buuctf刷题记录（2）

原创

[7earn](https://blog.csdn.net/weixin_53409153) 于 2021-06-06 21:16:43 发布 123 收藏

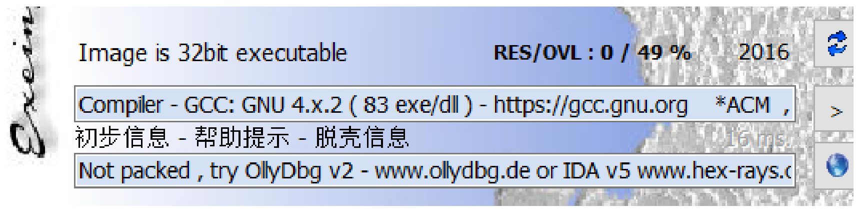
分类专栏： 笔记 学习

笔记 同时被 2 个专栏收录
10 篇文章 0 订阅
订阅专栏

学习
10 篇文章 0 订阅
订阅专栏

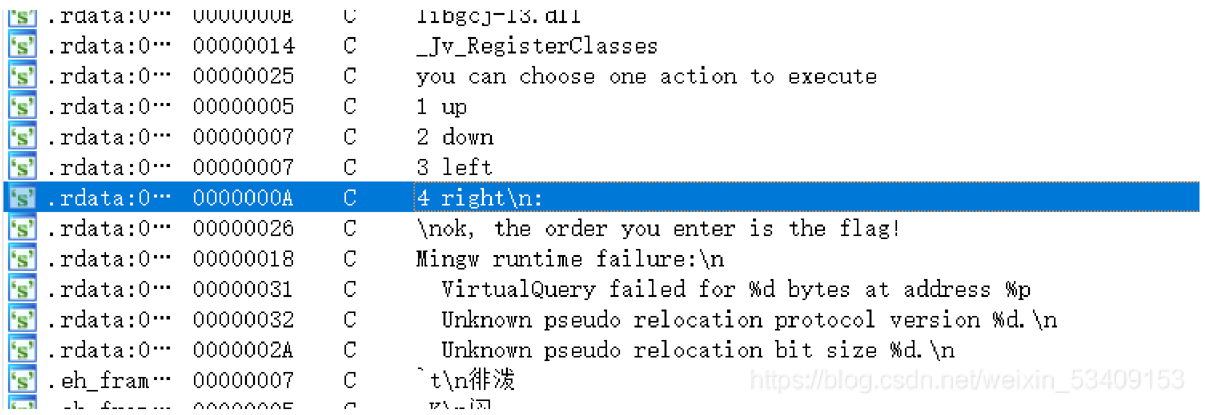
不一样的flag

对于这种题，一般最开始就是需要查壳吧：



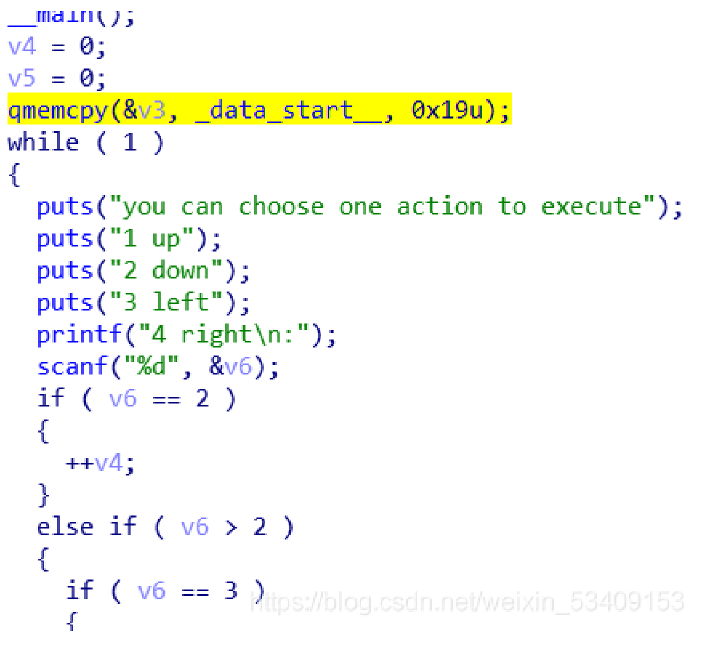
```
Image is 32bit executable          RES/OVL : 0 / 49 %     2016
Compiler - GCC: GNU 4.x.2 ( 83 exe/dll ) - https://gcc.gnu.org    *ACM ,
初步信息 - 帮助提示 - 脱壳信息                          16 ms.
Not packed , try OllyDbg v2 - www.ollydbg.de or IDA v5 www.hex-rays.c
```

查壳后发现无壳32位，然后就直接拖入IDA，shift+f12查看：



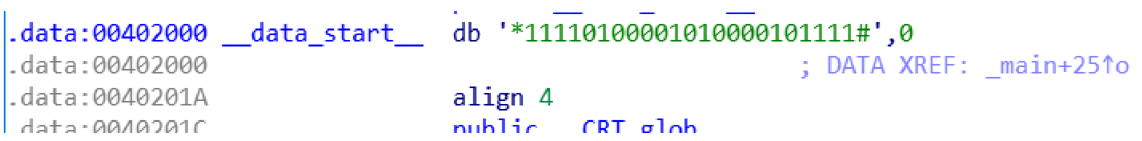
```
.rdata:0…  0000000E   C   libgcj-13.dll
.rdata:0…  00000014   C   _Jv_RegisterClasses
.rdata:0…  00000025   C   you can choose one action to execute
.rdata:0…  00000005   C   1 up
.rdata:0…  00000007   C   2 down
.rdata:0…  00000007   C   3 left
.rdata:0…  0000000A   C   4 right\n:
.rdata:0…  00000026   C   \nok, the order you enter is the flag!
.rdata:0…  00000018   C   Mingw runtime failure:\n
.rdata:0…  00000031   C     VirtualQuery failed for %d bytes at address %p
.rdata:0…  00000032   C     Unknown pseudo relocation protocol version %d.\n
.rdata:0…  0000002A   C     Unknown pseudo relocation bit size %d.\n
.eh_fram…  00000007   C   `t\n徘泼
```

跟进，然后f5查看伪代码：



```
  __main();
  v4 = 0;
  v5 = 0;
  qmemcpy(&v3, _data_start__, 0x19u);
  while ( 1 )
  {
    puts("you can choose one action to execute");
    puts("1 up");
    puts("2 down");
    puts("3 left");
    printf("4 right\n:");
    scanf("%d", &v6);
    if ( v6 == 2 )
    {
      ++v4;
    }
    else if ( v6 > 2 )
    {
      if ( v6 == 3 )
      {
```

可以看见，这是一个while循环，然后在while循环上面有一行代码，很重要`qmemcpy(&v3, _data_start__, 0x19u);`跟进后可以看见：



```
.data:00402000 __data_start__  db '*11110100001010000101111#',0
.data:00402000                                          ; DATA XREF: _main+25↑o
.data:0040201A                  align 4
.data:0040201C                  public __CRT_glob
```

然后再回头看伪代码：

```
while ( 1 )
{
  puts("you can choose one action to execute");
  puts("1 up");
  puts("2 down");
  puts("3 left");
  printf("4 right\n:");
  scanf("%d", &v6);
```

```
if ( v8[5 * v4 - 41 + v5] == 49 )
    exit(1);
  if ( v8[5 * v4 - 41 + v5] == 35 )
```

可以从这几行代码看出，这是一个5*5的一个矩阵迷宫游戏（刚好一共就是25个字符）：`*11110100001010000101111#`
然后转化为矩阵就是：

```
*1111
01000
01010
00010
1111#
```

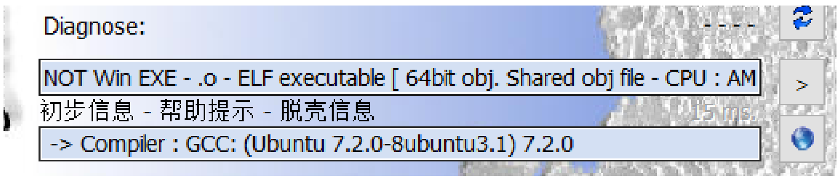
然后，再结合为了循环：

```
    puts("you can choose one action to execute");
    puts("1 up");
    puts("2 down");
    puts("3 left");
    printf("4 right\n:");
    scanf("%d", &v6)
```

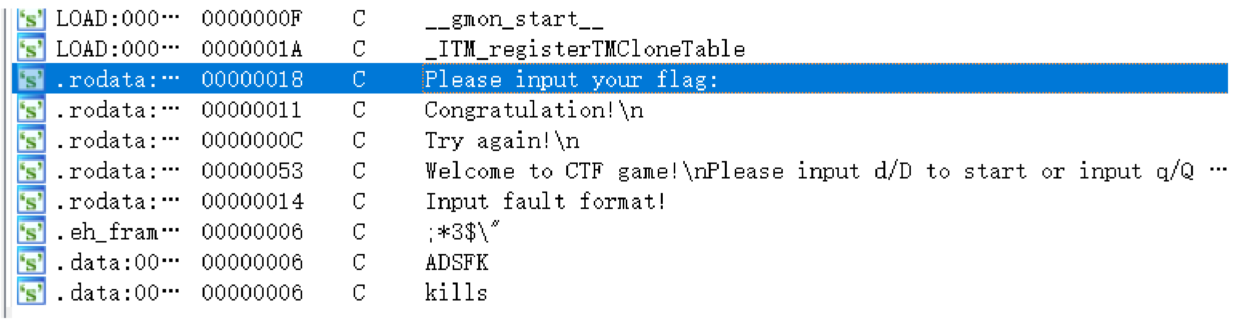
走零不走一，1,2，3,4就是上下左右，从*开始到#停止，那么路线就是：`222441144222`题目上也说flag就是字符串，所以得到：`flag{222441144222}`

SimpleRev

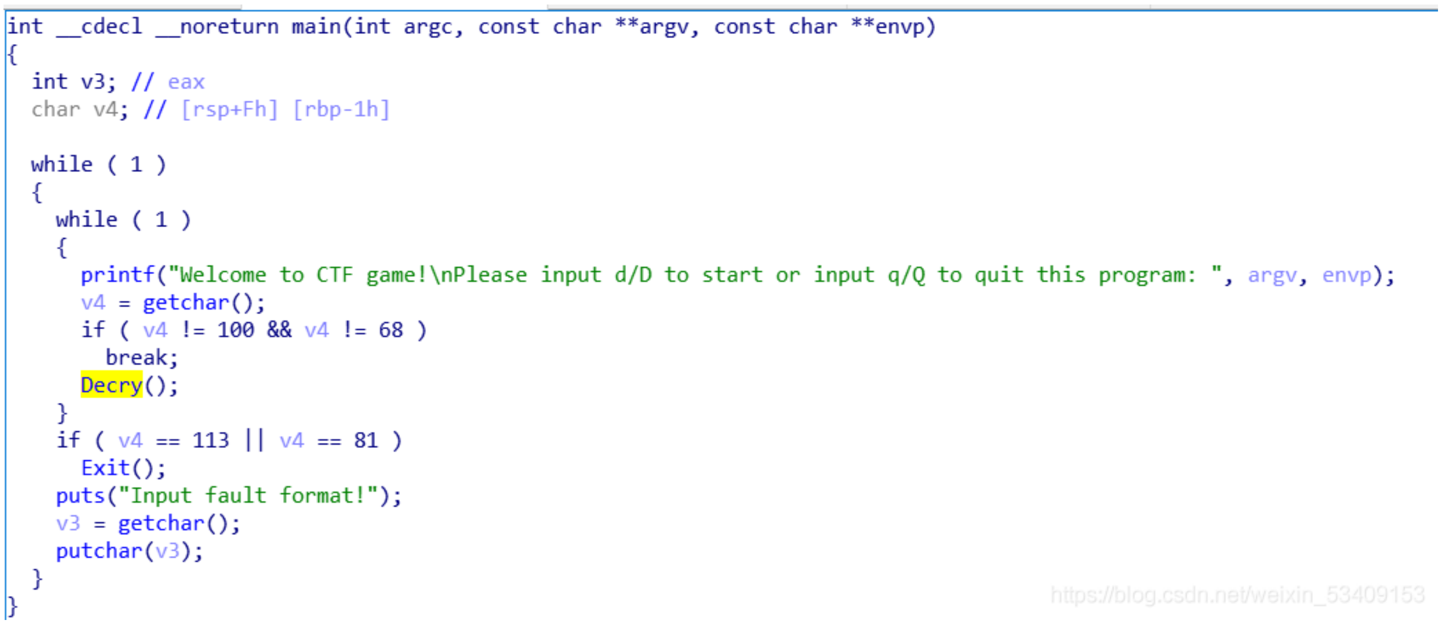
我开始的时候也不知道该从哪上手，后来就觉得无所谓了，反正一直就是查壳，就发现是64位的：


Diagnose:
NOT Win EXE - .o - ELF executable [64bit obj. Shared obj file - CPU : AM
初步信息 - 帮助提示 - 脱壳信息
-> Compiler : GCC: (Ubuntu 7.2.0-8ubuntu3.1) 7.2.0

就直接IDA呗，没什么花里胡哨的操作shift+f12:


```
LOAD:000…    0000000F    C    __gmon_start__
LOAD:000…    0000001A    C    _ITM_registerTMCloneTable
.rodata:…    00000018    C    Please input your flag:
.rodata:…    00000011    C    Congratulation!\n
.rodata:…    0000000C    C    Try again!\n
.rodata:…    00000053    C    Welcome to CTF game!\nPlease input d/D to start or input q/Q …
.rodata:…    00000014    C    Input fault format!
.eh_fram…    00000006    C    ;*3$\"
.data:00…    00000006    C    ADSFK
.data:00…    00000006    C    kills
```

然后就跟进，在f5查看伪代码：


```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  char v4; // [rsp+Fh] [rbp-1h]

  while ( 1 )
  {
    while ( 1 )
    {
      printf("Welcome to CTF game!\nPlease input d/D to start or input q/Q to quit this program: ", argv, envp);
      v4 = getchar();
      if ( v4 != 100 && v4 != 68 )
        break;
      Decry();
    }
    if ( v4 == 113 || v4 == 81 )
      Exit();
    puts("Input fault format!");
    v3 = getchar();
    putchar(v3);
  }
}
```

查看后，通过分析，就发现Decry（）函数是最重要的，但是要是想进入Decry函数，就需要输入的d/D，然后进入到函数中，查看：

```
unsigned __int64 Decry()
{
```

```
char v1; // [rsp+Fh] [rbp-51h]
int v2; // [rsp+10h] [rbp-50h]
int v3; // [rsp+14h] [rbp-4Ch]
int i; // [rsp+18h] [rbp-48h]
int v5; // [rsp+1Ch] [rbp-44h]
char src[8]; // [rsp+20h] [rbp-40h]
__int64 v7; // [rsp+28h] [rbp-38h]
int v8; // [rsp+30h] [rbp-30h]
__int64 v9; // [rsp+40h] [rbp-20h]
__int64 v10; // [rsp+48h] [rbp-18h]
int v11; // [rsp+50h] [rbp-10h]
unsigned __int64 v12; // [rsp+58h] [rbp-8h]

v12 = __readfsqword(0x28u);
*(_QWORD *)src = 357761762382LL;
v7 = 0LL;
v8 = 0;
v9 = 512969957736LL;
v10 = 0LL;
v11 = 0;
text = (char *)join(key3, &v9);
strcpy(key, key1);
strcat(key, src);
v2 = 0;
v3 = 0;
getchar();
v5 = strlen(key);
for ( i = 0; i < v5; ++i )
{
  if ( key[v3 % v5] > 64 && key[v3 % v5] <= 90 )
    key[i] = key[v3 % v5] + 32;//变大写为小写
  ++v3;
}
printf("Please input your flag:", src);
while ( 1 )
{
  v1 = getchar();
  if ( v1 == 10 )
    break;
  if ( v1 == 32 )
  {
    ++v2;
  }
  else
  {
    if ( v1 <= 96 || v1 > 122 )
    {
      if ( v1 > 64 && v1 <= 90 )
        str2[v2] = (v1 - 39 - key[v3++ % v5] + 97) % 26 + 97;
    }
    else
    {
      str2[v2] = (v1 - 39 - key[v3++ % v5] + 97) % 26 + 97;
    }
    if ( !(v3 % v5) )
      putchar(32);
    ++v2;
  }
}
if ( !strcmp(text, str2) )
```

```
  if ( !strcmp(text, str2) )
    puts("Congratulation!\n");
  else
    puts("Try again!\n");
  return __readfsqword(0x28u) ^ v12;
}
```

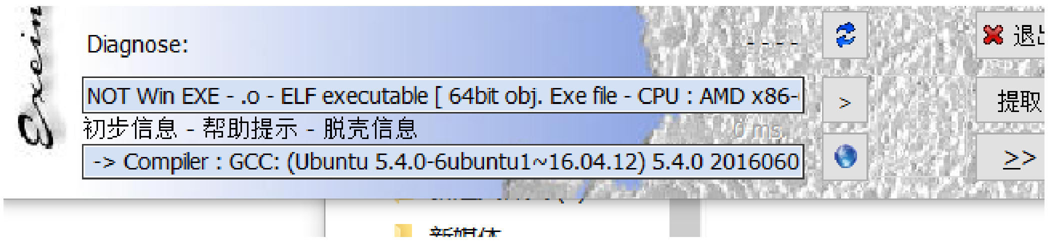
代码大概意思就是，通过复制key1和str到key中进行操作，在输入flag到str2
中，通过代码后在于text进行比较，要是比较真确就输出*Congratulation!*，佛则就输出*Try again!*在上脚本（C语言脚本）就是：

```
#include<stdio.h>
int main()
{
 char key[] = "adsfkndcls";
 char text[] = "killshadow";
 int i;
 int v3=10;
 for (int i = 0; i < 10; i++)
 {
  for (int j = 0; j < 128; j++)
  {
   if (j < 'A' || j > 'z' || j > 'Z' && j < 'a')
   {
    continue;
   }
   if ((j - 39 - key[v3 % 10] + 97) % 26 + 97 == text[i])
   {
    printf("%c",j);
    v3++;
    break;
   }
  }
 }
}
```

的到最后的flag：`KLDQCUDFZO`在包上flag{}就行了。

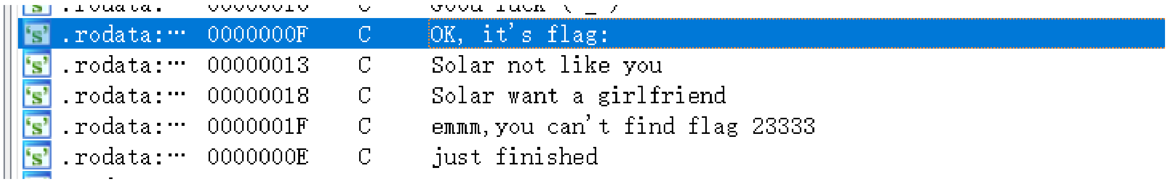
[GXYCTF2019]luck_guy

就自己查看位数（64位）：



```
Diagnose:
NOT Win EXE - .o - ELF executable [ 64bit obj. Exe file - CPU : AMD x86-
初步信息 - 帮助提示 - 脱壳信息
 -> Compiler : GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 2016060
```

拖入IDA，shift+f12查看：

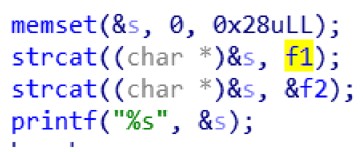


```
.rodata:…  0000000F   C    OK, it's flag:
.rodata:…  00000013   C    Solar not like you
.rodata:…  00000018   C    Solar want a girlfriend
.rodata:…  0000001F   C    emmm,you can't find flag 23333
.rodata:…  0000000E   C    just finished
```

跟进，f5查看伪代码：

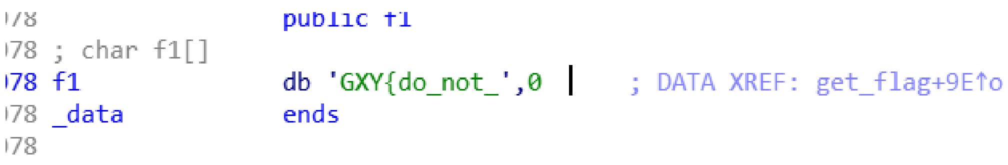
```c
unsigned __int64 get_flag()
{
  unsigned int v0; // eax
  char v1; // al
  signed int i; // [rsp+4h] [rbp-3Ch]
  signed int j; // [rsp+8h] [rbp-38h]
  __int64 s; // [rsp+10h] [rbp-30h]
  char v6; // [rsp+18h] [rbp-28h]
  unsigned __int64 v7; // [rsp+38h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  v0 = time(0LL);
  srand(v0);
  for ( i = 0; i <= 4; ++i )
  {
    switch ( rand() % 200 )
    {
      case 1:
        puts("OK, it's flag:");
        memset(&s, 0, 0x28uLL);
        strcat((char *)&s, f1);
        strcat((char *)&s, &f2);
        printf("%s", &s);
        break;
      case 2:
        printf("Solar not like you");
        break;
      case 3:
        printf("Solar want a girlfriend");
        break;
      case 4:
        v6 = 0;
        s = 9180147350284624745LL;
        strcat(&f2, (const char *)&s);
        break;
      case 5:
        for ( j = 0; j <= 7; ++j )
        {
          if ( j % 2 == 1 )
            v1 = *(&f2 + j) - 2;
          else
            v1 = *(&f2 + j) - 1;
          *(&f2 + j) = v1;
        }
        break;
      default:
        puts("emmm,you can't find flag 23333");
        break;
    }
  }
  return __readfsqword(0x28u) ^ v7;
}
```

查看代码可知，flag是由分和否拼接来的：



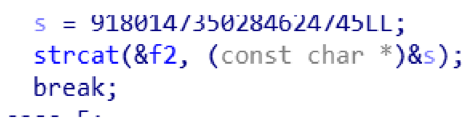
```
memset(&s, 0, 0x28uLL);
strcat((char *)&s, f1);
strcat((char *)&s, &f2);
printf("%s", &s);
```

我就跟进f1，看见：



```
178                  public f1
178 ; char f1[]
178 f1               db 'GXY{do_not_',0      ; DATA XREF: get_flag+9E↑o
178 _data            ends
178
```

f2是：



```
s = 9180147350284624745LL;
strcat(&f2, (const char *)&s);
break;
```

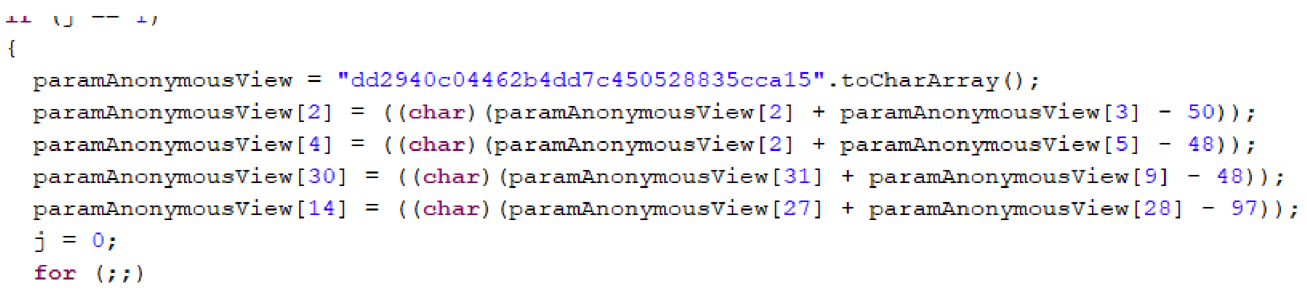
这题就离谱，取了0~199的随机数，然后case1 case4 case5才是有效路径，还要找准顺序，怪不得题目为luck_guy了，随缘吧，我们还是得才猜测他的顺序的，不出意外case1肯定是最后一位，然后就是猜测case4是第一位，因为f1和f2应该是要整合到一起然后再进行case5的操作的，然后，脚本如下：

```
flag="GXY{do_not_"
f2=[0x7F,0x66,0x6F,0x60,0x67,0x75,0x63,0x69][::-1] #小端序的问题，所以要逆序一下

for j in range(8):
    if j%2==1 :
        s=chr(f2[j]-2)
    else:
        s=chr(f2[j]-1)

    flag+=s

print (flag)
```

运行得到：`GXY{do_not_hate_me}`

简单注册器

这道题是我用的是安卓逆向做的，跳转到java就是这样的：



```
if (j == 1)
{
  paramAnonymousView = "dd2940c04462b4dd7c450528835cca15".toCharArray();
  paramAnonymousView[2] = ((char)(paramAnonymousView[2] + paramAnonymousView[3] - 50));
  paramAnonymousView[4] = ((char)(paramAnonymousView[2] + paramAnonymousView[5] - 48));
  paramAnonymousView[30] = ((char)(paramAnonymousView[31] + paramAnonymousView[9] - 48));
  paramAnonymousView[14] = ((char)(paramAnonymousView[27] + paramAnonymousView[28] - 97));
  j = 0;
  for (;;)
```

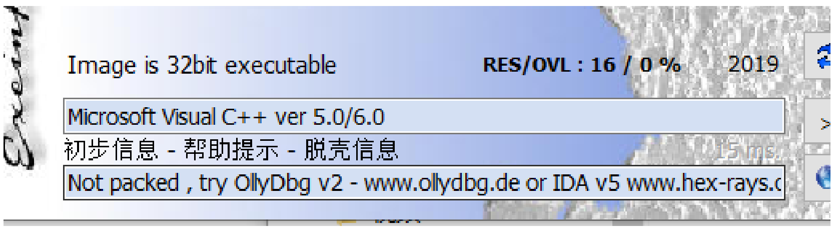
然后我就是直接上脚本了：

```python
str=['d','d','2','9','4','0','c','0','4','4','6','2','b','4','d','d','7','c','4','5','0','5','2','8','8','3','5'
,'c','c','a','1','5']

str[2]=chr(ord(str[2])+ord(str[3])-50)
str[4]=chr( ord(str[2])+ord(str[5])-0x30 )
str[30]=chr( ord(str[0x1f])+ord(str[9])-0x30)
str[14]=chr( ord(str[27])+ord(str[28])-97 )

for i in range(16):
    x=str[0x1f-i]
    str[0x1f-i]=str[i]
    str[i]=x

for i in str:
    print (i,end="")
```

运行结果如下：`59acc538825054c7de4b26440c0999dd`
在包上`flag{59acc538825054c7de4b26440c0999dd}`就行了

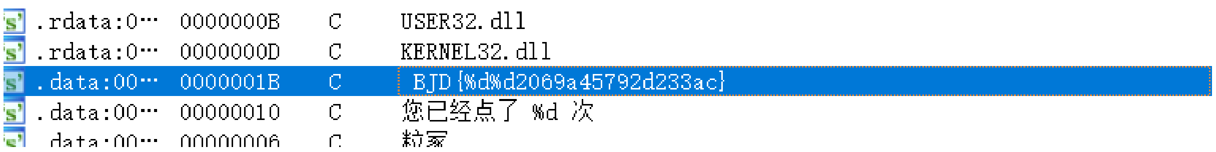
[BJDCTF2020]JustRE

首先，当然是查壳：



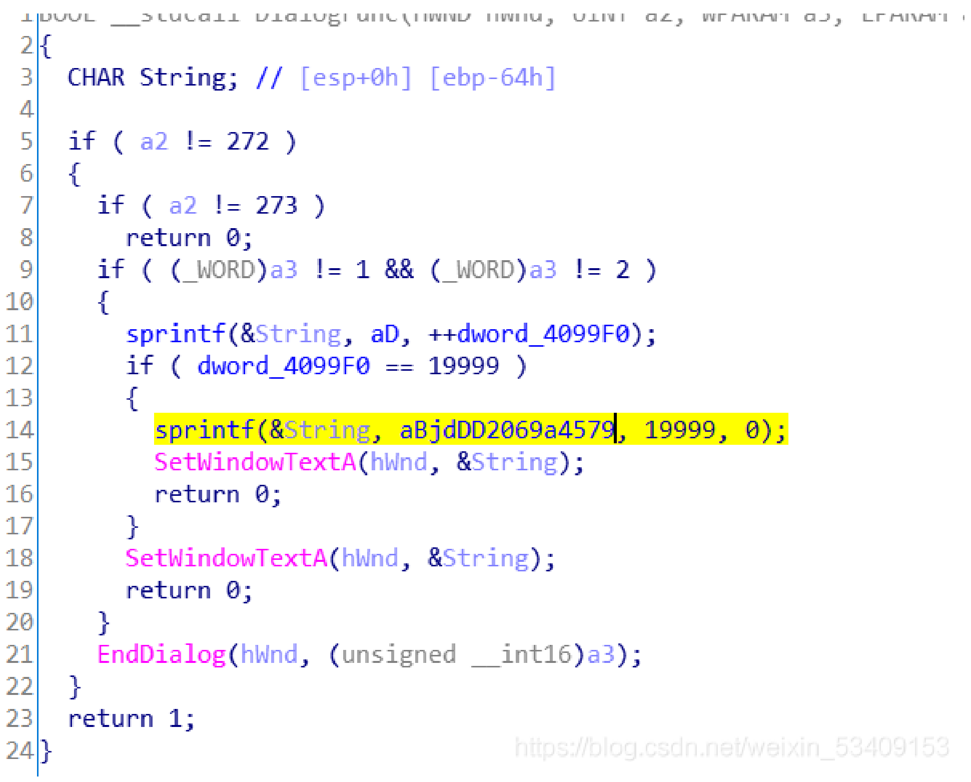
```
Image is 32bit executable                RES/OVL : 16 / 0 %    2019
Microsoft Visual C++ ver 5.0/6.0
初步信息 - 帮助提示 - 脱壳信息
Not packed , try OllyDbg v2 - www.ollydbg.de or IDA v5 www.hex-rays.c
```

32位，拖入IDA，shift+f2查看：



```
.rdata:0…  0000000B   C    USER32.dll
.rdata:0…  0000000D   C    KERNEL32.dll
.data:00…  0000001B   C    BJD{%d%d2069a45792d233ac}
.data:00…  00000010   C    您已经点了 %d 次
.data:00…  00000006   C    粒冢
```

其实这一步就有点像了，但是不可能这么简单，在就直接跟进，f5查看伪代码：



```
BOOL __stdcall DialogFunc(HWND hWnd, UINT a2, WPARAM a3, LPARAM
{
  CHAR String; // [esp+0h] [ebp-64h]

  if ( a2 != 272 )
  {
    if ( a2 != 273 )
      return 0;
    if ( (_WORD)a3 != 1 && (_WORD)a3 != 2 )
    {
      sprintf(&String, aD, ++dword_4099F0);
      if ( dword_4099F0 == 19999 )
      {
        sprintf(&String, aBjdDD2069a4579, 19999, 0);
        SetWindowTextA(hWnd, &String);
        return 0;
      }
      SetWindowTextA(hWnd, &String);
      return 0;
    }
    EndDialog(hWnd, (unsigned __int16)a3);
  }
  return 1;
}
```

看见第14行输出了aBjdDD2069a4579, 19999, 0，而最开始有：BJD{%d%d2069a45792d233ac}，%d%d应该是就是19999, 0整合一下，得到`BJD{1999902069a45792d233ac}`

[GWCTF 2019]pyre

下载附件是pyc文件，用pyc在线解密

```python
#!/usr/bin/env python
# visit http://tool.lu/pyc/ for more information
print 'Welcome to Re World!'
print 'Your input1 is your flag~'
l = len(input1)
for i in range(l):
    num = ((input1[i] + i) % 128 + 128) % 128
    code += num

for i in range(l - 1):
    code[i] = code[i] ^ code[i + 1]

print code
code = [
    '\x1f',
    '\x12',
    '\x1d',
    '(',
    '0',
    '4',
    '\x01',
    '\x06',
    '\x14',
    '4',
    ',',
    '\x1b',
    'U',
    '?',
    'o',
    '6',
    '*',
    ':',
    '\x01',
    'D',
    ';',
    '%',
    '\x13']
```

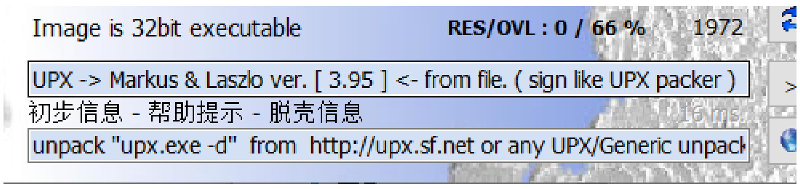
编写脚本如下：

```python
code = ['\x1f','\x12','\x1d','(','0','4','\x01','\x06','\x14','4',
        ',','\x1b','U','?','o','6','*',':','\x01','D',';','%','\x13']

for i in range(len(code)-2,-1,-1):
    code[i]=chr(ord(code[i])^ord(code[i+1]))

for i in range(len(code)):
    print(chr((ord(code[i])-i)%128),end="")

```

运行后得到：`GWHT{Just_Re_1s_Ha66y!}`

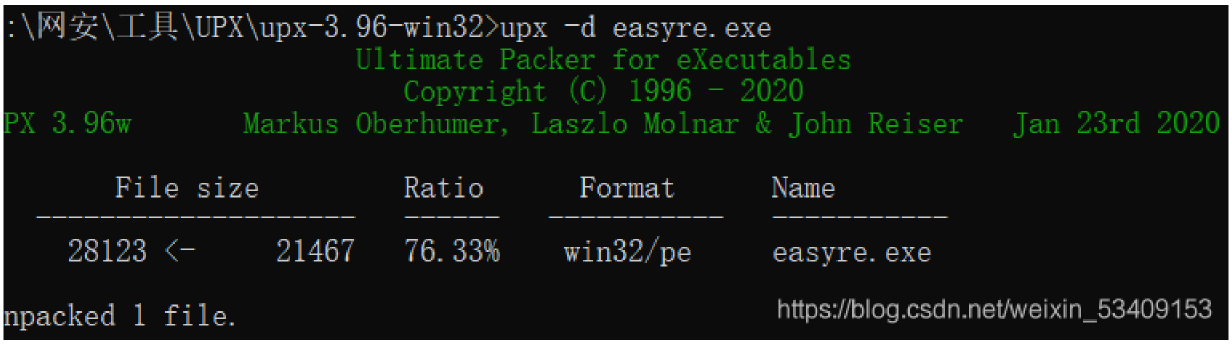
[ACTF新生赛2020]easyre

查壳，发现是32位但是有upx壳：



```
Image is 32bit executable          RES/OVL : 0 / 66 %     1972
UPX -> Markus & Laszlo ver. [ 3.95 ] <- from file. ( sign like UPX packer )
初步信息 - 帮助提示 - 脱壳信息
unpack "upx.exe -d"  from  http://upx.sf.net or any UPX/Generic unpack
```

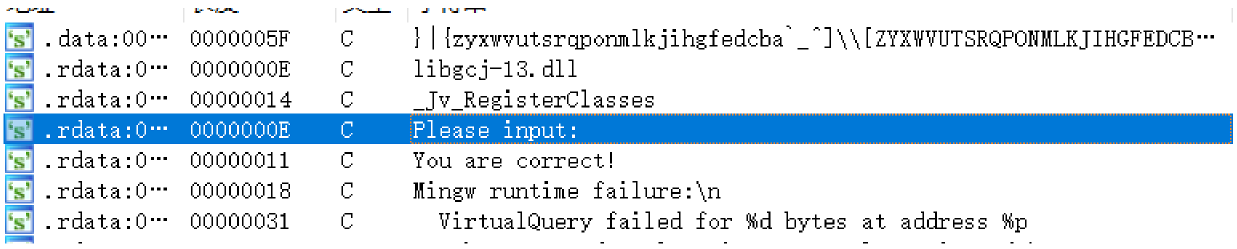
脱壳：



```
:\网安\工具\UPX\upx-3.96-win32>upx -d easyre.exe
                      Ultimate Packer for eXecutables
                         Copyright (C) 1996 - 2020
PX 3.96w        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

       File size         Ratio      Format      Name
  --------------------   ------   -----------   -----------
     28123 <-     21467   76.33%    win32/pe     easyre.exe

npacked 1 file.
```

https://blog.csdn.net/weixin_53409153

然后就直接拖入IDA，shift+f12：

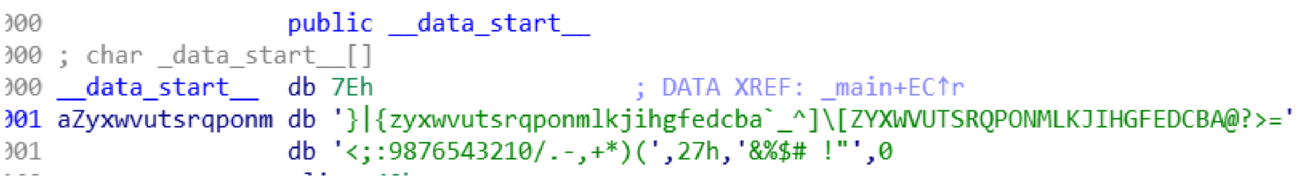


```
.data:00…   0000005F   C   }|{zyxwvutsrqponmlkjihgfedcba`_^]\\[ZYXWVUTSRQPONMLKJIHGFEDCB…
.rdata:0…   0000000E   C   libgcj-13.dll
.rdata:0…   00000014   C   _Jv_RegisterClasses
.rdata:0…   0000000E   C   Please input:
.rdata:0…   00000011   C   You are correct!
.rdata:0…   00000018   C   Mingw runtime failure:\n
.rdata:0…   00000031   C     VirtualQuery failed for %d bytes at address %p
```

在跟进f5：

```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4; // [esp+12h] [ebp-2Eh]
  char v5; // [esp+13h] [ebp-2Dh]
  char v6; // [esp+14h] [ebp-2Ch]
  char v7; // [esp+15h] [ebp-2Bh]
  char v8; // [esp+16h] [ebp-2Ah]
  char v9; // [esp+17h] [ebp-29h]
  char v10; // [esp+18h] [ebp-28h]
  char v11; // [esp+19h] [ebp-27h]
  char v12; // [esp+1Ah] [ebp-26h]
  char v13; // [esp+1Bh] [ebp-25h]
  char v14; // [esp+1Ch] [ebp-24h]
  char v15; // [esp+1Dh] [ebp-23h]
  int v16; // [esp+1Eh] [ebp-22h]
  int v17; // [esp+22h] [ebp-1Eh]
  int v18; // [esp+26h] [ebp-1Ah]
  char v19; // [esp+2Ah] [ebp-16h]
  char v20; // [esp+2Bh] [ebp-15h]
  char v21; // [esp+2Ch] [ebp-14h]
  char v22; // [esp+2Dh] [ebp-13h]
  char v23; // [esp+2Eh] [ebp-12h]
  int v24; // [esp+2Fh] [ebp-11h]
  int v25; // [esp+33h] [ebp-Dh]
  int v26; // [esp+37h] [ebp-9h]
  char v27; // [esp+3Bh] [ebp-5h]
  int i; // [esp+3Ch] [ebp-4h]

  __main();
  v4 = 42;
  v5 = 70;
  v6 = 39;
  v7 = 34;
  v8 = 78;
  v9 = 44;
  v10 = 34;
  v11 = 40;
  v12 = 73;
  v13 = 63;
  v14 = 43;
  v15 = 64;
  printf("Please input:");
  scanf("%s", &v19);
  if ( v19 != 65 || v20 != 67 || v21 != 84 || v22 != 70 || v23 != 123 || v27 != 125 )
    return 0;
  v16 = v24;
  v17 = v25;
  v18 = v26;
  for ( i = 0; i <= 11; ++i )
  {
    if ( *(&v4 + i) != _data_start__[*((char *)&v16 + i) - 1] )
      return 0;
  }
  printf("You are correct!");
  return 0;
}
```

跟进_data_start__函数：



```
00                    public __data_start__
00 ; char _data_start__[]
00 __data_start__  db 7Eh                   ; DATA XREF: _main+EC↑r
01 aZyxwvutsrqponm db '}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>='
01                 db '<;:9876543210/.-,+*)(',27h,'&%$# !"',0
```

然后就直接解密：

```python
s = [42,70,39,34,78,44,34,40,73,63,43,64]
key = '~}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.-,+*)('+chr(0x27)+'&%$# !"'
flag = ''
for i in range(12):
   x = key.find(chr(s[i]))+1
   flag += chr(x)
print(flag)
```

解密得到：`flag{U9X_1S_W6@T?}`