计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(11):76−81 [doi: 10.15888/j.cnki.csa.006039]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Thin Hypervisor 的 USB 设备访问控制①

刘　欢[1,2], 马恒太[2], 赵　培[2]

[1](中国科学院大学, 北京 100049)
[2](中国科学院 软件研究所, 北京 100190)

摘　要: USB 移动存储设备体积小、容量大、便于携带等优点, 被广泛应用于数据的传输和备份. 但是 USB 移动存储设备的这些特点也给数据的保护带来了很大的挑战. 因为盗窃数据者可以轻易的利用 USB 移动存储设备带走数据. 目前存在的针对 USB 存储设备访问控制的研究, 主要基于应用层或操作系统内核层. 当系统中存在恶意代码时, 这些安全访问控制实施的模块很容易被旁路. 为解决实施模块的安全性问题, 实现了一种基于 Thin Hypervisor 的 USB 存储设备安全访问控制系统, 它利用 Thin Hypervisor 对操作系统透明的特点, 使得该系统不受操作系统安全性的影响, 从而达到更加安全的目的.

关键词: USB 存储设备; 访问控制; Thin Hypervisor; 数据安全

引用格式: 刘欢,马恒太,赵培.基于 Thin Hypervisor 的 USB 设备访问控制.计算机系统应用,2017,26(11):76–81. http://www.c-s-a.org.cn/1003-3254/6039.html

Access Control on USB Mass Storage Devices Based on Thin Hypervisor

LIU Huan[1,2], MA Heng-Tai[2], ZHAO Pei[2]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: USB mobile storage devices are widely used to transfer and exchange data for their small size and large capacity. These features provide challenges for us to protect our confidential information because thieves can take secrets away by USB storage devices easily. At present, there are many studies on how to protect confidential data on USB storage devices. Most of these studies are based on application layer or operating system layer. When there are malicious codes on operating system, the operations protecting confidential information can be easily bypassed by attackers. In this paper, we present a USB devices access control system which is implemented on a thin hypervisor. The thin hypervisor is transparent to OS, which can guarantee that the security of the system is independent of OS, so that the system can be more secure.

Key words: USB storage device; access control; Thin Hypervisor; data security

随着信息技术的发展, 人们需要进行大量的数据存储与数据交换. USB 移动存储设备以它体积小、容量大、易插拔、易携带等优点, 目前被广泛的使用. 但是由于 USB 病毒以及恶意的 USB 硬件设备的出现给计算机系统数据的安全带来了很大的威胁[1,2]. 而且 USB 设备的特点也使得数据盗窃者可以很方便的利用 USB 移动存储设备带走机密数据, 而不被轻易的察觉. 通过上述分析, USB 存储设备安全访问控制的研究对数据保护具有很重要的意义.

目前针对该问题的研究主要分为两部分. 其中一部分是在 USB 设备访问控制中针对如何改进设备安全协议身份认证机制的研究[3,4]. 另一部分主要是针对

① 基金项目: 国家“核高基”科技重大专项 (2014ZX01029101-002)
收稿时间: 2017-02-14; 修改时间: 2017-03-06; 采用时间: 2017-03-09

USB 设备访问控制模块在系统中实现方法的研究. 而目前针对实现方法的研究主要是基于应用层或系统内核层. 并且大部分是针对特定操作系统类型的, 它们主要利用操作系统提供的有关设备的 API 函数或者操作系统本身的特点来进行 USB 设备数据保护. 目前存在的针对 Linux 操作系统的研究主要有: 利用 Linux 操作系统的驱动层进行 USB 设备监控[5], 以及利用 Linux LSM 安全框架对 USB 设备进行挂载、读、写等操作的控制[6,7]; 针对 Windows 操作系统的研究主要有: 利用 Windows 提供的 API 函数与系统驱动层进行通信来完成 USB 设备安全访问控制[8]以及基于 Windows 平台开发对 USB 设备授权管理的应用程序[9]. 由于这些研究主要是在应用层和操作系统内核层实现的, 当系统中存在恶意代码时, 这些安全访问控制模块很容易被旁路, 并且针对特定类型操作系统的研究, 使得系统的可应用性不好.

针对上述研究中存在的问题, 本文中主要实现了一种基于 Thin Hypervisor 的 USB 存储设备的访问控制系统, 该系统利用 Thin Hypervisor 对操作系统透明的特点, 使得访问控制系统的安全性独立于操作系统的安全性, 安全性更高. 同时该系统不限于操作系统类型, 使得该系统应用性更好.

1 系统概述

本系统主要分为两部分: 可信终端和 USB 存储设备数据管理中心. 系统总体框架图如图 1 所示.

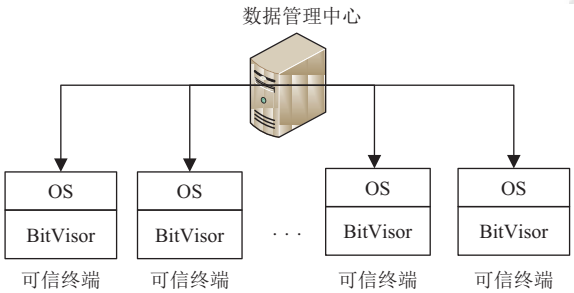

图 1 系统框架图

可信终端主要是安装了 Thin Hypervisor 的主机. 该系统选用的 Thin Hypervisor 是 BitVisor[10]. BitVisor 是一个半穿透的薄虚拟机监控器. 当有敏感事件发生时, 例如: 当执行 I/O 指令时, 处理器执行的内容会自动从 Guest OS 转换到 BitVisor. BitVisor 利用 Guest OS 的设备驱动去处理设备与操作系统之间的数据传输. 它让大多数的设备 I/O 访问直接穿透, 而只是在它内部实现了一个很小的驱动集合去拦截特定种类设备的 I/O[10]. 这样设计的半穿透性薄虚拟机监控器大大减少了虚拟机监控器的代码量, 所以相比较于 XEN、VMware Server 等虚拟化架构, BitVisor 的可信计算基 (TCB) 小, 安全性更高. BitVisor 上只支持一个 guest OS, 不用考虑一个平台上多个 Guest OS 之间对 USB 访问权限不同而导致的数据安全性问题, 以及减少了多个 Guest OS 之间 USB 设备数据共享所带来的性能上的开销. 而且一般的办公环境一个 Guest OS 完全够用, 所以使用 BitVisor 也不会带来不便. 根据上述 BitVisor 这些特点我们选用 BitVisor 作为可信终端 Thin Hypervisor.

当安装了 BitVisor 的可信终端发生 USB 存储设备插入的事件后, 会将该 USB 设备 ID 发送给数据管理中心. 当数据管理中心收到可信终端发来的设备 ID, 会根据设备 ID 得到该设备的安全访问权限, 然后将设备的访问权限发送回可信终端.

在该系统中只有在数据管理中心注册过的设备才能在系统中使用. 所以该系统使得数据对外界 USB 设备是安全的.

2 数据管理中心的设计与实现

数据管理中心是保存所需信息的服务器, 其中主要保存了可以在系统中使用的 USB 存储设备的白名单, 以及每个设备所对应的设备安全访问权限. 我们用供应商 ID(VID), 产品 ID(PID)[11] 和硬件序列号 (Serial) 来作为设备 ID, 在白名单中标识设备的唯一性. 设备的安全访问权限主要划分为四类: 不可识别, 只读、只写、读写. 当服务器端接收到可信终端发来的设备 ID 时, 会在白名单中查找对应的设备 ID, 如查找不到, 则返回禁止设备识别的安全访问权限, 若查找到, 则返回白名单中该设备所对应的访问权限.

在数据管理中心, 只有中心管理人员可以根据需求修改管理的设备 ID 以及修改 USB 设备对应的安全访问权限.

3 可信终端设计与实现

系统中可信终端的模块图如图 2 所示.

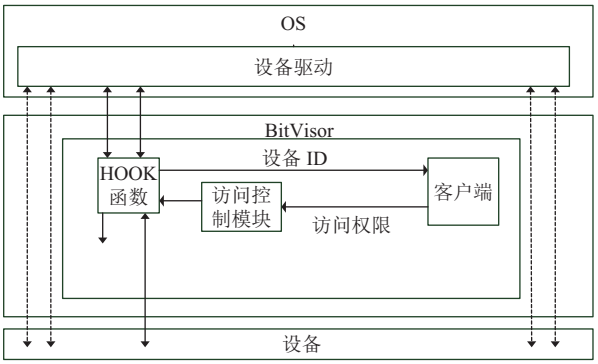

图 2 可信终端模块图

可信终端主要是安装了 Bitvisor 的计算机, 可信终端中所有的 USB 存储设备安全访问的操作全部在 BitVisor 中实现. 其中我们主要实现了一个客户端模块, 一个访问控制模块, 以及 I/O 拦截模块. 在 BitVisor 中有很多 HOOK 函数, I/O 拦截模块中的操作主要是在相关的 HOOK 函数中实现的.

当 OS 中插入 USB 存储设备时, BitVisor 中 HOOK 函数 new_usb_device 被调用, 该函数主要是在 BitVisor 中生成一个 USB 存储设备对应的设备对象, 保存设备的所有信息. 在这个函数中我们通过设备对象中的设备描述符得到设备的 ID. 将设备 ID 通过客户端发送给服务器端, 然后客户端接收从服务器端返回的设备访问权限: 只读、只写、读写、不可识别. 将设备对应的访问权限保存到访问控制模块中. 当设备发生读写等操作时, 相应的 HOOK 函数会被调用, 里面实现的拦截模块根据访问控制模块中的访问权限对数据流进行相应的拦截, 而不被拦截的数据流才会进入设备中.

3.1 客户端

客户端主要是在 BitVisor 中与服务器端进行通信的模块. 它利用 BitVisor 中现有的 LWIP 通信协议栈的接口实现. LWIP 协议不需要操作系统的支持. BitVisor 利用实现的客户端与服务器完成主动通信的工作, 而不需经过操作系统, 所以可信终端与 USB 设备服务器通信过程对 OS 层完全透明. 当发生 USB 设备插入的情况时, 该客户端将设备 ID 发送给服务器. 在没有收到服务器端的访问权限的情况下, 设备不被操作系统所识别.

3.2 访问控制模块

访问控制模块主要是保存从服务器端得到的相应设备的安全访问权限. BitVisor 中每当插入一个 USB 存储设备都会生成一个代表该设备的 struct usb_device 对象, 拔出时会销毁该对象. 为了避免每次数据 I/O 通信都在访问控制模块中查找访问权限而带来的开销, 我们在数据结构 struct usb_device 中增加设备的访问权限字段. 由于代表 USB 设备的 usb_device 对象会贯穿该设备所有的操作, 所以当设备需要判断其安全访问权限时, 只要从该设备对象中直接读取即可.

3.3 I/O 拦截模块

我们根据 BitVisor 的特点来分别对设备的不可识别、只读、只写、读写四种访问权限做相应的拦截操作.

3.3.1 设备不可识别

若从服务器端收到设备的访问权限是设备不可识别时, 根据 USB 协议当设备插入系统时, 系统会对设备进行配置操作, 我们会在 Bitvisor 中为该设备注册一个 HOOK 函数, 当系统对该设备进行配置时, 该 HOOK 函数会被调用, 并在 HOOK 函数中返回给 OS 配置错误的信息, 此时 OS 对设备的配置失败, 设备不被操作系统识别. 当从服务器端收到的访问权限是其他访问权限时, 则在 BitVisor 中不做任何拦截操作, USB 存储设备进行正常的配置操作, OS 层对设备正常识别.

判断 USB 存储设备是否可识别的操作流程图如图 3 所示.

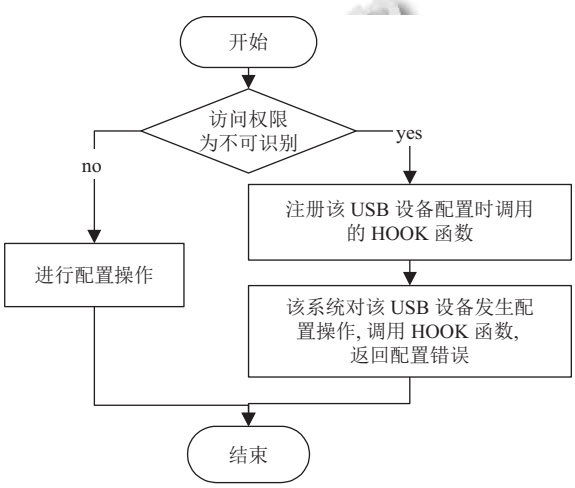

图 3 USB 存储设备识别操作控制流程图

3.3.2 设备只读权限实现

当从服务器端收到的设备访问权限是只读访问权限时, 我们根据 BitVisor 中实现的 USB host controller 的特点来进行写操作 I/O 的拦截.

在 BitVisor 中的 USB host controller 中模仿了 USB

BULK-ONLY 协议[12]. 在 BULK-ONLY 传输协议下有三种类型的数据可以在 OS 和设备之间传送: CBW、CSW 和普通数据包.

CBW 是命令块包, 该命令块包是 USB host 向设备发送的命令, 其中主要包括该 CBW 的命令块标识 dCBWTag、此次传输的数据长度 dCBWDataTransferLength、传输的具体命令 CBWCB、数据传输方向 bmCBWFlags 等信息. CBW 的命令块标识 dCBWTag 主要是用来关联此次传输对应的 CSW 的, 当 USB 设备收到 CBW 后, 对 CBW 进行解析, 然后执行相应的命令, 进入数据传输阶段, 传输完成后, 将此次命令执行的状态封装到 CSW 中返回给主机. CSW 命令状态包中包含与相对应的 CBW 中 dCBWTag 相一致的 dCSWTag. Host 根据 CSW 来决定是否继续发送下一个 CBW 或是数据.

BULK-ONLY 传输协议的数据流程图如图 4 所示.

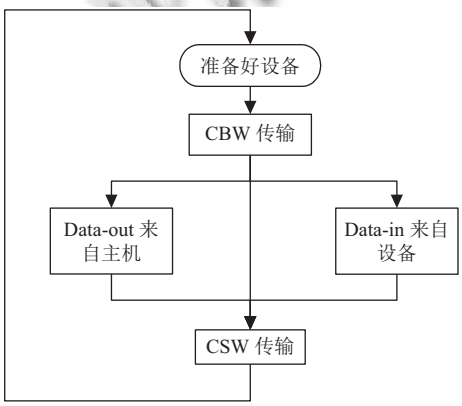

图 4 BULK-ONLY 数据流程[12]

BitVisor 作为 OS 与设备传输之间的桥梁. 模拟 BULK-ONLY 协议传输的流程, 将从 OS 层复制的 CBW 与要写的数据发送给设备, 接收 CSW 以及要读的数据, 并将其复制给 OS. 在 BitVisor 中, 发送数据后, 它会根据自身内部实现的传输描述符 qtds 中的传输状态标志判断发送的数据是否完成, 若没有完成则 BitVisor 不会接收设备发送的 CSW 状态包, 而是会一直阻塞到数据传输状态.

为了完成与设备之间的数据传输, BitVisor 拥有自己缓冲区列表, 被称为 shadow buffer, 它属于 BitVisor 自身的内存区域, 在 guest OS 中与它对应的是 guest buffer. BitVisor 中实现了 guest buffer 与 shadow buffer 中的数据交换. 而与设备直接进行数据传输的是 shadow buffer, 当有数据从 USB 存储设备到主机时, 数据首先从设备传输到 BitVisor 的 shadow buffer 中, 然后再从 shadow buffer 中拷贝到 OS 的 guest buffer 中, 此时设备中的数据才会真正到 OS 层供用户使用. 当有数据从主机到 USB 存储设备时, 数据首先从 guest buffer 中拷贝到 shadow buffer 中, 然后再由 BitVisor 控制 shadow buffer 中的数据传输到设备中.

在 BitVisor 中, 已经注册了 USB 写操作 BULK OUT 所调用的 HOOK 函数, 我们在该函数中进行写操作 I/O 的拦截. 首先在该函数中得到 CBW 命令块, 解析该命令块, 根据命令块中数据的传输方向来判断数据是否是主机发送至设备的, 若是则将其传输长度修改为 0, 并记录下命令块中 SCSI 命令, 以及该命令块的标识. 当 CBW 传输完成后, 开始传输数据, 由于设备接收到的 CBW 传输长度是 0, 不会接收 BitVisor 的数据, 所以要阻止 OS 层数据被复制到 BitVisor. 通过上述记录下的 SCSI 命令来进行判断, 当 SCSI 命令为 0x2a 或者 0xaa 时, 则阻止 guest buffer 中的数据被复制到 BitVisor 中 shadow buffer, 从而阻止了数据到 Bitvisor. 然后为了让 BitVisor 能够接收设备发来的 CSW 传输状态包, 完成一次完整的 I/O 传输过程, 还要将 Bitvisor 中此次数据传输的传输描述符 qtds 中状态标识设为传输完成. 当 BitVisor 检查数据传输状态时, 数据传输完成, 则接收从设备返回给 CSW 命令状态块, 我们根据 CSW 中的标识来判断是否是 CBW 命令包对应的 CSW, 若是则将该 CSW 的状态修改为 1, 代表此次读写操作传输错误, 返回给 OS.

在 BitVisor 中禁止写操作的流程图如图 5 所示.

上述操作使得 BitVisor 层认为此次传输已经完成, 进行 BitVisor 层传输所消耗内存的回收操作, 但是 OS 层则被告知此次传输发生错误. OS 层则会选择重传, 这样又会陷入 BitVisor 中, 循环进行上述的操作, 当 OS 层经过多次重传之后, 则放弃了此次数据的传输. 对 USB 设备的写操作失败.

3.3.3 设备只写权限实现

在 BitVisor 中, 已经注册了 USB 读操作 BULK IN 所调用的 HOOK 函数, 我们在该函数中进行读操作 I/O 的拦截. 与拦截写操作的原理相同, 首先解析 CBW 命令块, 根据命令块中数据的传输方向来判断数据是否是来自设备, 若是则将其传输长度修改为 0, 设备不会发送数据至 OS, 故要阻止 BitVisor 数据复制到 OS, 根据 SCSI 命令, 当 SCSI 命令为 0x28 或者 0xa8 时阻止 BitVisor 中 shadow buffer 数据复制到 guest

buffer 中, 从而完成阻止数据到 OS 内存的操作. 然后将 qtds 中的状态设置为传输完成, BitVisor 接收 CSW 传输状态包, 修改其中的传输状态为传输错误, 然后返回给 OS. 在 BitVisor 中禁止读操作的流程图如图 6 所示.

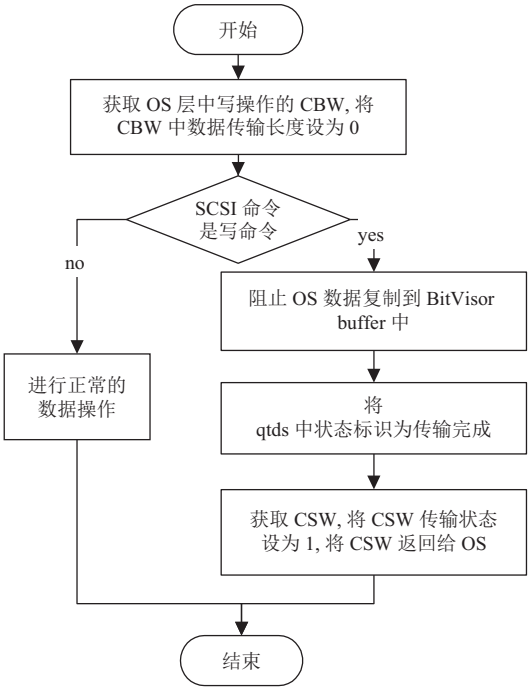

图 5 拦截写操作数据流程图

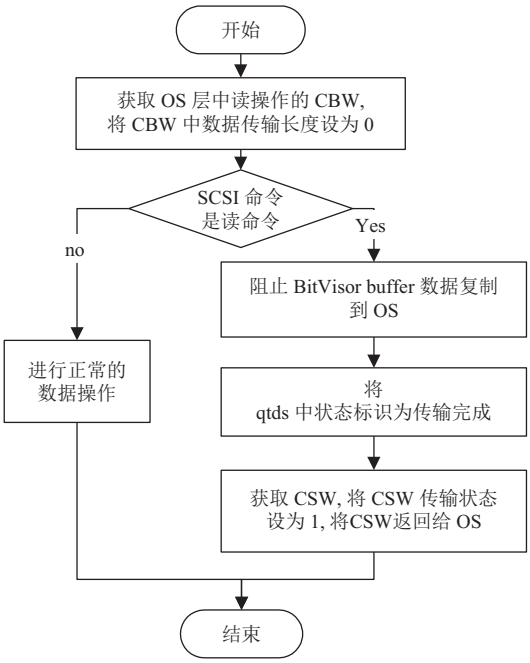

图 6 拦截读操作数据流程图

3.3.4 设备读写权限实现

若从服务器端收到的访问权限是读写访问权限时, 则在 BitVisor 中不做任何拦截操作. USB 存储设备正常识别, 以及进行正常的读写操作.

4 系统分析

我们主要从系统的安全性角度以及系统性能的角度进行了分析.

4.1 安全性分析

该系统主要在 BitVisor 中实现, BitVisor 本身可信计算基小, 安全性高. 而系统中的各个模块, 不论是与服务器进行网络通信的客户端模块, 还是系统中 USB 设备数据安全访问模块和 I/O 拦截模块均对操作系统完全透明, 这样使得该系统的安全性不依赖于操作系统的安全性, 即使操作系统中存在恶意代码的攻击, 该系统的各个模块也不会被旁路. 并且由于 BitVisor 支持主流的操作系统, 所以该系统对操作系统的类型没有任何要求. 综上所述该系统相比于目前存在的系统, 该系统安全性更高, 可应用性更好.

4.2 性能分析

针对该系统对 OS 与设备之间的数据传输速度的影响做了测试, 由于系统中读操作与写操作实现原理相同, 以下测试只给出了对写数据速率的影响.

4.2.1 测试环境

硬件条件: CPU QuadCore 四核; 内存 2 G; 硬盘 120 G; USB 存储设备 (U 盘、usb2.0 接口).

软件条件: 操作系统版本: Ubuntu 16.04; Bitvisor 版本: BitVisor tip, 且该版本经过兆芯公司的修改: 利用一个处理器一直监控 USB 设备线程.

4.2.2 测试结果

该实验主要在不安装 BitVisor 的系统环境、安装 BitVisor 的系统环境, 以及安装实现了该系统的 BitVisor 三种实验环境下做了对比实验, 实验中对 U 盘进行写操作的文件大小分别为 10.5 M、27.8 M、61.4 M 三种文件大小. 实验结果如表 1 所示.

表 1 系统性能测试

文件大小 (M)	平台		
	不安装 BitVisor (s)	安装 BitVisor (s)	安装该系统的 BitVisor (s)
10.5	6.57	7.47	7.54
27.8	43.18	47.12	47.67
61.4	80.47	87.79	88.39

从实验结果看出, BitVisor 对操作系统 USB 设备的写操作造成了一定的影响, 但是在可接受的范围之内, 而在 BitVisor 中实现的该系统中的 I/O 拦截等操作, 对性能的影响很小.

5 结语

本文实现了一个基于 BitVisor 的 USB 存储设备安全访问控制系统. 该系统主要利用 BitVisor 的特点, 使得该系统的安全性不依赖于操作系统的安全性, 达到更加安全的目的. 并且性能方面, 经过测试得出该系统对 USB 设备数据传输造成了一定的影响, 但是在可接受的范围内. 下一步的研究工作是在系统中加入安全访问策略, 使得该系统应用性更强.

参考文献

1 Tetmeyer A, Saiedian H. Security threats and mitigating risk for USB devices. IEEE Technology and Society Magazine, 2010, 29(4): 44–49. [doi: 10.1109/MTS.2010.939228]

2 吕志强, 刘喆, 常子敬, 等. 恶意 USB 设备攻击与防护技术研究. 信息安全研究, 2016, 2(2): 150–158.

3 秦春芳. 移动存储介质安全管理技术研究[硕士学位论文]. 南京: 南京师范大学, 2013.

4 张慧敏. USB 存储设备安全机制的研究与实现[硕士学位论文]. 成都: 电子科技大学, 2016.

5 刘蕊红, 蔡皖东, 张赟. 面向 Linux 的 USB 设备监控技术研究与实现. 微电子学与计算机, 2007, 24(5): 149–152.

6 龚演, 吴庆波, 谭郁松, 等. 基于 Linux 的 USB 存储设备访问控制机制研究. 计算机技术与发展, 2012, 22(3): 1–5.

7 龚演. 基于 LSM 框架的 USB 存储设备数据泄漏防护研究[硕士学位论文]. 长沙: 国防科学技术大学, 2011.

8 张军伟, 罗红, 乔向东. 基于文件过滤的移动存储设备实时监控系统设计与实现. 通信技术, 2009, 42(2): 283–285.

9 李永强, 谭立清, 马同茂, 等. USB 移动存储设备密级保护系统的设计与实现. 计算机光盘软件与应用, 2014, (13): 89–91.

10 Shinagawa T, Eiraku H, Tanimoto K, *et al*. BitVisor: A thin hypervisor for enforcing I/O device security. Proc. of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. Washington, DC, USA. 2009. 121–130.

11 Verma S, Singh A. Data theft prevention & endpoint protection from unauthorized USB devices—implementation. Proc. of 2012 the 4th International Conference on Advanced Computing. Chennai, India. 2012. 1–4.

12 孙庚, 解晓茹. Bulk-Only 协议及其实现. 福建电脑, 2004, (1): 5–7.