**NAME : PRATIK JAIN**

## Wireshark Capture

While sitting in a coffee shop, I captured a Wireshark trace of the network traffic. There seems to be some hidden data in the air. Can you retrieve the concealed information?

192.168.49.134 made a TCP handshake with 10.10.10.2 vie three way handshake(SYN SYN-ACK ACK).

Data conceived:

Source: 192.168.49.134 Destination:10.10.10.2:21 Type: FTP USER: 0ffs3cUs3r3 PASSWORD: very_secret_password

Open Ports found:

| | |
|---|---|
| 10.10.10.2 | 21 ftp |
| | 80 http |
| www.offensive-security.com (192.124.249.5) | |
| | 80 http |

DNS Server:

IP:192.168.49.134 DNS Server: 192.168.49.2 Name: www.offensive-security.com

Hidden Secret Flag:

GET /flag.jpg HTTP/1.1

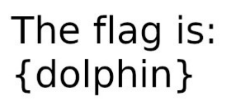
Host: 10.10.10.2

Accept: */*

User-Agent: {Ariadne}

A JFIF file found Which is operated using GIMP. So

flag.jpg is found through a GET request



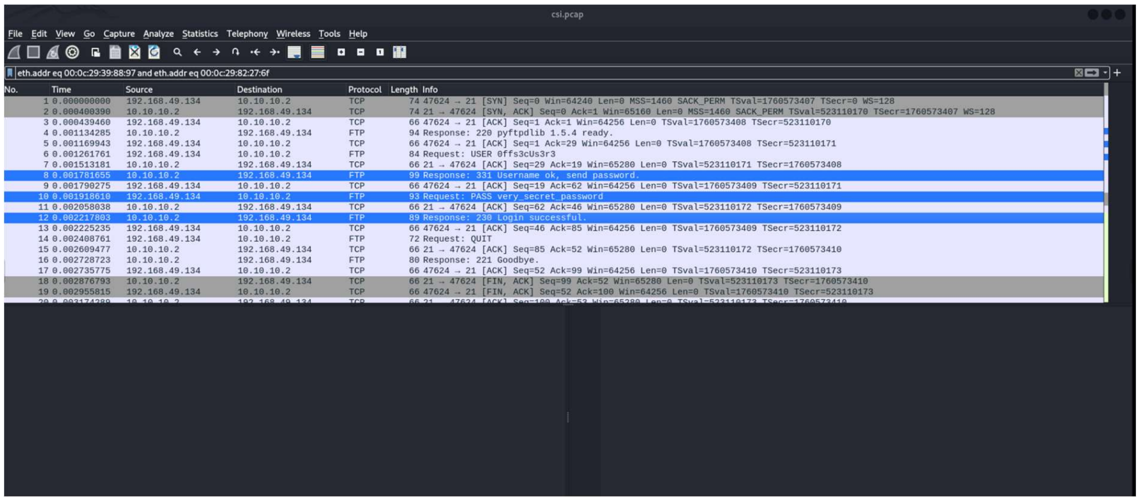
The flag is:
{dolphin}

{dolphin}

Tool used: Wireshark.

Methodology: Opened the pcap file in Wireshark and analyzed all the source and destination IP addresses, Hosts, Server, Open Ports, and all the FTP requests.

Secret Hidden flag: A JFIF file found which is operated using GIMP. So used this information to deduce the image in the form jpeg.


csi.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

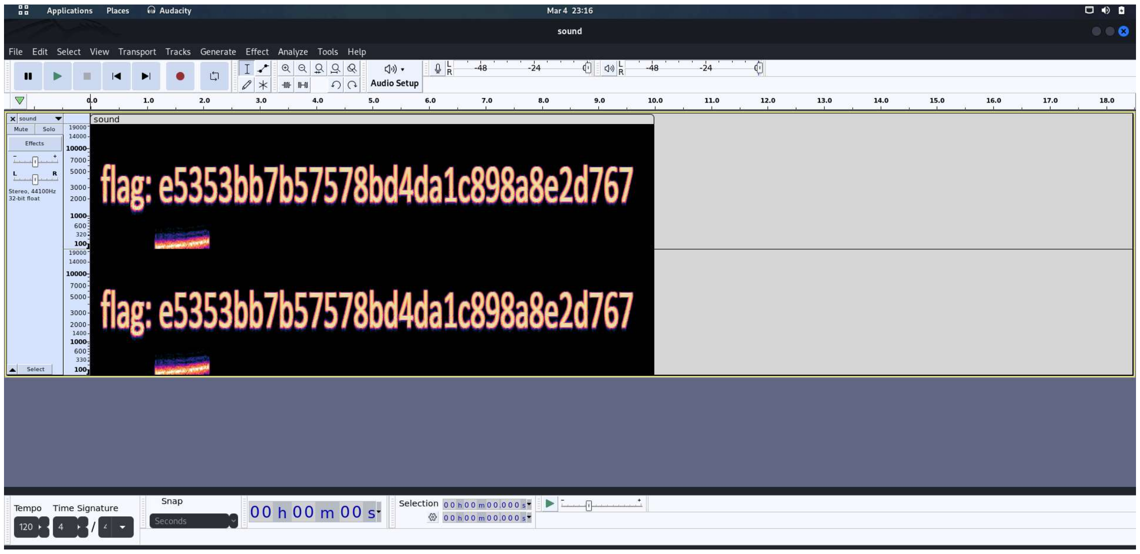
```
eth.addr eq 00:0c:29:39:88:97 and eth.addr eq 00:0c:29:82:27:6f
```

```
No.  Time         Source          Destination     Protocol Length Info
   1 0.000000000  192.168.49.134  10.10.10.2      TCP      74 47624 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1760573407 TSecr=0 WS=128
   2 0.000400390  10.10.10.2      192.168.49.134  TCP      74 21 → 47624 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=523110170 TSecr=1760573407 WS=128
   3 0.000439460  192.168.49.134  10.10.10.2      TCP      66 47624 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1760573408 TSecr=523110170
   4 0.001134285  10.10.10.2      192.168.49.134  FTP      94 Response: 220 pyftpdlib 1.5.4 ready.
   5 0.001169943  192.168.49.134  10.10.10.2      TCP      66 47624 → 21 [ACK] Seq=1 Ack=29 Win=64256 Len=0 TSval=1760573408 TSecr=523110171
   6 0.001261761  192.168.49.134  10.10.10.2      FTP      84 Request: USER 0ffs3cUs3r3
   7 0.001513181  10.10.10.2      192.168.49.134  TCP      66 21 → 47624 [ACK] Seq=29 Ack=19 Win=65280 Len=0 TSval=523110171 TSecr=1760573408
   8 0.001781655  10.10.10.2      192.168.49.134  FTP      99 Response: 331 Username ok, send password.
   9 0.001790275  192.168.49.134  10.10.10.2      TCP      66 47624 → 21 [ACK] Seq=19 Ack=62 Win=64256 Len=0 TSval=1760573409 TSecr=523110171
  10 0.001918610  192.168.49.134  10.10.10.2      FTP      93 Request: PASS very_secret_password
  11 0.002058038  10.10.10.2      192.168.49.134  TCP      66 21 → 47624 [ACK] Seq=62 Ack=46 Win=65280 Len=0 TSval=523110172 TSecr=1760573409
  12 0.002217803  10.10.10.2      192.168.49.134  FTP      89 Response: 230 Login successful.
  13 0.002225235  192.168.49.134  10.10.10.2      TCP      66 47624 → 21 [ACK] Seq=46 Ack=85 Win=64256 Len=0 TSval=1760573409 TSecr=523110172
  14 0.002408761  192.168.49.134  10.10.10.2      FTP      72 Request: QUIT
  15 0.002609477  10.10.10.2      192.168.49.134  TCP      66 21 → 47624 [ACK] Seq=85 Ack=52 Win=65280 Len=0 TSval=523110172 TSecr=1760573410
  16 0.002728723  10.10.10.2      192.168.49.134  FTP      80 Response: 221 Goodbye.
  17 0.002735775  192.168.49.134  10.10.10.2      TCP      66 47624 → 21 [ACK] Seq=52 Ack=99 Win=64256 Len=0 TSval=1760573410 TSecr=523110173
  18 0.002876793  10.10.10.2      192.168.49.134  TCP      66 21 → 47624 [FIN, ACK] Seq=99 Ack=52 Win=65280 Len=0 TSval=523110173 TSecr=1760573410
  19 0.002955815  192.168.49.134  10.10.10.2      TCP      66 47624 → 21 [FIN, ACK] Seq=52 Ack=100 Win=64256 Len=0 TSval=1760573410 TSecr=523110173
  20 0.003174289  10.10.10.2      192.168.49.134  TCP      66 21 → 47624 [ACK] Seq=100 Ack=53 Win=65280 Len=0 TSval=523110173 TSecr=1760573410
```

**Problem: Sound.wav File**

Sound files don't just contain sounds.

Analysing the file using audacity steganography analysis:


Applications
Places
Audacity
Mar 4 23:16
sound
File Edit Select View Transport Tracks Generate Effect Analyze Tools Help
Audio Setup
Stereo, 44100Hz
32-bit float
flag: e5353bb7b57578bd4da1c898a8e2d767
flag: e5353bb7b57578bd4da1c898a8e2d767
Tempo
Time Signature
Snap
Seconds
Selection

Flag found: e5353bb7b57578bd4da1c898a8e2d767

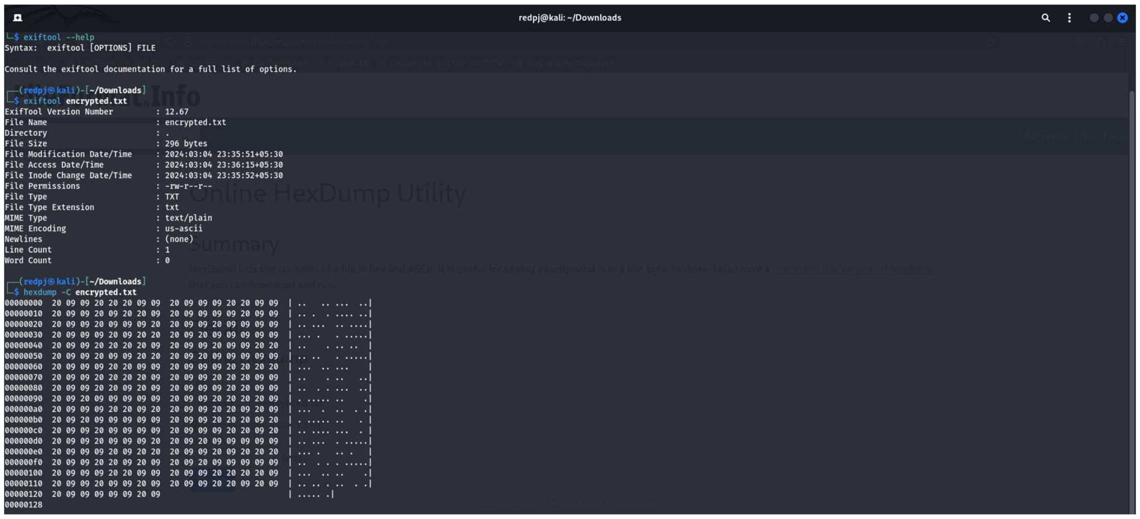
Tool used: Audacity.

Methodology: Loaded the file in Audacity and changed the waveform to Spectrogram to reveal the information hidden in .wav file.

Problem: Blank Space Decoding.

Tools used: hexdump and python.

Since the file contained some empty characters I used hexdump tool and the output is:



```
redpj@kali: ~/Downloads
└─$ exiftool --help
Syntax:  exiftool [OPTIONS] FILE

Consult the exiftool documentation for a full list of options.

┌──(redpj㉿kali)-[~/Downloads]
└─$ exiftool encrypted.txt
ExifTool Version Number         : 12.67
File Name                       : encrypted.txt
Directory                       : .
File Size                       : 296 bytes
File Modification Date/Time     : 2024:03:04 23:35:51+05:30
File Access Date/Time           : 2024:03:04 23:36:15+05:30
File Inode Change Date/Time     : 2024:03:04 23:35:52+05:30
File Permissions                : -rw-r--r--
File Type                       : TXT
File Type Extension             : txt
MIME Type                       : text/plain
MIME Encoding                   : us-ascii
Newlines                        : (none)
Line Count                      : 1
Word Count                      : 0

┌──(redpj㉿kali)-[~/Downloads]
└─$ hexdump -C encrypted.txt
00000000  20 09 09 20 20 20 09 09  20 09 09 09 20 20 09 09  | ..   .. ...  ..|
00000010  20 09 09 20 09 20 20 09  20 09 09 09 09 20 09 09  | .. .  . .... ..|
00000020  20 09 09 20 09 09 09 20  20 09 09 20 09 09 09 09  | .. ...  .. ....|
00000030  20 09 09 09 20 09 20 20  20 09 20 09 09 09 09 09  | ... .   . .....|
00000040  20 09 09 20 20 20 20 09  20 09 09 20 09 09 20 20  | ..    . .. ..  |
00000050  20 09 09 20 09 09 20 20  20 09 20 09 09 09 09 09  | .. ..   . .....|
00000060  20 09 09 09 20 20 09 09  20 09 09 09 20 20 20 20  | ...  .. ...    |
00000070  20 09 09 20 20 20 20 09  20 09 09 20 20 20 09 09  | ..    . ..   ..|
00000080  20 09 09 20 20 09 20 09  20 09 09 09 20 20 09 09  | ..  . . ...  ..|
00000090  20 09 20 09 09 09 09 09  20 09 09 20 20 20 20 09  | . ..... ..    .|
000000a0  20 09 09 09 20 20 09 20  20 09 09 20 20 09 20 09  | ...  .  ..  . .|
000000b0  20 09 20 09 09 09 09 09  20 09 09 20 20 20 09 20  | . ..... ..   . |
000000c0  20 09 09 20 09 09 09 09  20 09 09 09 20 20 09 20  | .. .... ...  . |
000000d0  20 09 09 20 09 09 09 20  20 09 20 09 09 09 09 09  | .. ...  . .....|
000000e0  20 09 09 09 20 09 20 20  20 09 09 20 09 20 20 20  | ... .   .. .   |
000000f0  20 09 09 20 20 09 20 09  20 09 20 09 09 09 09 09  | ..  . . . .....|
00000100  20 09 09 09 20 20 09 09  20 09 09 20 20 20 20 09  | ...  .. ..    .|
00000110  20 09 09 20 09 09 20 09  20 09 09 20 20 09 20 09  | .. .. . ..  . .|
00000120  20 09 09 09 09 09 20 09                           | ..... .|
00000128
```

After that I copied all the values and wrote a simple python program:

The original string **s** consists of a sequence of numbers separated by spaces. Each number is either '20' or '09'.

In the code, **s.replace('20', '0').replace('09', '1').replace(' ', '')** is used to manipulate the original string.

Here's what each **.replace()** function does:

**replace('20', '0')**: Replaces every occurrence of '20' with '0'.

**replace('09', '1')**: Replaces every occurrence of '09' with '1'.

**replace(' ', '')**: Removes all spaces from the string.

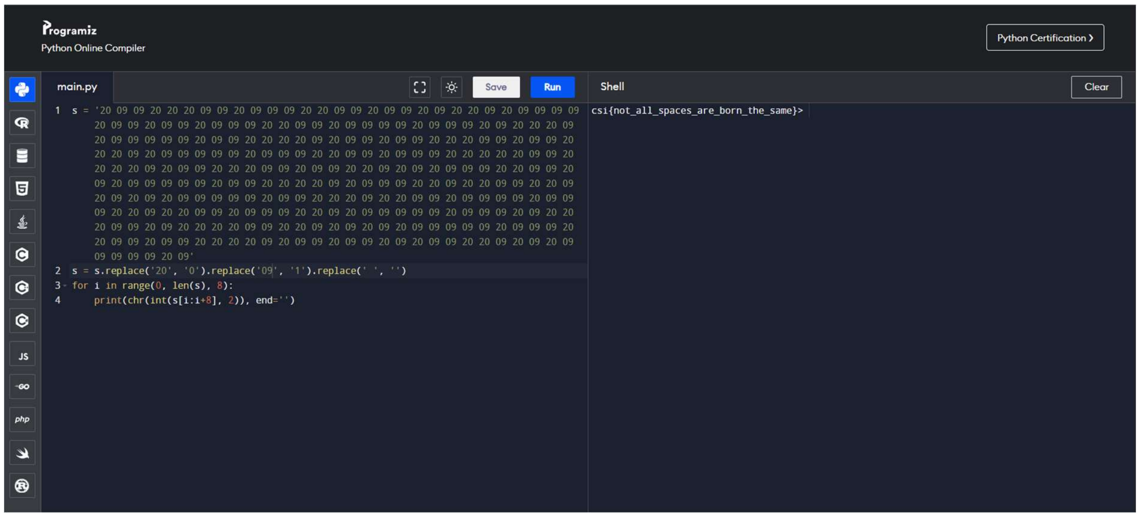
After these replacements, the string **s** now consists of only '0's and '1's without any spaces.

The code then processes the binary string in chunks of 8 characters (s[i:i+8]).

It converts each chunk from binary to decimal using int(s[i:i+8], 2).

Then, it converts the decimal value to the corresponding ASCII character using chr().

The resulting characters are printed out.



```python
s = '20 09 09 20 20 20 09 09 20 09 09 09 20 20 09 09 20 09 09 20 09 20 20 09 20 09 09 09 09
     20 09 09 20 09 09 20 09 09 09 20 20 09 09 20 09 09 09 09 20 09 09 09 20 09 20 20 20 09
     20 09 09 09 09 09 20 09 09 20 20 20 20 09 20 09 09 20 09 09 20 20 20 09 09 20 09 09 20
     20 20 09 20 09 09 09 09 09 20 09 09 09 20 20 09 09 20 09 09 09 20 20 20 20 20 09 09 20
     20 20 20 09 20 09 09 20 20 20 09 09 20 09 09 20 20 09 20 09 20 09 09 09 20 20 09 09 20
     09 20 09 09 09 09 09 20 09 09 20 20 20 20 09 20 09 09 09 20 20 09 20 20 09 09 20 20 09
     20 09 20 09 20 09 09 09 09 09 20 09 09 20 20 20 09 20 20 09 09 20 09 09 09 09 20 09 09
     09 20 20 09 20 20 09 09 20 09 09 09 20 20 09 20 09 09 09 09 09 20 09 09 09 20 09 20 20
     20 09 09 20 09 20 20 20 20 09 09 20 20 09 20 09 20 09 20 09 09 09 09 09 20 09 09 09 20
     20 09 09 20 09 09 20 20 20 20 09 20 09 09 20 09 09 20 09 20 09 09 20 20 09 20 09 20 09
     09 09 09 09 20 09'
s = s.replace('20', '0').replace('09', '1').replace(' ', '')
for i in range(0, len(s), 8):
    print(chr(int(s[i:i+8], 2)), end='')
```

```
csi{not_all_spaces_are_born_the_same}>
```

OUTPUT: flag

csi{not_all_spaces_are_born_the_same}>