攻防世界 Reverse 新手练习区 1-12 全详解

原创

思源湖的鱼 于 2020-11-28 12:40:43 发布 1173 收藏 20

分类专栏： ctf 文章标签： ctf reverse 攻防世界

ctf 专栏收录该内容

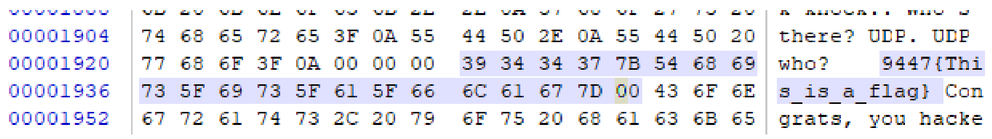
前言

本篇是攻防世界 Reverse 新手练习区的全解

1、insanity

下下来一个无后缀文件

扔进winhex



```
00001904   74 68 65 72 65 3F 0A 55  44 50 2E 0A 55 44 50 20   there? UDP. UDP 
00001920   77 68 6F 3F 0A 00 00 00  39 34 34 37 7B 54 68 69   who?    9447{Thi
00001936   73 5F 69 73 5F 61 5F 66  6C 61 67 7D 00 43 6F 6E   s_is_a_flag} Con
00001952   67 72 61 74 73 2C 20 79  6F 75 20 68 61 63 6B 65   grats, you hacke
```

得到flag

2、python-trade

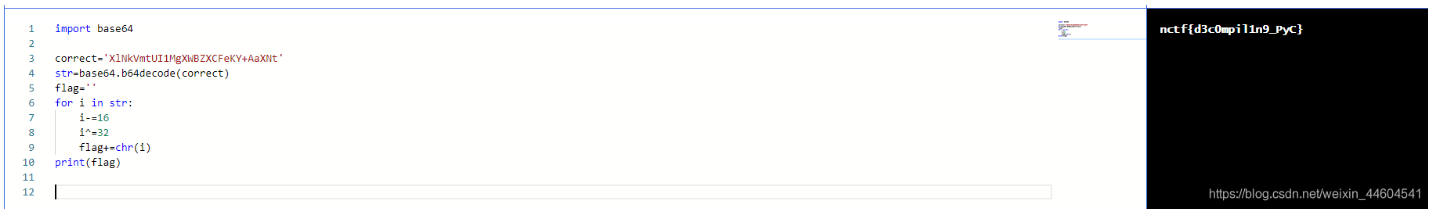
下下来一个pyc文件

反编译
得到

```
#!/usr/bin/env python
# visit http://tool.lu/pyc/ for more information
import base64

def encode(message):
    s = ''
    for i in message:
        x = ord(i) ^ 32
        x = x + 16
        s += chr(x)

    return base64.b64encode(s)

correct = 'XlNkVmtUI1MgXWBZXCFeKY+AaXNt'
flag = ''
print 'Input flag:'
flag = raw_input()
if encode(flag) == correct:
    print 'correct'
else:
    print 'wrong'
```

逆着写脚本

```
import base64

correct='XlNkVmtUI1MgXWBZXCFeKY+AaXNt'
str=base64.b64decode(correct)
flag=''
for i in str:
    i-=16
    i^=32
    flag+=chr(i)
print(flag)
```



```
import base64

correct='XlNkVmtUI1MgXWBZXCFeKY+AaXNt'
str=base64.b64decode(correct)
flag=''
for i in str:
    i-=16
    i^=32
    flag+=chr(i)
print(flag)
```

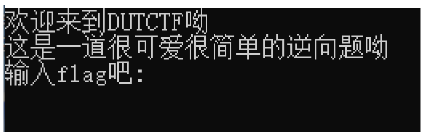
```
nctf{d3c0mpil1n9_PyC}
```

https://blog.csdn.net/weixin_44604541

得到flag

3、re1

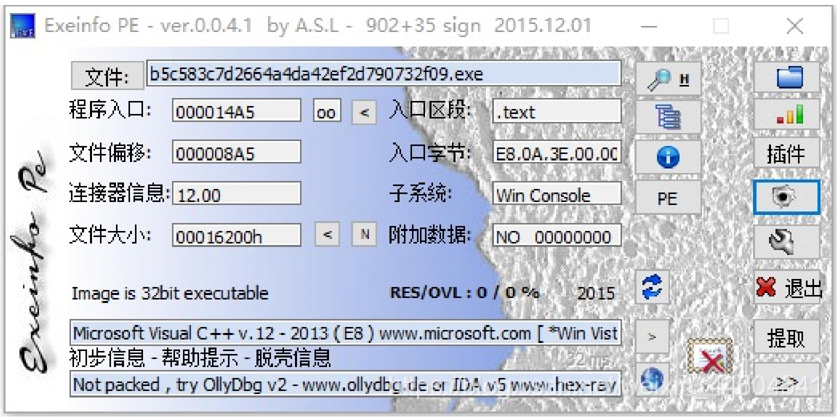
下下来一个exe



```
欢迎来到DUTCTF呦
这是一道很可爱很简单的逆向题呦
输入flag吧:
```

很简单的flag判断

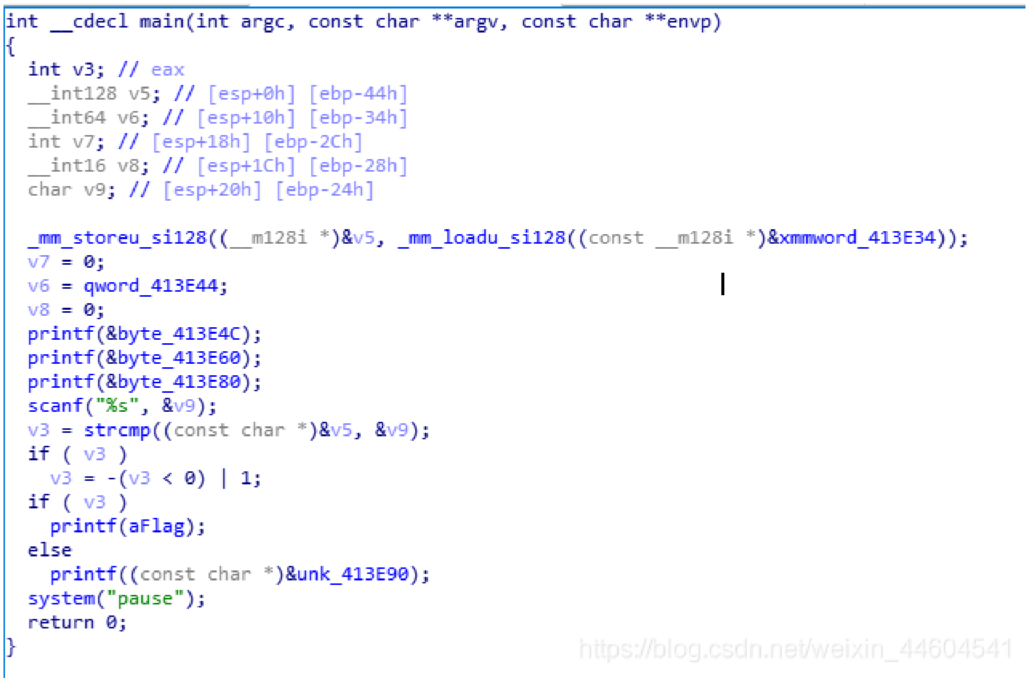
先查壳



```
Exeinfo PE - ver.0.0.4.1  by A.S.L -  902+35 sign  2015.12.01
文件: b5c583c7d2664a4da42ef2d790732f09.exe
程序入口: 000014A5        入口区段: .text
文件偏移: 000008A5        入口字节: E8.0A.3E.00.0C
连接器信息: 12.00         子系统: Win Console
文件大小: 00016200h       附加数据: NO  00000000
Image is 32bit executable      RES/OVL : 0 / 0 %     2015
Microsoft Visual C++ v.12 - 2013 ( E8 ) www.microsoft.com [ *Win Vist
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-ray
```

32位 Visual c++编译，没有加壳

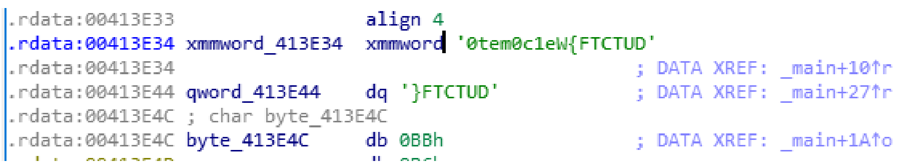
扔进IDA
查看伪码



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  __int128 v5; // [esp+0h] [ebp-44h]
  __int64 v6; // [esp+10h] [ebp-34h]
  int v7; // [esp+18h] [ebp-2Ch]
  __int16 v8; // [esp+1Ch] [ebp-28h]
  char v9; // [esp+20h] [ebp-24h]

  _mm_storeu_si128((__m128i *)&v5, _mm_loadu_si128((const __m128i *)&xmmword_413E34));
  v7 = 0;
  v6 = qword_413E44;
  v8 = 0;
  printf(&byte_413E4C);
  printf(&byte_413E60);
  printf(&byte_413E80);
  scanf("%s", &v9);
  v3 = strcmp((const char *)&v5, &v9);
  if ( v3 )
    v3 = -(v3 < 0) | 1;
  if ( v3 )
    printf(aFlag);
  else
    printf((const char *)&unk_413E90);
  system("pause");
  return 0;
}
```

- v3为真则flag判断正确
- v3是v5和v9的比较
- v9是输入
- 那flag就是v5了

跟踪找到

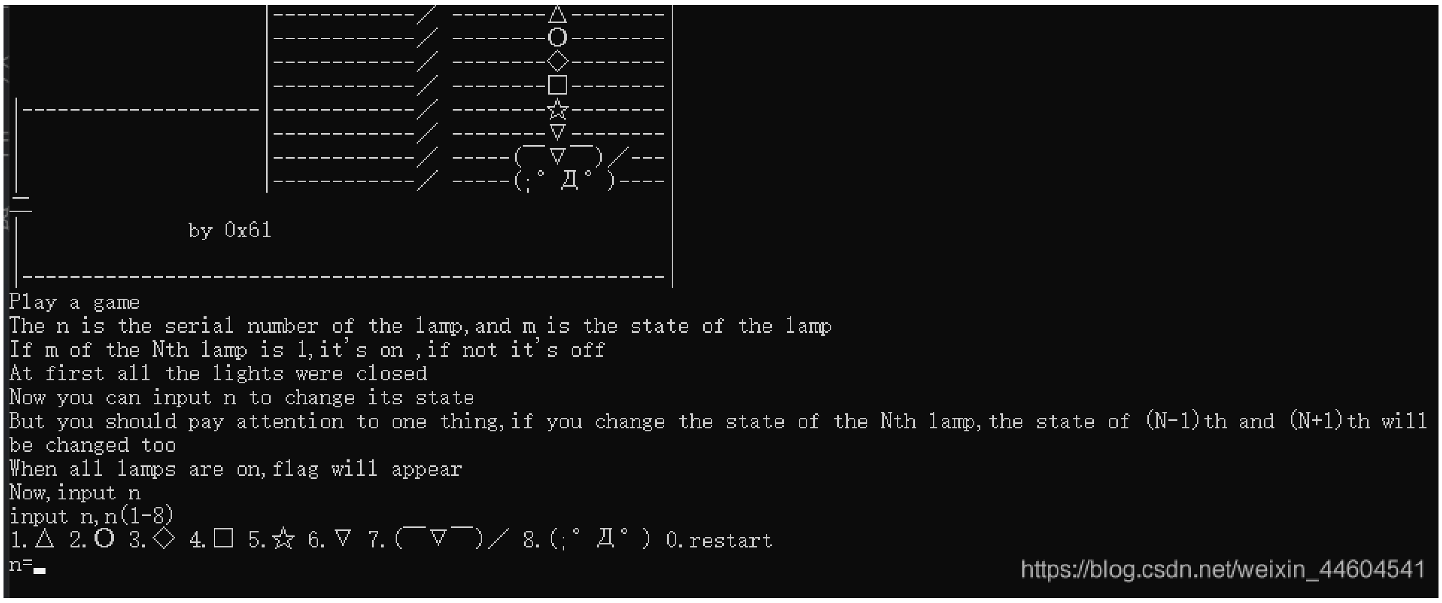


```
.rdata:00413E33                         align 4
.rdata:00413E34 xmmword_413E34          xmmword '0tem0c1eW{FTCTUD'
.rdata:00413E34                                                 ; DATA XREF: _main+10↑r
.rdata:00413E44 qword_413E44            dq '}FTCTUD'            ; DATA XREF: _main+27↑r
.rdata:00413E4C ; char byte_413E4C
.rdata:00413E4C byte_413E4C             db 0BBh                 ; DATA XREF: _main+1A↑o
```

得到flag：`DUTCTF{We1c0met0DUTCTF}`

4、game

据说是个游戏

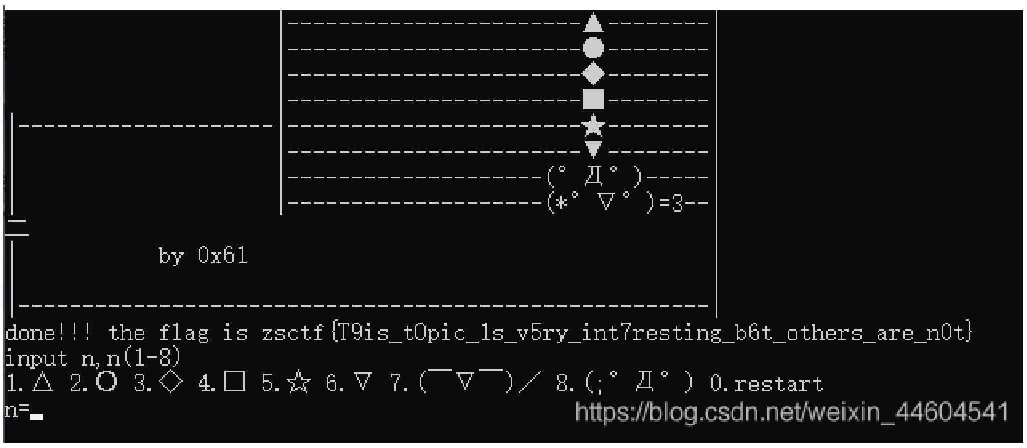


```
                        by 0x61

Play a game
The n is the serial number of the lamp,and m is the state of the lamp
If m of the Nth lamp is 1,it's on ,if not it's off
At first all the lights were closed
Now you can input n to change its state
But you should pay attention to one thing,if you change the state of the Nth lamp,the state of (N-1)th and (N+1)th will
be changed too
When all lamps are on,flag will appear
Now,input n
input n,n(1-8)
1.△ 2.O 3.◇ 4.□ 5.☆ 6.▽ 7.(￣▽￣)/ 8.(;°Д°) 0.restart
n=
```

https://blog.csdn.net/weixin_44604541

非预期解

直接玩出来
写个算法，得到12345678

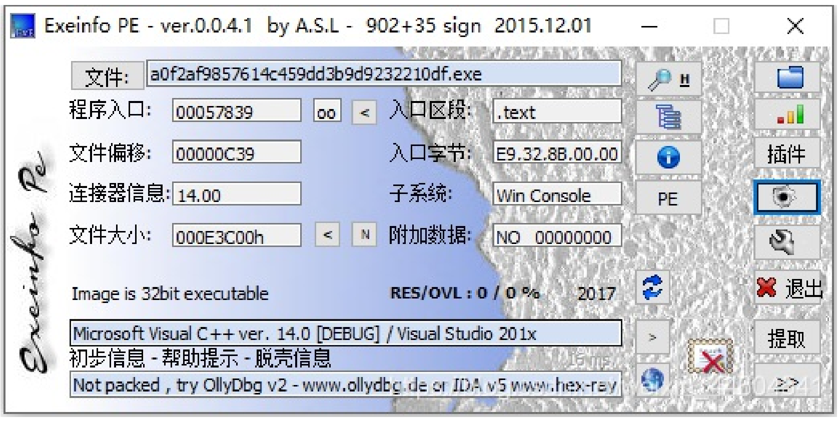


```
                by 0x61

done!!! the flag is zsctf{T9is_tOpic_1s_v5ry_int7resting_b6t_others_are_n0t}
input n,n(1-8)
1.△ 2.O 3.◇ 4.□ 5.☆ 6.▽ 7.(￣▽￣)/ 8.(;°Д°) 0.restart
n=
```

https://blog.csdn.net/weixin_44604541

预期解

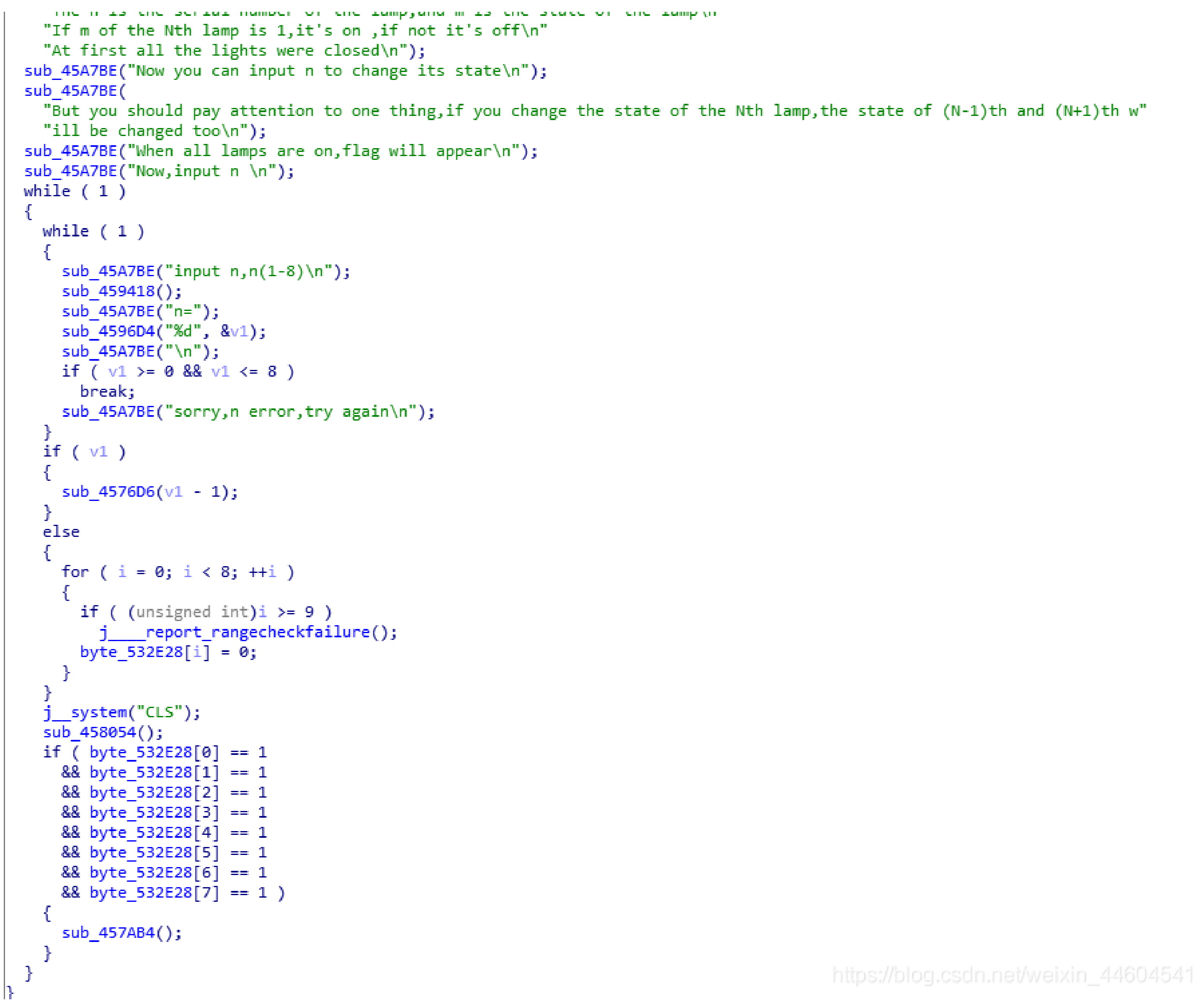
PE看信息



Exeinfo PE - ver.0.0.4.1 by A.S.L - 902+35 sign 2015.12.01

文件: a0f2af9857614c459dd3b9d9232210df.exe
程序入口: 00057839 　入口区段: .text
文件偏移: 00000C39 　入口字节: E9.32.8B.00.00
连接器信息: 14.00 　子系统: Win Console
文件大小: 000E3C00h 　附加数据: NO 00000000
Image is 32bit executable 　RES/OVL : 0 / 0 % 　2017
Microsoft Visual C++ ver. 14.0 [DEBUG] / Visual Studio 201x
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-ray

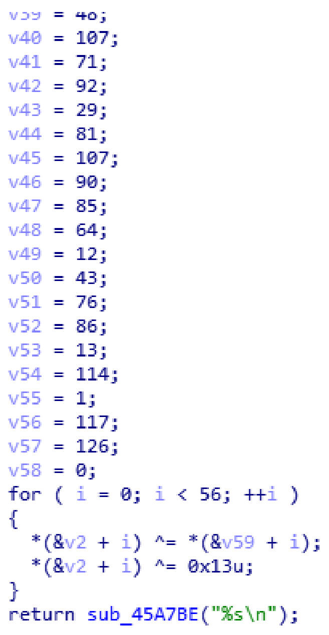
扔进IDA



```
    "The n is the serial number of the lamp,and m is the state of the lamp\n"
    "If m of the Nth lamp is 1,it's on ,if not it's off\n"
    "At first all the lights were closed\n");
  sub_45A7BE("Now you can input n to change its state\n");
  sub_45A7BE(
    "But you should pay attention to one thing,if you change the state of the Nth lamp,the state of (N-1)th and (N+1)th w"
    "ill be changed too\n");
  sub_45A7BE("When all lamps are on,flag will appear\n");
  sub_45A7BE("Now,input n \n");
  while ( 1 )
  {
    while ( 1 )
    {
      sub_45A7BE("input n,n(1-8)\n");
      sub_459418();
      sub_45A7BE("n=");
      sub_4596D4("%d", &v1);
      sub_45A7BE("\n");
      if ( v1 >= 0 && v1 <= 8 )
        break;
      sub_45A7BE("sorry,n error,try again\n");
    }
    if ( v1 )
    {
      sub_4576D6(v1 - 1);
    }
    else
    {
      for ( i = 0; i < 8; ++i )
      {
        if ( (unsigned int)i >= 9 )
          j____report_rangecheckfailure();
        byte_532E28[i] = 0;
      }
    }
    j__system("CLS");
    sub_458054();
    if ( byte_532E28[0] == 1
      && byte_532E28[1] == 1
      && byte_532E28[2] == 1
      && byte_532E28[3] == 1
      && byte_532E28[4] == 1
      && byte_532E28[5] == 1
      && byte_532E28[6] == 1
      && byte_532E28[7] == 1 )
    {
      sub_457AB4();
    }
  }
}
```

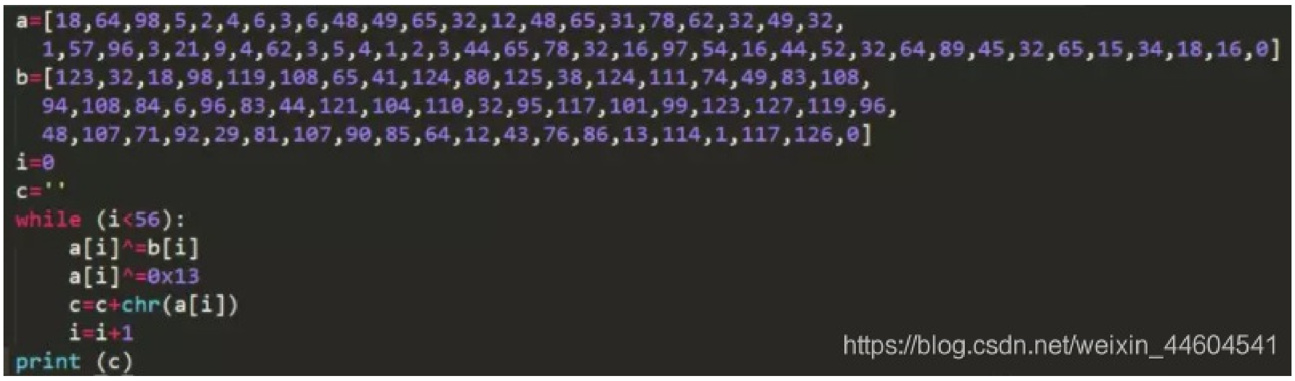
- 先是判断是输入的是否是1-8
- 然后进入后面的if判断然后进行循环，这个时候应该就是程序的亮暗的显示
- 如果byte_532E28每一位都是1，那么，就会进入sub_457AB4，应该就是最后的flag的地方

跟进sub_457AB4



```
v39 = 48;
v40 = 107;
v41 = 71;
v42 = 92;
v43 = 29;
v44 = 81;
v45 = 107;
v46 = 90;
v47 = 85;
v48 = 64;
v49 = 12;
v50 = 43;
v51 = 76;
v52 = 86;
v53 = 13;
v54 = 114;
v55 = 1;
v56 = 117;
v57 = 126;
v58 = 0;
for ( i = 0; i < 56; ++i )
{
  *(&v2 + i) ^= *(&v59 + i);
  *(&v2 + i) ^= 0x13u;
}
return sub_45A7BE("%s\n");
```

逆着写脚本

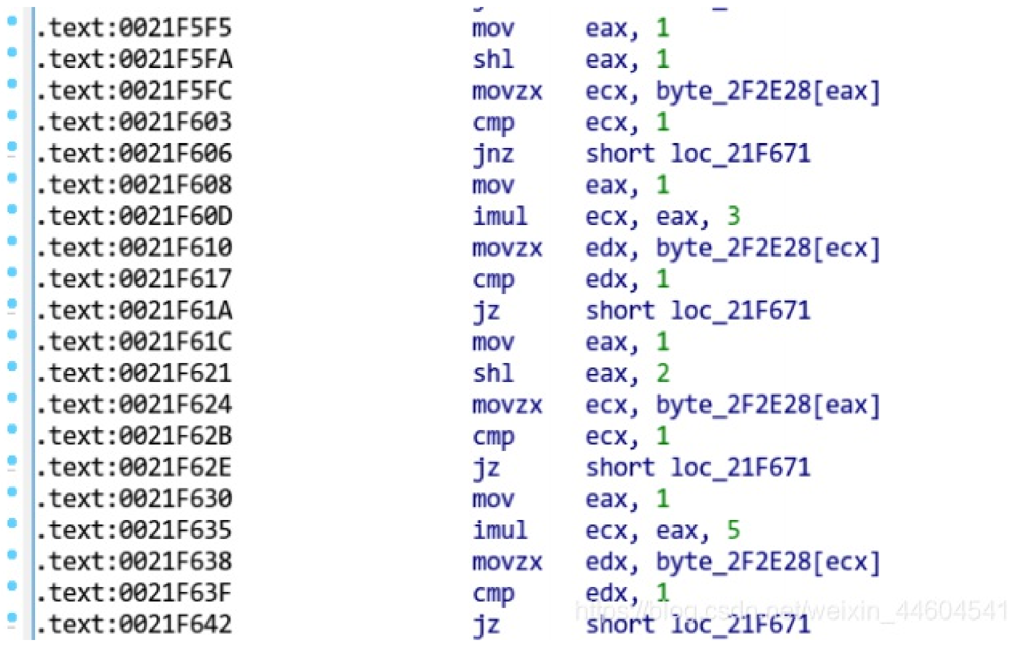


```
a=[18,64,98,5,2,4,6,3,6,48,49,65,32,12,48,65,31,78,62,32,49,32,
  1,57,96,3,21,9,4,62,3,5,4,1,2,3,44,65,78,32,16,97,54,16,44,52,32,64,89,45,32,65,15,34,18,16,0]
b=[123,32,18,98,119,108,65,41,124,80,125,38,124,111,74,49,83,108,
  94,108,84,6,96,83,44,121,104,110,32,95,117,101,99,123,127,119,96,
  48,107,71,92,29,81,107,90,85,64,12,43,76,86,13,114,1,117,126,0]
i=0
c=''
while (i<56):
    a[i]^=b[i]
    a[i]^=0x13
    c=c+chr(a[i])
    i=i+1
print (c)
```

即可

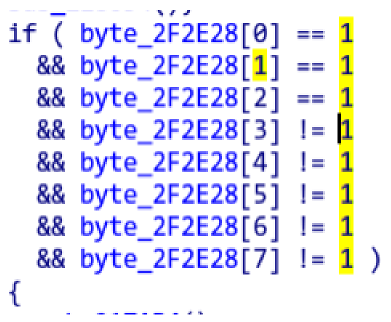
一种修改思路

把对标3-7的jnz改为jz



```
.text:0021F5F5        mov     eax, 1
.text:0021F5FA        shl     eax, 1
.text:0021F5FC        movzx   ecx, byte_2F2E28[eax]
.text:0021F603        cmp     ecx, 1
.text:0021F606        jnz     short loc_21F671
.text:0021F608        mov     eax, 1
.text:0021F60D        imul    ecx, eax, 3
.text:0021F610        movzx   edx, byte_2F2E28[ecx]
.text:0021F617        cmp     edx, 1
.text:0021F61A        jz      short loc_21F671
.text:0021F61C        mov     eax, 1
.text:0021F621        shl     eax, 2
.text:0021F624        movzx   ecx, byte_2F2E28[eax]
.text:0021F62B        cmp     ecx, 1
.text:0021F62E        jz      short loc_21F671
.text:0021F630        mov     eax, 1
.text:0021F635        imul    ecx, eax, 5
.text:0021F638        movzx   edx, byte_2F2E28[ecx]
.text:0021F63F        cmp     edx, 1
.text:0021F642        jz      short loc_21F671
```

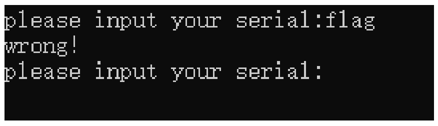
使得初始状态变为



```
if ( byte_2F2E28[0] == 1
  && byte_2F2E28[1] == 1
  && byte_2F2E28[2] == 1
  && byte_2F2E28[3] != 1
  && byte_2F2E28[4] != 1
  && byte_2F2E28[5] != 1
  && byte_2F2E28[6] != 1
  && byte_2F2E28[7] != 1 )
{
```

这样进入游戏输个2就行了

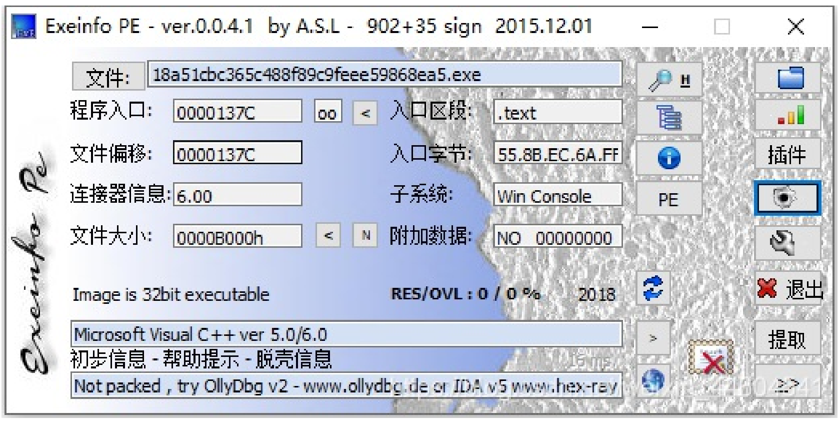
5、Hello, CTF



```
please input your serial:flag
wrong!
please input your serial:
```

flag判断

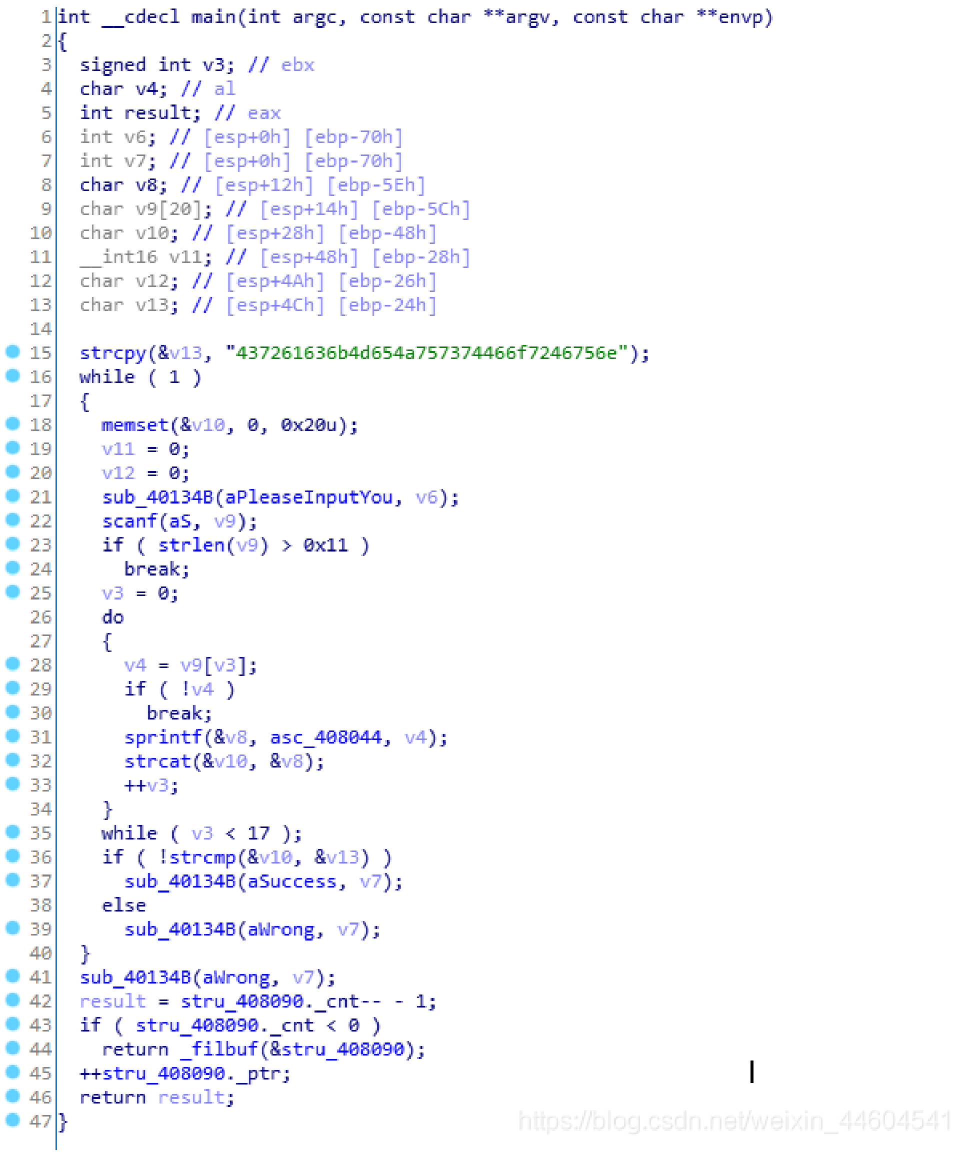
扔进PE



```
Exeinfo PE - ver.0.0.4.1  by A.S.L -  902+35 sign  2015.12.01
文件: 18a51cbc365c488f89c9feee59868ea5.exe
程序入口: 0000137C        入口区段: .text
文件偏移: 0000137C        入口字节: 55.8B.EC.6A.FF
连接器信息: 6.00          子系统: Win Console
文件大小: 0000B000h       附加数据: NO  00000000
Image is 32bit executable        RES/OVL : 0 / 0 %   2018
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-ray
```

32位无壳

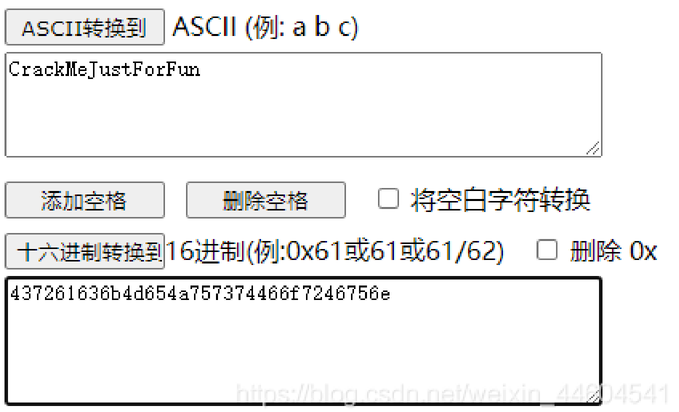
扔进IDA



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  signed int v3; // ebx
  char v4; // al
  int result; // eax
  int v6; // [esp+0h] [ebp-70h]
  int v7; // [esp+0h] [ebp-70h]
  char v8; // [esp+12h] [ebp-5Eh]
  char v9[20]; // [esp+14h] [ebp-5Ch]
  char v10; // [esp+28h] [ebp-48h]
  __int16 v11; // [esp+48h] [ebp-28h]
  char v12; // [esp+4Ah] [ebp-26h]
  char v13; // [esp+4Ch] [ebp-24h]

  strcpy(&v13, "437261636b4d654a757374466f7246756e");
  while ( 1 )
  {
    memset(&v10, 0, 0x20u);
    v11 = 0;
    v12 = 0;
    sub_40134B(aPleaseInputYou, v6);
    scanf(aS, v9);
    if ( strlen(v9) > 0x11 )
      break;
    v3 = 0;
    do
    {
      v4 = v9[v3];
      if ( !v4 )
        break;
      sprintf(&v8, asc_408044, v4);
      strcat(&v10, &v8);
      ++v3;
    }
    while ( v3 < 17 );
    if ( !strcmp(&v10, &v13) )
      sub_40134B(aSuccess, v7);
    else
      sub_40134B(aWrong, v7);
  }
  sub_40134B(aWrong, v7);
  result = stru_408090._cnt-- - 1;
  if ( stru_408090._cnt < 0 )
    return _filbuf(&stru_408090);
  ++stru_408090._ptr;
  return result;
}
```

逻辑

- 将用户输入的字符单个与v13字串单个进行比对，然后判断是否输入正确
- v13对应的字串是16进制



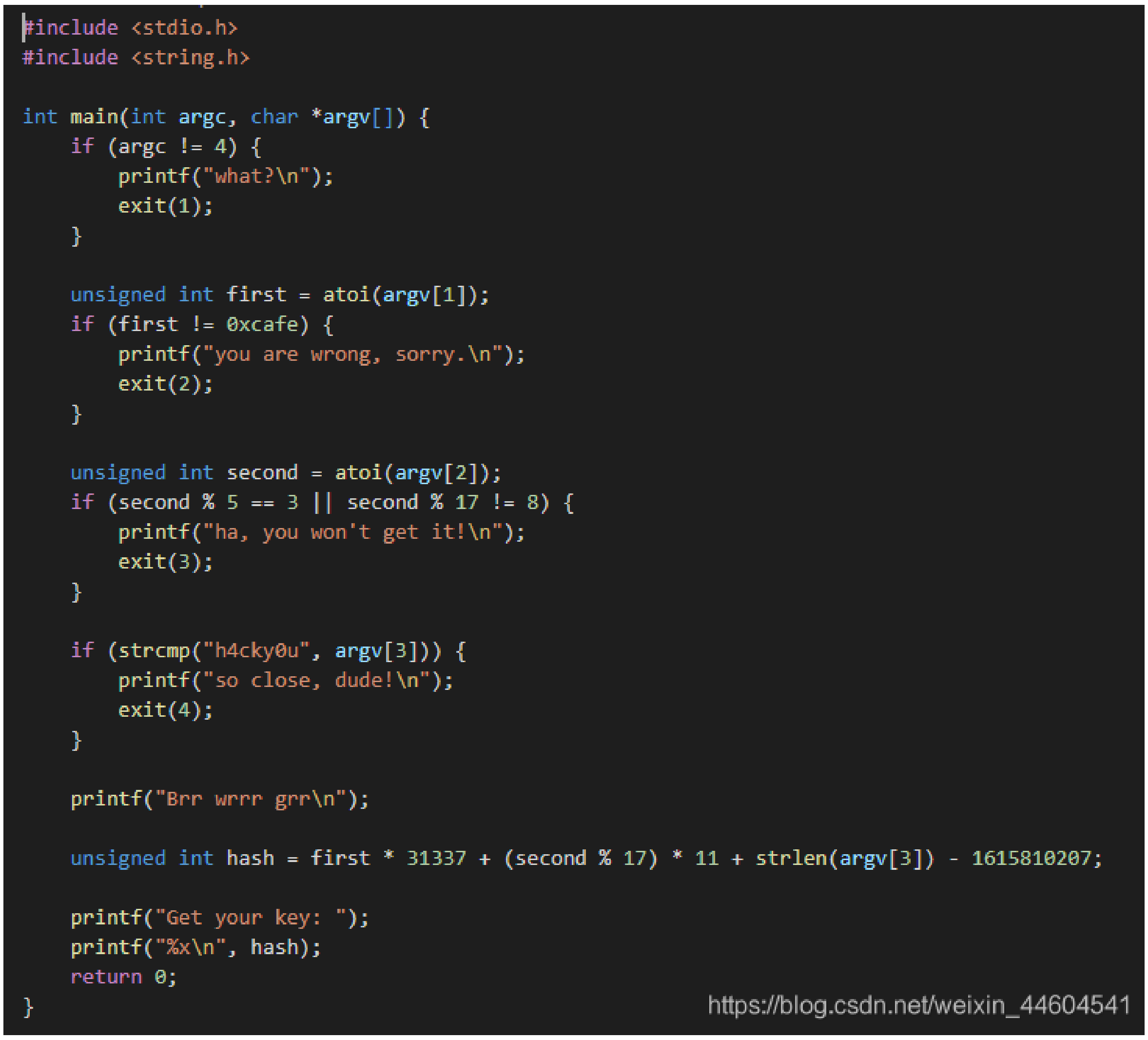
```
CrackMeJustForFun
```

```
437261636b4d654a757374466f7246756e
```

得到flag

6、open-source

下下来一段c源码



```c
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]) {
    if (argc != 4) {
        printf("what?\n");
        exit(1);
    }

    unsigned int first = atoi(argv[1]);
    if (first != 0xcafe) {
        printf("you are wrong, sorry.\n");
        exit(2);
    }

    unsigned int second = atoi(argv[2]);
    if (second % 5 == 3 || second % 17 != 8) {
        printf("ha, you won't get it!\n");
        exit(3);
    }

    if (strcmp("h4cky0u", argv[3])) {
        printf("so close, dude!\n");
        exit(4);
    }

    printf("Brr wrrr grr\n");

    unsigned int hash = first * 31337 + (second % 17) * 11 + strlen(argv[3]) - 1615810207;

    printf("Get your key: ");
    printf("%x\n", hash);
    return 0;
}
```

https://blog.csdn.net/weixin_44604541

分析

- `argv[1] = 0xcafe`
- `argv[2] % 5 != 3 && argv[2] % 17 == 8`，就让他为25吧
- `argv[3] = "h4cky0u"`
- 然后计算hash

由此

```c
#include <stdio.h>
#include <string.h>

int main(int argc, char* argv[]) {
    int first = 0xcafe;
    int second = 25;
    argv[3] = "h4cky0u";
    unsigned int hash = first * 31337 + (second % 17) * 11 + strlen(argv[3]) - 1615810207;
    printf("Get your key: ");
    printf("%x\n", hash);
    system("PAUSE");
    return 0;
}
```

运行得到flag

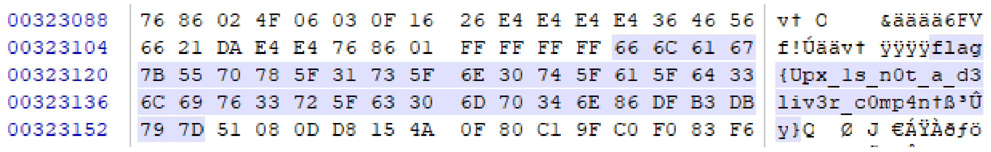
```
Get your key: c0ffee
```

7、simple-unpack

下下来一个无后缀文件

非预期解

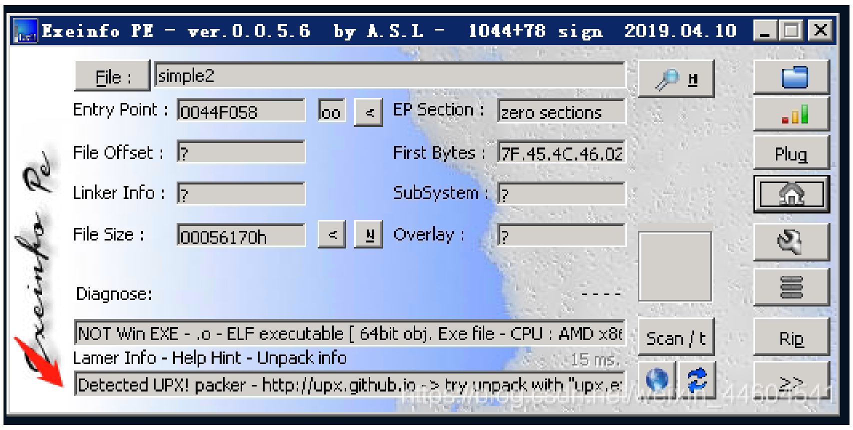
扔进winhex
直接找到flag



```
00323088  76 86 02 4F 06 03 0F 16  26 E4 E4 E4 E4 36 46 56  vt O     &ääää6FV
00323104  66 21 DA E4 E4 76 86 01  FF FF FF FF 66 6C 61 67  f!Úäävt ÿÿÿÿflag
00323120  7B 55 70 78 5F 31 73 5F  6E 30 74 5F 61 5F 64 33  {Upx_1s_n0t_a_d3
00323136  6C 69 76 33 72 5F 63 30  6D 70 34 6E 86 DF B3 DB  liv3r_c0mp4ntß³Û
00323152  79 7D 51 08 0D D8 15 4A  0F 80 C1 9F C0 F0 83 F6  y}Q  Ø J €ÁŸÀðƒö
```

预期解

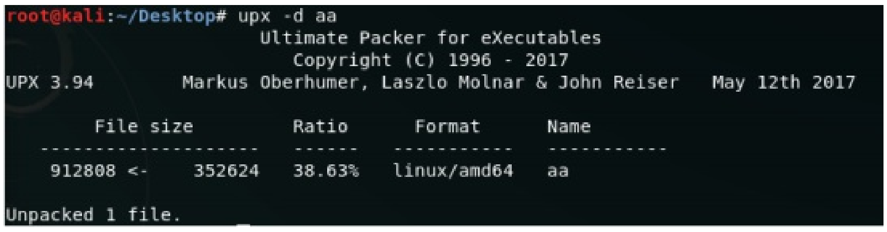
PE查壳



```
Exeinfo PE - ver.0.0.5.6  by A.S.L - 1044+78 sign  2019.04.10
File :        simple2
Entry Point : 0044F058        oo  <   EP Section : zero sections
File Offset : ?                       First Bytes : 7F.45.4C.46.02
Linker Info : ?                       SubSystem :   ?
File Size :   00056170h       <  H    Overlay :     ?
Diagnose:
NOT Win EXE - .o - ELF executable [ 64bit obj. Exe file - CPU : AMD x8
Lamer Info - Help Hint - Unpack info                         15 ms.
Detected UPX! packer - http://upx.github.io -> try unpack with "upx.e
```

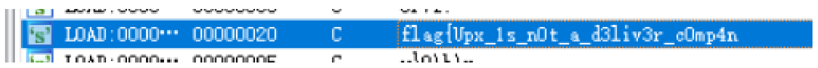
有upx壳

upx -d 脱壳



```
root@kali:~/Desktop# upx -d aa
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2017
UPX 3.94        Markus Oberhumer, Laszlo Molnar & John Reiser   May 12th 2017

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    912808 <-    352624   38.63%   linux/amd64   aa

Unpacked 1 file.
```

扔进IDA

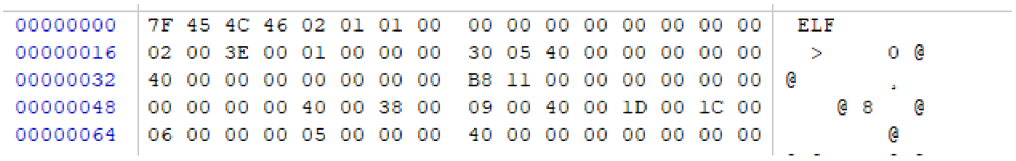


```
LOAD:0000··· 00000020    C    flag{Upx_1s_n0t_a_d3liv3r_c0mp4n
```

得到flag

8、logmein

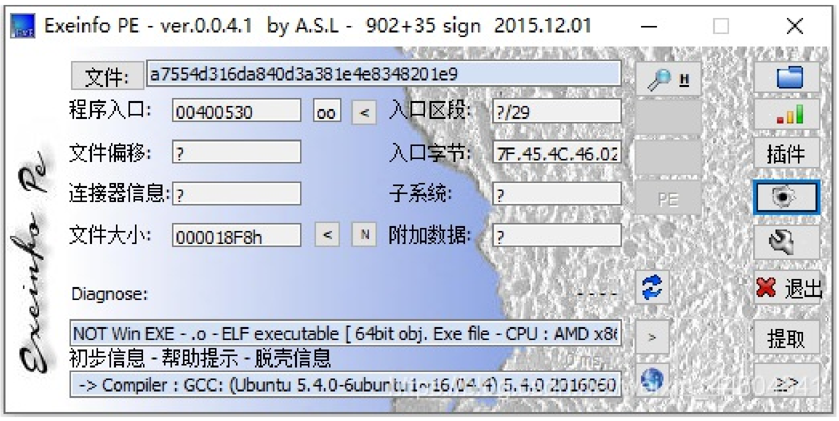
扔进winhex



```
00000000  7F 45 4C 46 02 01 01 00  00 00 00 00 00 00 00 00   ELF
00000016  02 00 3E 00 01 00 00 00  30 05 40 00 00 00 00 00    >      0 @
00000032  40 00 00 00 00 00 00 00  B8 11 00 00 00 00 00 00  @       ¸
00000048  00 00 00 00 40 00 38 00  09 00 40 00 1D 00 1C 00      @ 8   @
00000064  06 00 00 00 05 00 00 00  40 00 00 00 00 00 00 00         @
```

ELF文件

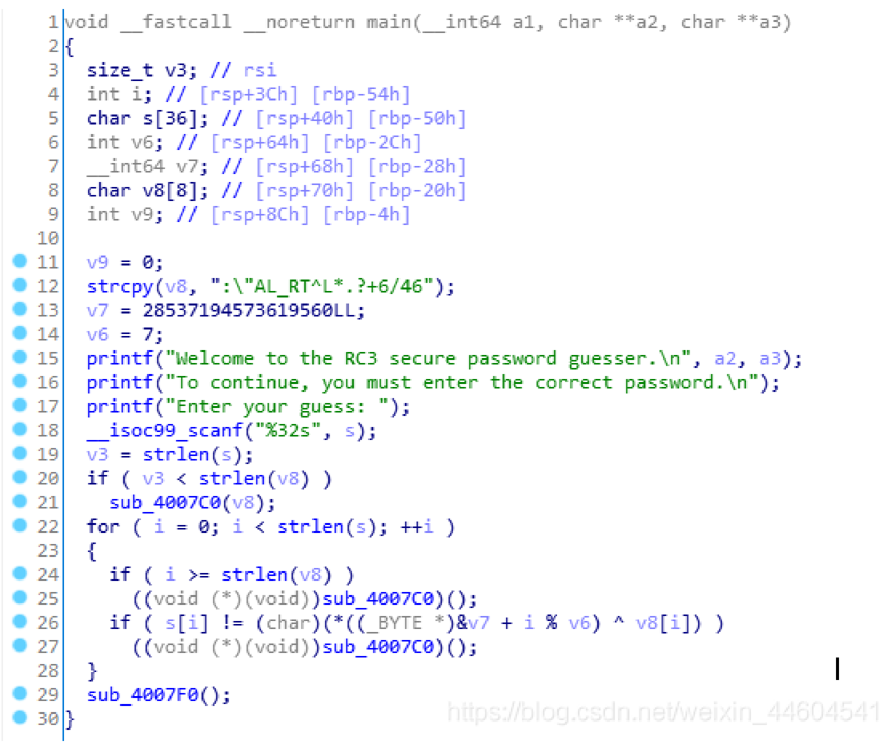
PE查壳


Exeinfo PE - ver.0.0.4.1 by A.S.L - 902+35 sign 2015.12.01

文件: a7554d316da840d3a381e4e8348201e9
程序入口: 00400530
入口区段: ?/29
文件偏移: ?
入口字节: 7F.45.4C.46.02
连接器信息: ?
子系统: ?
文件大小: 000018F8h
附加数据: ?
Diagnose:
NOT Win EXE - .o - ELF executable [64bit obj. Exe file - CPU : AMD x8
初步信息 - 帮助提示 - 脱壳信息
-> Compiler : GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 2016060

64位

扔进IDA
主函数伪码


```
void __fastcall __noreturn main(__int64 a1, char **a2, char **a3)
{
  size_t v3; // rsi
  int i; // [rsp+3Ch] [rbp-54h]
  char s[36]; // [rsp+40h] [rbp-50h]
  int v6; // [rsp+64h] [rbp-2Ch]
  __int64 v7; // [rsp+68h] [rbp-28h]
  char v8[8]; // [rsp+70h] [rbp-20h]
  int v9; // [rsp+8Ch] [rbp-4h]

  v9 = 0;
  strcpy(v8, ":\"AL_RT^L*.?+6/46");
  v7 = 28537194573619560LL;
  v6 = 7;
  printf("Welcome to the RC3 secure password guesser.\n", a2, a3);
  printf("To continue, you must enter the correct password.\n");
  printf("Enter your guess: ");
  __isoc99_scanf("%32s", s);
  v3 = strlen(s);
  if ( v3 < strlen(v8) )
    sub_4007C0(v8);
  for ( i = 0; i < strlen(s); ++i )
  {
    if ( i >= strlen(v8) )
      ((void (*)(void))sub_4007C0)();
    if ( s[i] != (char)(*((_BYTE *)&v7 + i % v6) ^ v8[i]) )
      ((void (*)(void))sub_4007C0)();
  }
  sub_4007F0();
}
```

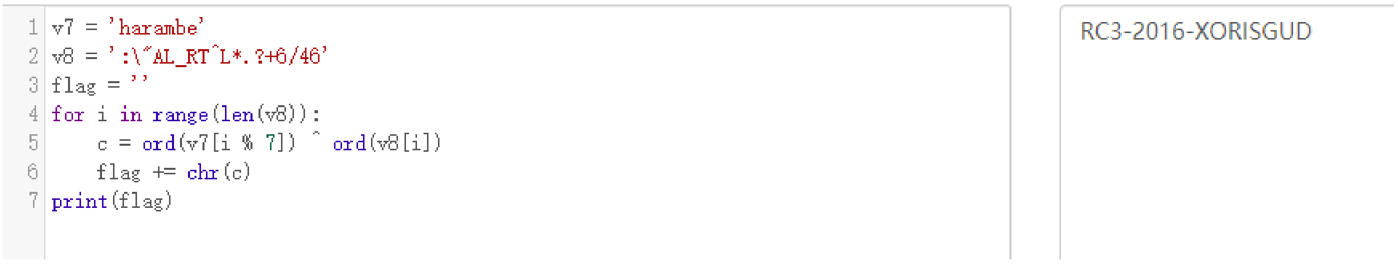
分析

- v7赋值，字符型是`v7 = 'ebmarah';`，需要注意的是，x86系列的CPU都是以小端序储存数据的，即低位字节存入低地址，高位字节存入高地址，所以正确的字符串应该反过来`v7='harambe';`
- v8赋值`v8 = ':\"AL_RT^L*.?+6/46'`
- v6为7
- 输入的flag应该是`v7[i%v6]^v8[i]`

于是有脚本

```python
v7 = 'harambe'
v8 = ':\"AL_RT^L*.?+6/46'
flag = ''
for i in range(len(v8)):
    c = ord(v7[i % 7]) ^ ord(v8[i])
    flag += chr(c)
print(flag)
```

得到flag



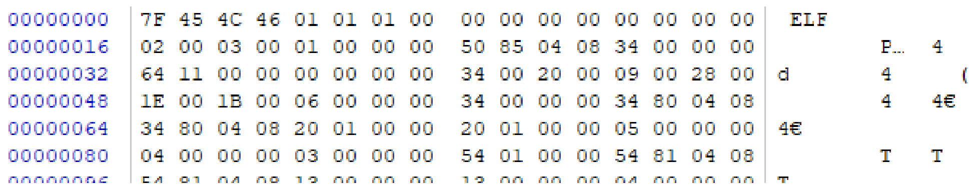
```python
v7 = 'harambe'
v8 = ':\"AL_RT^L*.?+6/46'
flag = ''
for i in range(len(v8)):
    c = ord(v7[i % 7]) ^ ord(v8[i])
    flag += chr(c)
print(flag)
```

```
RC3-2016-XORISGUD
```

9、no-strings-attached

下下来一个无后缀文件

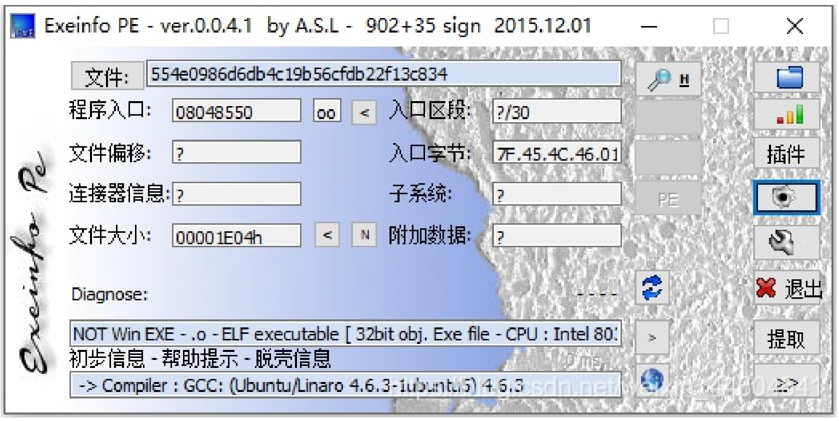
扔进winhex



```
00000000  7F 45 4C 46 01 01 01 00  00 00 00 00 00 00 00 00   ELF
00000016  02 00 03 00 01 00 00 00  50 85 04 08 34 00 00 00          P…   4
00000032  64 11 00 00 00 00 00 00  34 00 20 00 09 00 28 00  d       4      (
00000048  1E 00 1B 00 06 00 00 00  34 00 00 00 34 80 04 08          4    4€
00000064  34 80 04 08 20 01 00 00  20 01 00 00 05 00 00 00  4€
00000080  04 00 00 00 03 00 00 00  54 01 00 00 54 81 04 08          T    T
```

是个ELF文件

PE查壳



Exeinfo PE - ver.0.0.4.1 by A.S.L - 902+35 sign 2015.12.01

文件:	554e0986d6db4c19b56cfdb22f13c834		
程序入口:	08048550	入口区段:	?/30
文件偏移:	?	入口字节:	7F.45.4C.46.01
连接器信息:	?	子系统:	?
文件大小:	00001E04h	附加数据:	?

Diagnose:

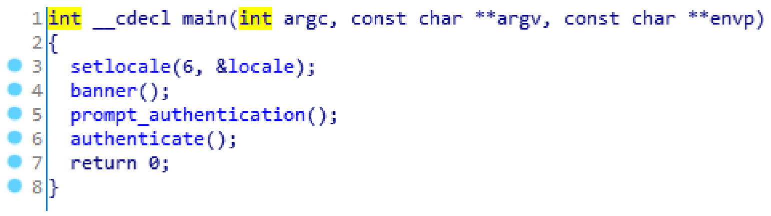
NOT Win EXE - .o - ELF executable [32bit obj. Exe file - CPU : Intel 80:

初步信息 - 帮助提示 - 脱壳信息

-> Compiler : GCC: (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3

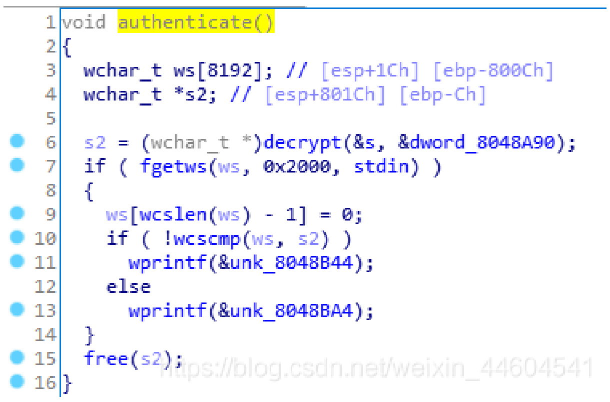
32位

扔进IDA



```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  setlocale(6, &locale);
  banner();
  prompt_authentication();
  authenticate();
  return 0;
}
```

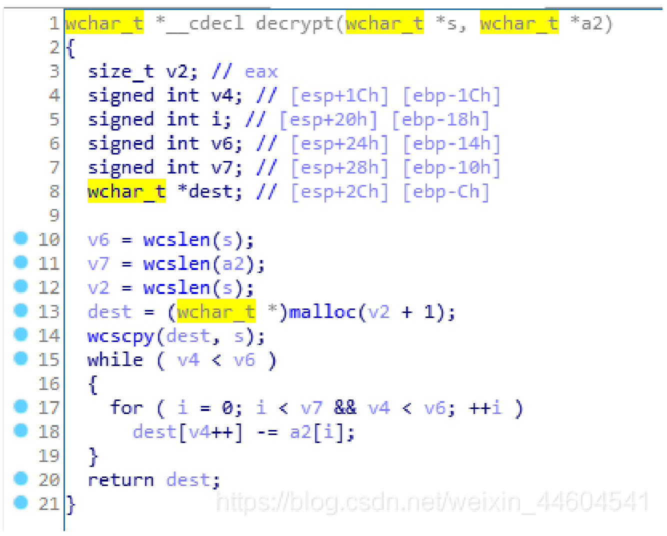
看了看关键函数`authenticate()`



```
void authenticate()
{
  wchar_t ws[8192]; // [esp+1Ch] [ebp-800Ch]
  wchar_t *s2; // [esp+801Ch] [ebp-Ch]

  s2 = (wchar_t *)decrypt(&s, &dword_8048A90);
  if ( fgetws(ws, 0x2000, stdin) )
  {
    ws[wcslen(ws) - 1] = 0;
    if ( !wcscmp(ws, s2) )
      wprintf(&unk_8048B44);
    else
      wprintf(&unk_8048BA4);
  }
  free(s2);
}
```

- 调用了 decrypt 函数加密得到 s2
- 然后和从命令行中输入的 ws 做对比
- 输入正确，输出 8048B44 处的值,查找可知这个值是个字符串，即“Success! Welcome back!”

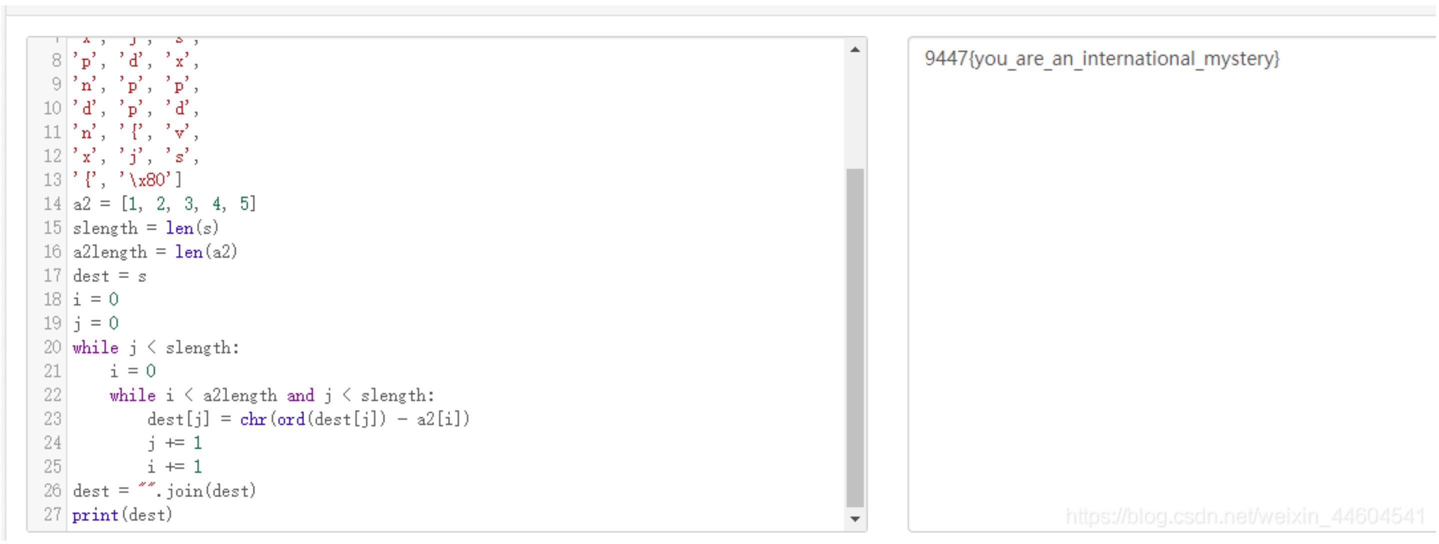
可见s2就是flag
看下decrypt



```
wchar_t *__cdecl decrypt(wchar_t *s, wchar_t *a2)
{
  size_t v2; // eax
  signed int v4; // [esp+1Ch] [ebp-1Ch]
  signed int i; // [esp+20h] [ebp-18h]
  signed int v6; // [esp+24h] [ebp-14h]
  signed int v7; // [esp+28h] [ebp-10h]
  wchar_t *dest; // [esp+2Ch] [ebp-Ch]

  v6 = wcslen(s);
  v7 = wcslen(a2);
  v2 = wcslen(s);
  dest = (wchar_t *)malloc(v2 + 1);
  wcscpy(dest, s);
  while ( v4 < v6 )
  {
    for ( i = 0; i < v7 && v4 < v6; ++i )
      dest[v4++] -= a2[i];
  }
  return dest;
}
```

根据他的意思编写下

```python
s = [ ':', '6', '7',
';', '\x80', 'z',
'q', 'x', 'c',
'f', 's', 'g',
'b', 'e', 's',
'`', 'k', 'q',
'x', 'j', 's',
'p', 'd', 'x',
'n', 'p', 'p',
'd', 'p', 'd',
'n', '{', 'v',
'x', 'j', 's',
'{', '\x80']
a2 = [1, 2, 3, 4, 5]
slength = len(s)
a2length = len(a2)
dest = s
i = 0
j = 0
while j < slength:
 i = 0
 while i < a2length and j < slength:
  dest[j] = chr(ord(dest[j]) - a2[i])
  j += 1
  i += 1
dest = "".join(dest)
print(dest)
```



```python
'p', 'd', 'x',
'n', 'p', 'p',
'd', 'p', 'd',
'n', '{', 'v',
'x', 'j', 's',
'{', '\x80']
a2 = [1, 2, 3, 4, 5]
slength = len(s)
a2length = len(a2)
dest = s
i = 0
j = 0
while j < slength:
    i = 0
    while i < a2length and j < slength:
        dest[j] = chr(ord(dest[j]) - a2[i])
        j += 1
        i += 1
dest = "".join(dest)
print(dest)
```

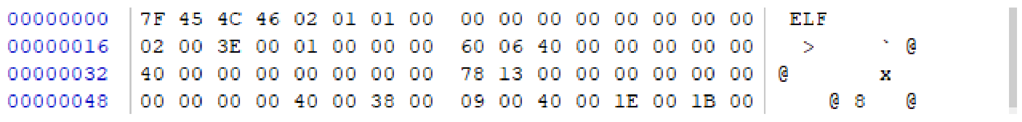
```
9447{you_are_an_international_mystery}
```

得到flag

10、getit

下下来一个无后缀文件

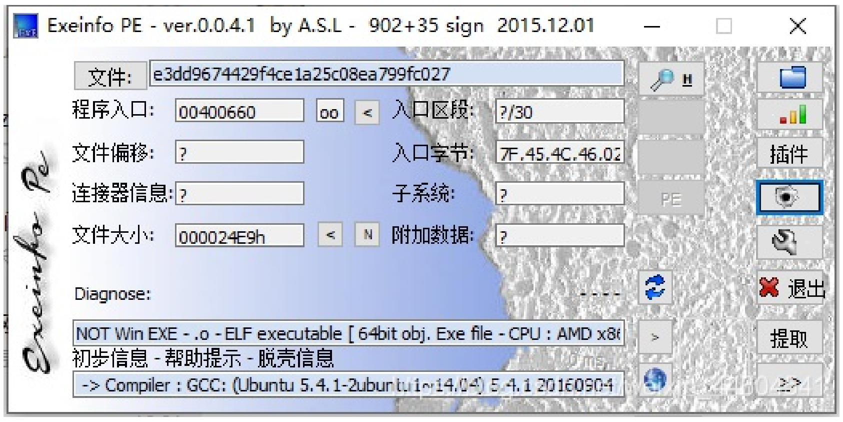
扔进winhex



```
00000000  7F 45 4C 46 02 01 01 00  00 00 00 00 00 00 00 00   ELF
00000016  02 00 3E 00 01 00 00 00  60 06 40 00 00 00 00 00    >     ` @
00000032  40 00 00 00 00 00 00 00  78 13 00 00 00 00 00 00  @       x
00000048  00 00 00 00 40 00 38 00  09 00 40 00 1E 00 1B 00      @ 8   @
```

是个ELF文件

PE查壳

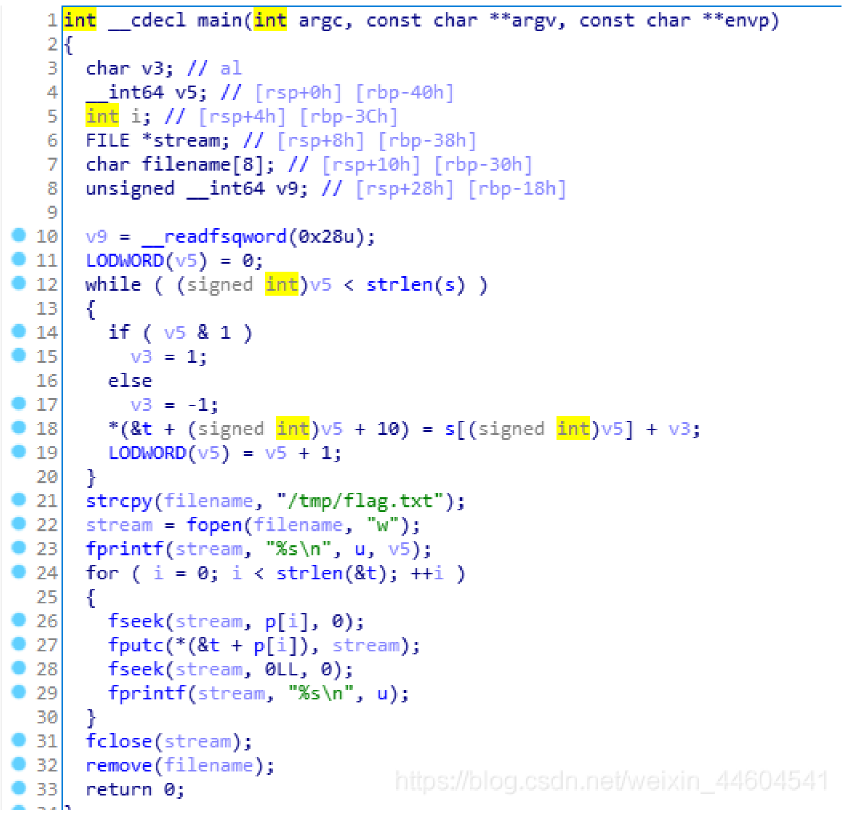


Exeinfo PE - ver.0.0.4.1 by A.S.L - 902+35 sign 2015.12.01

文件: e3dd9674429f4ce1a25c08ea799fc027
程序入口: 00400660 oo <
入口区段: ?/30
文件偏移: ?
入口字节: 7F.45.4C.46.02
连接器信息: ?
子系统: ?
文件大小: 000024E9h < N
附加数据: ?
Diagnose:
NOT Win EXE - .o - ELF executable [64bit obj. Exe file - CPU : AMD x8
初步信息 - 帮助提示 - 脱壳信息
-> Compiler : GCC: (Ubuntu 5.4.1-2ubuntu1~14.04) 5.4.1 20160904

64位

扔进IDA

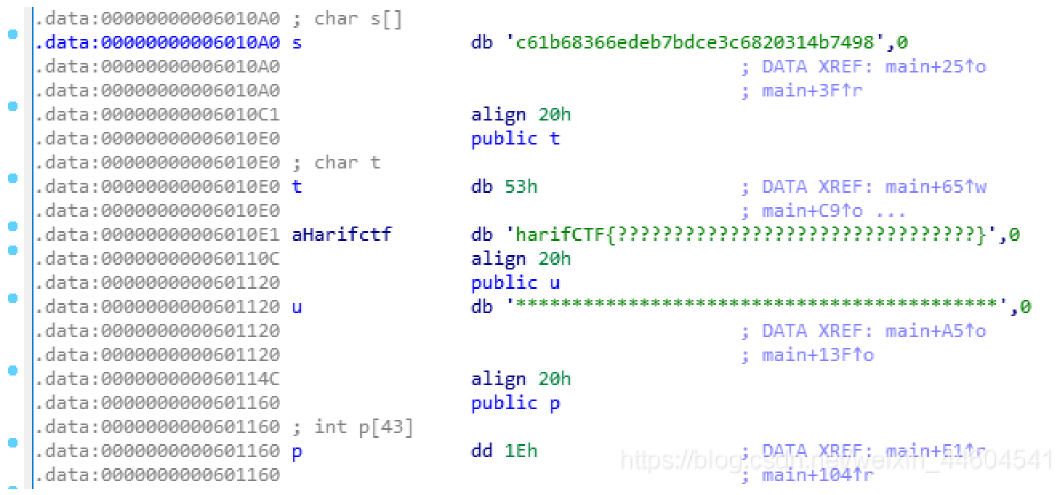


```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v3; // al
  __int64 v5; // [rsp+0h] [rbp-40h]
  int i; // [rsp+4h] [rbp-3Ch]
  FILE *stream; // [rsp+8h] [rbp-38h]
  char filename[8]; // [rsp+10h] [rbp-30h]
  unsigned __int64 v9; // [rsp+28h] [rbp-18h]

  v9 = __readfsqword(0x28u);
  LODWORD(v5) = 0;
  while ( (signed int)v5 < strlen(s) )
  {
    if ( v5 & 1 )
      v3 = 1;
    else
      v3 = -1;
    *(&t + (signed int)v5 + 10) = s[(signed int)v5] + v3;
    LODWORD(v5) = v5 + 1;
  }
  strcpy(filename, "/tmp/flag.txt");
  stream = fopen(filename, "w");
  fprintf(stream, "%s\n", u, v5);
  for ( i = 0; i < strlen(&t); ++i )
  {
    fseek(stream, p[i], 0);
    fputc(*(&t + p[i]), stream);
    fseek(stream, 0LL, 0);
    fprintf(stream, "%s\n", u);
  }
  fclose(stream);
  remove(filename);
  return 0;
}
```

是个生成flag并写入文件的过程
但flag文件在tmp
写入的file又remove了
所以找不到flag文件

要么动态调试
但更简单点就是根据flag的生成过程
复现就好

复现需要知道其中的几个常量
跟踪下

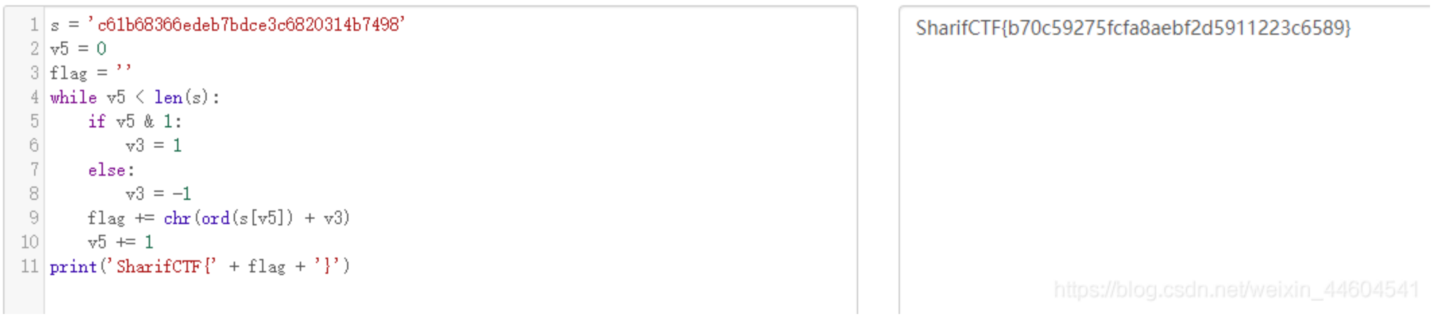


```
.data:00000000006010A0 ; char s[]
.data:00000000006010A0 s               db 'c61b68366edeb7bdce3c6820314b7498',0
.data:00000000006010A0                                         ; DATA XREF: main+25↑o
.data:00000000006010A0                                         ; main+3F↑r
.data:00000000006010C1                 align 20h
.data:00000000006010E0                 public t
.data:00000000006010E0 ; char t
.data:00000000006010E0 t               db 53h                  ; DATA XREF: main+65↑w
.data:00000000006010E0                                         ; main+C9↑o ...
.data:00000000006010E1 aHarifctf       db 'harifCTF{????????????????????????????????}',0
.data:000000000060110C                 align 20h
.data:0000000000601120                 public u
.data:0000000000601120 u               db '********************************************',0
.data:0000000000601120                                         ; DATA XREF: main+A5↑o
.data:0000000000601120                                         ; main+13F↑o
.data:000000000060114C                 align 20h
.data:0000000000601160                 public p
.data:0000000000601160 ; int p[43]
.data:0000000000601160 p               dd 1Eh                  ; DATA XREF: main+E1↑r
.data:0000000000601160                                         ; main+104↑r
```

主要是s和t

然后就是写脚本了

```python
s = 'c61b68366edeb7bdce3c6820314b7498'
v5 = 0
flag = ''
while v5 < len(s):
 if v5 & 1:
  v3 = 1
 else:
  v3 = -1
 flag += chr(ord(s[v5]) + v3)
 v5 += 1
print('SharifCTF{' + flag + '}')
```



```python
s = 'c61b68366edeb7bdce3c6820314b7498'
v5 = 0
flag = ''
while v5 < len(s):
    if v5 & 1:
        v3 = 1
    else:
        v3 = -1
    flag += chr(ord(s[v5]) + v3)
    v5 += 1
print('SharifCTF{' + flag + '}')
```

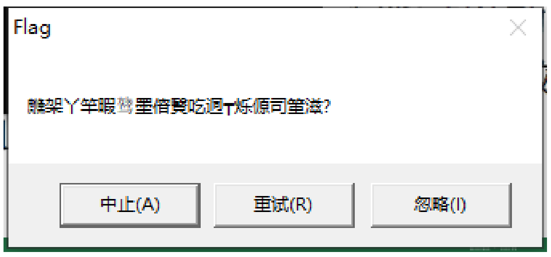
```
SharifCTF{b70c59275fcfa8aebf2d5911223c6589}
```

得到flag

11、csaw2013reversing2

下下来一个exe

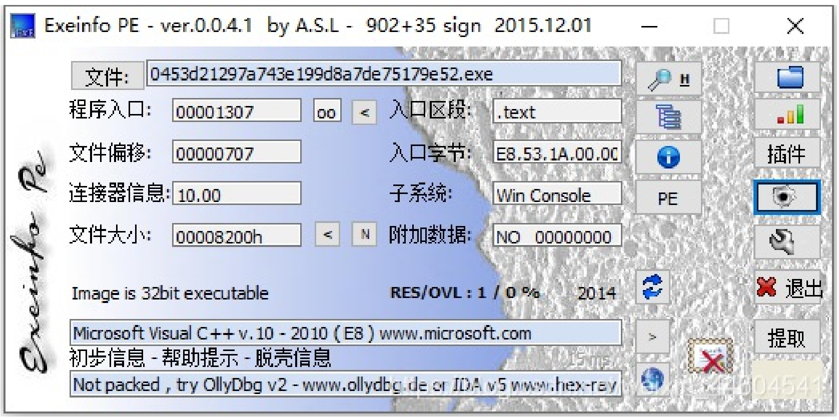
运行弹框



Flag

瞷絮丫竿暇嚩墨偫贀吃週┳烁傎司箽漥?

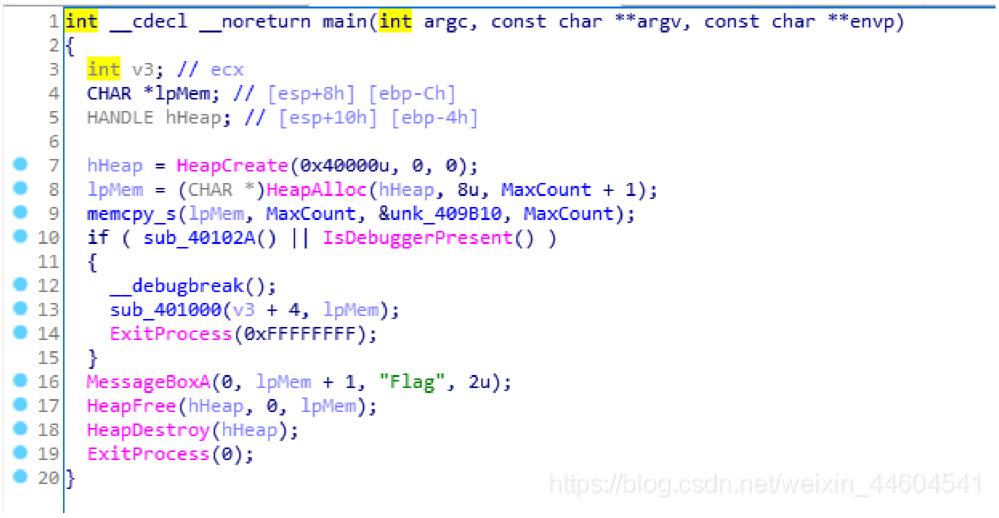
PE查壳



```
Exeinfo PE - ver.0.0.4.1  by A.S.L -  902+35 sign  2015.12.01
文件:       0453d21297a743e199d8a7de75179e52.exe
程序入口:   00001307        入口区段:  .text
文件偏移:   00000707        入口字节:  E8.53.1A.00.00
连接器信息: 10.00           子系统:    Win Console
文件大小:   00008200h       附加数据:  NO  00000000
Image is 32bit executable            RES/OVL : 1 / 0 %    2014
Microsoft Visual C++ v.10 - 2010 ( E8 ) www.microsoft.com
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-ray
```

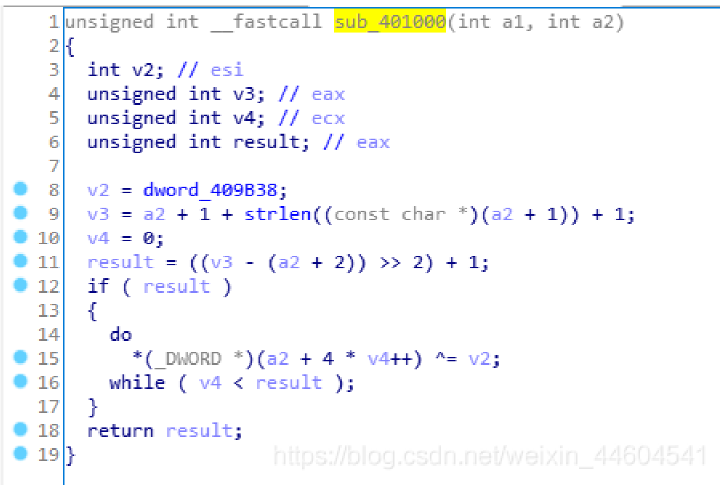
32位无壳

扔进IDA



```c
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  int v3; // ecx
  CHAR *lpMem; // [esp+8h] [ebp-Ch]
  HANDLE hHeap; // [esp+10h] [ebp-4h]

  hHeap = HeapCreate(0x40000u, 0, 0);
  lpMem = (CHAR *)HeapAlloc(hHeap, 8u, MaxCount + 1);
  memcpy_s(lpMem, MaxCount, &unk_409B10, MaxCount);
  if ( sub_40102A() || IsDebuggerPresent() )
  {
    __debugbreak();
    sub_401000(v3 + 4, lpMem);
    ExitProcess(0xFFFFFFFF);
  }
  MessageBoxA(0, lpMem + 1, "Flag", 2u);
  HeapFree(hHeap, 0, lpMem);
  HeapDestroy(hHeap);
  ExitProcess(0);
}
```

跟进 `sub_401000`

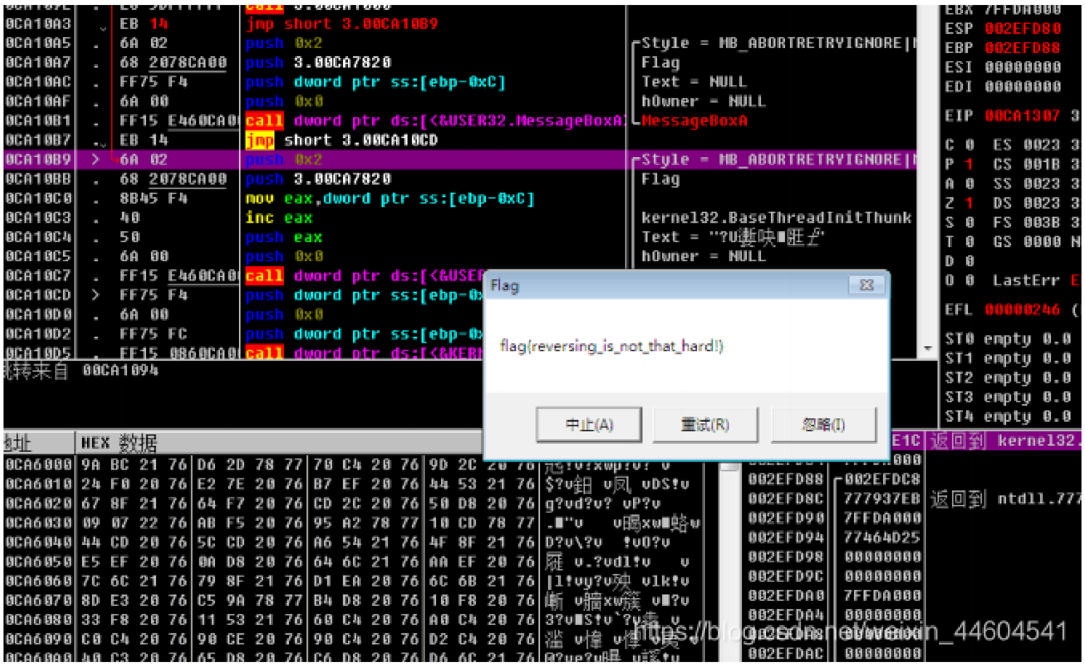


```c
unsigned int __fastcall sub_401000(int a1, int a2)
{
  int v2; // esi
  unsigned int v3; // eax
  unsigned int v4; // ecx
  unsigned int result; // eax

  v2 = dword_409B38;
  v3 = a2 + 1 + strlen((const char *)(a2 + 1)) + 1;
  v4 = 0;
  result = ((v3 - (a2 + 2)) >> 2) + 1;
  if ( result )
  {
    do
      *(_DWORD *)(a2 + 4 * v4++) ^= v2;
    while ( v4 < result );
  }
  return result;
}
```

动态调试发现程序会跳过 call 401000

扔进OD

把 `int3` 改为 `nop(0x90)` ,再跳到 `loc_4010B9` 进行输出



```
0CA10A3   .v  EB 14            jmp short 3.00CA10B9
0CA10A5   .   6A 02            push 0x2                              ┌Style = MB_ABORTRETRYIGNORE|
0CA10A7   .   68 2078CA00      push 3.00CA7820                       │Flag
0CA10AC   .   FF75 F4          push dword ptr ss:[ebp-0xC]           │Text = NULL
0CA10AF   .   6A 00            push 0x0                              │hOwner = NULL
0CA10B1   .   FF15 E460CA00    call dword ptr ds:[<&USER32.MessageBoxA] └MessageBoxA
0CA10B7   .v  EB 14            jmp short 3.00CA10CD
0CA10B9   >   6A 02            push 0x2                              ┌Style = MB_ABORTRETRYIGNORE|
0CA10BB   .   68 2078CA00      push 3.00CA7820                       │Flag
0CA10C0   .   8B45 F4          mov eax,dword ptr ss:[ebp-0xC]        │
0CA10C3   .   40               inc eax                               │kernel32.BaseThreadInitThunk
0CA10C4   .   50               push eax                              │Text = "?U蜇呋■脏£"
0CA10C5   .   6A 00            push 0x0                              │hOwner = NULL
0CA10C7   .   FF15 E460CA00    call dword ptr ds:[<&USER
0CA10CD   >   FF75 F4          push dword ptr ss:[ebp-0x
0CA10D0   .   6A 00            push 0x0
0CA10D2   .   FF75 FC          push dword ptr ss:[ebp-0x
0CA10D5   .   FF15 0860CA00    call dword ptr ds:[<&KERN
跳转来自 00CA1094

EBX 7FFDA000
ESP 002EFD80
EBP 002EFD88
ESI 00000000
EDI 00000000
EIP 00CA1307
C 0  ES 0023
P 1  CS 001B
A 0  SS 0023
Z 1  DS 0023
S 0  FS 003B
T 0  GS 0000
D 0
O 0  LastErr
EFL 00000246
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0

地址     HEX 数据
0CA6000 9A BC 21 76|D6 2D 78 77|70 C4 20 76|9D 2C 20 76
0CA6010 24 F0 20 76|E2 7E 20 76|B7 EF 20 76|44 53 21 76
0CA6020 67 8F 21 76|64 F7 20 76|CD 2C 20 76|50 D8 20 76
0CA6030 09 07 22 76|AB F5 20 76|95 A2 78 77|10 CD 78 77
0CA6040 44 CD 20 76|5C CD 20 76|A6 54 21 76|4F 8F 21 76
0CA6050 E5 EF 20 76|0A D8 20 76|64 6C 21 76|AA EF 20 76
0CA6060 7C 6C 21 76|79 8F 21 76|D1 EA 20 76|6C 6B 21 76
0CA6070 8D E3 20 76|C5 9A 78 77|B4 D8 20 76|10 F8 20 76
0CA6080 33 F8 20 76|11 53 21 76|60 C4 20 76|A0 C4 20 76
0CA6090 C0 C4 20 76|90 CE 20 76|90 C4 20 76|D2 C4 20 76

       E1C 返回到 kernel32.
002EFD88 ┌002EFDC8
002EFD8C  777937EB 返回到 ntdll.777
002EFD90  7FFDA000
002EFD94  77464D25
002EFD98  00000000
002EFD9C  00000000
002EFDA0  7FFDA000
002EFDA4  00000000
002EFDAC  00000000
```

Flag

flag{reversing_is_not_that_hard!}

中止(A)　重试(R)　忽略(I)

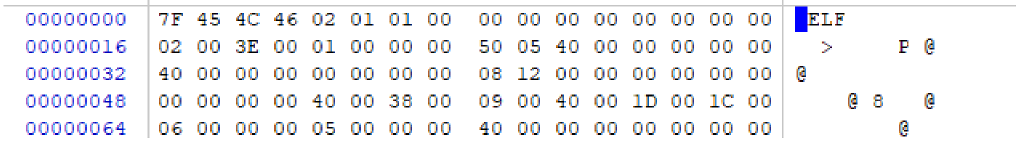
https://blog.csdn.net/weixin_44604541

得到flag

12、maze

下下来一个无后缀文件

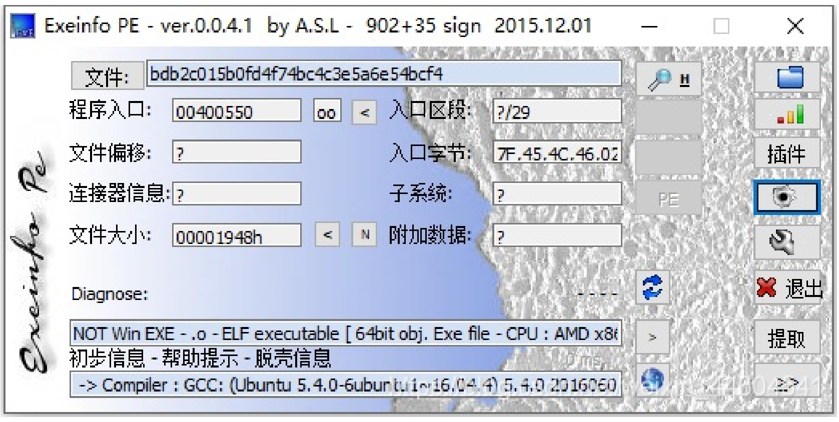
扔进winhex



```
00000000  7F 45 4C 46 02 01 01 00  00 00 00 00 00 00 00 00   ELF
00000016  02 00 3E 00 01 00 00 00  50 05 40 00 00 00 00 00     >      P @
00000032  40 00 00 00 00 00 00 00  08 12 00 00 00 00 00 00   @
00000048  00 00 00 00 40 00 38 00  09 00 40 00 1D 00 1C 00       @ 8   @
00000064  06 00 00 00 05 00 00 00  40 00 00 00 00 00 00 00           @
```

是个ELF

PE查壳

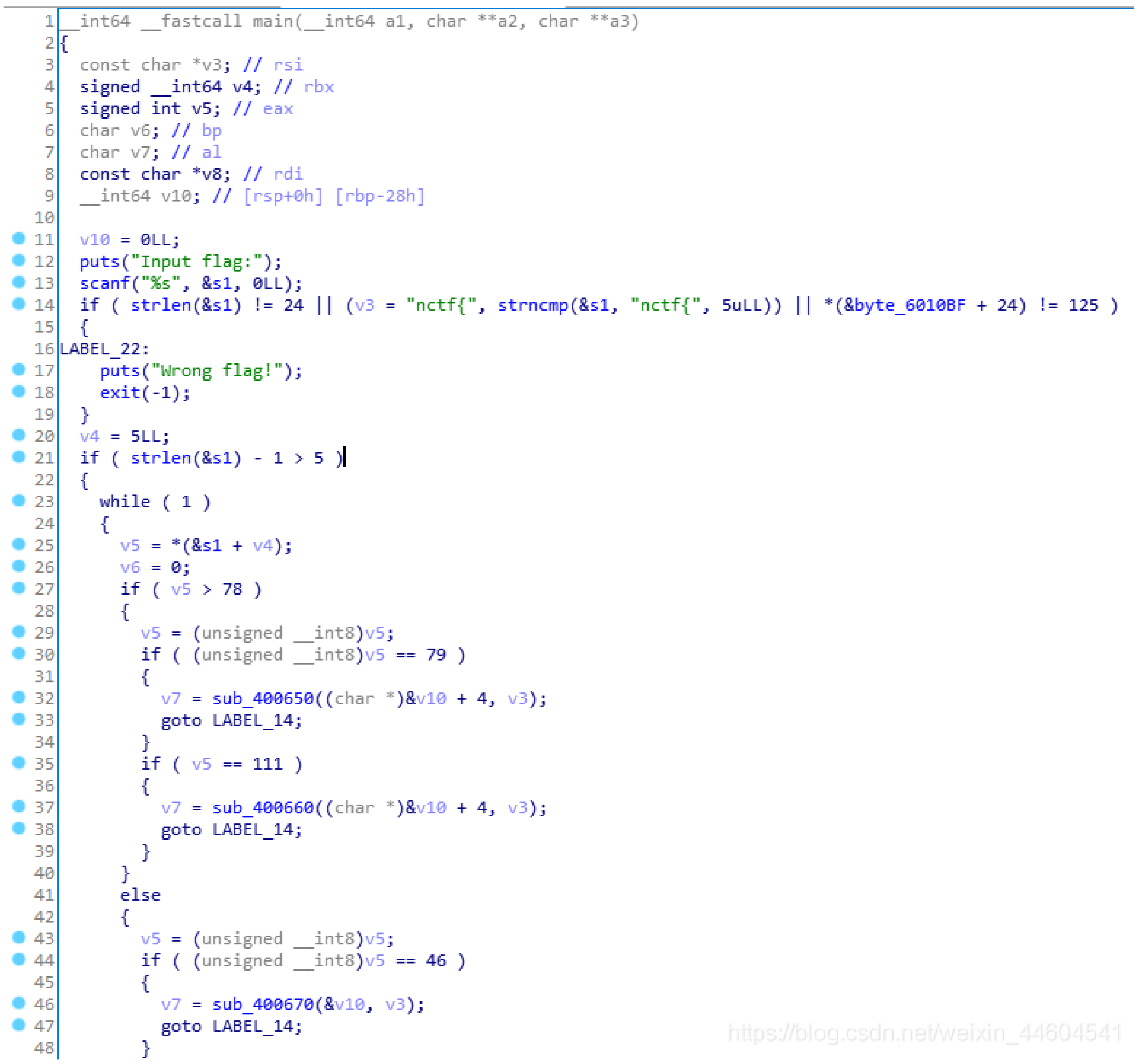


Exeinfo PE - ver.0.0.4.1 by A.S.L - 902+35 sign 2015.12.01

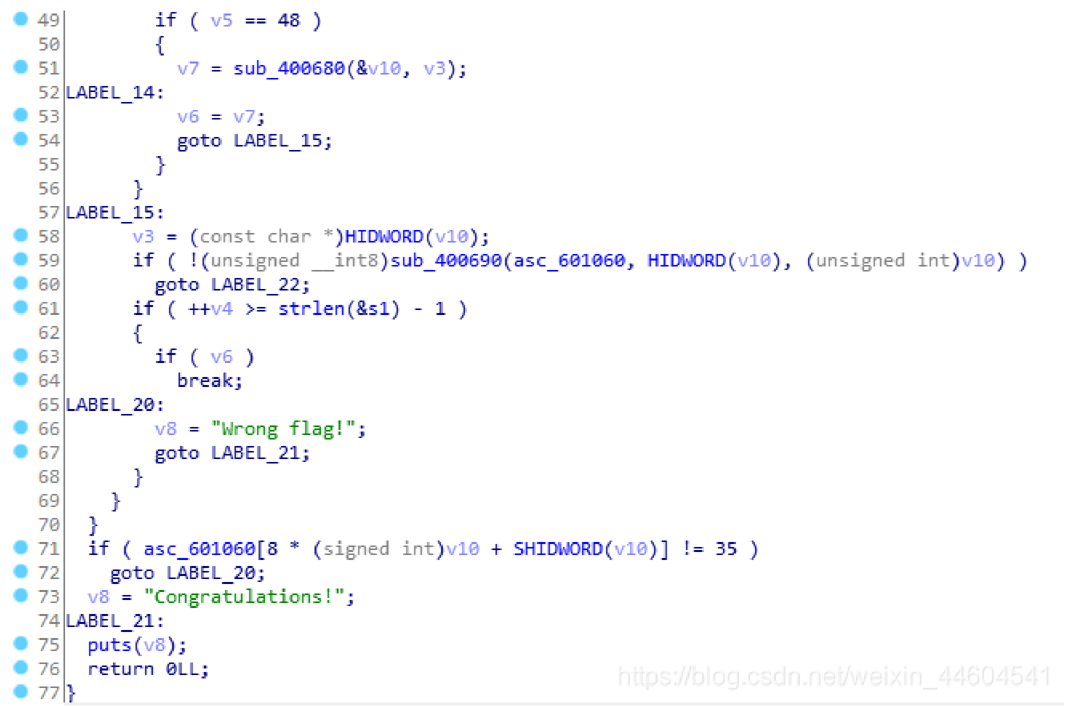
文件: bdb2c015b0fd4f74bc4c3e5a6e54bcf4
程序入口: 00400550　　入口区段: ?/29
文件偏移: ?　　入口字节: 7F.45.4C.46.02
连接器信息: ?　　子系统: ?
文件大小: 00001948h　　附加数据: ?
Diagnose:
NOT Win EXE - .o - ELF executable [64bit obj. Exe file - CPU : AMD x8
初步信息 - 帮助提示 - 脱壳信息
-> Compiler : GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609

64位

扔进IDA



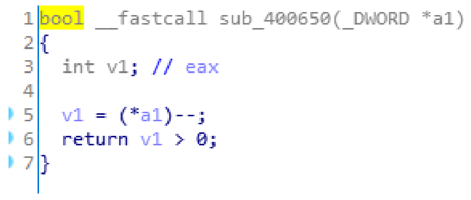
```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  const char *v3; // rsi
  signed __int64 v4; // rbx
  signed int v5; // eax
  char v6; // bp
  char v7; // al
  const char *v8; // rdi
  __int64 v10; // [rsp+0h] [rbp-28h]

  v10 = 0LL;
  puts("Input flag:");
  scanf("%s", &s1, 0LL);
  if ( strlen(&s1) != 24 || (v3 = "nctf{", strncmp(&s1, "nctf{", 5uLL)) || *(&byte_6010BF + 24) != 125 )
  {
LABEL_22:
    puts("Wrong flag!");
    exit(-1);
  }
  v4 = 5LL;
  if ( strlen(&s1) - 1 > 5 )
  {
    while ( 1 )
    {
      v5 = *(&s1 + v4);
      v6 = 0;
      if ( v5 > 78 )
      {
        v5 = (unsigned __int8)v5;
        if ( (unsigned __int8)v5 == 79 )
        {
          v7 = sub_400650((char *)&v10 + 4, v3);
          goto LABEL_14;
        }
        if ( v5 == 111 )
        {
          v7 = sub_400660((char *)&v10 + 4, v3);
          goto LABEL_14;
        }
      }
      else
      {
        v5 = (unsigned __int8)v5;
        if ( (unsigned __int8)v5 == 46 )
        {
          v7 = sub_400670(&v10, v3);
          goto LABEL_14;
        }
```

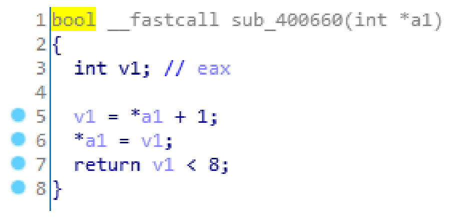


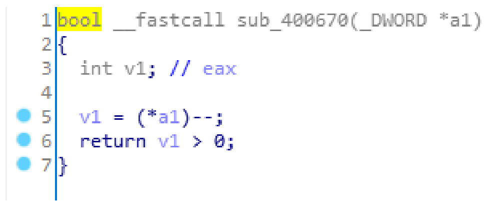
```
        if ( v5 == 48 )
        {
          v7 = sub_400680(&v10, v3);
LABEL_14:
          v6 = v7;
          goto LABEL_15;
        }
      }
LABEL_15:
      v3 = (const char *)HIDWORD(v10);
      if ( !(unsigned __int8)sub_400690(asc_601060, HIDWORD(v10), (unsigned int)v10) )
        goto LABEL_22;
      if ( ++v4 >= strlen(&s1) - 1 )
      {
        if ( v6 )
          break;
LABEL_20:
        v8 = "Wrong flag!";
        goto LABEL_21;
      }
    }
  }
  if ( asc_601060[8 * (signed int)v10 + SHIDWORD(v10)] != 35 )
    goto LABEL_20;
  v8 = "Congratulations!";
LABEL_21:
  puts(v8);
  return 0LL;
}
```

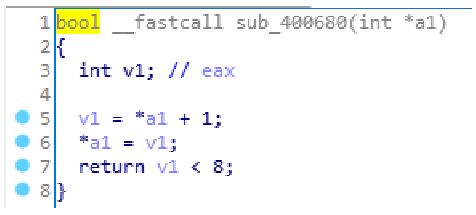
- 开头必须是`nctf{`，总长24
- 四个判断，瞅着是迷宫，Maze problem

跟踪四个sub看看


```
bool __fastcall sub_400650(_DWORD *a1)
{
  int v1; // eax

  v1 = (*a1)--;
  return v1 > 0;
}
```

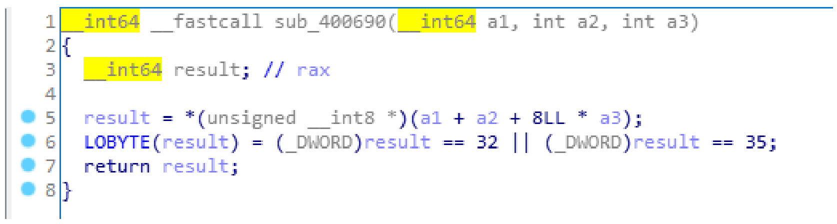

```
bool __fastcall sub_400660(int *a1)
{
  int v1; // eax

  v1 = *a1 + 1;
  *a1 = v1;
  return v1 < 8;
}
```


```
bool __fastcall sub_400670(_DWORD *a1)
{
  int v1; // eax

  v1 = (*a1)--;
  return v1 > 0;
}
```

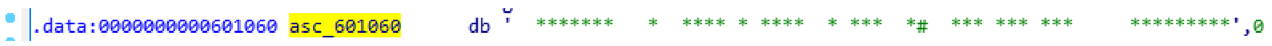

```
bool __fastcall sub_400680(int *a1)
{
  int v1; // eax

  v1 = *a1 + 1;
  *a1 = v1;
  return v1 < 8;
}
```

上下左右对应`'.','0','O','o'`

还有个边界条件


```
__int64 __fastcall sub_400690(__int64 a1, int a2, int a3)
{
  __int64 result; // rax

  result = *(unsigned __int8 *)(a1 + a2 + 8LL * a3);
  LOBYTE(result) = (_DWORD)result == 32 || (_DWORD)result == 35;
  return result;
}
```

迷宫本体


```
.data:0000000000601060 asc_601060      db '  *******   *  **** * ****  * ***  *#  *** *** ***     *********',0
```

整理下

```
'''
00111111
10001001
11101011
11001011
1001#001
11011101
11000001
11111111
'''
```

对应给出移动`o0oo00O000oooo..OO`

于是有flag：`nctf{o0oo00O000oooo..OO}`

结语

对reverse有个大体概念