看雪加密解密第一个traceme程序破解

原创

revercc 于 2020-01-16 23:47:29 发布 537 收藏 2

分类专栏： 逆向工程

本文链接：https://blog.csdn.net/qq_43628140/article/details/104011483

逆向工程 专栏收录该内容

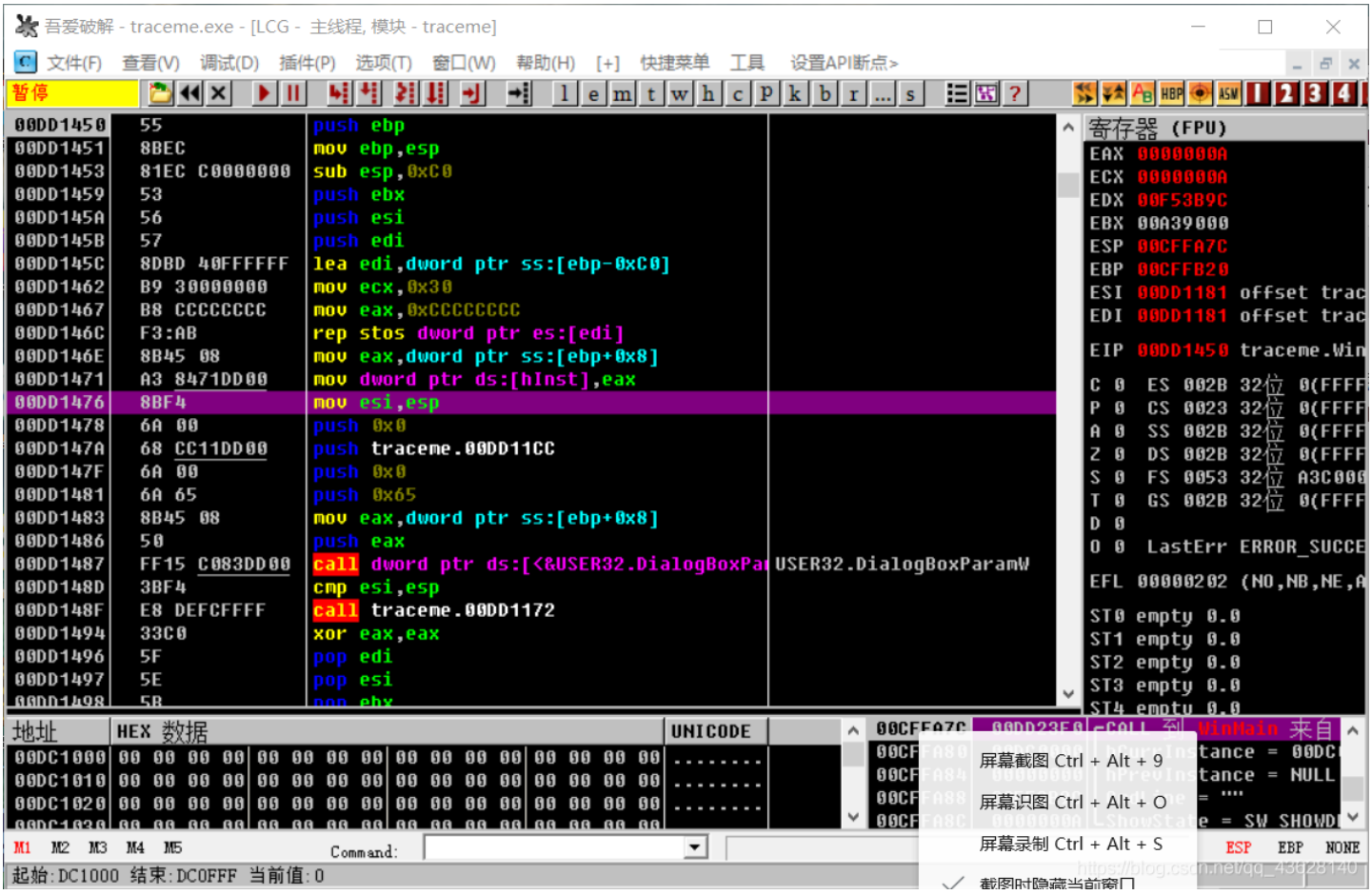
11 篇文章 0 订阅

订阅专栏

工具：ollydbg（吾爱破解2.10版）

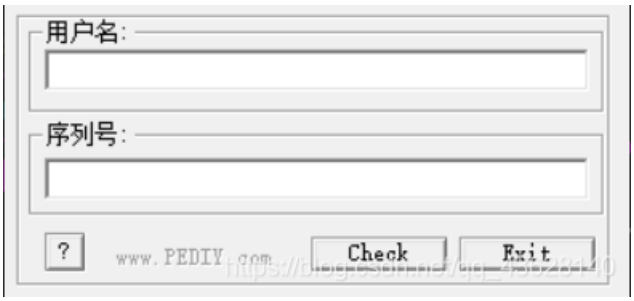
工具设置：因为traceme是一个win32图形用户程序，所以其程序入口点在WinMain（）函数处，设置ollydbg的调试设置的事件选项，选中在WinMain函数处中断。（选项->调试设置->事件）

正式开始，打开traceme程序后程序会停在winmain函数的入口处：

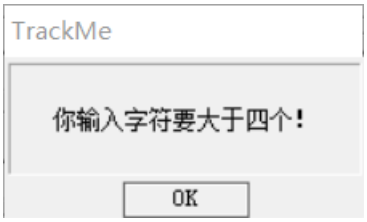


00DD1450即为WinMain函数的入口地址

运行一下程序发现弹出一个模态对话框，



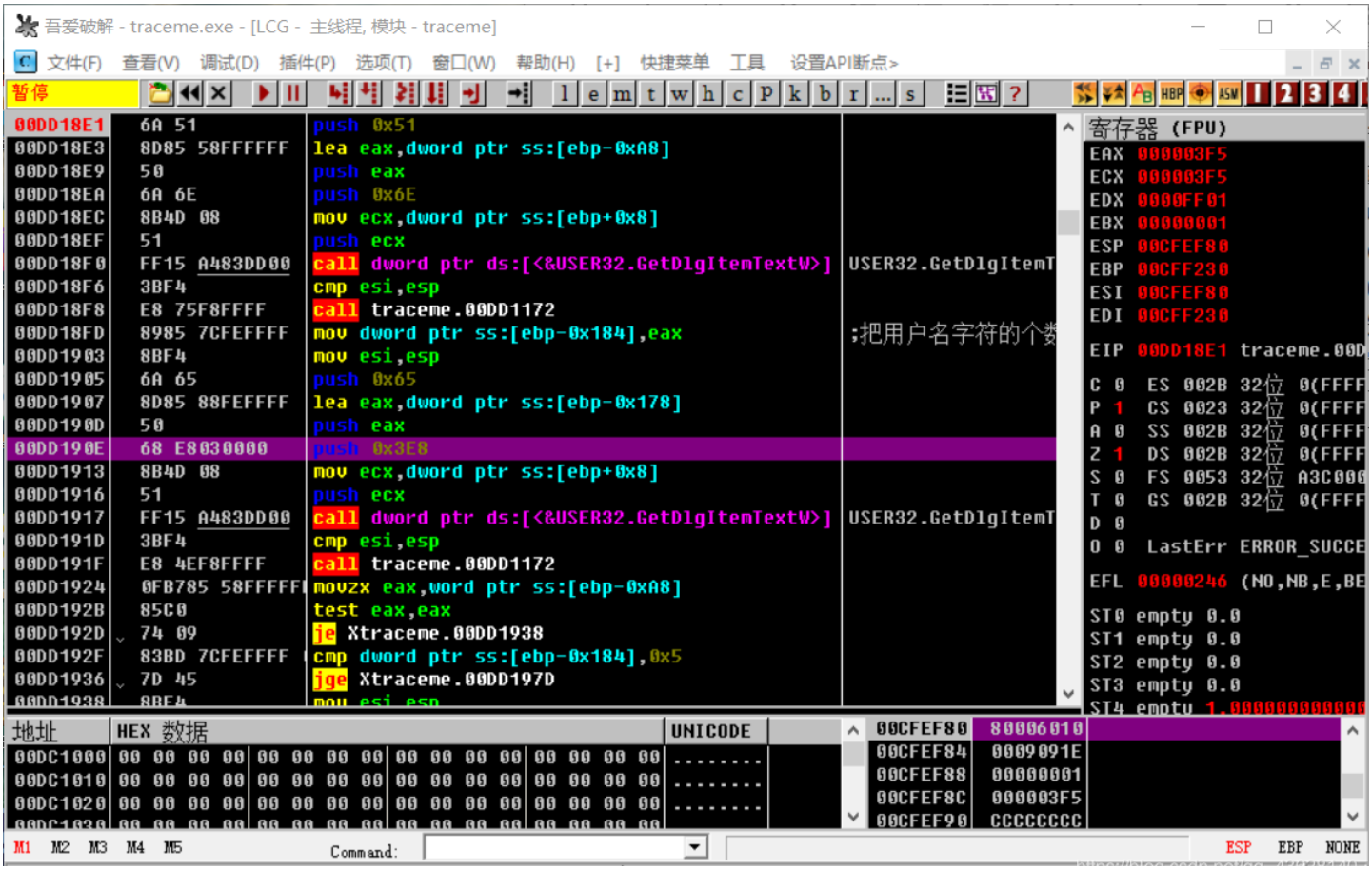
随便输入用户名和序列号点击check后发现弹出，



说明用户名要大于四个字符，点击ok后再次进行输入发现显示输入的序列号不对，大致了解了程序的流程，首先用户名要大于四个字符，然后当序列号不与用户名匹配时则弹出序列号错误，再来一次的消息框。

因为用户名在模态对话框输入后程序要获取它进行处理，对应的API函数为GetWindowTextW（unicode版本）或GetDlgItemTextA（ANSI）或GetDlgItemTextW（）

具体使用哪个函数不知道，只能多试几次，这就需要在ollydbg中设置断点是程序停在对应函数处，快捷键Ctry+G打开跟随表达式窗口输入相应函数并设置断点（也可以直接在设置API断点菜单项中直接设置断点），设置完断点后再运行，

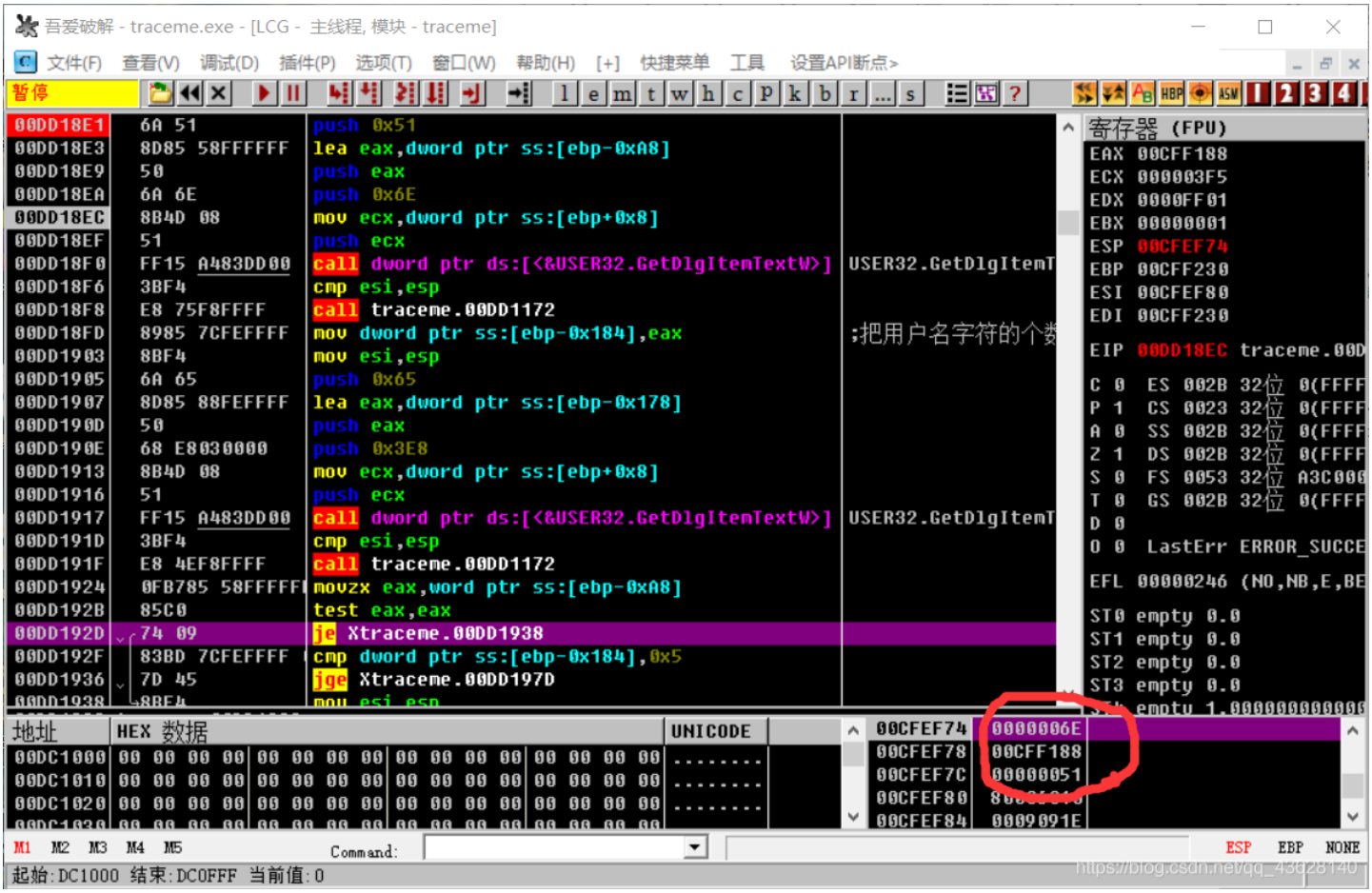


```
00DD190E -> 00DD1913 : (0x00000005  -::-  00000005) bytes   |   OFFSET : 0x00000D0E -> 0x00000D13
```

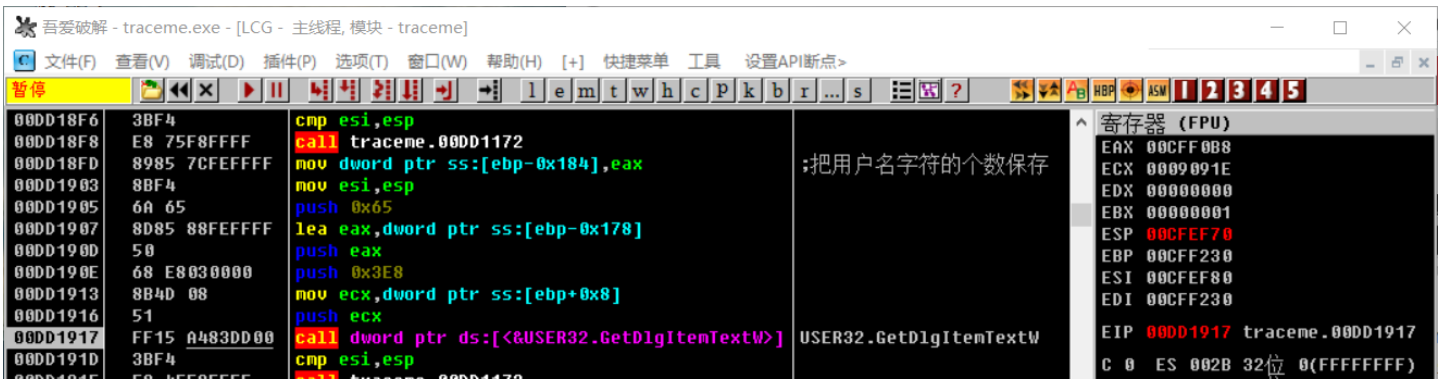
程序停在00DD18E1处，发现其先调用函数,参考GetDlgItemTextW函数的API档案

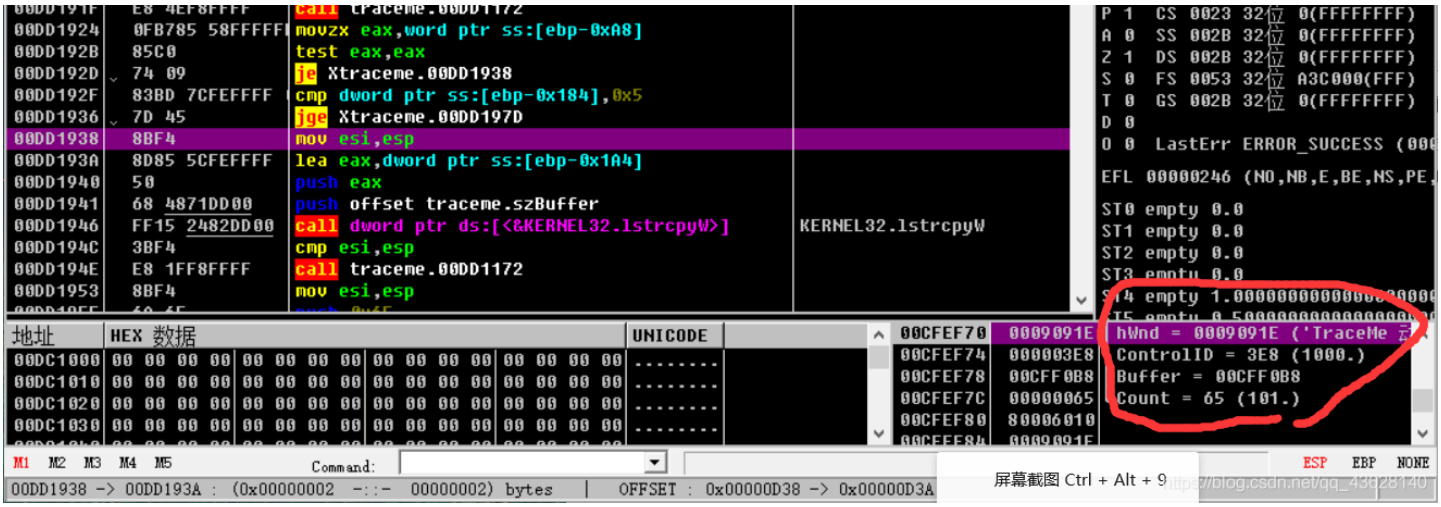
```
1 int WINAPI GetDlgItem(hwnd,IDC_EDITNAME,str,sizeof(str));
2 //第一个参数是窗口的句柄，第二个参数是要取得的控件内容的ID
3 //第三个参数就是字符串指针，第四个参数就是指针指向区域的大小
```

四个参数从右到左压入栈中（stdcall调用约定），按F7单步执行，记住用户名所存的缓冲区的地址为00CFF188,

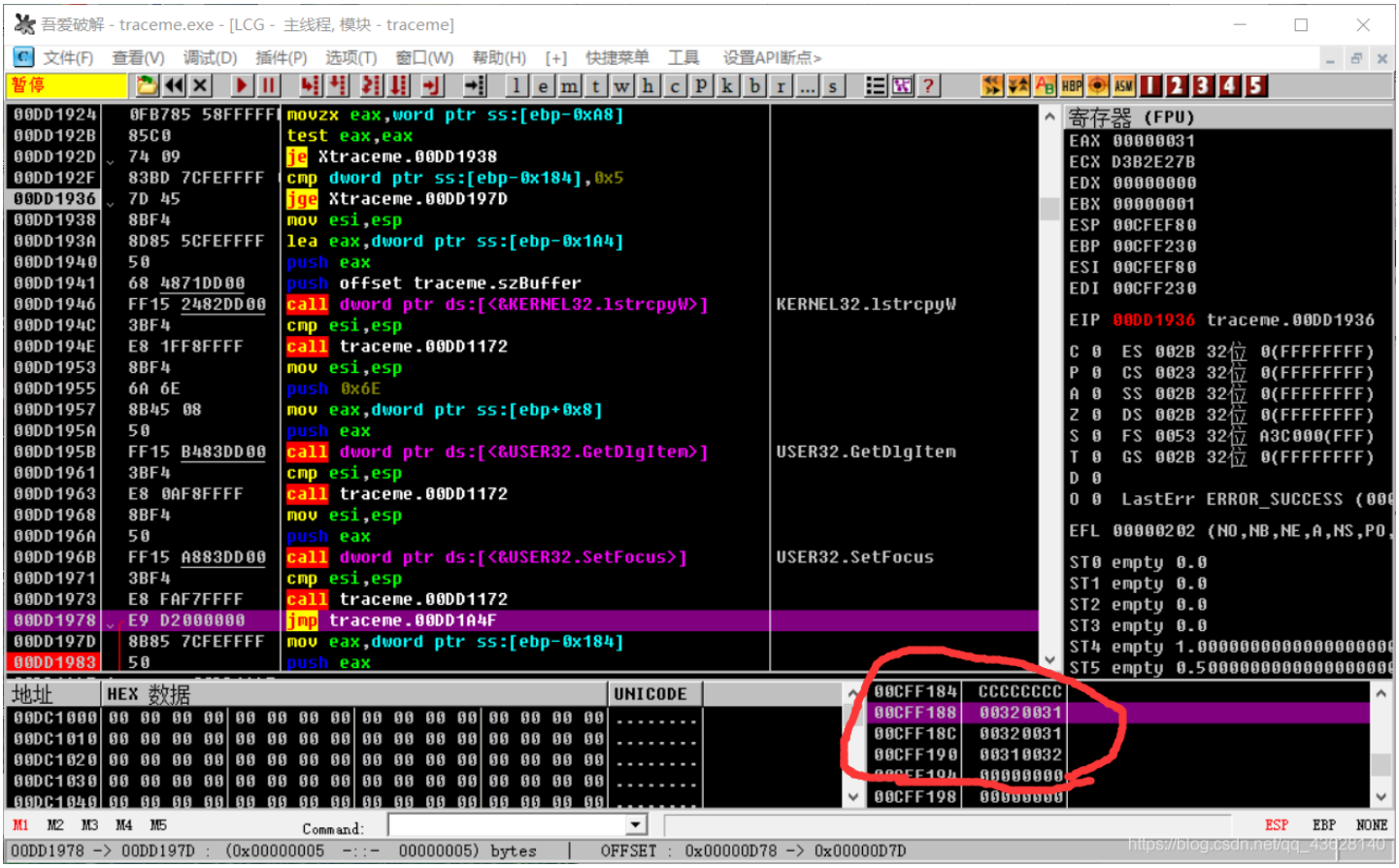


按F8单步步过call GetDlgItemTextW，运行完之后函数返回值在eax中为所读字符的个数，在F7继续运行
发现traceme.00DD1172只是判断一下esi与esp的大小，一样大就什么都做。在继续f7，
其将eax也就是所读字符的个数保存到指定单元
然后下面又是一个GetDlgItemTextW函数的调用其是获得序列号的字符，
ollydbg其在call函数的时候其会把API函数参数信息在栈中注释出来

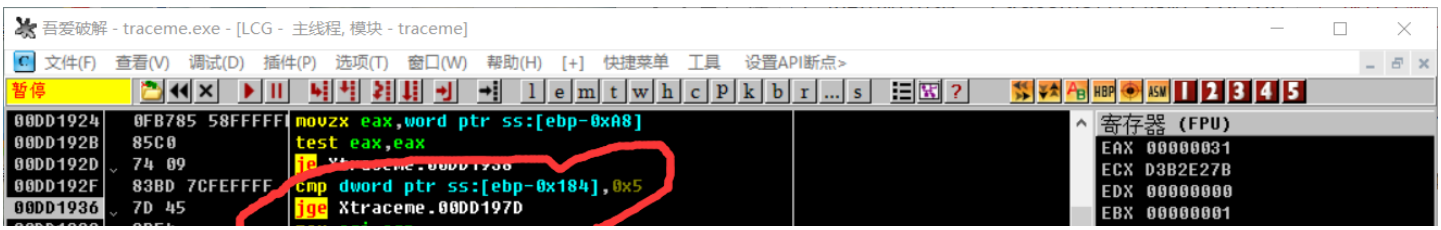


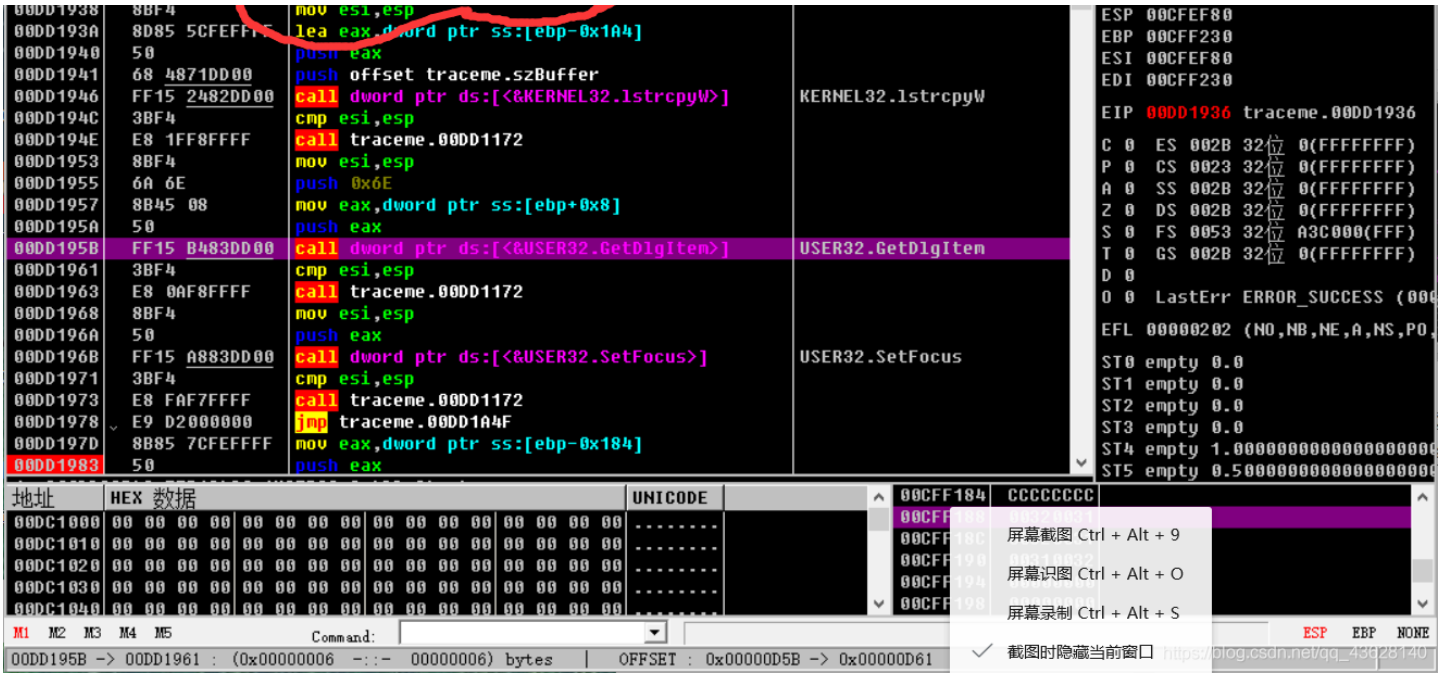


很容易知道其序列号字符存在地址为00CFF0B8处，
之后在继续F7发现其会用 movzx eax,word ptr ss:[ebp-0xA8]
test eax，eax；
当eax为0时其会跳到指定位置，如果把ip设置并跟进
其会显示输入字符大于4，说明eax不能为0
eax== ss:[ebp-0xA8]



也就是用户名字符缓冲区的第一个字符，不为零说明不能不能不填用户名，其字符得大于4,。

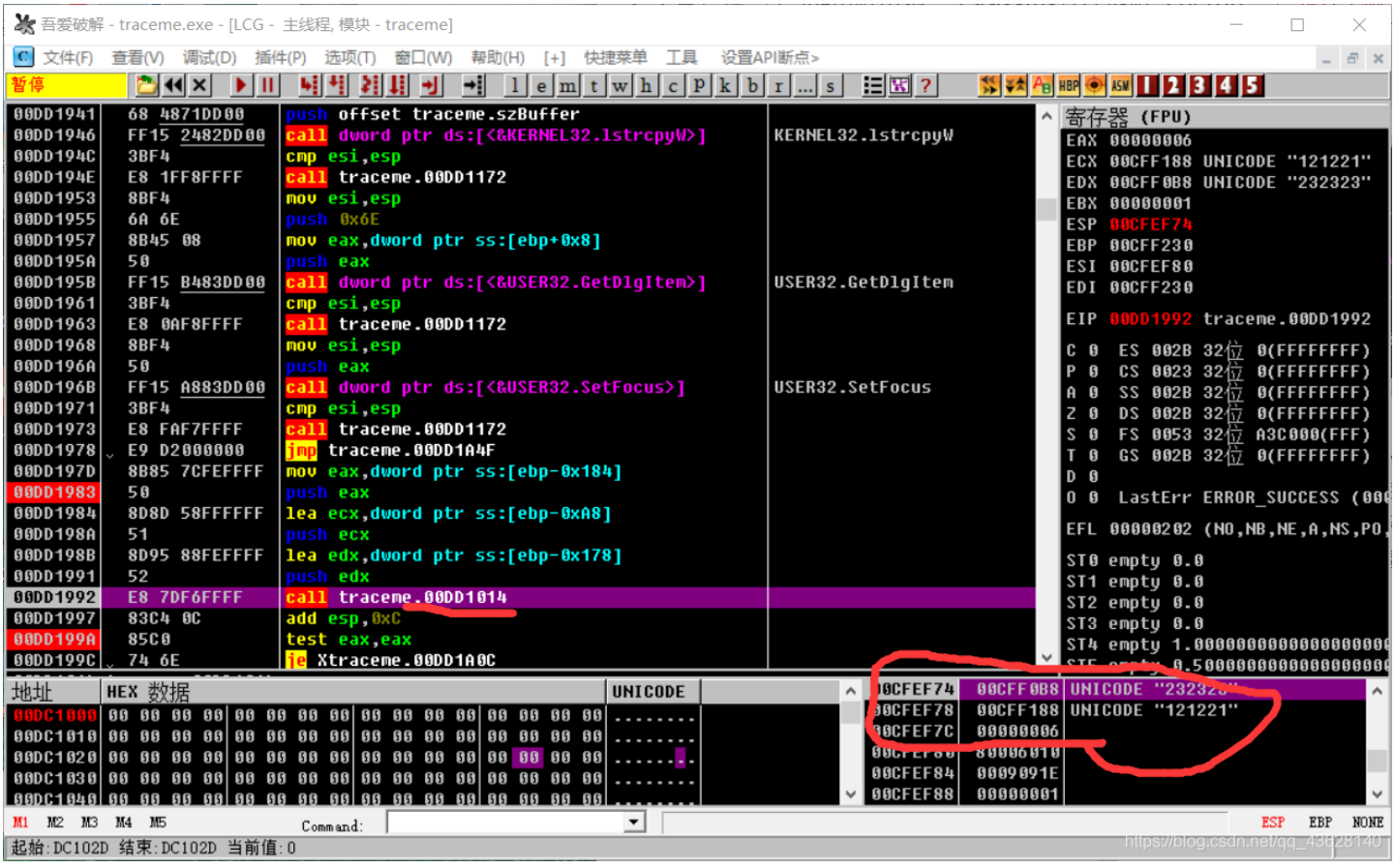




cmp dword ptr **ss:[ebp-0x184]**,0x5

此单元存放的为用户名字符数，当其字符大于等于5时跳转（否则就显示字符数不能小于5），

继续F7进入一个函数参数为用户名，序列，和用户名字符数



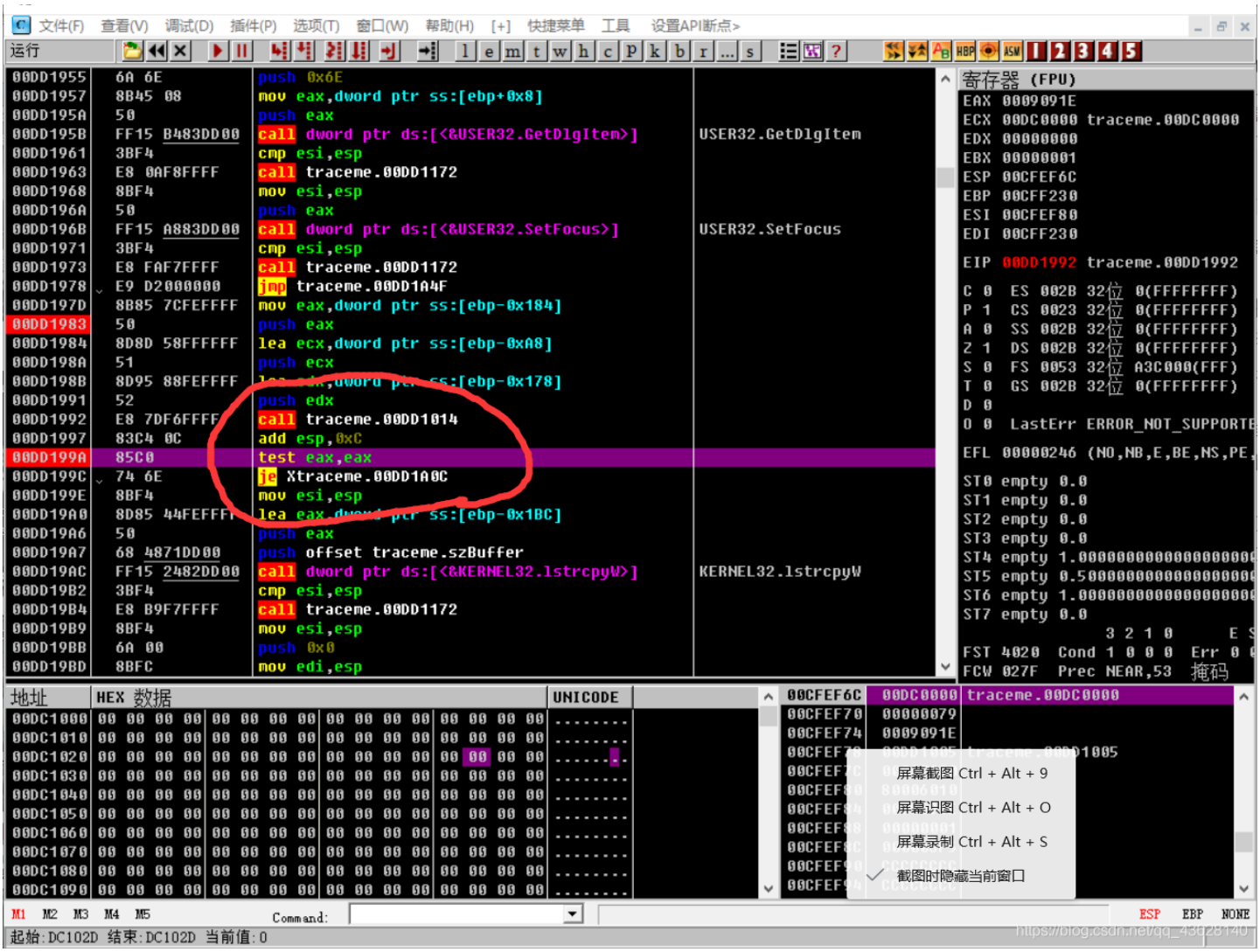
F7进入函数会发现其会进行大量运算，其为关键函数

解析后的出其会把原来的用户名给经过运算转变

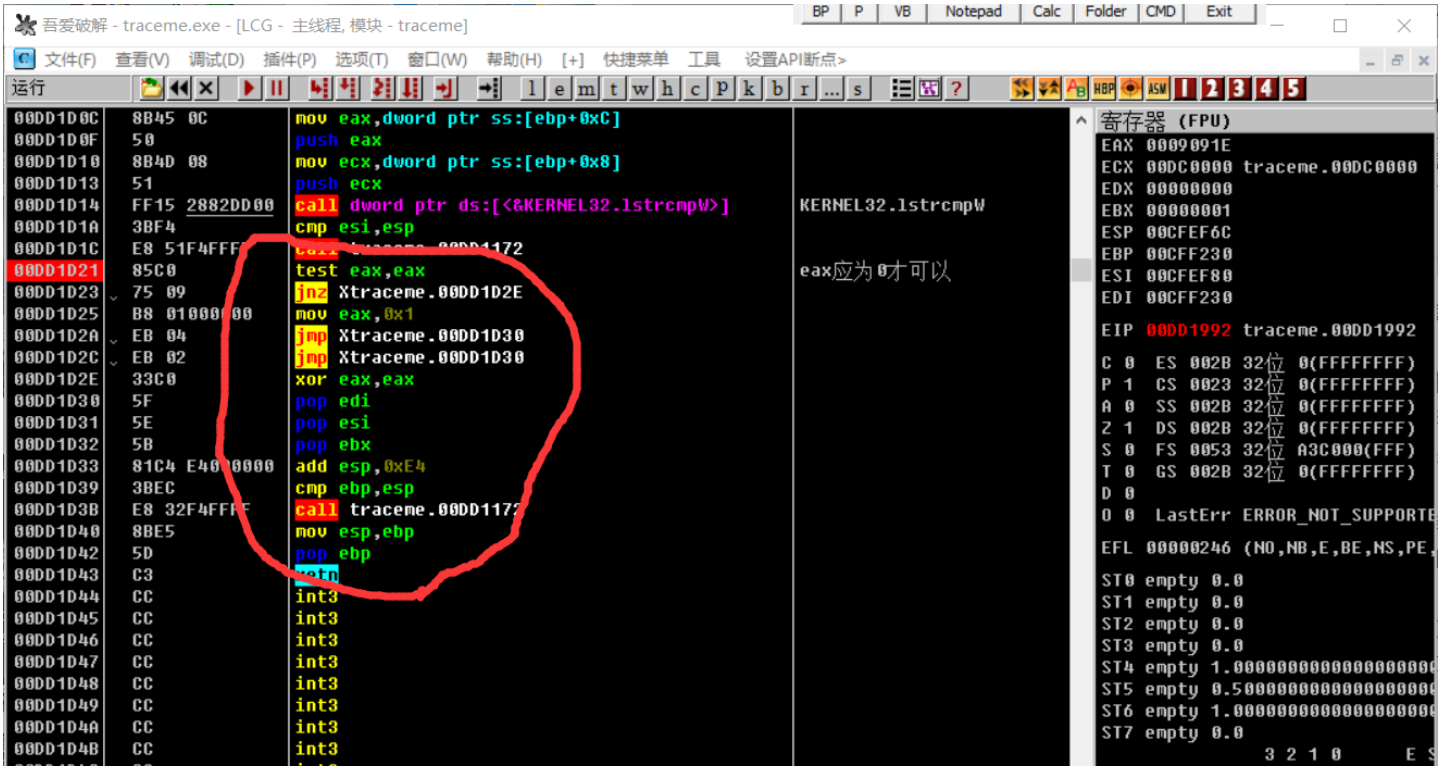
，调用wsprintf函数把转换后的字符覆盖原来的缓冲区（参数%d，说明是已是以十进制形式覆盖的）

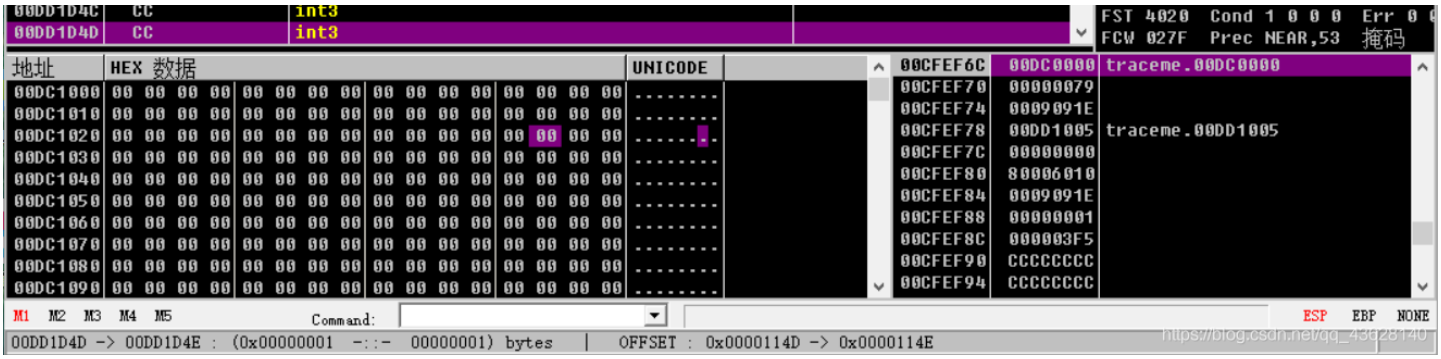
接着调用lstrcmp字符串比较函数，比较覆盖后的用户名和序列，如果相等则返回0，不相等则返回其他值，

因为本次输入的值不正确所以其返回0，之后再F7返回到关键参数的出口



其会测试test返回值，一直f7之后其执行了Je跳转到
弹出了失败的对话框，可以联想到其如果不跳转则执行下面的语句，尝试强制改变ip后再一直F7果然弹出了成功的对话框，说明只有当不执行je跳转时才会弹出成功对话框，又因为不执行je则关键函数返回值eax应不为0，所以一切的秘密还在关键函数中（只有关键函数的返回值为eax！=0才行），



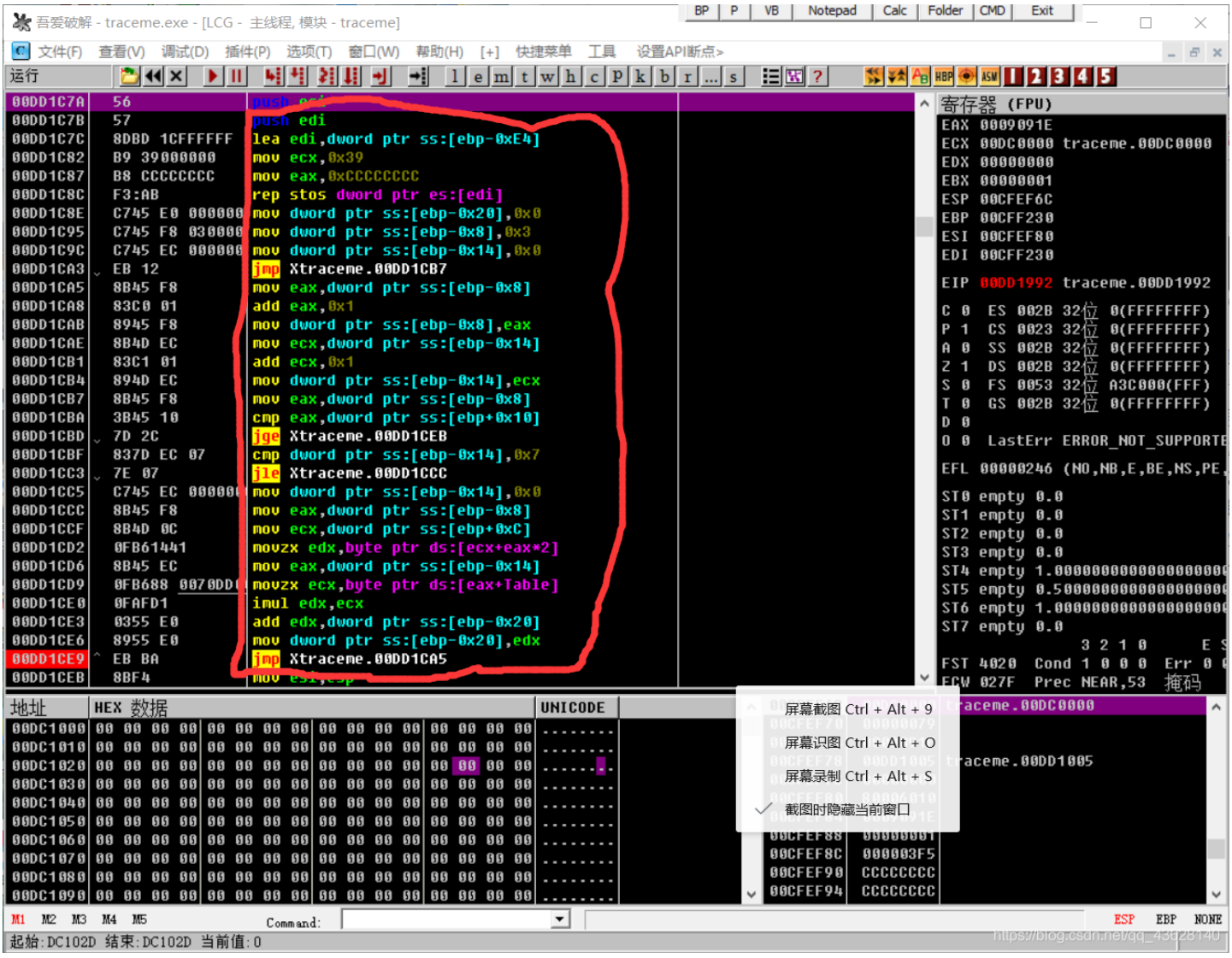


回到关键函数中，发现当执行完lstrcmp函数后如果返回值不为0则最后关键函数返回值为0，如果返回值为零则关键函数返回值eax也为0，

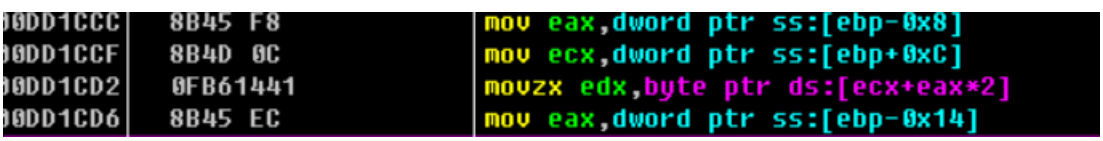
所以只要lstrcmp函数返回值为0就行，而lstrcmp返回值为0说明两个比较字符串相等，两个比较的字符串为 **序列号和关键函数处理后的用户名**

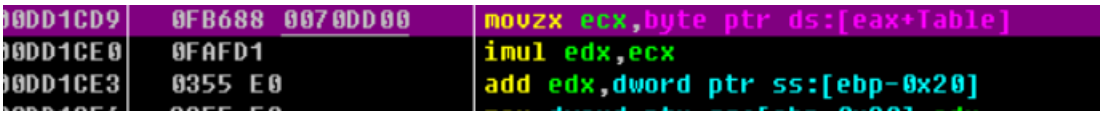
，所以只要关键函数处理后的用户名就是序列号

进一步分析关键函数的算法!



走一下发现其大致流程但不知道table所指的内存中的数据，因为不知道table的值（这时就要注意机器码中，带下划线的机器码都是指地址，）





可以右键选中数据跟踪-》地址常量跟踪，也可以用Ctrl+G搜索地址，注意机器码中的地址为内存中存放的形式，所以是低位在前高位在后。搜索后得到

table地址的数据，编写算法为`#include

```
using namespace std;
int main()
{
int a = 3;
int b;
int c = 0;
int str[] = {0x0c, 0x0a, 0x13, 0x09, 0x0c, 0x0b, 0x0a, 0x08};
int w;
int d = 0;
cout<<“请输入用户名：”;
char name[0x51];
cin>>name;
b = strlen(name);
```

```
while(1)
{
 if(a>=b)
  break;
 else
 {
  if(c<=7)
  {
   w = name[a];
   w = w * str[c];
   w = w + d;
   d = w;
  }
  else
   c = 0;
 }
 a++;
 c++;
}

cout<<"序列号为:"<<d;
```

```
}`
```

总结：破解注意通过不断设置新的断点来反复分析程序，可以采用试错的方法和（由错究因，由因再进一步分析），注意对关键函数的分析与寻找。调试时F8与F7在合适时机配合使用。