攻防世界 Mysterious

原创

别害怕我在 于 2021-08-12 09:28:52 发布 72 收藏

分类专栏： CTF逆向reverse新手

本文链接：https://blog.csdn.net/afanzcf/article/details/119633674

---

title: 攻防世界 Mysterious
date: 2021年8月11日 20点30分
tags: 攻防世界
categories: 攻防世界

这个题真是差点就自己解出来了，卡在了最后一步，看了别人wp之后，有点懊悔。

先讲讲自我分析的过程

自我分析

1、PE分析

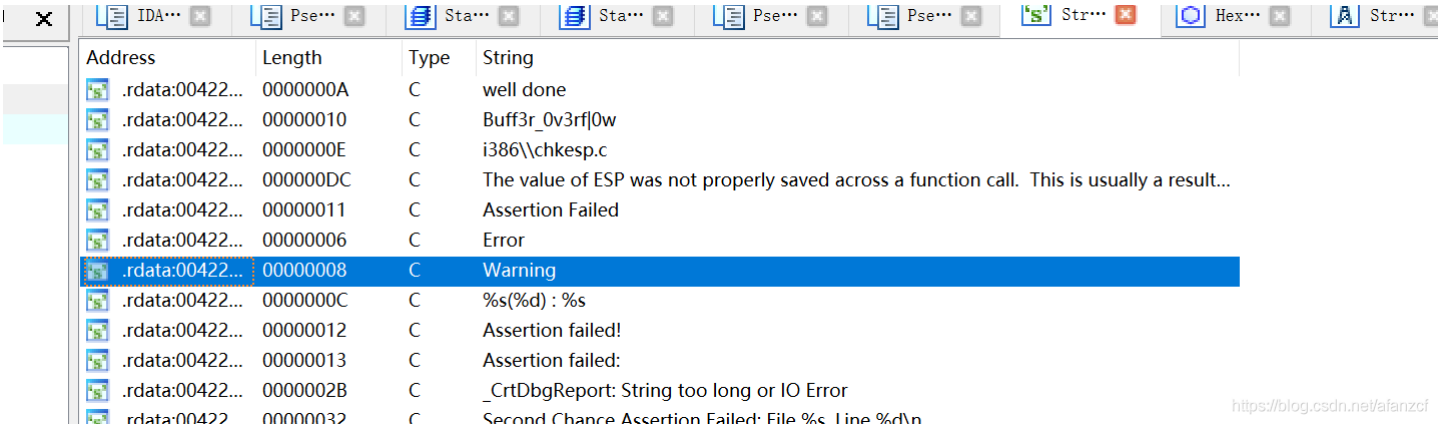
32位。

2、IDApro

IDA之前，先看看这个软件，打开发现是一个弹窗。

随便输入一串字母，发现点不动。直接上手IDA

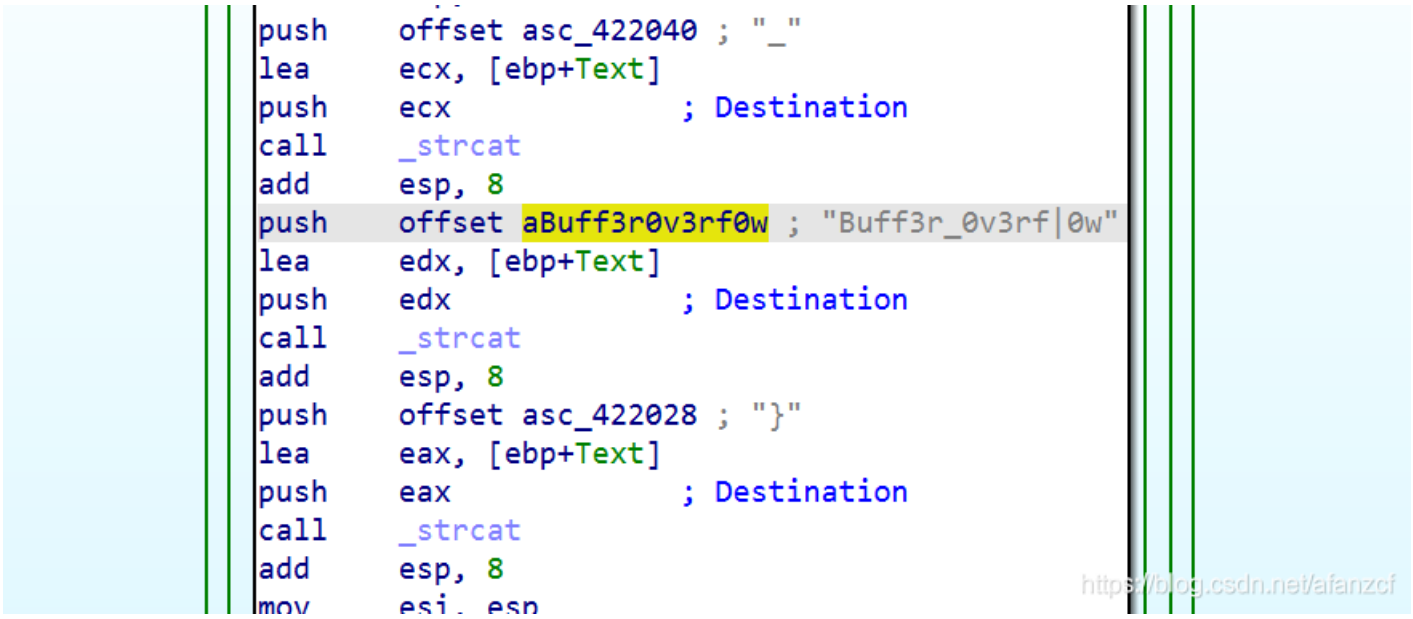
（1）shift + F12查看字符串窗口



这里看到，有一个well done，和Buff3r_0v3rf|0w，我分析的时候，是看的Buff3r_0v3rf|0w，这两个一样的

（2）ctrl + x交叉引用

```
.rdata:00422028 asc_422028      db '}',0                ; DATA XREF: sub_401090+1E7↑o
.rdata:0042202A                 align 4
.rdata:0042202C ; const char aBuff3r0v3rf0w[]
.rdata:0042202C aBuff3r0v3rf0w  db 'Buff3r_0v3rf|0w',0  ; DATA XREF: sub_401090+1D3↑o
.rdata:0042203C                 align 10h
.rdata:00422040 ; const char asc_422040[]
.rdata:00422040 asc_422040      db '_',0                ; DATA XREF: sub_401090+1BF↑o
.rdata:00422042                 align 4
.rdata:00422044 ; const char Source[]
.rdata:00422044 Source          db '{',0                ; DATA XREF: sub_401090+195↑o
.rdata:00422046                 db    0
.rdata:00422047                 db    0
.rdata:00422048                 db    0
```


push offset asc_422040 ; "_"
lea ecx, [ebp+Text]
push ecx ; Destination
call _strcat
add esp, 8
push offset aBuff3r0v3rf0w ; "Buff3r_0v3rf|0w"
lea edx, [ebp+Text]
push edx ; Destination
call _strcat
add esp, 8
push offset asc_422028 ; "}"
lea eax, [ebp+Text]
push eax ; Destination
call _strcat
add esp, 8
mov esi, esp

（3）F5反汇编

```
int __stdcall sub_401090(HWND hWnd, int a2, int a3, int a4)
{
  int v4; // eax
  char Source[260]; // [esp+50h] [ebp-310h] BYREF
  CHAR Text[5]; // [esp+154h] [ebp-20Ch] BYREF
  char v8[252]; // [esp+159h] [ebp-207h] BYREF
  __int16 v9; // [esp+255h] [ebp-10Bh]
  char v10; // [esp+257h] [ebp-109h]
  int Value; // [esp+258h] [ebp-108h]
  CHAR String[260]; // [esp+25Ch] [ebp-104h] BYREF

  memset(String, 0, sizeof(String));            // 初始化
  Value = 0;
  if ( a2 == 16 )
  {
    DestroyWindow(hWnd);                        // 函数功能：销毁指定的窗口。
                                                // 这个函数通过发送WM_DESTROY 消息和 WM_NCDESTROY 消息使窗口无效并移除其键盘焦点。
                                                // 这个函数还销毁窗口的菜单，清空线程的消息队列，
                                                // 销毁与窗口过程相关的定时器，解除窗口对剪贴板的拥有权，打断剪贴板器的查看链。
    PostQuitMessage(0);                         // PostQuitMessage，函数名。该函数向系统表明有个线程有终止请求。通常用来响应WM_DESTROY消息
  }
  else if ( a2 == 273 )
  {
    if ( a3 == 1000 )
    {
      GetDlgItemTextA(hWnd, 1002, String, 260); // GetDlgItemText是C++中的函数，
                                                // 调用这个函数以获得与对话框中的控件相关的标题或文本。
                                                // GetDlgItemText成员函数
                                                // 将文本拷贝到lpStr指向的位置并返回拷贝的字节的数目。
      strlen(String);
      if ( strlen(String) > 6 )
        ExitProcess(0);                         // ExitProcess 结束调用的进程及其所有的线程
                                                // windows函数，用于多线程编程
      v4 = atoi(String);                        // v4这里有一个加密 把字符串转换成整型数的一个函数，应用在计算机程序和办公软件中。
                                                // int atoi(const char *nptr) 函数会扫描参数 nptr字符串，会跳过前面的空白字符（例如空格，
                                                // 如果 nptr不能转换成 int 或者 nptr为空字符串，那么将返回 0
      Value = v4 + 1;
      if ( v4 == 122 && String[3] == 'x' && String[5] == 'z' && String[4] == 'y' )
      {
        strcpy(Text, "flag");
        memset(v8, 0, sizeof(v8));              // 初始化，下面有一个加密，结果存到Source里面
        v9 = 0;
        v10 = 0;
        _itoa(Value, Source, 10);               // a2 = 273，a3 = 1000， char *itoa( int value, char *string,int radix); [1]
                                                // 原型说明：
                                                // value：欲转换的数据。
                                                // string：目标字符串的地址。 也就是Source
                                                // radix：转换后的进制数，可以是10进制、16进制等。
        strcat(Text, "{");
        strcat(Text, Source);
        strcat(Text, "_");
        strcat(Text, "Buff3r_0v3rf|0w");
        strcat(Text, "}");
        MessageBoxA(0, Text, "well done", 0);   // MessageBox指的是显示一个模态对话框，
                                                // 其中包含一个系统图标、 一组按钮和一个简短的特定于应用程序消息，
                                                // 如状态或错误的信息。消息框中返回一个整数值，该值指示用户单击了哪个按钮。
      }
      SetTimer(hWnd, 1u, 1000u, TimerFunc);
    }
    if ( a3 == 1001 )                           // SetTimer是一种API函数，位于user32.dll中。
                                                // 你想每隔一段时间执行一件事的的时候，你可以使用它。
                                                // 使用定时器的方法比较简单，通常告诉Windows一个时间间隔，
                                                // 然后Windows以此时间间隔周期性触发程序。
                                                // 通常有两种方法来实现：发送WM_TIMER消息和调用应用程序定义的回调函数。
                                                // 不需要指定定时器时，可以调用对应的KillTimer函数销毁指定的时钟。
      KillTimer(hWnd, 1u);
  }
  return 0;
}
```

000012D0 sub_401090:62 (4012D0)

这里，也是百度了好几个函数。

```
Value = v4 + 1;
if ( v4 == 122 && String[3] == 'x' && String[5] == 'z' && String[4] == 'y' )
{
  strcpy(Text, "flag");
  memset(v8, 0, sizeof(v8));                // 初始化，下面有一个加密，结果存到Source里面
  v9 = 0;
  v10 = 0;
  _itoa(Value, Source, 10);                 // a2 = 273，a3 = 1000，  char *itoa( int value, char *
                                            // 原型说明：
                                            // value：欲转换的数据。
                                            // string：目标字符串的地址。 也就是Source
                                            // radix：转换后的进制数，可以是10进制、16进制等。
  strcat(Text, "{");
  strcat(Text, Source);
  strcat(Text, "_");
  strcat(Text, "Buff3r_0v3rf|0w");
  strcat(Text, "}");
  MessageBoxA(0, Text, "well done", 0);     // MessageBox指的是显示一个模态对话框，
                                            // 其中包含一个系统图标、 一组按钮和一个简短的特定于应用程
                                            // 如状态或错误的信息。消息框中返回一个整数值，该值指示用户
}
```


我分析到这里的时候，我知道是拼接字符{Source_Buff3r_0v3rf|0w}这个形式，但是Source我不知道是什么，可以看到，itoa函数里面有Source，百度一波，说的是这个函数是将，字符串转为整型数的。后面的那个10，是十进制的意思，value是欲转换的数据。

而value = v4 + 1；

```
                                            // int atoi(const char *nptr) 函数会扫描
                                            // 如果 nptr不能转换成 int 或者 nptr为空
  Value = v4 + 1;
  if ( v4 == 122 && String[3] == 'x' && String[5] == 'z' && String[4] == 'y' )
  {
    strcpy(Text, "flag");
    memset(v8, 0, sizeof(v8));              // 初始化，下面有一个加密，结果存到Source
    v9 = 0;
    v10 = 0;
    _itoa(Value, Source, 10);               // a2 = 273，a3 = 1000，  char *itoa( in
                                            // 原型说明：
                                            // value：欲转换的数据。
```


v4 = atoi（String）；

以为关键地方在这个函数中，百度一波，发现atoi也是将字符串转为整型数的函数。

跟进去之后，是一堆看不太懂的东西。

```
  const char *Stringa; // [esp+1Ch] [ebp+8h]

  while ( cbMultiByte <= 1 ? *((_WORD *)off_424A60 + *(unsigned __int8 *)String) & 8 : _isctype(
                                                                                         *(unsigned __int8 *)String,
                                                                                         8) )
    ++String;
  C = *(unsigned __int8 *)String;
  Stringa = String + 1;
  v4 = C;
  if ( C == '-' || C == '+' )
    C = *(unsigned __int8 *)Stringa++;
  v5 = 0;
  while ( cbMultiByte <= 1 ? *((_WORD *)off_424A60 + C) & 4 : _isctype(C, 4) )
  {
    v5 = 10 * v5 + C - 48;
    C = *(unsigned __int8 *)Stringa++;
  }
  if ( v4 == 45 )
    result = -v5;
  else
    result = v5;
  return result;
}
00001550 atol:7 (401550)
```


一开始，准备硬着头发分析这堆代码，但是我觉得应该没这么难。就先放一边了。

又去查了查上面的几个函数

```
{
  DestroyWindow(hWnd);                          // 函数功能：销毁指定的窗口。
                                                // 这个函数通过发送WM_DESTROY 消息和 WM_NCDESTROY 消息使窗口无效并移除其键盘焦点。
                                                // 这个函数还销毁窗口的菜单，清空线程的消息队列，
                                                // 销毁与窗口过程相关的定时器，解除窗口对剪贴板的拥有权，打断剪贴板器的查看链。
  PostQuitMessage(0);                           // PostQuitMessage，函数名。该函数向系统表明有个线程有终止请求。通常用来响应WM_DESTROY消
}
else if ( a2 == 273 )
{
  if ( a3 == 1000 )
  {
    GetDlgItemTextA(hWnd, 1002, String, 260); // GetDlgItemText是C++中的函数，
                                                // 调用这个函数以获得与对话框中的控件相关的标题或文本。
                                                // GetDlgItemText成员函数
                                                // 将文本拷贝到lpStr指向的位置并返回拷贝的字节的数目。
    strlen(String);
    if ( strlen(String) > 6 )
      ExitProcess(0);                           // ExitProcess 结束调用的进程及其所有的线程
                                                // windows函数，用于多线程编程
00010E7 sub_401090:19 (4010E7)
```


发现没什么用，又查了一下后面的函数，发现还是没什么用。

```
      strcat(Text, "Buff3r_0v3rf|0w");
      strcat(Text, "}");
      MessageBoxA(0, Text, "well done", 0);    // MessageBox指的是显示一个模态对话框，
                                               // 其中包含一个系统图标、 一组按钮和一个简短的特定于应用程序消息，
                                               // 如状态或错误的信息。消息框中返回一个整数值，该值指示用户单击了哪个按钮。
    }
    SetTimer(hWnd, 1u, 1000u, TimerFunc);
  }
  if ( a3 == 1001 )                            // SetTimer是一种API函数，位于user32.dll中。
                                               // 你想每隔一段时间执行一件事的的时候，你可以使用它。
                                               // 使用定时器的方法比较简单，通常告诉Windows一个时间间隔，
                                               // 然后Windows以此时间间隔周期性触发程序。
                                               // 通常有两种方法来实现：发送WM_TIMER消息和调用应用程序定义的回调函数。
                                               // 不需要指定定时器时，可以调用对应的KillTimer函数销毁指定的时钟。
    KillTimer(hWnd, 1u);
}
return 0;
```


思索一会之后，还是放弃了，查看了大哥的wp，顿时懂了。

查看大哥的wp

1、不太一样的地方

首先，我观察网上的wp，他们用的IDA反编译得到的伪代码，跟我的IDA反编译出来的不太一样。

这是网络上的

```
}
else if ( a2 == 273 )
{
  if ( a3 == 1000 )
  {
    GetDlgItemTextA(hWnd, 1002, &String, 260);// 获取输入
    strlen(&String);
    if ( strlen(&String) > 6 )                // 输入长度不能大于6
      ExitProcess(0);
    v10 = atoi(&String) + 1;                  // atoi把字符串转成整形，跟php的类型转换差不多
    if ( v10 == 123 && v12 == 'x' && v14 == 'z' && v13 == 'y' )
    {
      strcpy(Text, "flag");
      memset(&v7, 0, 0xFCu);
      v8 = 0;
      v9 = 0;
      _itoa(v10, &v5, 10);                    // itoa把字符串转为整形
      strcat(Text, "{");
      strcat(Text, &v5);
      strcat(Text, "_");
      strcat(Text, "Buff3r_0v3rf|0w");
      strcat(Text, "}");
      MessageBoxA(0, Text, "well done", 0);
```



```
22  {
23    if ( a3 == 1000 )
24    {
25      GetDlgItemTextA(hWnd, 1002, &String, 260);
26      strlen(&String);
27      if ( strlen(&String) > 6 )
28        ExitProcess(0);
29      v10 = atoi(&String) + 1;
30      if ( v10 == 123 && v12 == 120 && v14 == 122 && v13 == 121 )
31      {
32        strcpy(Text, "flag");
33        memset(&v7, 0, 0xFCu);
34        v8 = 0;
35        v9 = 0;
36        _itoa(v10, &v5, 10);
37        strcat(Text, "{");
38        strcat(Text, &v5);
39        strcat(Text, "_");
40        strcat(Text, "Buff3r_0v3rf|0w");
41        strcat(Text, "}");
42        MessageBoxA(0, Text, "well done", 0);
```

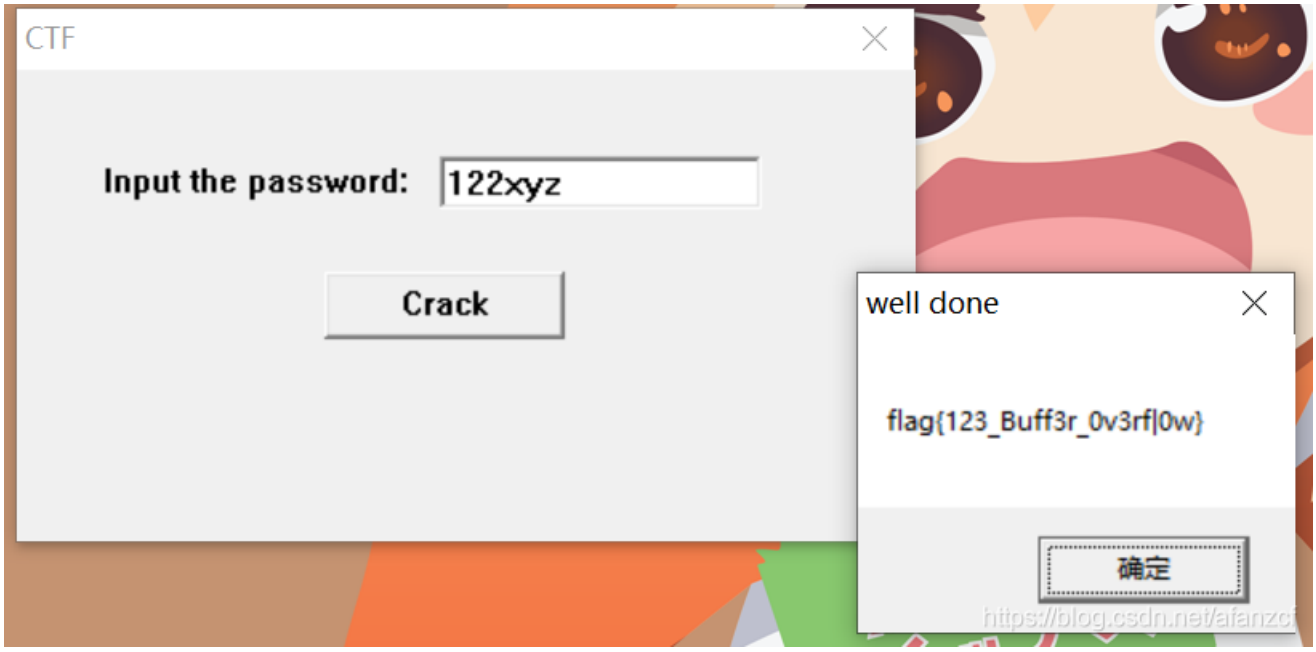

可以观察到，第一个地方就是我的IDA那里，没有v5这个东西，我的是Source，然后第二个地方，就是在if语句里面，他们的v4 = 123，我的v4 = 122.第三个地方是他们的IDA分析if语句里面xyz的时候，是v12，v14，v13，我的是string【3】，string【5】，string【4】.

2、上帝视角分析

其实当我站在上帝视角的时候，整个题已经通了。

我从一开始就忽略了if语句里面给的条件，我看到了string【3】，string【5】，string【4】，是xzy，我也变成了xyz。但是当时没想到，这是输入到弹窗里面的东西，站在上帝视角，可以看到if语句里面的条件是

122xyz，把这个输入到弹窗中。



得到flag。

知道flag之后，再来分析这个代码，

```
35                                      // int atoi(const char *nptr) 函数会扫描参数 nptr字符串，会跳过前面的空白字符（例如空格，
36                                      // 如果 nptr不能转换成 int 或者 nptr为空字符串，那么将返回 0
37      Value = v4 + 1;
38      if ( v4 == 122 && String[3] == 'x' && String[5] == 'z' && String[4] == 'y' )
39      {
40        strcpy(Text, "flag");
41        memset(v8, 0, sizeof(v8));            // 初始化，下面有一个加密，结果存到Source里面
42        v9 = 0;
43        v10 = 0;
44        _itoa(Value, Source, 10);             // a2 = 273, a3 = 1000,  char *itoa( int value, char *string,int radix); [1]
45                                              // 原型说明：
46                                              // value：欲转换的数据。
47                                              // string：目标字符串的地址。 也就是Source
48                                              // radix：转换后的进制数，可以是10进制、16进制等。
49        strcat(Text, "{");
50        strcat(Text, Source);
51        strcat(Text, "_");
```

这里可以看到，v4 = 122；而value = v4 + 1；也就是value = 123，而且这个itoa函数，就是将字符串转为整型数的，value是欲转换的数据，目标是Source，也就是Source = 123；然后拼接字符

```
{
  strcpy(Text, "flag");
  memset(v8, 0, sizeof(v8));
  v9 = 0;
  v10 = 0;
  _itoa(Value, Source, 10);



  strcat(Text, "{");
  strcat(Text, Source);
  strcat(Text, "_");
  strcat(Text, "Buff3r_0v3rf|0w");
  strcat(Text, "}");
  MessageBoxA(0, Text, "well done", 0);

}
```


也就是flag{123_Buff3r_03rf|0w},就得到了flag了。

总结

这个题最关键的地方，if语句里面的条件，就能得到flag。自己为什么不会做。

第一，Source是关键，itoa函数，当时也没注意value，导致卡在了Source，value = v4 +1 跟if语句里面的v4 = 122；没连接起来，导致value这里脱节，Source就跟着脱节。这里脱节之后，想法就到了v4 上面去了，一开始以为关键之处在v4 = atoi（String）这个地方，跟进之后，也确实有类似加密的代码。还准备分析一波。

其实还是自己想法简单了，题做的少了。继续加油。