2019红帽杯easyRE-ACTF新生赛2020rome-[FlareOn4]login-Youngterdrive

原创

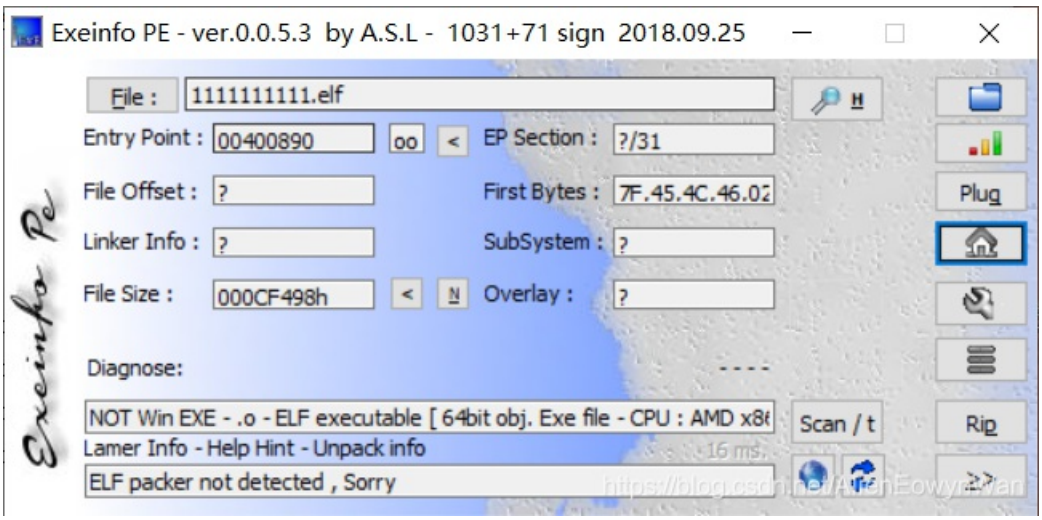
^Norming 于 2021-06-03 01:27:01 发布 248 收藏

分类专栏： BUUCTF

本文链接：https://blog.csdn.net/AlienEowynWan/article/details/117446056

BUUCTF

[2019红帽杯]easyRE



elf文件，ida64打开，感兴趣的同学也可以开Linux系统看一看文件

```
v56 = __readfsqword(0x28u);
 v14 = 73;
 v15 = 111;
 v16 = 100;
 v17 = 108;
 v18 = 62;
 v19 = 81;
```

```
v20 = 110;
v21 = 98;
v22 = 40;
v23 = 111;
v24 = 99;
v25 = 121;
v26 = 127;
v27 = 121;
v28 = 46;
v29 = 105;
v30 = 127;
v31 = 100;
v32 = 96;
v33 = 51;
v34 = 119;
v35 = 125;
v36 = 119;
v37 = 101;
v38 = 107;
v39 = 57;
v40 = 123;
v41 = 105;
v42 = 121;
v43 = 61;
v44 = 126;
v45 = 121;
v46 = 76;
v47 = 64;
v48 = 69;
v49 = 67;
memset(v50, 0, sizeof(v50));
v51 = 0;
v52 = 0;
v0 = v50;
sub_4406E0(0LL, v50, 37LL);
v52 = 0;
v1 = v50;
if ( sub_424BA0(v50) == 36 )
{
  for ( i = 0; ; ++i )
  {
    v1 = v50;
    if ( i >= (unsigned __int64)sub_424BA0(v50) )
      break;
    if ( (unsigned __int8)(v50[i] ^ i) != *(&v14 + i) )
    {
      result = 4294967294LL;
      goto LABEL_13;
    }
  }
  sub_410CC0((__int64)"continue!");           // 第一部分处理
  memset(&v53, 0, 0x40uLL);
  v55 = 0;
  v0 = &v53;
  sub_4406E0(0LL, &v53, 64LL);
  v54 = 0;
  v1 = &v53;
  if ( sub_424BA0(&v53) == 39 )
  {
    v3 = sub_400E44((__int64)&v53);           // 大次base64处理
```

```
      v4 = sub_400E44(v3);
      v5 = sub_400E44(v4);
      v6 = sub_400E44(v5);
      v7 = sub_400E44(v6);
      v8 = sub_400E44(v7);
      v9 = sub_400E44(v8);
      v10 = sub_400E44(v9);
      v11 = sub_400E44(v10);
      v12 = sub_400E44(v11);
      v0 = off_6CC090;
      v1 = (char *)v12;
      if ( !(unsigned int)sub_400360(v12, off_6CC090) )
      {
        sub_410CC0((__int64)"You found me!!!");
        v1 = "bye bye~";
        sub_410CC0((__int64)"bye bye~");
      }
      result = 0LL;
    }
    else
    {
      result = 4294967293LL;
    }
  }
  else
  {
    result = 0xFFFFFFFFLL;
  }
LABEL_13:
  if ( __readfsqword(0x28u) != v56 )
    sub_444020(v1, v0);
  return result;
}
```

第一部分的处理是个异或，第二部分打开函数之后发现是base64的运算（10次），而且得到的结果与off_6cc090比较

先解出第一部分

```
str = [73,111,100,108,62,81,110,98,40,111,99,121,127,121,46,105,127,100,96,51,119,125,119,101,107,57,123,105,121
,61,126,121,76,64,69,67]
input1 = ''
for i in range(len(str)):
   input1 += chr(str[i]^i)
print(input1)
```

```
C:/Users/86135/Desktop/first/.idea/kjgijtkjmj.py
Info:The first four chars are `flag`
```

第二部分用在线工具解十次base64，解出来发现是个网站

https://bbs.pediy.com/thread-254172.htm

打开发现没有用，是个干扰项

看了其他师傅的wp后才知道，在我们base64加密结束后，下面有调用一个sub_400D35的函数

```
.data:00000000006CC08F                  db    0
.data:00000000006CC090 off_6CC090       dq offset aVm0wd2vhuxhtwg
.data:00000000006CC090                                          ; DATA XREF: sub_4009C6+31B↑r
.data:00000000006CC090                                          ; "Vm0wd2VHUXhTWGhpUm1SWVYwZDRWVll3Wk
.data:00000000006CC098                  align 20h
.data:00000000006CC0A0 byte_6CC0A0      db 40h                  ; DATA XREF: sub_400D35+95↑r
.data:00000000006CC0A0                                          ; sub_400D35+C1↑r
.data:00000000006CC0A1                  db  35h ; 5
.data:00000000006CC0A2                  db  20h
.data:00000000006CC0A3 byte_6CC0A3      db 56h                  ; DATA XREF: sub_400D35+A6↑r
.data:00000000006CC0A4                  db  5Dh ; ]
.data:00000000006CC0A5                  db  18h
.data:00000000006CC0A6                  db  22h ; "
.data:00000000006CC0A7                  db  45h ; E
.data:00000000006CC0A8                  db  17h
```


过去看看

```
12  v9 = __readfsqword(0x28u);
13  v2 = 0LL;
14  v5 = sub_43FD20(0LL) - qword_6CEE38;
15  for ( i = 0; i <= 1233; ++i )
16  {
17    v2 = v5;
18    sub_40F790(v5);
19    sub_40FE60();
20    sub_40FE60();
21    v5 = (unsigned __int64)sub_40FE60() ^ 0x98765432;
22  }
23  v8 = v5;
24  if ( ((unsigned __int8)v5 ^ byte_6CC0A0[0]) == 102 && (HIBYTE(v8) ^ (unsigned __int8)byte_6CC0A3) == 103 )
25  {
26    for ( j = 0; j <= 24; ++j )
27    {
28      v2 = (unsigned __int8)(byte_6CC0A0[j] ^ *((_BYTE *)&v8 + j % 4));
29      sub_410E90(v2);
30    }
31  }
32  v4 = __readfsqword(0x28u);
33  result = v4 ^ v9;
34  if ( v4 != v9 )
35    sub_444020(v2, a2);
36  return result;
37}
```


24行判断第一个和第四个是不是f和g，那就猜测这四个字符是flag。

然后循环24次做异或操作。

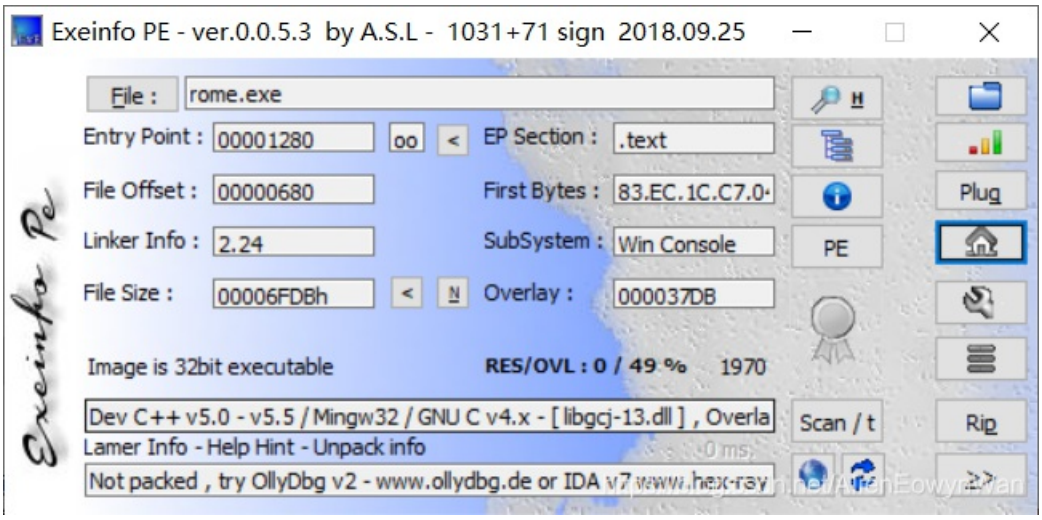
```
num2=[0x40,0x35,0x20,0x56,0x5D,0x18,0x22,0x45,0x17,0x2F,0x24,0x6E,0x62,0x3C,0x27,0x54,0x48,0x6C,0x24,0x6E,0x72,0
x3C,0x32,0x45,0x5B]
text2='flag'
key=''
for i in range(0,4):
    key=key+chr(ord(text2[i])^num2[i])
print (key)
```

得到v5 == &YA1
在求出v2

```
num2=[0x40,0x35,0x20,0x56,0x5D,0x18,0x22,0x45,0x17,0x2F,0x24,0x6E,0x62,0x3C,0x27,0x54,0x48,0x6C,0x24,0x6E,0x72,0
x3C,0x32,0x45,0x5B]
key='&YA1'
flag=''
for i in range(0,25):
    flag=flag+chr(num2[i]^ord(key[i%4]))
print(flag)
```

flag是：flag{Act1ve_Defen5e_Test}

[ACTF新生赛2020]rome



ida 32，通过main函数或者字符串搜索都可以找到函数`func()`
发现其实就是将输入的字符串大写和小写分开进行变换，然后比较输出

```
37  v15 = 81;
38  v16 = 115;
39  v17 = 119;
40  v18 = 51;
41  v19 = 115;
42  v20 = 106;
43  v21 = 95;
44  v22 = 108;
45  v23 = 122;
46  v24 = 52;
47  v25 = 95;
48  v26 = 85;
49  v27 = 106;
50  v28 = 119;
51  v29 = 64;
52  v30 = 108;
53  v31 = 0;
54  printf("Please input:");
```

```
54  printf( Please input: );
55  scanf("%s", &v5);
56  result = v5;
57  if ( v5 == 'A' )
58  {
59    result = v6;
60    if ( v6 == 'C' )
61    {
62      result = v7;
63      if ( v7 == 'T' )
64      {
65        result = v8;
66        if ( v8 == 'F' )
67        {
68          result = v9;
69          if ( v9 == '{' )
70          {
71            result = v14;
72            if ( v14 == '}' )
73            {
74              v1 = v10;
75              v2 = v11;
76              v3 = v12;
77              v4 = v13;
78              for ( i = 0; i <= 15; ++i )
79              {
80                if ( *((_BYTE *)&v1 + i) > '@' && *((_BYTE *)&v1 + i) <= 'Z' )// 是大写字母
81                  *((_BYTE *)&v1 + i) = (*((char *)&v1 + i) - 51) % 26 + 65;
82                if ( *((_BYTE *)&v1 + i) > '`' && *((_BYTE *)&v1 + i) <= 'z' )// 是小写字母
83                  *((_BYTE *)&v1 + i) = (*((char *)&v1 + i) - 79) % 26 + 97;
84              }
85              for ( i = 0; i <= 15; ++i )
86              {
87                result = (unsigned __int8)*(&v15 + i);
88                if ( *((_BYTE *)&v1 + i) != (_BYTE)result )
89                  return result;
90              }
91              result = printf("You are correct!");
92            }
93          }
94        }
95      }
96    }
00000830 func:79 (401430)
```


写一个脚本跑一下

```
v15= [81,115,119,51,115,106,95,108,122,52,95,85,106,119,64,108]
flag=""

for i in range(16):
    for j in range(128):
        x=j
        if chr(x).isupper():
            x=(x-51)%26+65
        if chr(x).islower():
            x=(x-79)%26+97
        if chr(x)==chr(v15[i]):
            flag+=chr(j)

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

flag{Cae3ar_th4_Gre@t}

[FlareOn4]login

网页打开，F12查看HTML源码

```
<!DOCTYPE html>
<html>
▼<head>
    <title>FLARE On 2017</title>
  </head>
▼<body>
    <input type="text" name="flag" id="flag" value="Enter the flag"> == $0
    <input type="button" id="prompt" value="Click to check the flag">
  ▼<script type="text/javascript">
                document.getElementById("prompt").onclick = function () {
                    var flag = document.getElementById("flag").value;
                    var rotFlag = flag.replace(/[a-zA-Z]/g, function(c){return String.fromCharCode((c <= "Z" ? 90 :
      122) >= (c = c.charCodeAt(0) + 13) ? c : c - 26);});
                    if ("PyvragFvqrYbtvafNerRnfl@syner-ba.pbz" == rotFlag) {
                        alert("Correct flag!");
                    } else {
                        alert("Incorrect flag, rot again");
                    }
                }

    </script>
  </body>
</html>
```

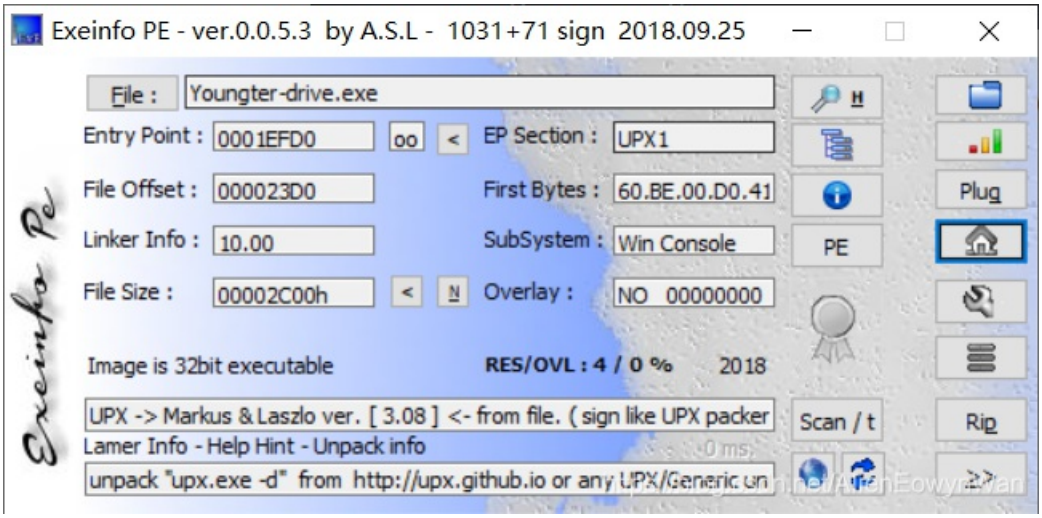

`var rotFlag = flag.replace(/[a-zA-Z]/g, function(c){return String.fromCharCode((c <= "Z" ? 90 : 122) >= (c = c.charCodeAt(0) + 13) ? c : c - 26);});` 这句代码的意思就是将字符后移13位，越界了就转回字母开头，最后得到了字符串 `PyvragFvqrYbtvafNerRnfl@syner-ba.pbz`

其实就是ROT13，找一个在线网站解一下，得到：

```
ClientSideLoginsAreEasy@flare-on.com
```

flag是：flag{ClientSideLoginsAreEasy@flare-on.com}

Youngter-drive

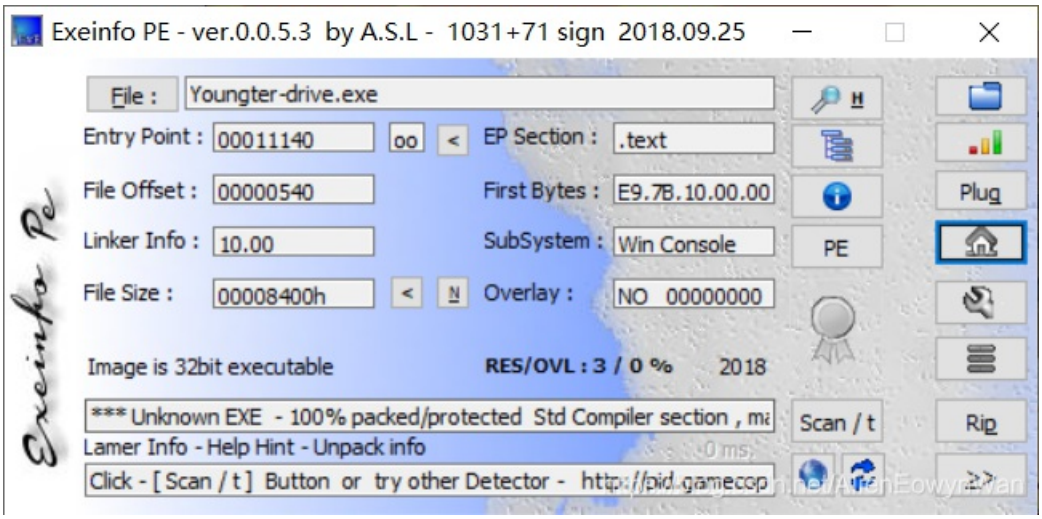


kali脱壳

```
root@AEW:~/桌面# upx -d ./Youngter-drive.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2018
UPX 3.95        Markus Oberhumer, Laszlo Molnar & John Reiser   Aug 26th 2018

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     33792 <-     11264   33.33%    win32/pe     Youngter-drive.exe

Unpacked 1 file.
root@AEW:~/桌面#
```

ida打开，找到主函数

```
1  int main_0()
2  {
3    HANDLE v2; // [esp+D0h] [ebp-14h]
4    HANDLE hObject; // [esp+DCh] [ebp-8h]
5
6    sub_4110FF();                                 // 输入source
7    ::hObject = CreateMutexW(0, 0, 0);
8    j_strcpy(Dest, Source);                       // source赋值给Dest
9    hObject = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)StartAddress, 0, 0, 0);// 线程1
10   v2 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_41119F, 0, 0, 0);// 线程2
11   CloseHandle(hObject);
12   CloseHandle(v2);
13   while ( dword_418008 != -1 )
14     ;
15   sub_411190();                                 // 输出flag
16   CloseHandle(::hObject);
17   return 0;
18 }
```


大致是这样，我们依次看一下

函数`sub_4110FF()`

```
1  int sub_411BD0()
2  {
3    printf(
4      "1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111\n"
5      "**************************************************************************************\n"
6      "**************              ********************************************************\n"
7      "**************   ********   ********************                  ************\n"
```

```
    "**************   *********  ********************   ***************************\n"
    "**************   *********  ********************   ***************************\n"
    "**************   *********  ********************   ***************************\n"
    "**************   *******   ********************   ***************************\n"
    "**************   ****   ************************   ***************************\n"
    "**************   *    ****************************               **************\n"
    "**************   ***    ************************   ***************************\n"
    "**************   ******   **********************   ***************************\n"
    "**************   ********   ********************   ***************************\n"
    "**************   **********   ******************   ***************************\n"
    "**************   ***********    *****************               **************\n"
    "*********************************************************************************\n"
    "11111111111111111111111111111111111111111111111111111111111111111111111111111111\n");
  printf("input flag:\n");
  return scanf("%36s", Source);
}
```


输出部分：函数`sub_411190()`

```
int sub_411880()
{
  int i; // [esp+D0h] [ebp-8h]

  for ( i = 0; i < 29; ++i )
  {
    if ( Source[i] != off_418004[i] )           // 变换后的Source == "TOiZiZtOrYaToUwPnToBsOaOapsyS
      exit(0);
  }
  return printf("\nflag{%s}\n\n", Dest);        // flag == Dest==之前的Source
}
```

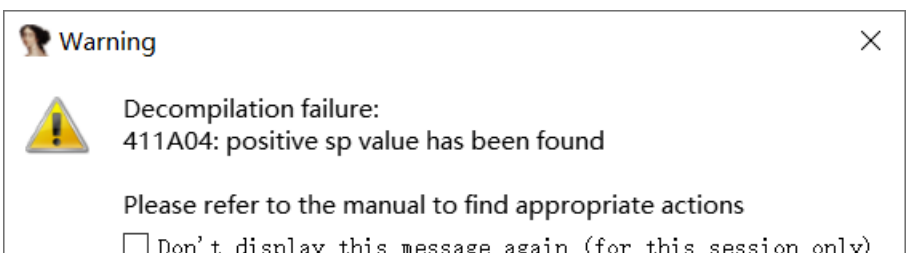

然后着重看一下两个线程的处理

线程1：`StartAddress`

```
void __stdcall StartAddress_0(int a1)
{
  while ( 1 )
  {
    WaitForSingleObject(hObject, 0xFFFFFFFF);
    if ( dword_418008 > -1 )                    // dword_418008 == 29
    {
      sub_41112C((int)Source, dword_418008);    // 将source和dword_418008传入函数处理
      --dword_418008;                           // dword_418008-1
      Sleep(0x64u);
    }
    ReleaseMutex(hObject);
  }
}
```


看函数`sub_41112C`，函数`sub_411940`

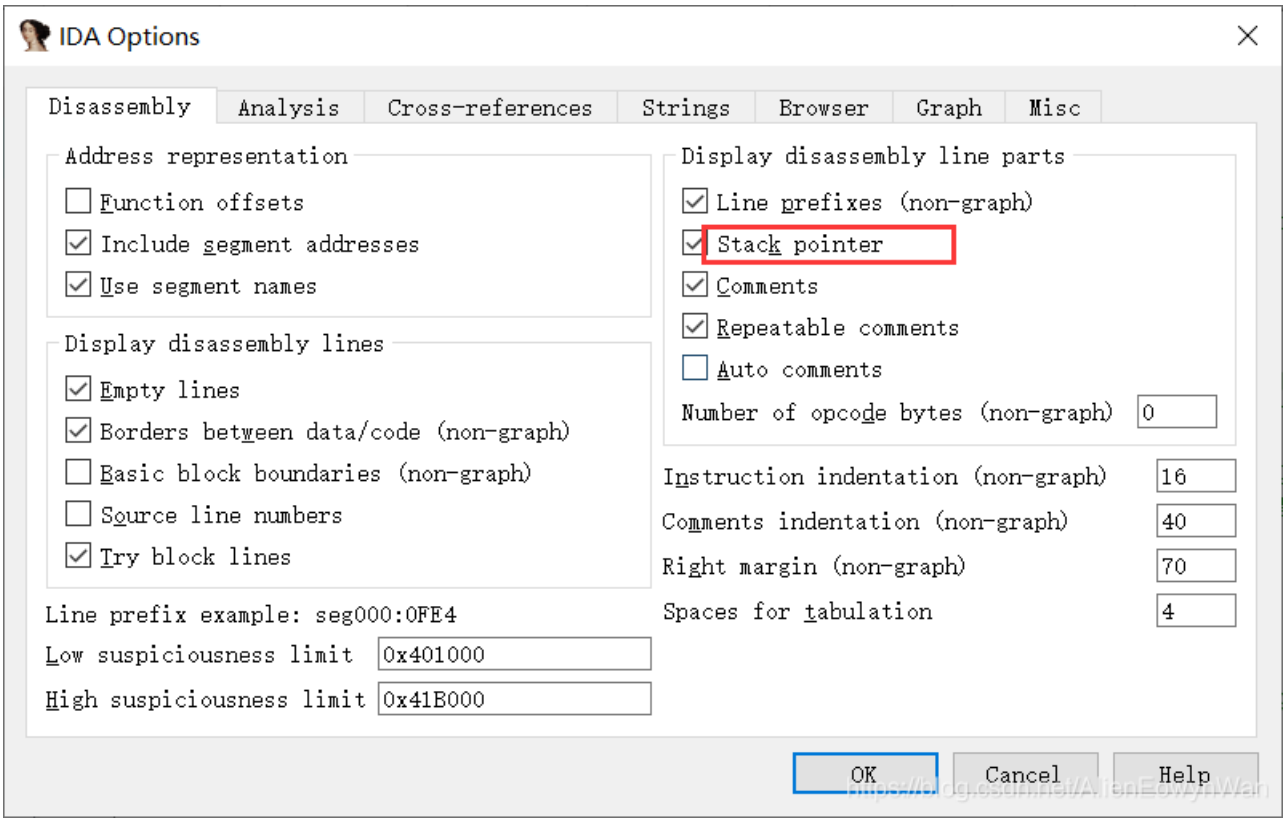
这里会出现报错，函数`sub_411940`无法反编译



是由于堆栈不平衡的问题，一般是程序有一些干扰代码，让IDA的反汇编分析出现错误。比如用push + n条指令 + retn来跳转。搜索这个函数，看汇编窗口：

```
1A03                pop     ebp
1A04                retn
1A04 sub_411940     endp ; sp-analysis failed
1A04
1A04 ; ---------------------------------------------
```

Option->General->Disassembly, 将选项Stack pointer打钩



然后就可以看见汇编窗口汇编指令前面有绿色的SP值

```
.text:004119F1     loc_4119F1:                             ; CODE XREF:
.text:004119F1 0D8                 pop     edi
.text:004119F2 0D4                 pop     esi
.text:004119F3 0D0                 pop     ebx
.text:004119F4 0CC                 add     esp, 0CCh
.text:004119FA 000                 cmp     ebp, esp
.text:004119FC 000                 call    j___RTC_CheckEsp
.text:00411A01 000                 mov     esp, ebp
.text:00411A03 000                 pop     ebp
.text:00411A04 -04                 retn
.text:00411A04     sub_411940      endp ; sp-analysis failed
.text:00411A04
```

找到这串函数的retn指令

```
19F1 0D8                pop     edi
```

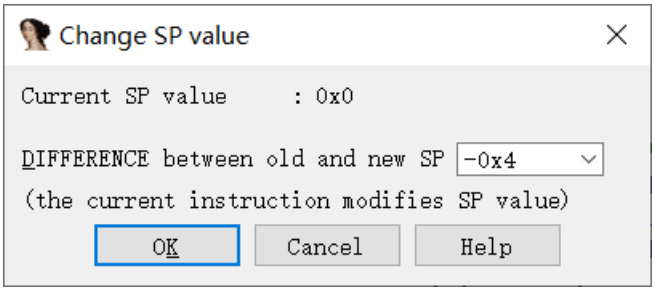
```
19F2 0D4                pop     esi
19F3 0D0                pop     ebx
19F4 0CC                add     esp, 0CCh
19FA 000                cmp     ebp, esp
19FC 000                call    j___RTC_CheckEsp
1A01 000                mov     esp, ebp
1A03 000                pop     ebp
1A04 -04                retn
1A04    sub_411940      endp ; sp-analysis failed
1A04
```


可以发现前面的SP值变成了负数，我们需要修改它成0使得堆栈保持平衡。

找到距离retn最近的call指令

```
19FA 000                cmp     ebp, esp
19FC 000                call    j___RTC_CheckEsp
1A01 000                mov     esp, ebp
1A03 000                pop     ebp
1A04 -04                retn
1A04    sub_411940      endp ; sp-analysis failed
1A04
```

alt+k修改SP值



修改之后

```
000                 cmp     ebp, esp
000                 call    j___RTC_CheckEsp
004                 mov     esp, ebp
004                 pop     ebp
000                 retn
    sub_411940      endp ; sp-analysis failed
```

然后就可以可以查看

```
// a1==source a2==dword_418008
char *__cdecl sub_411940(int a1, int a2)
{
  char *result; // eax
  char v3; // [esp+D3h] [ebp-5h]

  v3 = *(_BYTE *)(a2 + a1);                     // 因为开始是29然后一直减所以是从Source倒序处理
  if ( (v3 < 'a' || v3 > 'z') && (v3 < 'A' || v3 > 'Z') )// 不是英文字母则退出
    exit(0);
  if ( v3 < 'a' || v3 > 'z' )                   // 大写字母处理
  {
    result = off_418000[0];                     // off_418000 == QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm
    *(_BYTE *)(a2 + a1) = off_418000[0][*(char *)(a2 + a1) - 38];// 我们知道变换后的Source== TOiZiZtOrYaToUwPnToBsOaOapsyS
                                                // 反推就是，
                                                // 变换后的Source[i]==off_418000[某个数值j],求出j
                                                // j==变换之前的Source[i]-38,求出变换之前的Source[i]
                                                //
  }
  else                                          // 小写字母处理
  {
```

```
21    result = off_418000[0];
22    *(_BYTE *)(a2 + a1) = off_418000[0][*(char *)(a2 + a1) - 96];// 与上面同理，+96
23  }
24  return result;
25}
```


线程2：函数 `sub_41119F()`

```
1 void __stdcall sub_411B10(int a1)
2 {
3   while ( 1 )
4   {
5     WaitForSingleObject(hObject, 0xFFFFFFFF);
6     if ( dword_418008 > -1 )
7     {
8       Sleep(0x64u);
9       --dword_418008;
10    }
11    ReleaseMutex(hObject);
12  }
13}
```


线程2并不对代码做任何处理，只是调用一次Sleep然后释放线程。

这里说一个细节，这道题是线程1和线程2 交替执行的，因为函数 `CreateMutex()`

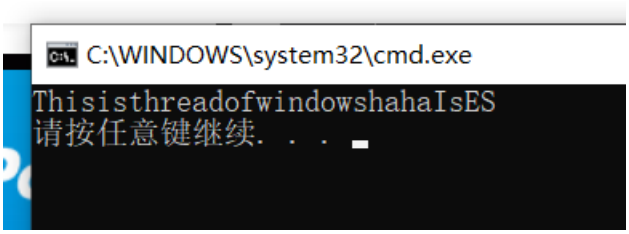
```
7  ::hObject = CreateMutexW(0, 0, 0);
```

这个函数是创建两个相互排斥的线程的函数，hobject 可以理解为一个“锁”，两个线程依次获取锁(waitforsingleobject)，释放锁(releasemutex)。一个线程开始之后，抢占了锁，这时候另一个线程没抢到，处于等待状态，等待运行线程执行完成之后第一个线程解锁。
不能用来控制两个以上的线程，因为剩下的几个线程都有几率抢到，会导致程序逻辑混乱。
两个线程时候，第一次抢到锁的线程在释放后也会和第二个线程抢锁，但是由于它此时没有第二个线程快，所以抢不到，就造成了两个线程交替被调用的现状。

所以这题就是奇数的会被线程1处理而偶数走线程2 不处理（因为是从29往下减的）

写脚本

```c
#include <stdio.h>
int main()
{
    char off1[] = "TOiZiZtOrYaToUwPnToBsOaOapsyS";
    char off2[] = "QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm";
    char flag[30]={0};
    int i,j;
    for(i=28;i>-1;i--)
    {
        if(i%2==0)
        {
            flag[i] = off1[i];
            continue;
        }
        for(j=0;j<52;j++)
        {
            if(off1[i] == off2[j])
            {
                flag[i] = j+38;
                if(!(flag[i]>=65 &&flag[i]<=90))
                    flag[i] = j+96;
                break;
            }
        }
    }
    puts(flag);
    return 0;
}
```



这里还有一个问题，就是最后输出flag的函数，它显示的是比较i<29
而实际上进程处理的时候是到>-1的，也就是其实应该是30个字符
也就是说少了一位我们不知道，需要一个个英文字母试出来，发现是“E”

flag：flag{ThisisthreadofwindowshahaIsESE}