 本课中我们将要学习WINDOWS钩子函数的使用方法。WINDOWS钩子函数的功能非常强大，有了它您可以探测其它进程并且改变其它进程的行为。 理论： WINDOWS的钩子函数可以认为是WINDOWS的主要特性之一。利用它们，您可以捕捉您自己进程或其它进程发生的事件。通过“ 钩挂 ”，您可以给WINDOWS一个处理或过滤事件的回调函数，该函数也叫做“ 钩子函数 ”，当每次发生您感兴趣的事件时，WINDOWS都将调用该函数。一共有两种类型的钩子：局部的和远程的。 局部钩子仅钩挂您自己进程的事件。 远程的钩子还可以将钩挂其它进程发生的事件。远程的钩子又有两种： 基于线程的 它将捕获其它进程中某一特定线程的事件。简言之，就是可以用来观察其它进程中的某一特定线程将发生的事件。 系统范围的 将捕捉系统中所有进程将发生的事件消息。 安装钩子函数将会影响系统的性能。监测“ 系统范围事件 ”的系统钩子特别明显。因为系统在处理所有的相关事件时都将调用您的钩子函数，这样您的系统将会明显的减慢。所以应谨慎使用，用完后立即卸载。还有，由于您可以预先截获其它进程的消息，所以一旦您的钩子函数出了问题的话必将影响其它的进程。记住：功能强大也意味着使用时要负责任。 在正确使用钩子函数前，我们先讲解钩子函数的工作原理。当您创建一个钩子时，WINDOWS会先在内存中创建一个数据结构，该数据结构包含了钩子的相关信息，然后

把该结构体加到已经存在的钩子链表中去。新的钩子将加到老的前面。当一个事件发生时，如果您安装的是一个局部钩子，您进程中的钩子函数将被调用。如果是一个远程钩子，系统就必须把钩子函数插入到其它进程的地址空间，要做到这一点要求钩子函数必须在一个动态链接库中，所以如果您想要使用远程钩子，就必须把该钩子函数放到动态链接库中去。当然有两个例外：工作日志钩子和工作日志回放钩子。这两个钩子的钩子函数必须在安装钩子的线程中。原因是：这两个钩子是用来监控比较底层的硬件事件的，既然是记录和回放，所有的事件就当然都是有先后次序的。所以如果把回调函数放在DLL中，输入的事件被放在几个线程中记录，所以我们无法保证得到正确的次序。故解决的办法是：把钩子函数放到单个的线程中，譬如安装钩子的线程。 钩子一共有14种，以下是它们被调用的时机： WH_CALLWNDPROC 当调用SendMessage时 WH_CALLWNDPROCRET 当SendMessage的调用返回时 WH_GETMESSAGE 当调用GetMessage 或 PeekMessage时 WH_KEYBOARD 当调用GetMessage 或 PeekMessage 来从消息队列中查询WM_KEYUP 或 WM_KEYDOWN 消息时 WH_MOUSE 当调用GetMessage 或 PeekMessage 来从消息队列中查询鼠标事件消息时 WH_HARDWARE 当调用GetMessage 或 PeekMessage 来从消息队列种查询非鼠标、键盘消息时 WH_MSGFILTER 当对话框、菜单或滚动条要处理一个消息时。该钩子是局部的。它时为那些有自己的消息处理过程的控件对象设计的。 WH_SYSMSGFILTER 和WH_MSGFILTER一样，只不过是系统范围的 WH_JOURNALRECORD 当WINDOWS从硬件队列

中获得消息时 WH_JOURNALPLAYBACK 当一个事件从系统的硬件输入队列中被请求时 WH_SHELL 当关于WINDOWS外壳事件发生时，譬如任务条需要重画它的按钮. WH_CBT 当基于计算机的训练(CBT)事件发生时 WH_FOREGROUNDIDLE 由WINDOWS自己使用，一般的应用程序很少使用 WH_DEBUG 用来给钩子函数除错 现在我们知道了一些基本的理论，现在开始讲解如何安装/卸载一个钩子。 要安装一个钩子，您可以调用SetWindowHookEx函数。该函数的原型如下： SetWindowsHookEx proto HookType:DWORD, pHookProc:DWORD, hInstance:DWORD, ThreadID:DWORD HookType 是我们上面列出的值之一,譬如： WH_MOUSE, WH_KEYBOARD pHookProc 是钩子函数的地址。如果使用的是远程的钩子，就必须放在一个DLL中，否则放在本身代码中 hInstance 钩子函数所在DLL的实例句柄。如果是一个局部的钩子，该值为NULL ThreadID 是您安装该钩子函数后想监控的线程的ID号。该参数可以决定该钩子是局部的还是系统范围的。如果该值为NULL，那么该钩子将被解释成系统范围内的，那它就可以监控所有的进程及它们的线程。如果您指定了您自己进程中的某个线程ID 号，那该钩子是一个局部的钩子。如果该线程ID是另一个进程中某个线程的ID，那该钩子是一个全局的远程钩子。这里有两个特殊情况：WH_JOURNALRECORD 和 WH_JOURNALPLAYBACK总是代表局部的系统范围的钩子，之所以说是局部，是因为它们没有必要放到一个DLL中。WH_SYSMSGFILTER 总是一个系统范围内的远程钩子。其实它和WH_MSGFILTER钩子类似，如果把参数ThreadID设

成0的话，它们就完全一样了。 如果该函数调用成功的话，将在eax中返回钩子的句柄，否则返回NULL。您必须保存该句柄，因为后面我们还要它来卸载钩子。 要卸载一个钩子时调用UnhookWidowHookEx函数，该函数仅有一个参数，就是欲卸载的钩子的句柄。如果调用成功的话，在eax中返回非0值，否则返回NULL。 现在您知道了如何安装和卸载一个钩子了，接下来我们将看看钩子函数。. 只要您安装的钩子的消息事件类型发生，WINDOWS就将调用钩子函数。譬如您安装的钩子是WH_MOUSE类型，那么只要有一个鼠标事件发生时，该钩子函数就会被调用。不管您安装的时那一类型钩子，钩子函数的原型都时是一样的： HookProc proto nCode:DWORD, wParam:DWORD, lParam:DWORD nCode 指定是否需要处理该消息 wParam 和 lParam 包含该消息的附加消息 HookProc 可以看作是一个函数名的占位符。只要函数的原型一致，您可以给该函数取任何名字。至于以上的几个参数及返回值的具体含义各种类型的钩子都不相同。譬如： WH_CALLWNDPROC nCode 只能是HC_ACTION，它代表有一个消息发送给了一个窗口 wParam 如果非0，代表正被发送的消息 lParam 指向CWPSTRUCT型结构体变量的指针 return value: 未使用，返回0 WH_MOUSE nCode 为HC_ACTION 或 HC_NOREMOVE wParam 包含鼠标的事件消息 lParam 指向MOUSEHOOKSTRUCT型结构体变量的指针 return value: 如果不处理返回0，否则返回非0值 所以您必须查询您的WIN32 API 指南来得到不同类型的钩子的参数的详细定义以及它们返回值的意义。这里还有一个问题需要注意：所有的钩子都串在一个链表上，最近加入的钩子放在链表的

头部。当一个事件发生时，WINDOWS将按照从链表头到链表尾调用的顺序。所以您的钩子函数有责任把消息传到下一个链中的钩子函数。当然您可以不这样做，但是您最好明白这时这么做的原因。在大多数的情况下，最好把消息事件传递下去以便其它的钩子都有机会获得处理这一消息的机会。调用下一个钩子函数可以调用函数CallNextHookEx。该函数的原型如下： CallNextHookEx proto hHook:DWORD, nCode:DWORD, wParam:DWORD, lParam:DWORD hHook 时是您自己的钩子函数的句柄。利用该句柄可以遍历钩子链。nCode, wParam and lParam 您只要把传入的参数简单传给CallNextHookEx即可。 请注意：对于远程钩子，钩子函数必须放到DLL中，它们将从DLL中映射到其它的进程空间中去。当WINDOWS映射DLL到其它的进程空间中去时，不会把数据段也进行映射。简言之，所有的进程仅共享DLL的代码，至于数据段，每一个进程都将有其单独的拷贝。这是一个很容易被忽视的问题。您可能想当然的以为，在DLL中保存的值可以在所有映射该DLL的进程之间共享。在通常情况下，由于每一个映射该DLL的进程都有自己的数据段，所以在大多数的情况下您的程序运行得都不错。但是钩子函数却不是如此。对于钩子函数来说，要求DLL的数据段对所有的进程也必须相同。这样您就必须把数据段设成共享的，这可以通过在链接开关中指定段的属性来实现。在MASM中您可以这么做： /SECTION:, S 已初期化的段名是.data，未初始化的段名是.bss。`加入您想要写一个包含钩子函数的DLL，而且想使它的未初始化的数据段在所有进程间共享，您必须这么做： link /section:.bss,S /DLL /SUBSYSTEM:WINDOWS .....S 代

表该段是共享段。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com