

基于 LSM 的分布式强制访问控制的设计与实现

阮越¹, 杨学兵^{1,2}, 周建钦¹, 纪滨¹

(1. 安徽工业大学 计算机学院, 安徽 马鞍山 243002;

2. 南京大学 计算机系, 江苏 南京 210093)

摘 要: LSM(Linux Security Modules)是得到 Linux Torvalds 本人支持的安全访问控制的底层架构, 强制访问控制是操作系统安全增强技术中经常采用的方式。为了在基于 Linux 的集群系统上, 实现节点间信息的安全流动, 在简要介绍 LSM 和强制访问控制的原理的基础上, 讨论了如何拓展 LSM 及其相应的安全政策来实现集群系统节点间的安全访问控制, 并设计和实现了基于 LSM 的分布式强制访问控制。最后指出了这一实现在性能上的缺陷和进一步改进的方向。

关键词: LSM; 强制访问控制; 集群系统; 分布式

中图分类号: TP393.08

文献标识码: A

文章编号: 1673-629X(2006)12-0250-03

Design and Implementation of Distributed Mandatory Access Control Based on LSM

RUAN Yue¹, YANG Xue-bing^{1,2}, ZHOU Jian-qin¹, JI Bin¹

(1. School of Computer Science, Anhui University of Technology, Ma'anshan 243002, China;

2. Dept. of Computer Science, Nanjing University, Nanjing 210093, China)

Abstract: The LSM(Linux security modules), which is supported by Linus Torvalds, is general access control framework for Linux kernel. The MAC(mandatory access control) is often used security enhancing technique of operating system. In order to implement secure information flow of nodes for Linux clusters, describes the principle of LSM and MAC concisely, and based on that, discusses how to expand LSM and security policies to guarantee secure access control of cluster nodes. At the same time, describes the design and implementation of distributed MAC based on LSM. In the end, point out performance limitation and improvement in the future.

Key words: LSM; MAC; clusters; distributed

0 引言

Linux 现在已逐渐成长为一个高端操作系统,越来越多的企业、政府、科研机构开始选用 Linux 构建分布式计算平台,集群系统是其中的主要解决方案。在充分利用 Linux 的高性能的分布式计算能力的同时,如何保证节点间信息的安全流动和安全访问,目前仍然是一个待解决的课题。

Linux 安全模块(LSM)是 Linux 内核的一个轻量级通用访问控制框架,现在已经成功融入主流的 Linux 内核 2.6.x 中(不再以内核补丁的形式给出),成为事实上的 Linux 内核级的安全访问控制的框架标准。但 LSM 只是考

虑了如何为单机环境下的内核级的访问控制提供支持,并没有考虑如何为分布式或网络环境下的节点间信息的安全流动提供支持,如何拓展 LSM 及其相应的安全政策来实现集群系统节点间的安全访问控制是文中讨论的主要问题。

1 LSM 及基于 LSM 的 MAC

1.1 LSM 的由来及工作原理

LSM^[1]这一设计构想起源于 2001 年的 Linux 峰会,当时 NSA(美国国家安全局)展示了它的 Security Enhanced Linux^[2]原型系统,该系统在 Linux 内核上实现了“可变的访问控制结构”——FLASK。Linus Torvalds 本人在看到 NSA 这一研究成果后,也觉得有必要在 Linux 内核中增加支持访问控制的底层架构。他一方面希望通过内核模块技术增强 Linux 内核安全性能,另一方面也希望尽可能地减少对内核的修改,使那些具体的安全策略及实现与访问控制框架无关,最终目的是希望将这一访问控制框架放到主流 Linux 内核版本中。

LSM 是一个相当成功的项目,许多著名的基于 Linux

收稿日期:2006-03-25

基金项目:国家自然科学基金(60473142);安徽省教育厅自然科学基金项目(2006KJ063B)。

作者简介:阮越(1972-),男,湖北红安人;硕士,讲师,研究方向为系统安全和嵌入式系统;杨学兵,副教授,研究方向为数据挖掘、网络安全;周建钦,副教授,研究方向为密码学、网络拓扑结构;纪滨,讲师,研究方向为信息安全。

的安全系统纷纷将其实现移植到其上,这其中包括 NSA 的 SELinux, Domain and Type Enforcement, LIDS (the Linux Intrusion Detection System) 等。

LSM 通过在一些内核数据结构中增加模糊的安全域 (opaque security fields), 通过“void *”实现, 和在程序流程的关键点插入 hook 函数来操作安全域和进行访问控制。LSM 提供了内核模块的载入机制和模块栈 (module stacking) 技术, 可以使实现具体安全策略的模块与 LSM 的框架相接, 并使得多个安全模块可以堆叠在一起并保持 LSM 接口的单一性。LSM 只对内核空间的内部对象进行操作, 如进程 (task)、i 节点、打开的 FILE 结构等, 并依据这些内部对象的安全信息来进行访问仲裁。

1.2 MAC 及基于 LSM 的实现

安全访问控制技术大概分成两种:一是自主访问控制, 包括细粒度的 ACL 和权能列表, 理论基础是访问控制矩阵; 一是强制访问控制, 根据某种规则通过比较主客体的安全敏感标记来实施访问控制, 理论基础包括实现密级控制的 BLP^[3] 模型和完整性控制的 Biba 模型等。

自主访问控制可以提高传统 UNIX 类操作系统的访问控制粒度, 但很难防范“特洛伊木马”这样的恶意程序, 强制访问控制可以有效解决这个问题。

图 1 给出了基于 BLP 模型的 MAC 的一个简单示例, 安全管理员按照安全政策将主体和客体赋予不同的安全属性标记。当某一主体 (进程, 安全标记为 Ssec) 在用户空间发出对某一客体 (文件, 安全标记为 Tsec) 的访问请求时, 最终必然通过系统调用接口陷入内核, LSM 截获这些系统调用, 利用钩子函数调用判定模块 (函数 f), 在函数 f 中将根据主客体安全标记中的等级分类和非等级类别蕴涵、相等关系, 利用“向下读”、“向上写”这种梯度安全标记单向流通的规则执行安全访问判定 (BLP 模型的判定规则详见文献[4]), 并最终通知内核本次访问可不可以实施。

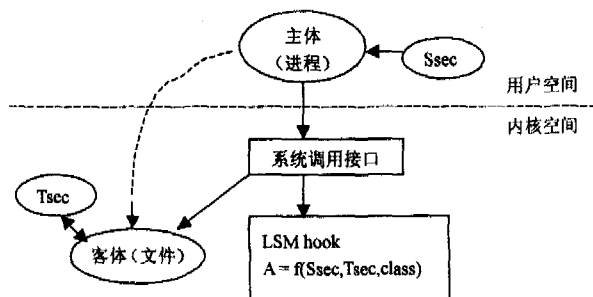


图 1 基于 LSM 的强制访问控制

2 基于 LSM 的分布式 MAC

2.1 分布式 MAC 的体系结构

分布式 MAC 是分布式安全框架的一个具体实现, 框架支持不同的安全策略并允许动态扩展, 框架的设计参考了文献[5,6]。其结构如图 2 所示。

该框架主要由安全服务器 (Security Server, SS)、安全

管理者 (Security Manager, SM) 和安全连接通道 (Secure Communication Channel, SCC) 组成。它们的作用简略描述如下:

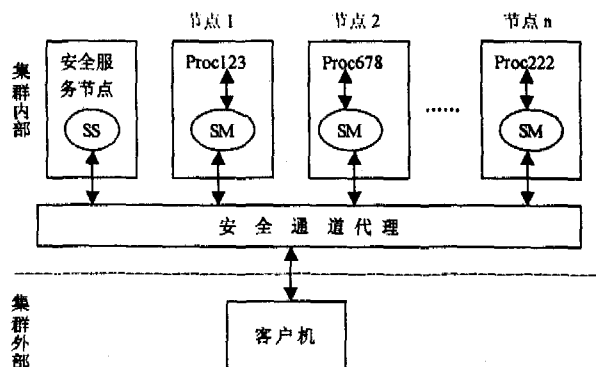


图 2 分布式 MAC 的体系结构

(1)SS: 整个体系结构的中心, 位于安全服务节点机上, 是安全管理的入口点 (人工配置安全策略) 和与外部的接口 (比如入侵检测系统返回的信息)。它负责动态生成分布式安全策略, 并将变化实时广播到各个节点上, 同时还负责为集群内的每个普通节点分配安全节点标识。

(2)SM: 位于集群的普通节点机上。它负责接收 SS 送来的安全策略, 并在本节点上实施策略和设置安全上下文 (本节点内主客体的安全标识)。为了防备恶意程序, SM 只和 SS 交换安全信息。

(3)SCC: 在 SS 和 SM 间以及 SS 和集群外部间提供加密的授权通道。

2.2 分布式 MAC 的实现

分布式强制访问可以分成两个层次: 一是主体 (进程) 和客体 (文件等资源) 位于相同的节点上; 二是主体 (进程) 和客体 (文件等资源) 位于不同的节点上。这里使用的分布式强制访问控制策略是扩展的基于 BLP 模型^[3] 的 MAC, 它的理论基础仍然是 BLP 模型。对于第一种情况就是基于 BLP 模型的 MAC, 判定规则和方法参见 1.2 节 (详细描述见文献[4])。下面重点讨论一下第二种情况。

由于访问控制跨节点, 所以访问判定不仅仅跟主客体自身的安全标识相关, 还跟主客体所处的安全节点有关, 主客体的安全标识必须做某种程度的扩展。

$Ssec = (SnID, SID)$, $Tsec = (SnID, TID)$, $Access = Function(SnID2, SID, SnID1, TID)$

上式的 SnID 为节点安全标识 (security node identifier); SID 为主体标识 (subject identifier); TID 为目标标识 (target identifier)。这时的访问判定将依据主体的节点安全标识和主体标识同客体 (目标) 的节点安全标识和客体标识共同做出。所谓安全标识实际是安全上下文的索引, 安全上下文中包含了主体/客体的等级分类和非等级类别以及节点标识等信息。等级分类和非等级类别存于系统的配置文件中, 节点标识在当前实现中是在系统初启时手工赋予的。等级分类和非等级类别以及节点标识, 它们之间的关系 (规则) 存于规则文件中, 配置文件和规则文件都

是标准的 XML 文件,它们存于安全服务器上且在整个集群内可见。

安全模块一旦加载,系统内主客体需要赋予安全标识,在本实现中,主体就是进程。

1) 主体。

(1) 模块初始化时,扫描所有进程,为每个进程赋予安全标识;

(2) 父进程执行 fork() 系统调用创建子进程,首先判断父进程有没有 fork() 权限,如果有,子进程将自动获取父进程的安全标识,除非显式地为这个子进程指定一新的安全标识;

(3) 如果子进程执行了 exec 类的系统调用,首先判断进程有没有 exec 类系统调用权限,如果有,将依据 exec 调用的客体(文件)安全属性重新赋予安全标识。

2) 客体。标识是动态创建的,在每次访问时,由安全模块检查该客体是否有安全标识,如没有就赋予一默认标识。默认标识的创建方法由配置文件和规则文件给出,利用原有的 Linux 传统访问控制标记的权限设定(文件属主、同组用户、其他用户),通过简单的换算关系得到。

3) 网络标识。当一个节点上的主体(进程)访问另一节点的客体时,它首先要取得本地 socket 的访问权限,在得到本地 socket 后,SnID(security node identifier, 节点安全标识)和 SID(subject identifier, 主体标识)将被添加到 IP 包的包头中(通过 LSM 在 IP 协议栈中的 hook)。在接收端,SnID 和 SID 将从包头中取出,然后创建一唯一的网络安全标识 NSID(network security ID), $NSID = \text{Function}(\text{SnID}, \text{SID})$,在当前的实现中,NSID 通过查找表换算得到,然后 NSID 就可以用于本地访问控制。

图 3 给出了跨节点的 MAC 的工作过程。假设节点 SnID2 上的主体 2(678# 进程,拥有标识 SID2)想要访问位于节点 SnID1 上的文件。那么首先主体 2 访问本地的通信资源(本地 socket),得到标识对(SnID2, SID2),然后写到 IP 包头中传递给远端节点。在远端节点 SnID1,这些标识从包头中取出,然后映射成唯一的 NSID。根据 NSID 将在本地得到主客体非等级类别和等级分类以及节点间的访问控制关系,最后依据安全判定规则裁决本次访问。当本次访问得到允许,主体 1(123# 进程)将成为主体 2 的代理执行本次访问。

3 结 论

经过测试,基于 LSM 的分布式强制访问控制可以用于 Linux 的集群环境中,可以有效抵御“木马”、“缓冲区溢出”等恶意程序的攻击。但某些情况下,性能会有比较大的损失,比如远程(跨节点的)UDP 访问时间效率大约损失 30% 左右(限于篇幅省略了详细的测试过程),系统的优化以及其他安全策略的实现将留待将来实现。

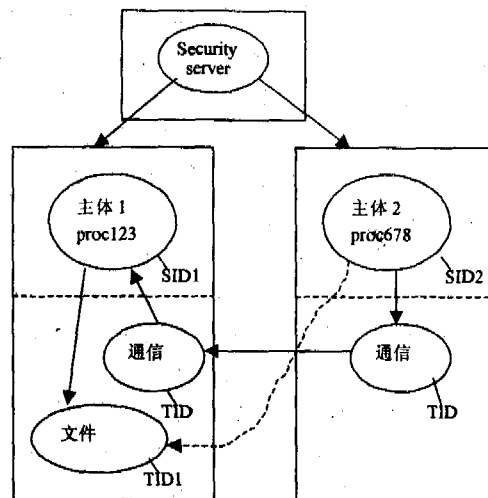


图 3 跨节点的 MAC

参考文献:

- [1] Wright C, Cowan C, Morris J, et al. Linux Security Module Framework[EB/OL]. 2002. <http://www.immunix.org>.
- [2] Loscocco P, Smalley S. Integrating Flexible Support for Security Policies into the Linux Operating System[EB/OL]. 2001. <http://www.nsa.gov/selinux>.
- [3] Bell D E, LaPadula L J. Secure Computer System: Unified Exposition and MULTICS Interpretation [M]. MTR - 2997 Rev. 1. Bedford, MA: The MITRE Corporation, 1976.
- [4] 阮越, 王成耀. 基于 LSM 的安全访问控制实现[J]. 计算机工程, 2004, 30(1): 4-5.
- [5] ISO 10181-3 Security Frameworks for Open Systems: Access Control Framework[S]. 1996.
- [6] Zakrzewski M, Haddad I. A Distributed Security Infrastructure for Carrier Class Linux Clusters[R]. Canada: Open Systems Lab, Ericsson Research Canada, 2003.
- [3] Suhai M A, Obaidat A M S. A comparative study of digital watermarking in JPEG and JPEG2000 environments[J]. ELSEVIER, Information Sciences, 2003, 151: 93-105.
- [4] LI C T. Digital fragile watermarking scheme for authentication of JPEG images[J]. IEE Proc. - Vis. Image Signal Process, 2004, 151: 460-465.
- [5] Loo P, Kingsbury N. Watermark Detection Based on the Properties of error control codes[J]. IEE, - Vis: Image Signal Process, 2003, 150: 115-121.
- [6] Delaigle J F, De Vleeschouwer C, Macq B. Watermarking Algorithm Based on A Human Visual Model[J]. ELSEVIER, Signal Processing, 1988, 66: 319-335.
- [7] 孙鑫, 易开祥, 费敏. 一种基于图像特征区域的数字水印系统[J]. 计算机工程与应用, 2002(23): 88-91.
- [8] 丁玮, 闫伟齐, 齐东旭. 基于离散余弦变换的数字水印图像[J]. 北方工业大学学报, 1999, 11(9): 71-75.
- [9] 牛夏牧, 陆哲明, 孙圣和. 彩色数字水印嵌入技术[J]. 电子学报, 2000, 28(9): 10-12.

(上接第 249 页)