DLL注入技术之远线程注入

    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。  
    远线程注入原理是利用Windows 系统中CreateRemoteThread()这个API，其中第4个参数是准备运行的线程，我们可以将LoadLibrary()填入其中，这样就可以执行远程进程中的LoadLibrary()函数，进而将我们自己准备的DLL加载到远程进程空间中执行。  
    当然除了CreateRemoteThread()和LoadLibrary()这个两个主要的API还是远远不够的，我们还需要以下表格所示的API：

|  |  |
| --- | --- |
| OpenProcess | 打开远程进程 |
| VirtualAllocEx | 在远程进程中申请空间 |
| WriteProcessMemory | 在远程进程中写入数据 |
| WaitForSingleObject | 等待信号量 |
| VirtualFreeEx | 释放远程进程中申请空间 |
| CloseHandle | 关闭句柄 |

    主要代码如下：

1. int CRemoteThreadInjectDLL::InjectDll(DWORD dwProcessId, PTCHAR szDllName)
2. {
3. if (szDllName[0] == NULL)
4. return -1;
5. //提高权限相关操作
6. EnablePrivilege(TRUE);
7. //1. 打开进程
8. HANDLE hProcess = ::OpenProcess(  PROCESS\_ALL\_ACCESS,   //打开进程权限
9. FALSE,                                              //是否可继承
10. dwProcessId);                                       //进程ID
11. if (hProcess == INVALID\_HANDLE\_VALUE)
12. return -1;
13. //2. 在远程进程中申请空间
14. LPVOID pszDllName = ::VirtualAllocEx(hProcess, //远程进程句柄
15. NULL,                                  //建议开始地址
16. 4096,                                  //分配空间大小
17. MEM\_COMMIT,                            //空间初始化全0
18. PAGE\_EXECUTE\_READWRITE);               //空间权限
19. if (NULL == pszDllName)
20. {
21. return -1;
22. }
23. //3. 向远程进程中写入数据
24. BOOL bRet = ::WriteProcessMemory( hProcess, pszDllName,
25. szDllName, MAX\_PATH, NULL);
26. if (NULL == bRet)
27. {
28. return -1;
29. }
30. //4. 在远程进程中创建远程线程
31. m\_hInjecthread = ::CreateRemoteThread(hProcess,      //远程进程句柄
32. NULL,                                            //安全属性
33. 0,                                               //栈大小
34. (LPTHREAD\_START\_ROUTINE)LoadLibrary,             //进程处理函数
35. pszDllName,                                      //传入参数
36. NULL,                                            //默认创建后的状态
37. NULL);                                           //线程ID
38. if (NULL == m\_hInjecthread)
39. {
40. DWORD dwErr = GetLastError();
41. return -1;
42. }
43. //5. 等待线程结束返回
44. DWORD dw = WaitForSingleObject(m\_hInjecthread, -1);
45. //6. 获取线程退出码,即LoadLibrary的返回值，即dll的首地址
46. DWORD dwExitCode;
47. GetExitCodeThread(m\_hInjecthread, &dwExitCode);
48. m\_hMod = (HMODULE)dwExitCode;
49. //7. 释放空间
50. BOOL bReturn = VirtualFreeEx(hProcess, pszDllName,
51. 4096, MEM\_DECOMMIT);
52. if (NULL == bReturn)
53. {
54. return -1;
55. }
56. CloseHandle(hProcess);
57. hProcess = NULL;
58. //恢复权限相关操作
59. EnablePrivilege(FALSE);
60. return 0;
61. }

复制代码

此外，我们还需要提升进程权限以便于提高注入成功率，所需API如下表所示：

|  |  |
| --- | --- |
| OpenProcessToken | 得到令牌句柄 |
| LookupPrivilegeValue | 得到权限值 |
| AdjustTokenPrivileges | 提升令牌句柄权限 |

    主要代码如下：

1. int CRemoteThreadInjectDLL::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

当要在指定的进程中加载DLL时，我们就需要过滤指定名称的进程，这时遍历进程ID并进行对比，得到所指定的进程，所需API如表所示：

|  |  |
| --- | --- |
| CreateToolhelp32Snapshot | 创建进程快照 |
| Process32First | 第一个进程快照 |
| Process32Next | 循环下一个进程快照 |

    主要代码如下：

1. DWORD CRemoteThreadInjectDLL::GetProcessId(PTCHAR pszProcessName)
2. {
3. HANDLE hProcess = ::CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);
4. if (INVALID\_HANDLE\_VALUE == hProcess)
5. {
6. return 0;
7. }
8. DWORD dwProcessId = 0;
9. PROCESSENTRY32 process32 = {0};
10. process32.dwSize = sizeof(PROCESSENTRY32);
11. BOOL bRetProcess = FALSE;
12. bRetProcess = ::Process32First(hProcess, &process32);
13. do
14. {
15. if (\_tcscmp(pszProcessName, process32.szExeFile) == 0)
16. {
17. dwProcessId = process32.th32ProcessID;
18. break;
19. }
20. bRetProcess = ::Process32Next(hProcess, &process32);
21. }while (bRetProcess);
22. ::CloseHandle(hProcess);
23. return dwProcessId;
24. }

复制代码

远线程注入API使用较多，不易实现。但是可以批量注入和卸载，这样对于需要反复调试的注入就非常的方便。