在Linux安全模块框架授权钩子函数放置的一致性分析

TRENT JAEGER IBM T. J. Watson Research Center，ANTONY EDWARDS  Symbian Ltd.  and  XIAOLAN ZHANG IBM T. J. Watson Research Center

我们提出了一种一致性分析方法来帮助Linux社区验证Linux安全模块（LSM）框架中授权钩子放置的正确性。LSM框架由一组插入到Linux内核中的授权钩子组成，用于执行额外的授权（例如，用于强制访问控制）。与系统调用插入相比，内核中的授权更具有安全性和性能上的优势，但更难的是验证LSM钩子的放置是否能够确保所有内核的安全敏感操作都得到授权。静态分析以前用于验证这些中介（即，某些钩子中介对安全敏感操作的访问），但该工作无法确定是否检查了所有必要的授权集。在本文中，我们开发了一种方法来测试安全敏感操作和LSM钩子之间关系的一致性。其思想是，每当安全敏感操作作为可指定事件的一部分执行时，一定有一组特定的LSM钩子中介了该操作。这项工作表明，影响一致性的事件数量是可管理的，一致性的概念对于验证正确性很有用。我们描述了执行验证的一致性方法、实现此方法的运行时工具的实现、在打上了LSM补丁的Linux 2.4.16内核中发现的异常情况，以及此方法的静态分析版本的实现。

1. 介绍

Linux安全模块（LSM）项目旨在提供一个通用框架，在这个框架之下可以使得各种授权机制和策略得到强制执行。这样的框架将使开发人员能够实现他们为Linux内核选择的授权模块。然后，系统管理员可以选择最能实施其系统安全策略的模块。通常，其目标是除了传统的Unix自由选择策略之外，强制执行一个访问控制策略，以允许包含受损的系统服务。从2.6版开始，LSM框架已经以SELinux和Linux功能LSMs的形式实现，并且代码被合并到主分支Linux内核（www.kernel.org）中。

LSM框架通过在Linux内核中插入一组必要的授权钩子来实现引用监控器接口[Anderson 1972]。这些钩子定义了它们应该插入的位置以及对应模块可以实施的授权类型。将钩子放在内核本身而不是系统调用边界更具有安全性和性能上的优势。主要问题是，在一些系统中，是调用对象的名称，而不是其引用，从用户级进程传递到内核的（例如，open）。首先，系统调用行为必须将对象名解析为对象引用来对其进行授权。因为内核也会执行这个解析，所以两次解析就会造成不必要的性能开销。第二，更重要的是，对象名称和对象引用之间的映射可能在授权和内核解析之间发生更改，从而导致未经授权的访问。因此，系统调用插入据说容易受到时间检查到使用时间（Toctou）(time-of-check-to-time-of-use)攻击的影响[Bishop和Dilger，1996年]，在这种攻击情况下，在授权后，另一个非授权对象会被替换为授权对象。­

------------------------------------------------------------------------------------------------------------------------

分类和主题描述：D.2.9 [软件工程]：管理 –软件配置管理; K.6.5 [计算和信息系统的管理]：安全和保护 – 未授权访问

包含本文部分内容的早期版本发布在“Linux安全模块框架中授权钩子函数的放置的运行时验证”，第九届ACM会议计算机与通信安全，页225-234，十一月 2002年。

作者地址：Trent Jaeger and Xiaolan Zhang，19 Skyline Drive，Hawthorne，NY，USA 10532，电子邮件：{jaegert，cxzhang}@us.ibm.com; Antony Edwards，2-6 Boundary Row，London，SE1 8HP，United Kingdom，email：[antonye@cse.unsw.edu.au](mailto:antonye@cse.unsw.edu.au)。笔者是在IBM TJ Watson研究中心实习时完成这项工作的。

允许制作本材料的全部或部分数字/硬拷贝，不收取个人或课堂使用费，前提是这些拷贝不是为了牟利或商业利益而制作或分发的，并且这些拷贝在展示的第一页或初始屏幕上显示本通知以及完整的引文。必须尊重本作品中非ACM所有的组件的版权。允许用信用证提取。否则，要复制、重新发布、在服务器上发布或重新分发到列表，需要事先获得特定的许可和/或费用。

一般术语：设计，管理，信息安全

其他关键词和短语：访问控制模型，授权机制，基于角色的访问控制

引用监控器接口的一个关键点是，它确保所有受控制的操作（即那些为了保证安全性而必须受到控制的操作）在运行之前都得到授权。虽然在内核中放置LSM引用监控器的授权钩子可以提高安全性，但是更难的是确定钩子是否有真的对所有受控操作进行了介入和授权。系统调用接口是一个很好的中介点，因为所有内核控制的操作（即访问安全敏感数据的操作）最终都必须通过这个接口。在内核内部，系统调用接口没有明显的模拟。任何内核函数都可以实现对一个或多个安全敏感数据结构的访问。因此，任何媒体接口都处于较低的抽象级别（例如，inode成员访问）。除了中介之外，还必须确保为每个安全敏感操作强制执行正确的访问控制策略（例如，写入数据）。如果执行的策略与根据该策略执行的受控制操作不匹配，则会使得执行未经授权的操作成为可能。我们认为手动验证低级中介接口的正确授权是不现实的。

近年来，许多工作已经证明了静态源代码分析对于发现安全漏洞[Engler等人2000年；Larochelle和Evans 2001年；Shankar等人2001年]，甚至在一些合理假设范围内对一些安全属性的验证【Chen和Wagner，2002年；Zhang等人。2002】可能具有一定的效果。

在其他工作中，我们开发了一种静态分析方法，能够全面验证安全敏感操作中使用的变量是否已获得授权[张等人2002]。但是，更难的是确定安全敏感操作中使用的所有变量在执行所有必要的操作时是否是确定被授权的。首先，我们需要一个模型来帮助我们预测什么时候我们已经确定了必要的授权。第二，我们需要一种分析方法来验证这个模型。我们发现一个有用的见解是，许多LSM授权钩子如果都被正确放置了，那么授权需求的不一致性通常表示这之中存在问题。我们发现的另一个想法是，授权的一致性依赖于上下文，因此我们需要某种方式来表达和测试我们期望一致授权的上下文。我们发现使用运行时分析工具更容易探索出可能的分析选项，因此我们在这里描述了这种工具的性质。然而，我们最终发现，静态分析方法也可以利用这种方法，因此我们简要描述了这个原型。因此，我们提出了通过运行时工具开发分析以及检查它们在静态分析工具中的应用的想法。­

本文提出了一种一致性分析方法，来帮助LSM社区和Linux内核开发人员验证LSM授权钩子是否能够对访问进行完全的授权。我们还介绍了这种方法的实现，使用动态和静态的分析技术。在这两种情况下，实现都由两部分组成：（1）生成包含与测量一致性相关事件的系统日志的数据收集工具；（2）识别受控操作和LSM钩子之间一致性的一致性分析工具。系统日志的生成要么通过Linux内核的运行时工具，要么通过对Linux内核源代码的静态分析来完成。运行时收集是准确的，但缺少许多可能的执行路径，因此我们实现了一个静态分析收集机制，该机制生成兼容的日志。一致性分析通过从收集的数据中识别与期望一致性授权的不一致的内容，从中发现钩子放置的错误。我们设计了一种过滤语言来描述预期的一致授权的上下文。我们的分析工具生成了两种不同的表示，从这种表示中我们来发现其中的不一致性：（1）显示受控操作在执行与其授权之间一致性的授权图，以及（2）显示对于授权一致性敏感的受控操作的属性的敏感类列表。

使用这种方法，我们在文件系统中发现了三个lsm钩子放置的bug，这些bug现在已经被修复了，以及另一个导致重大讨论的异常。虽然我们目前使用的方法还不完整（例如，可能会遗漏一些错误），但我们对使用这些工具发现错误的能力感到鼓舞。我们将在打上了LSM补丁的Linux内核版本2.4.16上演示这些工具的使用。

论文的其余部分结构如下。在第2节中，我们定义了一般的钩子放置问题。在第3节中，我们开发了一种解决一般钩子放置问题的方法。在第4节中，我们概述了运行时数据收集和一致性分析工具的实现，并讨论了所执行的分析及其结果。在第5节中，我们描述了静态分析如何用于日志收集。在第6节中，我们描述了与使用这些工具相关的问题，例如回归测试。在第7节中，我们总结并描述了未来的工作。

1. 一般性的钩子放置问题
   1. 基本概念

我们在构建授权框架时确定了以下关键概念：

-安全敏感操作：这些操作会影响系统的安全性。

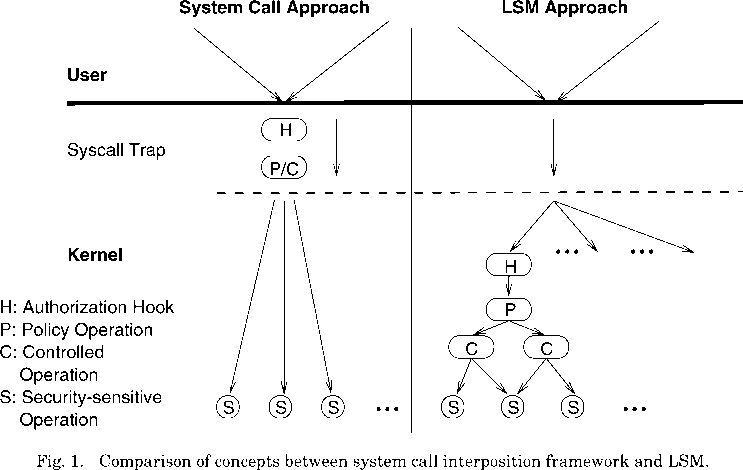
-受控操作：安全敏感操作的一个子集，可访问所有其他安全敏感操作。这些操作定义一个中介接口。

-授权钩子：系统中所具有的授权检查功能（例如，打上了LSM补丁的Linux内核）。

-策略操作：授权钩子授权的概念操作。

放置正确的授权钩子必须确保授权钩子授权了所有安全敏感操作。这些授权测试系统的授权策略是否允许请求主体执行特定的安全敏感操作。在更高级别（例如，文件读或写）表达授权策略更为方便，因此我们授权概念操作（我们称之为策略操作）而不是授权单个安全敏感操作。此外，由于安全敏感操作的数量可能很大，因此最好在调解所有安全敏感操作的接口上对它们进行一次授权。受控操作集定义了这样一个中介接口。因此，我们定义了我们的问题，用LSM授权钩子验证所有受控操作是否被授权用于预期的策略操作。­

与以往的系统调用中介机制相比，LSM的内核中介更难识别受控操作。如图1所示，系统调用接口以提供系统调用中所有安全敏感操作的中介而闻名。因此，系统调用接口既可以用作控制操作，也可以用作策略操作



当在内核中插入授权钩子时，中介接口不那么明显了，因此受控操作及其到策略操作的映射不再容易识别。例如写文件时，不是在系统调用接口上验证文件是否是写访问权限打开，而是在执行这些操作时比如目录（exec）、链接（follow link）或者最终写文件（write）的时候进行LSM授权。这种方法的好处是消除了对toctou（time-of-check-to-time-of-use）攻击的敏感性[Bishop和Dilger，1996]和冗余授权处理，但为了验证钩子的放置是否正确，需要更多的工作来识别受控操作以及它们对应的策略操作，并验证授权钩子是否正确的对它们进行了授权。­

* 1. 需要验证的关系

图2显示了概念之间的关系。

（1）识别受控操作：找到定义中介接口的一组操作，通过该接口可以访问所有安全敏感操作。

（2）确定授权要求：对于每个受控操作，确定必须由LSM钩子授权的授权要求（即策略）。

（3）验证完整授权：对于每个受控操作，验证LSM钩子是否授权了正确的授权要求。

（4）验证钩子放置的清晰性：实施策略操作的受控操作应易于从其授权中识别钩子。否则，即使对源代码进行细微的更改，也可能导致钩子无法生效。

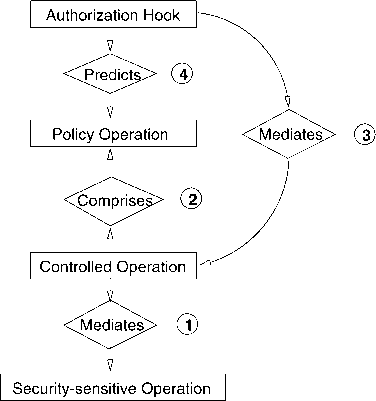


图2的授权概念之间的关系。需要验证的问题是：（1）确定控制操作; （2）确定授权的要求; （3）验证是否完全授权;（4）验证钩子放置清晰度。­

其基本思想是识别受控操作及其授权需求，然后验证授权钩子是否正确地介入了这些受控操作。首先，我们需要识别内核中具有代表性的受控操作。其次，由于受控操作的级别低于策略操作（即授权要求），我们需要一种方法来确定每个受控操作的授权要求。第三，我们需要将LSM 钩子的授权与预期的授权需求进行比较。这些任务对于内核内授权来说很复杂，因此很明显需要将任务自动化。­­­­

受控操作到授权需求的映射不一定是静态的。例如，可以对打开进行读取的文件执行许多相同的操作，就像对打开进行写入的文件执行一样。因此，上下文也是将受控操作映射到授权需求的决定因素。我们的方法必须能够有效地管理上下文依赖性，以便测试受控操作和授权需求之间的预期关系。­