

# buuctf Youngter-drive

原创

菜逼的ctf之路 于 2020-10-31 15:26:05 发布 1145 收藏 1

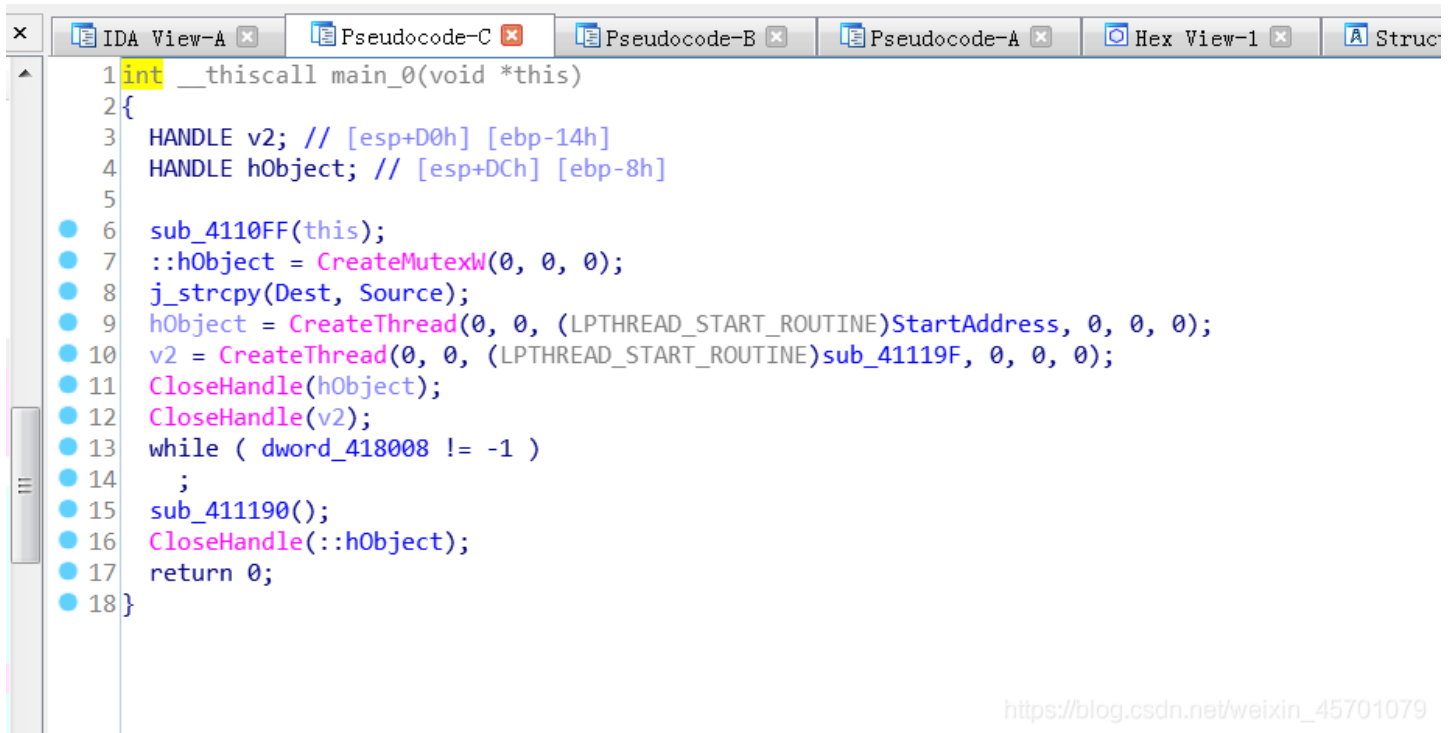
版权声明：本文为博主原创文章，遵循 [CC 4.0 BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) 版权协议，转载请附上原文出处链接和本声明。

本文链接：[https://blog.csdn.net/weixin\\_45701079/article/details/109402704](https://blog.csdn.net/weixin_45701079/article/details/109402704)

版权

## buuctf Youngter-drive

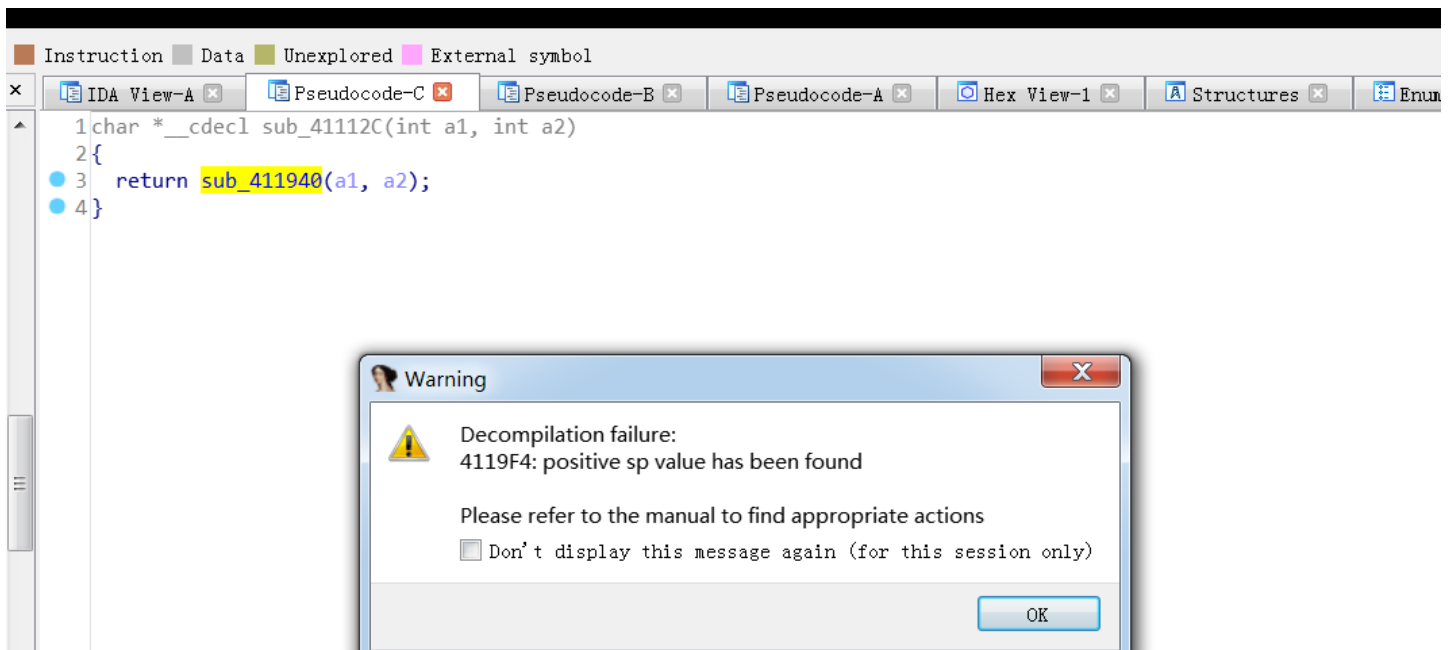
拿到题目，查壳(upx),首先脱壳，(不会脱壳自行百度,很多教程),脱壳之后是不能运行的，好像是因为文件重定向，但是不影响ida静态分析，ida打开



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

[https://blog.csdn.net/weixin\\_45701079](https://blog.csdn.net/weixin_45701079)

分析，多线程，最近刚好在学多线程，第一个线程hObject会执行StartAddress函数，第二个线程v2执行sub\_41119f函数,然后挨个分析打开StartAddress函数



```
1 char *__cdecl sub_41112C(int a1, int a2)
2 {
3     return sub_411190(a1, a2);
4 }
```

**Warning**

Decompilation failure:  
41119F4: positive sp value has been found

Please refer to the manual to find appropriate actions

Don't display this message again (for this session only)

OK

会出现这种情况，原因是堆栈不平衡利用ida调堆栈就好，

```

View-A x Pseudocode-C x Pseudocode-B x Pseudocode-A x Hex View-1 x Struct
.text:004119F4      add     esp, 0CCh
.text:004119FA      cmp     ebp, esp
.text:004119FC      call   j__RTC_CheckEsp
.text:00411A01      mov     esp, ebp
.text:00411A03      pop     ebp
.text:00411A04      retn
.text:00411A04      sub_411940  endp ; sp-analysis failed
.text:00411A04      ; -----
.text:00411A05      align 40h
.text:00411A40      ; ===== S U B R O U T I N E =====
.text:00411A40      ; Attributes: bp-based frame
.text:00411A40      StartAddress_0  proc near                ; CODE XREF: StartAddress↑j
.text:00411A40      var_C0          = byte ptr -0C0h
.text:00411A40      push    ebp
.text:00411A41      mov     ebp, esp
.text:00411A43      sub     esp, 0C0h
.text:00411A49      push    ebx
.text:00411A4A      push    esi
.text:00411A4B      push    edi
.text:00411A4C      lea    edi, [ebp+var_C0]
.text:00411A52      mov     ecx, 30h
00000E40 00411A40: StartAddress_0 (Synchronized with Hex View-1)  https://blog.csdn.net/weixin_45701079

```

在红色部分用alt+k，打开![在这里插入图片描述]

```

Instruction Data Unexplored External symbol
IDA View-A x Pseudocode-C x Pseudocode-B x Pseudocode-A x Hex View-1 x Struct
.text:004119EC 0D8      call   j__RTC_CheckEsp
.text:004119F1
.text:00411A04      ; CODE XREF: sub_411940:loc_4119DE
.text:00411A04      retn
.text:00411A04      sub_411940  endp ; sp-analysis failed
.text:00411A04      ; -----
.text:00411A05      align 40h
.text:00411A40      ; ===== S U B R O U T I N E =====
.text:00411A40      ; Attributes: bp-based frame

```

**Change SP value**

Current SP value : 0xD0

DIFFERENCE between old and new SP

(the current instruction modifies SP value)

OK Cancel Help

```

.text:00411A40      StartAddress_0  proc near                ; CODE XREF: StartAddress↑j
.text:00411A40
.text:00411A40      var_C0          = byte ptr -0C0h
.text:00411A40
.text:00411A40 000          push    ebp
.text:00411A41 004          mov     ebp, esp

```

00000E03 00411A03: sub\_411940+C3 (Synchronized with Hex View-1)

[https://blog.csdn.net/weixin\\_45701079](https://blog.csdn.net/weixin_45701079)

把原来的0x-4改成0x00，堆栈平衡，然后就可以打开函数了

```

IDA View-A x Pseudocode-C x Pseudocode-B x Pseudocode-A x Hex View-1 x Structure
1 char *__cdecl sub_411940(int a1, int a2)
2 {
3     char *result; // eax
4     char v3; // [esp+D3h] [ebp-5h]
5
6     v3 = *(_BYTE *)(a2 + a1);
7     if ( (v3 < 97 || v3 > 122) && (v3 < 65 || v3 > 90) )
8         exit(0);
9     if ( v3 < 97 || v3 > 122 )
10    {
11        result = off_418000[0];
12        *(_BYTE *)(a2 + a1) = off_418000[0][*(char *)(a2 + a1) - 38];
13    }
14    else
15    {
16        result = off_418000[0];
17        *(_BYTE *)(a2 + a1) = off_418000[0][*(char *)(a2 + a1) - 96];
18    }
19    return result;
20 }

```

00000E01 sub\_411940:19 (411A01)

[https://blog.csdn.net/weixin\\_45701079](https://blog.csdn.net/weixin_45701079)

逻辑就出现了

然后有坑的地方出现了

他是两个线程同时工作的，所以要交替处理数据

看另一个线程，发现没有处理任何数据

```

IDA View-A x Pseudocode-C x Pseudocode-B x Pseudocode-A
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 }

```

[https://blog.csdn.net/weixin\\_45701079](https://blog.csdn.net/weixin_45701079)

只是把原来的字符串向前移动

注意数据是从后向前处理的，查看数据大小发现是0x1d，就是29个，然后分析就行

最后贴上脚本

```
str="0abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
text1='TOiZiZtOrYaToUwPnToBs0a0apsyS'
text2='QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm'
flag=''
s=0
for i in range(len(text1)):
    if(i%2==0):
        flag+=text1[i]
    else:
        s=text2.index(text1[i])
        flag+=str[s]
print(flag)
```

然后要求有30位数据，发现有一位数据没有对比，一个一个试，发现"E"成立