在Linux安全模块框架授权钩放置的一致性分析

TRENT JAEGER

IBM TJ Watson研究中心的安东尼·爱德华Symbian公司和

小榄镇张

IBM TJ Watson研究中心

我们提出了一个一致性分析的方法来帮助Linux社区中核实Linux安全模块（LSM）框架授权钩子放置的正确性。LSM框架包括插入到Linux内核，以使附加授权的一组授权钩要执行（例如，用于强制访问控制）的。相较于系统调用的设置，在内核中的授权信息安全和性能上的优势，但它更难以核实的LSM钩子的放置确保所有内核的安全敏感的操作授权。静态分析先前已经用于验证调解（即，一些钩介导访问安全敏感操作），但这项工作没有确定一套必要的授权是否进行了检查。­­一致性的安全敏感操作和LSM钩子之间的关系。我们的想法是，每当一个安全敏感操作为可指定事件的一部分执行，LSM钩子的一组特定的必须介导该操作。这项工作表明，该影响的一致性事件的数量是可控和一致性的概念是用于验证正确性有用。我们描述进行验证一致性方法中，实现这个方法的运行时工具的实施，在LSM-补丁的Linux 2.4.16内核中发现的异常情况，并且这种方法的静态分析版本的实现。­

分类和主题描述：D.2.9 [软件工程]：管理 - softwarecon-无花果^比^管理; K.6.5 [Computingand信息系统的管理]：安全和保护 - ­越权存取

包含本文的某些部分的早期版本出现的“在Linux安全模块框架授权钩放置的运行时验证”，在 出发o ^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：特伦特积和小兰张，19 Skyline大道，霍桑，NY，USA 10532，电子邮件：{jaegert，cxzhang}@us.ibm.com; 安东尼·爱德华，2-6边缘行，伦敦，SE1 8HP，英国，电子邮件：[antonye@cse.unsw.edu.au](mailto:antonye@cse.unsw.edu.au)。而笔者是在在IBM TJ Watson研究中心的实习这项工作已完成。

权限，使这种材料的全部或部分的数字/硬拷贝不收取费用的个人或教室使用提供的副本没有制作或分发利润或商业利益和副本显示第一页或初始画面上这个通知与完全引证一起显示。这项工作由别人比ACM拥有的部件版权必须得到遵守。信贷抽象是允许的。要复制否则，重新发布，张贴在服务器或重新分配到名单需要事先特定的权限和/或费用。

（d 2004年ACM 1094-9224 / 04 / 0500-0175 $ 5.00

一般条款：设计，管理，信息安全

另外的关键词和短语：访问控制模型，授权机制，基于角色的访问控制

1. 介绍

Linux安全模块（LSM）项目旨在提供一个通用的框架，从中各种各样的授权机制和政策，可以强制执行。这种框架使开发人员能够实现他们的choosingfor的Linux内核授权模块。那么系统管理员可以选择最能强制他们systemJs安全政策的模块。通常情况下，目的是强制执行，除了传统的UNIX自由裁量政策，使受损的系统服务遏制强制访问控制策略。LSM框架已被接受到主线Linux内核（­[www.kernel.org](http://www.kernel.org)）为2.6版本了SELinux和Linux能力LSMS一起。

LSM框架通过插入一组作为必要的授权钩Linux内核的实现参考监视器接口[安德森1972]。这些钩定义类型的模块可以执行和它们的位置授权。在内核本身，而不是在系统调用边界配售钩具有安全性和性能优势。主要的问题是，在几个系统调用的对象，而不是其引用的名称，从用户级进程的内核（例如，打开）通过。首先，系统调用介入必须解决的对象名称的对象引用以进行授权。由于内核也执行这一决议，因此有不必要的性能开销。其次，更重要的是，对象名称和对象引用之间的映射可以授权和内核分辨率之间被改变，从而导致未授权访问。因此，系统调用插入被认为是易受时间的检查 - 时间 - 的使用（TOCTTOU）攻击[Bishop和迪尔格1996]，其中另一个目的是为交换授权对象授权之后。­

其中的一个参考显示器接口的关键方面是，它确保所有 控制操作（即，这些操作其控制是必要的安全性），它们运行之前被授权。同时将LSM参考监视器的授权挂钩内核可以提高安全性，更难以确定是否挂钩调解和授权所有控制操作。系统调用接口是一个很好的调解点，因为所有的内核的控制操作（即，访问安全敏感数据的操作）­­必须最终会通过这个接口。在内核中，有一个系统调用接口没有明显的类似物。任何内核函数可以包含访问一个或多个安全敏感的数据结构。因此，任何中介的接口是在抽象的一个较低的水平（例如，索引节点构件访问）。除了调解，还必须确保适当的访问控制策略（例如，写入数据）被强制为每个安全敏感的操作。如果实施的政策和属于该政策下执行的控制操作之间的不匹配，未授权的操作可以被执行。­

我们相信，一个低级别的调解接口的正确授权的手工验证是不切实际的。

最近，各种各样的工作已经证明静态源代码分析的安全漏洞[Engler等人的发现可能的有效性。2000; 拉罗歇尔和2001年埃文斯; Shankar等人。2001]和一些合理的假设[陈和2002年瓦格纳内的一些安全特性，即使验证; 张某等人。2002]。

在其他工作中，我们已经开发出一种静态分析方法，使全面核查，在安全敏感操作中使用的变量已授权[张等人。2002]。然而，更难以确定是否在安全敏感操作中使用的所有变量都被授权­所有必要的歌剧^^附件。首先，我们需要一个模型来帮助我们预测时，我们已经确定了必要的授权。第二，我们需要分析的方法，让我们来验证这个模型。我们发现有用的见解是，许多LSM授权钩子放置正确，所以在授权要求不一致常反映的问题。我们发现的另一个观点是，在授权的一致性是依赖于上下文的，所以我们需要一些方法来表达和我们预期一致的授权测试环境。我们发现它更容易探索利用运行时分析工具可以分析选项，所以我们在这里描述这样的工具的性质。但是，我们最终发现，这种方法也可以通过静态分析方法加以利用，让我们简要地描述这个原型。从而，­

在本文中，我们提出了一个一致性的分析方法来协助LSM社区和Linux内核开发人员在核实该LSM授权钩完全授权的访问。我们还提出实施这种方法的，usingboth运行时和静态分析技术。在这两种情况下，执行由两个部分组成：（1）数据收集工具，可生成包含相关于测量的一致性和（2）的稠度分析工具，识别控制操作和LSM钩之间的一致性的事件系统日志。系统日志生成则通过Linux内核的运行时间检测或由Linux内核源代码的静态分析来完成。运行时收集准确的，但是偏出许多可能的执行路径，因此，我们已经实施了产生兼容日志静态分析收集机制。一致性分析发现通过识别一致的授权预期不一致钩从该收集到的数据放置误差。我们设计，描述在其中授权一致预期上下文的过滤语言。我们的分析工具产生两种不同的表述，我们用来发现不一致：（1）­­­­授权图 该显示和受控的操作和它的授权的执行之间的一致性;（2） 灵敏度类列表­ 其是示出与所述授权一致性是敏感控制操作的属性。

使用这种方法，我们已经发现在文件系统LSM钩放置3个缺陷已经被修复，而另一个异常，导致显著的讨论。虽然我们目前使用的方法是不完整的（即，一些错误可能会错过），我们被发现使用这些工具的错误的能力感到鼓舞。我们证明在LSM-补丁的Linux内核版本2.4.16使用这些工具。

在本文的其余结构如下。在第二节中，我们定义一般钩放置问题。在第3节，我们开发的方法来解决一般的钩放置问题。在第4节，我们列出的运行时数据的收集和一致性分析工具的实施和讨论进行的分析和结果。在第5节，我们描述静态分析是如何可以用于日志收集。在第6节，我们将介绍有关使用此类工具，比如回归测试的问题。在第7节，我们的结论和描述未来的工作。

1. 一般HOOK PLACEMENT问题
   1. 概念

我们确定在授权框架的构建以下关键概念：

- 安全敏感的操作： 这些是影响系统安全的操作。

- 控制操作： 介导访问所有其他安全敏感的操作安全敏感操作的子集。这些操作定义­

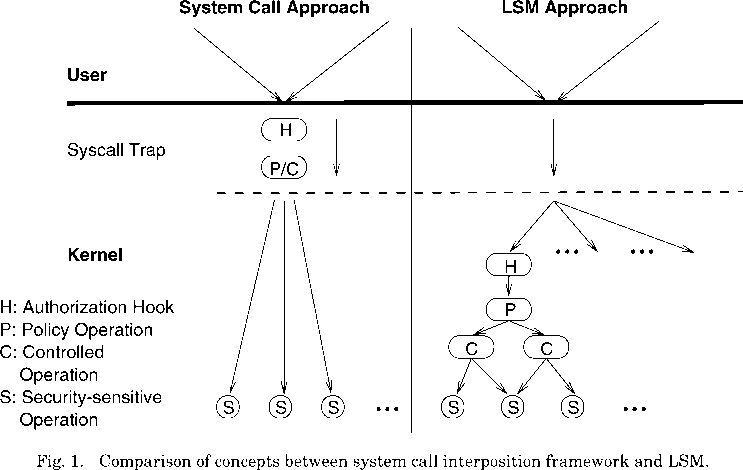
一个 调解接口。

- 授权钩： 这些都是在系统（例如，LSM打补丁的Linux内核）的授权检查。

- 政策操作： 这些都是由授权钩授权的概念操作。

正确的授权钩安置必须确保 授权挂钩­ 所有授权 安全敏感的歌剧^^附件。这样的授权测试系统的授权策略是否允许请求的主要执行特定安全敏感的操作。这是更方便的在较高水平（例如，文件读取或写入）来表达授权策略，因此而不是授权个人安全敏感的操作，我们授权的概念操作，我们称之为poZicy操作。此外，由于安全性敏感操作的数量可以很大，优选的是，在介导所有的安全性敏感操作的接口一次授权他们。该组控制操作定义了这样的中介的接口。因此，我们定义了我们的问题，以确认所有的控制操作被授权使用LSM授权钩预期的政策操作。­

识别控制的操作是比较困难的LSM比以往的系统调用调解机制的内核中的调解。如图1所示，该系统调用接口是众所周知的，在系统调用提供所有的安全性敏感操作的调解。因此，系统调用接口可以同时用作控制操作和政策操作。­­



当授权挂钩插入内核，调解界面不再明显，因此控制操作及其对政策操作映射不再那么容易辨认。例如，而不是在系统调用接口验证文件打开进行写访问，对于目录（可执行程序），链路（跟随链接），最终，该文件（写）LSM授权在当时，这些操作都是要执行完成。这种方法具有消除了TOCTTOU攻击[主教和迪尔格1996]和冗余授权处理易感的好处，但为了验证挂钩放置更多的工作是必要的，以确定控制操作，它们对应的政策操作，并验证授权钩正确授权他们。­

* 1. 关系验证

图2示出的概念之间的关系。

1. 识别控制操作： 找到一组定义通过所有安全敏感操作访问的调解接口操作的。
2. 确定授权要求： 对于每一个控制操作，确定必须由LSM钩子被授权的授权要求（即政策）。­­

⑶验证完成授权： 对于每个控制动作，验证正确的授权要求是由LSM钩授权。­

⑷检查钩放置净度： 实施政策操作来控制操作应该从他们的授权，很容易识别

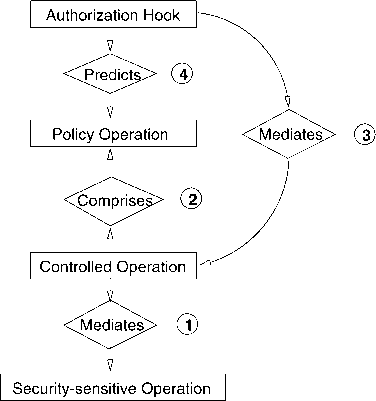


图2的授权概念之间的关系。验证问题是：（1）确定控制操作; （2）确定的授权的要求; （3）验证完整授权; 和（4）验证钩放置清晰度。­

挂钩。否则，即使给源琐碎的变化可能会使钩

无法操作。

其基本思想是，我们确定的控制操作及其授权要求，那么我们验证授权挂钩适当调解的控制操作。首先，我们需要确定在内核中代表控制操作。其次，由于控制操作是在比政策操作（即授权要求）一个较低的水平，我们需要每个控制操作的授权要求，可确定的方法。第三，我们需要比较到预期的授权要求进行的LSM钩子授权。这些任务是复杂的对内核的授权，所以很明显，自动化的支持是必需的。­­­­

控制操作来授权要求的映射不一定是静态的。例如，许多相同的操作都可以在开放阅读作为对写开放的文件的文件执行。因此，上下文也是在映射控制操作，以授权要求的决定性因素。我们的做法必须启用上下文的依赖，以进行有效的管理，从而控制运营和授权要求之间的预期关系可以进行测试。­

* 1. 相关工作

近日，静态分析来验证安全性已显示出希望。首先，现有的程序分析工具已被用于查找常见的安全错误，如缓冲区溢出和printf漏洞[球等。2003; Das等。2002; 拉罗歇尔和2001年埃文斯; Shankar等人。2001; Wagner等。2000。

其次，专门的工具已经被开发为寻找安全漏洞，如xgcc [恩格勒等。2000]，ITS4 /大鼠[Viega等。2000]，MOPS [陈和瓦格纳2002]，MAGIC [沙基等。2003]等[Ganapathy等。2003]。­

静态分析工具基于编程语言的形式属性，因此它们可用于完整的分析（即没有出现假阴性）。但是，静态程序验证的计算成本高昂，所以简化常常在分析模型方面。这些简化可导致更保守的分析（即，更多的假阳性）或某些特性（即，假阴性）的抽象。此外，静态分析工具可能需要的精力用于代码的注释，这是必要建立所需的分析模型显著量。­­­

专业分析工具专注于特定类型的ofbugs。恩格勒等。能够延长GCC的，叫xgcc， 做源分析，他们称其为 元编译[阿什克拉夫特和2002恩氏; 恩格勒等。2000; Hallem等。2002]。规则语言，称为金属，用来表示在高级语言进行必要的分析说明。而不是直接注释的代码中，金属规范定义的有限状态自动机，指导分析引擎。由于规则匹配多个语句，注释工作的量减少。各种软件漏洞，包括安全漏洞，已经被这个工具[阿什克拉夫特和2002恩格勒]研究发现。

大多数的专业分析工具缺乏完整性（即，可能会导致假阴性），但MOPS专门旨在为便于规范和分析的完整性[陈和2002年瓦格纳。使用MOPS，安全性被表达为有限状态自动机和方案表示为下推自动机。数据流是不是表示，因此混叠和价值的关系被忽略。然而，对于许多分析有用的错误仍然可以发现[陈等人。2004年]，而且往往可以显示很多数据流量关系不会通过其他手段[Zhang等人存在。2002]。­­­

在另一项努力，我们使用一个程序分析工具，CQual [Foster等。1999]，在一个方法来找到LSM钩放置错误静态[Zhang等人。2002]。使用GCC分析自动化CQual注释，我们就可以执行验证所有的控制操作由至少一个LSM钩介导的CQual分析。在一般情况下，我们还需要验证时，其所需授权钩子已经查了控制，操作只运行。CQual提供了可用于定义预期的授权类型点阵，虽然它是复杂的概念得到它的权利。此外，在控制运营和授权要求之间的关系的背景下，相关性是超出了CQual可以处理。­­­

Java静态分析工具，叫 JaB.A[Koved等。2002年]，已被用来收集对Java控制操作的实际授权。对于我们而言，这种方法有两个缺点：（1）它不分析Linux内核的C代码和（2）它没有提供有关所做的授权是否正确指导。关于第一点，我们实际上定义从C到雅巴分析概念[Zhang等人翻译。]，并建立了一个原型实现。关于第二点，JABA确实提供了一种上下文敏感控制流图，并且可以利用任何分析上下文敏感的数据流图。因此，我们将研究在第5生成分析日志使用这些图形。

由于使用这些方法的复杂性，我们发现，运行时数据采集帮助我们在得到准确的数据，很快，这样我们就可以探讨可能的分析选项。从检查收集到的数据，我们已经开发出我们在本文中，使我们能够确定是否适当的授权挂钩用于控制操作检查描述一致性分析方法。最终，方法是独立的，我们是否做在运行时或通过代码的静态分析收集到的数据一致性分析。在本文中，我们将考察数据采集的两种手段。­­

另一个相关的问题是系统的认证。从历史上看，黄皮书[NCSC 1985]用于在安全操作系统的施工指导，但现在正由通用标准[1998 ITSEC]取代。然而，认证工作是临时性的和费力的，并没有普遍提高通常使用的操作系统的安全性是成功的。古特曼认为在他的论文[古特曼2000]该认证的方法，包括正式的验证工具，是注定要失败的，除非他们表示在源代码级的概念。古特曼还主张的静态组合和运行时分析。我们使用不同于认证，在某种意义上说，它检查特定的错误，而不是提供一个自上而下保证整个系统满足其需求的方法。一个有趣的研究问题是，这样的检查有足够的广度和深度，是否可以提供一个信心堪比认证。与认证，源代码的发展这样的信心可以保持。­­­

1. 解决方案说明

关键的洞察力，我们在LSM框架运行时分析杠杆的LSM授权钩放置在很大程度上是正确的，使得不一致相对于常态的情况下很可能是一个指示错误的。例如，它会被认为是不寻常的，如果一个特定的控制操作对同一系统调用的不同运行不同的授权要求。虽然这种认识并不保证我们发现所有LSM钩放置错误（见第6节），它可以让我们确定了一些错误，并一直担任该工具开发宝贵的指导。

在所有下面的讨论中，我们采用了以下假设。首先，我们利用在控制操作对象在一个类型安全的方式在内核中进行处理的假设。这并不能否定任何发现错误的，但也可能有其他错误也是如此。其次，我们假设访问授权的数据类型的对象定义一组控制操作（即调解界面）。这些数据类型是对应于系统调用的概念（例如，文件，索引节点，插座，skbuffs，IPC消息队列，等等）的那些。到内核​​数据的访问旨在经过这些数据结构。虽然我们还没有明确证实这一点，我们已经做了其他地方提出了更详细的分析[爱德华兹等人。2001]。­­

* 1. 授权一致性

我们首先定义一个控制操作和一组授权要求之间的一致性。­

定义 1 （授权一致性）。 一个控制操作和一组授权需求（被授权即政策操作）之间的关系是 ­­利弊，3S ^ ^恩 如果每当执行受控的操作授权钩相关联的那些授权要求被调用。

我们发现，这种形式的一致性不是绝对需要的。在不同的系统调用，其具有不同的授权的情况下，可能会出现一个控制操作的执行。显然，在这种情况下满足授权要求会有所不同。­

因此，有必要能够限定其中一致性预期上下文。在一般情况下，上下文可以是任意的，但我们的经验是，三种类型的上下文关系：⑴ 系统调用;⑵系统调用与特定的输入（例如，标志）; 和（3）一组特定的控制操作。在第一种情况下，所述授权是在大的系统调用（例如，的fcntl）。这些授权适用于系统调用所有的控制操作。在第二种情况下，授权取决于一些参数到系统调用，通常是一个标志（例如，开放读）。因此，一些系统调用来下一个背景和一些以另一个。在第三种情况下，该组控制操作的外观，独立于它们出现的系统调用的需要特定的授权（例如，与访问相关联的该组所有者字段的操作）。在这种情况下，一致性忽略了系统调用的信息。

定义2（执行上下文）。 一个 执行上下文定义了一组执行路径。执行上下文可通过（1）的系统调用（所有执行）来定义; （2）系统调用特别参数值（或值的范围）; 和（3）一组受控操作（包括它们所有的路径）的。其他情况下的定义是可能的。­

我们的解决方案必须支持我们预期一致的授权环境的描述。通常，在静态分析上下文敏感的数据流是指不同的输入之间进行区分，以相同的功能。在这种情况下，上下文灵敏度是窄得多（仅取决于系统调用），或者可以忽略剩余的上下文的大部件（受控的操作集）。这类分析需要的静态分析器显著量注释的，并且可能取决于变量静态分析器（例如，用户提供的标志）的理解之外。例如，JABA完全忽略基本类型的值，但显然这些可以影响分析。­­­­­­

* 1. 授权一致性水平

一个执行环境通常由许多控制操作的，所以它是有帮助的聚合控制操作是在以同样的方式是一致的。

表一，授权一致性水平：名称和授权影响

|  |  |
| --- | --- |
| 水平 | 授权 |
| 系统调用 | 在系统调用的所有控制操作 |
| 系统调用输入 | 在同一个系统调用相同的输入所有控制操作 |
| 数据类型 | 相同的数据类型的对象，所有控制操作 |
| 宾语 | 同一对象上的所有控制操作 |
| 会员 | 上相同的数据类型的所有控制操作，访问同一部件，用相同的操作 |
| 功能 | 在相同功能的所有成员一样控制操作 |
| Intrafunction | 同样的控制操作实例 |
| 路径 | 相同的执行路径相同的控制操作实例 |

例如，如果所有的上下文控制操作具有相同的授权，那么我们就可以在大的，而不是单独的操作视图相对于上下文一致性。­

我们发现，我们可以描述每个受控运行时，它是在特定情况下由一组我们称之为离散值的执行被称为授权钩之间的一致性 ­一致性水平。 此外，一致性水平形成总阶如下。

定义 3 （一致性级别总订单）。 如果两个不同的控制操作权限一致为同一值oflevel我，那么他们的授权级别的任何值一致 ­Ĵ 哪里 I>Ĵ 在里面 一致性水平全序 （见表一）。

如果两个不同的控制操作都在同一个对象上执行的，但他们有一致的授权，那么该成员和访问这些操作的值不影响一致性。例如，如果一个特定的对象上的所有控制操作具有相同的授权要求，那么也不要紧成员访问是什么。表1列出了离散的一致性水平。我们统称这一组级别作为授权一致性水平。­ 这些级别包括受控OP- erationJs执行的各个方面，包括（在函数系统调用，系统调用输入，功能，位置，路径控制操作）下它被执行的情况下，将其在（数据类型执行的对象和对象），并执行操作（构件和访问）。

一致性水平骨料控制操作成 稠度等级 所有的控制操作具有相同的权限挂钩称为鉴于目前的位置。

定义 4 （一致性类）。 两个不同的控制操作属于同 ­稠度等级 对于执行上下文，如果他们有相同的权限挂钩称为每次他们在这方面的执行。

* 1. 授权一致性影响

控制操作可以通过授权一致性的分类划分控制操作分为两类：⑴ 众所周知异常

和（2）稠度类，它们的授权要求需要验证。在第一种情况下，我们考虑一些授权的一致性水平是非法的。我们定义下面这些情况不变。在第二种情况下，我们必须确定在执行上下文每个受控操作的最大一致性水平是否指示上可接受的授权要求或没有。例如，如果一个受控操作属于一组在物体一致性电平控制操作，这表明所有的对象上的控制操作具有相同的授权检查。它是那么的手动任务，以确定这是否是正确的。然而，一致性的聚集体的数量表示一个分区的受控操作成具有相同授权的最大尺寸的类。­­­­­

* + 1. 异常。 授权于表I中的双线下方的水平的一致性， intrafunction 和 路径，一直被认为是反常的。这些类型的敏感性意味着一个函数（intrafunction）内执行路径（路径）或位置确定特定受控操作的相同部件上的授权要求。­

下面不变正式表达了我们的路径不一致不变。

定义5（路径不一致不变）。

VC1，C2式C， E1，E2ËË（C1 = C2）八（E1 = E2） - ^（C1， E1） = E（C2， E2）（1）

这不变的状态，同样的控制操作（C1 = C2）在同一事件中运行（E1 = E2 通过系统调用和它的输入端定义）必须具有相同的授权要求（由函数定义 [R）。也就是说，事件内执行路径可以在不影响受控operationJs授权要求。­

同样，我们定义intrafunction不一致不变。

定义6（Intrafunction不一致不变）。

VC1，C2式C， E1，E2ËË， （F（d）= F（C2））八

（中号（C1） = M（C2））八（E1 = E2） - K（C1，E1） = E（C2，E2）（2）

在这种情况下，在同一个函数两个受控操作（通过函数计算 F），并且使相同的成员访问（通过函数计算 中号 ）必须具有相同的授权要求 [R。

* + 1. 授权一致性类。对于其他情况下，我们不能很容易地识别为错误。相反，我们分区控制业务并入其授权的一致性类，并确定它们的授权要求是否正确。­

授权稠度等级计算如下。起价为最高（系统调用）每个一致性水平，我们分区控制操作进入一致性类，所有控制操作具有一致性水平相同的值，然后我们测试类是否也有相同的权限。如果没有，那么我们尝试基于两个层面，并再次测试下一个较低的水平和分区。这种方法反复进行，直到我们已经交给的各项控制操作一致性类。­­

分类是通过一致性水平定义。系统调用级别，所有系统调用的控制操作都在一个班。为系统调用输入，相同的系统调用的，并用相同类型的输入所有受控操作是聚合（见3.4节）。对于数据类型级别，受控操作由系统调用，输入，并且该操作的对象的数据类型进行分类。因此，在分析的每一步中创建依次更细的分区。

甲分类成功（即，为x一致的，其中x是水平），如果它是在其中所有在该类受控操作具有相同的授权的第一级。需要注意的是其他类，在具有相同授权凝聚而形成的最大规模的类相同的一致性。一旦类被创建它是一个手动过程来验证每一类的授权是正确的。对于文件系统，类的数量是足够小，手工验证是可行的。­

作为一个例子，考虑读系统调用。文件操作是数据类型一致的，因为在文件对象的所有控制操作被授权读取。手动验证涉及到检查读取文件的权限就足够了。由于读的授权也适用于文件的inode，我们纪念文件的inode作为授权读为好。然而，分类后，一个索引节点的控制操作未被授权。这是一个不同的对象，所以索引节点的操作可以是目标一致的。这是该文件的目录inode的操作，以确定是否一个信号应该在这个目录中读取的结果被发送。[[1]](#footnote-2)

除此之外，当授权完全缺失，用于识别错误最常见的方式是找到两个分类（即两个积累了与不同的授权）执行的重要的共同操作。这种情况多发生在那里的fcntl两个不同的类别（根据不同的系统调用输入）在相同的操作F\_­­〇wner字段（参见4.2.4节）。

* 1. 必要的数据采集

通过登录系统调用进入/退出/参数，函数入口/出口，控制操作（即，对象，数据类型，部件，和操作），和授权，我们收集所有的一致性水平必要的值。所有的信息可以很容易地记录，但有意义的对象标识符和系统调用输入的变化的识别需要进一步的分析。

在执行期间，对象通过函数指针引用，但这不是必需的对象的足够的识别。例如，一个inode具有在授权使用持久标识符（即，设备，节点号码）。因此，对于每个数据类型，我们定义了用于计算特定的方法其

表二。日志记录类型

|  |  |
| --- | --- |
| 记录类型 | 数据 |
| 控制操作授权函数入口函数退出 | 上下文ID控制操作ID OID上下文ID授权ID OID上下文ID指令地址上下文ID |

对象标识符。这些标识符用于确定对象上的所有操作和授权。

在系统调用的情况下，我们假定在同一个变量使用的对象具有相同的授权要求。为了模拟这种情况，我们使用第一控制操作在其中一个对象显示为一个标识符。如果两个对象在相同的受控操作必须将它们分配给同一个可变第一访问（因为变量将在两个受控操作相同）。然而，不同的执行路径可能导致在一个不同的受控操作中使用第一相同的变量之中。然而，与相同的授权要求类聚合将合并这些情况，所以这个假设已经被证明有效。

该系统调用的参数改变几乎每一个电话，但只有少数的论点真正影响的授权（例如，存取标志上打开）。因此，我们收集到的参数，但是只能使用我们发现影响授权要求做到分区的参数。只有很少的系统调用我们已经讨论基于他们的输入参数，如打开，读写控制，和fcntl有不同的权限。因为不同的授权是根据不同的输入所使用的，这些系统调用都比较复杂，因此，更容易出错。­­

1. 运行时实现

完全授权的运行时分析包括两个步骤：产生内核执行日志和一致性的离线分析的。本节介绍了创建执行日志，用来准备和显示数据分析日志过滤工具的实施，我们的分析结果工具的执行情况。

* 1. 收集运行时信息
     1. 日志内容。表II示出了运行时分析过程中收集的信息。控制操作由元组确定­（在^^构作^上指针，对象类型，构件，访问）。一个控制操作ID分配给每个唯一组合。授权被唯一标识（LSM\_hook，政策操作）。等控制操作，一个唯一的授权ID被分配给每个。函数入口和出口记录为好。该函数入口地址唯一标识功能。­

对于每个控制操作或执行授权，日志必须包括所涉及的对象（例如，索引节点）的身份。 ­对象身份 （OID）数据库中每个对象类型定义，例如，i节点是由设备ID，索引节点标识

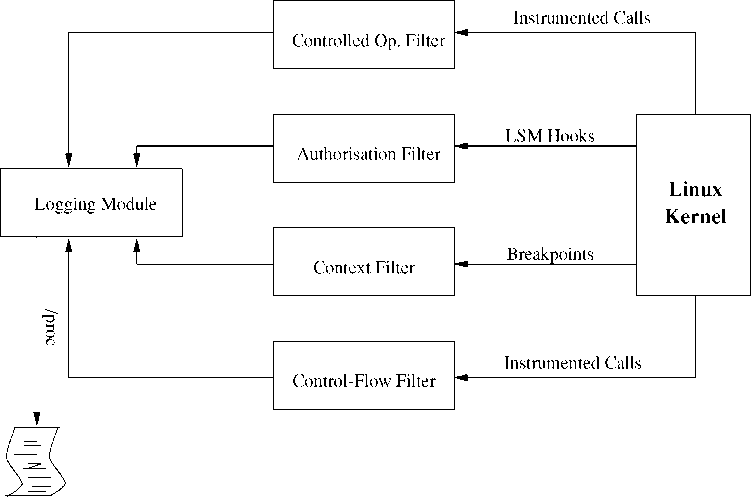


图3.实施的架构。

数量，同时任务由进程ID标识。OID是唯一的要求是上下文中是唯一的。

我们使用的一个概念 上下文意味着一个内核事件（例如，系统调用）的处理。授权显然仅在其执行上下文中有效，因此，日志条目还必须包括控制操作和授权的情况下。

* + 1. 商品系列。图3给出了工具的概述。日志的创建包括三个阶段：必须生成所需的信息，它必须被收集，而且必须写入日志。­­

信息以三种不同的方式产生。首先，通过在LSM钩生成的授权信息。其次，通过与海湾合作委员会的修改版本标识控制操作和仪器与之前所有的这些操作的调用到一个处理函数内核编译内核生成的控制操作的细节。控制流信息也通过在编译时插装内核产生的。通过将断点在内核生成的第三，上下文信息。这三种方法更详细地在下面的章节中讨论。­­­

第四，内核模块被加载到接收图3中所示的这些模块执行粗粒过滤中的信息，并安排信息转换成正确的格式，使所述记录到记录模块之前。记录模块分配的上下文ID到传入记录并将该信息写入到缓冲区中。­­

* + 1. 授权信息。已经由LSM补丁提供鱼钩登录授权信息，则需要那么一点额外的实施。授权过滤器是一个简单的LSM模块，增加了对每个授权日志条目。这些日志条目标识（例如，RMDIRPARENT，RMDIRTARGET）进行了授权和授权的对象。
    2. 控制操作。 要登录控制的操作，我们首先必须在内核中找到控制操作，然后检测这些操作的执行的机制。

在内核识别控制操作需要源分析。而不是直接源代码分析（这是困难的），我们选择了通过分析GCC的中间树表示来识别控制操作。随着Linux依赖GCC扩展，源代码分析需要使用GCC解析器，因此利用它已经构建树的似乎是合乎逻辑。为了识别控制的操作，我们遍历树寻找其介导的数据类型的成员访问表达式。­­[[2]](#footnote-3)当检测到控制操作中，我们插入到功能\_\_controlled\_op包括对象，类，构件，和访问，其中，所述表达存在语句之前的呼叫。如果表达式是一个循环的条件语句，则呼叫在循环之前和在每次迭代的结尾插入。此调用包含了所有识别控制的操作，并允许处理器以提取对象的身份所需的信息。­­

一对夫妇的访问造成了这种方法的问题。首先，有可能通过经由指针拍摄部件的地址，将其存储到指针，以及改变部件来修改结构构件。由于最初的访问是读入指针变量，这是可能的，我们可能错过了随后写入。而不是执行更广泛的源分析，以确定这些情况下，我们简单地当出现混叠检测。其次，这也有可能是我们错过了，当他们投以非控制型访问控制的数据结构。这也是检测。我们的初步分析表明，发生在少数的方式（虽然第一，大量的倍），因此它们可以被作为特殊情况处理这些案件。

* + 1. 控制流。通过编译与-finstrument-功能开关通过GCC-3.0提供内核产生的控制流程信息。此选项使编译器插入在每个函数的入口和出口调用处理函数。这些处理函数然后将信息传递给相应的模块。
    2. 上下文信息。由于可能存在多个执行上下文中随时内核，所有日志条目必须包含上下文ID，因此分析可以知道哪些项目之间的相互关系。不幸的是，没有钥匙是可用的唯一标识一个执行上下文，因此，我们必须选择一个非唯一键定义一个方法使用相同的密钥来区分上下文。

我们选择了目前内核堆栈作为非唯一钥匙的基础，因为我们需要的关键是同时活动的执行中至少独一无二的，它似乎不可能被侵犯堆栈这个属性。虽然同时积极执行中是独一无二的，内核堆栈是不是有三个原因每个方面与众不同：所有的系统调用来自同一过程中使用相同的内核堆栈，一旦进程终止其内核堆栈可能会被重新分配到一个新的进程，并中断与他们中断处理的内核堆栈执行。这里的关键特性是，尽管其内容的关键不是唯一的，使用相同的密钥上下文从不交错。因此，通过记录上下文（以及相关联的密钥）的开头和结尾，我们可以明确地分配日志条目上下文。

幸运的是，只有几个点，其中上下文可以开始（全部位于entry.S中），以及数量大致相等的那个情境可以结束的地方。出射系统调用是因为它不会返回，因此，在该do\_exit schedule.call（）也识别为上下文出口点的例外情况。由于入口/出口点的数量是可控的，我们手动确定它们的位置为每个内核版本，并自动在运行时将其插入。为了收集在运行时该信息，上下文过滤器插入断点指令到内核在所有出入境口岸（的记忆图像）。当执行一个断点，上下文过滤器创建一个包含上下文关键日志条目，以及这是否是一个上下文的开头或结尾。­­­

* + 1. 性能。我们做了一个简单的性能检查，以确定在检测的内核性能下降。在未修改Linux内核，配置的LMBench为“快标杆”用了3分钟4秒运行。仪器化内核用了3分钟24秒至运行的略高于10％的下降同样的基准。我们认为，这种开销是对这种分析完全可以接受的。回想一下，内核是分析仪，以及钩子不需要后续使用的内核。­

在该试验中，与上述的结果集合中，我们采样1出of20系统调用。这样做的原因是为了保持日志的增长速度比磁盘吞吐率低。由于这些基准执行相同的系统调用很多次，我们没有注意到我们“丢失”的任何安全相关信息。如果有必要，以确定何时丢弃日志项政策可以设计。

* 1. 日志分析

我们还建立了一个工具，使日志分析为3.2节所述识别授权要求的一致性。该工具使规则规范提取所需的日志条目，称为日志过滤规则，并计算给定的所提取的条目授权的一致性。我们在两个方面显示一致性的结果：（1）授权图 该显示每个授权和控制的操作和（2）之间的一致性 一致性类名单 该显示控制操作的授权和一致性水平的聚集。

* 在0xc014f046操作路径一致的规则

1. =（+，ID\_TYPE，CONTEXT）（+，di\_cfm\_eax，READ）
2. （d，1）=（+，ID\_TYPE，CNTL\_0P）（+，di\_dfm\_ip，0xc014f046）
3. （d，L） =（+，ID\_TYPE，SEC\_CHK）

* 对于inode的成员i\_flock会员一致的规则，读访问

1. =（+，ID\_TYPE，CONTEXT）（+，di\_cfm\_eax，READ）
2. （d，1）=（+，ID\_TYPE，CNTL\_0P）（+，di\_dfm\_class，0T\_IN0DE）

（+，di\_dfm\_member，i\_flock）（+，di\_dfm.access，0P\_READ）

1. （d，1）=（+，ID\_TYPE，SEC\_CHK）

* 输入一致的开放读访问规则，但不path\_walk

1. =（+，ID\_TYPE，CONTEXT）（+，di\_cfm\_eax，OPEN）（+，co\_ecx，RD0NLY）
2. （d，1）=（+，ID\_TYPE，FUNC）（+，di\_ffm\_ip，path\_walk）
3. （d，1）（N，2）=（+，ALL，0,0）

图4实施例的授权一致性过滤规则。

1. 日志过滤规则。日志过滤工具把执行日志，并设置过滤规则作为输入，并输出符合规则的日志条目。规则语言是目前水平还比较低，因为我们一直在用更证明可行性，而不是创造一个很好的高层次的规则语言有关。然而，我们证明规则语言给了可能的类型的分析感。

规则库由一组定义匹配要求规则定义的。一个规则由（1）的索引的; （2）的依赖关系规范; 和（3）一组语句。索引标识的规则库中规则。依赖状态的指标，其他规则的关系。我们可以说，一个规则只能匹配也被另一个规则匹配的条目，（d， 一世），其中 一世是其他规则的指标。此外，我们可以说依赖一个规则不包括由其他规则匹配的条目­一世，如（ñ， 一世）。最后，语句描述条目匹配条件。这些通过识别的条目类型（ID\_TYPE），然后匹配的特定于类型的水平指定。条目类型包括：事件（上下文），授权（SEC\_CHK），功能（FUNC）和控制操作（CNTL\_OP）。­

图4示出了一些示例性规则。运行在指定地址的控制操作时，路径一致的规则找到在读系统调用的上下文中的所有授权。第一行收集所有上下文条目读取系统调用（即系统调用的开始）。第二行收集在指定的位置控制操作的所有条目。在（d，1）意味着，这种说法是依赖于声明1，所以仅读出系统调用上下文内的条目将被收集。第三行收集read系统调用范围内的所有授权。在这种情况下，该控制操作的每次执行应具有相同的授权或有违反的­­路径不一致不变 从具有多组法律授权的禁止控制的操作。

接着，功能一致的规则收集读取系统调用范围内的所有授权和“读取的信息节点成员Lflock”的控制操作。（d，1）上的第二行中的指定意味着这种类型的读出系统调用中的所有受控操作将被提取。如果这个控制操作相关的授权是不一样的，那么成员访问是与它的定位是一致的。­

系统调用输入一致的规则收集在每个开放的系统调用只读访问的所有日志条目。开放的系统调用的权限取决于打开文件进行了访问，如此开放的系统调用输入 - 是一致的。此外，我们还显示了此规则，消除了path\_walk功能的范围内的所有条目的负过滤器。文件查找的授权，包括任何链接遍历，可以从那些授权打开此文件来进行分离。这种过滤功能使我们能够灵活地选择我们的分析范围。­

我们面临的挑战是编写表达可以进行有意义的分析的情况下登录过滤规则。在一般情况下，我们希望日志过滤规则来描述其导致的一致性类符合我们的预期的情况。我们已经发现，其次是细化一致性的乐观最初的假设有效地适用于设计这样的规则。­

我们假设的一致性，系统调用一致性的所有控制操作的最高水平开始。如果所有的系统调用执行的控制操作具有相同的权限（即，被系统调用一致），那么我们只需要验证的授权是正确的。如果没有，我们考察我们是否可以在一致性的一个新的水平，该系统调用输入编写规则。如果输入有没有明显的效果，或者它们不能解决一致性，以我们的预期（即，所有的控制操作不具有相同的权限），然后我们去到一个新的水平，数据类型一致性，等等。­­­

通常情况下，有效的分析是可能的较低水平的一致性，而不改变规则。我们只是需要一种方法在同一一致性水平，以查看这些聚集体，如每个数据类型以它的授权的在数据类型一致的情况下的映射。我们使用的一致性类列表来查看多个一致性类（参见4.2.3节）。

1. 图形日志分析。该分析工具还可以生成图表，使经滤波的数据的可视化分析。使用这些图，可以通过检查来验证授权的一致性，我们将在下面说明。一个授权图包括在过滤后的日志两组节点：（1）控制操作和（2）的授权制成。边从每个控制操作吸引到已不满足在运行时的授权。有两种类型的边缘的：（1）总是 边缘意味着相关联的授权是满意每次运行受控的操作时间和（2） 有时 边缘是指在运行控制操作时，相关授权满足至少一次。

一个总是边缘（以及缺乏的边缘的）指的是，授权不与较低级别的水平相一致。甲有时边缘指示不一致。缺少其中一个边缘将预期将指示缺少授权的边缘。­

图5示出了示例授权曲线图。在示例曲线图是使用多点的图形可视化工具[Koutsofios和北]中显示。在这种情况下，授权图表显示控制操作和两种类型的fcntl呼叫的授权：（1）的fcntl（FD，F\_SETOWN，pid\_owner）和­­

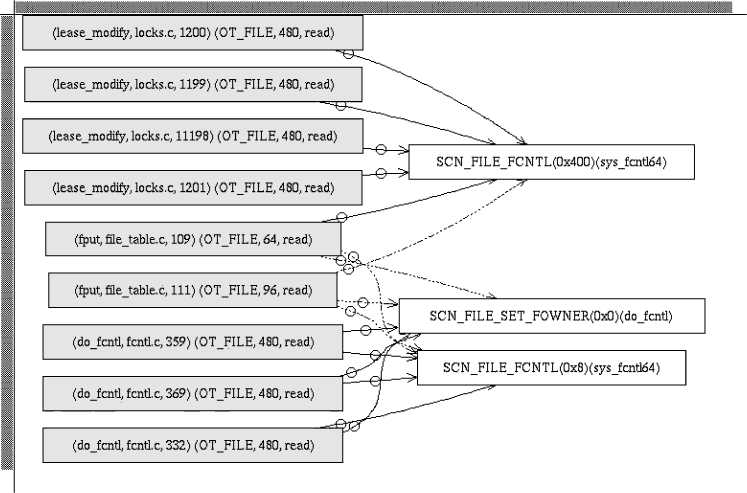


图5曲线图授权为呼叫的fcntl关于^ SETLEASE（控制操作中的LEA ^修改和fput）和^ SETOWN（在d ^的fcntl控制操作和放）。当命令是^ SETOWN既FCNTL和SET\_0WNER被授权，但只能FCNTL被授权F\_SETLEASE。

（2）的fcntl（FD，F\_SETLEASE，F\_UNLCK）。受控的操作节点包括位置（功能名，文件名，行号）和操作（数据类型，构件偏移，操作类型）的信息。授权节点包括所述授权，命令和功能包含授权。­­­总是 边缘由实线表示，并 有时边缘由虚线表示。如果控制操作和授权之间不存在边缘，则该授权从不该操作进行。

通过视觉分析这个图我们可以找出是否在3.3节保持当前图形或没有描述的变量。在这种情况下，有时fput及其授权之间的关系表示可能存在问题。同样，不同组的授权为同一领域取得的事实（构件偏移480，其恰好是f\_owner）可以指示有问题的。然后手动调查需要识别任何不一致是否是由于错误或合法的一致性。­­

1. 稠度等级列表。一致性类列出了一致性水平，并在这些级别的授权要求显示控制操作的分区。这个分区使用在3.3节中描述的算法来计算。一致性类列表提供比相同的授权结果的授权图形的不同视图。而授权曲线图示出每个单独的控制操作和授权之间的关系，所述稠度等级列表显示具有相同的授权要求来控制操作的集合。一致性类名单­­­

DFN d 0 FILE f\_dentry -1 DFN d 0 FILE Ldentry 1 DFN d 0 FILE f.vfsmnt -1 DFN d 0 FILE罗布-1

SFN（ALWAYS）d 0 FILE.READ

DFN d 1超级块s.blocksize -1 DFN d 1超级块S\_TYPE -1

DFN d 1 TASK状态-1 DFN d 1个TASK状态0 DFN d 1任务标志-1

（SFN）NONE

DFNØ0 INODE i.blocks -1 DFNØ0 INODE i.blocks 1 DFNØ0 INODE i\_version -1

SFN（ALWAYS）O 0 FILE-READ

DFN·1 INODE i\_dnotify\_mask -1 SFN（）NONE

图6.一致性类列表用于读系统调用具有以下字段：（1）入口类型（DFN或SFN）; （2）稠度（d为数据类型以及o为对象）; （3）类号码; （4）数据类型; （5）部件; （6）访问标识符。

使得中存在的数据不同的授权的情况下更加明显的数量。此外，由于它们是文本[Jaeger等人的一致性类列表是更容易回归测试使用。2002]。

图6示出了控制操作的分割成一致性类用于读系统调用。这个分区被用作在第3.3节的实施例。与那里描述，一致性类列表显示两个类的数据类型水平相一致：一个任务和超级块没有授权和一个用于读取授权文件。然后，一致性类列表中有两个类对象是一致的：一个用于读取被授权的inode，一个用于它的目录不具有授权。最终，我们希望注释当前任务，文件所在的目录，并作为读取授权，这将导致在具有相同的授权控制所有操作文件的超级块（即，被系统调用一致）。

我们的大多数的经验是与文件系统，虽然我们还考察任务授权。大多数对象有一个或没有授权，所以一致性类名单是不是太复杂。该系统调用取消链接是一个对象可以多次授权为数不多。使用一致性类名单不难看出，目录inode有三个授权（EXEC，写unlink\_dir），并移除了i节点，一个（unlinkfile），因为它们是对象的一致性，并放置在不同的班级。因此，对于文件系统，我们已经检查了任务操作，授权图表和一致性类名单已足以验证授权。

* fcntl函数的F-SETOWN只有场f\_owner

1. =（+，ID\_TYPE，CONTEXT）（+，di\_cfm\_eax，的fcntl）（+，co\_ecx，F\_SETOWN）
2. （d，1）=（+，ID\_TYPE，SEC-CHK）
3. （d，1）=（+，ID\_TYPE，CNTL\_OP）（+，di\_dfm\_member，f.owner）

* 的fcntl适用于F ^ SETLEASE只有场f.owner

1. =（+，ID\_TYPE，CONTEXT）（+，二cfnueax，的fcntl）

（+，co\_ecx，F\_SETLEASE）（+，co\_edx，F\_UNLCK）

1. （d，4）=（+，ID\_TYPE，SEC\_CHK）
2. （d，4）= C +，ID\_TYPE＃CNTL\_OP）（+，di\_dfm\_member，F-所有者）

图7.规则异常发现歼所有者。

1. 样本分析。我们简要地表明了，我们发现了一个异常的样品分析。虽然寻找异常的方法是同时开发的真正找到异常情况，我们粗略地使用相同的方法所描述的，虽然有些是不是自动的。这种异常发生在的fcntl 系统调用。一致性类列表的fcntl 显示，其授权的系统调用输入一致。的值­CMD 和 ARG 参数 的fcntl 可以改变了所需的授权。我们使用授权图表看授权下不同的输入，因为它更容易看到粗粒度的问题，很多的有时 边缘发生。

图7包含两套规则：（1）一个用于收集所述文件结构的所有授权字段和控制操作 ˚F老板 在一个 的fcntl（FD，F-SETOWN，pid\_owner） 系统调用，和（2）一种用于收集在球场上的所有授权和控制的操作 ­f\_owner 在一个 的fcntl（FD，F\_SETLEASE，F\_UNLCK） 系统调用。请注意，这是相同的规则（少fput 控制操作）用来产生图5中的曲线图。­

在图5中，我们看到，一些控制操作的被授权为 的fcntl 和 set\_fowner 授权和一些只授权 的fcntl。尽管这是一个事实，即控制操作访问相同的领域，f\_owner （偏移480）。鉴于这种异常，我们研究内核源代码，以确定是否一个开发的这种异常是可能的。我们讨论这个分析在下一节的结果。

* 1. 结果

我们采用2001年12月10日LSM补丁到Linux 2.4.16源和使用我们的GCC-3.0的修改版本编译的内核。[[3]](#footnote-4) 要创建执行日志分析，我们得到的LMBench，圣漏洞工具（并行三个实例执行­[www.wwdsi.com/saint/](http://www.wwdsi.com/saint/)），内核编译，而我们写在我们开始怀疑异常的一些测试程序。由于运行时间分析依赖于运行足够的代码有效性，覆盖有趣的路径足够基准的发展，必须发展。例如，仅的LMBench运行在内核代码的约20％。­

我们已经仪表内核收集的主要内核数据结构来控制操作：文件，索引节点，超级块，任务，插座和skbuffs。到目前为止，我们只是做上的文件系统权限进行详细的分析，并在任务授权进行了初步分析。由于文件系统是相当好理解，我们没有想到的大量的异常的，但我们发现了一些仍然。

- 会员一致（多系统调用）： setgr〇ups16是几个ABI向后兼容的16位任务操作，如setuid16和setchown16之一。这些操作通常是他们的16位值转换为32位值，并呼吁那些包含授权的当前版本。然而，由于setgroups16设置一个数组，这是不容易的数组转换，所以目前版本（包含钩）不叫。我们发现，在功能setgroups16没有授权挂钩，所以我们可以重置任务的组集合未经授权。请注意，是在libc中的当前版本没有setgroups16调用，所以我们不得不编写一个汇编程序来执行这个漏洞。­­

- 会员一致（单系统调用）： 结构文件的f\_owner.pid人员告诉进程发送信号给这个文件关于IO其内核。设置此字段由file\_ops-> set\_fowner如果用户试图直接通过的fcntl（FD，F\_SETOWN，pid\_owner）设置它的授权。然而，如果用户移除经由的fcntl从文件租赁（FD，F\_SETLEASE，F\_UNLCK），所有者withoutthe authorizationbeingperformed设置为零。此外，进程可以设置一个通用TUN设备（驱动器/净/ tun.c）到其自身的所有者，而不进行授权。为了实现这一点，该过程调用的ioctl（FD，F\_SETFL，FASYNC）的开放，附接，TUN设备上。

-Member一致（单系统调用）： 在我们上述filp.f\_owner的一致性的调查，我们发现，获得的filp。f\_owner.signum（应该IO完成时发送的信号）可以不经由的fcntl授权（FD，F\_SETSIG，SIG）被设置。

- 系统调用一致（多系统调用）： 读授权在每个read系统调用的开始执行。因为当文件最初打开可能不再是有效的，由于工艺改变其安全级别，改变其安全级别的文件，正在由一个新的进程的文件，或者在一个变化的授权进行这种授权是必需的安全策略。此授权，但是，是不是在内存映射文件中的页面出错时进行。因此，一旦某个进程的内存映射文件可以继续读文件，而不管更改安全级别或安全策略。

公司主要从事与LSM社区造成了补丁的所有异常的讨论，除了一个用于读取内存映射文件。社区决定需要读权限的文件不能是内存映射。我们感到鼓舞的是，我们已经能够帮助查找和解决钩安置问题。

1. EXAMINING STATICANALYSIS

现在，我们已经实现并采用雅巴静态分析工具收集的在Linux内核相关的执行路径记录静态分析的原型。对于静态分析，完整的内核执行路径覆盖范围可以更直接地评估比运行时分析。不幸的是，内核执行路径的数目在一般指数，所以该模型必须被设计为除去尽可能多的不相关的路径尽可能和被优化，以便许多路径可以被评定为可能的。静态生成的日志可以被输入到在第4.2节作为一个运行时生成的日志将所描述的分析工具。­­­

* 1. JaBAAnalysisTool

JABA与指针 - 基于数据流分析[Koved等人的流动敏感，上下文敏感的静态分析工具。2002]。JABA具有处理的Java类的字节码来构建过程内的基本块的图，过程间控制流程的图表，以及用于变量和对象字段的数据流图的基函数。我们建立了使用此基础分析信息作为输入日志收集进一步的静态分析。这些分析是程序性的，而不是在声明性分析语言，如MC [Engler等人。2000。如第4.2节我们再处理这些日志。­

在JABA上下文敏感控制流程意味着，不同的路径进入相同功能可以彼此，使得不同的输入对象到相同的功能可以区分和它们授权的关系能够精确地跟踪被区分。

在雅巴上下文敏感的数据流量跟踪从程序的分配点的数据流。对于Linux内核，同一分配点用于大多数对象（如文件和inode），所以在上下文不敏感的数据流，相同的对象会在该类型的所有操作使用。JABA通过调用点到创建该对象（或任意水平）的函数区分对象使上下文灵敏度的定义。创建的多个对象，越贵的分析却变。

由于Linux内核显然不是在Java中，我们编写了从C到Java字节代码转换[Zhang等人。改造仅保留那些相关的雅巴分析了Linux内核的语义。例如，指针C结构被翻译为含有相称类型的Java对象大小1的阵列。此外，无类型安全的指针操作被转换为对象wildPtr。­­

* 1. 日志生成过程

所述JABA日志分析过程使用两个输入：（1）根控制流图的节点，通常为一个系统调用的功能（例如，SY ^打开）和（2）中的数据类型，其控制操作被分析的结构。所有JABA分析仅限于那些从根节点到达的功能（即，CFG节点）。日志生成返回日志由运行时间数据收集工具收集的形式。

使用雅巴日志生成过程如下：

1. 转储ID命名映射功能，授权，并通过日志分析工具的使用来控制操作。
2. 收集指定的数据类型的对象的所有控制操作。回想一下，类型被指定为输入到分析处理和集合的范围，这些数据是通过从根到达的功能的限制。
3. 收集指定的数据类型的对象的所有授权。
4. 定义每个路径的一致性方面，当前定义为在一个系统调用中使用的每个个体对象实例。此上下文中也可包括条件分支限制模拟由于系统调用的标志值的开关。­
5. 识别节点与导致任一受控的操作或授权路径。所有其他节点无关的分析。
6. 产生的所有相关过程内路径（见下文）。
7. 从生成过程内的路径序列过程间的路径。
8. 从第4步重复，直到没有新的环境审查。

第一步使日志分析工具通过从数字表示（例如，函数CFG节点号码），姓名（例如，函数名）提供映射正常显示从JABA-生成的日志的结果。­

如在运行时的情况下完成的第二步收集的数据类型感兴趣的所有控制操作。在这个证明了概念，我们只收集写访问迄今。读的收集访问需要找到一些错误（例如，从读F\_­〇wner字段更新其成员），那么扩展读访问是必要的。虽然增加更多的访问只会增加分析的复杂性，我们不认为在第6章中讨论的额外的复杂性将是显著。­

第三步收集的数据类型感兴趣的授权。在系统运行时，我们不得不手动列出被认为已授权各LSM钩子的所有对象。在这里，我们都希望能使用的数据流以更精确地描述为在第6节然而，目前，只有直接授权被收集讨论被授权的对象。我们定义一种机制，授权推断授权的对象，但是这一步是目前比在运行时分析可大可小。

第四步定义了一组分析上下文。在运行时间工具，系统调用的执行限定了上下文。在静态分析，系统调用定义了许多道路，所以我们需要确定的路径上下文是值得收藏的。我们已经确定了一些方法来减少值得考虑（见5.3）的过程内路径的数量，但我们仍然发现，进一步降低是必要的及时的分析。我们效仿xgcc项目级别，分析自动机处理是可变依赖性[Hallem等。2002]。

由于雅巴跟踪对象，而不是变量，对象定义上下文。也就是说，每个系统调用，一个物体被识别，只有在该对象上的操作和授权定义的上下文。之后该对象的路径分析完成的系统调用另一个对象选择和所有相关的上下文（即路径），该对象被记录。这重复进行，直到所有相关对象（即，目标数据类型的对象）在每个系统调用进行了分析和他们的日志收集。­

第五步骤在所述上下文描述，并确定是值得分析的CFG节点。节点是值得分析如果其除去，从而除去任何其后代将导致受控操作或授权日志条目的损失。因此，调用具有受控的相对于当前上下文对象的操作或授权的任何功能的功能是值得记录的。

第六步创建的节点值得分析的所有相关过程内路径。这是在分析的关键步骤，并且其详细描述在第5.3节中讨论。

第七步结合过程内的路径成单独的日志。从逻辑上讲，它列举ofintraprocedural在执行顺序记录所有组合。在实际中，日志生成被优化，以从提供新过程内路径的下一个组合，而不是从头开始构建每个路径中的功能建立新的日志。日志由收集的过程内路径序列为那些会在这些路径中调用的函数建。在接下来的通过，即具有多个过程内路径的日志中的最后一个函数被选择为定义下一个组合。日志到这一点是一样的，所以日志生成由新路径的开始只是做了。­

在第4.2节中描述的日志分析如之前在步骤（7）中生成的日志进行。在日志条目的唯一区别是：（1）JABA对象标识符用于对象，而不是特定于对象的标识符作为在运行时和（2）原语参数不被JABA捕获定义的，所以系统调用标记必须表示由我们在目前的前四个条件句的系统调用的参数的位置存储条件分支标识符。

* 1. IntraproceduralAnalysis

过程内分析定义每个节点相关的路径的数量。由于过程内路径组合在普通指数，应采取一切合理措施以保持尽可能小的路径数。我们同时执行节点级和有条件级的分析，以尽量消除冗余路径。其结果是，只有一个在Linux文件系统功能需要三个以上的路径，大部分都只有一条路径。

我们正在寻找的主要事情是控制操作和权限之间的关系。因此，如果在相同的关系的节点结果的所有路径，则在该节点的所有路径可以被组合成一个。因为授权的数量比控制操作的数目小得多，我们取一个授权中心的观点：如果在一个节点的所有路径具有相同的授权（即，无引线为上下文对象的新的授权），然后所有的控制操作可以组合成一个路径。由于授权控制的操作之前，通常会出现，这种优化是非常有效的。

在功能不同的路径是由它的条件语句创建。的条件规定，任一两个或更多个路径的一个可以采取，因此每个选择形成的路径。如果路径的选择不影响控制操作与授权之间的关系，那么这些条件路径不必单独考虑。我们称这个过程合并。 在大多数功能，这导致在路径数一个显著减少。对于过程内分析中，我们执行下列步骤：

⑴ 排序在拓扑顺序节点与CFG离开第一。

⑵确定节点调用有任何授权的节点。如果没有，压缩控制操作一个路径。

1. 收集节点的基本块到一个路径。如果找到一个条件语句，确定此有条件的可以“合并”。如果不是，选择第一路径，然后继续。推栈后续路径生成的条件。­
2. 在过程间路径生成路径存储函数调用的位置。
3. 当一个过程内路径是否完整，快退条件栈，直到条件被发现有树枝留下来检验。生成从该位置的下一个过程内的日志。

在Linux虚拟文件系统的唯一功能造成的困难是path\_walk，其中有28条相关的路径。首先，这是有许多可能的路径很长且复杂的功能。然而，我们认为，这个函数调用，实际上是指从我们的角度不同语义对象的子程序，所以雅巴数据流分析不能捕获正确的语义呢。许多子程序真的不应该是相关的分析。无论如何，的path\_walk路径的生成需要3分钟，和的过程间路径，系统生成调用使用path\_walk（有很多），还是相当上的秒的数量级，因为大多数其他节点具有1或2个路径。­­

总的处理时间的系统调用在Linux虚拟文件系统，大约需要1小时，这实际上类似于它需要放在一起运行时分析的时间。但是，我们乐观地认为，静态分析性能可以显著改善。目前，我们的雅巴系统的架构方式，我们必须重新计算path\_walk为每个系统调用中使用它。除去该多余的计算将通过略微超过50％减少分析时间。另外，我们可以通过启动该节点的分析进一步提高性能的对象是第一次看到，而不是在系统调用。

* 1. 静态一致性结果

我们的目标是使用雅巴一致性分析，发现无论是\_文件〇在第4.3节和TOCTTOU漏洞[Bishop和1996迪尔格]，我们使用静态分析工具CQUAL [Zhang等人先前发现描述PS-> set\_fowner错误。2002]。两者都可以被归类为一致性错误，但后者是很难找到一个运行时分析，因为它需要一个活动成功的攻击造成的不一致。如果我们已经知道的漏洞，我们就不需要工具。

回想一下，file\_ops-> set\_fowner错误被造成的，因为在该领域（读取在这种情况下操作）的受控操作由对于除lease\_modify所有功能的set\_fowner授权。静态日志包括fcntl\_setlease项，这表明，在某些日志set\_fowner授权并在某些日志事实并非如此。有趣的是，日志不会收集日志条目lease\_modify，因为它总是与相同的授权调用，只有的fcntl。因为关系是固定的，我们不需要考虑功能区内的路径。不过，我们也必须考虑到一个特定的控制操作可以使用可识别的一致性问题，比如这里的情况。在这种情况下，我们想知道的是，日志可以调用lease\_modify等功能，并执行其控制操作。由于控制运营和授权之间的关系是静态的，我们可以聚集所有的控制操作（即，将它们合并成一个总的路径）。这样做将使瓦利分析工具，以确定缺乏lease\_modify授权的。­­­­

该TOCTTOU漏洞Zhang等人发现。[2002]发生，因为文件指针被授权sys\_fcntl，而是一个新的文件指针是基于fcntl\_getlk用户提供的文件描述符提取。由于用户可以控制文件描述符和它引用的文件之间的映射，竞争条件导致攻击者可以利用来进行未经授权的fcntl操作。JABA提供跟踪程序中的变量和字段的可能值的数据流图。在这种情况下，我们需要根据其来源来区分文件指针变量（即从文件描述符fget）。生成的静态记录表明，文件指针对象是sys\_fcntl被授权在所有情况下，而不同的文件指针对象在被从未授权fcntl\_getlk使用。­

因此，我们建立在雅巴分析工具能够分析的建设，可以实现比运行时工具，它可以覆盖我们以前的静态分析，以及更广泛的一致性分析。

1. 讨论

在本节中，我们简要分析一下在使用LSM一致性分析的三个问题：（1）静态分析的有效性; （2）LSM钩放置的验证; 和（3）用于回归测试。­

6.1静态分析的问题

在一般情况下，我们发现，我们可以使用通过两种运行时或静态方法收集的日志数据所描述的一致性分析。目前proof-概念的静态分析日志收集大约需要同样多的时间，构建和执行运行时的日志收集。可以使用任何一种工具来执行相同的日志分析。

静态分析具有路径覆盖和分析的准确性的优点。首先，它是很难找到执行所有相关的路径基准。基本上，代码的静态分析是需要写入必要的基准，以执行正确的路径。其次，以前的CQual静态分析表明，TOCTTOU错误可以发现，无法找到­­­

使用运行时分析。查找使用运行时分析这些错误的要求有一个主动的攻击者进行攻击TOCTTOU，所以我们已经知道攻击的存在。

有效利用静态分析依赖于处理所有必要的控制操作（即，将读访问）和使用数据流量来识别授权的对象。读访问扩展分析会延长日志的长度，并可能增加相关路径的数量。相关的路径取决于控制操作和授权之间的新关系，但由于每个系统调用的授权的数量较少，我们不希望这是一个问题。我们可以进一步细化上下文的概念，以进一步限制日志生成的范围。

静态分析的一个重要用途是推断通过LSM钩子授权的对象。该LSM钩子没有明确描述他们授权的对象，因此必须确定。使用手动指定的运行时分析这样的特性有，但推断静态分析可以使得这样的性质。在一般情况下，传递到挂钩对象的授权。另外，也可以认为是被授权的对象通过固定字段引用的对象进行授权。这些场是否是不可改变的取决于是否有一个系统调用，可以得到一个别名相同的对象，并且可以修改字段。创建对象一般要求不符合，因为他们创造一个新的对象，所以他们不能修改现有对象。虽然文件的文件描述符关系可以通过系统调用进行修改，文件对象和其索引节点之间的关系一旦创建了文件对象很可能不会被修改。我们正在研究的数据流分析，使这样的性能进行验证。­

1. 验证

大多数分析工作至今的目的是错误的发现。尽管一些分析方法是完整的（即没有出现假阴性，如MOPS [陈和2002年瓦格纳]），这些工具还没有被用来证明任何显著，哪怕是轻微的安全性能。有了足够的分析支持，我们希望能够真正验证系统的安全性。我们审视我们在实现这一目标的地位。­

我们的分析使验证，即一个特定的一组 中介业务­ （例如，LSM钩授权）被一致地执行当一组的 使用操作（例如，内核控制操作）被执行。如果中介操作可以在事先确定（即，我们可以找出所需的授权）就可以（通过对象和操作如）涉及到使用操作，那么这个分析完成。

在LSM钩分析，事先未指定用于控制操作的授权，因此一致性分析所述来确定所需操作。有三个问题，使这个假设diffi-邪教：⑴ 有可能是LSM的钩子只是完全缺失，因此不存在不一致的问题;⑵上下文描述可能不包括所有相关路径; 和（3）的设定由LSM钩标识的对象可以不包括被默许某些对象。在第一种情况下，如果是一个钩子在任何情况下不丢失不一致性问题。这个想法是，授权控制的操作关系的聚集使这个手动验证。然而，在人工验证错误的可能性。在第二种情况下，上下文规范必须保证一致性分析涵盖的相关日志路径。为了实现这一目标，每个控制操作必须属于一个范围内声明的授权关系是一致的，对失踪挂钩验证。三种类型的上下文的简单性（即召回系统调用，系统调用的标志，和操作集），使这个覆盖面相当方便LSM钩子。在第三种情况下，LSM钩子都没有明确申明被授权的对象。正如上一节中所讨论的，我们的目标是使用JABA数据流分析，以验证该属性。

因此，如果我们能保证所有的内核路径，这种影响的一致性，关联与他们的授权介导的所有对象的全覆盖，并可靠地验证授权要求，那么我们就可以进行有效的验证。我们预计，我们可以验证，如对象标识（使用前标签），对象初始化，审计和对象重用性。例如，审计业务应与安全决策配对。这些都是很容易识别，所以这个验证应该是简单的推断，虽然受影响的对象依然是一个问题。对于标签，我们可以识别标签的操作，并确定它们是否对前安全检查的所有对象运行。然而，对象可以被重新标记。确定是否发生重新贴在必要时需要重新标记定义的要求。那是，的中介操作是指示重新标签和表示使用一个重新标记对象的受控操作必须被确定。因此，初始标记的验证是可能的，但正确的重新贴标签的验证似乎是困难得多。­­­­­­

1. 回归

一旦安全性能得到验证，这是必要的，以确保当系统更新此属性保持不变。该过程通常被称为回归测试。由于更新可以通过很多人在一个开源的系统，如Linux进行，有必要对大多数回归比原来的验证简单得多。特别是，我们想最大限度地发挥一个简单的变化将导致回归的数据没有手动检查的可能性。

对于LSM钩验证，使用稠度类提供针对改变一些弹性。只要一致性类列表不改变，那么输出基本上是相同的。由于一致性的变化是指示性的安置问题，在回归测试一致的一致，表明有效安置。为了让这个假设持有，我们还必须考虑的事实是必要的挂钩可以在任何情况下丢失。因此，只要没有新的控制操作，则此回归方法的工作原理。对于增加新的控制操作，需要做更多的工作。­­

考试在国税发其控制的操作似乎不管是良性还是一个潜在的问题其他背景。一般而言，较大的不同集合的授权给该受控操作在一致性列表相关联的数是从特定的授权其独立的一个量度。这种情况将被视为­低对于需要一个新的LSM钩子。在另一方面，如果控制操作是一组定义的上下文控制操作的一部分，则有必要对一个LSM钩的可能性要高得多。我们可以汇总基于这些种类的标准来指示需要的严重性更深审核新的控制操作。

1. 结论

在本文中，我们提出了一致性分析核实LSM框架的正确性帮助Linux社区。LSM框架由一组放置在内核内部授权挂钩的，所以它更难以识别完整的调解点。我们利用的事实，大多数LSM钩子得到妥善安置，以确定错放钩子。我们使用对主要内核数据结构作为中介界面结构件业务，并收集这些操作的授权。通过分析运行时测井仪的输出，我们发现它的授权是不一致的操作。我们分析了文件系统和一些任务操作，发现可能被利用一些异常。与LSM社区工作，这些问题已经被修复。例如，­­

最终，我们发现一致性分析验证，其中从规范不一致的一个可能是错误的系统非常有用。然而，运行时间日志收集在路径覆盖的局限性和会错过一些重要的错误，比如TOCTTOU，依赖于异常的输入数据或上下文。我们已经发现，这种分析方法也可以应用于通过静态分析收集的执行日志。我们演示了如何使用雅巴分析工具收集这些日志。此外，我们还讨论了改进，可以使用雅巴的数据流分析进行了全面的分析。我们正在积极寻求完整的，易于使用的验证的LSM授权钩子和检查等安全性能的验证。­­­

参考

一个nderson，JP 1972。 计算机安全技术规划研究。技术。众议员ESD-TR-73-51，空军电子系统部。

一个shcraft，K. 和 Ë格勒，2002年D.使用程序员编写编译器扩展赶安全漏洞。在­诉讼o ^的IEEE研讨会Ø〜n的安全性和保密性。

乙所有，T，COOK，B.，L埃文，V， 和 [Rajamani，SK 2003 SLAM和静态驱动程序验证：内微软形式化方法技术转让。在集成的形式化方法，讲座在计算机科学，卷注释。2999 1-20。也出现在MSR-TR-2004-8。

d如，M。，Lerner，S.， 和 小号eigle，2002年M. ESP：在多项式时间的路径敏感的程序验证。在ACM SIGPLAN会议上编程语言的设计与实现（PLDI'02）。

乙ishop，M. 和 dilger，M. 1996.检查竞态条件的文件访问。 ComputingSystems，9， 2，131-152。

C母鸡，H.和瓦格纳，2002年D. MOPS：用于检查的安全性的基础设施 软件。在计算机和通信安全第九届会议论文集。C母鸡，H.，院长，博士，和瓦格纳，2004年D.检查型的C代码百万行。在网络和DIS ^肋骨论文集〜^^编系统安全研讨会（^^^ DSS 2004）。­

CHAKI，S.，Clarke，E.，GROCE，A，J哈，S.， 和 Veith在C中的软件组件，H. 2003模块化验证 诉讼o ^ 25〜届软件工程国际会议（ICSE2003）。 385-395。

Ëdwards，A.，积家，T.，和张，X. 2001年。 授权钩放置在Linux安全模块框架的运行验证。­技术。众议员RC22254，IBM研究。

Ë格勒，D，CHELF，B.，C侯， 一个。， 和 Hallem，S. 2000使用系统特定，程序员编写的编译器扩展检查系统规则。在诉讼o ^的佛^^^ ^ h研讨会操作系统设计与实现（OSDI）。

Hallem，S.，CHELF，B.，X即，Y.， 和 Ë格勒，2002年D.系统和语言构建系统特定，静态分析（出现在2002年PLDI）。在ACM SI ^计划会议PROG ^ ^早上我^ g语言设计与实现（PLDI '02）。

F奥斯特，J.，Fahndrich，M.，和艾肯，A. 1999年类型限定符的理论。在ACM SIGPLAN会议上编程语言的设计与实现（PLDI '99）。192-203。

Ganapathy，五，J哈，S.，C处理器，D.，男elski，D.， 和 V伊泰克，D. 2003缓冲区溢出检测使用线性规划和静态分析。在诉讼O〜f。关于计算机与通信安全的10 ^ ^ h ACM会议。

Gutmann，P. 2000。 设计密码安全体系的验证。Suh-mitted论文。可在[www.cs.auckland.ac.nz/pgut001/pubs/thesis.html](http://www.cs.auckland.ac.nz/pgut001/pubs/thesis.html)。

一世台湾证券交易所。1998年。信息安全技术〜nology E共有标准*^*人*^*ATIO*〜*ñ。ITSEC。可在[www.commoncriteria.org](http://www.commoncriteria.org)。

Ĵaeger，T，Z挂， X。， 和 Ëdwards，2002年A.保持正确的Linux安全模块框架。在­诉讼O〜fthe 2002Ø^^ AWA的Linux研讨会。 223-241。

大号纵火，E. 和 一个ustin，2003吨高覆盖侦测输入相关的安全故障。在程序o ^ 12〜第USE ^ NI ^ XSecurity研讨会。­

ķoutsofios，E. 和 ñ奥尔特，S.绘图的曲线图与点。可在[HTTP：//www.research](http://www.research)。att.com/sw/tools/graphviz/。

ķOved的，L.，皮斯托亚，M.，和Kerschenbaum，2002年A.对Java的访问权限分析。在第17届ACM会议邻论文集〜n的面向对象P'rog'ram'mi'ng系统，拉〜NG ^ a ^ ES和应用程序（OOPSLA）­。

大号arochelle，D. 和 Ë面包车，D. 2001.静态检测可能缓冲区溢出的安全漏洞。在第十届USENIX安全研讨会论文集。177-190。

ñ中信建投。1985年可信计算机安全E 1人^ ^ ATIO标准。国家计算机安全中心。国防部5200.28-STD，又称橙皮书。

小号行家，U。，T艾瓦，K，F奥斯特，JS， 和 w ^瓦格纳，2001年D.检测格式化字符串漏洞与类型限定。在­诉讼o ^碲〜^^ ^ h乌塞尼^ X安全研讨会。 201-216。

Viega，J。，B湖，J。，K不好了，Y.， 和 中号CG生的，G. 2000 ITS4：静态漏洞扫描器为C和C ++代码。在诉讼of2000年度安全应用学术会议。

w ^瓦格纳，D，F奥斯特，JS，Brewer，EA， 和 一个iken，A. 2000朝向缓冲区溢出漏洞的自动化检测的第一步骤。在网络的程序和分布式系统安全研讨会（NDSS 2000）。

ž挂，X.，Edwards， 一个。， 和 Ĵaeger，2002年T.使用CQual授权钩子放置的静态分析。在诉讼O〜fthe 11乌塞尼^ XSecurity研讨会。

ž挂，X.，男爱马仕Arceau，G.，Jaeger，T.， 和 ķOved的，由雅巴框架静态分析C程序的L.翻译。内部文件草案，成为IBM的技术报告。

收到2003年7月; 修订后的2004年3月; 接受2004年3月

1. 其实，这个对象也应该读LSM钩授权，所以我们把它添加到组通过此挂钩授权的对象。 [↑](#footnote-ref-2)
2. 这些是COMPONENTREF节点，其中所述第一操作数的所得类型是介导的类型。 [↑](#footnote-ref-3)
3. 与内核版本跟上不工作的很大。我们在Linux 2.4.18现在运行的系统，我们不得不做的唯一的事情就是更新我们的授权过滤器，以当前的LSM接口。然而，LSM界面被重新设计显著作为其列入主要的Linux内核的一部分，所以我们还没有更新的本次改版。它不应该超过一天来更新。 [↑](#footnote-ref-4)