XCTF社区中有关re的题目超级详细手把手的题解（附带做题实际记录）

原创

jovy-rtt 于 2019-12-06 15:00:31 发布 436 收藏 3

分类专栏： CTF 文章标签： xctf ctf

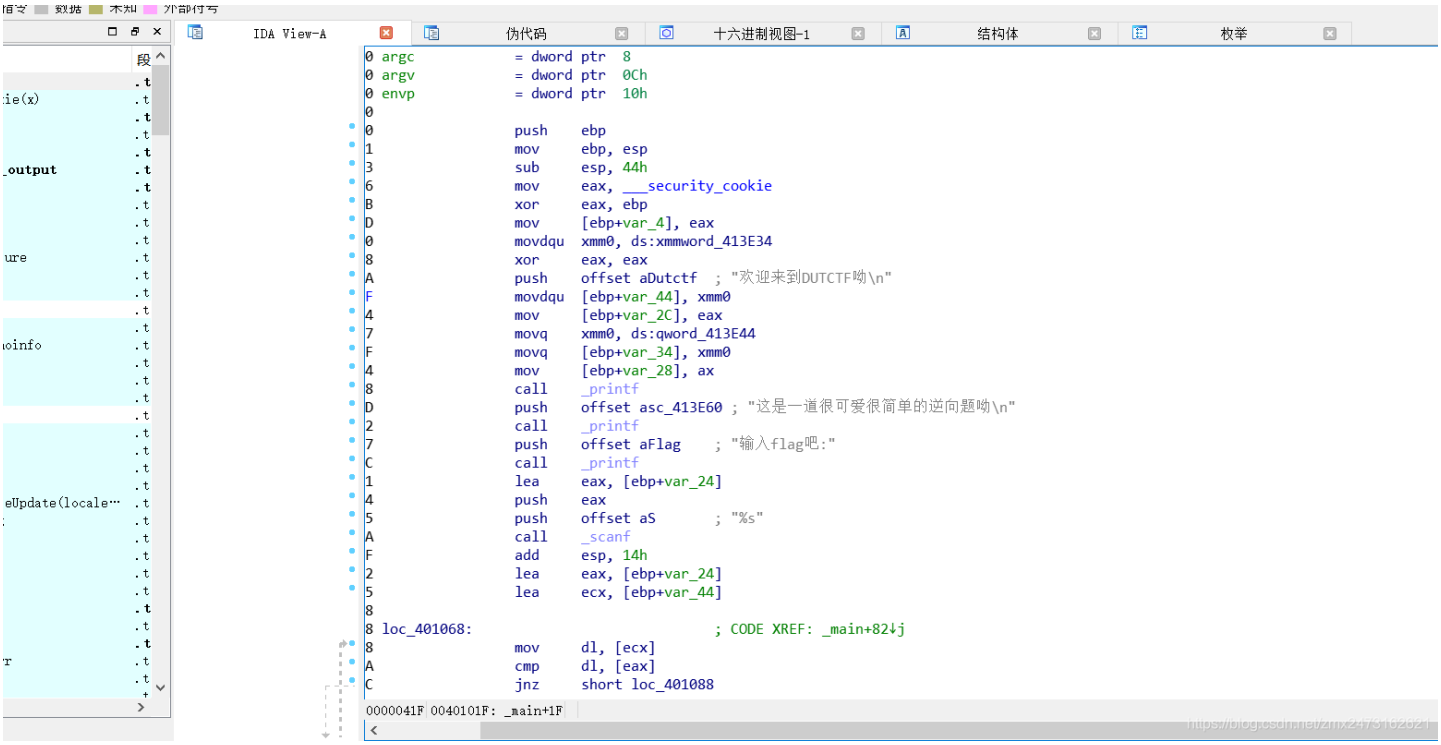
本文链接：https://blog.csdn.net/zmx2473162621/article/details/103374532

声明

PS：如果文中的一些函数看不懂，可以去我写的和本文配套的一个知识点的总结博客中去找功能介绍，还不懂的可以留言，我将与你共同分享经验，共同进步，博客链接：https://blog.csdn.net/zmx2473162621/article/details/103416005

re1

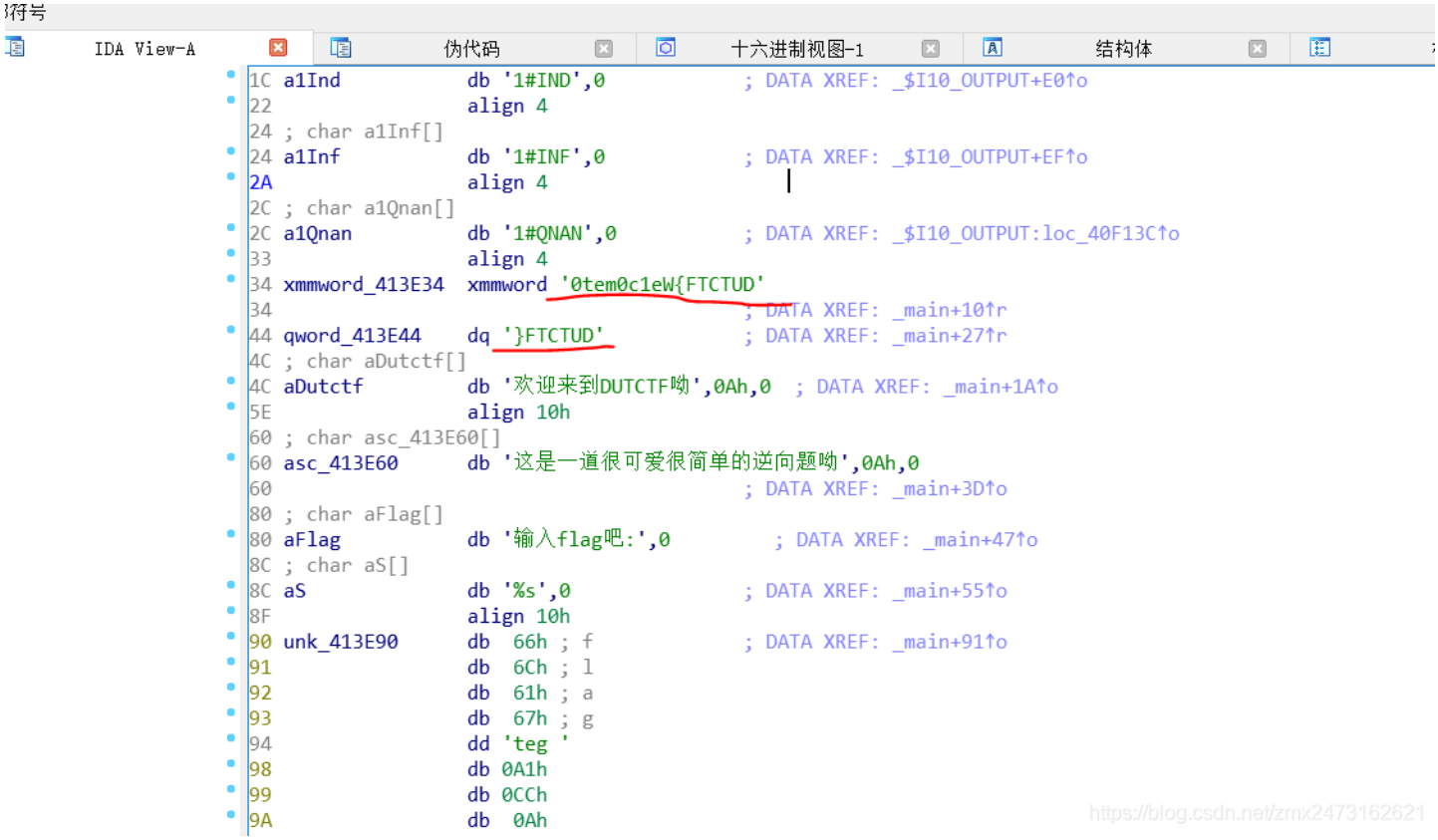
下载附件是个exe文件，由于是入门题，不需要那么多的所谓的peid，所以直接拉入ida中，找到main函数



然后f5进行反汇编，查看伪代码，

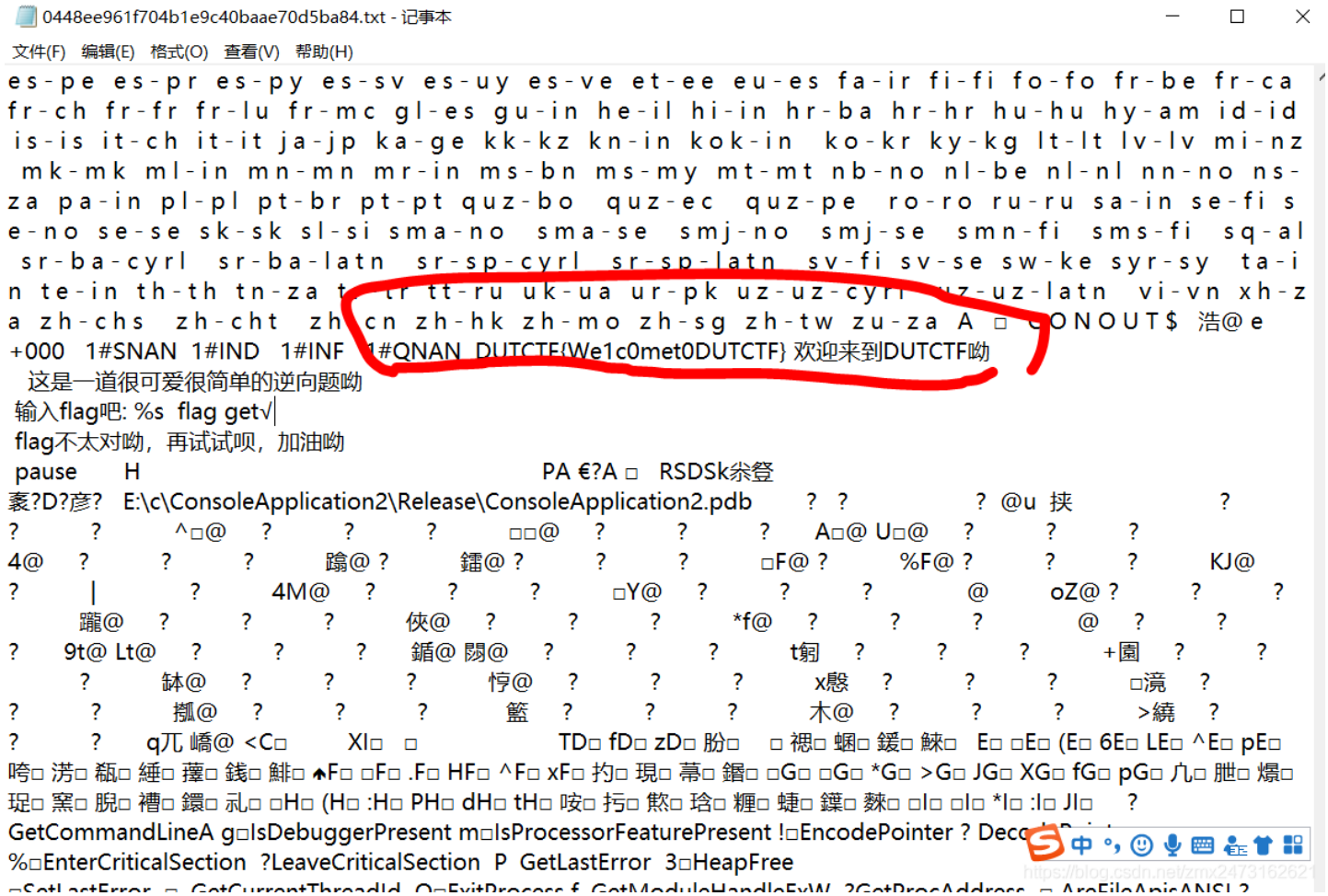
```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  __int128 v5; // [esp+0h] [ebp-44h]
  __int64 v6; // [esp+10h] [ebp-34h]
  int v7; // [esp+18h] [ebp-2Ch]
  __int16 v8; // [esp+1Ch] [ebp-28h]
  char v9; // [esp+20h] [ebp-24h]

  _mm_storeu_si128((__m128i *)&v5, _mm_loadu_si128((const __m128i *)&xmmword_413E34));
  v7 = 0;
  v6 = qword_413E44;
  v8 = 0;
  printf("欢迎来到DUTCTF呦\n");
  printf("这是一道很可爱很简单的逆向题呦\n");
  printf("输入flag吧:");
  scanf("%s", &v9);
  v3 = strcmp((const char *)&v5, &v9);
  if ( v3 )
    v3 = -(v3 < 0) | 1;
  if ( v3 )
    printf(aFlag_0);
  else
    printf((const char *)&unk_413E90);
  system("pause");
  return 0;
}
```


发现了这个东西，点进去一看



然后倒序输出就好，得出flag：DUTCTF{We1c0met0DUTCTF}

不过有一说一，，用笔记本打开，发现也有flag

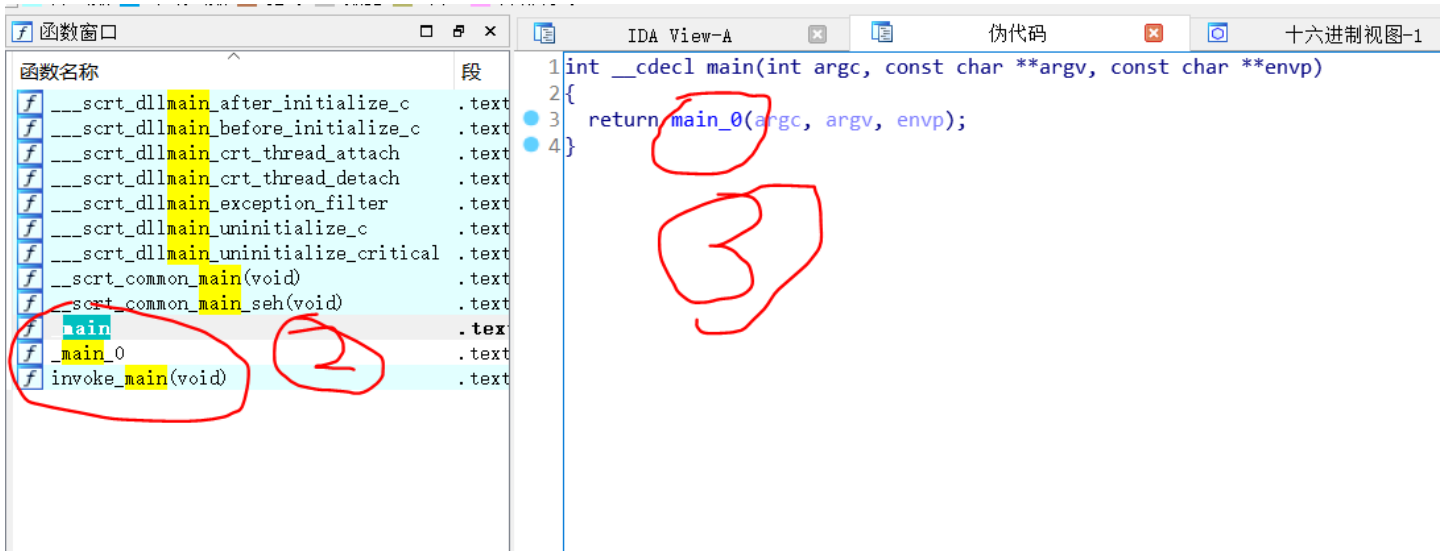


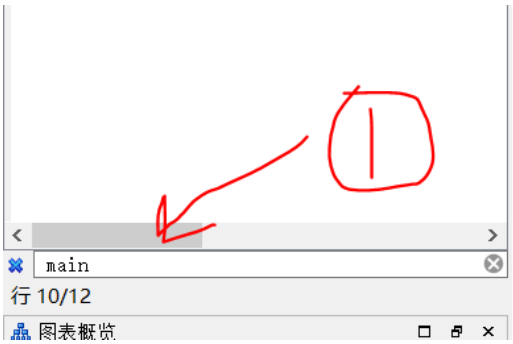
game：

说实话，这道题我确实有点蒙蒙的...按照顺序输入了1,2,3,4,5,6,7,8然后就莫名其妙的出来了flag：

done!!! the flag is zsctf{T9is_tOpic_1s_v5ry_int7resting_b6t_others_are_n0t}

大致题意是玩一个游戏（一共八个灯，打开第i个灯的时候，改变第N个灯的状态，第（N-1）和（N＋1）的状态也会改变），通关才会出flag，我刚拿到这道题的时候确实有点懵逼的感觉，也不知道改干什么，然后上一题就可以通过记事本得出来，所以我又想了一下，可不可以用记事本直接搜flag，然后查了一下，发现不能，出来的flag一共有两处单词，但后面跟着的都不是真正的flag，那么记事本不行，然后就用peid判断一下是不是有壳，发现是vc编写的...然后？？？那就把它拉入ida吧，拉入之后，发现函数有点多...真的多！！！！然后就懵逼了，搜了一下发现，是可以直接查找的（我记住了！！！），所以我就在函数窗口直接查找main主函数：





具体的查找过程是，右键函数名称，然后选中快速筛选，直接在1处直接输入main查找，然后找到_main，打开之后发现，好短...不管了，先f5...：就是上面的3；然后就发现是和main_0有关的，所以双击打开：

```
void __cdecl main_0()
{
  signed int i; // [esp+DCh] [ebp-20h]
  int v1; // [esp+F4h] [ebp-8h]

  sub_45A7BE(&unk_50B110);
  sub_45A7BE(&unk_50B158);
  sub_45A7BE(&unk_50B1A0);
  sub_45A7BE(&unk_50B1E8);
  sub_45A7BE(&unk_50B230);
  sub_45A7BE(&unk_50B278);
  sub_45A7BE(&unk_50B2C0);
  sub_45A7BE(&unk_50B308);
  sub_45A7BE("二                                                    |\n");
  sub_45A7BE("|              by 0x61                               |\n");
  sub_45A7BE("|                                                    |\n");
  sub_45A7BE("|----------------------------------------------------|\n");
  sub_45A7BE(
    "Play a game\n"
    "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_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();
    }
  }
}
20008815 main 0:34 (45B515)
```


这个代码看着就有内味了，所以我们来分析一下：
定义的全局变量先记一下…
首先，那一堆的sub就不说了，是提示词（可以进去看一下）
然后对于那些绿色的代码，发现也是游戏的提示部分，那么接着往下面看，，两个while循环，结合题意以及exe来看的话，应该是控制整个游戏的结束的，然后里面的while是为了判断输入的字符是否合法，所以这个可以不用看

if else如果是0的就reset 也是结合着exe和代码共同来看的，所以核心代码就是if下面的sub_4576D6(v1 - 1);还有下面的if（用于判断正确）下的sub_457AB4();，至于中间的那个sub，点进去之后发现好像是输入未完成的时候的提示词…
然后分析第一个if下的函数sub_4576D6(v1 - 1）（我将其命名为：label01）：

IDA View-A 伪代

```
1 bool __cdecl label01(int a1)
2 {
3   return sub_45E640(a1);
4 }
```

接着进：

IDA View-A

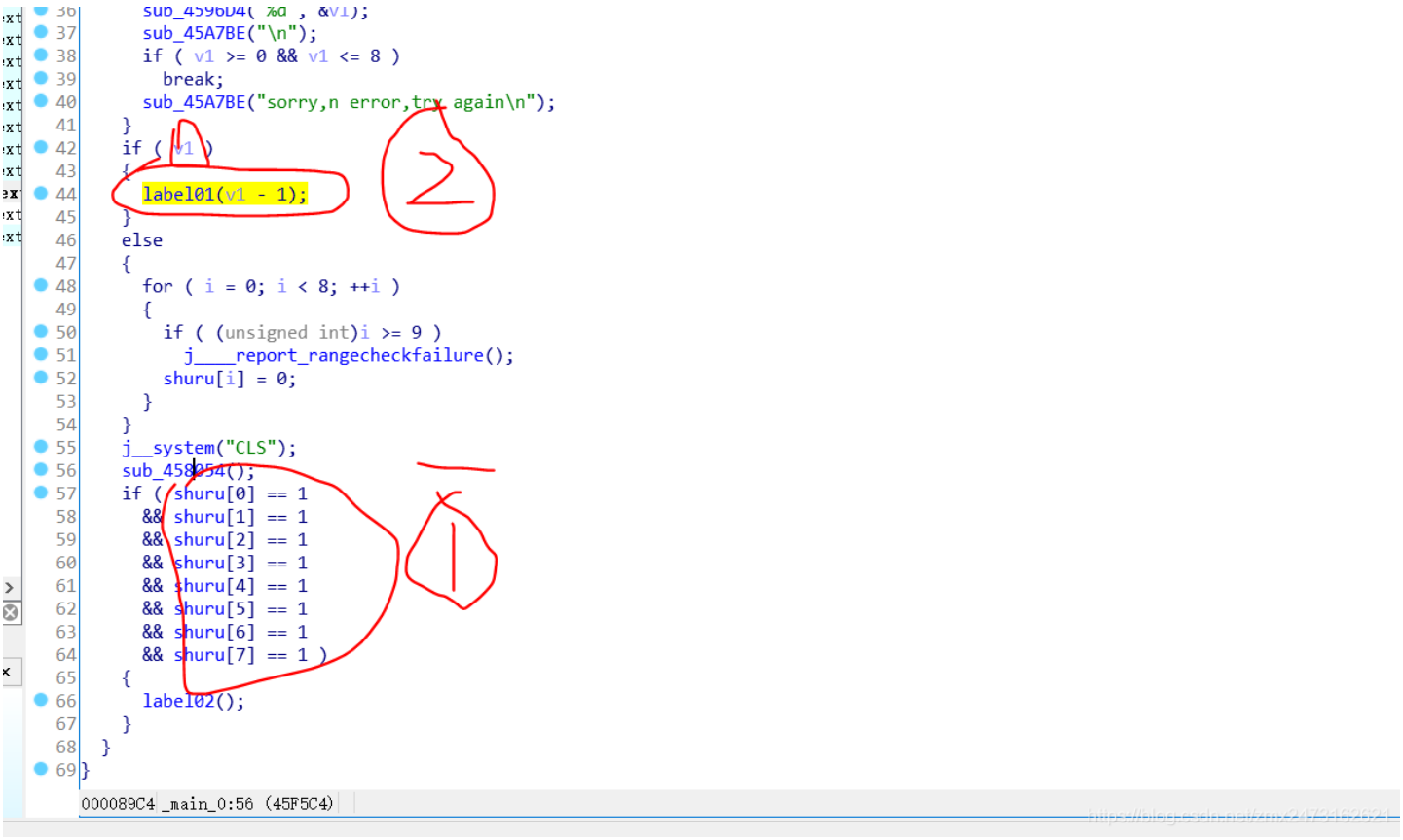
```
1  bool __cdecl sub_45E640(int a1)
2  {
3    bool result; // al
4
5    if ( a1 )
6    {
7      if ( a1 == 7 )
8      {
9        byte_532E28[7] = byte_532E28[7] == 0;
10       byte_532E27[7] = byte_532E27[7] == 0;
11       result = 1;
12       byte_532E28[0] = byte_532E28[0] == 0;
13     }
14     else
15     {
16       byte_532E28[a1] = byte_532E28[a1] == 0;
17       byte_532E27[a1] = byte_532E27[a1] == 0;
18       result = byte_532E29[a1] == 0;
19       byte_532E29[a1] = result;
20     }
21   }
22   else
23   {
24     byte_532E28[0] = byte_532E28[0] == 0;
25     byte_532E29[0] = byte_532E29[0] == 0;
26     result = 1;
27     byte_532E28[7] = byte_532E28[7] == 0;
28   }
29   return result;
30 }
```


返回的都是bool类型？？？这是什么鬼，返回的既然不是数据，而是bool类型，那么就是和判断有关的，而题中，输入数据的合法性判断了，什么时候输入正确flag也判断了，发现唯独没有输入超出数据范围等的判断，所以这个bool，我就猜是一个数据范围或者说规则更加合理一点；
接着看，if条件那么多，就是数据的判断喽，也就是输入数据的判断，到这里，解题就有两个思路叭

1.直接找到输入的数值，然后正常通关
2.分析通关后发生的事件，然后得出flag
（好像知道可以这样）3.在od下，暴力破解，就是把一些关键的位置，跳转直接nop（空指令），然后最后无条件的跳转到结果那里

- 第一种思路

首先第一种思路：遵守游戏规则，按输入来：
分析输入的数应该是什么：



```
sub_4596B4("%d", &v1);
sub_45A7BE("\n");
if ( v1 >= 0 && v1 <= 8 )
  break;
sub_45A7BE("sorry,n error,try again\n");
}
if ( v1 )
{
label01(v1 - 1);
}
else
{
for ( i = 0; i < 8; ++i )
{
  if ( (unsigned int)i >= 9 )
    j____report_rangecheckfailure();
  shuru[i] = 0;
}
}
j__system("CLS");
sub_458054();
if ( shuru[0] == 1
  && shuru[1] == 1
  && shuru[2] == 1
  && shuru[3] == 1
  && shuru[4] == 1
  && shuru[5] == 1
  && shuru[6] == 1
  && shuru[7] == 1 )
{
  label02();
}
}
}
000089C4 _main_0:56 (45F5C4)
```

为了读的舒服，把函数变量改一下名字，按键盘上的n，然后改为shuru，（其他函数改法一样），点入label01（处理函数，点入进去）：

```
bool __cdecl sub_45E640(int a1)
{
  bool result; // al 就是输入的数值-1

  if ( a1 )
  {
    if ( a1 == 7 )                          // 发现输入 8 和 1结果好像是一样的
    {
      shuru[7] = shuru[7] == 0;             // 分析后发现，只要不输入重复的数据，就可以使其变成全1
      aaaaa[7] = aaaaa[7] == 0;
      result = 1;
      shuru[0] = shuru[0] == 0;
    }
    else
    {
      shuru[a1] = shuru[a1] == 0;
      aaaaa[a1] = aaaaa[a1] == 0;
      result = bbbbb[a1] == 0;
      bbbbb[a1] = result;
    }
  }
  else
  {
    shuru[0] = shuru[0] == 0;
    bbbbb[0] = bbbbb[0] == 0;
    result = 1;
    shuru[7] = shuru[7] == 0;
  }
  return result;
}
```


然后分析发现，只要输入的数字不一样，就可以直接过也就是把所有数字输入完，相同的数字输入奇数个（你只一你细细的品）

然后分析发现，只要输入的数字不一样，就可以直接过也就是把所有数字输入完，相同的数字输入可数作（你品，你细细的品）
所以到这里就是第一种思路了

- 第二种思路

：不管输入的是什么，直接分析出来我们想要的答案：

```
  v37 = 119;
  v38 = 96;
  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");
}
```


也就是，从v2开始的数，对59开始的数进行异或，然后再次异或13h，最后输出：
那么写出一个python：
pyhton代码链接：https://pan.baidu.com/s/1_pxbXvhalydXl9MEb285bQ
提取码：z66s

```
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]              #手打确实难顶
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
flag = ''
while(i<56):
    a[i]^=b[i]
    a[i]^=19
    flag+=chr(a[i])
    i+=1
print(flag)
```

然后就得出了flag

- 第三种思路
  我知道有这个思路，但是现在还不会，呜呜呜，过几天我问问大佬，先放放...

Hello，CTF

有关这一题的话，看题目的提示，是一个密文flag，不是明文flag，所以刚开始有点慌，对于这些了解的确实不是太多...不多说，我还是先...运行一些这个程序，发现是只要输入flag正确就可以，那么这种简单的功能，就先用笔记本打开了一下，

```
MZ? □  □        ?     @                    ?  □? ???L?This program cannot be run in DOS mode.

$        9呖橄欖橄欖橄欖寧歕嬇糰?爓尟橄欖寧歖寠橄欖橄糈寬橄仜  嬇糰學  嬇糰孳ich考W?            PE  L□□ ?[      ? □□□□□ `  P
|□  □  p   @ □  □ □     □     ?  □    □    □ □   □ □    □       蓋  (
  p  ?                  .text   錌  □  `  □          `.rdata  ?  p  □  p          @  @.data  ?  €  0  €          @  ?
```

先发现这个，然后用ctrl+f查了一下flag字符串，发现并没有，所以就往下手动查找一下：

```
          )□@ 諷@        ?@              wrong!
success!
  %x  %s  please input your serial:   437261636b4d654a757374466f7246756e      €甪     €甪 □□          □                    □


 ?@ □
 0q@  q@           -
]  ]     □  ?       ?     ?  ?     ?  ?     ?  ?     ?  ?     ?  ?     ?  ?     ?  ?      □  □
  ?  □   □?              蘚@ □   爂@        ts@
  Ps@ □   $s@ □   魆@ □   衧@ □     @ □   lr@ □   Dr@ □   r@ □   詫@    琿@ x   渜@ y   寀@ z   |q@ ?   xq@       hq@ □
□          □       □  □  □
  □           □  ♠  □  ♠           ♠
  □  □  □  ♠  □
  □  □       □
  □  □  □      !
  5       A
  C       P  □  R
  c
```


就发现了，success后面是有一串数字的，看着有些像十六进制的数字串，437261636b4d654a757374466f7246756e 先放着
然后peid脱一下壳，之后把他拉进去ida中，找到main主函数，然后直接f5：

IDA View-A | 伪代码 | 十六进制视图-1 | 结构体 | 枚举

```
  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("please input your serial:", v6);
    scanf("%s", v9);
    if ( strlen(v9) > 0x11 )
      break;
    v3 = 0;
    do
    {
      v4 = v9[v3];
      if ( !v4 )
        break;
      sprintf(&v8, "%x", v4);
      strcat(&v10, &v8);
      ++v3;
    }
    while ( v3 < 17 );
    if ( !strcmp(&v10, &v13) )
      sub_40134B("success!\n", v7);
```

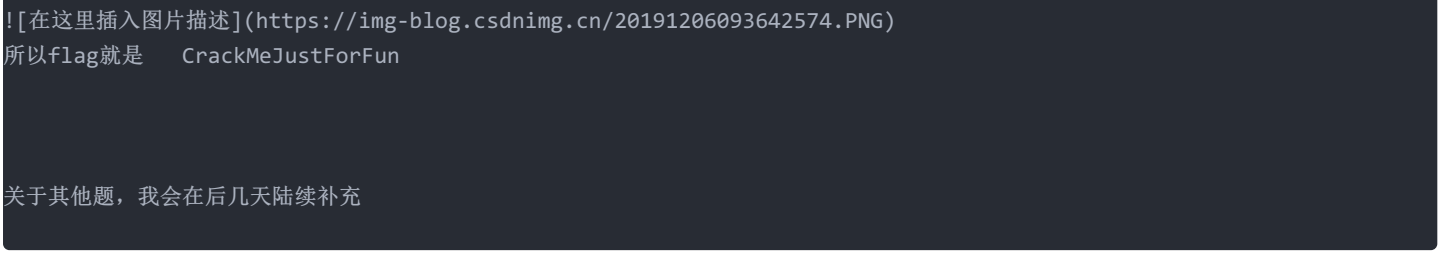
```
  else
    sub_40134B("wrong!\n", v7);
}
sub_40134B("wrong!\n", v7);
result = stru_408090._cnt-- - 1;
```

00001034 _main:23 (401034)

分析一下，把那一串数字放到了以v13为首地址的一个空间中，然后输入是v9，判断v9长度这些就是对于输入的规范,核心代码应该是do…while里面的，然后对于输入的每一个字符，赋值给v4，然后把v4格式化(sprintf … 变成十六进制)到v8中（这里如果看不懂，可以参考我写的有关这题的知识点，里面有sprintf和strcat函数的功能介绍，链接：https://blog.csdn.net/zmx2473162621/article/details/103416005），然后就拼接到v10（可看做flag+=v8）上，然后循环到结束，而判断的过程就是判断输入的字符串，转化为十六进制后是不是v13的那个数字，也正印证了用笔记本打开后看见那串数字的猜想；

知道了这么多，就可以通过py来得到这个字符串；

```
s="437261636b4d654a757374466f7246756e"
flag=bytes.fromhex(s)
print(flag)

``
```



所以flag就是　CrackMeJustForFun

关于其他题，我会在后几天陆续补充