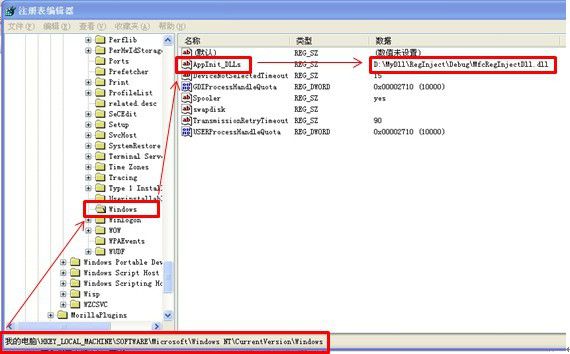
DLL注入技术之REG注入

    DLL注入技术指的是将一个DLL文件强行加载到EXE文件中，并成为EXE文件中的一部分，这样做的目的在于方便我们通过这个DLL读写EXE文件内存数据，（例如 HOOK EXE文件中的API），或以被注入EXE的身份去执行一些操作等等。  
    REG注入原理是利用在Windows 系统中，当REG以下键值中存在有DLL文件路径时，会跟随EXE文件的启动加载这个DLL文件路径中的DLL文件。当如果遇到有多个DLL文件时，需要用逗号或者空格隔开多个DLL文件的路径。

1. HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\
2. Windows\AppInit\_DLL

复制代码

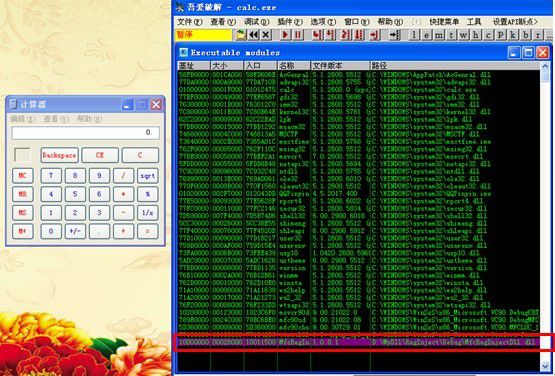
REG注入就好比在食堂（Windows 系统）发放给学生（EXE文件）饭菜（DLL文件）的过程中，食堂在将原有饭菜的基础上多发给一份紫菜鸡蛋汤（AppInit\_DLL中的DLL文件），这个紫菜鸡蛋汤可以是英伟达的UI加速的DLL文件，也可以是是病毒文件的DLL，这就要看使用者是怎么利用这个特性了。  
    REG注入DLL路径在REG文件中如下图所示：



    设置完毕后，当我们启动一个EXE文件时（例如计算器），用调试器工具OllyDBG附加到计算器，调用OllyDBG菜单命令，“查看—可执行模块”，如下图所示：



    此时我们已经可以看到DLL文件已经附加在EXE文件中了，如下图所示：



    我们可以利用这个特性来进行DLL的注入，接下来需要解决的就是关于注册表操作的Windows API了，如下列表格所示：

|  |  |
| --- | --- |
| RegOpenKeyEx | 打开注册表键值 |
| RegQueryValueEx | 查询键值 |
| RegSetValueEx | 设置键值 |
| RegCloseKey | 关闭键值 |

    主要代码如下：

1. //打开键值
2. nReg = RegOpenKeyEx(
3. HKEY\_LOCAL\_MACHINE,
4. m\_szRegPath,
5. 0,
6. KEY\_ALL\_ACCESS,
7. &hKey);
8. if(nReg != ERROR\_SUCCESS)
9. {
10. return FALSE;
11. }
12. //查询键值
13. DWORD dwReadType;
14. DWORD dwReadCount;
15. TCHAR szReadBuff[1000] = {0};
16. nReg = RegQueryValueEx(hKey,
17. \_T("AppInit\_DLLs"),
18. NULL,
19. &dwReadType,
20. (BYTE\*)&szReadBuff,
21. &dwReadCount);
22. if(nReg != ERROR\_SUCCESS)
23. {
24. return FALSE;
25. }
26. //是否dll名称已经在内容中
27. tstring strCmpBuff;
28. strCmpBuff = szReadBuff;
29. if (!strCmpBuff.find(InjectFilePath))
30. {
31. return FALSE;
32. }
33. //有字符串就加入空格
34. if (0 != \_tcscmp(szReadBuff,\_T("")))
35. {
36. \_tcscat\_s(szReadBuff,\_T(" "));
37. }
38. \_tcscat\_s(szReadBuff,InjectFilePath);
39. //把dll路径设置到注册表中
40. nReg = RegSetValueEx(hKey,
41. \_T("AppInit\_DLLs"),
42. 0,
43. REG\_SZ,
44. (CONST BYTE\*)szReadBuff,
45. (\_tcslen(szReadBuff)+1)\*sizeof(TCHAR));

复制代码

当我们完成了注册表的注入时，并不是希望所有程序都运行DLL里面的内容，这时我们就需要在DLL中过滤窗口名称，让指定窗口名称的EXE文件运行DLL里的线程。所需API如下表所示：

|  |  |
| --- | --- |
| CreateThread | 创建线程 |
| Sleep | 睡眠 |
| EnumWindows | 遍历窗口 |
| GetWindowText | 得到窗口名称 |
| GetCurrentProcessId | 得到当前进程ID |
| GetWindowThreadProcessId | 由HWND获得进程ID |

    为了实现此功能，我们需要在注入的DLL中创建线程，并在线程中执行遍历窗口函数，我们需要先获取窗口名称，与我们想运行的EXE名称进行对比，并进行进程ID对比，因为不光只有一个EXE文件的运行实例，经过这些过滤后，我们就可以在指定的EXE文件中运行代码了。  
  
    主要代码如下：

1. BOOL CALLBACK lpEnumFunc(HWND hwnd, LPARAM lParam)
2. {
3. TCHAR str[MAXBYTE] = {0};
4. //得到窗口名称
5. GetWindowText(hwnd,str,sizeof(str));
6. //是否名称是计算器
7. if(0 == \_tcscmp(str,\_T("计算器")))
8. {
9. //由于存在可能多个计算器,需要过滤线程ID
10. //得到本身线程的ID
11. DWORD dwCurrentProcessId = GetCurrentProcessId();
12. DWORD dwFindCurrentProcessId = 0;
13. //得到窗口线程ID
14. GetWindowThreadProcessId(hwnd,&dwFindCurrentProcessId);
15. //比较
16. if (dwCurrentProcessId == dwFindCurrentProcessId)
17. {
18. \*(PDWORD)lParam = 1;
19. return FALSE;
20. }
21. }
22. return TRUE;
23. }
24. DWORD ThreadProc(CMfcRegInjectDllApp\* pThis)
25. {   //切换mfc模块
26. AFX\_MANAGE\_STATE(AfxGetStaticModuleState());
27. //等待1秒时间以便于让windows创建窗口
28. Sleep(1000);
29. DWORD dwFind = 0;
30. //遍历窗口,过滤窗口名称
31. EnumWindows(lpEnumFunc,(LPARAM)&dwFind);
32. if (!dwFind) return 0;
33. //显示对话框
34. if (!pThis->m\_pUIDlg)
35. {
36. pThis->m\_pUIDlg = new RegInjectMessageBox();
37. pThis->m\_pUIDlg->Create(IDD\_DIALOG1);
38. pThis->m\_pUIDlg->ShowWindow(SW\_SHOW);
39. pThis->m\_pUIDlg->RunModalLoop();
40. }
41. return 0;
42. }
43. BOOL CMfcRegInjectDllApp::InitInstance()
44. {
45. //CWinApp::InitInstance();
46. DWORD dwThreadId;
47. m\_hThread = ::CreateThread(NULL, NULL,
48. (LPTHREAD\_START\_ROUTINE)ThreadProc,
49. this, NULL,&dwThreadId);
50. return TRUE;
51. }

复制代码

REG注入操作简单易懂，甚至不用写程序都可以完成注入操作，但是正是由于他的简单性，每个EXE都被注入，效率低，程序的扩展性差。