【技术分享】最新2016华山杯CTF writeup

转载

普通网友 于 2016-09-13 12:53:39 发布 2920 收藏
分类专栏： CTF


CTF 专栏收录该内容
10 篇文章 0 订阅
订阅专栏

作者：FlappyPig

稿费：700RMB

投稿方式：发送邮件至linwei#360.cn，或登陆网页版在线投稿

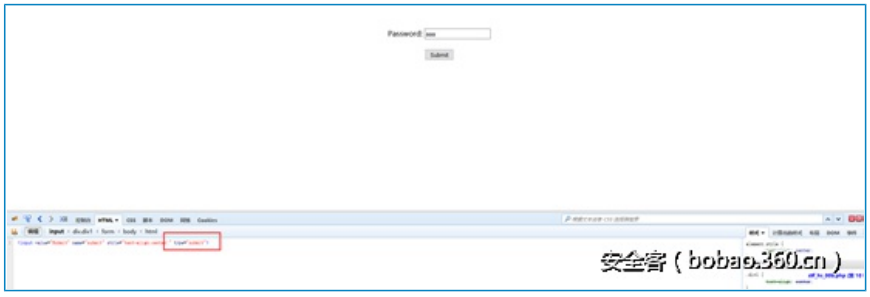
2016 华山杯 网络安全技能大赛 解题报告

队伍: FlappyPig

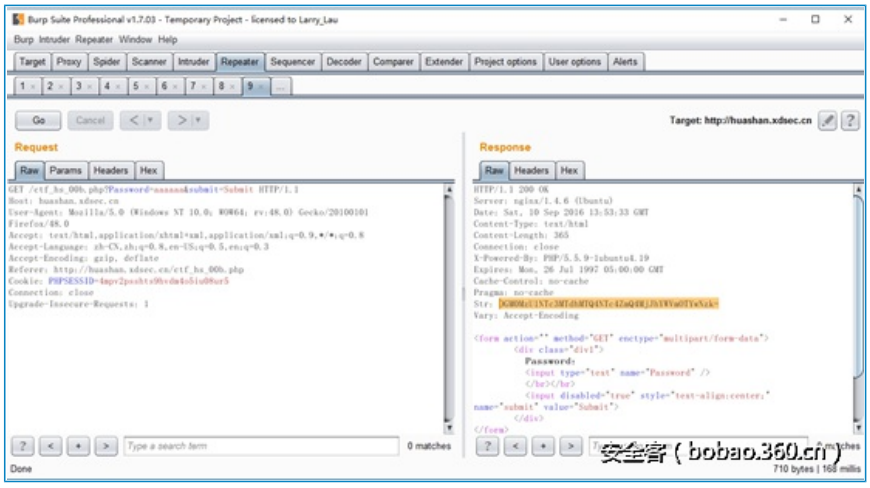
Web渗透

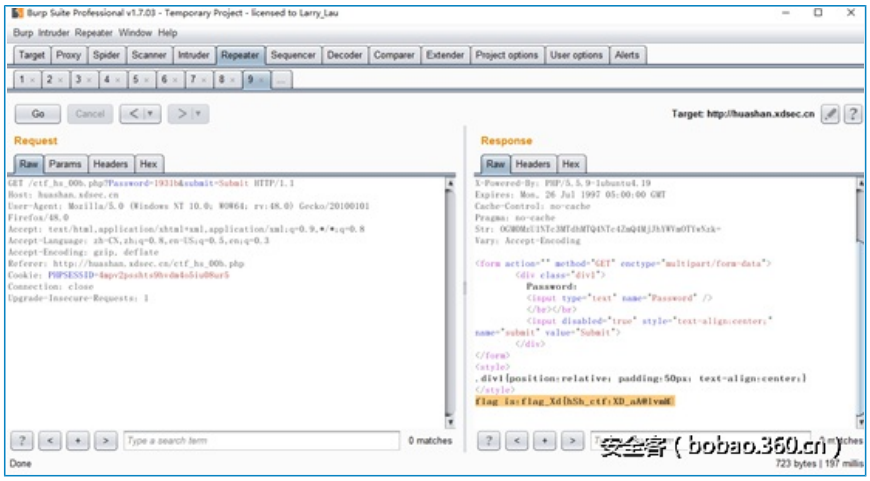
0x01 打不过~

添加type=”submin”，点击提交抓包



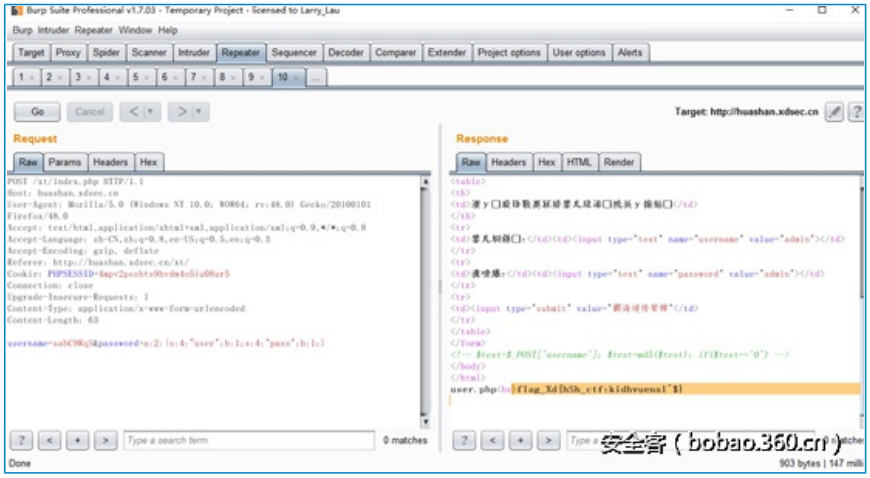
有一串字符串，base64->md5，1931b。提交getflag





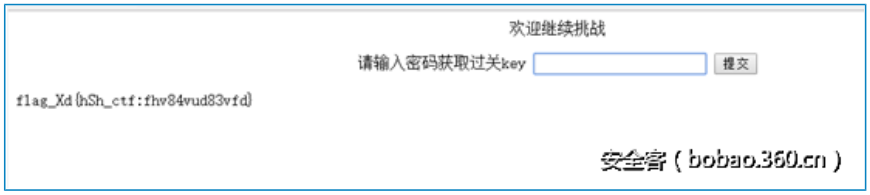
0x02 系统管理

源码有代码，先找0e开头的md5，然后user.php，直接反序列化绕过



0x03 简单js

看了下js,直接alert(a)　14208



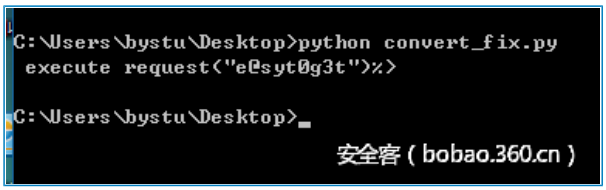
0x04 弹弹弹！

<img src=1 onerror=alert(1)>

0x05 233

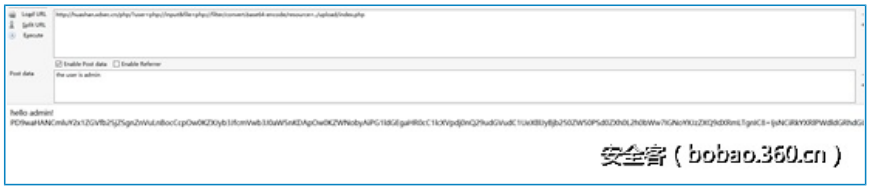
Jsfuck，解密后是一句话

用工具解不开，直接自己写脚本吧ANSI->Unicode



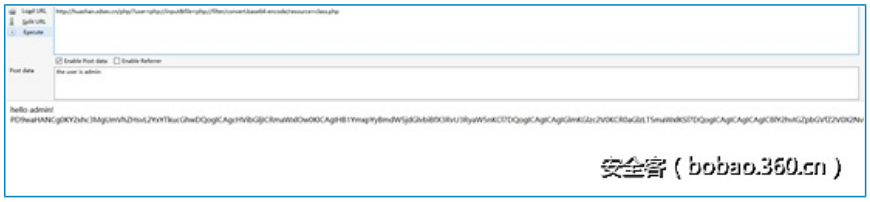
0x06 无间道

这题怀疑出错了吧，函数的都没定义，咋传？还没get到出题人的意图，通过下一题直接读的源码



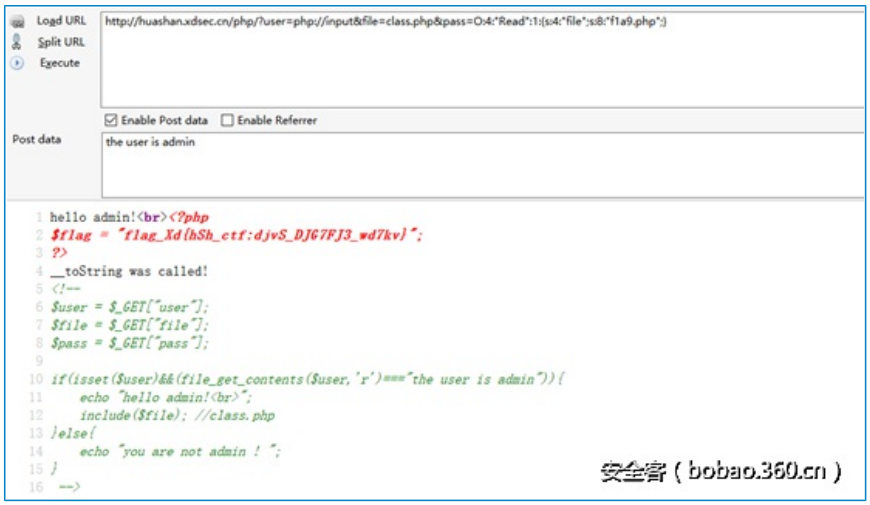
0x07 php很烦人

先看源码，用php://input改成admin，然后可以直接读文件，index中有个class.php



```
1    <span style="color: rgb(0, 0, 0);"><?php<br>class Read{//f1a9.php<br>    public $file;<br>    put
```

没法直接读f1a9.php,反序列化去读

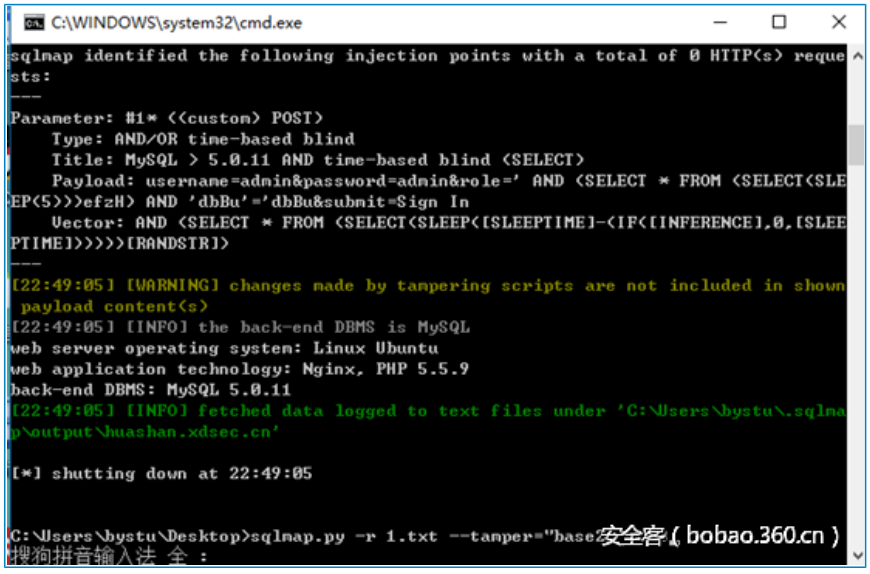


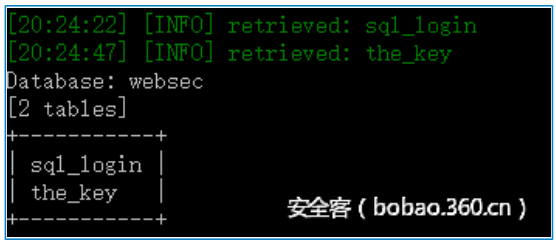
0x08 More try

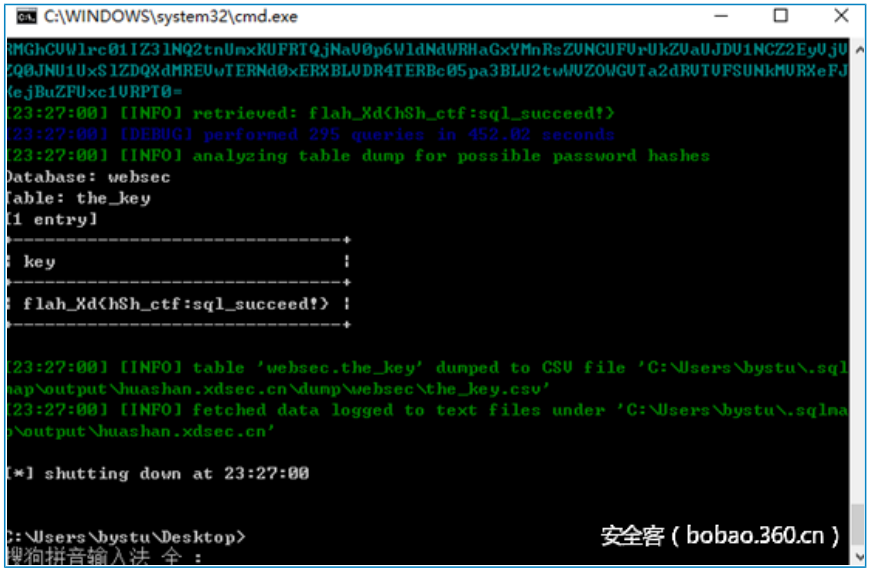
靠上个题读到源码，然后看了下role有注入，还有两层base64

Sqlmap有个base64encode.py的tamper，所以自己改下，改成两层

然后sqlmap.py –r –tamper=” base2.py” –v 3 ,the_key表，key字段





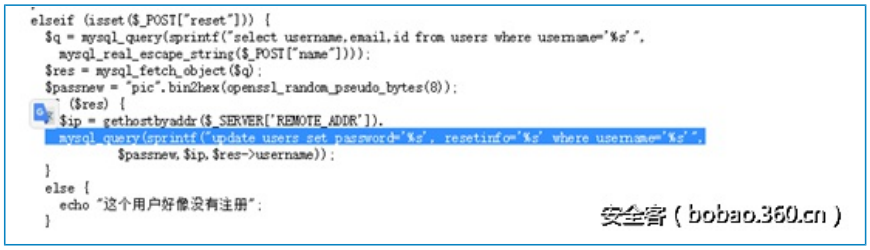


0x0A 三秒钟记忆

http://huashan.xdsec.cn/pic/login

这里可以看到源码，

重置密码的地方可以二次注入



注册的时候带’ 的用户名，然后重置密码的时候会注入

‘ and LEFT ((select flag from flag),x)=’ flag_Xd{hSh_ctf:dutwq}’

如果充值成功了，密码就会变，所以就无法登陆了，写脚本跑下就好了,太慢了……

0x0B 疯狂的js

这个是plaidctf2014的原题，不过改了一个地方，

var args = [].slice.apply(arguments).sort().filter(function(x,i,a){return a.indexOf(x) == i;});

if(args.length != 5) return "数够参数了吗？";

var flag = false; args.map(function(x){flag |= x >= 999;});

if(flag) return "有点大了哦";

```
var m = args.map(cal);

if(m.filter(function(x,i){return m[2]+3*i==x;}).length < 1) return "no";

if(m.filter(function(x,i){return x == args[i];}).length < 2) return "nono";

if(m.filter(function(x,i){return x > m[i-1];}).length > 2) return "bala";

if(m.filter(function(x,i){return x < m[i-1];}).length > 1) return "balana~";
```

满足条件即可弹出flag

五次分别输入

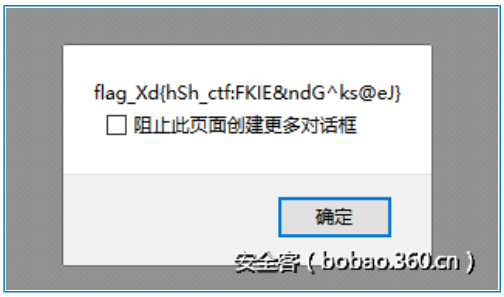
2.0

2.00

6

76

949



Reverse逆向破解

0x01 Crackme1. Warming Up

代码就是个简单变化，动态跟了几步，发现进行了如下操作：

```
"""

xor 0x30 ^ 1

xor 0x32 ^ 2

xor 0x33 ^ 3

xor 0x34 ^ 1

xor 0x35 ^ 2

"""
```

最后进行字符串比较，反过来写下就可以，如下：

```
target = "VgobmndVlBVE"

result = ""

for index, item in enumerate(target):

result += chr(ord(item)^(((index)%3)+1))

print result
```

0x02 Crackme2. 到手的钥匙

这题的逻辑就不是常人的，有两个用户名和密码。

开始那个还正常点

用户名：amdin，

密码的md5值知道，然后反查了下值为：xdadmin

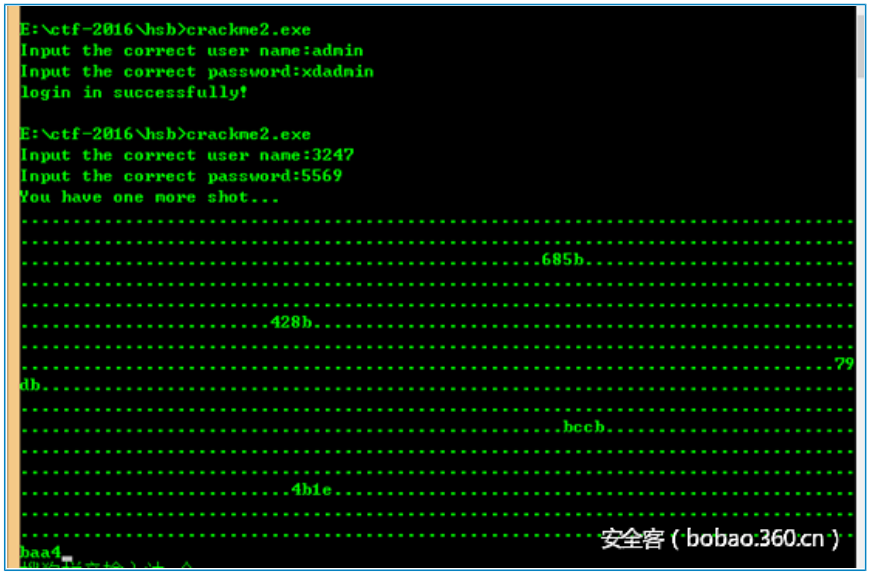
```
v14 = v20 - v10;
v13 = md5_401000(pass_41630C, v20 - v10, v4);
if ( !v13 )
  goto LABEL_24;
v16 = aEc9add6bd30ea6;
v17 = v4;
while ( 1 )
{
  v22 = *v17;
  v1 = v22 < (unsigned __int8)*v16;
  if ( v22 != *v16 )
    break;
  if ( !v22 )
```


但是输完并没有什么用，提交也不对：

后来发现另外一个函数也用到了用户名密码，如下：

```
  v0 = sub_402DA7(user_416300);
  result = sub_402DA7(pass_41630C);
  v2 = result;
  if ( v0 == 3247 && result == 5569 )
  {
    for ( i = 0; i < 6; ++i )
    {
      for ( j = 0; j < 8; ++j )
      {
        for ( k = 0; k < 30; ++k )
        {
          if ( j >= 7 )
          {
            sub_401090(j, i);
            break;
          }
          printf((int)a_, v2);
        }
      }
```


输入3247,5569得到如下：



组合下输出的结果，提交不对，，，，，，，，，，

不对啊，，，，，，，，，，

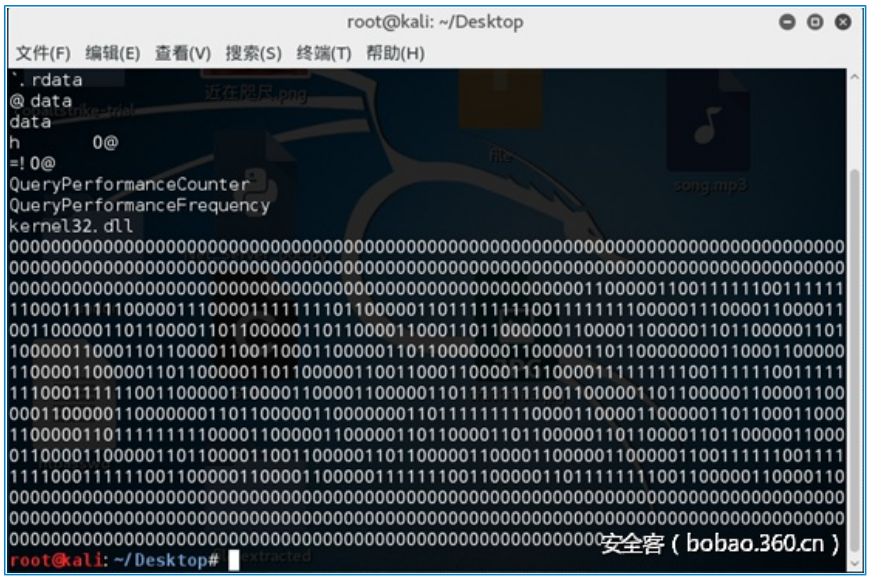
仔细看代码，没别的逻辑了啊，，，，纠结了好久

突然队友提交了“用户名+密码+输出”的结果，通过。。。。。。。

竟然通过了。。。。。。。。

0x03 Crackme3. 探囊取物

题目是个图片，直接strings crackme3.jpg，得到一串01



目测可以拼出来字，一共1177个，11*107

0x04 Crackme4. 移动迷宫

代码就是个简单的走迷宫，图如下：

输入的东西，进行各简单变化，对应于走的方向，如下：

```
6
7   v3 = 0;
8   for ( i = 0; i < 6; ++i )
9   {
0     while ( v3 < 4 )
1     {
2       if ( a1[i] == *(&a0a1ba2b34c5dc6[4 * i] + v3) )
3       {
4         a2[i] = v3 + 1;
5         break;
6       }
7       ++v3;
8     }
9     v3 = 0;
0     result = i + 1;
1   }
2   return result;
```


走的逻辑如下：

```
for ( i = 0; i < 4; ++i )
  sub_401000(&v6[6 * i], &v8[6 * i]);
while ( ii < 24 )
{
  switch ( v8[ii] )
  {
    case 1:
      --pos_x;
      break;
    case 2:
      ++pos_x;
      break;
    case 3:
      --pos_y;
      break;
    case 4:
      ++pos_y;
      break;
    default:
      break;
  }
  if ( *(&asc_40E018[10 * pos_x] + pos_y) != '#' )
  {
```


根据坐标生成方向即可，逆代码如下：

```
<span style="color: rgb(0, 0, 0);">map_info = "**********#####******#**#*****##*##*******#*****
```

flag如下：最后一行

```
[(1, 1), (1, 2), (1, 3), (1, 4), (2, 1), (2, 4), (3, 0), (3, 1), (3, 3), (3, 4),
 (4, 3), (5, 3), (5, 5), (5, 6), (5, 7), (5, 8), (5, 9), (6, 3), (6, 4), (6, 5),
 (6, 9), (7, 9), (8, 9), (9, 8), (9, 9)]
24
[4, 1, 1, 4, 4, 4, 2, 2, 3, 2, 2, 2, 4, 4, 1, 4, 4, 4, 4, 2, 2, 2, 2, 3]
24
['0A1B', 'a2b3', '4C5D', 'c6d7', '8E9F', 'e0f1']
Ba47F1A256E0B347F1B2C6Ef
```


0x05 Crackme5. Do something

虽然题目给了个jpg，但其实是个程序，主要的判断逻辑如下：

```
<span style="color: rgb(0, 0, 0);">int __cdecl sub_401000(char *Src)<br>{<br>  char Dst[20]; // [
```

就是一些列的条件，满足就会输出得到了flag，约束如下：

```
<span style="color: rgb(0, 0, 0);">Dst[0] = Dst[8]<br>Dst[0] == Dst[9]<br>Dst[1] == Dst[10]<br>Ds
```

直接用个求解器求解即可，结果如下：

```
sat
[Dst__13 = 12,
 Dst__1 = 8,
 Dst__3 = 19,
 Dst__4 = 9,
 Dst__10 = 8,
 Dst__15 = 7,
 Dst__14 = 1,
 Dst__8 = 20,
 Dst__12 = 6,
 Dst__2 = 9,
 Dst__0 = 20,
 Dst__6 = 14,
 Dst__5 = 19,
 Dst__9 = 20,
 Dst__16 = 1,
 Dst__7 = 15,
 Dst__11 = 5]
```


直接算出与加上0x60即可得到flag，如下：

0x06 Crackme6. Help me

这题目就是运行得时候，对一些不可访问的地址进行了写入，导致崩溃，看代码貌似是专门这样写的，如下：

```
.text:0040104D                inc     ecx
.text:0040104E                mov     eax, 10h
.text:00401053                mov     dword ptr [eax], 2
.text:00401059                mov     ecx, 6
```

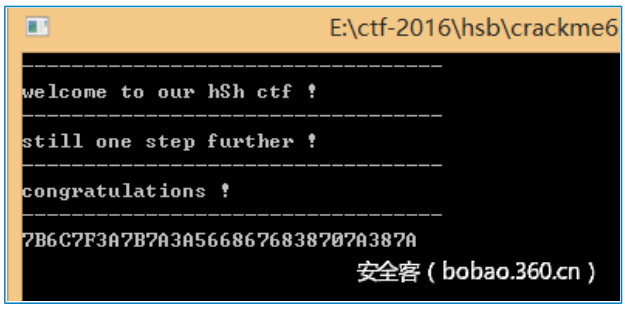

直接对[0x10]处进行了赋值，程序这样的位置还有好几处，如下：

```
::v10 = 2;
qmemcpy(&v10, "still one step f
dword_40DCFC = 16;
dword_40CF70 = 1;
v11 = aStillOneStepFu[26];
memset(&v12, 0, 0x49u);
printf(&v10);
printf("----------------------
dword_40DCF8 = 0;
v0 = 0x12345678;
dword_40CF74 = -1;
v3 = lstrlenA(String);
v4 = 0;
printf(aCongratulation);
printf("----------------------
dword_40DD00 = 19;
v13 = 0x12345678;
dword_40DCF8 = 0;
v0 = 0x12345678;
dword_40CF
```


直接对其进行nop，然后将输出，转成printf即可，如下:

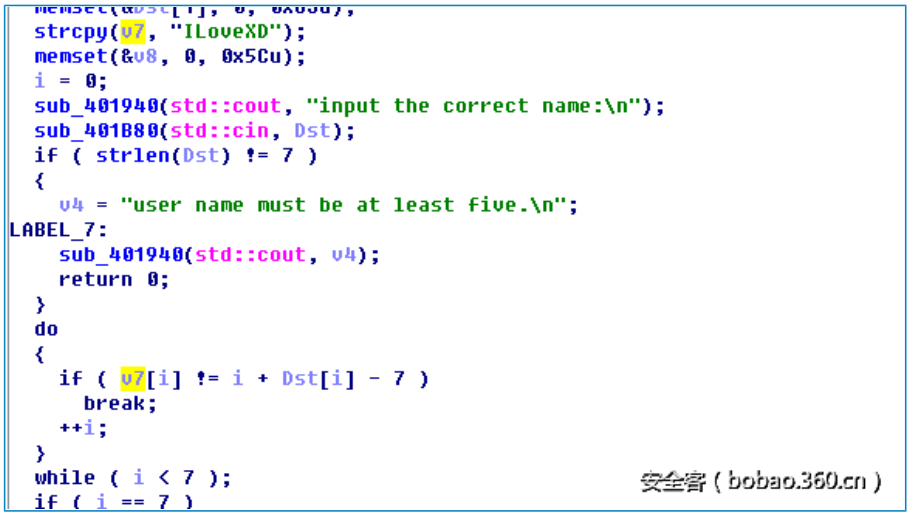
```
    OutputString = 0;
    sub_401200("%2X", v6);
    OutputDebugStringA(&OutputString);
    ++v4;
```


Flag直接就打印出来了，如下：



0x08 Crackme8. 忘记用户名

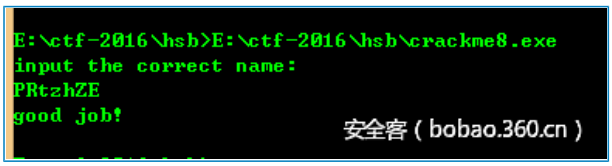
代码就很简单，如下图：



直接计算即可， 代码如下：

```
info = "ILoveXD"

result_info = ""

for i in range(7):

result_info += chr(ord(info[i])+7-i)

print result_info
```

结果如下：



0x09 Crackme9. 捉迷藏

用户名: FindKey

密码：T25Zb3VyQ29tcHV0ZXI= base64解码得: OnYourComputer

生成了一个flag.jpg，里面的内容为FindKeyOnYourComputerArvinShow

Flag的品相好差。

Crypto加密解密

0x01 紧急报文

ADFGX加密

0x02 is it x or z ?

给了3个文件 clear-1.txt crypt-1.txt和crypt-2.txt，用clear-1.txt和crypt-1.txt异或可以得到重复循环的片段，推测循环节即为密钥，用该密钥解密crypt-2即可得到flag

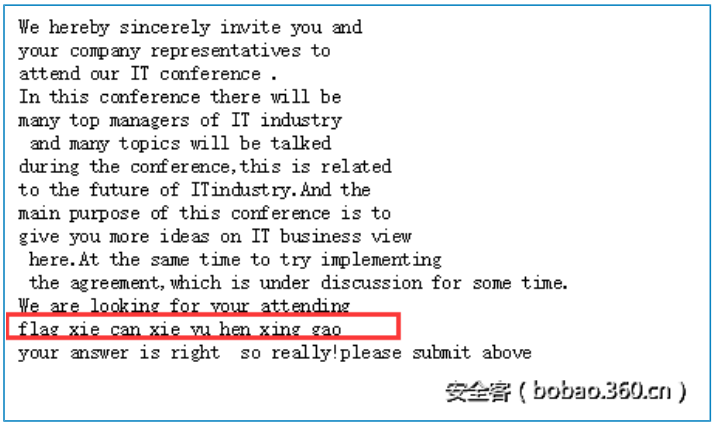
0x03 分组加密模式检测

这是个原题，见这里：https://github.com/truongkma/ctf-tools/tree/master/cryptopals-solutions-master/set1/8

主要就是从一大堆CBC密文里检测出ECB密文，脚本一模一样抄即可。

0x04 修复一下这份邀请函的部分内容

打开就是flag，明文，直接交



flag_Xd{hSh_ctf:flag xie can xie yu hen xing gao}

0x0 5协议？认证？加密？

这题先进行了DH交换密钥，然后用交换后的密钥加密的flag。A B P都不是很大，猜想这个离散对数问题比较容易解。

https://www.alpertron.com.ar/DILOG.HTM

用这个工具可以直接求解出离散对数算出a的私有指数，然后计算B^a就作为密钥了。但是这题有一个地方很坑，得到的密钥只有8个字节，但是AES需要16个字节作为密钥，一开始卡这里卡了很久。后来才脑洞出来高位全部补\x00，然后解完发现后一半flag是乱码，又是很坑，后来用CBC模式试了一下，iv取全0，解出来才正常。

0x06 时间决定一切

web的任意文件读取，直接读源码

Android

0x01 错错错

这题算法其实很简单，就是对随机字符串进行啦哈希操作然后进行一个替换作为密码。由于运行的时候用的hash函数是随机的，所以4个都试一下。

```
<span style="color: rgb(0, 0, 0);">#!/usr/bin/env python<br>import hashlib<br>dic = "AabRcQPXdYVe
```

最后尝试发现：

序列号：skxxRWi23

哈希值：521c0892b9dc0a7026fbe9664e6a339e7fee9492605733ea09968fbd83f18dfff91fe87d9d620fa4d3dd3010b47495dc

解锁码：545048447596050

这一是正确的。

0x02 寻找密码

这题其实是给apk加了个壳，程序里把真实的apk经过了加密(异或255)拼到了apk的dex文件后面，所以我直接把dex文件提取出来，整个文件异或255，然后从第一个PK开始提取出原始APK。然后原始的APK扔到jeb里就很容易看出源代码了。

算法很简单：

```
<span style="color: rgb(0, 0, 0);">#!/usr/bin/env python<br>import base64<br>import hashlib<br>us
```

0x03 顺藤摸瓜

apk用了zip伪加密，首先用010editor打开，将所有 0x50 0x4B 0x01 0x02（PK..）的位置后的第五个字节改为0，即可成功安装或解压。可参考吾爱破解的文章帖子http://www.52pojie.cn/thread-287242-1-1.html。

反编译apk后发现会调用Native函数check来验证密码，直接用ida将libdemo.so打开。如下

```
env = env_;
v4 = a3;
v5 = ((int (*)(void))(*env_)->FindClass)();
v6 = (void *)((int (__fastcall *)(JNIEnv *, const char *))(*env)->NewStringUTF)(env, "GB2312");
v7 = ((int (__fastcall *)(JNIEnv *, int, const char *, const char *))(*env)->GetMethodID)(
       env,
       v5,
       "getBytes",
       "(Ljava/lang/String;)[B");
v8 = ((int (__fastcall *)(JNIEnv *, jstring, int, void *))(*env)->CallObjectMethod)(env, v4, v7, v6);
array_len = ((int (__fastcall *)(JNIEnv *, int))(*env)->GetArrayLength)(env, v8);
src = (void *)((int (__fastcall *)(JNIEnv *, int, _DWORD))(*env)->GetByteArrayElements)(env, v8, 0);
if ( array_len <= 0 )
{
  input = 0;
}
else
{
  input = (char *)j_j_malloc(array_len + 1);
  j_j_memcpy(input, src, array_len);
  input[array_len] = 0;
}
((void (__fastcall *)(JNIEnv *, int, void *, _DWORD))(*env)->ReleaseByteArrayElements)(env, v8, src, 0);
j_j_memset(s1, 0, 0xC8u);
j_j_memset(s2, 0, 0xC8u);
j_j_memset(s, 0, 0x100u);
j_j_memcpy(dest, &unk_2454, 0x38u);
str_len = j_j_strlen(input);
for ( i = 0; i < str_len; ++i )
  s1[i] = input[i] + 97 - dest[4 * i];
s1[str_len & (~str_len >> 31)] = 0;
n1(s1, "nbrcdpassword", (int)s2);
n2(s2, s);
result = (unsigned int)j_j_strcmp(s, "74058473948333034392948223334") <= 0;
if ( v19 != _stack_chk_guard )
  j_j___stack_chk_fail(result);
return result;
}
```


三段比较简单的加密，直接用ipython解了

```
In [1]: s="74058473948333034392948223334"
```

```
In [14]: for i in range(len(s)/2):
    ...:     result+=chr(int(s[2*i+1]+s[2*
    ...:     i])+72)
    ...:
    ...:

In [15]: result
Out[15]: 'wzxmynifjeydhs'

In [16]: s2=result

In [17]: len(s2)
Out[17]: 14

In [18]: len(key)
Out[18]: 13

In [19]: len(s)
Out[19]: 28

In [20]: key
Out[20]: 'nbrcdpassword'

In [21]: key+='n'
```



```
In [24]: result=''

In [25]: for i in range(len(s2)):
    ...:     result+=chr(ord(s2[i])-ord(key[i])+97)
    ...:

In [26]: result
Out[26]: 'jygkv_iTXOkSef'

In [27]: key2='?Ml[T[l[TF8F?'

In [28]: len(key2)
Out[28]: 13

In [29]: key2+='\x1c'

In [30]: len(key2)
Out[30]: 14

In [31]: key2
Out[31]: '?Ml[T[l[TF8F?\x1c'

In [32]: s3=result
```



```
In [33]: len(s3)
Out[33]: 14

In [34]: result=''

In [35]: for i in range(len(s3)):
    ...:     result+=chr(ord(s3[i])+ord(key2[i])-
    ...:     97)
    ...:
    ...:

In [36]: result
Out[36]: 'HereiYtNK
```


把上面的result的值输入手机中，即可显示“碰头地点：太白南路2号”

0x04 神奇的zip

这个题首先也是一个伪加密，修复后即可正常安装和解压。

首先apk一启动就会调用libgeneratekey.so中的isExit函数，如果该函数返回0那么apk就退出，而ida查看isExit函数的唯一作用就是返回0。因此可以使用apktool反编译apk，将SplashActivity.smali文件中第52行的if-eqz改为if-nez，即可绕过这个检测。

随后会启动MainActivity，这个类的唯一操作就是将输入的字符串传入native层的函数encodePassword中，并且显示出encodePassword返回的字符串。因此我们使用ida查看encodePassword函数。主要逻辑如下

```
int __fastcall Java_com_example_testndk4_MainActivity_encodePassword(int a1)
{
  int v1; // r5@1
  const char *v2; // r7@1
  char *src; // ST04_4@1
  char *v4; // ST04_4@1
  char *v5; // r0@2
  const char *v6; // r1@2
  int result; // r0@4
  char v8; // [sp+8h] [bp-58h]@1
  char v9; // [sp+14h] [bp-4Ch]@1
  char s; // [sp+28h] [bp-38h]@1
  int v11; // [sp+44h] [bp-1Ch]@4

  v1 = a1;
  v2 = (const char *)Jstring2CStr();
  j_j_memcpy(&v9, "thinkingInAndroid", 0x12u);
  src = (char *)encodePS(&v9);
  j_j_memset(&s, 0, 0x1Au);
  j_j_strcpy(&s, src);
  v4 = (char *)encodePS(&s);
  j_j_memset(&v8, 0, 0xAu);
  if ( j_j_strcmp(v2, v4) )
  {
    v5 = &v8;
    v6 = "Sorry!";
  }
  else
  {
    v5 = &v8;
    v6 = "Sucess!";
  }
  j_j_strcpy(v5, v6);
  result = (*(int (__fastcall **)(int, char *))(*(_DWORD *)v1 + 668))(v1, &v8);
  if ( v11 != _stack_chk_guard )
    j_j___stack_chk_fail(result);
  return result;
}
```


可以看出，该函数会将输入的字符串与一串经过极其复杂变形的字符串进行比较，这里我们可以不去深入研究变形的过程，因为该函数没有将输入的字符串做任何变化，而是去直接比较的，因此我们可以使用调试或者hook的方法直接将变形完的字符串打印出来。这里我用frida直接hook了encodePS函数，打印出它的返回值即可，会打印两遍，取后一次。

hook代码

```
1          <span style="color: rgb(0, 0, 0);">let F =   Module.findExportByName('libgeneratekey.so'
```

输出：

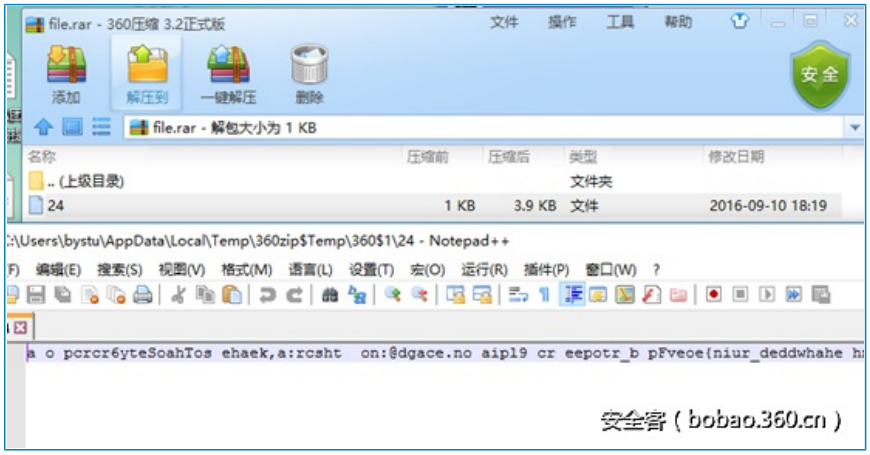
```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
00000000  6c 78 69 65 6e 69 65 74 49 65 41 65 68 66 79 69  lxienietIeAehfyi
00000010  68 00 00 00 00 00 00 00 00 00 00 00 da 11 dd 96  h...............
00000020  a8 12 7e 42 48 01 92 6d c8 63 72 41 04 00 00 00  ..~BH..m.crA....
00000030  04 0d 60 41 d0 2b 56 41 f8 0c 60 41 01 00 00 00  ..`A.+VA..`A....
```


上图以l开头的字符串即为flag。

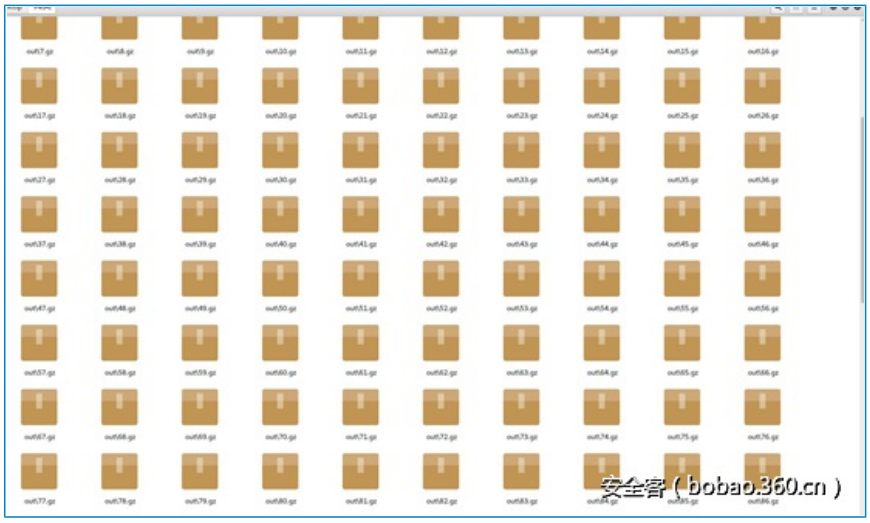
Misc

0x01 Try Everything

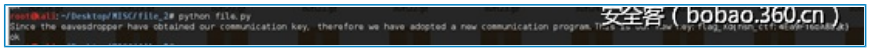
这题并不难，直接解压后发现是乱的



然后扔binwalk，得到文件名和偏移量，脚本分解出文件



然后按照文件名排序解出并且合并文本



0x02 挣脱牢笼

Python沙盒逃逸题。

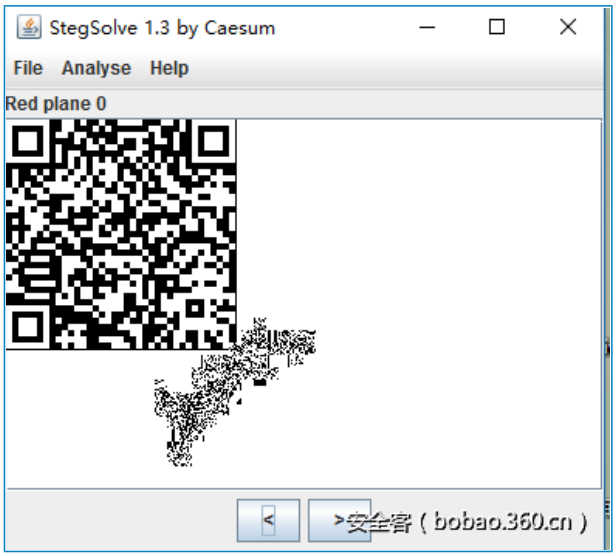
一开始设想用[].__class__.__base__.__subclasses__()[40]来使用file读文件。后来发现他命令限制长度50，非常蛋疼。后来才发现可以直接设定__builtins__变量来把指令分成多条进行，就不会受这个限制了。最后的exp如下：

```
1       <span style="color: rgb(0, 0, 0);">__builtins__['ww']=().__class__.__base__<br>__builtins__['w']=
```

Forensics

0x01蒲公英的约定

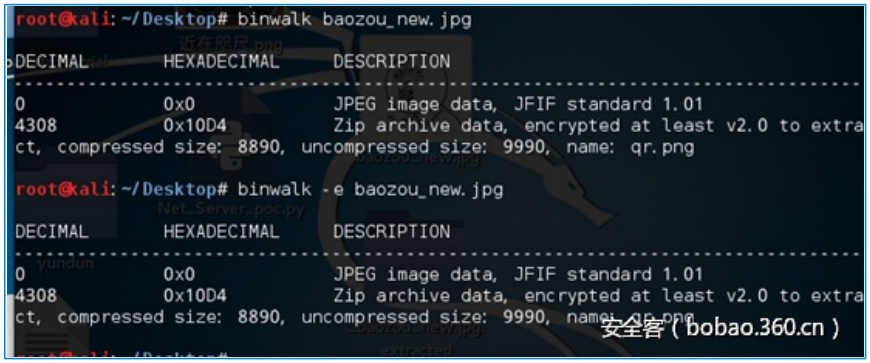
Stegsolve打开，里面有张二维码，反色下就好了



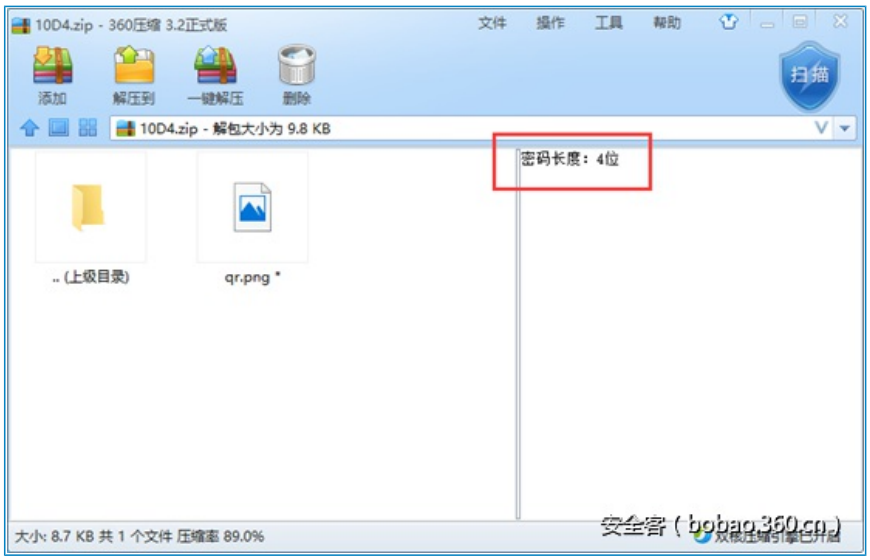
扫码后base32

0x02什么鬼

Binwalk可以看到一个zip



密码长度4位，直接爆破，密码：19bZ

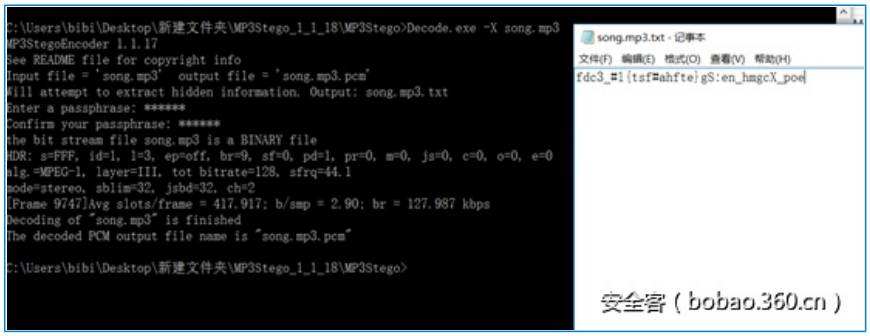


解开后将右边的块补上即可

0x03客官，听点小曲儿？

那个http的头里发现了：

直接用mp3stego decode掉得到：



可见字符，顺序乱了，考虑栅栏，长度为6的试了不行，后面应该长度有些许变化，手动切割，得到flag：

```
<span style=
"color: rgb(0, 0, 0);"
>a=
"fdc3_# l{tsf# ahfte} gS:en _hmgc X_poe"
<br>   b=a.split(
" "
)<br>   s=""<br>
for
i
in
range(6):<br>
for
j
in
b:<br>
if
i < len(j):<br>                s+=j
[i]
<br>   print s<br></span>
```

登录 | 注册 | 匿名评论

参与讨论，请先

匿名

发布

用户评论

360U2753148133 2016-09-12 18:35:07

有视频吗

回复 | 点赞(0)

helen的小弟 2016-09-12 16:57:03

233那题，记事本保存成Unicode格式的就可以了

回复 | 点赞(0)