第 34 卷 第 8 期 计 算 机 工 程 2008 年 4 月
Vol.34 No.8 Computer Engineering April 2008
·工程应用技术与实现· 文章编号：1000—3428(2008)08—0255—03 文献标识码：A 中图分类号：TP303

基于驱动层的 USB 存储设备安全监控技术

李锦山，舒 辉，董卫宇，谢余强
(解放军信息工程大学信息工程学院，郑州 450002)

摘 要：针对 USB 存储设备提出基于驱动层安全监控的设计方案。介绍 Windows 驱动程序模型(WDM)的结构模型，分析 Windows 系统下经过 USB 集线设备 IRP 包的拦截技术，并利用下层过滤驱动和文件系统过滤驱动程序实现对 USB 存储设备的监控，给出 USB 存储设备安全监控软件的总体设计。

关键词：USB 存储设备；过滤驱动；监控

Security Monitoring Technology of USB Storage Device Based on Driver Layer

LI Jin-shan, SHU Hui, DONG Wei-yu, XIE Yu-qiang
(Department of Information Engineering, PLA Information Engineering University, Zhengzhou 450002)

【Abstract】 Aiming at USB storage device, this paper presents a design scheme of security monitoring based on driver layer. It introduces the architecture of Win32 Driver Model(WDM), and analyzes the technology of IRP tracking from USB hub device in Windows system and implements to monitor the USB storage device by lower filter driver and file filter driver. It describes the total design of the security monitoring system of USB storage device.

【Key words】 USB storage device; filter drive; monitoring

1 概述

近年来，由于 USB 总线较高的接口速率和灵活方便的使用特性，使得越来越多的存储设备使用 USB 接口来接入计算机，如闪存盘、移动硬盘等。基于 USB 接口的存储设备已取代软盘，成为一种重要的信息交换方式。但 USB 存储设备的广泛使用带来了许多安全隐患：(1)窃密者可以在合法用户不在场的情况下，利用 USB 存储设备快速地将个人隐私、国家机密或商业敏感信息取走，并且不会留下痕迹；(2)合法用户的违规操作和逾权动作可以把 USB 存储设备作为中转媒体；(3)USB 存储设备可作为病毒载体以及开机钥匙等。上述安全隐患都对主机安全构成了较大威胁。因此，对 USB 存储设备的安全实施监控具有重要意义。

目前很多保密单位在物理上禁用 USB 端口，这会给用户带来很多不便。而一些基于用户态的 USB 存储设备监控软件很容易被病毒、木马等恶意程序绕过，无法完成有效、实时的监控。本文采用基于驱动层的技术实现其主要的监控功能，相对应用层，更加安全和稳定。

2 Windows 驱动程序模型(WDM)

在 Windows 驱动程序模型(Win32 Driver Model, WDM)中，驱动程序的层和堆栈一起处理 I/O 请求包，如图 1 所示。

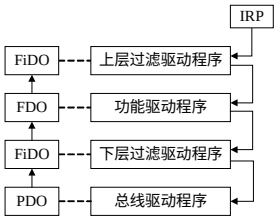

图 1 WDM 中设备对象和驱动程序的层次结构

WDM是Microsoft 公司的一种符合Windows2k/XP下的内核模式驱动程序的分层体系结构的驱动程序模式[1]。在这个模型中，

其中的设备驱动程序是指一系列控制硬件设备的函数，是操作系统中控制和连接硬件的关键模块。

图 1 左边是一个设备对象堆栈，设备对象是系统为帮助软件管理硬件而创建的数据结构，一个物理硬件可以有多个这样的数据结构。处于堆栈最底层的设备对象称为物理设备对象(Physical Device Object, PDO)。在设备对象堆栈的中间某处有一个对象称为功能设备对象(Functional Device Object, FDO)。在 FDO 的上面和下面还会有一些过滤器设备对象(filter device object)。位于 FDO 上面的过滤器设备对象称为上层过滤器，位于 FDO 下面(但仍在 PDO 之上)的过滤器设备对象称为下层过滤器。

当一个应用层程序需要读取磁盘文件数据时，首先会调用 WIN API 的接口函数，如 ReadFile。Win32 子系统模块(如 KERNEL32.DLL)通过调用平台相关的系统服务接口实现该 API，而平台相关的系统服务将调用内核模式支持例程。在 ReadFile 函数的调用中，调用首先到达系统 DLL(NTDLL.DLL)中的一个入口点——NtReadFile 函数。然后这个用户模式的 NtReadFile 函数接着调用系统服务接口，最后由系统服务接口调用内核模式中的服务例程。NtReadFile 将创建一个主功能代码为 IRP_MJ_READ(DDK 头文件中的一个常量)的 IRP。设备驱动程序最后会通过访问硬件来完成对该 IRP 包的

作者简介：李锦山(1982－)，男，硕士研究生，主研方向：网络信息安全；舒 辉，副教授；董卫宇，讲师；谢余强，教授

收稿日期：2007-05-22 **E-mail：**umbra33@126.com

处理。

对 USB 存储设备的所有操作也是通过给 USB 设备驱动发送相应的 IRP 包完成的。监控驱动程序可以拦截这些发往 USB 设备的 IRP 包，通过分析这些 IRP 包，修改用户的动作行为，来达到 USB 存储设备的全锁定、只读、只写以及读写的控制。

3 过滤驱动程序的设计

由上所述，在 WIN2000 下设备和驱动程序有堆栈式层次结构，这种栈式结构使 I/O 请求过程更加明了，每个影响设备的操作都会使用 IRP 包。通常 IRP 包会先送到设备堆栈的最上层(一般为上层过滤器驱动程序)，然后逐渐过滤到下面的驱动程序。每层的驱动程序都可以决定如何处理 IRP 包：(1)不做任何处理，直接向下传递该 IRP；(2)处理完该 IRP 包，不再向下传递；(3)既处理 IRP，又向下传递。这主要取决于设备及 IRP 包所携带的内容。

在 IRP 包的拦截过程中的一种主要技术是创建一个过滤驱动设备并将它加载到要拦截的设备所在的设备堆栈中。首先通过 IoCreateDevice 创建自己的设备对象，然后调用 AddDevice 函数获得需要拦截的设备对象指针，通过 IoAttachDeviceToDeviceStack 将过滤驱动设备对象加入到须过滤的设备堆栈上(这个设备对象就是 AddDevice 的第 2 个参数)，成为其中的一个过滤器。最后须给每种类型的 IRP 安装派遣例程，并不只是安装其希望处理的 IRP 类型。

3.1 驱动程序的安装

驱动程序安装分为 2 步：

(1)添加 USB 过滤驱动的服务键：首先须将驱动文件 USBFilter 拷贝到 system32\drivers 目录下，因为 Windows 在启动时，只能找到 Windows 目录(此时磁盘驱动没有完全加载，文件系统没有建立，无法访问其他位置)，否则驱动将无法加载；然后调用 CreateService 函数创建服务 USBFilter。服务的类型必须是 SERVICE_KERNEL_DRIVER(1)，服务启动类型必须是 SERVICE_BOOT_START(0)，否则在重启时将出现蓝屏现象。

(2)在类设备“磁盘驱动器”下添加过滤驱动键：调用 SetupDiBuildClassInfoList枚举系统中所有的类设备，访问其设备键(位于 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Class\{4D36E967-E325-11CE-BFC1-08002BE10318}“磁盘驱动器”)，新建“多字符串值”，名称为 LowerFilters，键值为USBFilter，驱动将过滤该类设备。

3.2 IRP 包的拦截

3.2.1 下层过滤驱动的 IRP 包拦截

该驱动主要实现了 USB 存储设备的禁用、只读和读写的功能。

当USB存储设备接入主机时，系统将调用“磁盘驱动器”为该设备分配地址空间和设备卷[2]，USBFilter过滤驱动程序会被自动加载并执行AddDevice函数。

由于是针对USB端口的下层过滤驱动，因此拦截写操作时就不能过滤主功能号为IRP_MJ_WRITE的IRP包，否则会无法拦截该IRP，因为到了下层过滤驱动所有对USB存储设备的操作都会转变为主功能号为IRP_MJ_SCSI的IRP包[3]，所以这里需要拦截的IRP也就是IRP_MJ_SCSI。

得到当前 SCSI 命令过程如下：使用 IoGetCurrentIrpStackLocation 函数得到与该 IRP 相关联的 I/O 堆栈单元，得到 Parameters.Scsi.Srb 参数，这是一个 PSCSI_REQUEST_BLOCK 结构，设为 PCSrb，其中一个参数为 Cdb，结构是 PCDB，根据 PCDB 的结构得到当前 SCSI 命令 opCode=cdb->CDB10.OperationCode。

分析当前 SCSI 命令过程如下：当 opCode 的值为 SCSIOP_MODE_SENSE(0x1A)时，判定为写操作，取 PCSrb->DataBuffer，结构为 PMODE_PARAMETER_HEADER，其中的一个参数 DeviceSpecificParameter 与 MODE_DSP_WRITE_PROTECT(0x80)或运算，就可以限制写操作了。如果需要全锁定，则直接调用 IoCompleteRequest 处理该 IRP 包。返回 STATUS_ACCESS_DENIED。

整个控制具体实现流程见图 2。

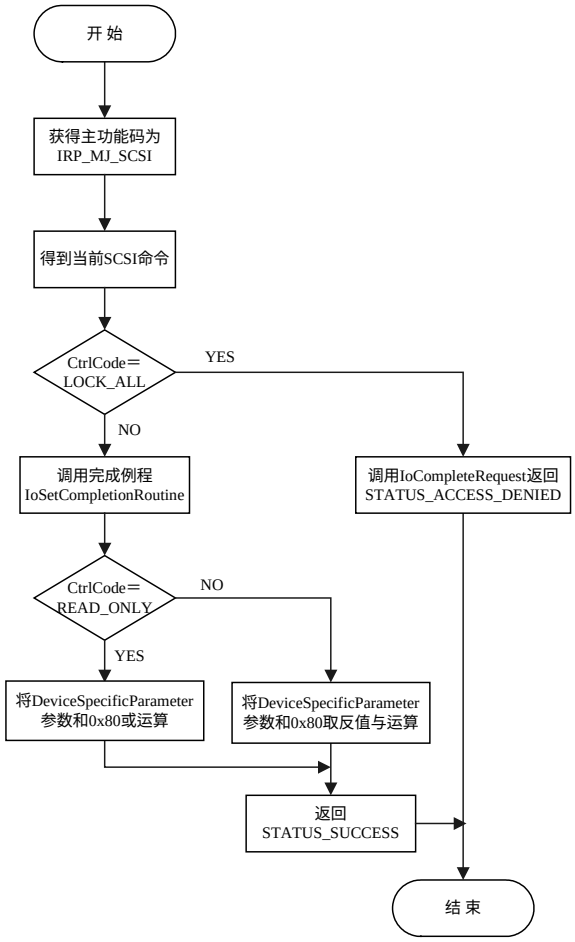

图 2 只读(禁用、无限制)控制实现流程

3.2.2 文件系统过滤驱动的 IRP 包拦截

该驱动主要实现了 USB 存储设备的只写控制。

首先得到需要过滤的设备对象：根据 USB 设备的符号连接(Symbol Link)名，使用 ZwCreateFile 函数得到文件句柄，ObReferenceObjectByHandle 函数得到文件对象，IoGetRelatedDeviceObject 得到文件设备对象；然后使用 IoAttachDeviceByPointer 将过滤驱动设备加载到该设备堆栈中。

通过实验发现：

(1)创建一个文件的操作不仅产生 IRP_MJ_WRITE，同时也产生了 IRP_MJ_READ，且 IRP_MJ_READ 是在 IRP_MJ_WRITE 之前发出的，如果只在拦截到 IRP_MJ_ READ 时就直接处理掉该 IRP 包，将导致文件创建的失败。因此，需要区分这 2 种操作，才能分别进行处理。

(2)进一步发现在创建文件时，首先会发一个主功能码为 IRP_MJ_CREATE 的 IRP 包，并且是在 IRP_MJ_READ 之前

发出的。因此，只写控制需要拦截的 IRP 包是 IRP_MJ_CREATE 和 IRP_MJ_READ。

分析 IRP_MJ_CREATE 请求包的参数信息时，分别进行了读操作和创建操作，发现了读操作和创建操作的不同之处，在创建文件时，IRP 包中参数 Parameters.Create.Options 的高两位值为 FILE_CREATE(2), FILE_OPEN_IF(3) 或 FILE_OVERWRITE_IF(5)。

具体实现流程见图 3。

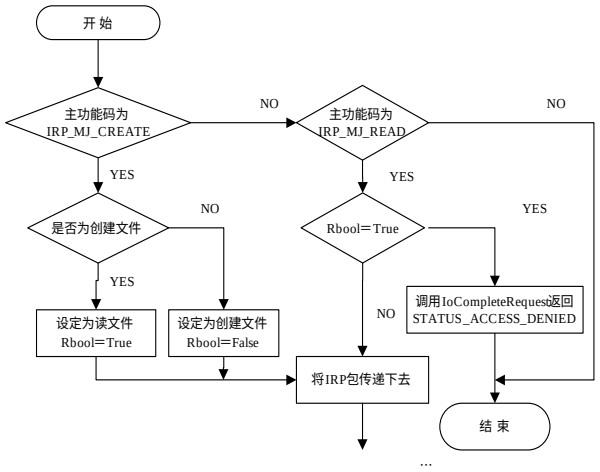

图 3　只写控制实现流程

4　USB 存储设备安全监控软件的总体实现

软件由客户端服务程序和驱动程序组成，客户端服务程序负责应用数据库记录 USB 存储设备的使用情况，包括接入时间、设备名、大小及所采取的控制方式等，并向驱动程序传递监控命令；驱动程序负责控制 USB 存储设备的只读、只写、全锁定和读写。软件在加入了驱动程序后的结构示意如图 4 所示，图中阴影部分为软件添加。

客户端服务程序与驱动程序的通信过程如下：(1)应用程序创建一个 Event 对象；(2)应用程序通过 DeviceIoControl 函数传递命令给驱动程序；(3)驱动程序接到命令后执行操作；(4)应用程序调用 DeviceIoControl 函数从驱动程序得到反馈信息；(5)返回(2)，循环处理。

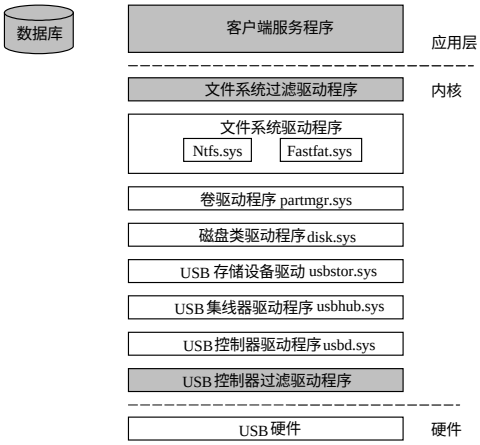

图 4　软件结构

5　结束语

本文提出基于驱动层的 USB 存储设备监控方法，成功解决了保密要求较高的个人和单位只能在物理上禁用 USB 端口、使用不灵活的问题。在 WIN2000\2003\XP 系统下的试验证明，此软件可以可靠、稳定地监管 USB 端口。软件关键功能的实现都在驱动层完成，可优先监控到 USB 储存设备，不易被恶意程序绕过，软件资源的占有率低。

参考文献

[1] Walter O. Programming the Microsoft Windows Driver Model[Z]. 1999.

[2] Axelson J. USB 大全[M]. 陈　逸, 译. 北京: 中国电力出版社, 2001.

[3] Znsoft. 求助磁盘下层过滤驱动的问题[Z]. http://bbs.driverdevelop.com/read.php?tid=107076&page=1.

(上接第 239 页)

4　结束语

本文在已有研究成果的基础之上，对误差控制、大块数据处理、地形简化等算法和实现技术进行了改进。使用包围盒进行 LOD 误差处理，大大地提高了算法的效率，该算法已经应用到研制的某卫星网管仿真测试系统中，取得了良好的效果。

算法使用的是存储在本地的地形数据，未来有可能通过网络使用实时的真实地形数据，此时内存映射文件将失效，因此需要进一步考虑如何有效地加载和传输来自网络的数据，从而实时显示真实的地形。另外，使用 C++语言实现算法和地形显示，可以进一步提高软件的效率。

参考文献

[1] 赵友兵, 石教英, 周　骥, 等. 一种大规模地形的快速漫游算法[J]. 计算机辅助设计与图形图像学报, 2002, 14(7): 624-628.

[2] 马照亭, 潘　懋, 胡金星, 等. 一种基于数据分块的海量地形快速漫游方法[J]. 北京大学学报: 自然科学版, 2004, 40(4): 619-625.

[3] Lindstrom P, Koller D. Real-time, Continuous Level of Detail Rendering of Height Fields[C]//Proc. of SIGGRAPH'96. Los Angeles, California, USA: [s. n.], 1996: 109-118.

[4] Lindstrom P, Koller D. Level-of-Detail Management for Real-time Rendering of Phototextured Terrain[EB/OL]. (1995-06-01). http://www.gvu.gatech.edu/people/peter.lindstrom/papers/tr/#95-06.

[5] Bradley D. Evaluation of Real-time Continuous Terrain Level of Detail Algorithms[EB/OL]. (2003-08-08). http://www.derekbradley.ca/Papers/carleton03_TerrainLOD.pdf.

[6] Röttger S. Real-time Generation of Continuous Levels of Detail for Height Fields[C]//Proc. of the 6th International Conference in Central Europe on Computer Graphics and Visualization. Plzen, France: Science Press, 1998: 315-322.

[7] Hoppe H. Smooth View-dependant Level-of-detail Control and Its Application to Terrain Rendering[C]//Proc. of IEEE Visualizaion. NC, USA: [s. n.], 1998: 35-42.

[8] Hoppe H. Progressive Meshes[J]. Computer Graphics, 1996, 30(2): 99-108.

[9] 普建涛, 查红斌. 大规模复杂场景的可见性问题的研究[J]. 计算机研究与发展, 2005, 42(2): 236-246.

[10] Kirmse A. Game Programming Gems4[M]. 沙　鹰, 译. 北京: 人民邮电出版社, 2005.

[11] Richter J. Windows 核心编程[M]. 4 版. 王建华, 译. 北京: 机械工业出版社, 2005.