第 44 卷　第 2 期　　计 算 机 工 程　　2018 年 2 月
Vol. 44　No. 2　　Computer Engineering　　February 2018

·安全技术·　　文章编号：1000-3428(2018)02-0171-06　　文献标志码：A　　中图分类号：TP391.9

# 基于环境属性的访问控制系统设计与实现

刘晓威，周　雷，王国军

（中南大学 信息科学与工程学院，长沙 410083）

**摘　要：**目前研究的访问控制机制大多将访问策略与主客体相关联，访问权限相对固定，但在实际应用中，访问控制权限需要根据服务环境变化而实时调整。为此，构建一种基于环境属性的访问控制模型，在 Linux 系统下设计并实现基于该模型的访问控制系统。在访问控制的判定过程中增加环境属性因素，实现根据环境属性动态调整访问控制权限。测试结果表明，该模型对系统开销少，不会降低 Linux 平台的正常运行效率，能有效提升系统的安全性和可用性。

**关键词：**访问控制；环境属性；Linux 安全模块；Linux 平台；钩子函数

**中文引用格式：**刘晓威，周　雷，王国军. 基于环境属性的访问控制系统设计与实现[J]. 计算机工程，2018，44(2)：171-176.

**英文引用格式：**LIU Xiaowei，ZHOU Lei，WANG Guojun. Design and Implementation of Access Control System Based on Environmental Attribute[J]. Computer Engineering，2018，44(2)：171-176.

# Design and Implementation of Access Control System Based on Environmental Attribute

LIU Xiaowei，ZHOU Lei，WANG Guojun

（School of Information Science and Engineering，Central South University，Changsha 410083，China）

**【Abstract】** The current research of access control mechanism is clearly distinguished between the subject and object，and user's permission is relatively fixed，but in practical applications，user's permission needs to be adjusted according to the change of the service environment. This paper presents an environmental attribute-based access control model，designed and implemented a prototype system under the Linux system. The system adds environmental attribute factors into the access control decision process，adjusts user's permission dynamically. Based on small system overhead，it can enhance the system security and availability without affecting the operating efficiency of the Linux platform.

**【Key words】** access control；environmental attribute；Linux Security Module（LSM）；Linux platform；hook function

**DOI：**10.3969/j. issn. 1000-3428. 2018. 02. 030

## 0　概述

在信息安全领域的研究中，访问控制（Access Control，AC）技术作为关键技术一直备受关注[1]。在该技术方向上先后提出了自主访问控制、强制访问控制、基于角色的访问控制以及基于属性的访问控制等模型[2]。基于属性的访问控制模型（Attribute-based Access Control，ABAC）是近些年提出的一种较新的访问控制模型，其在管理访问时，按照主体、客体、操作等的属性是否匹配，来确定访问是否合法，因此，具有很强的灵活性和扩展性[3]。Linux 操作系统是当下科研和商业领域里主流的操作系统，然而其上还没有建立起成熟的基于属性的访问控制机制。当前 Linux 系统采用基于权限位的自主访问控制（UGO 权限模型）来进行基本的访问控制，此外 Linux 逐步增加了 ACL 机制、Capabilities 机制、LSM 机制等，同时也增加了 SELinux、Smack 等安全子系统[4]。然而 Linux 中这些传统的访问控制机制，基本上都指明了主体和客体，权限设置都与主客体相关联，且在访问控制过程中固定不变。但在实际情况中，系统的环境属性对访问权限也存在很大的影响，若将环境属性加入到对访问权限的判断中将能够满足更多的安全需求[5]。

本文构建一种基于环境属性的访问控制模型（Environmental Attribute Based Access Control Model，EABAC），该模型将环境属性加入到访问控制判定范围中，动态监测环境属性的变化，以此来调整访问控制权限。

**基金项目：**国家自然科学基金（61272151，61472451）。
**作者简介：**刘晓威（1992—），男，硕士，主研方向为信息安全、访问控制；周　雷，博士；王国军，教授、博士。
**收稿日期：**2017-02-12　**修回日期：**2017-03-17　**E-mail：**362414189@qq.com

## 1 相关技术原理

### 1.1 Linux 现有的访问控制机制

Linux 属于多用户的操作系统,其基本的访问控制策略是基于权限位的自主访问控制。系统对用户采用分组的方式进行管理,用户组和用户都有独立的标识,分别为 GID 和 UID[6]。Linux 系统内的用户被划分成 root、user、group 和 other 4 类,分别指代的是根用户、文件所属用户、文件同组用户和其他用户。系统中软、硬件资源都被当作文件处理,每种文件都分别被定义了读、写和执行的权限。文件还有控制权限,默认情况下,文件的创建者是文件的所有者,可自主地控制文件的读、写、执行的权限[7]。root 用户在系统中拥有最高权限,不会受到访问控制的限制。

Linux 系统中的权限控制列表(Access Control List,ACL)机制给每个文件都分配一列 ACL 条目,所有用户和用户组的权限都采用 ACL 条目表示,通过此方式可以为特定的用户和用户组设置对文件的访问权限,其扩大了权限约束的范围[8]。

Linux 系统中的 Capabilities 机制实现了更加细粒度的权限控制[9]。Capabilities 机制将 root 用户的权限划分成更加细粒度的能力,如 Linux 内核 3.13 中包含了 37 种不同的能力,此外与 ACL 机制将权限策略设置给文件不同,Capabilities 机制将权限策略设置给进程。其通过给每个进程设置能力集实现让一般用户拥有 root 用户的权限,且能实现对 root 用户的访问限制[10]。

### 1.2 LSM 安全框架

Linux 安全框架(Linux Security Module,LSM)是从内核 2.6 版本增加到 Linux 系统中的一个轻量级通用访问控制框架,其提供了一个能够成功实现访问控制模块的接口[11],其体系结构如图 1 所示。

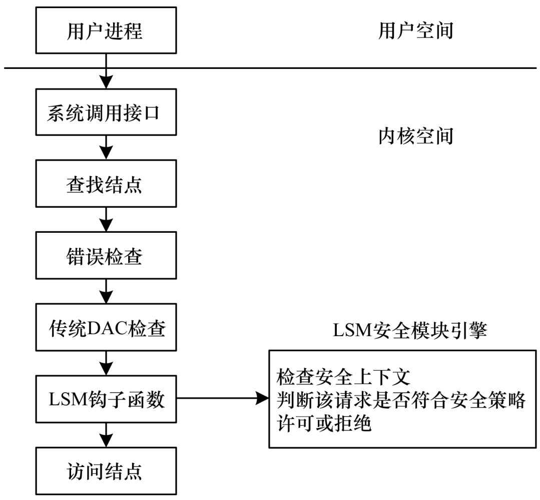


**图 1 LSM 体系结构**

LSM 主要的工作方式,是在内核资源访问的源码中,放置了钩子函数,钩子函数会在相应的内核资源被访问时被调用,用户可以自行在钩子函数中自定义判定过程,用于对访问请求进行判断。钩子函数位于内核自主访问控制之后,用户的访问请求先需经历自主访问控制的过滤,再发送到钩子函数中进行判定,所以,LSM 不会影响 Linux 系统中原有的访问控制机制[12]。利用 LSM 可以完成各种细粒度和复杂的访问控制,且有着较高的灵活性和安全性。

## 2 基于环境属性的访问控制模型 EABAC

### 2.1 环境属性的含义

本文所描述的环境属性定义:与系统安全相关的,表示用户在访问系统资源时的系统环境信息以及相关的上下文信息,例如系统的 CPU 负载、进程数、内存负载、磁盘吞吐量、系统时间等。不同的系统在不同的应用场景下被关注的环境属性都有所不同,具体在应用时可根据不同的系统特征和应用需求来进行调整。

在实际的应用场景中,若能将当前环境属性加入到访问控制权限的判定中,将会使系统的安全性、可用性及用户体验得到提高[13]。例如一个提供计算服务的服务器,如果服务器 CPU 和内存负载不高,每个用户都能快速地获取到正常的计算服务。伴随着访问量的增加或服务器上运行进程的增加,导致服务器 CPU 和内存负载过高,若用户继续尝试获取该服务,服务器的服务进程容易阻塞,并陷入长时间的等待,无法保障用户及时的获取服务。更加高效的访问控制策略是,实时监测服务器的用户访问量和 CPU、内存负载等信息的变化,当服务器负载过高时,提示用户暂时无法访问该服务,或者只允许访问资源消耗更小的低一级计算服务,而无法访问之前可访问的正常计算服务。此场景说明了在客体都未变化的情况下,系统环境属性对用户获取资源也存在很大的影响。传统的访问控制策略基本上依据的是主客体的静态安全信息,在访问开始前权限即固定,无法动态调整权限。若是能通过环境属性的变化来调整访问控制的权限,则能进一步提高系统的安全性和可用性,同时改善用户体验。

### 2.2 EABAC 的模型结构

基于访问控制权限根据动态的环境属性变化而调整的需求,本文提出基于环境属性的访问控制模型 EABAC,模型架构如图 2 所示。策略分配点(Policy Administration Point,PAP):PAP 负责对主体和客体的限制条件进行设置。有 2 个时间点可以进行设置,一是在访问开始之前,管理员根据系统状态、应用要求和安全等级,通过 PAP 为主体和客体设置限制条件。此外在系统运行过程中,管理员也可以通过 PAP 修改主体和客体的限制条件。

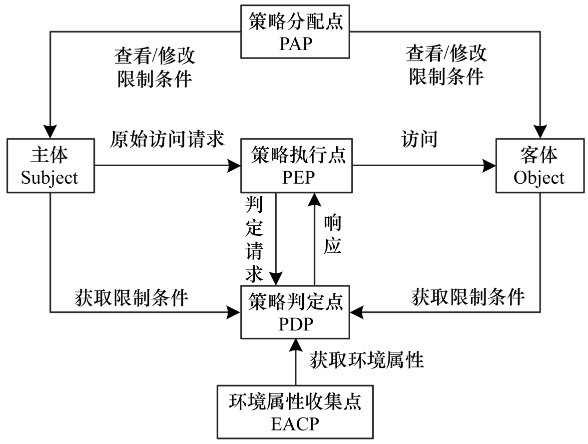


图 2　EABAC 模型结构

环境属性收集点(Environmental Attribute Collection Point,EACP):EACP 负责监控系统的运行状态,并周期性地收集各项环境属性值并传递给 PDP。

策略执行点(Policy Execution Point,PEP):PEP 负责接受主体的原始访问请求,并将访问请求传递给 PDP,最后通过接受到的返回结果,决定是否放行该访问。

策略判定点(Policy Decision Point,PDP):PDP 负责接收 PEP 传递的判定请求,并根据从主客体处获取的限制条件信息,以及 EACP 传递的环境属性值来对判定请求进行判断,并将判断结果返回给策略执行点 PEP。

在主体开始访问客体前,系统管理员通过策略分配点 PAP 对主体和客体访问控制的限制条件进行设置,限制条件作为主体和客体的一种标识属性,其标识了主体对客体的访问范围。

主体在访问客体时,会将原始访问请求传递给策略执行点 PEP,策略执行点 PEP 在接收到主体原始的访问请求后,将会向策略判定点 PDP 发送判定请求。策略判定点 PDP 根据从主体 Subject 和客体 Object 中获取的限制条件以及环境属性收集点 EACP 收集到的环境属性值对判定请求作出判断,并将响应结果返回给策略执行点 PEP,决定是否访问客体。

环境属性收集点 EACP 负责实时收集系统环境属性,并将收集到的数据传递给策略判定点 PDP。策略判定点 PDP 负责对 EACP 收集到的系统环境属性进行管理,同时从主体 Subject 和客体 Object 中获取访问控制的限制条件,并根据系统环境属性及主客体的限制条件,对访问请求进行判定。

## 3　原型系统的设计与实现

### 3.1　系统总体结构

本文所实现的原型系统包含 3 个功能模块,分别是内核判定模块、用户控制模块和通信模块。内核判定模块运行于 Linux 内核空间,负责在内核资源被访问前,根据当前系统环境的属性值,对访问行为进行判定。用户控制模块运行于 Linux 用户空间,负责实时收集系统的环境属性值并将其传递给内核判定模块,此外用户控制模块还可用来对访问控制策略进行查看和修改。通信模块主要用于内核判定模块和用户控制模块之间的数据传递。原型系统的系统结构如图 3 所示。

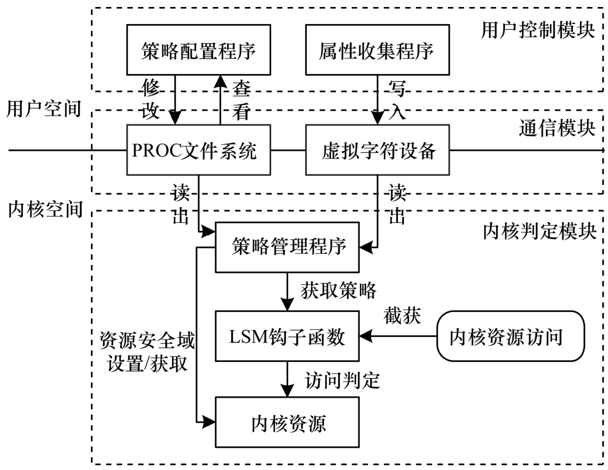


图 3　系统架构

系统运行过程中,基于 LSM 的消息截获函数将同步执行,用于拦截 Linux 内核中资源访问的进程。同时,环境属性采集程序异步搜集并处理各项系统环境属性。访问程序被截获后,当前系统环境属性被输入到预先定义的访问控制策略中进行决策分析。访问判定程序将判定当前访问操作的合法性,合法则继续执行访问操作。策略配置模块用于对文件或目录的限制条件进行设置,该操作可由系统管理员来完成。

### 3.2　内核安全模块

系统中的内核判定模块采用 LSM 框架作为基础平台实现。LSM 目前作为 Linux 内核补丁的形式实现,其提供了一个通用的基础体系给安全模块,允许安全策略以驱动模块的形式动态加载到 Linux 内核中。

LSM 钩子函数分布在 Linux 内核中各个涉及安全操作的函数中,在相关安全操作被执行的时候自动被调用。在本文系统中主要用到和文件操作相关的钩子函数,如重写了 file_permission 和 file_alloc_security 等钩子函数。

钩子函数是定义在全局表 security_ops 中的函数指针所指向的函数[14]。本文系统定义了自己的全局表 eabac_security_ops 和自己的钩子函数,用于监控系统所要监控的访问操作。安全模块可通过 register_security 函数注册到 LSM 框架中。

### 3.3 用户控制模块

#### 3.3.1 环境属性的收集

本文原型系统中涉及到的环境属性包括 CPU 负载、内存负载和访问时间。属性具体介绍见表 1。

**表 1 环境属性**

| 属性名 | 访问控制意义 |
| --- | --- |
| CPU 负载百分比/% | 在 CPU 高负载的情况下,阻止用户访问相关资源,提高系统运行效率 |
| 内存负载百分比/% | 在内存高负载的情况下,阻止用户访问相关资源,提高系统运行效率 |
| 访问时间/h | 在特定时间(如系统维护期间),阻止用户访问,改善用户体验 |

环境属性收集程序是一个守护进程,周期性地收集环境属性值,并传递给内核判定模块。

1)CPU 负载的获取

CPU 负载可以通过 proc 文件系统获取。/proc/stat 文件中记录着实时的 CPU 状态信息,根据机器的不同,可能存在多核的情况:

cpu 4447387 5090 587215 1520173257 886030 50 6610 0

cpu0 1967877 2115 202061 378871062 395505 46 3575 0

cpu1 1826065 424 228438 379268369 228746 0 970 0

……

这些数据表示从系统启动到当前时刻不同任务执行累积的时间片,其中,第一行是总的 CPU 运行时间,下面是单个 CPU 的运行时间。总的 CPU 运行时间记为 $t_{\mathrm{total}}$,可以通过将/proc/stat 文件中第 1 行各参数所表示的运行时间累加得到,总的 CPU 空闲时间记为 $t_{\mathrm{idle}}$,第 1 行中的第 4 个值即表示该时间。CPU 的使用率是一个瞬时值,可通过式(1)计算单位时间内的平均值来表示:

$$R_{\mathrm{cpu\_usage}}=((100-(t_{\mathrm{idle2}}-t_{\mathrm{idle1}}))\times 100/(t_{\mathrm{total2}}-t_{\mathrm{total1}}))\% \tag{1}$$

其中,$R_{\mathrm{cpu\_usage}}$ 表示 CPU 利用率,$t_{\mathrm{idle1}}$ 和 $t_{\mathrm{idle2}}$ 分别表示时间段首末时间点 CPU 空闲时间,$t_{\mathrm{total1}}$ 和 $t_{\mathrm{total2}}$ 分别表示时间段首末时间点 CPU 总的运行时间。

2)内存负载的获取

内存负载的获取与 CPU 负载的获取类似。/proc/meminfo 文件中记录着系统实时的内存状态信息,相比于 CPU 信息,其不存在多核的情况:

memtotal:32865856 kB

memtree:20881780 kB

……

内存利用率通过式(2)计算:

$$R_{\mathrm{mem\_usage}}=((m_{\mathrm{total}}-m_{\mathrm{free}})\times 100/m_{\mathrm{total}})\% \tag{2}$$

其中,$R_{\mathrm{mem\_usage}}$ 表示内存的使用率,$m_{\mathrm{total}}$ 表示内存总的大小,$m_{\mathrm{free}}$ 表示空闲内存的大小。

3)系统时间的获取

在 Linux 系统中,通过 time 函数可以得到自 1970 年 1 月 1 日 00:00:00(格林威治时间)以来的秒数。对于得到的时间信息,可以再通过 ctime 函数换算成日常所使用的日期时间表示格式,其返回的字符串格式形如“Wed Jun 28 21:34:08 2015”,从该字符串中提取出系统当前时间“21:34:08”即可。

#### 3.3.2 文件限制条件的配置

策略配置程序是一个用户层的控制程序,负责对文件的限制条件进行查看和设置。该用户层控制程序通过调用 inode_setxattr 钩子函数来设置文件限制条件,通过该钩子函数把环境属性值写到文件 inode 结构的 i_security 安全域中。此外,open/close 系统调用函数执行时,控制程序调用重写后的 file_alloc_security 函数,修改或销毁安全域文件信息。

### 3.4 系统内模块间通信

在 Linux 系统中,由于地址空间的限制,内核代码和用户代码间不能直接调用和访问[15],需要额外的方式实现它们之间的通信。本文原型系统实现基于 proc 文件系统和虚拟字符设备 2 种方式的系统间模块的通信方式。

定义一个虚拟字符设备作为环境属性值的传递模块,用户控制模块中的环境属性收集程序,每 2 s 获取一次环境属性值,并以结构体数据格式写入虚拟字符设备中。内核安全模块则实时地从虚拟字符设备中读出环境属性值。

内核判定模块中的全局变量,例如内核判定模块是否启用、主客体的限制条件环境属性值等需要提供查询和修改的接口。本文系统通过在/proc/sys 目录下建立目录树以显示内核中保存的全局变量,通过 register_sysctl_table 函数实现。

## 4 实验结果与分析

### 4.1 实现平台

基于环境属性的访问控制原型系统采用的实验平台参数如表 2 所示。基于普通的硬件平台,增加修改模块后的操作系统,测试访问控制权限的动态性变化情形。

**表 2 实验平台参数**

| 项目 | 具体配置 | 项目 |
| --- | --- | --- |
| 硬件平台 | Intel Mini PC | Linux 内核 |
| CPU | Intel Celeron 1037U 1.80 GHz | 编译器 |
| 内存 | 4 GB | 操作系统 |

### 4.2 测试

#### 4.2.1 功能测试

功能测试的目的是验证是否成功将 CPU 负载、内存负载以及访问时间等环境属性值添加到了访问控制的判定过程中。通过预先设置测试文件相关环

境属性的限制条件,然后访问文件,分析对应环境属性值,判断文件是否被访问。

图 4 所示为 CPU 负载的测试过程,首先,在初始状态下被测文件 hello 设置为可访问,经测试操作,文件可以进行相关访问操作。其次,执行大数据处理的计算程序,提高当前 CPU 计算负载,再次访问 hello 文件,访问被拒绝。最后,当结束高负载使用 CPU 的进程后,测试文件重新被允许访问。

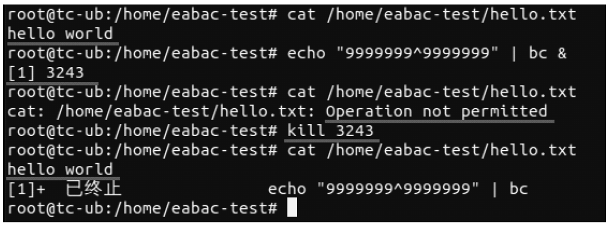


**图 4　CPU 负载测试**

图 5 所示为内存负载的测试过程,首先,在初始状态下被测文件 hello 设置为可访问,经测试操作,文件可以进行相关访问操作。其次,并行执行高内存负载的程序 upmem,该程序分配 1 GB 的内存,在休眠 5 min(模拟 upmem 执行时间)后访问 hello 文件,访问被拒绝。最后,结束 upmem 程序运行,测试文件 hello 重新被允许访问。

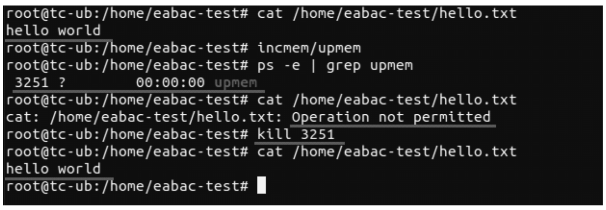


**图 5　内存负载测试**

实验结果表明,当 CPU、内存等环境属性的变化导致系统运行存在拥塞等情形中,自适应地调整文件的访问权限,能有效优化系统服务。

#### 4.2.2　性能测试

性能测试的目的是为了验证内核判定模块的添加对文件访问效率的影响程度。采用的测试方法是通过运行百万级的文件访问,记录对比在不同测试条件下访问文件的时间。测试中选取 3 种不同的运行环境以及 3 种不同的测试文件作为前提,分别如表 3 和表 4 所示。

**表 3　模块测试条件**

| 情况 | 条件 |
|---|---|
| 情况 1 | 未添加内核判定模块 |
| 情况 2 | 添加内核判定模块但未开启 |
| 情况 3 | 添加内核判定模块且开启 |

**表 4　文件测试条件**

| 文件名 | 类型 | 文件限制条件 |
|---|---|---|
| x. small | | 未添加限制条件 |
| y. small | 小文件(1 MB) | 添加了限制条件但未开启 |
| z. small | | 添加了限制条件且开启 |
| X. big | | 未添加限制条件 |
| Y. big | 大文件(50 MB) | 添加了限制条件但未开启 |
| Z. big | | 添加了限制条件且开启 |

如表 3 和表 4 所示,本文实验设置了 3 种模块测试条件,并将对小文件和大文件的文件限制也分成了不同的 3 种限制条件,目的是为了通过多种测试条件的组合,充分反映内核判定模块的添加对文件访问效率的影响。每一种测试条件下的文件操作都执行了二百万次,且在每种测试条件都执行了 3 次,最终的测试结果为 3 次的平均值。

表 5 ~ 表 7 中的测试数据记录是访问文件三百万次的时间,分别统计了在未对测试文件设置限制条件、设置了限制条件但未开启以及设置了限制条件且开启 3 种情况下,对小、大文件进行打开、读和写操作的时间。从实验数据可以看出,在添加和未添加内核判定模块的不同情况下,对文件访问操作的耗时并没有出现大的起伏,同时考虑到系统实时运行环境因素对测试的影响以及误差的存在,可以说明内核判定模块对文件访问效率影响很小。通过实验数据可以得出结论,该访问控制原型系统所执行的访问控制效率较高,不会影响到系统的正常运行,同时具备了较高的安全性和可用性。

**表 5　打开操作的访问时间**　　s

| 打开测试:小文件 | 未加载判定模块 | 加载未开启 | 加载且开启 | 打开测试:大文件/s | 未加载判定模块 | 加载未开启 | 加载且开启 |
|---|---|---|---|---|---|---|---|
| 未添加限制条件 | 13 | 14 | 14 | 未添加限制条件 | 15 | 16 | 16 |
| 设限未开启 | 13 | 14 | 15 | 设限未开启 | 15 | 15 | 17 |
| 设限且开启 | 14 | 14 | 16 | 设限且开启 | 17 | 16 | 18 |

**表 6　读操作的访问时间**　　s

| 读测试:小文件 | 未加载判定模块 | 加载未开启 | 加载且开启 | 读测试:大文件/s | 未加载判定模块 | 加载未开启 | 加载且开启 |
|---|---|---|---|---|---|---|---|
| 未添加限制条件 | 21 | 22 | 24 | 未添加限制条件 | 25 | 26 | 26 |
| 设限未开启 | 23 | 23 | 25 | 设限未开启 | 26 | 27 | 28 |
| 设限且开启 | 22 | 24 | 25 | 设限且开启 | 27 | 29 | 29 |

表 7 写操作的访问时间 s

| 写测试:小文件 | 未加载判定模块 | 加载未开启 | 加载且开启 | 写测试:大文件 | 未加载判定模块 | 加载未开启 | 加载且开启 |
|---|---|---|---|---|---|---|---|
| 未添加限制条件 | 25 | 26 | 27 | 未添加限制条件 | 29 | 29 | 31 |
| 设限未开启 | 28 | 28 | 29 | 设限未开启 | 30 | 28 | 32 |
| 设限且开启 | 27 | 27 | 30 | 设限且开启 | 32 | 29 | 33 |

## 5 结束语

本文构建一种基于环境属性的访问控制模型 EABAC,以丰富 Linux 平台中的访问控制机制。该模型将系统的环境属性加入到访问判定的过程中,扩展了 Linux 访问控制策略的范围。同时基于 EABAC 模型开发 Linux 平台下的访问控制原型系统。实验结果表明,该系统考虑内核空间和用户空间程序的运行效率,基于 LSM 框架实现内核判定模块,具有良好的通用性和可扩展性;在用户空间为用户提供友好的控制接口,实现策略的自定义配置。下一步将把更多的环境属性加入到安全模块中,添加系统日志等功能,方便系统管理员的审计工作。

### 参考文献

[1] 李凤华,苏 芒,史国振,等. 访问控制模型研究进展及发展趋势[J]. 电子学报,2012,40(4):805-813.

[2] 王小明,付 红,张立臣. 基于属性的访问控制研究进展[J]. 电子学报,2010,38(7):1660-1667.

[3] XU Zhongyuan, STOLLER S. Mini Attribute-based Access Control Policies [J]. IEEE Transactions on Dependable on Secure Computing, 2014, 12(5): 533-545.

[4] WANG Zhijie, HUANG Dijiang, ZHU Yan. Efficient Attribute-based Comparable Data Access Control [J]. IEEE Transactions on Computer, 2015, 64(12): 3430-3443.

[5] GUPTA P,STOLLER S,XU Zhongyuan. Abductive Analysis of Administrative Policies in Rule-based Access Control[J]. IEEE Transactions on Dependable and Secure Computing, 2014, 11(5):412-424.

[6] 李晓峰,冯登国,陈朝武,等. 基于属性的访问控制模型[J]. 通信学报,2008,29(4):90-98.

[7] RICHARD K, EDWARD J, TIMOTHY R. Adding Attributes to Role-based Access Control [J]. Computer, 2010,43(6):79-81.

[8] 韩道军,高 洁,翟浩良,等. 访问控制模型研究进展[J]. 计算机科学,2010,11(7):84-87.

[9] 吴新松,贺也平,周洲仪. 一个环境适应的基于角色的访问控制模型[J]. 计算机研究与发展,2011,48(6):112-123.

[10] ZHU Yan,HUANG Dijiang,HU Changyun,et al. From RBAC to ABAC: Construction Flexible Data Access Control for Cloud Strorage Services [J]. IEEE Transactions on Services Computing,2015,8(4):601-616.

[11] JAEGER T. Operating System Security [J]. Systhesis Lectures on Information Security, Privacy and Trust, 2008,1(1):1-218.

[12] FERRAIOLO D, SANDHU R, GAVRILA S. Proposed NIST Standard for Role-based Access Control[J]. ACM Transactions on Information and System Security,2001, 4(3):224-274.

[13] 李唯冠,赵逢禹. 带属性策略的 RBAC 权限访问控制模型[J]. 小型微型计算机系统,2013,5(9):103-107.

[14] LI Feng, SU Mei, SHI Guzhe. Research Status and Development Trends of Access Control Model [J]. Chinese Journal of Electronics,2012,40(4):805-813.

[15] NIU Shaozhang, TU Shanshan, HUANG Yongfeng. An Effective and Secure Access Control System Scheme in Cloud[J]. Chinese Journal of Electronics,2015,24(3): 524-528.

编辑 刘 冰

(上接第 170 页)

[28] FELT A P, WANG H J, MOSHCHUK A, et al. Permission Re-delegation: Attacks and Defenses [C]// Proceedings of Usenix Conference on Security. New York, USA:USENIX Association,2011:22.

[29] DIETZ M, SHEKHAR S, PISETSKY Y, et al. QUIRE: Lightweight Provenance for Smart Phone Operating Systems [C]//Proceedings of USENIX Conference on Security. San Francisco, USA: USENIX Association, 2011:23.

[30] ZHANG Yuan, YANG Min, XU Bingquan, et al. Vetting Undesirable Behaviors in Android Apps with Permission Use Analysis[C]//Proceedings of 2013 ACM SIGSAC Conference on Computer & Communications Security. New York, USA:ACM Press,2013:611-622.

[31] XU Rubin, SAIDI H, ANDERSON R. Aurasium: Practical Policy Enforcement for Android Applications[C]//Proceedings of USENIX Conference on Security Symposium. Bellevue, USA:USENIX Association,2012:27.

[32] BACKES M, GERLING S, HAMMER C, et al. AppGuard-Enforcing User Requirements on Android Apps [C]// Proceedings of Tools and Algorithms for the Construction and Analysis of Systems. Berlin, Germany: Springer-Verlag,2013:543-548.

[33] BACKES M, BUGIEL S, HAMMER C, et al. Boxify: Full-fledged App Sandboxing for Stock Android [C]//Proceedings of USENIX Conference on Security Symposium. Washington D. C., USA: USENIX Association, 2015: 691-706.

[34] XING Luyi, PAN Xiaorui, WANG Rui, et al. Upgrading Your Android, Elevating My malware: Privilege Escalation Through Mobile OS Updating [C]//Proceedings of IEEE Symposium on Security and Privacy. Washington D. C., USA:IEEE Press,2014:393-408.

编辑 刘 冰