平台无关的访问控制框架研究与实现①

胡大磊[1,2] 周学海[1,2](1.中国科学技术大学 计算机科学与技术系 安徽 合肥 230027;
2.中国科学技术大学 苏州研究院 江苏 苏州 215123)

摘 要: 为了使开发的访问控制模块可以在不同的平台上实施，通过总结不同平台异同点以及现有安全系统中访问控制实现方法，抽象出访问控制机制对平台的依赖，设计了一个平台无关灵活的访问控制框架PIFAC，且已经被实施在 Windows、Linux 以及 Rtems 系统上。

关键词: 访问控制；平台无关安全框架；核心安全服务体；多策略

Research and Implementation of Platform-Independent and Flexible Access Control Framework

HU Da-Lei[1,2], ZHOU Xue-Hai[1,2]
(1.Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China; 2.Suzou Institute for Advanced Study, University of Science and Technology of China, Suzou 215123, China)

Abstract: Access control is the mechanism to protect the operating system. At present, most systems lack proper access control mechanism. The implementation of access control mechanisms of the existing security system depends on the specific operating system platform, thus lacking portability. This paper sums up the differences between different platforms and the implementation methods of a variety of secureity systems. It abstracts the platform-dependence of access control mechanism, and designs a platform-independent and flexible access control framework PIFAC. This PIFAC has been implemented in Windows, Linux and Rtems.

Keywords: axess control; platform-independent secureityframework; core secureity server; multi-policies

目前，多数操作系统访问控制机制的不完善，是其容易被攻击的根本原因，访问控制机制的实施已成为各个操作系统的迫切需求。然而对访问控制实施的研究一般针对单个系统，当需要对新的操作系统进行安全加固时，往往需要重新开发新的访问控制模块。不同的安全实现，致使各个系统的安全配置也大相径庭，使得安全管理员的维护工作也相当繁琐。为了提高访问控制模块的可移植性，减少在新系统上实施访问控制时的重复工作，使不同操作系统具有相同的安全配置方法，必需设计一个可以方便的实施到不同操作系统中的访问控制框架。

基于对目前安全操作系统的实现机制以及访问控制理论的研究，本文提出了一个平台无关的访问控制框架(Platform-independent and Flexible Access Control Framework，简称 PIFAC)。PIFAC 对访问控制进行了抽象，将访问控制分为三层：操作系统 Hook 层(Operating System Hook Layers，简称 OSHL)、平台抽象层(PlatformAbstraction Layer，简称 PAL)以及核心安全服务器(Core Security Server，简称 CSS)。PIFAC 已经实施在 Windows、Linux 以及 Rtems 系统上，在新系统上实施访问控制时，只需要实现相应的 OSHL，增加 PAL 对新系统的支持，就可以将 CSS 实施到新的系统中，减少了在新平台上实施访问控制的工作量，并且使得实施 CSS 的平台具有相同的安全策略配制方法。本文以 PIFAC 在 Linux 和 Windows 系统上的实施为例，介绍 PIFAC 在不同系

① 基金项目:电子信息产业发展基金(财建[2008]329,工信部[2008]97)
收稿时间:2009-07-07

© 中国科学院软件研究所 http://www.c-s-a.org.cn

统上的实现方法。

1 相关研究

根据研究对象的不同，本文将对访问控制的研究分为两类，第一类是针对 BLP、BIBA、DTE、RBAC、MLS 等具体策略以及安全框架的研究，这类研究已经比较成熟；第二类是针对具体系统研究如何实施访问控制技术。

第一类研究的主要成果是动态策略框架的提出与实现，为各个平台实施访问控制提供了基础。文献[1]提出了一个访问控制通用框架 GFAC，将访问控制分为实施模块和执行模块；文献[2]提出了一种支持多种策略的Flask安全框架，Selinux[3]就是Flask在Linux上的实现。

第二类研究目前仍主要集中在单个操作系统上实施强制访问。如文献[4]介绍了嵌入式操作系统 WinCE 上强制访问控制的实现，文献[5]介绍了 Linux 上策略灵活的访问控制框架的实现，NSA 在文献[3]中介绍了其 Linux 安全加固方案 Selinux。国内安全系统还有 SecLinux、红旗 Linux 中文版、LIDS 以及计算所的 COSIX Linux V2.0 等[6]。

刘克龙等[6]将国内外安全操作系统的研究进程分为四个阶段：奠基时期、食谱时期、多策略时期以及动态策略时期。对于访问控制而言，这本质上是针对其不同发展阶段进行的划分，是对第一类研究的划分。本文认为，随着访问控制技术的逐渐成熟，以及各个平台实施访问控制的需求的增长，第二类研究将进入一个新的阶段：平台无关访问控制技术研究阶段。

目前已有的安全系统中访问控制模块的实现依赖于具体的系统，因而很难直接将访问控制模块应用到其他的系统中。访问控制实现的不同，访问控制的配置也必然不同，这就使得安全管理员必须熟悉多种平台的安全配置，大大增加了维护工作的难度。在对多种安全系统实现进行研究的基础上，本文总结了各个平台的不同点以及访问控制对平台的依赖，对不同平台进行抽象，将访问控制的实施分为三层，设计了一个平台无关的强制访问控制框架 PIFAC。

2 平台无关的访问控制框架(PIFAC)

GFAC 将访问控制抽象为安全执行模块以及安全实施模块[1]。在不同的平台上实施访问控制，最大的障碍是各个操作系统的实现不同。访问控制对平台的依赖主要体现在不同平台具有不同的内核对象以及不同的操作，以及安全执行模块使用的平台相关 API。下文首先以 Windows 和 Linux 为例，分析各个平台的不同点，然后介绍了 PIFAC 的结构。

2.1 平台间差异

2.1.1 操作集合

每个平台的操作集不同，以文件系统的实现来例，Linux 类操作系统中虚拟文件系统涉及的操作，如 Superblock、Inode 等操作是 Windows 不支持的；同样 Windows 系统有一些特色操作。故而各个平台都需要自己定义操作集合，操作集合的不同影响到了 OSHL 的实现。

2.1.2 内核对象

以进程为例，在 Linux 中每个进程由一个 Task_Struct 对象来描述，而在 Windows 中，每个进程由 Peprocess 结构来表示。而且不同的系统有特有内核对象，例如 Superblock 只会出现在 Linux 类系统中。内核对象的差异使得必须为不同的平台定义不同的内核对象集合。

2.1.3 内核 API

由于安全机制一般被放在内核空间执行，安全机制需要使用时间、锁、内存分配以及输入输出等操作，完成这些操作的 API 在不同平台下有不同的实现，要使得安全机制不依赖于具体的平台，必须消除安全机制对内核 API 的依赖。

2.1.4 重要的系统资源

不同操作系统的实现具有很大差异，每种系统都有独特的重要资源，比如 Linux 类的伪文件系统、Windows 下的注册表等系统关键资源，应该受到安全机制保护。所以，在设计安全模块时必须提供特定的机制保护这类资源。

2.2 PIFAC

PIFAC 将访问控制的实现分为三层：操作系统 Hook 层(OSHL)、平台抽象层(PAL)以及核心安全服务

© 中国科学院软件研究所 http://www.c-s-a.org.cn

器层(CSS)，如图 1 所示。

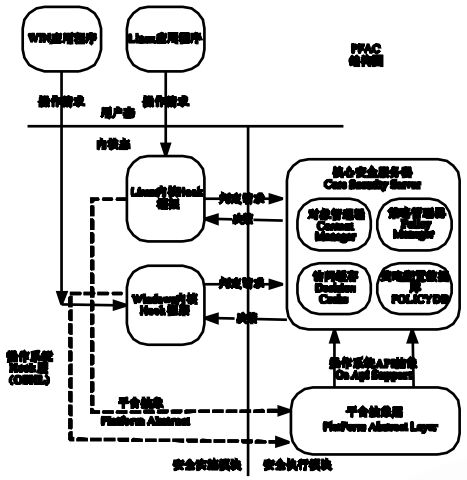

图 1　PIFAC 框架总体示意图

OSHL 层的作用是截获操作系统的关键操作后，交给 CSS 进行安全判断，然后根据 CSS 返回的结果，允许或者拒绝该操作。OSHL 层在不同的平台下有不同的实现，起到系统监控的作用。

PAL 层的作用是为 CSS 层提供平台抽象，使得设计 CSS 时不需要考虑各个平台的不同点；为 OSHL 层提供安全对象管理等功能。

CSS 层的作用是实现动态多策略安全模型以使 CSS 可以满足多样的安全需求。CSS 接到 OSHL 层的请求后，根据配置做出判断并且将判断结果返回给 OSHL 层。

3　PIFAC具体实现

3.1 OSHL 的具体实现

OSHL 是具体操作系统 Hook 机制实现的抽象，下面以 Linux 和 Windows 为代表，介绍 OSHL 的实现方式。

3.1.1 Linux 系统

Linux 为了支持各种安全模块，设计了 Linux 安全模块(LSM)。LSM 是 Linux 内核的一个轻量级通用访问控制框架，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出来。在 Linux 下，利用 LSM 框架实施了核心安全服务器(CSS)。

3.1.2 Windows 系统

Windows 是一个非开源系统，不能直接通过修改 Windows 内核代码以及修改 Windows 内核对象的方式来添加内核 API Hook。Windows Hook 技术的被分为 7 种[7]，本文综合考虑了可行性、性能与安全因素，采用 SSDT Hook 技术并参考了 LSM 框架[8]，设计了支持多种安全模块的 Windows Security Module(WSM)框架，如图 2 所示。

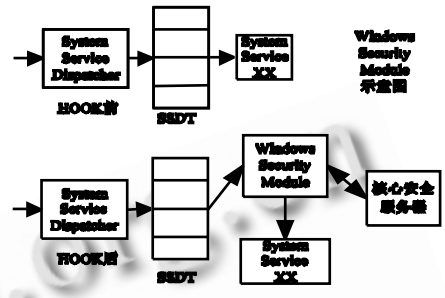

图 2　Windows Security Module 框架

3.1.3 其他系统实现方式

LSM 和 WSM 可以说代表了 OSHL 两种典型的实现方式，一种是对源码进行修改，另一种是类似 WSM 通过修改系统内核或重要文件的 HOOK 技术，本文分别称其为 Source-Hook 和 Binary-Hook。对于其他的操作系统，可以使用这两类技术之一实现 OSHL，然后加载 CSS 实现访问控制。

对于版本 2.6 以前不支持 LSM 框架的 Linux 内核版本，可以通过加载一个内核模块，Hook 系统调用表，实现 OSHL 层。但是，这将消耗更多的性能[8]，这也是本文采用 LSM 的原因。

3.2 PAL 的具体实现

PAL 层的主要目标是为 CSS 提供平台无关抽象，消除 CSS 对操作系统的依赖，并且为 OSHL 提供安全对象管理支持，操作集定义支持等功能。

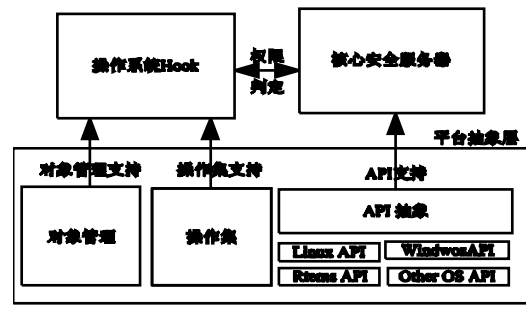

图 3　PAL 层设计

3.2.1 对象管理器 Object Manager

虽然不同系统平台的实现不同，但都有进程、线

© 中国科学院软件研究所 http://www.c-s-a.org.cn

程、文件、信号量、Socket 等类似概念。某些操作系统有特有的内核对象，例如 Linux 类系统有文件系统特有的概念如 Superblock、Inode 等。PAL 制订了一个默认集合，这个集合包含常见内核对象的安全对象，并且允许添加一些新的安全对象以支持扩展性，使得 PAL 的对象管理器可以支持所有的平台。本文称内核对象安全上下文的索引(SID)为安全对象，对象管理器的主要作用是维护安全对象，为 OSHL 层提供支持。

3.2.2 系统 API 抽象 OS_API

由于不同系统实现机制不同，其内核中使用的API也不尽相同。CSS 需要使用的内核 API 如内存管理、锁、输入输出、原子操作等机制在各个平台上的实现也必然不同。为了在实现 CSS 时，不需要考虑各个平台的差异，PAL 制订了一个 OSAPI 标准，使得各平台的 API 对 CSS 透明，这就消除了 CSS 对平台 API 的依赖。

3.2.3 操作集 Operation_Set

各个平台具有不同的操作集合，所以操作集的定义也被放在 PAL 层来实现，使得 CSS 无需关心 Operation_Set 的具体内容，而只需专心实现针对各操作的权限判定即可。和对象管理器类似，操作集也提供了各个平台操作的超集，各个平台在使用时只需要选择一部分即可。考虑到扩展性，允许各个平台使用自己定义的操作集。

3.3 CSS 的具体实现

CSS 的主要任务是管理内核对象的安全上下文，处理 OSHL 层的判定请求。

3.3.1 灵活的策略支持框架

CSS 的设计参考了 Flask,支持动态多安全策略，使得 CSS 具有高度的可配置性。 图 4 是 CSS 的设计结构，下面以 CSS 权限判定为例说明其工作流程：

(1) 内核向 CSS 发出判定请求。

(2) CSS 查询决策缓存，如果缓存中不存在对应项，则请求 VSS 进行权限计算；如果存在则跳至步骤(6)。

(3) VSS 向各个子安全模型发出判定申请。

(4) 各个子安全模型将判定结果交给 VSS。

(5) VSS 收到各个子策略的判定后，使用元策略做出判定并返回给决策缓存。

(6) 决策缓存返回判定结果给 OSHL 层。

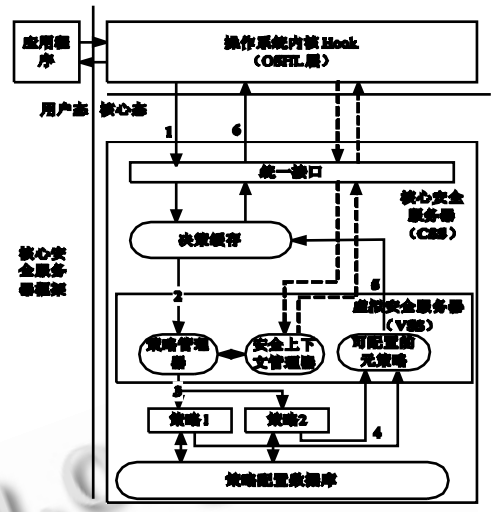

图 4　核心安全服务器设计

参考了 Flask[2]的设计，决策缓存用于提高访问控制判定的性能，可配置的元策略参考了 Meta- Policy。虚拟安全服务器(VSS)的作用是管理各个安全模型。VSS 制订了安全模型编写规范，规定了加载到 VSS 中的安全模型必须实现的接口。VSS 进行权限判定或其他操作时，会依次调用各个安全模块相应的操作。

3.3.2 CSS 中的平台无关特性

各个平台文件系统的不同实现影响了安全上下文的保存方式，本文将安全上下文保存方法分为两种：

(1) 保存在文件的扩展属性中，此方法适用于支持扩展属性文件系统中的文件。

(2) 保存在策略数据库中，此方法适用于不支持扩展属性文件系统中的文件，以及非文件对象。

CSS 没有规定文件安全上下文的保存方式，具体实现方式可以在 OSHL 层进行选择，消除了 CSS 保存安全上下文对各个系统的依赖。

4　系统实现与分析

PIFAC 已经在 Windows、Linux 以及 Rtems 系统上实施，本文总结了在新平台上实施 PIFAC 的步骤：

(1) 选择合适的安全对象，或者定义新的安全对象。

(2) 选择合适的操作集合，实现新系统的 OSHL，并且在其中启用 CSS。

在新平台上实施 PIFAC，只需要添加新的 OSHL，以及扩充 PAL，无需对 CSS 进行修改。 CSS 的设计支持多个安全模型，目前已经支持RBAC、DTE等安全模型。

(下转第 16 页)

© 中国科学院软件研究所 http://www.c-s-a.org.cn

(上接第 20 页)

5 结语

本文通过总结不同平台异同点和各种安全系统的访问控制实现方法，抽象了访问控制机制对平台的依赖，设计了一个平台无关灵活的访问控制框架 PIFAC，为在不同平台下实现访问控制提出了实际可操作的方法。PIFAC 框架是为了实现平台无关访问控制而设计，但其并不限于访问控制，其他的安全机制如安全审计，主动防御等技术，也可以参考这个框架的实现。

参考文献

1 Abrams MD, EK, La Padula LJ, Olson IW. A generalized framework for access control: An informal description. Mclean Virginia: Proc. of the 13th National Computer Security Conf, 1990. 135－143.

2 Spencer RSS, Loscocco P, et al. The Flask Security Architechture: System Support For Diverse Security Policies. Proc. of the 8th USENIX Security Symp. 1999.123－139.

3 Loscocco P, Smalley S. Integrating Flexible Support for Security Policies into the Linux Operating System. Proc. of the FREENIX Track: 2001 USENIX Annual Technical Conference. 2001. 29－42.

4 陈志平,雷航,杨霞,等.嵌入式安全操作系统的研究和实现.计算机工程, 2007,33(1):83－85.

5 訾小超,茅兵,曾庆凯,等.策略灵活的安全系统设计与实现.计算机研究与发展, 2004,41(2):299－305.

6 刘克龙,冯登国,石文昌.安全操作系统原理与技术.北京:科学出版社, 2004.

7 Ries, C. Inside Windows Rootkits. http://www.vigilantminds.com/files/inside_windows_rootkits.pdf

8 Wright CCC, Smalley S. Linux Security Module Framework. Ottawa Linux Symposium, Ottawa, Canada, 2002.

© 中国科学院软件研究所 http://www.c-s-a.org.cn