




【WriteUp】【入门】攻防世界_REVERSE_hello,CTF

原创

[E_stream](#)  于 2021-11-03 17:52:56 发布  76  收藏

分类专栏: [REVERSE](#) 文章标签: [系统安全](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 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_45283376/article/details/121125026

版权



[REVERSE](#) 专栏收录该内容

5 篇文章 0 订阅

订阅专栏

解决逆向的题大都需要IDA软件, 没下载软件的童鞋可以去这篇博客看看:

https://blog.csdn.net/re_psyche/article/details/78797689

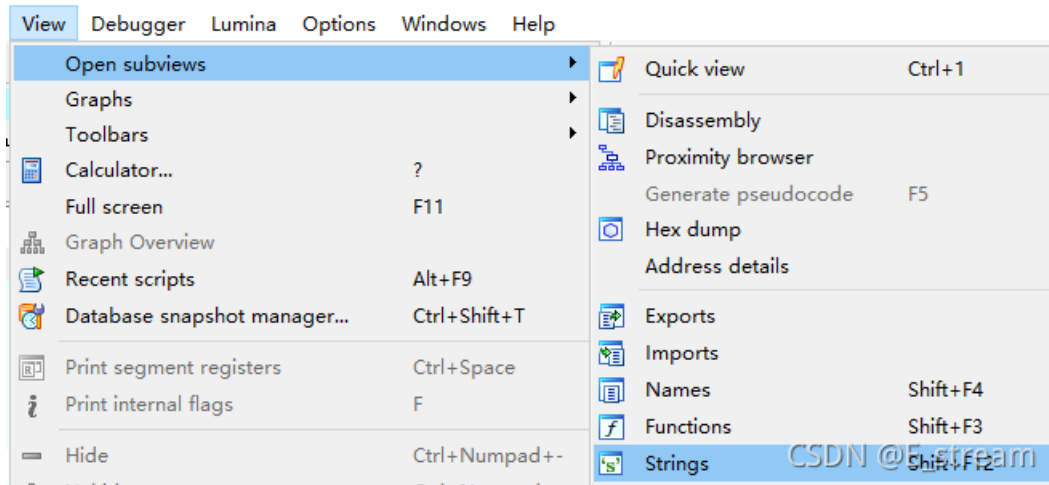
打开可执行文件（去攻防世界REVERSE板块下载，第一题就是）

Icon	Filename	Date	Time	Type	Size
📁	18a51cbc365c488f89c9fdee59868ea5...	2021-11-03	15:49	应用程序	44 KB
📄	18a51cbc365c488f89c9fdee59868ea5...	2021-11-03	15:51	ID0 文件	16 KB
📄	18a51cbc365c488f89c9fdee59868ea5...	2021-11-03	15:51	ID1 文件	0 KB
📄	18a51cbc365c488f89c9fdee59868ea5...	2021-11-03	15:51	ID2 文件	1 KB
📄	18a51cbc365c488f89c9fdee59868ea5...	2021-11-03	15:51	NAM 文件	0 KB
📄	18a51cbc365c488f89c9fdee59868ea5...	2021-11-03	15:51	TIL 文件	1 KB

随便输入字符串，提示wrong！并且发现这个输入还是个死循环。（这点比较重要，在后面看伪代码的时候有用）

```
please input your serial:12233
wrong!
please input your serial:mmm_L
wrong!
please input your serial:█
```

用IDA对可执行文件进行分析，我一般上来直接Shift+F12，在字符串子窗口看看有没有“可疑”的字符串。



下面是字符串界面的部分截图，可以看到第一行的字符串就是我们打开执行文件后的输入提示。而下面这一长串数字，是比较可疑的。有的REVERSE签到题（很简单的那种），可能这串数字就是flag了。但这个题，他穿“衣服”了，初步猜测这组十六进制数字是加密了。但至于用的什么加密算法，我们还得继续探索。

```
's' .data:0040804C 0000001A C please input your serial:
's' .data:00408068 00000023 C 437261636b4d654a757374466f7246756e
```

（当然，如果你比较有经验，知道一般flag都是字符串，那你很容易就联想到了十六进制转字符串，那就大功告成！）

在汇编语言界面（IDA View-A）单击main函数，按 tab 键（或者F5）查看函数的伪代码（汇编语言基础不扎实的话，看汇编很痛苦，伪代码类似C语言，更容易理解函数的作用）。

```
.text:00401000 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:00401000 _main          proc near          ; CODE XREF: start+AF↓
.text:00401000
```

这个界面就是main函数的伪代码界面了。现在重头戏才开始！



```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int i; // ebx
4     char v4; // al
5     int result; // eax
6     int v6; // [esp+0h] [ebp-70h]
7     int v7; // [esp+0h] [ebp-70h]
8     char Buffer[2]; // [esp+12h] [ebp-5Eh] BYREF
9     char v9[20]; // [esp+14h] [ebp-5Ch] BYREF
10    char v10[32]; // [esp+28h] [ebp-48h] BYREF
11    __int16 v11; // [esp+48h] [ebp-28h]
12    char v12; // [esp+4Ah] [ebp-26h]
13    char v13[36]; // [esp+4Ch] [ebp-24h] BYREF
14
15    strcpy(v13, "437261636b4d654a757374466f7246756e");
16    while ( 1 )
17    {
18        memset(v10, 0, sizeof(v10));
19        v11 = 0;
20        v12 = 0;
21        sub_401348(aPleaseInputYou, v6);
22        scanf("%s", v9);
23        if ( strlen(v9) > 0x11 )
24            break;
25        for ( i = 0; i < 17; ++i )
26        {
27            v4 = v9[i];
28            if ( !v4 )
```

前几行看不懂没关系，我理解的是变量的初始化工作等等。咱们主要看函数啊，最“扎眼”的就是第15行咯。strcpy()是一个用来复制字符串的函数，而第一个参数是一个长度为36的字符数组，第二个参数就是那个穿衣服的flag了。

```
strcpy(v13, "437261636b4d654a757374466f7246756e");
```

下面就是一个while(1)的死循环，前面我们说过，这个死循环很重要。因为在打开可执行文件后，我们随便输入的字符串在显示“wrong!”后还会提醒你继续输入。这就可以判断，flag的加密就在循环中。



```
15 strcpy(v13, "437261636b4d654a757374466f7246756e");
16 while ( 1 )
17 {
18     memset(v10, 0, sizeof(v10));
19     v11 = 0;
20     v12 = 0;
21     sub_401348(aPleaseInputYou, v6);
22     scanf("%s", v9);
23     if ( strlen(v9) > 0x11 )
24         break;
25     for ( i = 0; i < 17; ++i )
26     {
27         v4 = v9[i];
28         if ( !v4 )
29             break;
30         sprintf(Buffer, "%x", v4);
31         strcat(v10, Buffer);
32     }
33     if ( !strcmp(v10, v13) )
34         sub_40134B(aSuccess, v7);
35     else
36         sub_40134B(aWrong, v7);
37 }
```

第22行，一个scanf()再熟悉不过了，这个函数就是用来获取咱们的输入并把字符串赋值给v9变量。

第23行，一个判断语句，如果v9的长度大于0x11(即大于17)，就break（退出死循环）

第25行，for循环用来把v9的值再赋给v4。

第30行，我没咋接触过sprintf()，但可以大概猜出，是把v4中的字符串以"%x"(即16进制)的格式放入Buffer中。

第31行，strcat()我熟啊，又把Buffer的十六进制数放到了v10中。

大概总结一下就是，这个for循环先把存放我们输入的v9赋值给了v4，再把v4以16进制的形式（从字符串到16进制）存放到v10。

第33到第36行很简单，如果v10（咱们输入的字符串转变成的16进制数）和v13（那个穿衣服的flag）相同，那就成功了！

至此结束，解密是加密的逆过程（16进制->字符串）。

```
please input your serial:CrackMeJustForFun
success!
please input your serial:█
```