不依赖 .Net 的可以调用托管代码的非托管代码

不依赖 .Net 的可以调用托管代码的非托管代码公共语言运行库提供了在非托管代码中调用托管代码的机制。具体方法可以参考相关的文章，在此不做赘述。

本文讨论的是在非托管代码中使用公共语言运行库的 API 调用托管代码时发生的对 .Net 框架的依赖问题。

假设你希望写一个非托管程序，可以动态加载非托管或托管的可执行文件。你很快会发现，无论这个程序是否要操作托管代码，没有 mscoree.dll 都不能运行。也就是说，尽管这是个非托管程序，它却必须在 .Net 安装后才能运行。是否有一种方法，如果这个程序不调用托管代码，它就不需要 .Net ；只有在需要加载托管程序时， .Net 才是必要的。答案是肯定的。

在非托管代码中调用托管代码的通用途径是调用公共语言运行库 (CLR) 提供的 Hosting API。其中最重要的是 CorBindToRuntimeEx 。但是，这个函数是从 mscoree.dll 中导出的。因此，如果非托管代码调用了这个函数，要运行它， 必须有 mscoree.dll 。这就产生了对 .Net 的依赖。

其实， CorBindToRuntimeEx 的目的是建立一个

ICorRuntimeHost　接口指针的实例，我们可以使用　　COM 的函数来解决这种以来

性。具体代码如下：

CComPtr<ICorRuntimeHost> spRuntimeHost;

CComPtr<_AppDomain> spAppDomain;

CComPtr<IUnknown> spUnk;/*　　这个调用产生对　.Net 的依赖

if (FAILED(CorBindToRuntimeEx(NULL, // Latest Version by Default

L"wks", // Workstation build

STARTUP_LOADER_OPTIMIZATION_SINGLE_DOMAIN,

CLSID_CorRuntimeHost ,

IID_ICorRuntimeHost ,

(void**)&spRuntimeHost)))

{

return false;

}

……

*/

换成这个调用，可以避免对　　mscoree.dll　的依赖。

HRESULT hr =

spRuntimeHost.CoCreateInstance(__uuidof(CorRuntimeH

ost));

  if (FAILED(hr)) {

return false;

  }

　…… .. 当然,在调用 CoCreateInstance()　之前和之后,

要调用 CoInitialize()　和

CoUninitialize()　。详细信息参考　COM 的有关文章　To encrypt

the managed code and execute it with custom CLR host is

a

good idea. But who will guarantee that CLRHost->Start()

cannot be

hooked and by this way the cracker can get the decrypted

managed code in

the moment before it is sent to CLR?