

带敏感标签的 SELinux 安全策略信息流分析方法

张 阳

(中国科学院软件研究所信息安全国家重点实验室 北京 100190)

摘 要 针对 SELinux 操作系统中多安全策略的实现方式,文中在信息流分析方法的基础上引入了多级安全敏感标签,以自动机与线性时态逻辑为理论基础,提出了一种改进的信息流分析方法,对 SELinux 安全策略的完整性与机密性进行验证.

关键词 安全操作系统; SELinux; 安全策略; 信息流; 有限状态自动机

中图法分类号 TP309 **DOI 号**: 10.3724/SP.J.1016.2009.00709

A Information-Flow-Based Verification Solution with Security Sensitivity to Check Security Policy of SELinux

ZHANG Yang

(State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100190)

Abstract According to the implementation of multi-policy in SELinux, the authors introduce the multi-level sensitivity label into the traditional information flow analysis. On the basis of automata and linear temporal logic, an improved information flow analysis is proposed. It can verify both the integrity and confidentiality of the SELinux policies.

Keywords secure operating system; SELinux; secure policy; information flow; finite state machine

1 引 言

2001年2月,美国国家安全局发布了一个安全增强的 Linux 操作系统——SELinux,由于 SELinux 操作系统的策略配置方案十分灵活,并且实现了强制访问控制,所以该系统一经问世,即得到安全操作系统开发者、使用者以及研究者的广泛关注.目前,SELinux 已经进入了 Linux 官方内核,并且进入了 Red Hat 等 Linux 主流发行版本.

随着 SELinux 的不断推广,SELinux 的安全问题也引起人们的广泛关注.实验表明,SELinux 的运行开销很低^[1],但安全策略的管理和开发十分困难,同时验证所实现的安全策略能否实现其安全目标也

非常困难.在这种情况下,如何实现自动或半自动的针对 SELinux 系统安全策略配置的分析检测成为研究界的一个关注的重点.

到目前为止,研究界对 SELinux 系统安全策略的分析所采用的方法可大致分为访问控制类方法和信息流分析类方法.前者在面对安全分析中常常遇到的信息机密性和完整性的保护的问题时,往往需要分析者对涉及的系统用户、敏感资源以及辅助工具的访问控制特性进行完全的刻画.这样,就导致了整个分析过程繁杂且不直观.而对于现有的信息流分析类方法而言,其适用性大多局限于小规模较为简单的安全模型.特别是考虑到 SELinux 系统能够实现多等级多安全策略配置时,此类方法往往束手无策.

有鉴于此,本文提出一套通过在描述系统安全状态刻画时引入敏感级标签概念来为安全策略建立严格对应的有限状态自动机模型的方法,同时,还对 SELinux 规范中提及的常用安全目标进行分析归纳,并对这些安全目标进行形式化刻画.通过这两种方法,分析者能够在对安全策略形式化建模从而对该策略获得比通过以往方法更深刻认识的同时,亦能借用模型检测工具对安全策略是否实施了指定安全目标作数学意义上严格的断言.为验证本文提出方法的正确性和有效性,作者选取了两个安全策略实例进行实验分析.实验结果表明,本文中的安全策略形式化建模方法能够有效地应对实际系统中出现的敏感信息访问控制和保护机制,同时安全目标的逻辑公式化也使得分析者能够对安全策略的有效性和一致性做出可信的判断.同时,当安全策略模型违反安全目标时,模型检测技术能够提供直观反例,为分析者修正安全策略提供有效依据.

本文第 2 节介绍 SELinux 系统的基本概念以及其中安全策略声明以及强制实施的基本机制;第 3 节总结现有研究方法在分析 SELinux 系统安全策略应用上的不足,正是这些不足引发了本文描述的分析方法的提出;在第 4 节中,作者从安全策略形式化建模以及安全目标的逻辑刻画两个方面全面描述一套 SELinux 系统安全策略分析框架,并通过第 5 节的实验证明该框架的有效性;最后,第 6 节总结全文并指出未来可能的研究方向.

2 SELinux 系统及其安全策略

概而言之,SELinux 系统通过在系统启动时载入并在系统运行中严格执行用户预先定义的安全策略来实现高等级的系统安全.为支持用户灵活有效的配置系统安全策略,SELinux 系统定义一套接近于常用高等级编程语言的安全策略语言^[2].用户按照该语言的规范编写安全策略“程序”,并提交系统编译后形成二进制文件格式的策略配置,以指导系统在运行中动态地检查目标动作是否符合用户的安全设定.

SELinux 系统提供给用户的安全策略描述语言主要由以下 4 种类型语句构成:

(1) 声明语句. 包括主体 (subject)、客体 (object)、角色 (role) 以及类型 (type) 声明;

(2) 访问向量规则及转换规则语句. 其中前者

定义了安全服务器对访问请求进行响应的依据,而后者管理可能的对象角色转换和为新产生的对象指派类型;

(3) 约束语句. 用来定义主体对不同客体的合法的访问方式;

(4) 安全断言语句规定了系统运行时不仅需要符合用户定义的其他安全策略,同时亦需保证所有的安全断言计值为真.因此,SELinux 系统将动态地检查任何违反安全断言的运行,并向用户报错.

在实际系统中,为实现严格完备的安全机制,用户定义的安全策略程序通常规模较大.例如,SELinux 系统发布的安全策略范例涵盖了 3 个角色、28 个客体类、22 个属性、115 种权限和 253 个型以及多达二十万条的策略描述语句^[1].

3 现有相关工作与比较

在实际系统中,安全策略文件的规模通常较为庞大,从而导致手工检测这些文件所定义的安全机制的完备性和一致性极为困难.在这种情况下,如何实现自动或半自动的针对 SELinux 系统安全策略配置的分析检测成为研究界的一个关注的重点.随着近年来在此方面研究的深入,出现了多种在刻画和分析能力上各有不同的分析模型和工具.而这些研究成果亦因为研究对象的侧重不同,在实际分析任务中,具有不同的约束和缺陷.

针对 SELinux 系统安全策略的分析方法,根据其侧重的分析对象的不同,可大致分为访问控制 (access control) 类方法和信息流分析 (information flow analysis) 类方法.实现前者的典型性工具包括 IBM T. J. Watson 研究中心提出的 Gokyo 工具^[3-4]、Tresys 公司的 SETools 分析工具包^①以及由 Zanin 等人开发的 SELAC 分析模型^[5]等.而后的代表包括 Guttman 等实现的 SLAT 工具^{[6]②}和 Sarna-Starosta 等人提出的 PAL 工具^[7]等.

3.1 访问控制分析方法

2003 年,Jaeger 等首先提出访问控制空间 (access control space) 的概念^[3],并将其应用到访问控制策略的管理和分析中.该方法针对每个主体或者角色被指派的权限定义了一个访问控制空间,同

① <http://oss.tresys.com/projects/setools>

② Guttman Joshua D, Herzog Amy L, Ramsdell John D. SLAT: Information flow in security enhanced Linux. March 1, 2005. Available at <http://www.nsa.gov/selinux>

时将整个可能存取控制权限划分为几个子空间,并通过计算访问控制空间和合法的存取控制空间是否存在非法或不满足用户期望的交集来判断某具体的访问存取操作是否符合用户的安全策略定义.如图 1 所示,如某主体的所有可能存取动作被刻画为 Specified 空间,而用户定义的安全策略禁止主体访问某类资源(在图中被刻画为 Prohibited 空间),则 Specified 和 Prohibited 空间的交集证实了该主体违法访问该类资源的可能性,从而提示用户重新修改其安全策略文件或进一步限定该主体的访问控制权限.

在访问控制空间思想的基础上,Jaeger 等人实现了一个称之为 Gokyo 的原型系统.该系统允许用户以图形化的访问控制模型来定义和分析访问控制策略.图 2 给出了一个用户定义的安全策略的实例,

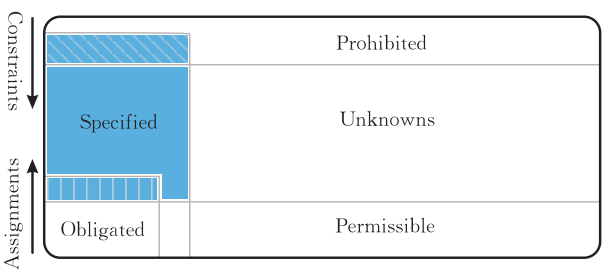


图 1 访问控制空间示意图

其中,系统中每一权限、角色和主体在图中均被赋予一个独立的节点,而节点间的指派关系也因此构成了权限、角色和主体间的等级层次关系.具体而言,节点间的指派关系可以是主体对角色具有某种权限的关系,或者是主体和权限集之间的指派关系,同时亦可能为角色的继承和约束关系.

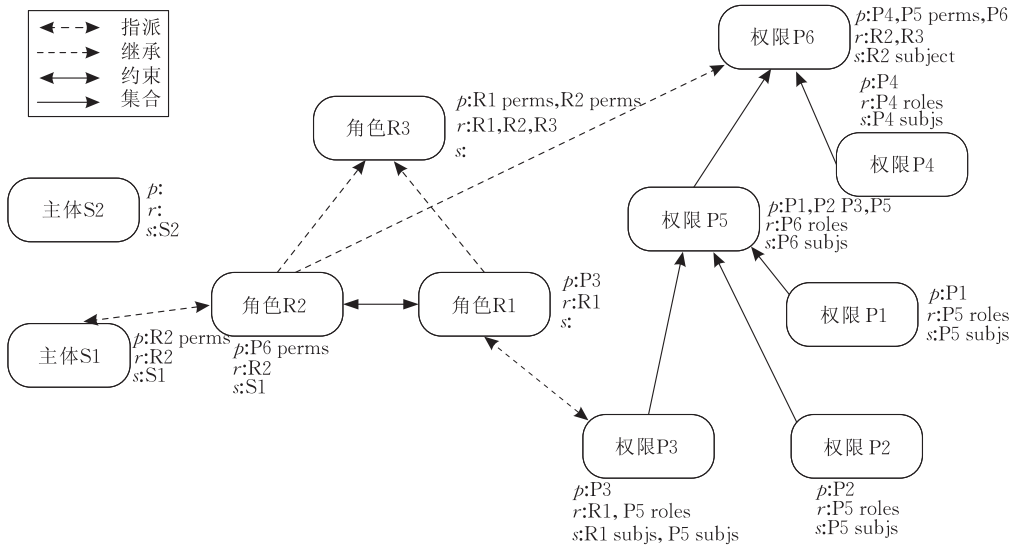


图 2 Gokyo 访问控制示意图

除计算和检测不用访问控制空间之间的非法交集之外,Gokyo 系统同时允许用户将系统中所有类型(换言之,资源类型)划分为可信(Trusted Computing Base,TCB)和不可信两类.其中,前者包含值得信任的高安全等级系统资源,而后者则包含低安全等级的不可信资源,如 `httpd_t` 等.在这种划分之下,Gokyo 工具可以根据可信度的高低检测出违反完整性(integrity)和完备性(completeness)的安全策略.例如,如果某类型资源为不能被不可信的类型进行写操作,或某可信的类型进行读操作,该策略可被视为符合完整性的定义.

由于 Gokyo 工具允许用户对访问存取的主体和对象进行可视化定义和分析,因此对用户而言,整

个分析过程显得直观,便于理解、修改以及维护.然而,为实现对系统层级的安全机制进行分析,用户不得不对系统中所有主体、角色和对象以及其间的相互关系做出刻画.这样,使得该定义分析过程较为繁琐,缺乏使用上的弹性.

3.2 信息流分析方法

直观而言,信息流分析类方法通过跟踪监测不同实体间的信息流动是否符合用户预期来判断系统中是否存在安全策略被违反的情况.在此类方法中,用户为每一实体定义专属的安全属性,同时设定各类安全属性的实体间合法信息流动的规则来刻画其对系统安全策略的配置.在用户完成相关定义后,该类方法能够跟踪在具体系统调用或运行中,实例信

息在实体间的流动,并回答诸如“高敏感级的客体中的信息是否会流向低敏感级的实体”或者“低完整级的信息是否会未经可信程序的处理就流向了高完整性的实体”等与安全相关的问题。

相对于信息流分析方法,访问控制类方法可以通过间接的方式实现对信息流动的跟踪,并完成对机密性和完整性策略的分析。但是,针对某一具体系统运行,访问控制类方法需要用户刻画多个实体、客体和角色的访问控制行为,从而导致整个分析并不直观且较为繁琐。同时,面对违反安全策略的信息流并非由主体直接对客体进行操作所导致的情况,访问控制类方法将无法准确有效地进行分析。另一方面此类情况在实际应用中却广泛地存在。例如,不可信主体不直接修改应有服务的关键配置文件,而是通过非法信息流导致应用服务的关键配置文件被非法篡改,从而影响应有服务的正常运行^[3]。因此,面对此类情况,信息流分析类方法往往更具优势。

信息流分析方法的成功范例包括 SLAT 工具和 PAL 工具。其中,SLAT 工具不仅提出安全策略(security policy)和安全上下文(security context)的概念,同时要求用户以安全流图(diagram)的方式来描述在不同安全上下文间信息的合法流动方式,从而定义、管理和实现其预期的安全目标。

尽管 SLAT 工具针对 TE 和 RABC 安全模型作了大量的策略验证,其对多级安全策略的 SELinux 系统却少有涉及。因此,在本文中,作者将提出在基本的信息流分析方法中引入敏感级标签的概念,从而将多级安全策略的 SELinux 系统的机密性与完整性策略的验证过程统一到一个形式化框架中。

4 SELinux 安全策略信息流分析框架

SELinux 系统用于提升系统安全等级的基本机制在于引导系统管理者制定合理、一致及有效的安全管理策略,并在系统运行的同时强制实施这一策略,从而实现对敏感资源机密性和完整性的有效保护。对于 SELinux 系统的这一安全提升机制我们可以从信息流和对系统的影响两个角度进行认识。

首先从信息流分析角度来说,SELinux 系统安全机制中,系统管理者制定的安全策略严格规定了系统运行时信息流必须按照策略管理者定义的方式流经合法的路径。举例来说,一个 Web 服务器,服务器管理员与用户的权限必须严格分离。由于管理员

基本上以普通用户进行常规操作,所以必须保证用户对敏感数据进行操作时,必须提供管理员口令。SELinux 安全策略范例里涉及了对 Web 服务器系统脚本的操作,系统管理员必须确保从 *user_t* 流向 *httpd_sys_script_t* 的信息流,必须经过 *httpd_admin_t*。

其次从系统角度来说,当处于安全状态下的系统中产生敏感信息的流动时,严格的系统安全机制必须保证敏感信息完成流动以后,系统必须同样处于安全状态。因此,如果我们能够形式化地描述合法的敏感信息的流动所能对系统安全状态产生的影响,则 SELinux 安全策略对合法信息流的规定和约束也能够得到合理的刻画,从而允许我们从信息流的角度对该安全策略进行严格的分析和评估。

有限状态自动机模型能够有效地刻画系统状态间的迁移关系,因而为研究界广泛采用。本文的主要工作在于提出为 SELinux 系统安全策略建立严格对应的有限状态自动机模型的方法,同时对于获得的安全策略状态机做较为深入的分析。在本节中,我们首先给出有限状态自动机的数学定义,之后在 4.1 节我们将给出与 SELinux 安全策略的描述和分析有关的信息以及这些信息的数学表示,同时利用这些表示在 4.2 节中给出严格的 SELinux 安全策略的形式化建模方法,最后在 4.3 节,我们将探讨如何对得到的模型进行进一步的分析和验证。

定义 1. 有限状态自动机。有限状态机 M 是一个五元组 (S, S_0, L, Γ, F) , 其中,

- (1) 有限集合 S 定义了 M 中所有合法状态;
- (2) $S_0 \subseteq S$ 定义了 M 中初始状态集合;
- (3) 有限集合 L 定义了 M 中所有合法的迁移标签;
- (4) $\Gamma \subseteq S \times L \times S$ 刻画 M 中所有可能的状态间迁移;
- (5) $F \subseteq S$ 定义了 M 中合法的最终状态。

4.1 安全策略相关信息和符号

从技术实现上来说,SELinux 安全服务器对系统调用是否符合安全策略进行判断的主要依据在于判定进程对特定的文件或资源的读写操作,如进程用包含特定路径的二进制文件覆盖内存空间等,是否合法安全。因此,SELinux 系统针对每一个系统调用均需进行一个或多个读写操作合法性的检查,以验证系统调用是否被允许。为形式化描述读写操作合法性检查,我们有必要理清这些读写操作中涉及

到的操作实施者和操作对象。

在 SELinux 系统中, 用户(user)能够任意被合法指派的角色(role), 如系统管理员或普通用户等, 提出操作请求。同时, SELinux 系统将所有资源或称客体, 包括文件、进程或者文件系统, 分为不同的类型(class)。不同用户的不同角色针对不同类型的资源, 被允许进行不同权限(permission)的操作。因此, 对每一具体客体而言, 其拥有者、能够对其进行操作的用户及用户的角色以及这些用户角色所能进行的合法操作构成了该客体的安全状态, 我们以安全上下文(security context)的概念来抽象之。而对某一用户的特定角色而言, 系统分配与其特定的权限。因此, 对每一用户(或主体), 其能够合法扮演的角色集合、每个角色对应的合法权限集合以及拥有该权限下能够合法访问的资源类型构成了该主体的安全上下文。在实现多安全等级的系统中, 主体的权限被扩展为安全等级区间(multiple level security range), 该区间规定了该主体的某角色所拥有的最高和最低安全等级。因此, 任一主体的安全上下文实际上可以被定义为包含用户、角色、类型和安全等级区间信息的四元组。而读写操作的实施者和对象对应的安全上下文给予了系统对该操作是否安全进行判断的依据。

为表达上的简洁, 我们引入了如下集合, 以描述系统中所有与安全策略刻画和分析相关的信息:

P_u : SELinux 安全策略中的用户集合;

P_r : SELinux 安全策略中的角色集合;

P_t : SELinux 安全策略中的类型集合;

P_c : SELinux 安全策略中的类集合;

P_p : SELinux 安全策略中的权限集合;

K_{mls} : SELinux 安全策略中主客体可能拥有的安全等级集合。

4.2 安全策略形式化建模

对 SELinux 系统的安全策略的建模过程可大致分为两个步骤: 首先对该策略中用户声明的主客体信息及安全规则给出严格的数学表达, 同时得到主客体的相关安全上下文信息; 然后对系统中可能的信息流, 该信息流涉及的主客体的安全上下文为相关状态集合, 生成这些状态间可能进行的合法迁移。

首先, 针对安全策略中的用户声明, 包括用户、角色、类型、安全等级及几者间的关系的声明, 我们将其描述为对应的集合, 如定义 2~9 所示。

定义 2. 安全上下文集合. 集合 $\sigma \subseteq P_u \times P_r \times P_t \times K_{mls}$ 定义了系统中所有的主客体所可能具有的安全上下文。具体而言, 对于某特定的主体或客体 e , 其安全上下文 $\sigma_e \in \sigma$ 是一个四元组 $(u, r, t, (s, c))$, 其中:

(1) u 在主体中代表正在执行的进程, 在客体中代表此客体的拥有者;

(2) 角色 r 代表了可以使用某种权限的一组用户集合;

(3) 类型(又称为域, domain) t 代表了主体所能访问或客体所包含的资源类别。在 SELinux 系统中, 系统中所有的资源, 包括进程、文件、套接字等被划分为多个类别。对于具体系统调用而言, 其涉及的主客体的当前安全上下文中的类型信息为 SELinux 系统提供了判断是否实施该调用的主要依据;

(4) 区间 $(s, c) \in K_{mls}$ 定义了主体或者客体的安全级别, 其中敏感级 $s = (l, h)$ 定义了主体所被允许的权限级别或者客体所属的分类级别的最低敏感级 l 和最高敏感级 h , 而类属范畴 c 定义了主体可以访问的等级区间, 从而仅对主体对象有实际意义。

借用安全上下文的概念, 安全策略自动机 M 中的状态空间可以因此定义为系统中所有主体或客体的安全上下文合法取值空间的积, 亦即 M 中所有状态均刻画了在安全状态下系统中所有主客体所具有的合法安全上下文的某特定组合。

定义 3. 迁移标签. 集合 $L \subseteq P_c \times P_p$ 定义了系统中允许的主体对客体的所有合法访问方式。对于主体 s 而言, 如其通过类 c 的方式以权限 p 对客体 o 的访问是合法的, 则必有 $(c, p) \in L$ 。

下面我们定义 SELinux 有限状态自动机模型中的状态转换关系。

定义 4. 主体合法角色集合. 关系 $\mu \subseteq P_u \times P_r$ 定义了系统中的主体(或用户)所能扮演的合法角色, 即主体 u 可以合法扮演角色 r , 当且仅当 $(u, r) \in \mu$ 。

如将集合 μ 按步扩展到 P_r 的 n 次空间, 即 $\mu \subseteq P_u \times P_r^n$, 则主体 u 可以合法扮演角色 r , 当且仅当 $\exists (u, r_1, r_2, \dots, r_n) \in \mu \exists 1 \leq i \leq n (r = r_i)$ 。

在 SELinux 系统中, 用户可以通过 `user root roles{user_r sysadmin_r}` 的形式来声明用户所能合法扮演的角色。通过这样的声明, 实际上用户向关系 μ 中加入了元素 $(root, user_r, sysadmin_r)$, 从而允许 `root` 用户可同时扮演一般用户和系统管理员用

户两个角色.

定义 5. 角色合法域集合. 关系 $\gamma \subseteq P_r \times P_t$ 定义了系统中用户能够合法进入的所有域, 即当用户的当前角色为 r 时, 其进入域 t 的要求是合法的, 当且仅当 $(r, t) \in \gamma$. 如将 γ 按步扩展到 P_t^n 空间, 则角色 r 进入域 t 的要求是合法的, 当且仅当 $\exists (r, t_1, t_2, \dots, t_n) \in \mu \exists 1 \leq i \leq n (t = t_i)$.

在 SELinux 系统中, 用户往往通过 `role user_t types user_t` 的形式来声明角色所能合法进入某特定域的关系. 通过这样的声明, 用户等同于向关系 γ 中加入了 $(\text{user}_r, \text{user}_t)$ 的二元组, 从而允许当用户能够合法扮演角色 `user_r` 时, 能够合法进入 `user_r` 所属的域 `user_t`.

定义 6. 合法角色转换关系. 集合 $\vartheta \subseteq P_r \times P_p \times P_r$ 定义了系统中合法的角色间的转换关系. 如在某系统调用中, 要求某主体的角色 r_1 经由权限 p 转换为角色 r_2 , 则该要求被视为合法的当且仅当存在 $(r_1, p, r_2) \in \vartheta$. 同样的, 如果将上述合法角色转换关系按步扩展到 P_p 的 n 次空间上, 成为 $\vartheta \subseteq P_r \times P_p^n \times P_r$ 时, 则某系统调用要求某主体的角色 r_1 经由一系列的权限变化 $p_1 \rightarrow p_2 \rightarrow \dots \rightarrow p_n$ 转换为角色 r_2 被视为合法的, 当且仅当 $(r_1, p_1, p_2, \dots, p_n, r_2) \in \vartheta$ 为真.

定义 7. 合法类型转换关系. 集合 $\tau \subseteq P_t \times P_t \times P_c \times P_p$ 定义了系统中合法的类型(换而言之, 域)间的合法转化关系. 如果在某系统运行中, 要求类型 t_1 可以经由权限 p 对具有类 c 的类型 t_2 进行操作, 那么这样的操作要求被视为合法的, 当且仅当 $(t_1, t_2, c, p) \in \tau$ 为真. 同样的, 如果将集合 τ 按步扩展到 P_p 的 n 次空间上, 成为 $\tau \subseteq P_t \times P_t \times P_c \times P_p^n$, 则系统运行时要求的类型 t_1 可以经由一系列的权限变化 $p_1 \rightarrow p_2 \rightarrow \dots \rightarrow p_n$ 对具有类 c 的类型 t_2 进行的操作将被视为合法的, 当且仅当 $(t_1, t_2, c, p_1, p_2, \dots, p_n) \in \tau$ 为真.

在 SELinux 安全策略配置时, 用户可以采用声明 `allow initrc_t httpd_t: process transition` 的形式定义具体的合法类型转换关系, 通过这样的声明, 用户实际上在合法域转换关系中加入了 $(\text{intric}_t, \text{httpd}_t, \text{process_transition})$ 成员, 从而声明了允许系统中的开机程序可以对 `httpd_t` 进行转换的操作.

定义 8. 多安全等级下合法权限(操作)关系. 集合 $\varphi \subseteq P_p \times K_{m1s} \times K_{m2s}$ 定义了当系统存在多安全等级时具有特定安全级别的主体对某安全级别的客

体所能进行的合法操作. 具体言之, 如果某主体的安全级别为 $\kappa_{m1s1} = (l_1, c_1)$, 客体的安全级别为 $\kappa_{m2s2} = (l_2, c_2)$, 且该主体可对该客体进行 p 操作, 当且仅当 $(p, \kappa_{m1s1}, \kappa_{m2s2}) \in \varphi$.

与此同时, 为适应系统机密性和完整性的要求, 集合 φ 必须满足 $(p, \kappa_{m1s1}, \kappa_{m2s2}) \in \varphi \Rightarrow ((l_1 \geq l_2) \wedge (c_1 \supseteq c_2) \wedge \text{readlike}(p)) \vee ((l_1 \leq l_2) \wedge (c_1 \subseteq c_2) \wedge \text{writelike}(p))$, 其中谓词 $\text{readlike}(p)$ 为真, 当且仅当操作 p 将读取其目标中的数据; 而谓词 $\text{writelike}(p)$ 为真当且仅当操作 p 向其目标中写入数据.

在具体应用中, 某些操作既会向目标写入也会从目标中读取数据, 例如 `transition` 动作类的 `relabel` 重新标注目标的安全上下文操作. 为适应和处理这种情况, 我们规定当该操作降低目标的安全级别时,

$$\text{readlike}(\text{relabel}) = 1, \text{writelike}(\text{relabel}) = 0;$$

反之, 当该操作提高目标的安全级别时,

$$\text{readlike}(\text{relabel}) = 0, \text{writelike}(\text{relabel}) = 1.$$

定义 9. 约束关系定义. 集合 $\chi_{cp} \subseteq P_u \times P_r \times P_t \times P_u \times P_r \times P_t$ 定义了用户对某一实体或客体的状态间能够经过操作 (c, p) 完成合法迁移的约束. 对于某特定的主体或客体而言, 如其初始安全状态为 (u_1, r_1, t_1) , 则其经过操作 (c, p) 到达状态 (u_2, r_2, t_2) 的迁移是合法的, 当且仅当 $(u_1, r_1, t_1, u_2, r_2, t_2) \in \chi_{cp}$.

为表达简洁起见, 在本文后续部分, 我们将“元素 s 属于集合 R ”直接记为 $R(s)$. 那么, 在完成了对安全策略中主客体及安全规则声明的形式化描述并产生所需安全状态空间之后, 我们可以直观地得到形如式(1)的这些状态间合法迁移的生成规则.

$$\begin{aligned} & \Psi_{c,p}(t_1, r_1, u_1, \kappa_{m1s1}; t_2, r_2, u_2, \kappa_{m2s2}) \Leftrightarrow \\ & \tau(t_1, t_2, c, p) \wedge \gamma(r_1, t_1) \wedge \gamma(r_2, t_2) \wedge \mu(u_1, r_1) \wedge \\ & \mu(u_2, r_2) \wedge \chi_{cp}(u_1, r_1, t_1, u_2, r_2, t_2) \wedge \varphi(p, \kappa_{m1s1}, \kappa_{m2s2}) \wedge \\ & ((c = \text{process}) \wedge (p = \text{transition}) \wedge \vartheta(r_1, r_2)) \quad (1) \end{aligned}$$

实际上, 式(1)亦可理解为 SELinux 系统针对主体对客体进行特定操作的请求的授权规则. 当式(1)右侧为真时, SELinux 系统可以授权拥有安全上下文, 当上式 $(t_1, r_1, u_1, \kappa_{m1s1})$ 的主体对被赋予安全上下文 $(t_2, r_2, u_2, \kappa_{m2s2})$ 的客体进行 (c, p) 操作, 同时不会影响系统的安全的特性. 然而, 为直观起见, 我们可以用图 3 来说明式(1)作为产生安全状态间合法迁移的规则的意义. 如图 3 所示, 图中的结点表示系统满足的安全状态, 而状态间的边则表示当系统满足该安全迁移条件时, 能够合法转移到下一个安全状态.

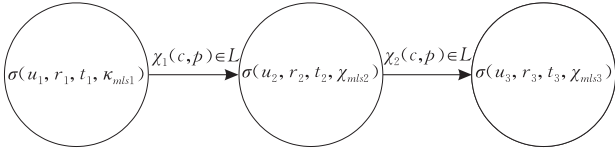


图 3 安全状态合法迁移关系

为进一步说明式(1),可以考虑如下实例:

在 SELinux 的安全策略范例文件中,严格地限制了对原始磁盘数据的访问,用户不能直接对 type fixed_disk_device_t 进行操作,只有系统管理员可以直接对 type fixed_disk_device_t 进行访问。

从信息流的角度来分析,上述规定意味着从 type user_t 流向 type fixed_disk_device_t 的信息流,必须经过 type fsadmin_t,那么,其安全状态迁移如图 4 所示。

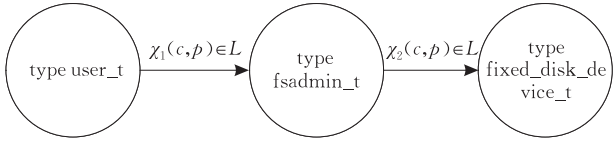


图 4 原始磁盘访问的信息流图

式(1)仅仅给出了建立安全策略模型中状态迁移关系的合理性(soundness),即任一对安全状态间如果存在迁移,则该迁移必然是合法迁移。但是,为保证有关安全模型的验证结果的正确性,有必要保证安全策略模型中迁移关系的完整性,即所有应该被考虑到的安全状态间的迁移均应被引入到安全策略模型中。而式(2)保证了这样的完整性。

$$\text{allflows} = \{ \bigcup \Gamma_{ij}, 1 \leq i, j \leq N \} \quad (2)$$

其中,allflows 定义了安全模型中所有迁移关系,而 $\Gamma(\sigma_i, \sigma_j) = \{ (\sigma_i, \sigma_j) \mid \forall (c, p) \in L. \text{writelike}(p) \wedge \Psi_{c,p}(t_1, r_1, u_1, \kappa_{mls1}; t_2, r_2, u_2, \kappa_{mls2}) \vee \text{readlike}(p) \wedge \Psi_{c,p}(t_2, r_2, u_2, \kappa_{mls2}; t_1, r_1, u_1, \kappa_{mls1}) \}$ 定义了主体安全上下文 i 和客体安全上下文 j 之间应该具有的迁移关系。

直观来说,主体 s 可以对客体 o 进行读操作从而产生从客体 o 到主体 s 的信息流动;或者 s 对 o 进行写操作时,从而产生从 s 到 o 的信息流动。而 $\Gamma(\sigma_i, \sigma_j)$ 正定义了这两个方向的合法信息流动。

4.3 SELinux 安全策略模型的分析

SELinux 系统安全策略对应的有限状态自动机的建立,为系统管理员以及系统安全分析人员全面有效地了解安全策略提供了直观的媒介,因而大大地方便了这些人员对安全策略的分析、跟踪和维护。然而,使用者对安全系统信心地建立,在于能够彻底

地证明该系统切实、完整、完备地实现了对敏感信息的控制和保护,而不能仅仅通过审查安全策略,或者遍历解安全策略自动机模型从而到达对安全策略的全面了解来实现。形式化方法中模型检测技术^[8]正为建立这样一套正确、彻底的证明提供了支持。

模型检测技术的核心思想在于为待测系统建立有限状态模型,并且将用户期待该系统应该展现的属性刻画为逻辑公式,同时通过状态穷举的方法遍历整个系统模型,从而得到可信的关于系统是否满足用户期望的论断。因此,在获得了 SELinux 安全策略的有限状态模型之后,为了能够对该模型进行深入分析并判断该策略的实施是否有效地保障了合理的安全目标的实现,我们有必要将该模型提交模型检测工具作全面的验证。本文的着眼点并不在于提出有效快速的针对安全策略模型的模型检测算法,而是对在提出安全策略建模方法之后,能够合理将 SELinux 用户对安全的共同期望提炼出来并刻画为正确的逻辑公式,从而使得用户可以将安全策略模型以及安全目标逻辑公式提交给通用的模型检测工具进行验证。

为实现上述目标,我们对 SELinux 安全策略范例的发行版中描述的基本安全目标进行了分析,并针对这些目标提出了相应的逻辑公式刻画方法。本文选择线性时序逻辑(Linear Temporal Logic, LTL)^[9]作为刻画安全目标的逻辑。为使读者能够理解 LTL 逻辑公式,本文仅简要介绍 LTL 中常用的量词和操作符,关于 LTL 逻辑的详细语法及语义,请参阅文献^[6]。

在 LTL 逻辑中,量词 G 的使用规定了期望属性需要在有限状态模型中被始终保持,亦即一个模型符合属性 Gp 当且仅当模型中所有可能路径均满足 p 。类似的,量词 F 代表了有限状态模型中任一路径总存在某个状态满足期望属性。在 LTL 逻辑中,操作符 X 代表了期望属性需要被当前状态的下一个状态所满足。而当操作符 U 同时作用于期望属性 p 和 q 形成公式 pUq 时,代表了如当前路径中某一状态满足 q ,则其所有前驱状态均满足 p 。类似的,当操作符 R 同时作用于期望属性 p 和 q 形成公式 pRq 时,代表了当前路径的所有前缀均满足 q ,或者存在当前的状态的某一前驱满足 p 。

从 SELinux 安全策略范例中,我们归纳了以下 6 个安全目标:

- (1) 控制各种形式对数据的访问;
- (2) 保护内核的完整性;

(3) 保护系统软件、系统配置信息、系统日志的完整性;

(4) 限制需要利用特权进程的漏洞而造成的潜在威胁;

(5) 防止在没有得到授权的情况下, 获得管理员角色或者域;

(6) 防止普通用户进程干扰系统进程或者管理员进程.

目标 1 的刻画较为复杂, 因为该目标针对主体对客体的具体操作提出了 3 个方面的要求: 首先主体必须具备相应的权限以完成操作, 即在任意状态下, 主体 s 可以拥有对客体 o 的权限 j , 仅当主体拥有安全上下文 i , 且客体拥有安全上下文 k ; 其次, 该客体不能为某恶意主体 x 所访问; 最后, 该操作也不能降低主/客体的安全等级, 即如主/客体的初始安全上下文为 c , 则操作后其安全上下文不能为 y . 对第 1 方面, 我们可以采用逻辑公式 $G(SC_s(i) \wedge SC_o(t) \rightarrow P_{so}(j))$ 进行刻画, 其中谓词 SC_s 和 SC_o 判定某安全上下文是否为 s 和 o 在当前状态下的安全上下文, 而谓词 P_{so} 则用来判定 s 对 o 是否具有某特定权限. 对于第 2 方面, 可用公式 $G(\neg(SC_s(i) \wedge SC_o(t) \wedge (s=x)))$ 进行刻画. 最后, 形如 $G(\neg(SC_s(c) \rightarrow F(SC_s(y))))$ 的逻辑公式能够有效地刻画目标 1 的第 3 个方面. 在完成对以上 3 个方面的刻画之后, 得到的所有逻辑公式的逻辑与构成了目标 1 对应的逻辑公式.

目标 2 和 3 在本质上均可表述为“如果行为 p 改变了安全上下文为 i 的客体 o 的内容, 那么这个行为必须源自安全上下文为 j 的主体的成功请求”的形式. 因此, 如果我们引入谓词 $Name_i(s)$ 以判定主/客体 s 在当前状态下是否具有安全上下文 i , 同时引入谓词 $Succ_{ji}(p)$ 以判断当前状态下, 具有安全上下文 j 的主体能否成功的对具有安全上下文为 i 的客体进行改变其内容的 p 操作, 则目标 2 和 3 均可 LTL 公式 $G(SC_o(i) \wedge \text{writelike}(p) \rightarrow \exists s \in S(\text{Name}_j(s) \cup \text{Succ}_{ji}(p)))$ 所刻画, 其中 S 为系统所有主体集合, 而 $\text{writelike}(p)$ 表示操作 p 为可写操作.

类似的, 目标 4 亦可表述为“如果行为 p 改变了安全上下文为 i 的客体 o 的内容, 那么这个行为必须源自安全上下文为 j_1, \dots , 或 j_n 的主体的成功请求”, 其中安全上下文 i 涵盖了那些不希望受到拥有安全上下文 $k \notin \{j_1, \dots, j_n\}$ 的主体的影响的客体. 因此, 目标 4 可以被刻画为如下公式:

$$G(SC_o(i) \wedge \text{writelike}(p) \rightarrow \exists s \in S(\neg(\text{Name}_{j_1}(s) \vee \text{Name}_{j_2}(s) \vee \dots \vee \text{Name}_{j_n}(s)) \cup \text{Succ}_{j_i}(p))).$$

目标 5 实际上说明了系统使用者期望任何导致主体 s 进入管理员角色或管理员域的安全上下文 i 的行为 p 都在事先得到授权. 因此 LTL 公式 $G((p \rightarrow X \text{Admin}(s)) \rightarrow Do(s, p) R \neg \text{Auth}_s(i))$ 能够合理地刻画目标 5 代表的安全期望, 其中 $Do(s, p)$ 表示主体 s 完成了行为 p , $\text{Admin}(s)$ 表示该主体有管理员权限, 而 $\text{Auth}_s(i)$ 表示主体已获得授权.

最后, 目标 6 规定了任何改变系统状态或管理员进程对象 t 的行为 p 均不能由普通用户 s 完成. 那么, 通过引入谓词 $Op_p(t)$ 以判定 t 是否是行为 p 的作用对象, 同时利用上面引入的谓词 Admin , 我们可以直观地得到目标 6 对应的逻辑公式

$$G(\neg((\text{writelike}(p) \vee (Op_p(t) \wedge \text{Admin}(t))) \rightarrow \neg \text{Admin}(s) \wedge Do(s, p))).$$

值得注意的是, 上述逻辑公式中主体、客体以及行为实例均是作为参数引入的. 因此, 在利用这些公式对安全策略模型进行检测时, 需要使用者将这些参数实例化为他所关注的具体对象以完成有效的验证过程. 同时, 对于实际安全系统而言, 其安全策略的建模过程不可避免地需要使用者选择放弃某些信息, 以降低建模过程的复杂程度, 同时减少最终获得的模型中包含的状态规模. 因此, 不恰当地放弃有关信息从而导致的精度丢失同样也会影响验证过程的精确程度. 通常来说, 使用者往往需要多次执行验证过程, 不断加入必要信息, 才能获得最终有说服力和使用价值的验证结果. 最后, 本节仅仅讨论了针对 SELinux 系统的 6 类安全目标, 对于实际系统而言, 值得关注的安全目标当然不止于此. 对于复杂的安全目标, 如保护特权进程不会被用来执行恶意代码以及保护系统不受利用 Netscape 浏览器漏洞进行攻击的恶意代码所带来的威胁等, 需要通过结合系统实现细节进行进一步分析刻画.

5 实例分析

在得到安全策略对应的自动机模型以及安全模型的逻辑刻画并将其中参数实例化为所关心的对象之后, 分析人员可以将这些结果作为输入提交给能够对有限状态自动机(具体结合到本文提出的方法, 实际为标号迁移系统(Labeled Transition System, LTS))以检测其是否符合 LTL 公式的模型检测工具, 如 SPIN^[10]等.

在本节中,我们将通过将前面描述的方法应用到一个简单范例以及一个较为复杂的实际系统上.由于本节所举实例中隐藏的非法信息流动并非由主体直接对客体进行操作所导致的情况,因此访问控制分析方法无法对此类情况进行有效分析;另一方面,这些实例中都应用了敏感级标签来定义安全属性,因此,现有的信息流分析工具,如 SLAT 和 PAL 等对此类情形也无能为力.通过下面的实例,我们可以说明:

- (1) 本文提出的安全策略建模方法能够为实际系统建立有足够表达能力的标号迁移系统;
- (2) 4.3 节中抽取并形式化的安全目标具有一定的代表性,能够描述实际分析中常常遇到的安全考虑.同时,这些安全目标对应的逻辑公式正确无误,能够被分析人员用来对安全模型进行检测;
- (3) 如果在建模过程中有效地保留了验证所需的信息,则验证结果能够有效地回到分析人员关心的安全问题.

由于篇幅的关系,在后文示例中,将仅给出与分析相关的安全策略模型的片断.

5.1 Apache 安全策略分析

首先,我们以文献[3]中给出的 Apache 安全策略配置为例,分析其是否符合前文提出的安全目标6,即“系统能够有效地防止普通用户进程干扰系统进程或者管理员进程”.

在 Apache 安全策略中,Web 服务器的管理员(httpd_admin_t)处理的是和系统相关的高机密性

信息,如系统脚本(httpd_sys_script_t)等,因而自身的密级也较高;而普通用户(users_t)则处于低密级,只能访问自己的配置脚本(httpd_user_script_t)和 Web 服务器上公开的信息等.因此,我们可以得到如下规则:

```
allow httpd_admin_t httpd_user_script_t: file { read };
allow users_t httpd_user_script_t: file { create read write };
```

然而上述的安全规则却会使得式(2)中包括以下两类迁移:

$$\Gamma(\sigma(u_1, r_1, users_t, (s2.c)), \sigma(u_1, r_1, httpd_user_script_t, (s2.c)));$$

$$\Gamma(\sigma(u_1, r_1, httpd_user_script_t, (s2.c)), \sigma(u_2, r_2, httpd_admin_t, (s5.c))).$$

即上述规则允许系统中出现从 users_t 到 httpd_admin_t 的信息流,而这违反了“不同用户不得干扰管理员进程”的初衷,因而该安全策略是不安全的.

5.2 采购系统安全策略分析

图 5 描述了一个简单的采购系统的采购流程图.在该系统中,供应商通过 socket 给出报价(quote),报价处理程序处理报价(E-declare),以确保订单合法.报价处理将合法的报价单放入报价列表(quote list)中,采购程序(purchasing)读入报价列表生成订单,再将订单放入订单列表(order list),最后由订单发放程序(order dispensing)将订单发放到各单位.

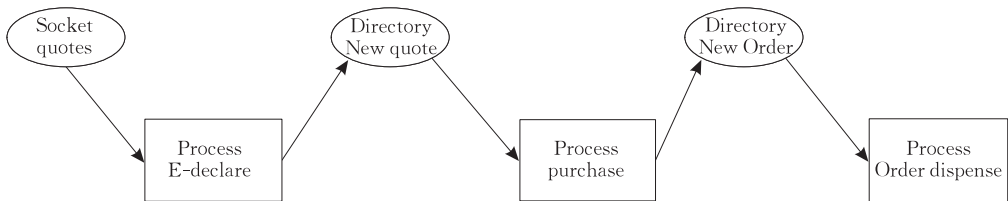


图 5 采购处理流程

出于机密性的考虑,供货商是禁止获得报价列表的,以防其篡改报价,同时,供货商也不能获得完整的订单列表,以防止供货商从订单列表中获得完整的装备计划.因此,报价列表和订单列表的密级应高于 socket 的密级,以避免信息泄露.因此,我们能够得到整个采购系统的安全策略,显示如下:

```
attribute domain;
attribute file_type;
attribute exec_type;

type quote_sock_t;
type declare_t, domain;
type order_dispense_t, domain;
```

```
type purchase_t, domain;
type quote_list_t, file_type;
type order_list_t, file_type;
type purchase_exec_t file_type, exec_type;

user sys_u roles {epurchase_r};
user vendor_u roles {epurchase_r};

role epurchase_r types declare_t;
role epurchase_r types purchase_t;
role epurchase_r types order_dispense_t;

allow declare_t quote_sock_t: tcp_socket { ioctl read
getattr append connect shutdown listen accept };
allow declare_t quote_list_t: file { create write };
```

```
allow purchase_t quote_list_t: file {read};
allow puschase_t order_list_t: file {create write};
allow order_dispense_t order_list_t: file {read};

type_transition sys_admin_t purchase_exec_t: process
declare_t;

allow sys_admin_t purchase_exec_t: file {read, getat-
tr, exec};
allow sys_admin_t declare_t: process {transition};
allow declare_t pursechase_exec_t: file {entrypoint
ecec read};
allow sys_admin_r epurchase_r;
allow system_r epurchase_r;
```

将本文提出的建模方法应用到这一安全策略上,可以得到形如图 6 所示的安全策略的信息流向,其中 C1~C7 表示安全上下文标记的系统状态,CP1~CP13 为操作许可,其具体值如表 1 所定义.

如果在上述策略中加入规则:allow declare_t order_list_t: file {create write},则由于 declare_t 需要向 quote_order_t 写入数据,其密级必然不会高于 order_list_t,这样这条规则是不违反机密性原则的.使用我们的方法可以发现这条规则会导致由

declare_t 直接流向 order_list_t 的信息流,如图 7 中虚线所示.然而,这条规则会允许 declare_t 未经确认的报价插入order_list_t,从而产生虚假订单.违反了系统的安全需求.

表 1 模型中标号的安全上下文定义

标号	定义
C1	(sys_admin_u, sys_sadmin_r, sys_admin_t)
C2	(vendor_u, epurchase_r, declare_t)
C3	(user_u, object_r, quote_list_t)
C4	(vendor_u, epurchase_r, purchase_t)
C5	(user_u, object_r, order_list_t)
C6	(vendor_u, epurchase_r, order_dispense_t)
C7	(user_u, object_r, quote_sock_t)
C8	(user_u, object_r, purchase_exec_t)
CP1	(process, transition)
CP2	(file, create)
CP3	(file, write)
CP4	(file, read)
CP5	(file, create)
CP6	(file, write)
CP7	(file, read)
CP9	(tcp_sock_t, ioctl)
CP10	(file, exec)
CP11	(file, read)
CP12	(file, exec)
CP13	(file, entrypoint)

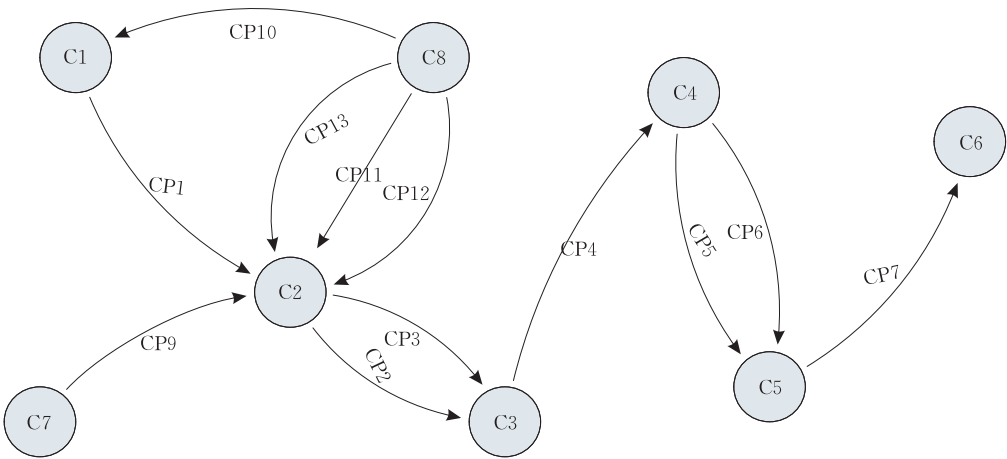


图 6 采购系统的信息流向图

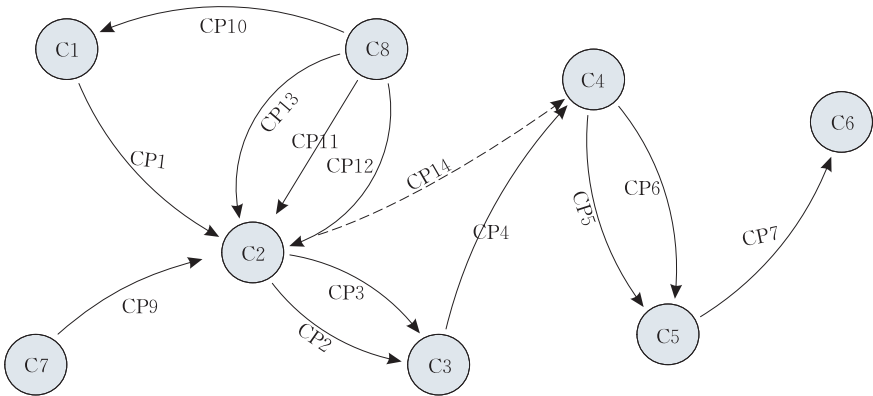


图 7 加入新规则后的信息流向图

6 论文总结

SELinux 的出现为操作系统领域提供有限的高安全等级实现的同时也带来了一些亟待解决的安全问题,特别是如何在 SELinux 系统引入多安全等级思想的情况下,对该类系统中安全策略进行行之有效的安全分析. 具体来说,多策略可动态配置安全操作系统的实现使得安全策略的实施更为方便和灵活. 然而,对于日益庞大的系统来说,完整的安全策略须囊括操作系统的各个方面,从而导致了安全策略实施文件规模庞大难于分析.

文章在综合分析了当前流行的对策略实施与模型一致性进行验证的方法的基础上,在信息流分析方法的基础上引入了多级安全敏感标签,从而实现了以有限状态自动机模型来刻画实际的安全策略. 同时文章在分析抽取 SELinux 系统的常用安全目标的基础上,以线性时态逻辑为工具,对这些安全目标进行了形式化刻画,从而使得严格验证安全策略是否达到了安全目标成为可能.

值得注意的是,文章仅仅是严格分析验证 SELinux 系统安全策略研究中的第一步. 在未来工作中,需要完善本文提出的方法,使其能够应用于对大规模复杂安全策略以及不断涌现的新安全目标的分析. 同时,提出一套专用于安全策略模型检测的有效算法,也是未来研究工作的一个重点.

从更广泛的角度来说,本方法的框架并不限制于 SELinux. 事实上,除了在方法的具体实施阶段,本方法并没有依赖于 SELinux 的特性,对策略和安全目标的形式化建模是可以适用于其它安全操作系统的策略分析的.

参 考 文 献

[1] Loscocco P A, Smalley S D. Integrating flexible support for security policies into the Linux operating system//Proceedings of the TREENIX Track: 2001 USENIX Annual Technical Conference. Berkeley, CA, USA, 2001: 29-42

[2] Smalley Stephen, Fraser Timothy. A security policy configuration for the Security-Enhanced Linux. SELinux Documentation Available at <http://www.nsa.gov/research/files/selinux/papers/policy.pdf>

[3] Jaeger T, Zhang Xiao-Lan, Edwards A. Policy management using access control spaces. ACM Transactions on Information and System Security, 2003, 6(3): 327-364

[4] Jaeger Trent, Sailer Reiner, Zhang Xiao-Lan. Analyzing integrity protection in the SELinux example policy//Proceedings of the USENIX Security Symposium. Berkeley, CA, USA, 2003: 5

[5] Zanin Giorgio, Mancini Luigi V. Towards a formal model for security policies specification and validation in the SELinux system//Proceedings of the 9th ACM Symposium on Access Control Models and Technologies. New York, USA, 2004: 136-145

[6] Guttman Joshua D, Herzog Amy L, Ramsdell John D, Skorupka Clemanet W. Verifying information flow goals in security-enhanced Linux. Journal of Computer Security, 2005, 13(1): 115-134

[7] Sarna-Starosta B, Stoller S D. Policy analysis for security-enhanced Linux//Proceedings of the 2004 Workshop on Issues in the Theory of Security(WITS). Barceloma, Spain, 2004: 1-12

[8] Clarke E M, Grumberg O, Peled D. Model Checking. Massachusetts: MIT Press, 2000

[9] Allen Emerson E. Temporal and Modal Logic, Handbook of Theoretical Computer Science (vol. B): Formal Models and Semantics. Cambridge, MA: MIT Press, 1991: 995-1072

[10] Holzmann G. The SPIN Model Checker. Boston: Addison-Wesley Press, 2003



ZHANG Yang, born in 1971, Ph. D..
Her research interests focus on secure operating system and its verification.

Background

The work attributes to the project “Research on the Key Technology of High-level Secure OS Design and Verification”, which is supported by the National High Technology Research and Development Program (863 Program) of China under grant Nos. 2007AA01Z465, 2006AA01Z433, 2007AA01Z414.

SELinux has been adopted in the mainstream Linux Distributions for the reason that it extends Linux with a flexible mandatory access control mechanism that enforce security policies expressed in SELinux’s policy language. However, determining whether a given policy meets security goals is difficult, due to the size and the complexity of SELinux policies. To analyze the validity and conformance with the security goal of SELinux policy configurations, many approaches has been developed. These methods can be divided into two

categories: access control space based and information flow based. While the access control space-based analysis can not deal with the illegal information flow caused by indirect access requirement and its analysis procedure is complex and not intuitive, the information flow-based method draws more and more researcher’s attention

In this paper, to the question that current information flow-based SELinux policy analysis can not manipulate all its semantic properties (mls configuration for example), the authors introduce the multi-level sensitivity label into the traditional information flow analysis. On the basis of automata and linear temporal logic, an improved information flow analysis is proposed. It can verify both the integrity and confidentiality of the SELinux policies.