder with 'ls -R!'"
    exit 1
fi

# If there's an error reading the file, print an error message
if ! openssl enc -d -aes-256-cbc -pbkdf2 -iter 100000 -salt -in "/home/ctf-player/drop-in/$file_name" -k picoCTF; then
    echo "Error: Failed to decrypt '$file_name'. This flag is fake! Keep looking!"
fi
ctf-player@pico-chall$ ^C
ctf-player@pico-chall$ ls
checksum.txt  decrypt.sh  files
ctf-player@pico-chall$ cat checksum.txt
fba9f49bf22aa7188a155768ab0dfdc1f9b86c47976cd0f7c9003af2e20598f7

```

in checksum.txt has reference SHA-256 hash, decrypt.sh file to decrypt, files(has some possible flag files)

```

ctf-player@pico-chall$ cd files/
ctf-player@pico-chall$ ls -la
total 1216
drwxr-xr-x 2 ctf-player ctf-player 8192 Mar 12 2024 .
drwxr-xr-x 3 ctf-player ctf-player 57 Mar 12 2024 ..
-rw-r--r-- 1 root      root      64 Mar 12 2024 02kLdPvr
-rw-r--r-- 1 root      root      64 Mar 12 2024 0AEMwSHP
-rw-r--r-- 1 root      root      64 Mar 12 2024 0QBK0M73
-rw-r--r-- 1 root      root      64 Mar 12 2024 0ReLqo7l
-rw-r--r-- 1 root      root      64 Mar 12 2024 0pYtGDyN
-rw-r--r-- 1 root      root      64 Mar 12 2024 13eD2Klj
-rw-r--r-- 1 root      root      64 Mar 12 2024 15s8svF3
-rw-r--r-- 1 root      root      64 Mar 12 2024 1BX0cysv
-rw-r--r-- 1 root      root      64 Mar 12 2024 1QyIAyVK
-rw-r--r-- 1 root      root      64 Mar 12 2024 1WzFkICS
-rw-r--r-- 1 root      root      64 Mar 12 2024 1fP07Wz1
-rw-r--r-- 1 root      root      64 Mar 12 2024 1lUcIRwR
-rw-r--r-- 1 root      root      64 Mar 12 2024 1kzeT7Bq
-rw-r--r-- 1 root      root      64 Mar 12 2024 1s9fb01L
-rw-r--r-- 1 root      root      64 Mar 12 2024 2EHY0UNu
-rw-r--r-- 1 root      root      64 Mar 12 2024 2HLM0yNv
-rw-r--r-- 1 root      root      64 Mar 12 2024 2SLddvKn
-rw-r--r-- 1 root      root      64 Mar 12 2024 2y6e1PnY
-rw-r--r-- 1 root      root      64 Mar 12 2024 2ypuxhr6
-rw-r--r-- 1 root      root      64 Mar 12 2024 34asQJAj
-rw-r--r-- 1 root      root      64 Mar 12 2024 3FS3M7tr
-rw-r--r-- 1 root      root      64 Mar 12 2024 3HWGffFE
-rw-r--r-- 1 root      root      64 Mar 12 2024 3KQWpDpt
-rw-r--r-- 1 root      root      64 Mar 12 2024 3RwrWxAh
-rw-r--r-- 1 root      root      64 Mar 12 2024 3UI1Q3Im
-rw-r--r-- 1 root      root      64 Mar 12 2024 3XFTF7vI
-rw-r--r-- 1 root      root      64 Mar 12 2024 3azP1Vr5
-rw-r--r-- 1 root      root      64 Mar 12 2024 3bw1ssRA
-rw-r--r-- 1 root      root      64 Mar 12 2024 3llQZxg9
-rw-r--r-- 1 root      root      64 Mar 12 2024 3niXrkT4
-rw-r--r-- 1 root      root      64 Mar 12 2024 4E42FQrx
-rw-r--r-- 1 root      root      64 Mar 12 2024 4RWg3C81
-rw-r--r-- 1 root      root      64 Mar 12 2024 4WsDciyV
-rw-r--r-- 1 root      root      64 Mar 12 2024 4YmX8P1z
-rw-r--r-- 1 root      root      64 Mar 12 2024 4lpXJyrZ
-rw-r--r-- 1 root      root      64 Mar 12 2024 5L6bCE0e
-rw-r--r-- 1 root      root      64 Mar 12 2024 5VDBcjMu
-rw-r--r-- 1 root      root      64 Mar 12 2024 5aE4Zjfg
-rw-r--r-- 1 root      root      64 Mar 12 2024 5guD01cS
-rw-r--r-- 1 root      root      64 Mar 12 2024 5oxXh5Er
-rw-r--r-- 1 root      root      64 Mar 12 2024 5uH65mOX

```

by checking the reference sha-256 with the codes given in the /files by converting the codes into sha-256

we get a file files/87590c24

by decrypting the file using the .sh files given we get the flag

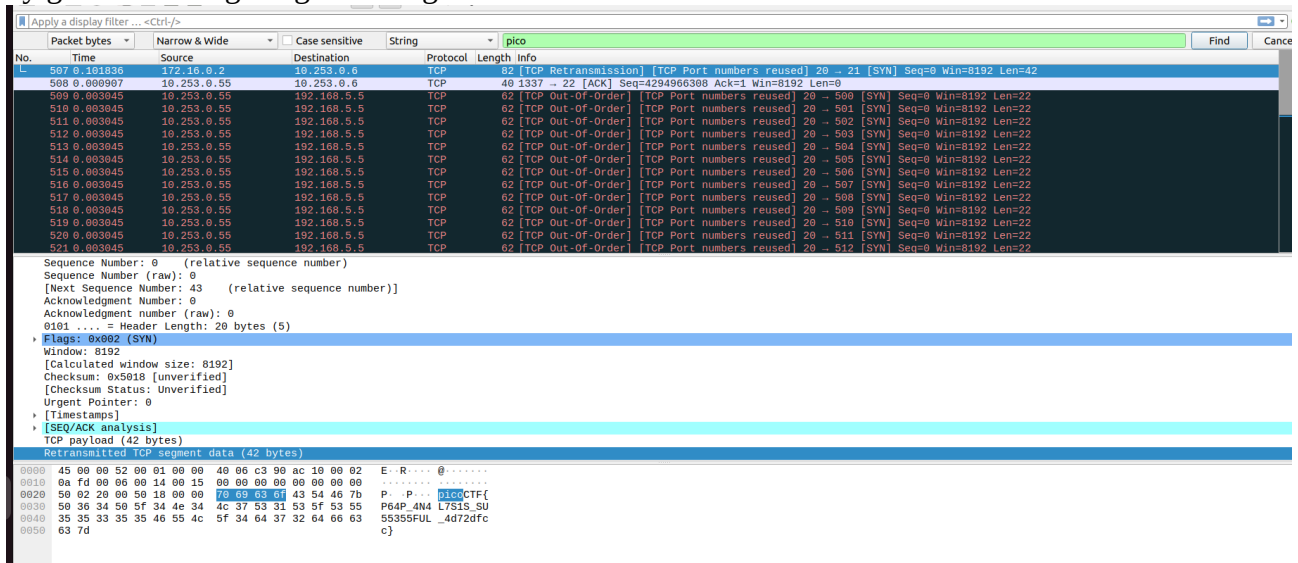
```
ctf-player@pico-chall$ sha256sum files/* | grep fba9f49bf22aa7188a155768ab0dfdc1f9b86c47976cd0f7c9003af2e20598f7
sha256sum: 'files/*': No such file or directory
ctf-player@pico-chall$ sha256sum files/* | grep fba9f49bf22aa7188a155768ab0dfdc1f9b86c47976cd0f7c9003af2e20598f7
sha256sum: 'files/*': No such file or directory
ctf-player@pico-chall$ cd ..
ctf-player@pico-chall$ sha256sum files/* | grep fba9f49bf22aa7188a155768ab0dfdc1f9b86c47976cd0f7c9003af2e20598f7
fba9f49bf22aa7188a155768ab0dfdc1f9b86c47976cd0f7c9003af2e20598f7  files/87590c24
ctf-player@pico-chall$ ./decrypt.sh files/87590c24
picoCTF{trust_but_verify_87590c24}
ctf-player@pico-chall$ Connection to rhea.picoctf.net closed by remote host.
Connection to rhea.picoctf.net closed.
ten1@ten1-IdeaPad-Slim-3-15IRH8:~$
```

PCAP POISONING

by downloading the file we get a trace.pcap
by opening it in the wire shark

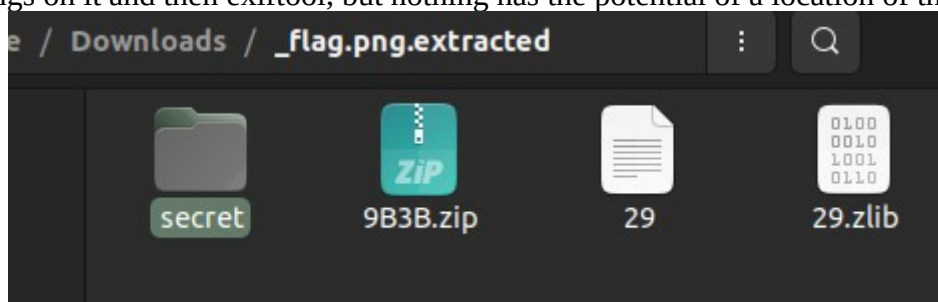
it has a lot of packets. The flag can be in the data transvered or in the bytes in the given list of the packets

we have a find option in the edit menu
by given checking we get the flag



HIDEME

i runned strings on it and then exiftool, but nothing has the potential of a location of the flag



then binwalk in the file extracted has a folder `/secret/` in it we get the flag(in like `flag.png` file)

```
picoCTF{Hiddinng_An_imag3_within_@n_ima9e_85e04ab8}
```

FINDANDOPEN

it has two files given one is the `dum.pcap` file and the other is a `flag.zip` file

now, to extract the `.zip` file we need a password which is in the `dump.pcap` file

in the `dump.pcap` file we have some hints like: flying on ethernet secret: is this the flag,couldtheflag have to be splitted?, we have some charcters with a '=' at the end

so the packet who's data has a '=' at the end has ome potential
by decoding it we get a part of the flag

```
AABBHPJ GTFRLKVG  
hpcyBpcy B0aGUgc2  
VjcmV0Oi BwaWNvQ1  
RGe1IzNE RJTkdfTE  
9LZF8=
```

Decode from Base64 format


Simply enter your data then push the decode button.

VGhpcyBpcyB0aGUgc2VjcmV0OiBwaWNvQ1RGe1IzNERJTkdfTE9LZF8=

 For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8  Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

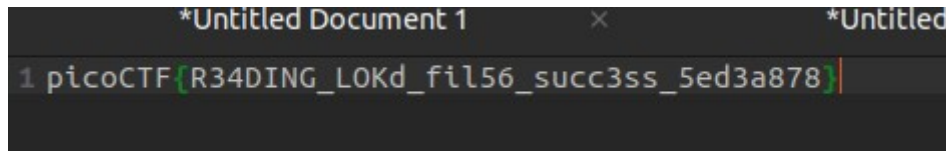
 Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

 **DECODE**  Decodes your data into the area below.

This is the secret: picoCTF{R34DING_LOKd_

by submitting as it is

gives us the final flag



HARD:

TCP Stream

- **TCP (Transmission Control Protocol)** is a transport-layer protocol.
- It provides a **reliable, ordered, and byte-stream-based** connection between two endpoints (like your browser and a server).

TLS Stream

- **TLS (Transport Layer Security)** is a cryptographic protocol that sits **on top of TCP**.
- It takes the raw TCP stream and encrypts it, ensuring:
 - **Confidentiality** (nobody can read the data),
 - **Integrity** (nobody can alter the data),
 - **Authentication** (you know who you're talking to, usually via certificates).
- A **TLS stream** is basically an **encrypted TCP stream**.
 - The application (like HTTPS, FTPS, IMAPS) sends data → TLS encrypts → TCP transmits → receiver decrypts.
 - To the application, it still looks like a clean stream of bytes, but secure.

A **TCP connection** wrapped in **TLS encryption**, often used in **HTTPS** or other secure protocols.

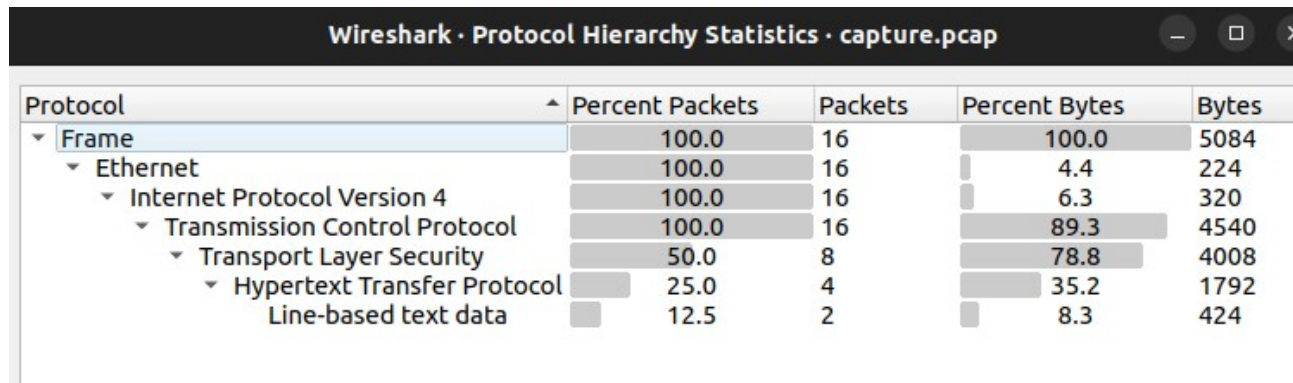
WEBNET0

we have two files given a .pcap file and a key file

using Wireshark in the .pcap file by the hints given to see the TLS stream

TLS is a protocol designed to ensure privacy and data integrity between client-server communications, often used to secure data transmitted over the internet.
we can see this by right clicking on the packet and then follow in that we have TLS stream

we see that the in the follow-> tcp stream we see that it is encrypted
If we see the protocol hierarchy we see that the the traffic is encrypted



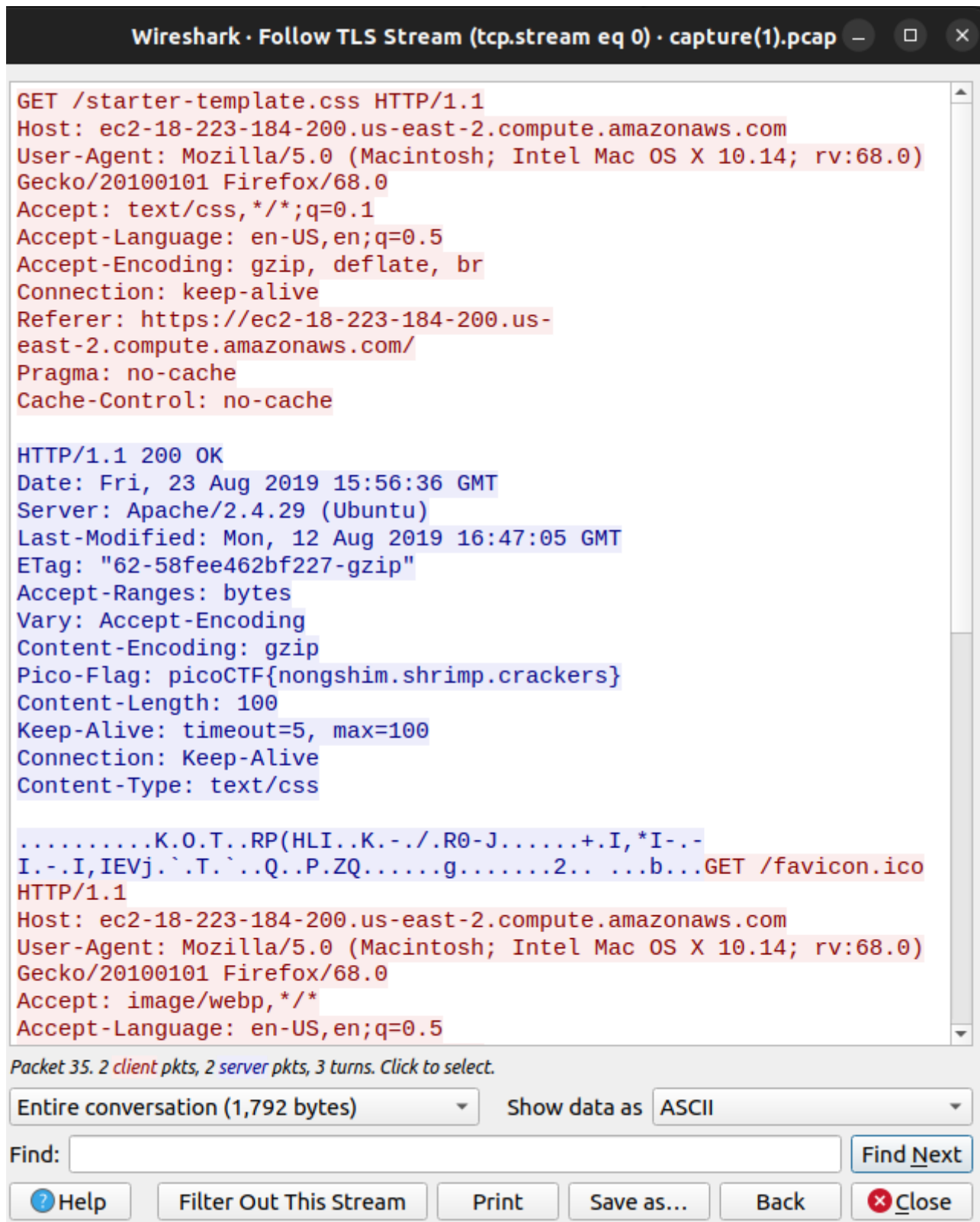
Wireshark · Protocol Hierarchy Statistics · capture.pcap

Protocol	Percent Packets	Packets	Percent Bytes	Bytes
Frame	100.0	16	100.0	5084
Ethernet	100.0	16	4.4	224
Internet Protocol Version 4	100.0	16	6.3	320
Transmission Control Protocol	100.0	16	89.3	4540
Transport Layer Security	50.0	8	78.8	4008
Hypertext Transfer Protocol	25.0	4	35.2	1792
Line-based text data	12.5	2	8.3	424

then with the key file provided try to decrypt the data

by going to edit -> preferences -> in protocols we see the TLS we have a RSA key to edit

there by uploading it the rsa key file and reloading the .pcap file by see the TCP stream we get the flag in that



WEBNET 1

this challenge has a .pcap file and a key file
then do the same process as the webnet0

by still by checking the follow -> tcptream it is still encrypted

Wireshark · Follow TLS Stream (tcp.stream eq 3) · capture(2).pcap

```

GET /second.html HTTP/1.1
Host: ec2-18-223-184-200.us-east-2.compute.amazonaws.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:68.0)
Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Fri, 23 Aug 2019 16:27:04 GMT
Server: Apache/2.4.29 (Ubuntu)
Last-Modified: Mon, 12 Aug 2019 17:35:36 GMT
ETag: "624-58feef3af8698-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Pico-Flag: picoCTF{this.is.not.your.flag.anymore}
Content-Length: 847
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

.....U]w...}n.E.k.....\...Q...G...-B.0..
$....o.Z;]3.....d.sv.{..@...De.{.V/@Q.{....9.n.QX
...4...."p.2..P(..4...P. !...BEf.....{...-gB. `.....nI../...`..
\...E.5e.(..+ .A...bfD..3..D..1%..@...0.'FT.....!UI.L0..&.G..
(..?.J.....Z....an.,...XM.....T...Q...&A..c....4..~5.o.K.
3.t.....h..80m./.....( .....7l..}.n....`R2Ab.
{..Y^f....N..R..... ..".....H.OE8h.Z...<...?..XXE...>..].-
4.p.!X9.....XXv....l@@...Q.....7.....;..&vw..n....[0...

1 client pkt, 1 server pkt, 1 turn.

```

No.	Time	Source	Destination	Protocol	Length	Info
17	0.892408	128.237.140.23	172.31.22.220	TCP	78	57581 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1386 WS=64 TSval=132865249 TSecr=0 SACK_PERM=1
18	0.892423	172.31.22.220	128.237.140.23	TCP	74	443 → 57581 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=8961 SACK_PERM=1 TSval=568332840 TSecr=132865249 WS=128
22	0.122048	128.237.140.23	172.31.22.220	TCP	66	57581 → 443 [ACK] Seq=1 Ack=1 Win=131904 Len=0 TSval=132865270 TSecr=568332840
23	0.122203	128.237.140.23	172.31.22.220	TLSv1.2	583	Client Hello
24	0.122229	172.31.22.220	128.237.140.23	TCP	66	443 → 57581 [ACK] Seq=1 Ack=518 Win=28932 Len=0 TSval=568332870 TSecr=132865270
25	0.122552	172.31.22.220	128.237.140.23	TLSv1.2	1073	Server Hello, Certificate, Server Hello Done
27	0.151669	128.237.140.23	172.31.22.220	TCP	66	57581 → 443 [ACK] Seq=518 Ack=1008 Win=130880 Len=0 TSval=132865303 TSecr=568332870
28	0.152210	128.237.140.23	172.31.22.220	TLSv1.2	384	Client Key Exchange, Change Cipher Spec, Finished
29	0.153206	172.31.22.220	128.237.140.23	TLSv1.2	324	New Session Ticket, Change Cipher Spec, Finished
30	0.183385	128.237.140.23	172.31.22.220	TCP	66	57581 → 443 [ACK] Seq=836 Ack=1260 Win=130752 Len=0 TSval=132865334 TSecr=568332901
31	0.187894	128.237.140.23	172.31.22.220	HTTP	500	GET / HTTP/1.1
32	0.188383	172.31.22.220	128.237.140.23	HTTP	1299	HTTP/1.1 200 OK (text/html)

by filtering the packets by searching it in the http packets then right click and then follow we get the tls stream to see it has a flag which says it is not the flag

by checking it fully give sthe flag

```

Wireshark - Follow TLS Stream (tcp.stream eq 0) - capture(2).pcap
Pragma: no-cache
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Fri, 23 Aug 2019 16:27:04 GMT
Server: Apache/2.4.29 (Ubuntu)
Last-Modified: Fri, 23 Aug 2019 16:26:33 GMT
ETag: "112fb-590cb44f2cbe6"
Accept-Ranges: bytes
Content-Length: 70395
Pico-Flag: picoCTF{this_is_not_your_flag_anymore}
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: image/jpeg

.....JFIF.....Exif.....MM.....J.....R.....Z.....picoCTF{honey.roasted.peanuts}.....ICC_PROFILE.....lcms.....mnrRGB XYZ
.....).....9acspAPPL.....lcms.....
desc.....^cmt.....

```

Investigative Reversing 0

In it we have a file which can be opened with a applications used for reversing like IDA, ghidra

then i have install ghidra

then i have opened it and it has a function void(main)

it has the function to decode the flag which is obtained by the .png file using xxd command or using hex editor(given in the hints)

```

Decompile: main - (mystery)
1
2 void main(void)
3
4 {
5     FILE *__stream;
6     FILE *__stream_00;
7     size_t sVar1;
8     long in_FS_OFFSET;
9     int local_54;
10    int local_50;
11    char local_38 [4];
12    char local_34;
13    char local_33;
14    char local_29;
15    long local_10;
16
17    local_10 = *(long *)(in_FS_OFFSET + 0x28);
18    __stream = fopen("flag.txt","r");
19    __stream_00 = fopen("mystery.png","a");
20    if (__stream == (FILE *)0x0) {
21        puts("No flag found, please make sure this is run on the server");
22    }
23    if (__stream_00 == (FILE *)0x0) {
24        puts("mystery.png is missing, please run this on the server");
25    }
26    sVar1 = fread(local_38,0x1a,1,__stream);
27    if ((int)sVar1 < 1) {
28        /* WARNING: Subroutine does not return */
29        exit(0);
30    }
31    puts("at insert");
32    fputc((int)local_38[0],__stream_00);
33    fputc((int)local_38[1],__stream_00);
34    fputc((int)local_38[2],__stream_00);
35    fputc((int)local_38[3],__stream_00);
36    fputc((int)local_34,__stream_00);
37    fputc((int)local_33,__stream_00);

```

then by understanding the function how the flag has been changed

Opens `flag.txt` (to read) and `mystery.png` (to append).

Reads 26 characters from `flag.txt`.

ppends them into `mystery.png`, but with transformations:

First 6 characters (index 0–5): written unchanged.

Next 9 characters (index 6–14): each written with +5 added to their ASCII value.

15th character (index 15): written with −3 subtracted from its ASCII value.

Last 10 characters (index 16–25): written unchanged.

Then by this we can create a python script to decode it

```
Pixels Per Unit Y      : 5669
Pixel Units            : meters
Warning                : [minor] Trailer data after PNG IEND chunk
Image Size             : 1411x648
Megapixels             : 0.914
teni@teni-IdeaPad-Slim-3-15IRH8:~/Downloads$ xxd -g 1 'mystery(3).png' | tail
0001e7f0: 82 20 08 82 20 08 82 20 08 82 20 64 1f 32 12 21  . . . . . d.2.!
0001e800: 08 82 20 08 82 20 08 82 20 08 42 f6 21 23 11 82  .. . . . .B.!#..
0001e810: 20 08 82 20 08 82 20 08 82 20 64 1f 32 12 21 08  .. . . . .d.2.!
0001e820: 82 20 08 82 20 08 82 20 08 42 f6 21 23 11 82 20  .. . . . .B.!#..
0001e830: 08 82 20 08 82 20 08 82 20 64 1f 32 12 21 08 82  .. . . . .d.2.!..
0001e840: 20 08 82 20 08 82 20 08 42 f6 21 23 11 82 20 08  .. . . . .B.!#.. .
0001e850: 82 20 08 82 20 08 82 20 64 17 ff ef ff fd 7f 5e  .. . . . .d.....^
0001e860: ed 5a 9d 38 d0 1f 56 00 00 00 00 49 45 4e 44 ae  .Z.8..V....IEND.
0001e870: 42 60 82 70 69 63 6f 43 54 4b 80 6b 35 7a 73 69  B`.picoCTK.k5zsi
0001e880: 64 36 71 5f 64 31 64 65 65 64 61 61 7d          d6q_d1deedaa}
teni@teni-IdeaPad-Slim-3-15IRH8:~/Downloads$ cd ..
teni@teni-IdeaPad-Slim-3-15IRH8:~$ python3 reversing.py
Flag: picoCTF{f0und_1t_5266a857}
teni@teni-IdeaPad-Slim-3-15IRH8:~$
```