BUUCTF-Reverse模块练习WP

原创

Weird0_ 于 2021-01-18 02:33:06 发布 302 收藏 2

分类专栏： Reverse

本文链接：https://blog.csdn.net/Sanctuary1307/article/details/112760851

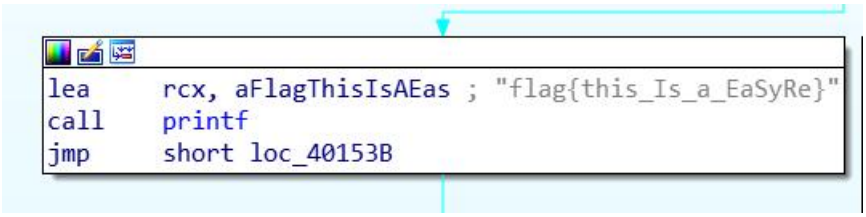
个人做re题练习笔记

BUUCTF-reverse模块：https://buuoj.cn/challenges

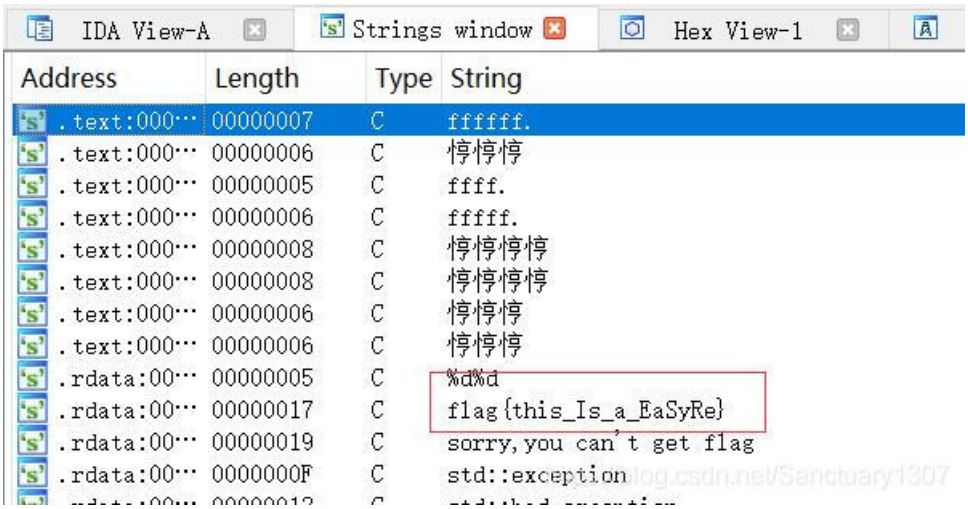
工具：IDA Pro7.0、Exeinfo PE、apkIDE等

easyre

将exe文件直接拖入IDA打开，流程图中一眼就能看到flag。



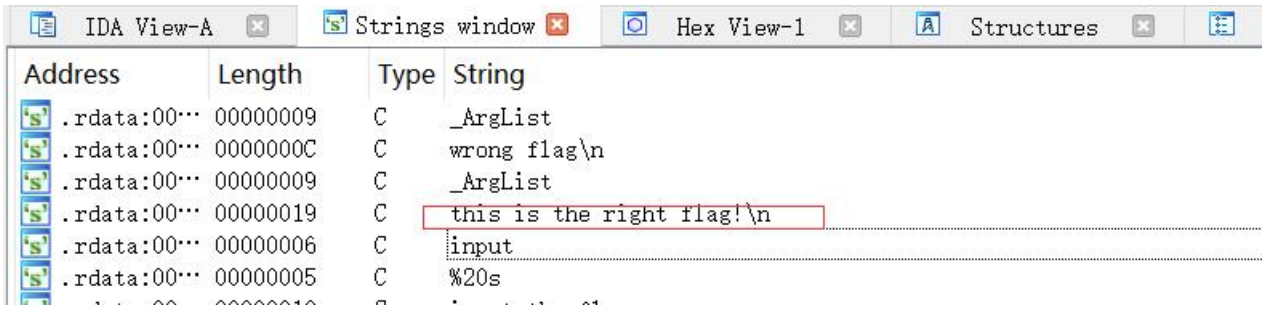
或者用shift+F12快捷键，打开IDA的strings窗口，也能直观看到flag。



`flag{this_Is_a_EaSyRe}`

reverse1

用IDA打开reverse_1.exe文件。通过Shift+F12快捷键查看Strings，其中一句是“this is the right flag!”

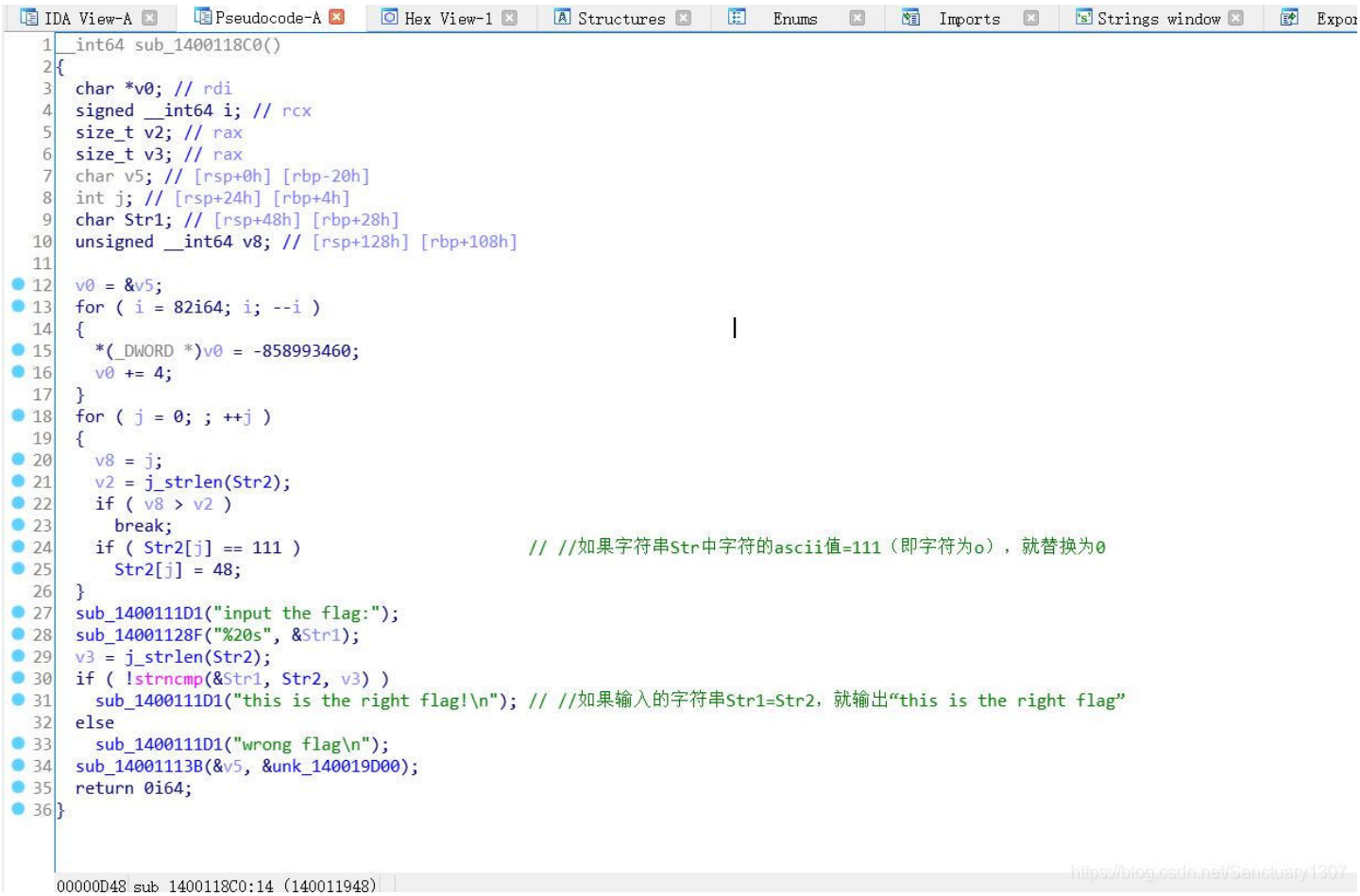


双击这个字符串，定位到对应位置

```
data:0000000140019C90 aThisIsTheRight db 'this is the right flag!',0Ah,0
data:0000000140019C90                                         ; DATA XREF: sub_1400118C0:loc_140011996↑o
                      test    eax, eax
                      jz      short loc_140011996
                      lea     rcx, aWrongFlag ; "wrong flag\n"
                      call    sub_1400111D1
                      jmp     short loc_1400119A2
; ---------------------------------------------------------------------------

loc_140011996:                          ; CODE XREF: sub_1400118C0+C6↑j
                      lea     rcx, aThisIsTheRight ; "this is the right flag!\n"
                      call    sub_1400111D1
```

字符串后面会注释有DATA XREF的字样，这是程序中引用到该字符串的代码片断的地址。双击定位到流程图

单击call之后跟着的关键函数，按F5可将选择的函数逆向生成C伪代码



分析C伪代码，可知flag为将字符串Str2中的所有o字符替换为0字符得到的。

双击伪代码页面的Str2，可知Str2为`{hello_world}`

因此flag为`flag{hell0_w0rld}`

reverse2

前几个步骤和reverse1差不多，都是将文件拖入IDA，通过关键字符串定位关键函数，F5获取C伪代码。

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int result; // eax
  int stat_loc; // [rsp+4h] [rbp-3Ch]
  int i; // [rsp+8h] [rbp-38h]
  __pid_t pid; // [rsp+Ch] [rbp-34h]
  char s2; // [rsp+10h] [rbp-30h]
  unsigned __int64 v8; // [rsp+28h] [rbp-18h]

  v8 = __readfsqword(0x28u);
  pid = fork();
  if ( pid )
  {
    argv = (const char **)&stat_loc;
    waitpid(pid, &stat_loc, 0);
  }
  else
  {
    for ( i = 0; i <= strlen(&flag); ++i )
    {
      if ( *(&flag + i) == 105 || *(&flag + i) == 114 )// 105='i'   114='r'    49='1'
                                                       // 将flag字符串中的i和r替换为1
        *(&flag + i) = 49;
    }
  }
  printf("input the flag:", argv);
  __isoc99_scanf("%20s", &s2);
  if ( !strcmp(&flag, &s2) )
    result = puts("this is the right flag!");
  else
    result = puts("wrong flag!");
  return result;
}
```


简单分析可知，flag由原flag字串中的i和r替换为1得。

原flag:`flag{hacking_for_fun}`

```
.data:0000000000601080 ; char flag
.data:0000000000601080 flag            db 7Bh                  ; DATA XREF: main+34↑r
.data:0000000000601080                                         ; main+44↑r ...
.data:0000000000601081 aHackingForFun  db 'hacking_for_fun}',0
.data:0000000000601081 _data           ends
```

最终flag：`flag{hack1ng_fo1_fun}`

内涵的软件

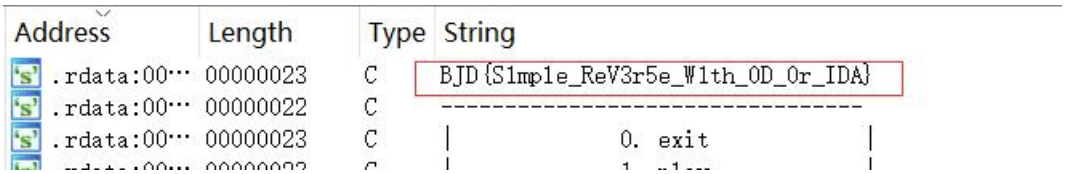
这道题直接显示了flag，把flag字符串替换为flag{}包裹的字串就可以了。

```
.rdata:00… 00000013  C  输入错误,没有提示.
.rdata:00… 0000000A  C  那没办法了
.rdata:00… 00000025  C  OD吾爱破解或者IDA这些逆向软件都挺好的
.rdata:00… 00000053  C  \n\n\n这里本来应该是答案的,但是粗心的程序员忘记把变量写进来了,你…
.rdata:00… 00000014  C  距离出现答案还有%d秒
.rdata:00… 00000028  C  DBAPP{49d3c93df25caad81232130f3d2ebfad}
.rdata:00… 00000008  C  scanf.c
```

`flag{49d3c93df25caad81232130f3d2ebfad}`

[BJDCTF 2nd]guessgame

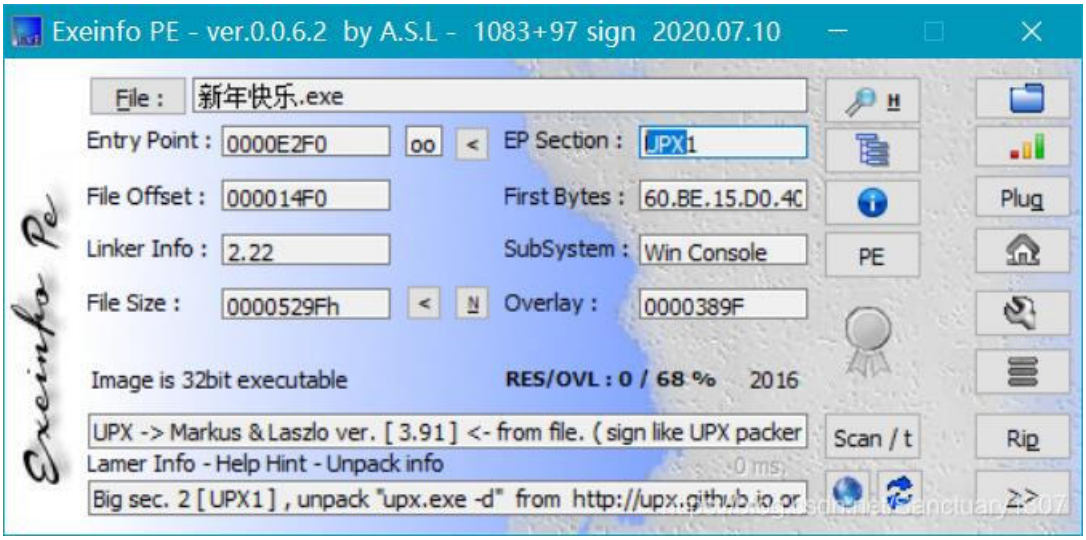
一样是在strings窗口直接显示了flag。



`flag{S1mple_ReV3r5e_W1th_0D_0r_IDA}`

新年快乐

通过Exeinfo PE查看，有upx壳



使用工具脱壳。下载地址：https://down.52pojie.cn/?query=UPX

```
F:\TOOL\upx-3.96-win64>upx -d F:\Download\Firefox\新年快乐.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2020
UPX 3.96w       Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     27807 <-     21151   76.06%    win32/pe     新年快乐.exe

Unpacked 1 file.
```


脱壳后用IDA打开

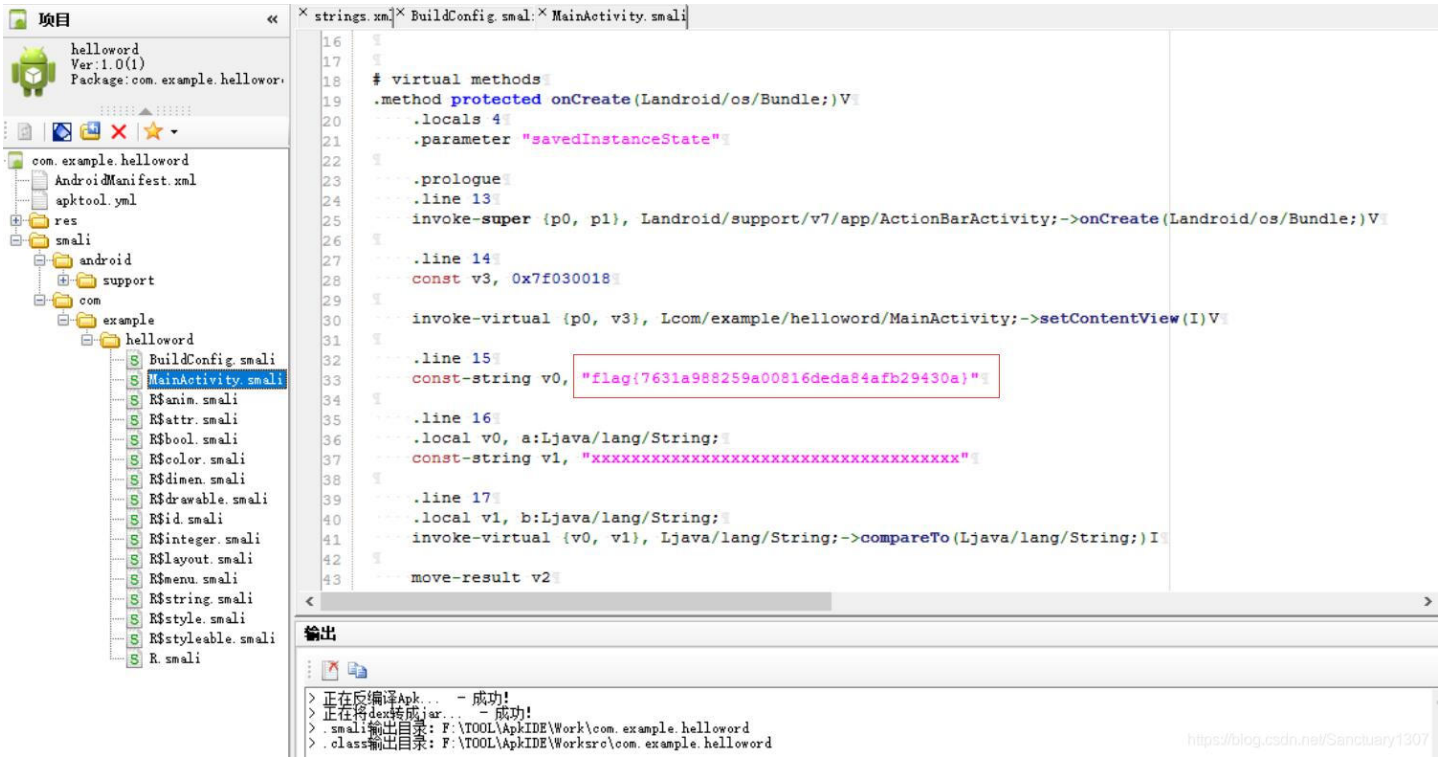
```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int result; // eax
  char v4; // [esp+12h] [ebp-3Ah]
  __int16 v5; // [esp+20h] [ebp-2Ch]
  __int16 v6; // [esp+22h] [ebp-2Ah]

  __main();
  strcpy(&v4, "HappyNewYear!");
  v5 = 0;
  memset(&v6, 0, 0x1Eu);
  printf("please input the true flag:");
  scanf("%s", &v5);
  if ( !strncmp((const char *)&v5, &v4, strlen(&v4)) )
    result = puts("this is true flag!");
  else
    result = puts("wrong!");
  return result;
}
```


分析可知flag即为v4字符串

`flag{HappyNewYear!}`

helloword

用APK改之理（即APKIDE）打开apk文件，找到helloworld路径下的MainActivity，即可得flag



flag为`flag{7631a988259a00816deda84afb29430a}`

xor

用IDA打开，F5获得C伪代码

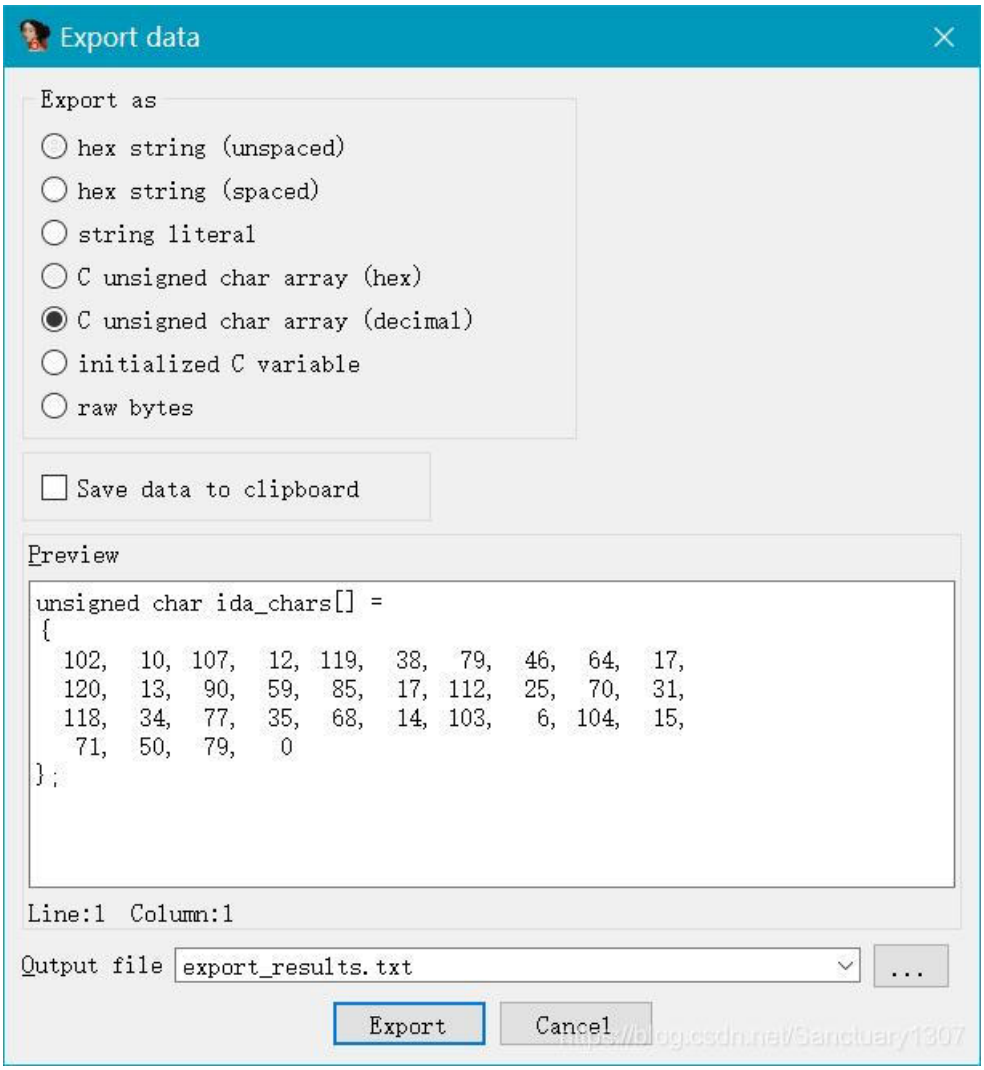
```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char *v3; // rsi
  int result; // eax
  signed int i; // [rsp+2Ch] [rbp-124h]
  char v6[264]; // [rsp+40h] [rbp-110h]
  __int64 v7; // [rsp+148h] [rbp-8h]

  memset(v6, 0, 0x100uLL);
  v3 = (char *)256;
  printf("Input your flag:\n", 0LL);
  get_line(v6, 256LL);
  if ( strlen(v6) != 33 )                       // flag长度为33
    goto LABEL_12;
  for ( i = 1; i < 33; ++i )
    v6[i] ^= v6[i - 1];                         // 将字符串内每连续的两个字符进行异或
  v3 = global;
  if ( !strncmp(v6, global, 0x21uLL) )          // 处理过的flag即global字串
    printf("Success", v3);
  else
LABEL_12:
    printf("Failed", v3);
  result = __stack_chk_guard;
  if ( __stack_chk_guard == v7 )
    result = 0;
  return result;
}
```


简单分析可知：flag长度为33

global字串是flag每连续两个字符进行异或得到的

```
_cstring:0000000100000F6E aFKWOXZUPFVMDGH db 'f',0Ah              ; DATA XREF: __data:_global↓o
_cstring:0000000100000F6E                 db 'k',0Ch,'w&O.@',11h,'x',0Dh,'Z;U',11h,'p',19h,'F',1Fh,'v"M#D',0Eh,'g'
_cstring:0000000100000F6E                 db 6,'h',0Fh,'G2O',0
_cstring:0000000100000F90 aInputYourFlag  db 'Input your flag:',0Ah,0
```

定位到global，通过Shift+E快捷键可得到global字串数组为




```
unsigned char ida_chars[] =
{
  102,  10, 107,  12, 119,  38,  79,  46,  64,  17,
  120,  13,  90,  59,  85,  17, 112,  25,  70,  31,
  118,  34,  77,  35,  68,  14, 103,   6, 104,  15,
   71,  50,  79,   0
};
```

编写python脚本

```
l = [102, 10, 107, 12, 119, 38, 79, 46, 64, 17,
120, 13, 90, 59, 85, 17, 112, 25, 70, 31,
118, 34, 77, 35, 68, 14, 103, 6, 104, 15, 71, 50, 79, 0]
flag = ''
for i in range(len(l)-1):
    flag += chr(l[i] ^ l[i-1])
print(flag)
```

xor.py

```python
l = [102, 10, 107, 12, 119, 38, 79, 46, 64, 17,
120, 13, 90, 59, 85, 17, 112, 25, 70, 31,
118, 34, 77, 35, 68, 14, 103, 6, 104, 15, 71, 50, 79, 0]
flag = ''
for i in range(len(l)-1):
    flag += chr(l[i] ^ l[i-1])
print(flag)
```

运行脚本：xor.py

```
flag{QianQiuWanDai_YiTongJiangHu}
>>>

---------- FINISHED ----------
exit code: 2 status: 0
```


运行脚本得到 `flag{QianQiuWanDai_YiTongJiangHu}`

reverse3

在流程图中找到主函数，F5获取C伪代码，关键位置如红框圈住部分

```
__int64 __cdecl main_0()
{
  int v0; // eax
  const char *v1; // eax
  size_t v2; // eax
  int v3; // edx
  __int64 v4; // ST08_8
  signed int j; // [esp+DCh] [ebp-ACh]
  signed int i; // [esp+E8h] [ebp-A0h]
  signed int v8; // [esp+E8h] [ebp-A0h]
  char Dest[108]; // [esp+F4h] [ebp-94h]
  char Str; // [esp+160h] [ebp-28h]
  char v11; // [esp+17Ch] [ebp-Ch]

  for ( i = 0; i < 100; ++i )
  {
    if ( (unsigned int)i >= 0x64 )
      j____report_rangecheckfailure();
    Dest[i] = 0;
  }
  sub_41132F("please enter the flag:");
  sub_411375("%20s", &Str);
  v0 = j_strlen(&Str);
  v1 = (const char *)sub_4110BE((int)&Str, v0, (int)&v11);
  strncpy(Dest, v1, 0x28u);
  v8 = j_strlen(Dest);
  for ( j = 0; j < v8; ++j )
    Dest[j] += j;
  v2 = j_strlen(Dest);
  if ( !strncmp(Dest, Str2, v2) )
    sub_41132F("rigth flag!\n");
  else
    sub_41132F("wrong flag!\n");
  HIDWORD(v4) = v3;
  LODWORD(v4) = 0;
  return v4;
}
```


输入的真正flag字符串Str经过函数 **sub_4110BE()** 处理后为Dest字符串，又将Dest字串的每一位字符值加上其字符位置即为Str2字符串。

```
.data:0041A034 ; char Str2[]
.data:0041A034 Str2            db 'e3nifIH9b_C@n@dH',0 ; DATA XREF: _main_0+142↑o
```

双击可知Str2字串内容为`e3nifIH9b_C@n@dH`

双击查看 **sub_4110BE()** 函数内容，如下

```
void *__cdecl sub_411AB0(char *a1, unsigned int a2, int *a3)
{
  int v4; // STE0_4
  int v5; // STE0_4
  int v6; // STE0_4
  int v7; // [esp+D4h] [ebp-38h]
  signed int i; // [esp+E0h] [ebp-2Ch]
  unsigned int v9; // [esp+ECh] [ebp-20h]
  int v10; // [esp+ECh] [ebp-20h]
  signed int v11; // [esp+ECh] [ebp-20h]
  void *Dst; // [esp+F8h] [ebp-14h]
  char *v13; // [esp+104h] [ebp-8h]

  if ( !a1 || !a2 )
    return 0;
```

```
  v9 = a2 / 3;
  if ( (signed int)(a2 / 3) % 3 )
    ++v9;
  v10 = 4 * v9;
  *a3 = v10;
  Dst = malloc(v10 + 1);
  if ( !Dst )
    return 0;
  j_memset(Dst, 0, v10 + 1);
  v13 = a1;
  v11 = a2;
  v7 = 0;
  while ( v11 > 0 )
  {
    byte_41A144[2] = 0;
    byte_41A144[1] = 0;
    byte_41A144[0] = 0;
    for ( i = 0; i < 3 && v11 >= 1; ++i )
    {
      byte_41A144[i] = *v13;
      --v11;
      ++v13;
    }
    if ( !i )
      break;
    switch ( i )
    {
      case 1:
        *((_BYTE *)Dst + v7) = aAbcdefghijklmn[(signed int)(unsigned __int8)byte_41A144[0] >> 2];
        v4 = v7 + 1;
        *((_BYTE *)Dst + v4++) = aAbcdefghijklmn[((byte_41A144[1] & 0xF0) >> 4) | 16 * (byte_41A144[0] & 3)];
        *((_BYTE *)Dst + v4++) = aAbcdefghijklmn[64];
        *((_BYTE *)Dst + v4) = aAbcdefghijklmn[64];
        v7 = v4 + 1;
        break;
      case 2:
        *((_BYTE *)Dst + v7) = aAbcdefghijklmn[(signed int)(unsigned __int8)byte_41A144[0] >> 2];
        v5 = v7 + 1;
        *((_BYTE *)Dst + v5++) = aAbcdefghijklmn[((byte_41A144[1] & 0xF0) >> 4) | 16 * (byte_41A144[0] & 3)];
        *((_BYTE *)Dst + v5++) = aAbcdefghijklmn[((byte_41A144[2] & 0xC0) >> 6) | 4 * (byte_41A144[1] & 0xF)];
        *((_BYTE *)Dst + v5) = aAbcdefghijklmn[64];
        v7 = v5 + 1;
        break;
      case 3:
        *((_BYTE *)Dst + v7) = aAbcdefghijklmn[(signed int)(unsigned __int8)byte_41A144[0] >> 2];
        v6 = v7 + 1;
        *((_BYTE *)Dst + v6++) = aAbcdefghijklmn[((byte_41A144[1] & 0xF0) >> 4) | 16 * (byte_41A144[0] & 3)];
        *((_BYTE *)Dst + v6++) = aAbcdefghijklmn[((byte_41A144[2] & 0xC0) >> 6) | 4 * (byte_41A144[1] & 0xF)];
        *((_BYTE *)Dst + v6) = aAbcdefghijklmn[byte_41A144[2] & 0x3F];
        v7 = v6 + 1;
        break;
    }
  }
  *((_BYTE *)Dst + v7) = 0;
  return Dst;
}
```

分析内容，并且查看其中出现的字符数组

```
.rdata:00417B30 aAbcdefghijklmn db 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/='
```

可确定为base64加密

编写Python脚本

```
#buu-reversed3
import base64
Dest = 'e3nifIH9b_C@n@dH'
flag = ''
for i in range(len(Dest)):
    flag += chr(ord(Dest[i])-i)

print(base64.b64decode(flag))
```

```
运行脚本：reverse3.py
b'{i_l0ve_you}'
>>>
```

运行脚本得到flag为

`flag{i_l0ve_you}`

不一样的flag

第一次接触这种迷宫题，不是很会，有参考其他大佬写的wp

用IDA打开，反编译C伪代码

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v3; // [esp+17h] [ebp-35h]
  int v4; // [esp+30h] [ebp-1Ch]
  int v5; // [esp+34h] [ebp-18h]
  signed int v6; // [esp+38h] [ebp-14h]
  int i; // [esp+3Ch] [ebp-10h]
  int v8; // [esp+40h] [ebp-Ch]

  __main();
  v4 = 0; //纵坐标
  v5 = 0; //横坐标
  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 )  //2即向下，纵坐标+1
    {
      ++v4;
    }
    else if ( v6 > 2 )
    {
      if ( v6 == 3 ) //3即向左，横坐标-1
      {
        --v5;
      }
      else
      {
        if ( v6 != 4 )  //大于4就退出
LABEL_13:
          exit(1);
        ++v5;   //4即向右，横坐标+1
      }
    }
    else
    {
      if ( v6 != 1 ) //输入数值不属于1234就退出
        goto LABEL_13;
      --v4;    //1即向上，纵坐标-1
    }
    for ( i = 0; i <= 1; ++i )
    {
      if ( *(&v4 + i) < 0 || *(&v4 + i) > 4 )
        exit(1);
    }
    if ( *((_BYTE *)&v8 + 5 * v4 + v5 - 41) == '1' ) //撞墙判定
      exit(1);    //根据5*v4判断迷宫一边长为5
    if ( *((_BYTE *)&v8 + 5 * v4 + v5 - 41) == '#' ) //判定到达终点
    {
      puts("\nok, the order you enter is the flag!");
      exit(0);
    }
  }
}
```

又可从程序中获取迷宫字符串“ *1111010000101000010111 1# ”，字符串长度25

```
.data:00402000 __data_start__  db '*1111010000101000010111 1#',0
.data:00402000                                         ; DATA XREF: _main+25↑o
```

按照5*5排列，*号为起点，#号为终点

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

要走出迷宫，走法为 222441144222 ，该顺序即为flag

`flag{222441144222}`

SimpleRev

ELF文件 用IDA64打开，F5反编译主函数main

```
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 != 'd' && v4 != 'D' )
        break;
      Decry();
    }
    if ( v4 == 'q' || v4 == 'Q' )
      Exit();
    puts("Input fault format!");
    v3 = getchar();
    putchar(v3);
  }
}
```

从主函数中进入关键函数`Decry()`

```
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 = 'SLCDN';    // 由Longlong转成字符串要逆序'NDCLS'(小端排序)
v7 = 0LL;
v8 = 0;
v9 = 'wodah';          // 同上 变为'hadow'
v10 = 0LL;
v11 = 0;
text = join(key3, (const char *)&v9);   // text即经过处理的flag
strcpy(key, key1);           // key=key1='ADSFK'
strcat(key, src);           // key=key+src='ADSFKNDCLS'
v2 = 0;
v3 = 0;
getchar();
v5 = strlen(key);    // v5 = 10
for ( i = 0; i < v5; ++i )
{      //将字母由大写转为小写
  if ( key[v3 % v5] > 64 && key[v3 % v5] <= 90 )
    key[i] = key[v3 % v5] + 32;
  ++v3;
}
printf("Please input your flag:", src);
while ( 1 )
{
  v1 = getchar();
  if ( v1 == '\n' )     // 回车结束输入
    break;
  if ( v1 == ' ' )
  {
    ++v2;          // v2代表字符数组str2下标
  }
  else
  {
    if ( v1 <= 96 || v1 > 122 )    // 处理输入的flag 结果存储于str2
    {
      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(' ');
    ++v2;
  }
}
if ( !strcmp(text, str2) )
  puts("Congratulation!\n");
else
  puts("Try again!\n");
return __readfsqword(0x28u) ^ v12;
}
```

代码里的`v3%v5`看着花里胡哨，其实就是0-9的循环。（flag长度为10）

又根据函数`join()`分析出关键参数text = ‘killshadow’

```
char *__fastcall join(const char *a1, const char *a2)
{
  size_t v2; // rbx
  size_t v3; // rax
  char *dest; // [rsp+18h] [rbp-18h]

  v2 = strlen(a1);          // a1='kills' v2 = 5
  v3 = strlen(a2);          // a2='hadow' v3 = 5
  dest = (char *)malloc(v2 + v3 + 1);   // +1是'\0'
  if ( !dest )
    exit(1);
  strcpy(dest, a1);
  strcat(dest, a2);
  return dest;         // text=dest='killshadow'
}
```

分析出几个关键参数的值，就能通过关键函数`Decry()`中的逻辑解出flag。

写python脚本

要注意的点就是对取余运算的逆向处理，代码中参数 j 代表余数。
去掉or条件的后半部分也是可以的，因为flag刚好都是大写字母。

```
#BUU-SimpleRev
k = 'ADSFKNDCLS'
t = 'killshadow'
key = list(k)
text = list(t)
l = len(key)
a = 0
flag = ''

for i in range(l):
    if(ord(key[i])>64 and ord(key[i])<=90):
        key[i] = chr(ord(key[i])+32)

for i in range(l):
    v = ord(text[i])-97
    for j in range(5):  #5是随便取的，只要包括0-2就ok
        n = j*26+v-97+39+ord(key[i])
        if((n>64 and n<=90)or(n>=97 and n<=122)):
            flag += chr(n)
            break

print(flag)
```

运行脚本，结果加上flag{}

`flag{KLDQCUDFZO}`

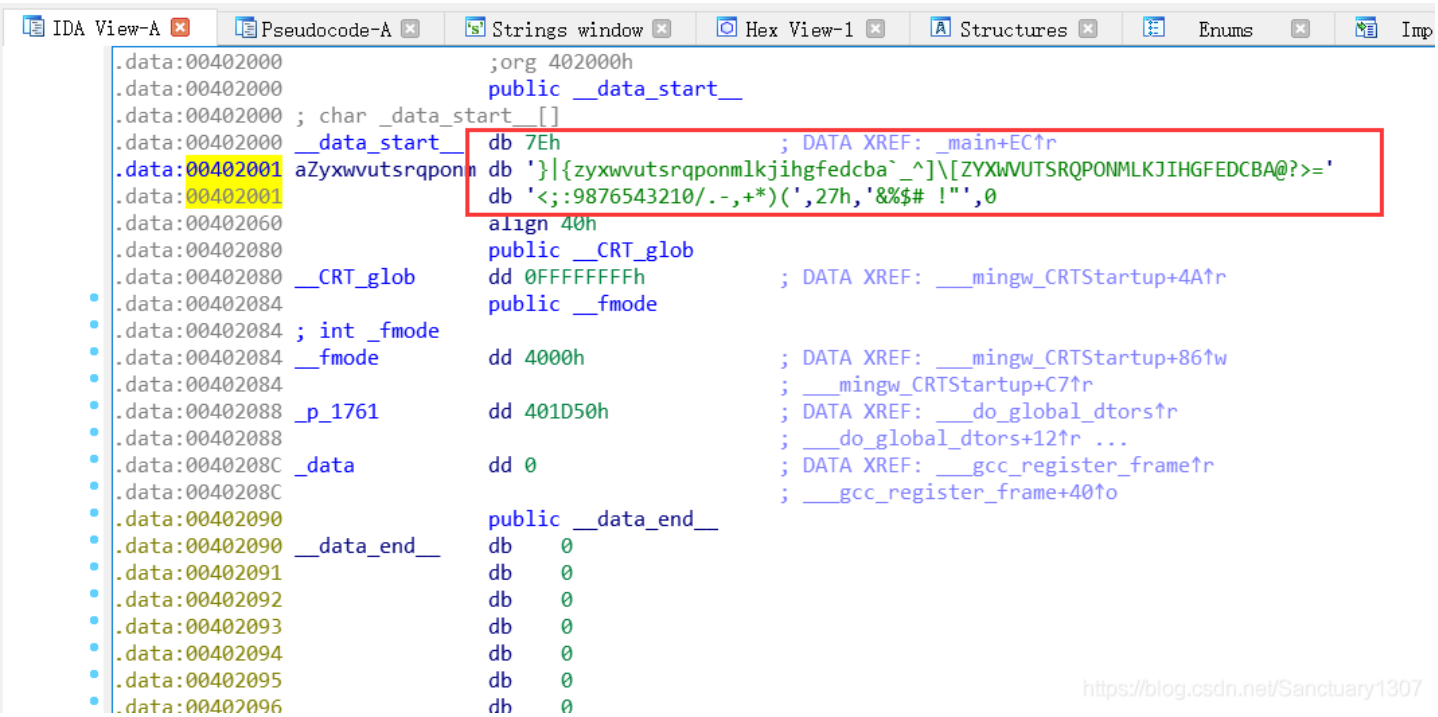
[ACTF新生赛2020]easyre

upx脱壳以后IDA打开

```
__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 ( (_BYTE)v19 != 'A' || HIBYTE(v19) != 'C' || v20 != 'T' || v21 != 'F' || v22 != '{' || v26 != '}' )
  return 0;
v16 = v23;
v17 = v24;
v18 = v25;
for ( i = 0; i <= 11; ++i )
{
  if ( *(&v4 + i) != _data_start__[*((char *)&v16 + i) - 1] )
    return 0;
}
printf("You are correct!");
return 0;
}
```


逻辑很简单，写个脚本就完事了

要注意的是不要搞错关键的`_data_start_`数组



ida选中红框部分，shift+e提取出来就行

```
data = [0x7E, 0x7D, 0x7C, 0x7B, 0x7A, 0x79, 0x78, 0x77, 0x76, 0x75,
  0x74, 0x73, 0x72, 0x71, 0x70, 0x6F, 0x6E, 0x6D, 0x6C, 0x6B,
  0x6A, 0x69, 0x68, 0x67, 0x66, 0x65, 0x64, 0x63, 0x62, 0x61,
  0x60, 0x5F, 0x5E, 0x5D, 0x5C, 0x5B, 0x5A, 0x59, 0x58, 0x57,
  0x56, 0x55, 0x54, 0x53, 0x52, 0x51, 0x50, 0x4F, 0x4E, 0x4D,
  0x4C, 0x4B, 0x4A, 0x49, 0x48, 0x47, 0x46, 0x45, 0x44, 0x43,
  0x42, 0x41, 0x40, 0x3F, 0x3E, 0x3D, 0x3C, 0x3B, 0x3A, 0x39,
  0x38, 0x37, 0x36, 0x35, 0x34, 0x33, 0x32, 0x31, 0x30, 0x2F,
  0x2E, 0x2D, 0x2C, 0x2B, 0x2A, 0x29, 0x28, 0x27, 0x26, 0x25,
  0x24, 0x23, 0x20, 0x21, 0x22, 0x00]
v = [42,70,39,34,78,44,34,40,73,63,43,64]
flag = ''
def find(num):
    n = 0
    for i in range(len(data)):
        if num==data[i]:
            n = i
            break
    return n

for i in range(12):
    flag+= chr(find(v[i])+1)

print('flag{'+flag+'}')
```

运行脚本：easyre.py

```
flag{U9X_1S_W6@T?}
>>>
```

`flag{U9X_1S_W6@T?}`

rsa

ctf中rsa的简单学习参考：https://www.cnblogs.com/nul1/p/13489269.html

题目给出了一个flag.enc和pub.key文件，分别为rsa密文和公钥。

首先从公钥分解出模数N和e，可以使用OpenSSL(以下为ubuntu Linux环境下命令及结果)

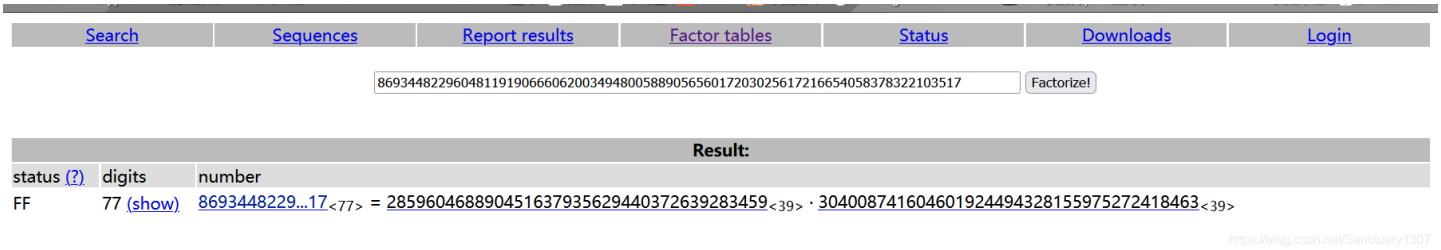
```
$ openssl rsa -pubin -text -modulus -in pub.key
RSA Public-Key: (256 bit)
Modulus:
    00:c0:33:2c:5c:64:ae:47:18:2f:6c:1c:87:6d:42:
    33:69:10:54:5a:58:f7:ee:fe:fc:0b:ca:af:5a:f3:
    41:cc:dd
Exponent: 65537 (0x10001)
Modulus=C0332C5C64AE47182F6C1C876D42336910545A58F7EEFEFC0BCAAF5AF341CCDD
writing RSA key
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhAMAzLFxkrkcYL2wch21CM2kQVFpY9+7+
/AvKr1rzQczdAgMBAAE=
-----END PUBLIC KEY-----
```

模数`C0332C5C64AE47182F6C1C876D42336910545A58F7EEFEFC0BCAAF5AF341CCDD`需先转十进制再从中提取p，q
转十进制结果`86934482296048119190666062003494800588905656017203025617216654058378322103517`
可用yafu工具或者在线网站(参考博客)

yafu结果如下

```
 C77, B1=50K, B2=gmp-ecm default, ETA: 0 sec

starting SIQS on c77: 869344822960481191906660620034948005889056560172030256172
78322103517

==== sieving in progress (1 thread):   36224 relations needed ====
====           Press ctrl-c to abort and save state           ====
1383 rels found: 1292 full + 91 from 12784 partial, (2695.19 rels/s2902 rels f
 full + 358 from 24877 4632 r8580 rels found: 6397 full + 2183 from 63618 part
36092 rels found: 18939 full + 17153 from 185028 partial, (2641.23 rels/sec))

SIQS elapsed time = 78.6184 seconds.
Total factoring time = 92.5398 seconds


***factors found***

P39 = 304008741604601924494328155975272418463
P39 = 285960468890451637935629440372639283459

ans = 1
```


在线网站结果如下



编写python3脚本（需安装gmpy2库和rsa库）

```python
import gmpy2
import rsa
p = 304008741604601924494328155975272418463
q = 285960468890451637935629440372639283459
e = 65537
with open('flag.enc','rb') as f:
    text = f.read()
print(f)
n = p*q
fn = (p-1) *(q-1)
d = int(gmpy2.invert(e, fn)) #需用int()转换类型 否则会报错
privatekey = rsa.PrivateKey(n,e,d,p,q)
flag = rsa.decrypt(text,privatekey).decode()
print(flag)
```

结果：`flag{decrypt_256}`

```
terra@terra-Ubuntu:~/桌面/output$ python3 test.py
<_io.BufferedReader name='flag.enc'>
flag{decrypt_256}
```

其他题目待更新