# 逆向分析技术

## 实验四进阶注册码分析

实验内容：

请各位同学参照“加密算法逆向”一章中的课程内容，对EXP4.exe分析。

实验要求：

请各位同学通过QQ作业于5月10日前在线提交如下内容：

1. 描述对EXP4.exe进行分析的过程（截图+文字）；
2. 找出一对能够通过认证的用户名/注册码（提供通过认证的截图）；
3. 解释为什么这一对用户名/注册码能够通过认证。

IDA快捷键<https://www.cnblogs.com/xuan52rock/p/11005976.html>

# 快速找到伪代码中变量所在的内存区域如要找到v4所在的数据区域，光标定位到一条引用了v4的代码，如上 LODWORD(v4) = 0;，Tab键跳到对应的IDA-view窗口。

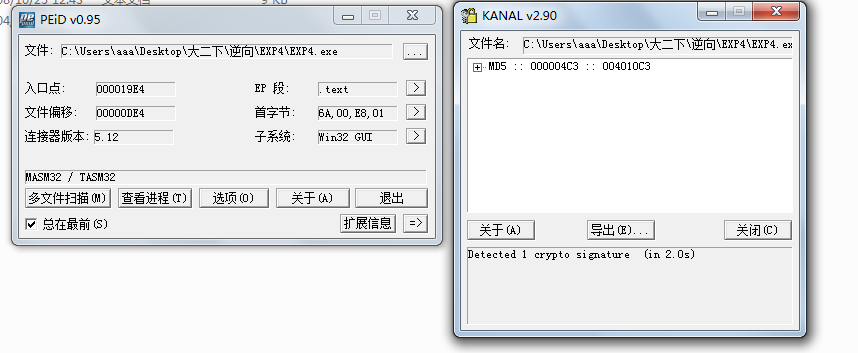
注册码比对处的关键代码

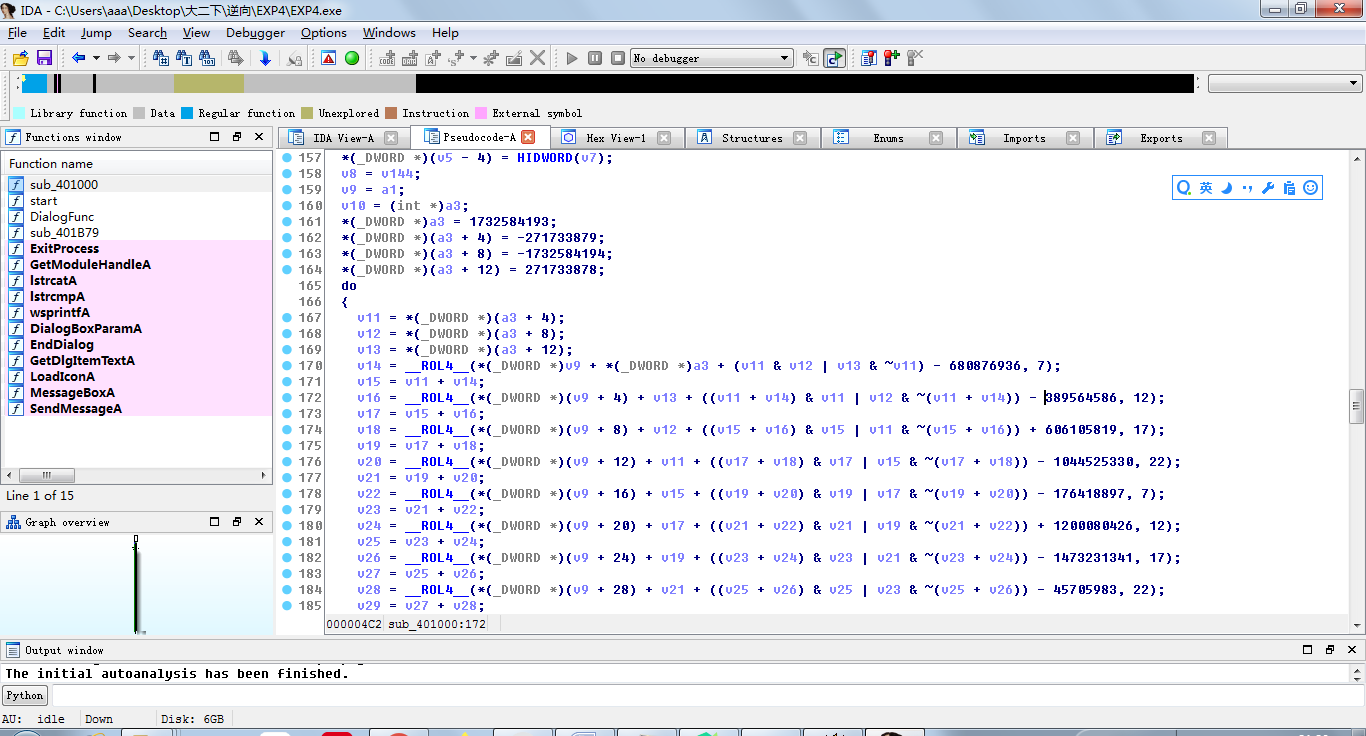
整个程序的逻辑是根据输入框Name的用户名，用Md5加密算法计算出对应的注册码。然后将计算出的注册码和我们输入的Serial比较，若相等，则运行成功。

String1为我们输入的Name。 String2为.data区的数据，用来参与MD5计算。byte\_4056A8为.data区定义的数据，用来参与MD5计算。byte\_4079D0为程序计算出来的理论注册码。byte\_409CF8为我们输入的Serial。

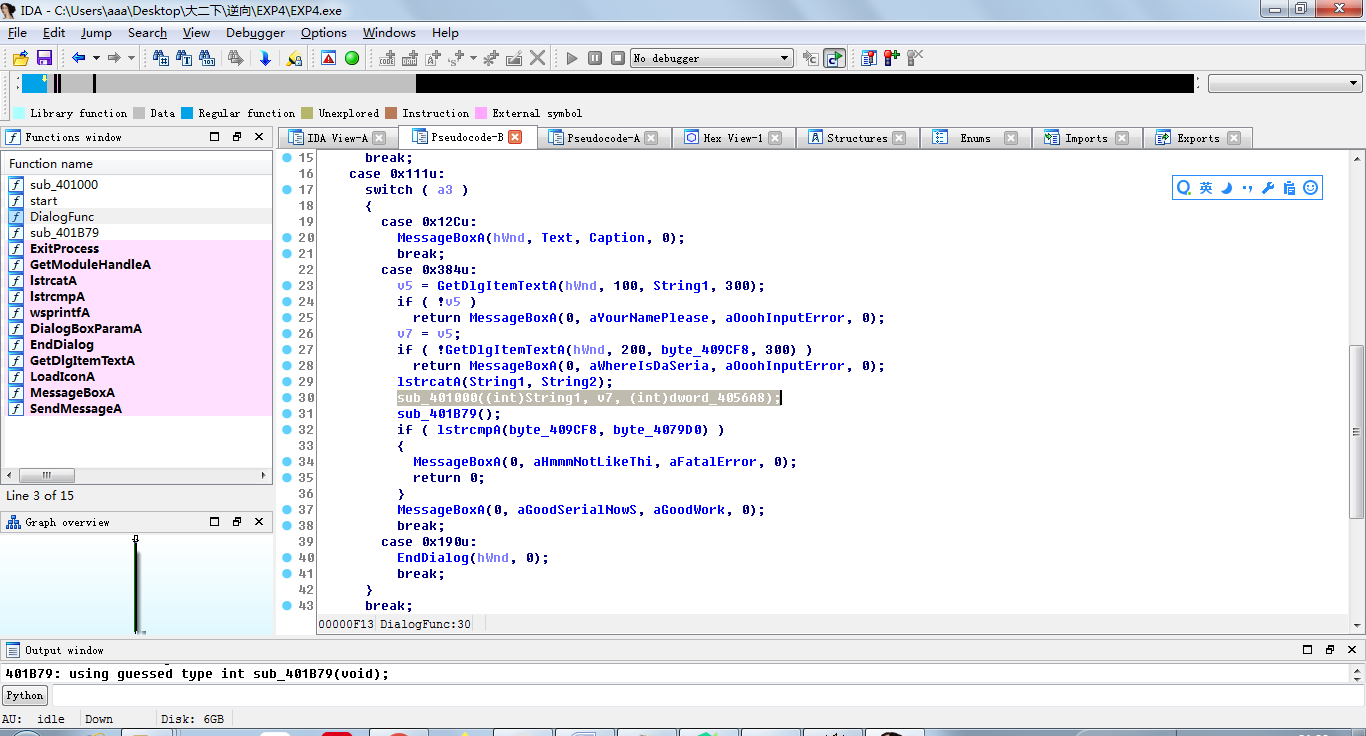
1. PEiD有一个叫做Krypto ANALyzer的插件，使用这个插件可以对程序进行扫描，通过特征匹配来识别程序内部可能用到的一些标准算法。Krypto ANALyzer的使用方法为：点击PEiD主界面右下角的“=>”按钮，选择“插件”菜单项，然后选择“Krypto ANALyzer”，就可以弹出Krypto ANALyzer插件了。Krypto ANALyzer插件会自动分析程序内部可能用到的标准算法。

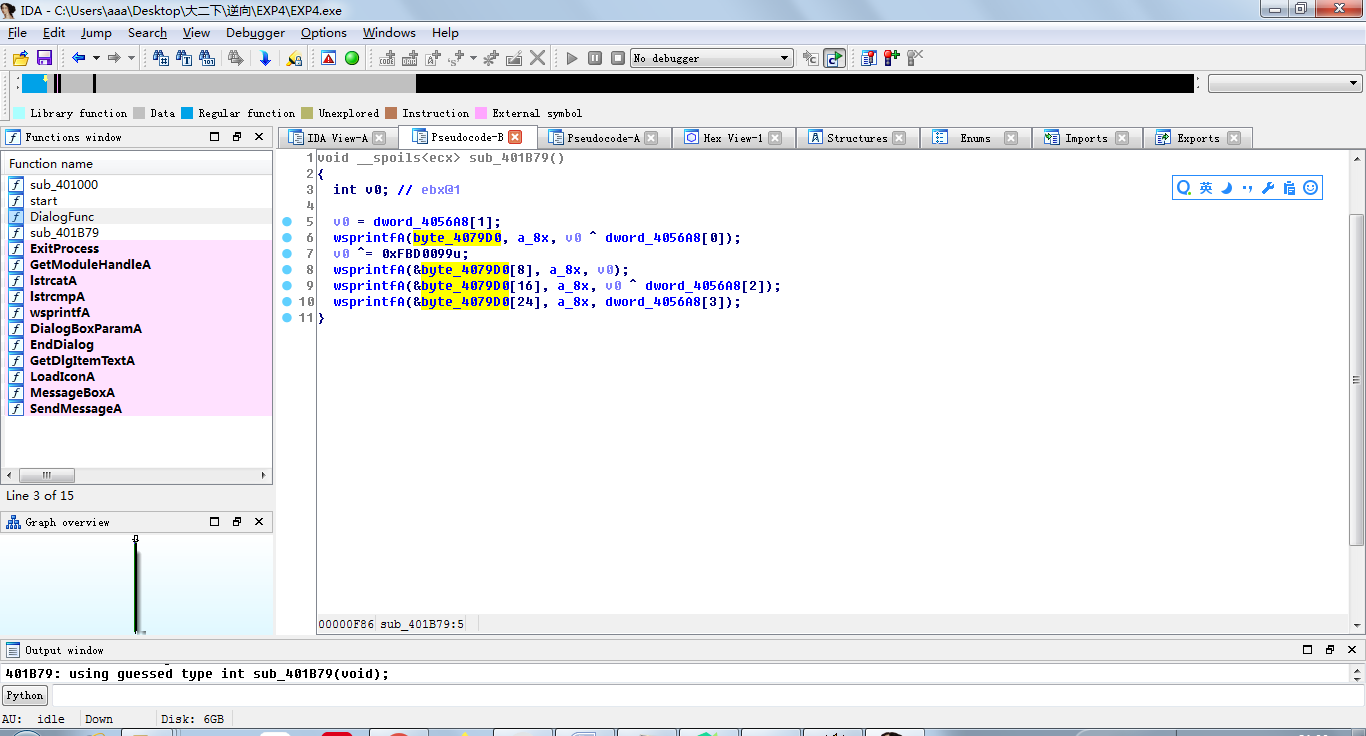
可以看出程序中在地址004010C3处存在MD5算法的特征



2.用IDA打开程序，使用快捷键g：直接跳转到004010C3地址。可以看出，这个地址位于sub\_401000 函数内。

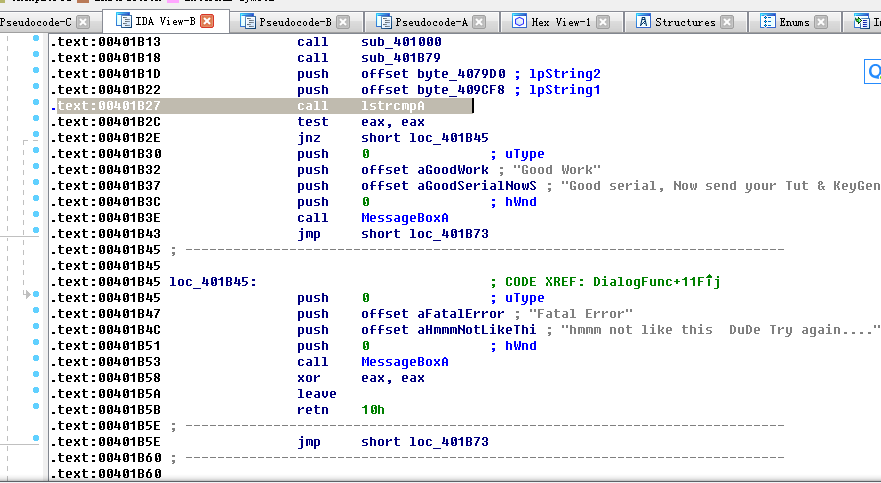
3.通过快捷键x，可以查看该地址的交叉引用。（对着某个函数、变量按该快捷键，可以查看它的交叉引用）。 sub\_401000 函数被DialogFunc函数调用



4.通过快捷键x追踪，我们可以看出lstrcmpA函数的byte\_409CF8参数来自我们的输入, byte\_4079D0参数则是在sub\_401B79函数中被赋值的。

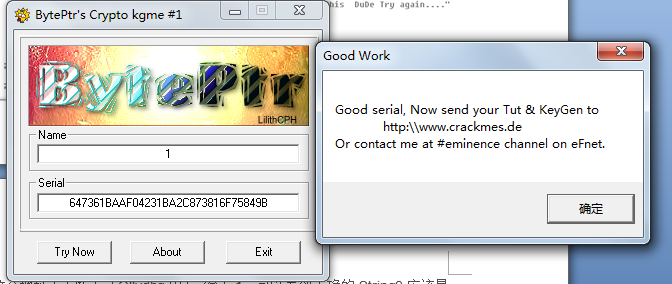
# 5. 这里是判断注册是否成功的关键步骤，判断(byte\_409CF8, byte\_4079D0)这两个变量是否相等。



**通过Tab键追踪变量****，**按Tab，跳转到对应的汇编，地址为00401B27

6.打开在Ollydbg，在00401B27这个地址上下断点，Name框输入1，可以看到byte\_4079D0是647361BAAF04231BA2C873816F75849B，也就是预期的注册码是647361BAAF04231BA2C873816F75849B

7.输入Name=1，Serial=647361BAAF04231BA2C873816F75849B，则运行成功



INT\_PTR \_\_stdcall DialogFunc(HWND hWnd, UINT a2, WPARAM a3, LPARAM a4)

{

HICON v4; // eax@2

UINT v5; // eax@9

unsigned int v7; // [sp-4h] [bp-8h]@11

switch ( a2 )

{

case 0x110u:

v4 = LoadIconA(hInstance, (LPCSTR)0x1F4);

SendMessageA(hWnd, 0x80u, 0, (LPARAM)v4);

break;

case 0x10u:

EndDialog(hWnd, 0);

break;

case 0x111u:

switch ( a3 )

{

case 0x12Cu:

MessageBoxA(hWnd, Text, Caption, 0);

break;

case 0x384u:

v5 = GetDlgItemTextA(hWnd, 100, String1, 300);// int GetDlgItemText( HWND hDlg , int nID, LPTSTR lpStr, int nMaxCount) const;

// hDlg指向含有控制的对话框的句柄。nID 指定了要获取其标题的控件的整数标识符，即你想要得到那个控件的ID。 lpStr 指向要接收控件的标题或文本的缓冲区，即存放字符串的地址。 nMaxCount 指定了要拷贝到lpStr的字符串的最大长度（以字节为单位）

// 如果函数调用成功，返回值为拷贝到缓冲区中的 TCHAR 字符个数（不包括结束空字符）。如果函数调用失败，返回值为 0 。

// String1为Name

if ( !v5 ) // 如果v5为0（函数调用失败），则报错

return MessageBoxA(0, aYourNamePlease, aOoohInputError, 0);// 报错

v7 = v5; // 令v7=v5

if ( !GetDlgItemTextA(hWnd, 200, byte\_409CF8, 300) )// 如果获取输入框的第二个值Serial失败

return MessageBoxA(0, aWhereIsDaSeria, aOoohInputError, 0);

lstrcatA(String1, String2); // 将字符串String2附加在另一个字符串String1后面,string2为定义的数据

sub\_401000((int)String1, v7, (int)byte\_4056A8);// 这是哈希函数，byte\_4056A8为定义的数据

sub\_401B79(); // 该函数对byte\_4079D0赋值(byte\_4056A8参与计算）

if ( lstrcmpA(byte\_409CF8, byte\_4079D0) )// 如果两数不相等（Serial！=byte\_4079D0），报错

{

MessageBoxA(0, aHmmmNotLikeThi, aFatalError, 0);

return 0;

}

MessageBoxA(0, aGoodSerialNowS, aGoodWork, 0);// 成功

break;

case 0x190u:

EndDialog(hWnd, 0);

break;

}

break;

}

return 0;

} 