基于Linux的USB存储设备访问控制机制研究

龚 演,吴庆波,谭郁松,汪 黎,彭 勇
(国防科学技术大学,湖南 长沙 410073)

摘 要:USB存储设备所造成的数据泄漏问题日益严重。对USB存储设备进行访问控制,可以有效阻止USB存储设备的数据泄漏。文中主要研究基于Linux操作系统的USB存储设备访问控制机制,并且从用户态、内核Lsm框架以及驱动这三个不同层次分别提出了三种可行的USB存储设备访问控制机制。在这基础上,结合这三种访问控制机制各自的特点以及关键技术对它们的有效性、可用性进行了分析。作为验证,在Linux平台上实现了这三种机制。三种方法均可以有效地进行USB存储设备访问控制。

关键词:USB存储设备;访问控制;驱动;LSM框架
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2012)03-0001-05

Research on Access Control Mechanism of USB Storage Devices Based on Linux

GONG Yan, WU Qing-bo, TAN Yu-song, WANG Li, PENG Yong
(National University of Defense Technology, Changsha 410073, China)

Abstract: USB storage devices lead to data leakage easily. USB storage device access control technology can effectively prevent data leakage by USB storage device. Study the USB storage device access control technology based on the Linux operating system, and propose three possible access control mechanism of USB storage devices from the three levels which are user space, kernel and driver respectively. On this basis, analyze effectiveness and availability according to their characteristics and key technologies. As the demonstration, have implemented three methods based on Linux kernel. All methods can be effective access control for USB storage devices.

Key words: USB storage device; access control; driver; LSM framework

0 引 言

现代社会突飞猛进地迈向信息电子化时代。随着信息化的发展,人们需要进行大量的数据存储和交换。USB存储设备以其接口体积小、支持热插拔等优势,广泛适用于各种场合。在计算机安全领域,计算机的易用性与安全性是相对矛盾的。易用性高容易带来一些安全隐患。人们在享受USB存储设备易用性的同时,也面临着它们带来的一系列数据泄漏问题。调查机构Forrester Research指出,有52%的机密资料外泄事件与USB存储设备有关,高居所有机密数据外泄肇因的第一位。而赛门铁克也曾在全球网络安全威胁报告中指出移动U盘等含有储存空间的设备是造成数据外泄的重要原因[1]。由此可见,对于USB存储设备进行访问控制是十分必要的,尤其对于一些国家重要机构和保密单位来说。Linux是一种Unix-like操作系统,具有开放源码、稳定可靠、安全性好等优势,在一些安全部门得到了广泛应用。

文中研究基于Linux操作系统的USB存储设备的访问控制机制,从不同层次提出了3种可行的USB存储设备访问控制方法。并且,我们分别对这三种访问控制机制进行了分析、实践。最后,根据实验结果对三种机制进行比较。

1 基于udev USB存储设备访问控制机制

1.1 udev简介

udev是Linux Kernel2.6系列的设备管理器。它以守护进程形式运行在用户空间,通过监听内核发出的uevent事件来管理/dev目录下的设备文件。

udev添加设备的过程如下[2]:

1)在内核启动过程中,总线驱动程序会用总线协

收稿日期:2011-08-02;**修回日期**:2011-11-10
基金项目:国家核高基专项(2011ZX01040-001)
作者简介:龚 演(1986-),女,硕士研究生,研究方向为数据泄漏防护;吴庆波,博士,研究员,研究方向为操作系统。

议进行总线枚举,并且为每一个设备建立一个设备对象。每一个总线对象有一个 kset 对象,每一个设备对象嵌入了一个 kobject 对象,kobject 连接在 kset 对象上,这样总线和总线之间,总线和设备之间就组织成一颗树状结构。当总线驱动程序扫描到的设备建立设备对象时,会初始化 kobject 对象,并把它连接到设备树中,同时会调用 kobject_uevent( ) 函数把这个(添加新设备的)事件,以及从 sysfs 中获取的相关信息(包括设备的 VendorID, DeviceID 等)通过 netlink 发送到用户态中。

2)在用户态的 udevd 通过标准的 socket 机制,创建 socket 连接来获取内核广播的 uevent 事件,并解析这个 uevent 事件。

3)udevd 守护进程读取/etc/udev. conf 中设置的 rules. d 规则文件存放路径。

4)udevd 守护进程从指定路径中读取规则文件(默认为/etc/udev/rules. d)至内存中。udevd 逐个检查该目录下的文件,这个目录下的文件都是针对某类或者某个设备应该实施扫描措施的规则文件。

5)udev 按照文件名的 ASCll 字母顺序来读取,当 udev 找到与新设备匹配的规则,则按照规则定义的措施在/dev 目录下创建设备文件、加载驱动模块以及完成其它的附加操作。

1.2 基于 udev 的 USB 存储设备访问控制设计方案

udev 添加、删除设备的行为主要受到/etc/udev/rules. d 中的规则文件控制。可以在该目录下添加一条规则,对所有插入计算机的设备都进行是否为 USB 存储设备的匹配操作,一旦匹配成功,则忽略这个设备插入事件,拒绝为这个设备插入事件做任何处理,从而实现 USB 存储设备不可用的目标。工作过程如图 1 所示。

在/etc/udev/rules. d 下创建名为 1. rules 的规则文件,并在文件中写入如下规则:

ACTION = = " add", DRIVER = = " usb_storage", SUBSYSTEM = = "usb", OPTIONS = "ignore-device"

这条规则表示当 udevd 收到设备插入事件的 uevent 时,会匹配这个设备的驱动是否为 USB_storage,设备所属子系统是否为 usb,通过匹配这两个 USB 存储设备的特征信息来确定当前插入系统的设备是否为 USB 存储设备,一旦匹配成功则表示当前插入系统的设备为 USB 存储设备,则执行忽略这个 uevent 事件的操作,不为设备插入事件做任何处理。

通过配置这条规则,成功地实现了 USB 存储设备的禁用。这种方法的优势在于实现简单,不需要任何代码实现,只需利用 udev 提供的现有工具,通过配置规则文件,即可实现对 USB 存储设备的访问控制。这种方式的缺点主要表现在两个方面:一方面,访问控制粒度太大,无法实现 USB 存储设备的读、写控制。另一方面,由于访问控制是在用户层实现的,容易被旁路。

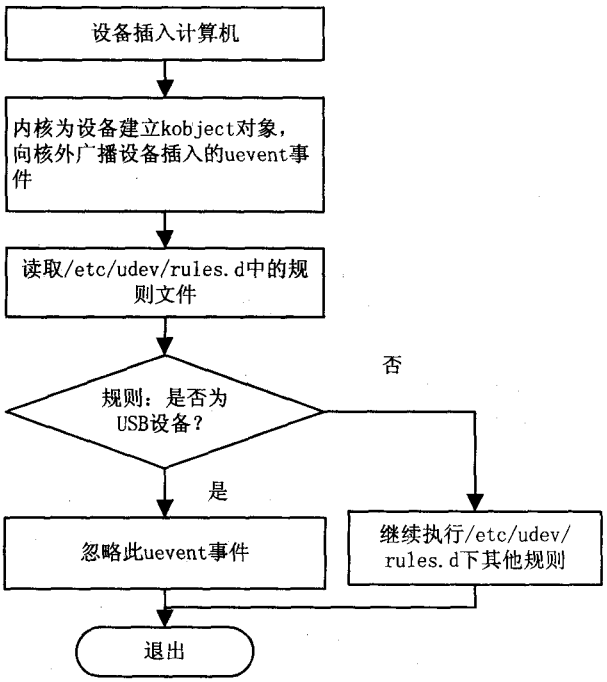

图 1 基于 udev 的 USB 存储设备访问控制

由于驱动程序是以内核模块形式实现的,比用户空间守护进程要安全。于是提出了基于驱动的 USB 存储设备访问控制机制,来改善易被旁路、以及访问控制粒度大的问题。

2 基于驱动的 USB 存储设备访问控制机制

2.1 Linux 的 USB 设备驱动加载过程简介

USB 设备都是通过 HUB 连入计算机的,当一个 USB 设备插入 HUB 时,该 HUB 的状态就会改变。系统接收到这个变化后,会为这个 USB 设备创建一个设备对象,然后将这个 USB 设备添加到 USB 子系统以及相应的 bus 设备列表中;接着,会调用 bus_attach_device( ) 函数匹配设备对应的驱动程序。匹配成功后,系统将为这个 USB 设备向总线注册,并加载相应驱动[3]。

2.2 基于驱动的 USB 存储设备访问控制设计方案

USB 存储设备在匹配驱动程序的过程中,会调用 probe( ) 函数判断插入设备与驱动是否匹配。如果 probe( ) 匹配成功则返回 1,匹配失败则返回 0。所以,我们对 USB 存储设备做访问控制时,可以在 probe( ) 函数中添加 USB 存储设备地判断。如果 probe( ) 探测到当前连入 HUB 的是 USB 存储设备,则返回 0。这样使 USB 存储设备无法匹配到合适的驱动,从而到达 USB 存储设备禁用的目标。整个实现过程如图 2 所

示。

在 Linux 内核源码目录/driver/usb/usb. c 中的 probe( )函数中添加 USB 存储设备的判断语句。一旦判断结果为 USB 存储设备,则返回 0,中断所有后续操作。在实验中测试插入 U 盘,系统能成功发现 USB 设备的插入,然后进行设备禁用,并提示用户设备插入事件。这种方法是在基于驱动层实现的,不容易被旁路。但是这种方式实现的 USB 存储设备访问控制灵活度较小,且驱动程序开发比较复杂。

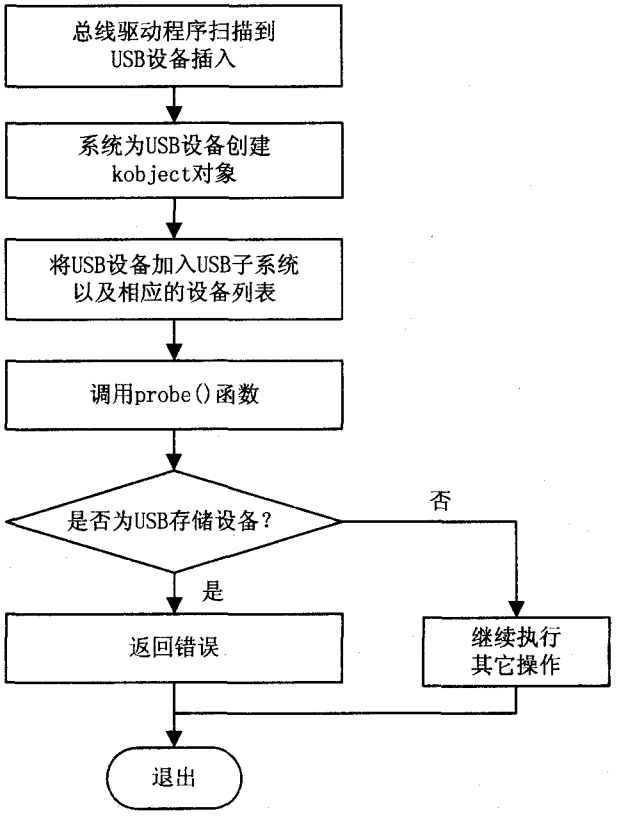

图 2　基于驱动的 USB 存储设备访问控制

Linux 在内核增加了一个通用的安全访问控制框架,它通过 hook 函数来控制程序执行,在 LSM 框架上可以实现各种各样的安全机制。所以,为了改善基于驱动的 USB 存储设备访问控制机制灵活度小、实现复杂的问题,又提出了基于 LSM 框架的 USB 存储设备访问控制机制。

3　基于 LSM 框架的 USB 存储设备访问控制机制

3.1　LSM(Linux Security Modules)框架简介

LSM 是 Linux 内核的轻量级通用访问框架[4]。它使得各种不同的安全访问控制模型能够以 Linux 可加载内核模块的形式实现。用户按照不同的需求加载所需的模块,大大提高了 Linux 安全访问控制机制的灵活性与易用性[5]。目前,已经有 POSIX. 1e capabilities、SELinux、DTE、LIDS 等很多增强访问控制系统移植到 LSM 实现[6]。LSM 的设计思想是在内核对象的数据结构中放置透明安全域作为其安全属性,并在内核源代码中放置 hook,由 hook 函数来仲裁对 task、inode 等内核对象的访问[7]。其基本原理如图 3 所示。用户执行系统调用时,先定位所访问的资源,进行错误检查,然后通过传统的 DAC 检查以后,在访问内核对象之前,用 LSM 函数调用相应的访问控制模块来仲裁访问的合法性[8]。

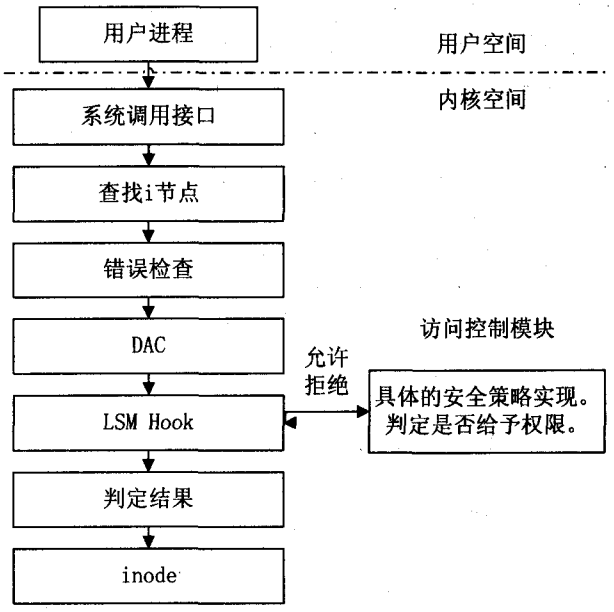

图 3　LSM 基本原理

3.2　设计与实现

3.2.1　设计目标

USB 存储设备访问控制模块向用户提供了两种控制模式:USB 存储设备挂载控制、USB 存储设备的读/写控制。通过这些方式适应不同场景的不同安全需求,最大限度地满足安全性与易用性的双重需求。

3.2.2　设计方案

USB 存储设备在插入计算机以后,总线驱动程序扫描到设备以后,向核外发送设备插入事件。udevd 用户态程序接收事件,为其加载驱动程序,创建/dev 下设备文件。用户可以通过 mount( )系统调用将/dev 下的 USB 存储设备文件挂载到任意文件夹下,使设备成为可用状态。

VFS(Virtual File System)是一个通用的文件模型,为各种文件系统提供一个通用的接口。用户态的程序通过 VFS 与文件系统进行交互。它可以用来处理与 Unix 标准文件系统相关所有的系统调用。VFS 通用模型由 superblock 对象、inode 对象等对象类型组成。其中 superblock 对象用来描述文件系统相关信息,inode 对象用来描述具体的文件信息。用户对文件的任何系统调用都会由 VFS 转换为对如 Ext2、MS-DOS 等具体

文件系统的一个调用。

在文件系统执行 mount( )操作的时候,文件系统会自动向内核注册,并将超级块信息复制至内存中,与其它文件系统超级块一起以双向链表方式保存。

1)USB 存储设备挂载控制。

LSM 框架在 mount( )系统调用中插入了 sb_check_sb( )这个 hook 函数[9]。当系统进入 mount( )系统调用的时候,会调用 sb_check_sb( )执行检查[10]:读取内存中的文件系统超级块信息中的 s_dev 字段,根据字段信息判断是否文件系统所在设备为 USB 存储设备。如果判断结果是 USB 存储设备,再检查是否允许挂载 USB 存储设备。若不允许挂载,则退出系统调用,拒绝挂载设备。若允许挂载,则将内存中被挂载文件系统的 super_block 复制至 sb_usb_list 的链表中,然后在完成系统调用后退出。

整个设备挂载控制过程如图4所示。

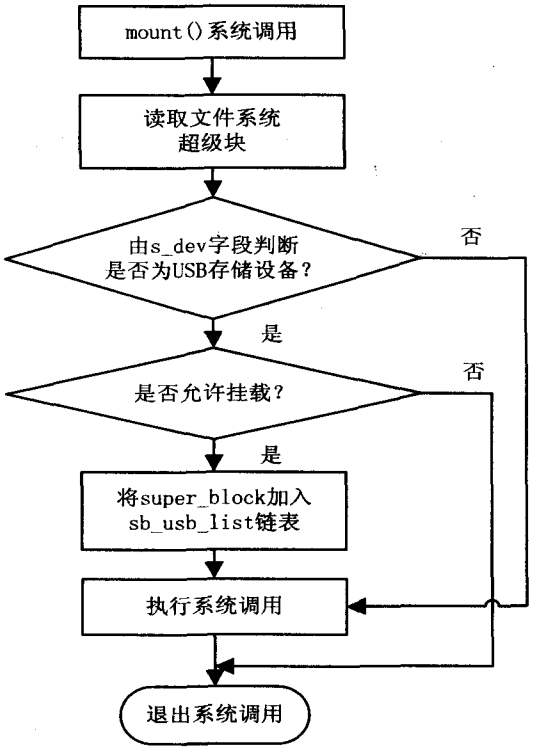

图4　USB 存储设备挂载控制

2)USB 存储设备读、写控制。

在 USB 存储设备挂载成功以后,同样也可以利用 LSM 提供的 inode_permission( )这个 hook 函数对其进行读、写控制[11]。LSM 在 open( )系统调用中提供了这个 hook 函数。在 USB 存储设备执行 mount( )系统调用过程中,已经将其 super_block 复制至 sb_usb_list 中。

在操作系统每次进入 open( )系统调用时,都可以通过 inode_permission( )函数执行检查[12]:如果系统以读方式打开文件时,首先读取文件的 super_block,然后遍历 sb_usb_list 链表,对超级块进行匹配。如果匹配成功,则说明当前读取的文件是 USB 存储设备上的文件,那么继续检查系统是否允许读操作。若允许读操作则继续执行系统调用,否则不执行任何操作,直接退出系统调用。

整个过程如图5所示。

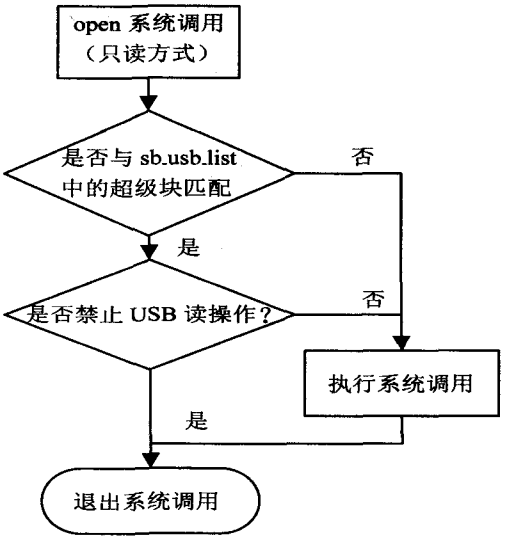

图5　USB 存储设备读控制

USB 存储设备的写控制同样也可以按照上述策略实现。在进入 open( )系统调用时判别是否是以写的方式打开的。同样,将文件的超级块与 sb_usb_list 进行匹配,匹配到一致以后,再执行是否禁用 USB 存储设备地判断。通过两次判断操作最后决定是否执行系统调用。

4　测　试

对基于 LSM 框架的 USB 存储设备访问控制进行了功能测试以及性能测试,其测试结果如下。

4.1　测试环境

硬件条件:

CPU Intel 奔腾双核 T2390;

内存 2G;

硬盘 250G;

USB 存储设备(U 盘、USB 接口移动硬盘)。

软件条件:

Linux kernel 2.6.18。

4.2　测试结果

从基于 LSM 框架的 USB 存储设备访问控制系统的功能以及性能两方面进行了测试,其测试结果如下。

首先,进行了 USB 存储设备访问控制系统的功能性测试。对 U 盘、USB 接口移动硬盘以及 USB 存储设备进行了测试,可以正确区分 USB 存储设备类型、禁止 USB 存储设备使用、控制 USB 存储设备读/写。

然后,对USB存储设备访问控制系统进行了性能测试。设计的安全策略都是在open()系统调用的时候,利用hook函数进行访问控制判断。所以安全模块对系统的整个性能的影响主要反应在执行open()系统调用时。

所以,编写了一个测试程序来计算对重复打开、关闭小文件100000次的时间开销。通过对比访问控制模块加载前、访问控制模块加载后的时间开销来反应安全模块的系统开销。

重复进行了三组对比测试。测试结果如图6所示。其中纵轴表示的是时间开销。从图中可以看出,在三组测验中,加载模块前后的打开关闭文件100000次的时间差很小。

系统测试表明,基于LSM框架的USB存储设备访问控制机制能对USB存储设备有效地进行细粒度的访问控制,并且不影响文件系统性能,对系统性能影响也很小。

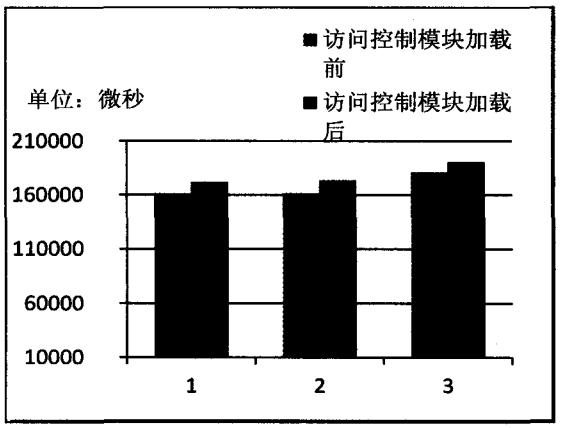

图6　测试结果柱状图

5　结束语

文中分别从用户态、内核LSM框架以及驱动这三个不同层次分别提出了三种可行的USB存储设备访问控制方法。

第一种方法是基于udev实现的,其实现简单,但是访问控制粒度太大,无法实现基于读、写的访问控制。而且由于udev是在用户层实现的,所以这种方法也比较容易被旁路。

第二种方法是基于USB驱动的访问控制机制,这种方法有效利用了probe()探测函数控制USB驱动程序的加载。这种方法虽然比较不容易被旁路,但是实现比较复杂,且同样无法实现读、写等细粒度的访问控制。

针对前两种机制的缺陷,又提出了基于LSM访问控制框架的USB存储设备访问控制机制。USB访问控制以加载内核模块的形式实现。相比前两种USB存储设备访问控制机制,方法有实现简单,使用灵活等特点。实际使用同样也证明,这种方法可以更加可靠有效的管理主机的USB存储设备。

文中提出的基于LSM框架的USB存储设备访问控制技术不仅对USB存储设备行为可以进行有效监控,还同样适用于其它并口设备、光驱类设备,只要选择合适的LSM框架提供的hook函数即可,具有普遍的指导意义。

参考文献:

[1] 计世网Ice.妥善管理USB防止数据泄漏[EB/OL].2011-07-01.http://soft.ccw.com.cn/news/htm2008/20080531_437321.shtml.

[2] Drake D.Writing udev rules[EB/OL].2011-07.http://reactivated.net/writing_udev_rules.html.

[3] 肖林甫,肖季东,任桥伟.Linux那些事儿之我是USB[M].北京:电子工业出版社,2010.

[4] Wright C,Cowan C,Morris J,et al.Linux Security Modules:General Security Support for the Linux Kernel[EB/OL].2001-09[2011-07].http://lsm.immunix.org.

[5] Bovet D P,Cesati M.Understanding the Linux Kernel[M].南京:东南大学出版社,2006.

[6] 刘　岩,王　箭.LSM实现机制的研究[C]//2009通信理论与技术新发展:第十四届全国青年通信学术会议论文集.出版地不详:出版者不详,2009:228-232.

[7] 马桂媛,何大可.通用访问控制框架LSM的研究[J].微机发展(现名:计算机技术与发展),2004,14(7):73-75.

[8] Wright C,Cowan C,Smalley S,et al.Linux Security Module Framework[C]//2002 Ottawa Linux Symposium.Ottawa:[s.n.],2002.

[9] Smalley S,Vance C,Salamon W.Implementing SELinux as a Linux Security Module[R].[s.l.]:NAI Labs,2002.

[10] 赵　亮.Linux安全模块(LSM)简介[M].[s.l.]:IBM Developer Works,2003.

[11] Smalley S,Fraser T,Vance C.Linux Security Modules:General Security Hooks for Linux[M].[s.l.]:[s.n.],2004.

[12] 刘威鹏,胡　俊,吕辉军,等.LSM框架下可执行程序的强制访问控制机制[J].计算机工程,2008(4):160-162.