攻防世界逆向入门题之Mysterious

原创

沐一 · 林 于 2021-08-18 10:21:50 发布 28 收藏

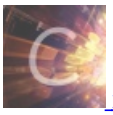
分类专栏： CTF 逆向 文章标签： unctf

本文链接：https://blog.csdn.net/xiao__1bai/article/details/119773766


CTF 同时被 2 个专栏收录

167 篇文章 6 订阅

订阅专栏


逆向

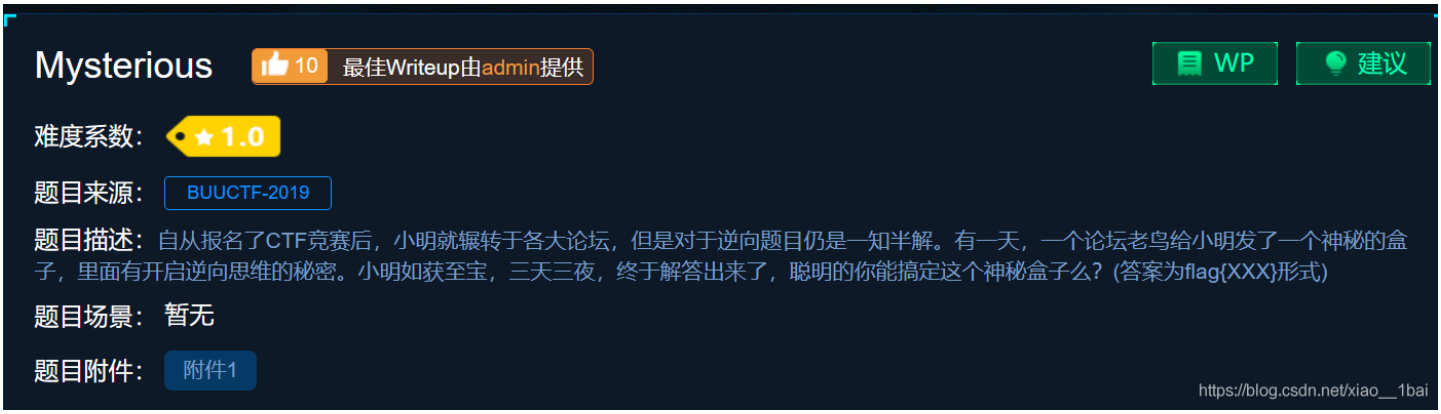
95 篇文章 6 订阅

订阅专栏

攻防世界逆向入门题之Mysterious

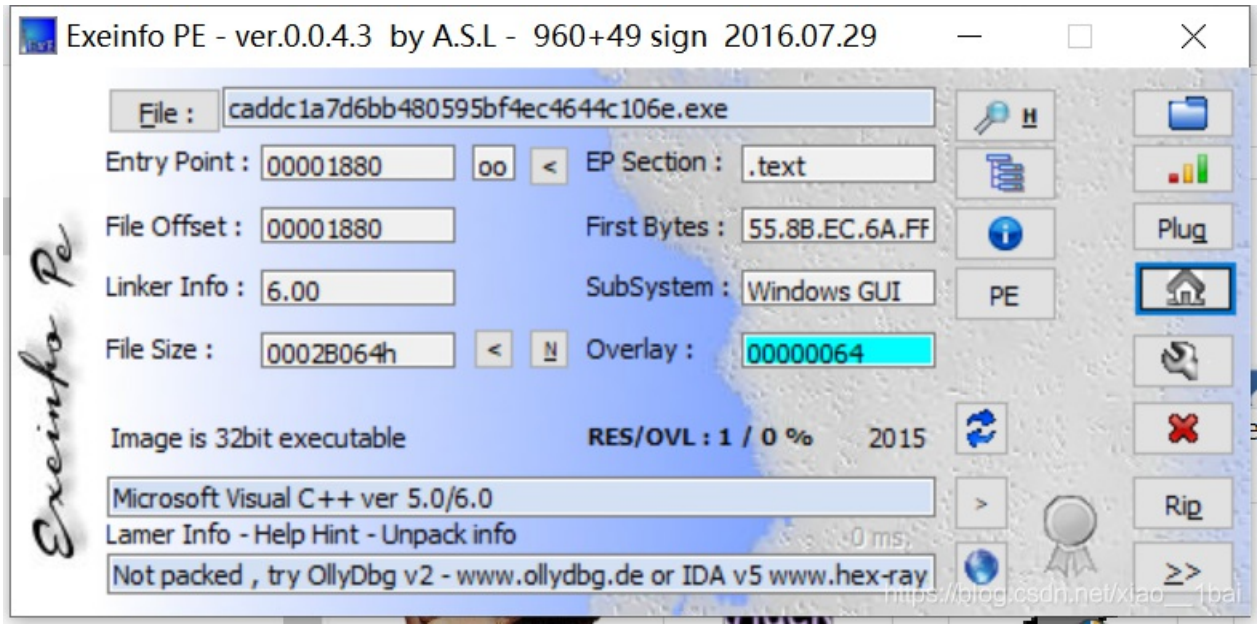
继续开启全栈梦想之逆向之旅~

这题是攻防世界逆向入门题的Mysterious

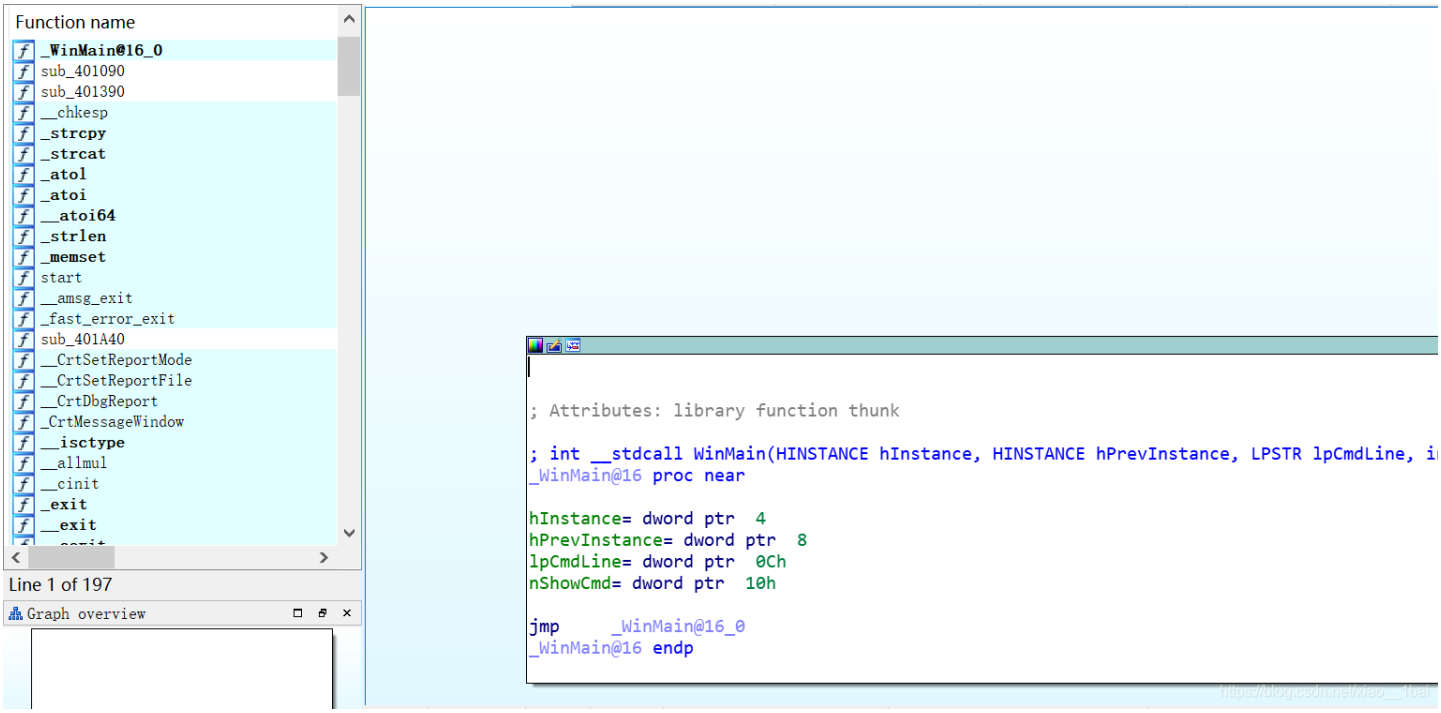


下载附件，是一个有百度图案字样的东西：

照例扔入exeinpefo中查看信息：

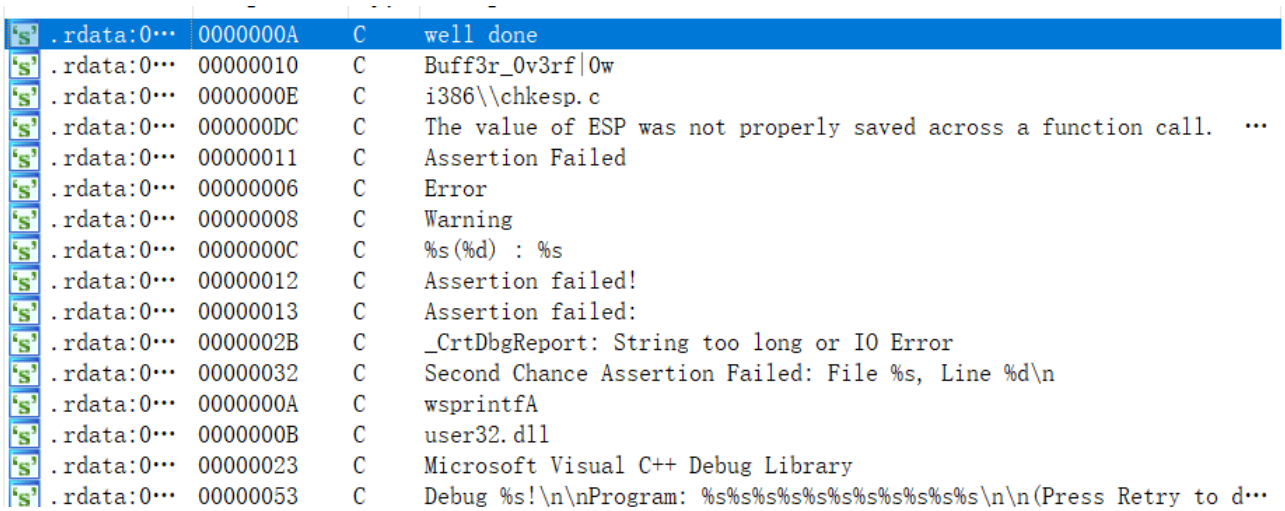


W32可执行文件，无壳，扔入IDA32中看伪代码判断题目类型：



没有主函数，看来是非正常文件，双击程序看看有什么信息可以提取：

一个输入密码型弹框，Crack按钮按不下去，信息够了，查看IDAstring窗口：



```
.rdata:0…  0000000A  C  \nModule:
.rdata:0…  00000008  C  \nFile:
.rdata:0…  00000008  C  \nLine:
.rdata:0…  0000000D  C  Expression:
.rdata:0…  00000073  C  \n\nFor information on how your program can cause an assertion\nf…
.rdata:0…  00000017  C  <program name unknown>
.rdata:0…  00000009  C  dbgrpt.c
.rdata:0…  00000016  C  szUserMessage != NULL
.rdata:0…  0000000A  C  stdenvp.c
.rdata:0…  0000000A  C  stdargv.c
.rdata:0…  00000008  C  a_env.c
.rdata:0…  00000009  C  ioinit.c
.rdata:0…  00000017  C  __GLOBAL_HEAP_SELECTED
.rdata:0…  00000015  C  __MSVCRT_HEAP_SELECT
```


没有发现input the password字眼，应该是隐藏了，想起弹框是用了Messagebox的windowsAPI函数，于是双击跟踪该函数：（要在import窗口才能跟踪，string窗口不行，因为import是导入API外部函数的窗口,string窗口那里可能只是刚好有同名字符串而已）

```
0042A254  MultiByteToWideChar  KERNEL32
0042A258  GetStringTypeA       KERNEL32
0042A25C  GetStringTypeW       KERNEL32
0042A260  IsBadWritePtr        KERNEL32
0042A264  IsBadReadPtr         KERNEL32
0042A268  HeapValidate         KERNEL32
0042A26C  GetCPInfo            KERNEL32
0042A270  GetACP               KERNEL32
0042A274  GetOEMCP             KERNEL32
0042A278  CloseHandle          KERNEL32
0042A2D0  DestroyWindow        USER32
0042A2D4  PostQuitMessage      USER32
0042A2D8  GetDlgItemTextA      USER32
0042A2DC  MessageBoxA          USER32
0042A2E0  SetTimer             USER32
0042A2E4  KillTimer            USER32
0042A2E8  DialogBoxParamA      USER32
```



```
{
  if ( a3 == 1000 )
  {
    GetDlgItemTextA(hWnd, 1002, &String, 260);
    strlen(&String);
    if ( strlen(&String) > 6 )
      ExitProcess(0);
    v10 = atoi(&String) + 1;
    if ( v10 == 123 && v12 == 120 && v14 == 122 && v13 == 121 )
    {
      strcpy(Text, "flag");
      memset(&v7, 0, 0xFCu);
      v8 = 0;
      v9 = 0;
      _itoa(v10, &v5, 10);
      strcat(Text, "{");
      strcat(Text, &v5);
      strcat(Text, "_");
      strcat(Text, "Buff3r_0v3rf|0w");
      strcat(Text, "}");
```

```
        MessageBoxA(0, Text, "well done", 0);
      }
      SetTimer(hWnd, 1u, 0x3E8u, TimerFunc);
    }
    if ( a3 == 1001 )
      KillTimer(hWnd, 1u);
  }
  return 0;
}
```


逻辑很简单，真的简单，但我就是错了：
错误1：习惯性的字符串反转，这里不是内存操作，就是打印Text这个字符串，所以不用反转：

```
MessageBoxA(0, Text, "well done", 0);
```

错误2：我把v10的123转成ASCII字符了，对应的字符是{，于是我就得到一个神奇的flag：flag{{_Buff3r_0v3rf|0w}
这个当然是错的，关键是我还直接以为这是假的flag代码而去寻找其它函数去了。

后面就引发了一系列问题，比如设想Crack按钮按不下是不是要动态调整调整跳转等等：

```
7600F913   E9 48FEFFFF   jmp   KERNEL32.7600F760
7600F918   83E9 02       sub   ecx, 0x2
7600F91B   EB F1         jmp   XKERNEL32.7600F90E
7600F91D   8BFF          mov   edi, edi
7600F91F   53            push  ebx
7600F920   33DB          xor   ebx, ebx
7600F922   2BD3          sub   edx, ebx
7600F924   74 5F         je    XKERNEL32.7600F985
7600F926   83EA 01       sub   edx, 0x1
7600F929   75 52         jnz   XKERNEL32.7600F97D
7600F92B   FF15 600B0776 call  dword ptr ds:[<&KERNELBASE.KernelBaseGe KERNELBA.Ke
7600F931   8958 18       mov   dword ptr ds:[eax+0x18], ebx
7600F934   64:A1 30000000 mov  eax, dword ptr fs:[0x30]
7600F93A   64:8B0D 30000000 mov ecx, dword ptr fs:[0x30]
7600F941   8B40 54       mov   eax, dword ptr ds:[eax+0x54]
7600F944   8B50 08       mov   edx, dword ptr ds:[eax+0x8]
7600F947   64:A1 30000000 mov  eax, dword ptr fs:[0x30]
7600F94D   2B91 48020000 sub   edx, dword ptr ds:[ecx+0x248]
7600F953   0350 4C       add   edx, dword ptr ds:[eax+0x4C]
7600F956   8915 3C080A76 mov   dword ptr ds:[0x760A083C], edx
7600F95C   E8 69000000   call  KERNEL32.7600F9CA
7600F961   E8 35000000   call  KERNEL32.7600F99B
7600F966   85C0          test  eax, eax
7600F968   78 15         js    XKERNEL32.7600F97F
```


结果是失败的，我都不知道它是从那个函数跳出来的！！！

后来查了资料(WP)才发现，flag的确在这里，关键是{写成123即可，不用转ASCII字符，想想也对flag就是数字字符的结合啊！！！

所以最后flag：

```
flag{123_Buff3r_0v3rf|0w}
```

至于那个crack按钮为什么按不下，可能考点不在那吧~

总结：

> 错误1：习惯性的字符串反转，这里不是内存操作，就是打印Text这个字符串，所以不用反转：

> 错误2：我把v10的123转成ASCII字符了，对应的字符是{，于是我就得到一个神奇的flag：flag{{_Buff3r_0v3rf|0w}
> 这个当然是错的，关键是我还直接以为这是假的flag代码而去寻找其它函数去了。

> {写成123即可，不用转ASCII字符，想想也对flag就是数字字符的结合啊！！！

解毕！敬礼！