Windows环境下的屏幕取词技术

合肥工业大学电子信息工程研究所
王 歆 张林山

许多流行的即时翻译、即时内码转换软件(金山词霸、南极星等)都使用屏幕取词技术。该技术是对Windows程序员的一项综合考试。它涉及到：Windows环境下的内存管理、动态链接库管理、PE文本以屏幕取词的实现为核心，并对这些相关技术作了详细的介绍。

主动获取与被动获取

直接从屏幕缓冲区中得到文本信息可能是DOS程序员的第一思路，然而，在Windows环境中。用户模式下的进程无权访问屏幕缓冲区，而编写管理模式的应用程序要使用复杂的DDK技术。借用古老的CCDOS汉化英文DOS思想，我们可以很容易地将思路转向另一种获取技术：被动获取。CCDOS在启动后接管了DOS/BIOS中字符显示的中断。应用程序显示字符的要求被重定向到新的中断处理函数。这样，新的中断处理函数就可以获取应用程序的字符显示信息，做出相应的处理。

在Windows中，这样的字符串显示函数位于GDI32.DLL中，共有四个：TextoutA，TextoutW，TextoutExA，TextoutExW，其中以A结尾的是ASCII函数，以W结尾的是UNICODE版本，Windows应用中的字符显示基本上都是调用它们实现的，如果我们能使用同样原形的自定义函数替换他们，我们就可以得到系统中所有的字符显示信息，从而使词条获取成为可能。我们面临的问题是如何找到这些函数并替换它们。

如何替换字符串显示函数 —Win32的DLL管理

在Win32环境下，DLL不是操作系统的一个部分，DLL被映射到进程的地址空间。由此可知，PE文件与DLL文件的内存映象与磁盘映象具有同样的结构。每个进程都有自己独立的DLL映像视图(Mapped View)，Windows将该DLL映射到各进程地址空间的不同地址。

当编译器与连接器产生可执行文件时，对DLL函数的任何调用被导向“操作映象表”，这个表由连接器在可执行文件内生成，每个可执行文件都有一个操作映象表，应用程序被执行时，Windows装入器将应用程序所需的DLL的映像视图连入，并修正操作映像表的入口项，应用程序利用该入口项实现对DLL中函数的调用。

因此，如果我们能找到系统中每个进程的操作映象表，将4个字符串显示函数的入口项指向我们自己定义的函数，就能够实现我们上面所说的被动获取技术。

使用代码注射技术打破进程边界

为了查找其他进程的操作映象表，必须访问其他进程的内存；为了使其他进程能调用我们自定义的字符串显示函数，必须使我们的函数位于该进程的地址空间内。而我们知道，在win32中跨越进程的地址值传递是没有意义的。假设我们自定义函数为MyTextOutA(以下我们将MyTextOutA作为其他三个自定义字符串显示函数的代表)位于进程A中，地址为0x77777777；我们将进程B的操作映象表中关于TextOutA的表项设为0x77777777，那么进程B对TextOutA的调用几乎肯定会引起错误，因为在进程B中，0x77777777指向的根本不是函数MyTextOutA。从上面的讨论看出，我们必须强制进程B将MyTextOutA装入其地址空间，这项技术被称做代码注入。我们可以将MytextOutA放在一个动态链接库中，然后强迫其他进程加载它，这样MyTextOutA便进入该进程的地址空间，但是，我们怎样强迫一个不是自己开发的应用程序的进程装入我们的DLL呢?系统钩子给我们提供了这种可能。

观察下面的Win32 API：SetWindowsHookEx()，最后一个参

数即是钩子响应函数所在的DLL句柄。假设我们安装了一个鼠标钩子，进程B收到鼠标消息后，即会尝试调用相应的钩子响应函数，当它发现该函数不在自己的地址空间内，便会主动调用LoadLibrary()函数将相应的DLL装入(这一切都由系统自动完成)，如果我们的MyTextOutA与钩子响应函数位于同一个DLL中，就“搭便车”进入了进程B的地位空间。

将MyTextOutA与鼠标事件联系起来

当鼠标指向系统中应用程序窗口的客户区或标题栏中的字符串时，如果能强迫它们调用MyTextoutA，我们就得到传给MyTextOut的字符串，屏幕取词就实现了。那么，如何实现这一设想呢?

假设我们已经安装了一个如上面所描述的鼠标钩子，当钩子函数被调用时，我们就得到了当前鼠标位置，调用Win32 API:WindowFromPoint，我们就可以得到当前位置所属窗口的句柄，进而调用API:InvalideRect将鼠标位置附近的一块矩形区域设为无效，并调用Update Window，这样Windows就会发出WM_PAINT消息给窗口函数，收到这条消息后，进程的窗口响应函数便会重画窗口，此时我们的MyTextoutA函数便被以合适的参数调用，屏幕取词技术就被成功地实现了。

我们看到，在屏幕取词过程中鼠标钩子充当了两个角色：代码注入与全局鼠标事件的捕获。

屏幕取词技术的实现

我们的程序由一个可执行文件和一个DLL构成。在DLL中，输出下列函数：钩子响应函数、钩子安装函数、MyTextOutA、ChangeEntry(操作映象表修改函数)，在可执行文件中，调用钩子装函数。文中给出钩子响应函数的流程图与MyTextOutA的流程图，如图1、图2所示。对图2有必要作以下说明，正常刷新是指窗口重叠、窗口大小、位置变化等程序正常工作引起的刷新，强制刷新是由我们调用图1中的InvalidateRect和Update Window引起的刷新。

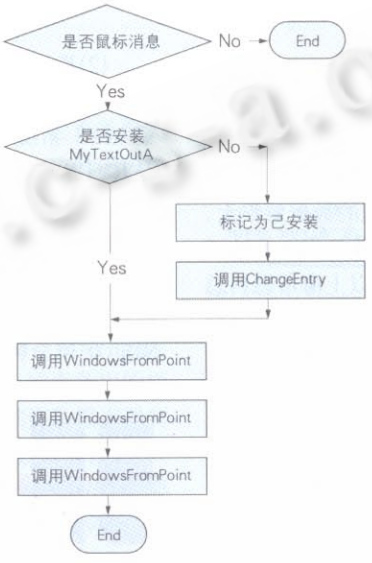

图1 钩子响应函数流程图

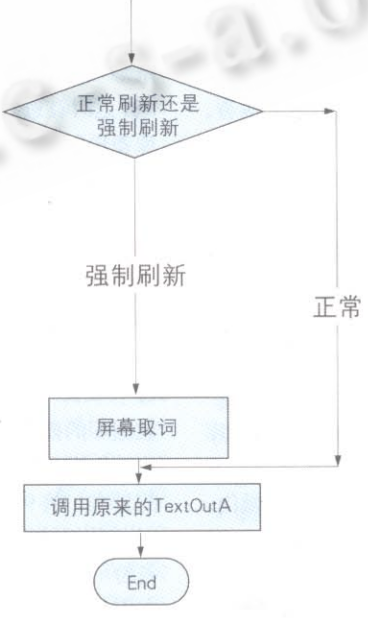

图2 MyTextOutA函数流程图

难点与细节

上面提到的其他技术对大多数Windows程序员都是非常熟悉的，而修改进程的操作映象表是主要的难点，下面主要来谈这个问题。修改操作映射表与其说是艰难不如说是繁杂，它涉及到对PE文件格式的理角，MSDN中关于PE格式的文档长达三十多页，全面的论述它们超出了本文的范围。作者在附录中给出一分修改操作映射表的代码，为了对PE格式有一个深入的理解，最好读一读MSDN中的相关文档。

如何判断MyTextOutA是否安装，如何辨别正常刷新还是强制刷新，可以通过在DLL中引入全局或静态变量作为标记来实现。在Win32环境中，DLL全局、静态变量使用Copy_On_Write机制，不用担心进程间的相互影响。■

参考文献

1 [美] Jeffrey Richter 著 郑全战 等译，Windows NT高级编程技术，清华大学出版社，1994.12

2 [美] Stefano Maruzzi 著 周靖 等译，The Microsoft Windows 95开发人员指南，机械工业出版社/西蒙与舒斯特国际出版公司，1997.1

3 Charles Petzold.Programming Windows 95. Microsoft press,1996