第 44 卷 第 10 期 计 算 机 工 程 2018 年 10 月
Vol. 44 No. 10 Computer Engineering October 2018

·体系结构与软件技术· 文章编号：1000-3428(2018)10-0080-05 文献标志码：A 中图分类号：TP311

一种可推演的外设细粒度管控模型

金俊平，杜军龙，周剑涛

（江西省信息中心，南昌 330001）

摘 要： 针对通用 Linux 平台现有外设管控方法存在管控要素分析片面、管控粒度粗与管控方式不够灵活等问题，提出一种可推演的外设细粒度管控模型 DCM。DCM 由需求目标、受控实体、管控客体与管控策略组成，管控客体可分层嵌套复合客体，管控策略以可扩展方式提供一套涵盖驱动预判、挂载权限和用户访问的配置库。在对 DCM 组成部分进行细粒度分析的基础上，给出形式化描述和推演。在 Linux 平台上的工程化实现结果表明，该模型具有较好的推演性与实用性。

关键词： 外设管控；推演；细粒度；管控模型；形式化描述

中文引用格式： 金俊平，杜军龙，周剑涛. 一种可推演的外设细粒度管控模型[J]. 计算机工程，2018，44(10)：80-84.

英文引用格式： JIN Junping，DU Junlong，ZHOU Jiantao. A deductive peripheral device fine-grained control model[J]. Computer Engineering，2018，44(10)：80-84.

A Deductive Peripheral Device Fine-grained Control Model

JIN Junping，DU Junlong，ZHOU Jiantao

（Jiangxi Information Center，Nanchang 330001，China）

【Abstract】 For common Linux platforms，the existing peripheral device control methods have the problems that the control factor analysis is one-sided，the control grain size coarse，the control way is not flexible enough，etc. This paper proposes a deductive peripheral Device Control Model（DCM） supporting fine-grained. DCM consists of requirement goal，controlled entity，control object and control strategy. Control object can be layered nested compound object. The control policy provides a set of configuration libraries that cover driver prejudgment，mount permissions，and user access in an extensible manner. On the basis of fine-grained analysis of components of DCM，the formal description and deduction are given. The engineering implementation of the Linux platform confirms its good deduction and practicality.

【Key words】 peripheral device control；deduction；fine-grained；control model；formal description

DOI：10.19678/j.issn.1000-3428.0048927

0 概述

计算机外围设备是一柄“双刃剑”，为计算机数据交互带来极大便利的同时，也存在一定安全隐患，据中国国家计算机病毒处理中心 CVERC 在 2016 年发布的第 15 次全国网络安全状况暨计算机调查分析报告统计，目前 33.46% 的信息泄露事件是由内部人员以外设作为载体进行盗窃造成的[1]。而美国 CSI/FBI 已连续 5 年在计算机犯罪与安全调查报告中表明，超过 85% 的安全威胁来自于企业内部，而非病毒与黑客攻击。由此可见，对于计算机外围设备的管控显得尤为必要。

目前，针对通用 Linux 平台的相关外设管控研究较少，利用软/硬件方式为代表的外设端口直接封控法，以作为内部保密机使用，能够有效防止外设的非法外联，但直接封控外设端口，导致所有设备均不可用，管控代价大[2-3]。面对商业化的使用与部署，文献[2]还提供了另外 2 种外设管控方法：1）通过修改系统 kernel 源码，以监控设备驱动行为的方式对外设实施管控，该方法具有管控可靠且不易旁路的特点，但 kernel 开发复杂，且管控迁移性差；2）基于 LSM 框架中的内核 hook 技术对外设挂载进行控制，以实现外设管控，该方法管控方式有效且灵活，但以 LSM 的 hook 方法将占用内核 security_ops 全局表，导致系统中原有安全模块无效（如 selinux）[4]。文献[5-6]都提供一种先注册后管控的方法，其中：文献[5]以外设的 serial 号作为唯一标识，以白名单方式实现外设管控；文献[6]对 I/O 端口进行可信检查，在可信状态下，外设可允许接入。在针对计算机移动存储类的设备管控方面，文献[7-9]以加解密手

基金项目： 国家高技术研究发展计划“大规模空间数据融合分析关键技术与应用服务”（2014AA123001）。

作者简介： 金俊平（1967—），男，研究员，主研方向为电子政务；杜军龙，高级工程师、硕士；周剑涛，硕士。

收稿日期： 2017-10-12 **修回日期：** 2017-12-07 **E-mail：** jinjp@jiangxi.gov.cn

段分别对存储外设和用户机进行算法、级别和使用域的匹配进而实现外设管控功能,其中:文献[7]更是以射频技术实现了外设远离使用域范围时将自动报警,甚至自动销毁;文献[10]通过构造 Linux 平台中 gadget 驱动模块以实现管控存储类设备的目的;文献[11]借助 ACL 访问控制表和强制性访问控制 MAC 技术,针对外设用户访问的角色权限进行细致管控,但在区分同类不同型号设备的特殊需求中,管控粒度粗。

为满足不同应用场景下外设细粒度的管控全面分析以及管控灵活的设计需求,本文通过对当前外设管控相关技术的研究,结合管控存在的问题,构建一种实用性强且可推演的外设细粒度管控模型,并针对该模型进行形式化描述与工程化验证。

1　DCM 设计

为使外设管控能更好地满足不同应用场景下的需求,本文提出一种实用性强且可推演的外设细粒度管控模型(Device Control Model,DCM)。DCM 由需求目标(Requirement Goal,RG)层、受控实体(Controlled Entity,CE)层、管控客体(Controlled Object,CO)层与管控策略(Control Strategy,CS)层组成,如图 1 所示。

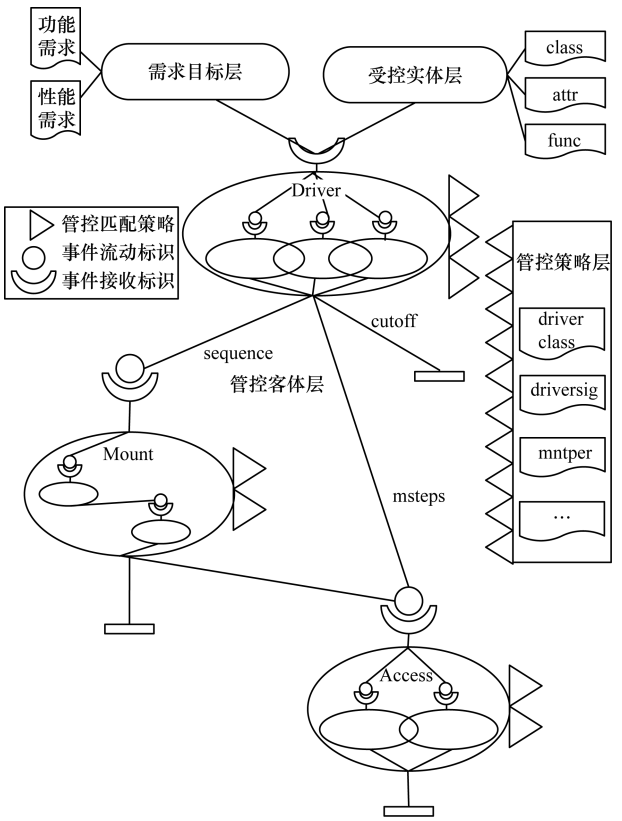

图 1　外设管控流程

1.1　需求目标层

与传统工程项目立项类似,需求目标是根据安全管理员在实际应用环境下,针对不同受控实体做出切合实际的管控要求,是项目的出发点,也是工程验收的校验点。就需求目标层(RG)中需求侧重点的不同,分为功能需求(Functional Requirement,FR)目标与性能需求(Performance Requirement,PR)目标 2 种。其中,FR 是指受控实体在特定环境下必须达到的管控力度,PR 是指满足 FR 的前提下,必须具备的系统性能(实时性、灵活性等)。

1.2　受控实体层

作为 RG 管控的对象,受控实体层(CE)具有特定功能与类型值以及若干的属性值,如图 1 所示。按功能划分 CE 有拷贝、打印、扫描、刻录和标准 I/O 等功能,按类型划分 CE 有字符设备与块设备 2 种,其中,块设备又分为存储类与非存储类。CE 的属性值包含 vid、pid、serial 和 product 等,属性值的集合可作为单个设备的唯一标识。

1.3　管控客体层

通过研究系统驱动外设原理[12-13],DCM 以装载过程层级嵌套可控点刻画了管控客体特征。该特征将管控客体层(CO)显示地区分为复合客体和粒子客体,其中,复合客体为 DCM 微层次化表达提供了切实可行的描述。管控客体 CO 中封装了驱动(driver)、挂载(mount)和访问(access)共计 3 个具有一定层级关联的复合客体。这些复合客体可层级嵌套粒子客体,粒子客体有平行和顺序 2 种结构,共同组成复合客体集。复合客体层级的连接流有顺连(aequence,层次性的复合客体管控流程互联)、跨越(msteps,层次性的复合客体跨级管控流程互联)和截止(cutoff,复合客体管控流程截止)3 种形式。

1.4　管控策略层

管控策略层(CS)作为 DCM 的核心层,延续了管控策略配置的思想,以策略库的形式存在于 DCM 中。CS 是 DCM 中管控粒度与方式的集中体现,粒度是指控制 CO 精细化和粗略化的界定,方式是控制 CO 可达效用的界定。CS 设计要素包含驱动总类、驱动单类、挂载权限、访问用户和访问用户权限等。鉴于在不同场景下 RG 和 CE 的不同,CS 要进行部分定制和扩展,且针对 CS 做出的仲裁结果具有一定的利弊性。

2　DCM 形式化描述

在 DCM 设计的基础上,以系统驱动外设和访问的顺序流程为主线,对 DCM 组成部分进行细粒度分析,并给出 DCM 形式化的描述及推演验证。

定义 1(需求目标)　设 RG 是需求目标集,则每个需求目标 rg 可定义为一个 3 元组,即 $rg = <id_{rg}, type, obj>$。其中:标识符 $id_{rg} \in ID$(ID 为标识符集合);类型 $type \in Type$,$Type = \{FR, PR\}$ 描述需求目标中功能性需求与性能性需求;宿主实体 $obj \in CE$,CE 是 DCM 中受控的实体集合。

定义 2(受控实体) 设 CE 是受控实体集,则每个受控实体 ce 可定义为一个 4 元组,即 $ce = <id_{ce}, class, attr, func>$。其中:标识符 $id_{ce} \in ID$(ID 为标识符集合;实体类型 $class \in Class$, $Class = \{chardev, blockdev_{storage,non\text{-}storage}\}$ 区分是字符设备和块设备的集合,块设备分为存储类和非存储类;受控实体 $attr \in Attr$, $Attr$ 是外设自带属性值(vid、pid、$serial$ 和 $product$ 等)的集合;$func \in Func$,为目前常见外围设备功能(拷贝、打印、扫描、刻录和标准 I/O 等)的集合。

定义 3(管控客体) 设管控客体 CO 是 DCM 中层级嵌套可控点的刻画特征集。CO 中的复合客体记作 CO_{cps},粒子客体记作 CO_{ptl}。CO 定义如下:

$$CO = CO_{cps}^{driver} \cup CO_{cps}^{mount} \cup CO_{cps}^{access}$$

且:

$$\phi = CO_{cps}^{driver} \cap CO_{cps}^{mount} \cap CO_{cps}^{access}$$

设有复合客体 CO_{cps},粒子客体 CO_{ptl} 是构成 CO_{cps} 的最小单元,且在 CO_{cps} 中有平行和顺序 2 种关系,CO_{cps} 定义如下:

$$CO_{cps}^{*} = CO_{ptl0}^{*} \cup CO_{ptl1}^{*} \cup \cdots \cup CO_{ptln}^{*}$$

且:

$$\phi = CO_{ptl0}^{*} \cap CO_{ptl1}^{*} \cap \cdots \cap CO_{ptln}^{*}$$

定义 4(连接流) 设 $cflow$ 是 2 个分属不同受控客体之间的连接流,则 $cflow$ 定义为一个三元组,即:$cflow = <co_1, co_2, sort>$。$Sort = \{sequence, cutoff, msteps\}$,且 $co_1, co_2 \in CO$, $sort \in Sort$。

定义 5(管控策略) 设 CS 是管控策略集,则 CS 可定义为一个可扩展的 7 元组集,即 $CS = <id_{cs}, driverclass, driversig, mntper, useracs, acsper, extern>$。其中:

1)标识符 $id_{cs} \in ID$(ID 为标识符集合);

2)$driverclass \in DriverClass$, $DriverClass$ 内核中的总类驱动(usb、$serio$、$scsi$ 和 $ps/2$ 等)的集合;

3)$driversig \in DriverSig$, $DriverSig$ 为内核中具体个类驱动(sd、sr、$usblp$、$usbhid$、$atkbd$ 和 $psmouse$ 等)的集合;

4)$mntper \in MntPer$, $MntPer$ 是系统挂载某种外设的 $user$、$group$ 和 $other$ 对应的 rwx 权限值集合;

5)$useracs \in UserAcs$, $UserAcs$ 是访问外设的用户 $\{root, user1, user2, \cdots, usern\}$ 的集合;

6)$acsper \in AcsPer$, $AcsPer$ 是用户外设访问的预设权限(安全、宽松和严格等)集合,其中安全对应可读写执行操作,宽松对应仅可读操作,严格对应不可读写执行操作;

7)$extern \in Extern$,表示在不同 RG、CE 和 CO 条件下自定义的扩展策略。

推演 1(可达性) 在三元组 $<CO, Total_{\{ce,cs\}}; R>$ 中,CO 称为管控客体,$Total_{\{ce,cs\}}$ 称为所有条件集,是 CE 和 CS 的集合,R 称为管控结果集,满足式(1)和式(2):

$$CO \cup Total\{ce, cs\} \neq \varphi \wedge CO \cap Total\{ce, cs\} = \varphi \tag{1}$$

$$R \subseteq RA <= CO \times Total\{ce, cs\} \cup Total\{ce, cs\} \times CO \tag{2}$$

式(1)表示 CO 和 $Total_{\{ce,cs\}}$ 是不相交的集合,式(2)表示 R 属于 CO 和 $Total_{\{ce,cs\}}$ 的笛卡尔积推导集合 RA。对于三元组 $<CO, Total_{\{ce,cs\}}; R>$,若存在 $total_{\{ce,cs\}} \in Total_{\{ce,cs\}}$,使得 $R[total_{\{ce,cs\}} => RG_x$。

则证明:RG_x 为 R 可达的。

推演 2(包含性) 有三元组 $<CO1_{ptl}, Total1_{\{ce,cs\}}, R1>$ 和 $<CO2_{ptl}, Total2_{\{ce,cs\}}, R2>$,当满足式(3)~式(5):

$$CO1_{ptl} \subseteq CO2_{ptl} \tag{3}$$

$$Total\{ce, cs\} \subseteq Total2\{ce, cs\} \tag{4}$$

$$R = R2 \cap ((CO1 \times Total\{ce, cs\}) \cup (Total\{ce, cs\} \times CO1)) \tag{5}$$

则证明:$<CO2_{ptl}, Total2_{\{ce,cs\}}, R2>$ 包含于 $<CO1_{ptl}, Total1_{\{ce,cs\}}, R1>$。

推演 3(安全性) 有三元组 $<CO, Total_{\{ce,cs\}}; R>$,存在条件集 $total0_{\{ce,cs\}}, total1_{\{ce,cs\}}, \cdots, totalk_{\{ce,cs\}}$ 使得 $Rtotal0_{\{ce,cs\}} => RG_0$, $Rtotal1_{\{ce,cs\}} => RG_1$, $Rtotal2_{\{ce,cs\}} => RG_2, \cdots, Rtotalk_{\{ce,cs\}} => RG_k$ 且 $RG_0 \cup RG_1 \cup \cdots \cup RG_k = RG$。

则证明:$<CO, Total_{\{ce,cs\}}; R>$ 具有安全性。

3 DCM 工程实现

针对项目的实际需求,将 DCM 在通用 Linux 平台上进行工程化实现。但为使开发过程更简便,实现过程并没有完全从零开始构建,而是借助了部分 udev[14] 和 cgroup[15] 技术。

需求目标集 $RG = \{RG_{id}, RG_{type}, RG_{obj}\}$,其中,$RG_{id}$ 为标识符,$RG_{obj} \in CE$, $RG_{type} = \{FR_{rglik}, FR_{rgucl}, FR_{rgcls}, PR_{rgpra}\}$,如表 1 所示。

表 1 需求目标集

标识	类型	缩略词	长距离
id1	FR	rglik	杜绝外设非法外联(驱动层)
id2	FR	rgucl	可对用户访问外设权限设置,非法用户无权访问所有外设
id3	FR	rgcls	可自动针对某类/单个外设实施自动管控,同时支持手动管控
id4	PR	rgpra	实时性、灵活性、可迁移、防旁路

受控实体 $CE = \{CE_{id}, CE_{class}, CE_{attr}, CE_{func}\}$,其中,$CE_{id}$ 为标识符集合,$CE_{class} = \{chardev, blockdev_{storage,non\text{-}storage}\}$,$CE_{attr}$ 为设备属性集合,$CE_{func} = \{Upan, cdrom, printer, keyboard, mouse\}$,分别对应通用串行总线 U 盘、刻录光驱、打印机、键盘和鼠标,其中,$\{printer, keyboard, mouse\} \in \{chardev\}$,$\{Upan, cdrom\} \in \{blockdev_{storage}\}$。

依据 DCM 和需求目标集 RG_{id1}、RG_{id2} 和 RG_{id3} 可

知,$CO = CO_{cps}^{driver} \cup CO_{cps}^{mount} \cup CO_{cps}^{access}$,且 $cflow = < CO_{cps}^{driver}, CO_{cps}^{access}, msteps >$。依据受控实体集 CE_{func} 可知,打印机、键盘和鼠标仅为系统字符设备,需进行驱动层的非法外联管控与接入后的实时管控,而 U 盘与刻录光驱为系统块存储类设备,在包含外联管控与接入管控功能的基础上,需进一步对用户访问权限进行管控。

鉴于 RG_{id4} 管控功能需具备可迁移性,传统修改 kernel 源码的驱动管控形式与需求不符,借助 udev 对外设热插拔监控技术,在用户层中对系统外设驱动实施实时管控为可行方案。同时,为体现管控的灵活性,对外设驱动进行了细粒度的划分,如图 2 所示。第 1 类驱动为通用串行总线驱动,它是内核驱动/sys/bus/pci/总线下的所有 usb 外设驱动总开关。在管控策略 CS 中对驱动管控策略 $driver_{status}$ 设定为 0、1、2 共 3 种状态,其中,0 表示驱动禁用,1 表示驱动放行,2 表示 default 状态(default 是指系统管理员没有对驱动进行状态设置,是外设管控后台程序自动获取并记录的结果)。第 2 类驱动为单类外设驱动,其中,*sd* 为 U 盘类驱动,*sr* 为刻录光驱类驱动,*usbhid* 为键盘与鼠标类驱动,*usblp* 为打印机类驱动。管控策略 CS 对第 2 类驱动管控同样设定为 0、1、2 共 3 种状态,其含义与第 1 类驱动状态相同。第 3 类驱动是单个外设驱动管控,同类不同外设以设备硬件属性值 CE_{attr} 中 *vid*、*pid*、*serial*、*product* 信息作为外设的唯一标识符,管控策略 CS 对第 3 类驱动管控同样有 3 种状态,其含义同上。

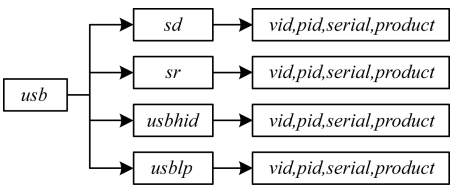

图 2　外设驱动粒度划分

FR_{rglik} 与 FR_{rgcls} 可达性推演:

在复合客体 CO_{cps}^{driver} 阶段,CO_{cps}^{driver} 包含粒子客体 CO_{ptl0}^{sd}、CO_{ptl1}^{sr}、CO_{ptl2}^{usblp} 和 CO_{ptl3}^{usbhid},且各粒子客体之间是平行结构。

在管控策略 CS 中,usb 总类驱动策略表 $CS_{driverusb} = \{usb, driver_{status}\}$,单类驱动策略表 $CS_{driversig} = \{sd, sr, usblp, usbhid, driver_{status}\}$、单个设备驱动策略表 $CS_{ctlsig} = \{vid, pid, serial, product, control_{status}\}$。

在 $< CO, Total_{\{ce,cs\}}; R >$ 三元组中,CO_{cps}^{driver} 为复合客体,$CE = \{CE_{id}, CE_{class}, CE_{attr}, CE_{func}\}$,$CS = \{id_{cs}, CS_{driverusb}, CS_{driversig}, CS_{ctlsig}\}$,且 $Total_{\{ce,cs\}}$ 为所有条件集,在满足式(1)和式(2)的前提下,存在 $CS_{driverusb}$、$CS_{driversig}$ 和 CS_{ctlsig},使得 $total_{\{ce,cs\}} \in Total_{\{ce,cs\}} => R[total_{\{ce,cs\}} => RG(FR_{rglik}) \ \& \ RG(FR_{rgcls})]$,由此证明:$FR_{rglik}$ 与 FR_{rgcls} 是可达的。

针对系统块存储类设备进行用户访问权限管控,本文借助了 cgroup 中 devices 子模块的资源管理机制,将访问用户以白名单的方式构建用户访问策略表 $CS_{useracs} = \{whitelist, CE_{func}, acsper\}$,白名单内为合法用户,同时,对合法用户访问系统块存储类设备权限设置为严格(null)、宽松(r)与安全(rwx)3 种模式。严格模式为用户合法,但无权限访问外设。宽松模式为用户合法,但对外设仅有读访问权限。安全模式为用户合法,对外设具有读写执行访问权限。

FR_{rgucl} 可达性推演:

在复合客体 CO_{cps}^{access} 阶段,CO_{cps}^{access} 包含粒子客体 CO_{ptl0}^{user} 和 $CO_{ptl1}^{permmision}$,且各粒子客体之间为顺连结构。管控策略 CS 中将访问用户以白名单方式构建用户访问策略表 $CS_{useracs} = \{whitelist, CE_{func}, acsper\}$。

在 $< CO, Total_{\{ce,cs\}}; R >$ 三元组中,CO_{cps}^{access} 为复合客体,$Total_{\{ce,cs\}}$ 为所有条件集,且满足式(1)和式(2)。存在 $CS_{useracs}$ 和 CS_{acsper},使得 $total_{\{ce,cs\}} \in Total_{\{ce,cs\}} => R[total_{\{ce,cs\}} => RG(FR_{rgucl})]$,由此证明:$FR_{rgucl}$ 是可达的。

在复合客体 CO_{cps}^{driver} 阶段,FR_{rglik} 与 FR_{rgcls} 具有可达性。在复合客体 CO_{cps}^{access} 阶段,FR_{rgucl} 具有可达性,由推演 3(安全性)可知,此外设管控系统依据 DCM 进行设计具备安全性,其实现设计的外设管控流程如图 3 所示。

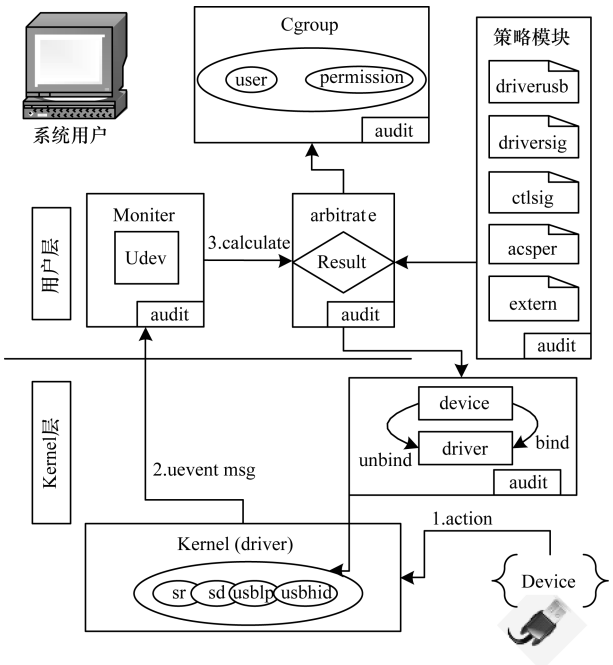

图 3　外设管控流程

外设管控后台程序借助 udev 中外设热插拔监控技术,通过 udev _ monitor _ new _ from _ netlink(udev,"udev")函数捕获内核发出 uevent 事件消

息,然后以 udev_monitor_filter_add_match_subsystem_devtype(mon,"usb","usb_device")函数对捕获的内核消息做进一步目标性信息筛选,以此达到管控后台程序对 CE_{func} 中 U 盘、刻录光驱、打印机、键盘和鼠标的插拔动作实时监控功能。

在系统/sys/bus/pci/总线下面分析捕获 CE_{func} 中设备硬件属性值 CE_{attr},以作为设备唯一的标识符集。同时,以 CE_{attr} 做参考基线,判定该设备在内核中的类驱动是否属于类驱动策略集合 $CS_{driversig}=\{sd,sr,usblp,usbhid\}$ 中的一种,如若属于,则外设属于管控范畴。

如若外设属于管控范畴,则仲裁模块对管控策略表 $CS_{driverusb}$、$CS_{driversig}$ 和 CS_{ctlsig} 进行综合仲裁,仲裁原理根据策略表中的驱动实际值进行按位与计算,以最终仲裁结果对外设与驱动进行绑定(*bind*,使外设与驱动正常绑定,允许设备合法外联,且设备可用)和解绑定(*unbind*,使外设与驱动解绑定,禁止外设合法外联,使设备不可用)操作。若不符合管控范畴,则外设管控后台程序将不做任何操作。

为防止 root 用户权限过大致使访问旁路,借助 cgroup 中的 devices 子模块资源管理机制,首先创建 cgroup 层级结构,并以 mount-t cgroup-o devices-/cgroup/devices 命令创建 devcies 子系统,此时将在/cgroup/devcies/目录下自动创建 devices.allow、devices.deny、devices.list 等伪文件和 group3-dev 文件夹,然后按照 CS 中的用户访问策略表 $CS_{useracs}$ 中用户和设备的权限关系向/cgroup/devices/group3-dev/tasks 进行用户所有进程的权限设置,一旦对某用户设备访问权限设置完成,此用户下所有进程对此设备访问均仅有一种权限。cgroup 资源机制设置对 root 用户和普通用户设置均有效,以此最终实现系统所有用户访问外设的权限约束功能。

经上述工程验证,针对通用 Linux 操作系统,依照 DCM 进行外设管控的设计与实现有以下特点:

1)粒度细:具体表现在设计过程的分析粒度与管控实施粒度;

2)可推演:具体表现在管控客体与管控策略构成条件集对需求目标的推导与验证;

3)实用性:具体表现在 DCM 对实际需求的包含性、扩展性和可实施性。

4 结束语

本文构建一种迎合实际外设管控开发的通用模型 DCM。该模型集细粒度分析与推演验证于一体,将管控要素进行分离,使得第三方可以用推演验证的形式对外设管控的实际需求进行扩展、配置和可达性分析,从而以较短的开发周期获取粒度细和实用性强的集中式外设管控系统。下一步将把时间因素 T 引入到模型中,在分析可达性的基础上对管控的效率进行分析。

参考文献

[1] CVERC. 第十五次全国信息网络安全状况暨计算机和移动终端病毒疫情调查分析报告[EB/OL]. [2017-10-12]. http://www.cverc.org.cn/head/diaocha2015/report2015.pdf.

[2] 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[J]. 计算机技术与发展,2012,22(3):1-4.

[3] DERONCELE E B,FUENTES A P,HEMANDEZ D C,et al. USB device management in GNU/Linux systems[C]//Proceedings of the 10th International Conference on Open Source Systems. Berlin, Germany: Springer,2014:218-225.

[4] 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,2008,34(7):160-162.

[5] 赵 俭. 国产平台外围设备管控系统研究与设计[J]. 网络安全技术与应用,2016,8(1):97-99.

[6] 梁志宏,梁智强,黄 曙,等. 设备可信管控方法及其系统:CN 103198037[P]. 2013-07-10.

[7] 王秋晨,王 雷,夏鲁宁,等. 基于环境感知的安全移动存储系统设计与实现[J]. 计算机工程设计,2014,35(4):1165-1171.

[8] 高出云,王正生. 移动存储设备安全管控的措施和功能[J]. 空军雷达学院学报,2010,24(1):58-60.

[9] 李爱国,冯国松. 一种安全的 USB2.0 设备控制器设计[J]. 计算机工程,2012,38(24):288-290.

[10] 熊聪聪,汪 鹏. 一种新型的 USB 存储设备访问控制方案[J]. 自动化仪表,2011,32(12):16-19.

[11] 陈松政,魏立峰. 基于用户身份标识的外设访问控制方法[J]. 计算机工程与科学,2015,37(10):1831-1835.

[12] CORBET J,RUBINI A,HARTMAN G K. Linux 设备驱动程序[M]. 魏永明,耿 岳,钟书毅,译. 3 版. 北京:中国电力出版社,2006.

[13] 王 欢,茅俊杰,王 丹,等. Linux 设备驱动重用研究[J]. 计算机科学,2017,44(4):39-42.

[14] 邱 勇,冯生强,黄 迅. 基于 UDEV 和 PAM 机制的 USB KEY 认证实现[J]. 计算机应用,2012,32(1):71-73.

[15] MENAGE P. Linux kernel documentation: cgroups.txt[EB/OL]. [2017-10-12]. https://www.mjmwired.net/kernel/Documentation/cgroups.txt.

编辑 顾逸斐