第 36 卷 第 4 期 计 算 机 工 程 2010 年 2 月
Vol.36 No.4 Computer Engineering February 2010
·博士论文· 文章编号：1000—3428(2010)04—0001—03 文献标识码：A 中图分类号：TP309

基于 HID 的 USB 监控技术的设计与实现

卢志刚[1]，刘建华[2]，刘宝旭[1]，许榕生[1]
(1. 中国科学院高能物理研究所计算中心，北京 100049；2. 北京市国家保密局，北京 100743)

摘 要：描述分布式 USB 监控系统的设计与实现。针对 USB 驱动开发复杂和通用性较差的问题，提出利用 IRP 拦截的方法实现一个扩展的 USB HID(人工接口设备)数据访问接口，在不影响接口 I/O 效率的前提下实现对硬件数据的侦听，并以仿真实验进行验证。实验结果表明，该方法能简化驱动开发流程，提供数据的封装和扩展接口。

关键词：人工接口设备；USB 监控；硬件数据侦听；扩展接口

Design and Implementation of USB Monitoring Technique Based on HID

LU Zhi-gang[1], LIU Jian-hua[2], LIU Bao-xu[1], XU Rong-sheng[1]
(1. Computing Center, Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049;
2. Beijing State Secrecy Bureau, Beijing 100743)

【Abstract】This paper presents the design and implementation of a distributed USB monitoring system. Aiming at the complexity and low commonality about development of USB driver, the system implements a scalable USB HID(Human Interface Device) data access interface by using the method of IRP interception. It achieves a technique to sniff the data of hardware under the premise of non-impact on effective of I/O transmission, and takes a simulation experiment as an example to test this method. Experimental result shows that this method simplifies the development of the drive, and supplies the encapsulation of data and a scalable interface.

【Key words】Human Interface Device(HID); USB monitoring; hardware data sniff; scalable interface

1 概述

近年来，由于 USB 设备管理不当而导致的安全事件频繁发生，究其原因，一方面 USB 接口已被各种流行的操作系统所支持，并应用于各种数据终端设备中，且由于 USB 接口设备终端用户的易用性、应用的广泛性、成本的低廉性，目前已成为计算机外围存储设备的主导方式；另一方面，USB 接口本身易用性和灵活性的特点给计算机系统带来了一定的安全隐患。由于对计算机 USB 端口缺乏有效管理，针对 USB 存储设备的木马已经比较流行，因此给计算机系统的安全性带来了很大隐患。本文提出了一种分布式 USB 监控管理，通过人工接口设备(Human Interface Device, HID)类扩展实现对 USB 硬件接口的侦听，在不影响 Client 用户计算机使用效率的前提下，提供统一的分布式 USB 端口监控方法，增强系统和数据的安全性。

2 USB/HID 类简介

USB 接口协议[1]是目前使用最广泛的通用串行总线协议，从最初的 USB1.1 到 USB2.0 及已经出现的 USB3.0[2]协议，USB 接口协议取得了很大的发展。但对于 USB 设备来说，现在比较普遍的是采用 Windows 驱动程序模型(WDM)[3]的方法实现 USB 接口的开发，或利用 Linux 中的 USB 核心子系统提供的大量 API 以及相关的支持机制进行驱动程序开发[4]，但开发驱动程序过于复杂。计算机系统中的 HID 类是为一些人工的输入或输出设备而设计的，连接到计算机的 USB 设备几乎都包含 HID 类，用于信号控制。对于 HID 类的设备，操作系统已经给出了相应的驱动程序，提供相应接口的对应描述。通过对 HID 类的扩展，实现 USB 的设备、配置、接口、端口、字符串描述以及其他必要的描述，并可以在现有 HID 类的基础上，提供一个可以扩展的通用 API 接口，支持其他需要的 USB 描述数据结构。

3 体系框架与基本原理

局域网或者广域网中的每个需要进行 USB 监控的计算机视为一个节点(node)，每个 node 中的 Client 完成对 USB 接口总线的硬件侦听，侦听的结果通过 Client 模块中的处理程序通过 Socket 通信发送给统一管理的 Server 端，完成对 USB 使用情况的解析和记录，并进行相应的数据记录。

系统采用 C/S 控制模式，进行分布式的统一管理控制。框架由 3 个部分组成，框架结构见图 1。由图 1 可以看出，整个系统架构主要由 USB 监控客户端(Client)和 USB 管理控制端(Server)组成。在任何 USB 进行动作时，系统的 I/O 管理器都会向 USB 端口发送 IRP 进行控制。在设备驱动层拦截系统的 IRP 信息，如果在 IRP 中发现了 USB 控制信息内容，将 IRP 内容传递给 USB 监控客户端中的 Application 进行数据封装和传输。HID Class 根据所获得的描述信息获取相应 USB 端口的配置信息。Monitor Application 要提供连接 Application

基金项目：国家“863”计划基金资助项目(2006AA01Z410)；北京市自然科学基金资助项目(4072010)
作者简介：卢志刚(1983－)，男，博士研究生，主研方向：网络信息安全；刘建华，高级工程师、硕士；刘宝旭，副研究员、博士；许榕生，研究员、博士、博士生导师
收稿日期：2009-08-20 **E-mail：**luzg@ihep.ac.cn

与 System 的 API 接口，获取 HID Class 传递的 USB 信息内容后，加以解析，按照预定的 TAG 进行数据结构的定义，发送至 Monitor System 进行解析和存储。

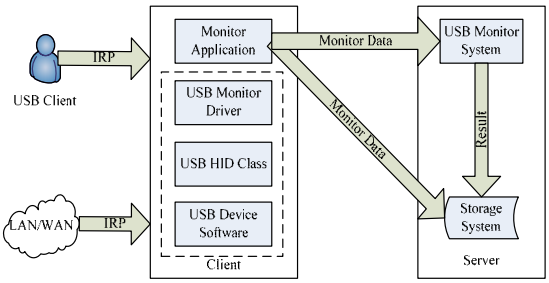

图 1　USB 监控框架结构

在该框架中，IRP 内容监控与在 HID Class 的扩展定义与信息获取是核心，并且能够在不影响总线效率的情况下，实现 USB 总线数据的侦听获取。这部分在 USB HID Class 中加以实现，提供一个扩展的 API 接口，对用户透明。而与具体存储系统交互的功能交由 Server 端的控制程序进行处理，并对 Client 端发送的数据包进行解析及展现用户界面。

4　系统的设计与实现

4.1　IRP 拦截及 USB 硬件数据侦听

WDM 是 Windows NT 系统都支持的驱动模型，在 USB 接口监控开发中得到了比较广泛的应用。在 WDM 驱动程序模型中，操作系统用物理设备对象(Physical Device Object, PDO)表示一个物理设备，并分配 PDO 各种描述属性。系统检测到 PnP 设备或 PM 事件时，发送 IRP 请求到 PDO，驱动程序可在 FDO 中拦截这些请求，作相应处理后再发送到 PDO 中。Windows NT 系统设备驱动程序依靠 PDO 存取系统总线驱动程序，驱动程序通过向 PDO 发送 IRP，实现与总线驱动程序的通信。

IRP 在 Windows DDK[5]中进行较好的封装，包含了很多 IRP 操作的数据成员，当 I/O 管理器收到一个 USB 请求，系统就会初始化一个相应的 IRP。对总线驱动、USB 设备驱动与 I/O 设备管理器间的 IRP 通信进行侦听，向 USB HID 类传递对应的功能代码，对要监控的 USB 接口进行定义。

为了实现数据传输，USB 内核提供了一个称为 URB(USB Request Block)的数据结构。一个 URB 由执行任何一个 USB 事务信息、分发数据信息和回传的状态信息组成。URB 中具有 USB 数据传输的所有信息，包括传输类型、传输方向、数据缓存区、数据传输的设备、端点、返回信息及指向传输完成的处理函数的指针。

监控程序侦听 USB 发送的 IRP 请求，从总线拷贝出对应的 URB 结构体内容，用户自己可以简化定义相应的监听 URB 数据结构，如下所示：

```
Stactic typedef struct urb_irq_interface
{
  spinlock_t lock;            /*lock the URB*/
  void *private_data;         /*data for host controller*/
  struct list_head urb_list;  /*list pointer to urb*/
  struct usb_device *dev;     /*associated USB device*/
  unsigned int pipe;          /*pipe information*/
  int status;                 /*returned status*/
  void *transfer_buffer;      /*data buffer*/
  int transfer_buffer_length;       /*buffer length*/
  int actual_length;          /*actual data buffer length*/
  int number_of_packets;      /*number of packets*/
  int interval;               /*polling interval*/
  int timeout;                /*timeout*/
  usb_complete_t;             /*completion time*/
  struct timeval completion_time; /*time stamping*/};
```

在数据侦听过程中，Client 根据定义的 URB 的这一映射关系传递具体的 USB 端口数据内容，从而使具体的 USB HID 扩展类可以接收并处理 URB 所获得的数据和请求。操作流程中需要与存储系统交互的地方亦可以由用户自己定义，在 URB 的数据结构中定义 API 接口，当数据操作完成时，向 Server 报告运行状态。

通过从 USB 总线上完全复制传输数据信号，对数据结构进行分析，并把相应数据传送到监控应用层定义的数据结构体中，完成 USB 硬件数据侦听。在这一过程中，定义 FILE_CREAT, FILE_READ, FILE_WRITE, DEVICE_IO_CONTROL，HANDLE_STOP 这 5 个基本操作完成对 USB 硬件设备的数据侦听与控制。5 个操作实现的简单描述如下：

(1)FILE_CREAT 操作。新 USB 设备开始活动时，比如一个新的 USB 存储设备加入 USB 总线，为 URB 提供一个从 IRP 获取信息进行初始化工作的机会。此时，可以打开设备且创建到设备的连接，初始化对设备的描述信息等。

(2)FILE_READ 操作。从 IRP 获取信息初始化成功后，FILE_READ 从 USB 驱动内核中读取 URB 中的数据，WDM 驱动有 FILE_ATTRIBUTE_NORMAL 数据获取方式，以同步方式打开设备，实现驱动层数据的同步获取，将数据写入内存缓冲区中，返回数据内存的地址。

(3)FILE_WRITE 操作。主要完成 2 个操作：1)从 FILE_CREAT 中获取打开 USB 设备的全局 GUID 标志，在设备接口中，每个设备由一个 128 位的全局唯一标识符 GUID 标志，在这里主要用于标志所操作的 USB 设备；2)从 FILE_ READ 返回参数中获取其写入内存的指针地址，遍历内存，分析数据，封装成 urb_irq_interface 结构体数组，供 USB HID 使用。

(4)DEVICE_IO_CONTROL 操作。当 FILE_CREAT 调用成功后，即设备被打开后，应用程序就可以调用 DEVICE_IO_CONTROL 操作与设备驱动程序通信。程序根据 I/O 控制命令决定如何获取 USB 驱动内核中的 URB 地址。

(5)HANDLE_STOP 操作。当数据拷贝完成时，根据 FILE_WRITE 操作返回值，释放占用的各种系统资源，包含所占用的系统内存开销以及获取的 GUID 标志，这个操作与 FILE_CREAT 相对应。

4.2　USB HID Class 类扩展

USB HID 扩展类使用标准请求 Get_Descriptor 对接口设备包括 USB 设备进行描述，当 Get_Descriptor 被调用时，它将定义如下 3 个描述：(1)返回接口描述，确定所使用的设备类型及使用哪一个 HID 基础类类型；(2)HID 类本体描述，经过上一步确定 USB 设备类型，这里使用 USB HID 的详细类描述，包含设备、接口类型、终端和其他附属描述信息，分别对应 GET_DESCRIPTOR, Descriptor Type and Descriptor Index, GUID 和 DATA 4 种数据类型；(3)结构数据获取，对应于上一节定义的 URB 类，使用 GUID 作为对应的 USB 接口标识，使用 FILE_WRITE 操作对 USB 传输数据进行描述。根据 urb_irq_interface 结构的定义，提供如下的 API：

```
USB_HID descriptor
{Get_Descriptor_Request (returns a descriptor for the device);
```

HID descriptor Request (associated with above Interface);

Get_Descriptor Request (get information through urb_irq_interface);}

不同的应用程序开发与硬件设备驱动之间的接口和兼容不同，不同的数据类型之间也有不同的接口，增加了开发难度。在上述设计的基础上，笔者实现了应用层软件与设备驱动的数据访问接口，在原有 HID 类的基础上，扩展出可以由上层应用软件访问的 API，对数据结构进行封装，提供了一种规范化的、可扩展的数据监控接口设计，见图 2。

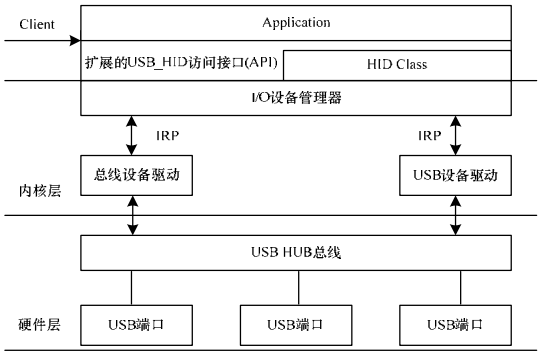

图 2　可扩展的数据监控接口设计

从 HID Class 扩展出的类内容提供相应的 API 给应用层软件进行调度，Client 对从 API 获取的内容进行满足软件本身定义的格式封装，大大简化了底层驱动接口的开发难度。目前 HID Class 并不能够提供相应的数据及接口主机的标识符，但由于 C/S 系统的需要，因此需要标志 USB 监控的追溯能力。笔者在 USB HID 访问接口提供了获取 USB 操作发生时间、计算机名称、IP 地址等应用扩展接口，提供了如下 API：ghid_date, ghid_time, ghid_address, ghid_cmptername; ghid_hdsrid。这些 API 能够在获取 USB 接口动作内容的同时，明确对计算机的定位，在实际应用中具有较大的实际意义，同时也提供了扩展接口的利用范例。

5　实验结果及分析

为了验证上述方法的准确性和效率，本文构建了图 3 所示的实验网络。由于在开发过程中对传输的加密并没有严格定义，系统缺乏有效的安全机制，因此只在局域网内进行实验，但实验不需要考虑网络中 Client 与 Server 数据传输的延迟，对性能分析不会造成太大影响。在本实验框架下，笔者分析的是系统处于监控状态时对 USB 接口传输效率的影响。

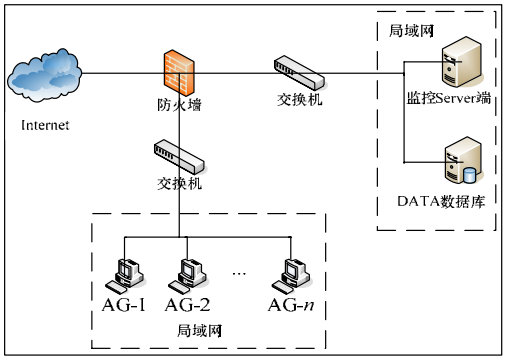

图 3　网络实例拓扑

在相同硬件配置的基础上分别进行 3 次实验，测试在不启用监控和启用监控的情况下各种不同文件大小传输速率。实验结果如表 1 和表 2 所示。传输速率比较结果如图 4 所示。

表 1　未启用监控下文件传输时间

文件大小/MB	第 1 次测试时间/s	第 2 次测试时间/s	第 2 次测试时间/s	平均传输时间/s
1	0.97	0.84	0.87	0.893
4	2.28	2.34	2.37	2.330
16	4.41	4.57	4.50	4.493
64	15.04	15.34	15.65	15.343
256	59.24	58.30	58.46	58.667
512	117.15	118.06	118.50	117.903
1 024	234.05	234.56	235.11	234.573

表 2　启用监控下文件传输时间

文件大小/MB	第 1 次测试时间/s	第 2 次测试时间/s	第 3 次测试时间/s	平均传输时间/s
1	1.18	0.82	0.89	0.963
4	2.27	2.30	2.29	2.287
16	4.20	4.28	4.23	4.237
64	15.04	15.11	15.75	15.300
256	58.31	58.55	58.60	58.487
512	117.00	117.34	118.01	117.450
1 024	235.67	235.39	235.55	235.537

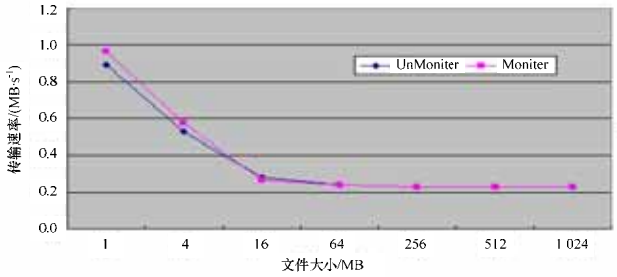

图 4　2 种情况下 USB 传输效率比较

实验数据显示，在大量数据传输时，监控程序对 USB 端口传输效率的影响很小，几乎可以忽略不计。在进行小文件传输时，由于文件句频繁的柄建立和释放，对系统的资源有一定占用，损失了约 5%的 USB 接口 I/O 读写性能，但总的时间在可以接受的范围之内。

在进行数据传输效率分析时，由于系统自身资源的不稳定性及硬件设备的不同，因此传输效率在测试过程中会出现一定的波动，例如系统服务的忽然启动，这些会造成额外的开销，对速率测试造成影响。

6　结束语

本文系统在基本不影响 USB 总线数据通信的基础上，实现了硬件的侦听功能，并且提供了扩展的 USB HID 数据访问接口，提供了数据的封装和扩展接口，在基本不影响 I/O 效率的情况下，实现了对 USB 访问数据的监控。随着 USB3.0 的逐渐普及，USB 在计算机接口中的地位和作用更加重要，下一步将研究 3.0 协议下的数据监控方法及监控其他计算机端口的数据传输，进行通用模型的扩展。

参考文献

[1] Compaq, Intel, Philips. Universal Serial Bus Specification Revision2.0[EB/OL]. (2000-10-10). http://www.usb.org.

[2] 武安河, 邰　铭, 于洪涛. Windows 2000/XP WDM 设备驱动程序开发[M]. 北京: 电子工业出版社, 2003.

[3] Herranz A. USB 3.0[J]. PC World Professional, 2008, 10(5): 136-139.

[4] 杨　伟, 刘　强. Linux 下 USB 设备驱动研究与开发[J]. 计算机工程, 2006, 32(19): 283-284.

[5] Microsoft Corporation. Windows 2000 DDK Document[EB/OL]. (2008-10-15). http://www.Microsoft.com.

编辑　张正兴