XCTF-EASYHOOK

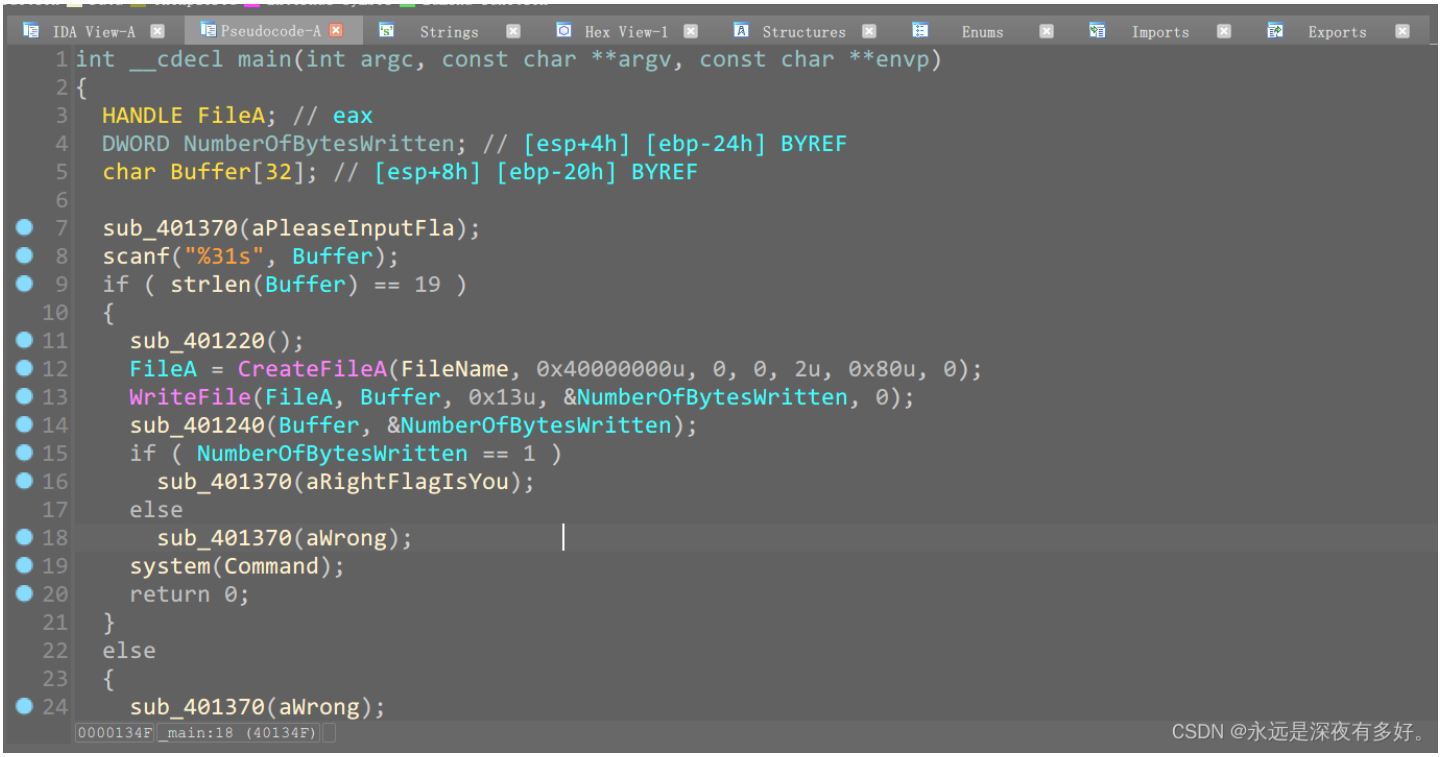
原创

永远是深夜有多好。 已于 2022-01-19 23:09:54 修改 109 收藏

分类专栏： XCTF 文章标签： 其他

于 2022-01-19 23:07:04 首次发布

本文链接：https://blog.csdn.net/qq_37370714/article/details/122590562




```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  HANDLE FileA; // eax
  DWORD NumberOfBytesWritten; // [esp+4h] [ebp-24h] BYREF
  char Buffer[32]; // [esp+8h] [ebp-20h] BYREF

  sub_401370(aPleaseInputFla);
  scanf("%31s", Buffer);
  if ( strlen(Buffer) == 19 )
  {
    sub_401220();
    FileA = CreateFileA(FileName, 0x40000000u, 0, 0, 2u, 0x80u, 0);
    WriteFile(FileA, Buffer, 0x13u, &NumberOfBytesWritten, 0);
    sub_401240(Buffer, &NumberOfBytesWritten);
    if ( NumberOfBytesWritten == 1 )
      sub_401370(aRightFlagIsYou);
    else
      sub_401370(aWrong);
    system(Command);
    return 0;
  }
  else
  {
    sub_401370(aWrong);
```

感觉对我而言有点复杂，但是还能看到关键在这几句

```
if ( strlen(Buffer) == 19 )
{
  sub_401220();
  FileA = CreateFileA(FileName, 0x40000000u, 0, 0, 2u, 0x80u, 0);
  WriteFile(FileA, Buffer, 0x13u, &NumberOfBytesWritten, 0);
  sub_401240(Buffer, &NumberOfBytesWritten);
  if ( NumberOfBytesWritten == 1 )
```

去看看`sub_401220`函数呢

```
int sub_401220()
{
```

```
{
  HMODULE LibraryA; // eax
  DWORD CurrentProcessId; // eax

  CurrentProcessId = GetCurrentProcessId();      // 百度了一下 这个是获取当前进程
  hProcess = OpenProcess(0x1F0FFFu, 0, CurrentProcessId);// 总感觉这个是hook
  LibraryA = LoadLibraryA(LibFileName);
  WriteFile_0 = (BOOL (__stdcall *)(HANDLE, LPCVOID, DWORD, LPDWORD, LPOVERLAPPED))GetProcAddress(LibraryA, Proc
  lpAddress = WriteFile_0;
  if ( !WriteFile_0 )
    return sub_401370(&unk_40A044);
  unk_40C9B4 = *(_DWORD *)lpAddress;
  *((_BYTE *)&unk_40C9B4 + 4) = *((_BYTE *)lpAddress + 4);
  byte_40C9BC = -23;
  dword_40C9BD = (char *)sub_401080 - (char *)lpAddress - 5;// 感觉应该在计算地址偏移量
  return sub_4010D0();
}
```


感觉没有头绪，动态调试看看
输入值后发现这句以后输入的值发生了变化

```
WriteFile(FileA, Buffer, 0x13u, &NumberOfBytesWritten, 0);
```

```
.text:00401322 call     ds:WriteFile
```

经过这步以后值改变了，说明可能hook就在里面，步入进去看看

```
EL32.DLL:757735B0 kernel32_WriteFile:
EL32.DLL:757735B0 jmp      sub_401080
EL32.DLL:757735B0 ;
```

```
{
  int v5; // ebx

  v5 = sub_401000(lpBuffer, nNumberOfBytesToWrite);
  sub_401140();
  WriteFile(hFile, lpBuffer, nNumberOfBytesToWrite, lpNumberOfBytesWritten
  if ( v5 )
    *lpNumberOfBytesWritten = 1;
  return 0;
```


到了`sub_401000`发现执行了此函数，我们输入的`lpBuffer`就发生了变化
那么就来重点分析一下这个函数

```
{
  if ( i == 18 )
  {
    *(_BYTE *)(a1 + 18) ^= 0x13u;             // a[18]末尾要异或一个0x13
  }
  else
  {
    if ( i % 2 )
      v3 = *(_BYTE *)(i + a1) - i;            // 奇数 v3 = a[i] - i
    else
      v3 = *(_BYTE *)(i + a1 + 2);            // 偶数 v3 = a[i+2]
    *(_BYTE *)(i + a1) = i ^ v3;              // 处理完奇偶然再 处理 a[i] = i ^ v3 此时的a[i]装着处理后的结果
  }
}
v4 = 0;
if ( a2 <= 0 )
  return 1;
v5 = 0;
while ( byte_40A030[v5] == *(_BYTE *)(v5 + a1) )// 用a[v5] 和 byte_40A030[v5]作比较
{
  v5 = ++v4;
  if ( v4 >= a2 )
    return 1;
```


可以看到sub_401000返回的是判断是否为真flag的依据

```
v5 = sub_401000(lpBuffer, nNumberOfBytesToWrite);
sub_401140();
```

这有一个假的flag

```
int __cdecl sub_401240(const char *a1, _DWORD *a2)
{
  int result; // eax
  unsigned int v3; // kr04_4
  char v4[24]; // [esp+Ch] [ebp-18h] BYREF

  result = 0;
  strcpy(v4, "This_is_not_the_flag");
  v3 = strlen(a1) + 1;
  if ( (int)(v3 - 1) > 0 )
  {
    while ( v4[a1 - v4 + result] == v4[result] )
    {
      if ( ++result >= (int)(v3 - 1) )
      {
        if ( result == 21 )
        {
```

然后逆向写出步骤

```
------
0lag{HoOk_w1th_Fun}
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
```

```
File  Edit  Format  Run  Options  Window  Help
tmp = [0x61, 0x6A, 0x79, 0x67, 0x6B, 0x46, 0x6D, 0x2E, 0x7F,
       0x5F, 0x7E, 0x2D, 0x53, 0x56, 0x7B, 0x38, 0x6D, 0x4C, 0x6E]
flag = list('012345678901234567')
#print (len(tmp)) 19位
for i in range(18):
    if i % 2:
        flag[i] = chr((tmp[i] ^ i) + i)
    else:
        flag[i+2] = chr(tmp[i] ^ i)
    if i==18:
        flag[i] = chr(flag[i] ^0x13)
print ("".join(flag))
```


当i等于0的时候 执行flag[2] = tmp[0] ^ i 而没有任何数据存储在flag[0]导致flag开头是0
把0换成f就对了

> 第一次做hook的题，有点头大不过还是有点思路。
> hook我看来就是放一个警报器，来探测某功能代码，如果执行代码并且遇到了警报器的报警范围，那么就把他勾出来，紧急处理后继续放回去处理前面没处理的事情，有点像中断。