DLL注入技术之APC注入

    APC注入的原理是利用当线程被唤醒时APC中的注册函数会被执行的机制，并以此去执行我们的DLL加载代码，进而完成DLL注入的目的，其具体流程如下：  
    1）当EXE里某个线程执行到SleepEx()或者WaitForSingleObjectEx()时，系统就会产生一个软中断。  
    2）当线程再次被唤醒时，此线程会首先执行APC队列中的被注册的函数。  
    3）利用QueueUserAPC()这个API可以在软中断时向线程的APC队列插入一个函数指针，如果我们插入的是Loadlibrary()执行函数的话，就能达到注入DLL的目的。  
  
**1．编写测试文件**  
    新建MFC工程，添加按钮控件，双击写代码如下所示：

1. void CMfcTextApcInjectDlg::OnBnClickedSleepex()
2. {
3. // TODO: 在此添加控件通知处理程序代码
4. SleepEx(5000,TRUE);
5. }

复制代码

这里我们需要注意一下SleepEx中第二个参数为TRUE，查下msdn，上面写到：

1. bAlertable [in]
2. If this parameter is FALSE, the function does not return until the time-out period has elapsed. If an I/O completion callback occurs, the function does not return and the I/O completion function is not executed. If an APC is queued tothe thread, the function does not return and the APC function is not executed.

复制代码

大概意思是说当第二个参数为FALSE，APC是不被执行的，从此可以认为APC注入的使用条件还是有很大约束的。  
  
**2．编写APC注入程序**  
    由于我们需要时使用LoadLibrary()函数完成注入，因此需要为其先准备好必要的参数，需要我们可以通过在远程进程中申请空间的方式写入LoadLibrary()函数所需要的参数（也就是DLL的路径）。关键代码如下所示：

1. //打开远程进程
2. handle = OpenProcess(PROCESS\_ALL\_ACCESS,FALSE,dwProcessId);
3. if (handle)
4. {
5. //在远程进程申请空间
6. lpData = VirtualAllocEx(handle,
7. NULL,
8. 1024,
9. MEM\_COMMIT,
10. PAGE\_EXECUTE\_READWRITE);
11. if (lpData)
12. {
13. //在远程进程申请空间中写入待注入DLL的路径
14. bRet = WriteProcessMemory(handle,
15. lpData,
16. (LPVOID)sDllName,
17. 1024,&dwRet);
18. }
19. //关闭句柄
20. CloseHandle(handle);
21. }

复制代码

当我们准备好用于注入DLL的LoadLibrary()函数后，接下来需要使用QueueUserAPC()函数将此函数插入到软中断线程的APC队列中。但是由于QueueUserAPC()函数的第三个参数是线程ID，因此我们需要根据现有进程ID，并通过遍历对比得到线程ID，具体API如下表所示：

|  |  |
| --- | --- |
| CreateToolhelp32Snapshot | 创建线程快照 |
| Thread32First | 得到第一个线程快照 |
| Thread32Next | 循环下一个线程快照 |

    关键代码如下所示：

1. THREADENTRY32 te = {0};
2. te.dwSize = sizeof(THREADENTRY32);
3. //得到线程快照
4. HANDLE handleSnap = CreateToolhelp32Snapshot(TH32CS\_SNAPTHREAD,0);
5. if (INVALID\_HANDLE\_VALUE == handleSnap)
6. {
7. return FALSE;
8. }
9. BOOL bStat = FALSE;
10. //得到第一个线程
11. if (Thread32First(handleSnap,&te))
12. {
13. do
14. {
15. //进行进程ID对比
16. if (te.th32OwnerProcessID == dwProcessId)
17. {
18. //得到线程句柄
19. HANDLE handleThread = OpenThread(
20. THREAD\_ALL\_ACCESS,
21. FALSE,
22. te.th32ThreadID);
23. if (handleThread)
24. {
25. //向线程插入APC
26. dwRet = QueueUserAPC(
27. (PAPCFUNC)LoadLibrary,
28. handleThread,
29. (ULONG\_PTR)lpData);
30. if (dwRet > 0)
31. {
32. bStat = TRUE;
33. }
34. //关闭句柄
35. CloseHandle(handleThread);
36. }
37. }
38. //循环下一个线程
39. } while (Thread32Next(handleSnap,&te));
40. }
41. CloseHandle(handleSnap);

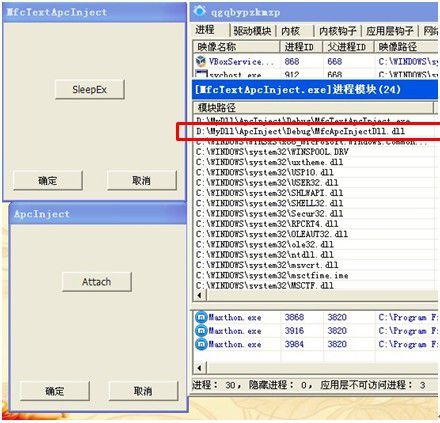
复制代码

**3．MFC工程设置和提升权限**  
    经过以上两步的操作，我们已经准备好APC注入的关键代码，现在我们需要将自己的程序提升权限以方便注入操作（另，动态MFC库编译有可能造成注入失败）。主要代码如下：

1. int CApcInjectDll::EnablePrivilege(bool isStart)
2. {
3. //1. 得到令牌句柄
4. HANDLE  hToken = NULL;      //令牌句柄
5. if (!::OpenProcessToken( GetCurrentProcess(),
6. TOKEN\_ADJUST\_PRIVILEGES | TOKEN\_QUERY | TOKEN\_READ,
7. &hToken))
8. {
9. return FALSE;
10. }
11. //2. 得到特权值
12. LUID    luid = {0};         //特权值
13. if (!::LookupPrivilegeValue(NULL, SE\_DEBUG\_NAME, &luid))
14. {
15. return FALSE;
16. }
17. //3. 提升令牌句柄权限
18. TOKEN\_PRIVILEGES tp = {0};  //令牌新权限
19. tp.PrivilegeCount = 1;
20. tp.Privileges[0].Luid = luid;
21. tp.Privileges[0].Attributes = isStart ? SE\_PRIVILEGE\_ENABLED : 0;
22. if (!::AdjustTokenPrivileges(hToken, FALSE, &tp, 0, NULL, NULL))
23. {
24. return FALSE;
25. }
26. //4. 关闭令牌句柄
27. ::CloseHandle(hToken);
28. return 0;
29. }

复制代码

**4．测试注入效果**  
    点击待注入的EXE进行SleepEx，这时EXE的窗口是不可以移动的，因为只有一个线程，处于SleepEx的挂起状态，然后进行注入，我们此时会发现处于挂起状态的进程窗口突然可以移动了，这是因为进程在挂起状态等待时，如果有APC队列就会退出等待并执行APC队列中的函数，然后程序继续运行。执行效果如下图所示：



     APC注入因为受目标进程使用API的条件而受限，并且处于等待的线程被注入后会立即返回，也有可能造成线程的运行错误，所以应用起来不是很通用。