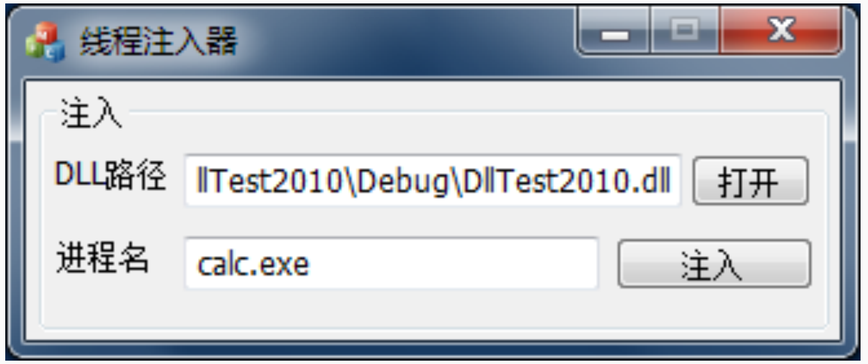
极客学院
jikexueyuan.com

C++实战：远程线程注入

远程线程注入— 效果展示


计算器
编辑(E)
查看(V)
帮助(H)
Welcome
Welcome to Jikexueyuan~
确定


线程注入器
注入
DLL路径
llTest2010\Debug\DllTest2010.dll
打开
进程名
calc.exe
注入

远程线程注入— **课程概要**

- DLL概述
- DLL的编写
- 线程注入

DLL概述

DLL概述

DLL(Dynamic Link Library)，又称为“应用程序拓展”。是一个个**独立的模块**，包含资源文件、函数等，供应用程序需要时调用。现阶段，大部分应用程序都会设计成**主程序+DLL**的框架。当主程序需要哪个功能的时候，就会去链接并调用对应的DLL，**因为DLL是彼此独立的，所以主程序的加载速度更快**。这样也提升了工作效率。

同时，由于DLL的独立性，使得程序的**维护和更新变得更加容易**，成本大大降低。因为程序只需更新相应的模块，而不是整体的更新。

DLL概述

线程注入，是**通过开启远程线程的方式，将DLL加载到目标宿主进程中**的常用方式。

目前，远程线程注入，作为WinNT平台下**最常用的DLL注入方式之一**，在游戏、安全等领域扮演着极为重要的角色。

由于WinNT系统下进程空间的独立性，获取其他进程的信息，就需要通过进入目标进程空间的方式，而使用线程注入就可以**轻易地实现目的**。

但是滥用线程注入，或者是注入一个具有编写BUG的DLL，很容易对宿主进程的稳定性造成破坏，进而导致进程崩溃。

所以，请记住： Power comes with responsibility （责任随权利而来）

DLL概述— **DLL的分类**

链接库分为**动态链接库**和**静态链接库**两种，后缀名分别是.dll和.lib

静态链接库，在运行的时候就直接把代码全部加载到程序中。通过如下方式调用

```
#include <Psapi.h>
#pragma comment (lib,"Psapi.lib”)
```

而动态链接库，在需要的时候加载，在不需要的时候就卸载释放资源

使用**LoadLibrary** 动态加载DLL

使用**GetProcAddress** 获取DLL中导出函数的指针

最后使用**FreeLibrary** 卸载指定DLL

DLL概述— **DLL的分类**

在我们的**Visual Studio**的编译环境下，DLL又分为三类：

非MFC的DLL --- 即使用 SDK API 进行编程，能被其他所有语言调用；

MFC规则DLL --- 可以使用 MFC 进行编程，能被其他所有语言调用；

MFC扩展DLL --- 可以使用 MFC进行编程，但只能被用MFC编写的程序调用。

本课我们使用第一种DLL。

第二种使用MFC框架开发，可以使用封装类；第三种不常用。

（MFC - Microsoft Foundation Class-Library 是微软用C++对 API 进行的封装，全部封装成了类，简化了使用）

DLL概述— DLL的入口点与参数

```
BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                         )
{
    return TRUE;
}
```

这是DLL文件的**入口点函数**

DLL概述— DLL的入口点与参数

DLL_PROCESS_ATTACH：

当进程**第一次**链接DLL并通过它的入口点时，会得到这个参数

DLL_PROCESS_DETACH：

进程在空间内取消DLL的映射时，会得到这个参数

DLL_THREAD_ATTACH：

每当新线程创建时，系统会对所有映射的DLL传此参数调用入口函数

DLL_THREAD_DETACH：

每当线程退出或返回时，系统会对所有映射的DLL传此参数要求执行对应清理工作

DLL的编写

DLL的编写— **DLL的编写与导出**

DLL的导出函数使用

```
extern "C" _declspec(dllexport)
```

而导入函数使用

```
extern "C" _declspec(dllimport)
```

```
extern  "C "
```

作用：作为一种**编译约定**

DLL的编写— DLL的动态加载

既然我们要把DLL注入进程，那么需要先了解一下，进程是怎样调用DLL的：

使用**LoadLibrary**加载进所需DLL

```
HMODULE  hMod = LoadLibrary (DLL路径)
```

定义导入函数指针

```
typedef int (*ADD_IMPORT) (int a,int b);    //定义一个指向返回值为int类型的函数指针
```

使用**GetProcAddress**获得函数入口点

```
ADD_IMPORT  add_proc = (ADD_IMPORT)GetProcAddress(hMod,"Add");
```

在程序中尽情使用

```
int  result = add_proc(1 , 2);
```

线程注入

线程注入— **注入的可行性**

kernel32.dll和**user32.dll**是两个在大部分程序上都会调用的DLL

同一个DLL，在不同的进程中，**不一定**被映射（加载）在同一个内存地址下。

但是**kernel32.dll和user32.dll例外**。他们总是被映射到进程的**内存首选地址**

因此，在所有使用这两个DLL的进程中，这两个DLL的内存地址是**相同**的。

我们在**本进程获取的kernel32.dll中函数的地址，在目标进程中也是一样的**

目标进程→传入DLL地址→开启远程线程→加载DLL→实现DLL的注入

线程注入— **实现线程注入**

依次使用以下函数：

```
OpenProcess          //获取已知进程的句柄;
VirtualAllocEx       //在进程中申请空间;
WriteProcessMemory   //向进程中写入东西;
GetProcAddress       //取得函数在DLL中的地址;
CreateRemoteThreadEx //在其他进程中创建新线程;
CloseHandle          //关闭句柄;
```

完成线程的注入

远程线程注入

本套课程中我们学习了线程注入的基本思路和实现方法。你应当掌握了以下知识：

- DLL的编写，使用
- 在代码中调用DLL
- 实现简单的远程线程注入

如果您想更多的了解本课中多线程，MFC的相关知识，可以关注我接下来的课程

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台