

基于标签的数据流转控制策略冗余与冲突检测方法

谢绒娜¹, 范晓楠², 李苏浙², 黄宇欣², 史国振¹

(1. 北京电子科技学院密码科学与技术系, 北京 100070; 2. 北京电子科技学院网络空间安全系, 北京 100070)

摘要: 基于标签的数据流转控制机制通过主客体标签实现数据流转控制, 具有轻量级、延伸控制的优势, 引起了广泛关注。数据流转时, 标签变更不可避免, 而在标签变更时, 新标签与已有标签难免存在冗余或者冲突。如何对标签进行冗余与冲突检测是基于标签的数据流转控制中急需解决的问题。针对上述问题, 提出了基于原子操作的标签描述方法。客体标签由多个原子标签的逻辑组合生成。其中, 原子标签用于描述最小的安全需求, 解决了标签描述简洁性和丰富性问题。为降低标签冗余与冲突检测难度、提高检测效率, 基于标签中不同集合的相关性, 提出了基于标签的数据流转控制策略冗余与冲突检测方法。该方法通过分析原子标签中各要素的集合关系对原子标签进行冗余与冲突检测, 基于原子标签检测结果和逻辑关系对客体标签进行检测, 提高了检测效率; 基于不同原子标签中包含操作的关系对原子标签进行冗余与冲突检测, 对于包含相同操作的不同原子标签, 通过分析不同原子标签中主体属性、环境属性以及规则类型之间关系进行检测, 对于包含不同操作的原子标签, 如果不同操作之间没有关系, 那么原子标签不存在冗余与冲突, 如果不同操作之间存在偏序关系, 则通过分析不同原子标签中操作的偏序关系、主体属性、环境属性以及规则类型之间关系进行检测。从理论和实验两个角度对提出的冗余与冲突检测方法性能进行分析, 通过实验验证了原子标签数量和复杂度对检测性能的影响。

关键词: 标签; 数据流转控制; 原子标签; 集合相关性; 策略冗余与冲突检测

中图分类号: TP302

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2023074

Redundancy and conflict detection method for label-based data flow control policy

XIE Rongna¹, FAN Xiaonan², LI Suzhe², HUANG Yuxin², SHI Guozhen¹

1. Department of Cryptography and Science Technology, Beijing Electronic Science and Technology Institute, Beijing 100070, China

2. Department of Cyberspace Security, Beijing Electronic Science and Technology Institute, Beijing 100070, China

收稿日期: 2023-02-13; 修回日期: 2023-08-18

通信作者: 谢绒娜, 486503266@qq.com

基金项目: 国家自然科学基金(61932015); 国家重点研发计划(2017YFB0802705)

Foundation Items: The National Natural Science Foundation of China (61932015), The National Key R&D Program of China (2017YFB0802705)

引用格式: 谢绒娜, 范晓楠, 李苏浙, 等. 基于标签的数据流转控制策略冗余与冲突检测方法[J]. 网络与信息安全学报, 2023, 9(5): 21-32.

Citation Format: XIE R N, FAN X N, LI S J, et al. Redundancy and conflict detection method for label-based data flow control policy[J]. Chinese Journal of Network and Information Security, 2023, 9(5): 21-32.

Abstract: To address the challenge of redundancy and conflict detection in the label-based data flow control mechanism, a label description method based on atomic operations has been proposed. When the label is changed, there is unavoidable redundancy or conflict between the new label and the existing label. How to carry out redundancy and conflict detection is an urgent problem in the label-based data flow control mechanism. To address the above problem, a label description method was proposed based on atomic operation. The object label was generated by the logical combination of multiple atomic tags, and the atomic tag was used to describe the minimum security requirement. The above label description method realized the simplicity and richness of label description. To enhance the detection efficiency and reduce the difficulty of redundancy and conflict detection, a method based on the correlation of sets in labels was introduced. Moreover, based on the detection results of atomic tags and their logical relationships, redundancy and conflict detection of object labels was carried out, further improving the overall detection efficiency. Redundancy and conflict detection of atomic tags was based on the relationships between the operations contained in different atomic tags. If different atomic tags contained the same operation, the detection was performed by analyzing the relationship between subject attributes, environmental attributes, and rule types in the atomic tags. On the other hand, if different atomic tags contained different operations without any relationship between them, there was no redundancy or conflict. If there was a partial order relationship between the operations in the atomic tags, the detection was performed by analyzing the partial order relationship of different operations, and the relationship between subject attribute, environment attribute, and rule types in different atomic tags. The performance of the redundancy and conflict detection algorithm proposed is analyzed theoretically and experimentally, and the influence of the number and complexity of atomic tags on the detection performance is verified through experiments.

Keywords: label, data flow control, atomic tag, set correlation, policy redundancy and conflict detection

0 引言

大数据、互联网等信息技术快速发展并广泛普及,使得协同办公、在线教育、社交网络等各类互联网应用用户数量大幅上升。为满足上述需求,数据往往需要在不同应用之间频繁地流转。数据的频繁流转大大增加了信息泄露的风险,如何确保数据全生命周期的受控流转成为数据流转中的关键问题。

基于标签的数据流转控制机制通过给主体和客体设置标签的方式实现对数据的流转控制,具有标签设置灵活、便捷等优点。同时,标签可以随着数据全生命周期流转,实现对数据全生命周期的流转控制,引起人们的广泛关注。在基于标签的数据流转控制机制中,针对如何提高用户便捷性和流转控制粒度等问题,Xie等^[1]提出了一种基于标签的数据流转控制(LBDFC, label based data flow control)机制。该机制从易于用户理解使用、标签丰富度等角度出发,基于对客体的操作,充分考虑了主体、客体、环境等属性进行标

签设计,实现了数据灵活的受控流转。

在数据跨域流转中,会有各种不同新的流转控制需求,标签的增加、修改、删除是基于标签数据跨域流转中经常面临的问题。目前在大多应用场景中,标签设置与变更一般由非安全专业人员操作,在标签变更中,新的标签与已有标签不可避免存在冗余或者冲突。如何解决不同标签的冗余与冲突问题是基于标签的数据流转控制中亟须解决的问题。

如何快速检测不同标签之间是否存在冗余或者冲突,并准确定位存在冗余或者冲突的位置,是设计基于标签的数据流转控制策略冗余与冲突检测方法首要解决的问题。标签的复杂度直接影响检测效率,不同描述方法会大大影响检测性能。简单的描述方式一般检测效率高,但一定程度上会影响策略描述的丰富程度。如何平衡标签描述的丰富度与检测效率是设计基于标签的策略冗余与冲突检测方法面临的挑战。现有的策略冗余与冲突检测方法大多基于图^[2]、树^[3]等,存在复杂度高、检测效率低等问题,而且,通用的冗余与冲

冲突检测方法难以直接迁移到基于标签的流转控制机制中。针对上述问题, 本文从标签描述方法入手, 通过分析标签中不同元素之间集合相关性, 提出了基于标签的策略冗余与冲突检测方法, 主要贡献如下。

1) 提出了基于原子操作的标签描述方法, 基于原子操作生成多个原子标签, 原子标签用于描述最小的安全需求, 客体标签为多个原子标签的逻辑组合, 解决了标签描述丰富性问题。

2) 提出了基于标签的策略冗余与冲突检测方法, 通过分析原子标签中各要素的关系对原子标签进行冗余与冲突检测, 并基于原子标签检测结果和逻辑关系实现对客体标签的检测, 提高了检测效率。

3) 从理论和实验两个角度对基于标签的策略冗余与冲突检测方法进行分析, 评估了原子标签数量和复杂度对检测效率的影响, 验证了检测方法的高效性和可用性。

1 相关工作

基于标签的访问控制具有标签设置灵活、便捷等优点, 同时标签随着数据全生命周期流转, 实现对数据全生命周期的控制, 引起了学术界和工业界的广泛关注。研究成果主要聚焦于模型构建、策略冗余与冲突检测等方面^[4-5]。

在基于标签的访问控制模型构建方面, 不少学者通过引入标签或属性标记的方式构建访问控制模型。李凤华等^[6]提出通过给客体设置隐私标签用于对接收信息的节点进行隐私保护和流转控制, 但文献没有给出形式化定义。王媛等^[7]提出按照主体属性实现访问者和角色间的映射, 并根据客体标签实现角色和权限间的映射, 支持个性化隐私偏好授权, 但该文献存在模型复杂、决策评估效率低等问题。Liu 等^[8]以标记的形式将资源与访问控制列表关联, 提出基于资源和属性的访问控制模型, 但该模型只有允许权限, 导致灵活性低等。Li 等^[9]提出了一种矩阵结构, 并建立了符合矩阵域特征的矩阵安全标签以及安全标签和必要约束的定义, 该文献中的安全标签相对比较复杂, 不易于用户理解和使用。Wang 等^[10]提出了一种基于

安全标签的最小扩散和跨域安全授权策略。基于安全标签属性, 通过引入多级约束机制, 来实现跨域授权。

在策略冗余与冲突检测方面, 也有不少研究成果。戚湧等^[3]采用有向无环图的拓扑排序计算规则合并优先级, 按优先级顺序为每个规则构建一棵空间区域选择树, 完成冲突消解, 但该方法不适用于复杂策略情况。吴迎红等^[11]提出了基于集合求交的递推算法, 以权限赋值单元集合为单位提高策略冲突的计算粒度, 但存在效率不高的问题。Chen 等^[12]利用矩阵描述复杂策略, 通过一致性检查完成冲突检测和消解, 但存在效率不高、空间消耗大的问题。江泽涛等^[13]针对基于属性的访问控制模型存在的静态策略冲突及冗余问题, 提出了一种基于属性集有序化及二进制屏蔽码的静态策略冲突检测算法, 支持属性的增加及策略的增加或删除。但算法在规则数量较大时检测效率较低。Liu 等^[14]通过补充缺失的属性表达式, 将隐式冲突规则转换为显式冲突规则, 再对比所有规则实现对规则集中所有可能的冲突规则, 但该检测方法同样存在规则数量较大时检测效率较低的问题。为提高访问控制策略冗余与冲突检测效率, Chowdhary 等^[15]提出了一种新的面向对象的策略冲突检测解析框架, 按照规则依赖关系创建依赖关系图, 提高策略冲突检测效率, 但该方法仅适用于防火墙、负载均衡器等虚拟网络应用。刘敖迪等^[2]将冲突检测问题转换为图的连通性问题, 提出了一种基于策略生成图模型的冲突检测机制, 但适用场景较为单一。

因此, 在基于标签的数据流转控制机制中, 如何提高复杂流转控制策略冗余与冲突检测的效率是亟待解决的问题。

2 基于标签的数据流转控制机制

本节对基于标签的数据流转过程进行分析, 并对基于标签的数据流转控制机制进行阐述。

2.1 基于标签的数据流转场景

在复杂网络环境下, 数据在流转时往往涉及不同的信息系统、访问主体和访问环境, 数据流转控制需要根据具体应用场景的安全需求设计对

应的控制策略。基于标签的数据流转控制机制通过为客体 and 主体设置不同标签来描述流转控制策略，标签随着客体在不同主体之间流转，实现客体在不同访问主体之间的灵活受控流转。基于标签的数据流转场景如图 1 所示，具体流程如下。

Step 1 数据所有者 (S, subject) S_1 生成客体资源 (O, object)，根据安全需求为其设置标签 OL_1 ，该标签用于对后续接收客体 O 的主体进行约束控制。

Step 2 S_{i-1} 发送带有标签 OL_{i-1} 的客体 O 发送给后面的主体 S_i ($i \geq 2$)。

Step 3 后面的主体 S_i 接收 S_{i-1} 发送的客体 O 及其标签 OL_{i-1} ，并根据客体标签 OL_{i-1} 进行授权控制。

Step 4 当主体 S_i 存在变更标签的权限时，根据新的安全需求生成标签 OL'_i ，并由 OL'_i 和 OL_{i-1} 生成新的标签 OL_i 。

Step 5 如果主体 S_i 有转发权限，则主体 S_i 可以将客体和更新后的标签 OL_i 发送给下一个主体 S_{i+1} 。

上述流转过程中，当所有者 S_1 在给客体设置标签 OL_1 时， OL_1 中会包含多个不同的安全需求，不同安全需求之间会存在冗余或者冲突。除此之外，当主体 S_i 根据新的安全需求生成新的标签 OL'_i 时，一方面，如果 S_i 设定的新的安全需求，在已有的标签 OL_{i-1} 中已经描述，那么根据新的安全需求生成的标签 OL'_i 与 OL_{i-1} 可能存在冗余，导致基于 OL'_i 和 OL_{i-1} 生成的标签 OL_i 表达的冗余信息越来越多，这样不仅增加了通信量，还大大降低了数据流转控制的效率；另一方面，如果 S_i 设定的新的安全需求与已有的标签 OL_{i-1} 中的安全意图相反，那么根据新的安全需求生成的标签 OL'_i 与 OL_{i-1} 存在冲突，基于 OL'_i 和 OL_{i-1} 生成的标签 OL_i 存在冲突，导致基于标签 OL_i 将得不到有效流转控制结果。因此，如何实现标签的冗余与冲突检测，如何提高标签冗余与检测的效率，继而实现数据受控流转以及流转控制的效率，是基于标签的数据流转控制机制中急需解决的问题。本文从标签描述入手，提出基于原子操作的标签描述方法，并基于原子标签的描述方法，通

过分析原子标签中不同要素之间的相关性，提出基于标签的策略冗余与冲突检测方法，实现对客体标签进行冗余与冲突检测。

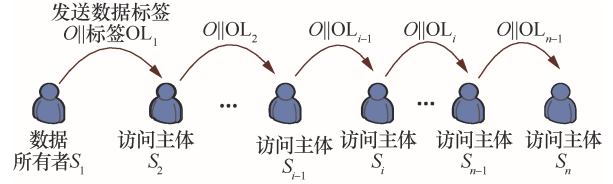


图 1 基于标签的数据流转场景
Figure 1 Label-based data flow scene

2.2 基于原子操作的标签描述方法

标签描述方法是影响基于标签的数据流转控制效率、冗余与冲突检测效率的关键。基于标签的数据流转控制机制表示为六元组 $\langle S, SL, O, OL, OP, P \rangle$ ， S 表示主体， O 表示客体， OP 表示操作， P 表示权限，主体标签 (SL, subject label) 和客体标签 (OL, object label) 用来表示流转控制策略，也就是说通过主体标签和客体标签描述客体流转时，必须满足的条件。在设计标签时，充分考虑了主体、客体、环境等属性，在提高标签描述的丰富性和粒度的同时，尽量降低标签复杂度。

主体标签：用于描述访问主体的属性 (SA, subject attribute)，形式化描述为 $SL = \{SA_S\}$ 。

客体标签：指所有者或者访问主体为客体设置的标签，用于描述对客体进行某种操作必须满足的条件或者在某种条件下禁止对客体进行某种操作。客体标签包括标签类型、操作、主体属性和环境属性。为提高流转控制评估以及冗余与冲突检测效率，客体标签分为多个原子标签的逻辑组合。原子标签是最小的标签，原子标签中包含一种类型、一个操作以及最小属性集合。用 tag_i 表示客体标签 OL 中第 i 个原子标签， $tag_i = \langle op_i, type_i, SA_i, CA_i \rangle$ 。 op_i 表示操作； $type_i$ 表示标签规则类型， $type_i = Y$ or N ， $type_i = Y$ 表示允许操作，反之 $type_i = N$ 表示不允许； SA_i 表示主体属性； CA_i 表示上下文环境。原子标签 tag_i 表示主体属性满足 SA_i 、环境属性满足 CA_i 的情况下，允许/不允许对客体执行操作 op_i 。客体标签由多个原子标签组合生成，可以形式化描述为

$$OL = \{tag_1, \dots, tag_i\} = \{\langle op_1, type_1, SA_1, CA_1 \rangle, \dots, \langle op_i, type_i, SA_i, CA_i \rangle\}$$

将客体标签表示为多个原子标签的逻辑组合，即使没有很多安全知识的普通用户也可以根据需求设置针对客体进行某种操作时必须满足的主体属性、环境属性，或者如果满足某些主体属性、环境属性就不能对客体进行某种操作，大大降低了描述的难度。同时，原子标签只包含一种操作以及最小属性集合，无论控制评估还是策略冗余与冲突检测都相对容易些。

3 基于标签的策略冗余与冲突检测方法

基于上述基于原子操作的标签描述方法，本节提出基于标签的策略冗余与冲突检测方法。该检测方法通过分析客体标签中原子标签不同要素的关系实现。为降低检测难度、提高检测效率，从客体标签包含的原子标签入手，基于不同原子标签中的操作关系进行检测，主要分为以下几种情况。对于包含相同操作的不同原子标签，通过分析不同原子标签中主体属性、环境属性以及规则类型之间关系进行检测。对于包含不同操作的原子标签，如果不同操作之间没有关系，那么不存在冗余与冲突，如果不同操作之间存在偏序关系，则通过分析不同原子标签中操作的偏序关系、主体属性、环境属性以及规则类型之间的关系进行检测。下面详细介绍如何通过分析集合相关性进行策略冗余与冲突检测。

3.1 原子标签的冗余与冲突分析

定义 1 策略冗余：如果两个客体标签表示的策略有重复或部分重复，那么这两个客体标签表示的策略存在冗余。

例如，以下两个原子标签， $\text{tag}_1 = \langle \text{read}, Y, g_1 \cup g_2, \text{Null} \rangle$ 表示允许用户组 g_1 和 g_2 的用户读， $\text{tag}_2 = \langle \text{read}, Y, g_2 \cup g_3, \text{Null} \rangle$ 表示允许用户组 g_2 和 g_3 的用户读，则显然 tag_1 和 tag_2 关于用户组 g_2 的读策略描述存在冗余。

定义 2 策略冲突：如果一个访问请求根据不同客体标签得到相反的授权结果，那么这两个客体标签表示的策略存在冲突。

例如，以下两个原子标签， $\text{tag}_1 = \langle \text{read}, Y, g_1 \cup g_2, \text{Null} \rangle$ 表示允许用户组 g_1 和 g_2 的用户读， $\text{tag}_2 = \langle \text{read}, N, g_2 \cup g_3, \text{Null} \rangle$ 表示不允许用户组 g_2 和 g_3 的用户读，则当 g_2 组的用户提出读操作申请时，显然 tag_1 和 tag_2 的授权结果相反，产生冲突。

在对不同原子标签冗余与冲突检测时，从原子标签包含操作出发，分成相同操作和不同操作进行检测，通过分析不同原子标签中操作间偏序关系、主体属性、环境属性以及规则类型之间关系进行冗余与冲突检测。下面分别详细介绍同操作冗余与冲突检测、不同操作冗余与冲突检测方法。

(1) 操作偏序关系

本文将数据流转操作分为两类：传播操作和运算操作。其中，传播操作包括发送（send）、转发（forward）、接收（receive）；运算操作包括读（read）、写（write）、保存（save）、删除（delete）、变更标签（alter label）、加密（encrypt）、解密（decrypt）、签名（sign）、验签（verify signature）、计算摘要（hash）等。不同操作之间具有一定的偏序关系。

定义 3 操作间的偏序关系：指操作间的包含关系，记为 $\prec$ ， $\text{op}_1 \prec \text{op}_2$ 表示操作 op_2 包含操作 op_1 ，如用户在获得“写（write）”的操作权限时，自然获得了“读（read）”操作权限，则“写”操作包含“读”操作，即 $\text{read} \prec \text{write}$ 。

(2) 集合关系

两个集合之间的相关关系可以分为以下 5 种形式（不相交、相等、A 包含 B、B 包含 A、相交），如图 2 所示。



图2 集合之间的相关关系
Figure 2 The correlation of sets

原子标签的冗余与冲突检测基于原子标签中操作的关系，以及主体属性和环境属性不同集合间相关关系进行。本文将冗余与冲突的检测结果细分为 6 种（完全冲突、部分冲突、无冲突；完全冗余、部分冗余、无冗余），并采用原子标签冗余列表（LR，list_redundancies）和原子标签冲突列表（LC，list_conflicts）来表示两个客体标签中原子标签的检测结果。

原子标签冗余列表：指存在策略冗余的原子标签列表，包括发生冗余的原子标签、冗余的属性集合以及冗余检测结果 result，形式化描述为 $\text{LR} = \{\dots, \text{list_redundancy}_j, \dots\} = \{\dots, \langle \text{tag}_i, \text{result}_i, \text{tag}_j, \text{result}_j, \text{SA},$

$CA_i, \dots\}$ 。其中 $list_redundancy_j = \langle tag_i, result_i, tag_j, result_j, SA, CA \rangle$ 表示原子标签 tag_i 与原子标签 tag_j 存在冗余, SA, CA 表示冗余的主体属性和环境属性, $result_i$ 和 $result_j$ 表示冗余检测结果, $complete$ 表示完全冗余, $part$ 表示部分冗余。

原子标签冲突列表: 指存在冲突的原子标签列表, 包括发生冲突的原子标签、冲突的属性集合、冲突检测结果 $result$, 形式化描述为 $LC = \{\dots, list_conflict_j, \dots\} = \{\dots, \langle tag_i, result_i, tag_j, result_j, SA, CA \rangle, \dots\}$ 。其中, $list_conflict_j = \langle tag_i, result_i, tag_j, result_j, SA, CA \rangle$ 表示原子标签 tag_i 与原子标签 tag_j 存在冲突, SA, CA 表示冲突的主体属性和环境属性, $result_i$ 和 $result_j$ 表示冲突检测结果, $complete$ 表示完全冗余, $part$ 表示部分冗余。

3.2 同操作原子标签的冗余与冲突检测

对于包含相同操作的原子标签, 冗余与冲突检测通过分析原子标签中的主体属性集合 SA 、环境属性集合 CA 之间的相关性及规则类型进行。

对于 $tag_i = \langle op_i, type_i, SA_i, CA_i \rangle$, $tag_j = \langle op_j, type_j, SA_j, CA_j \rangle$ 且 $op_i = op_j$, tag_i 与 tag_j 之间通过规则 1 和规则 2 进行检测。

规则 1 若环境属性 $CA_i = CA_j$, 根据主体属性集合相关性进行冗余与冲突检测。

1) 当两个原子标签主体属性集合不相交时, 即 $SA_i \cap SA_j = \emptyset$, 原子标签 tag_i 和 tag_j 无冗余和冲突。

2) 当两个原子标签主体属性集合相等时, 即 $SA_i = SA_j$, 如果规则类型 $type_i = type_j$, 则原子标签 tag_i 和 tag_j 完全冗余且无冲突, 如果规则类型 $type_i \neq type_j$, 则原子标签 tag_i 和 tag_j 无冗余但完全冲突。

3) 当两个原子标签主体属性集合为包含关系时, 即 $SA_i \subseteq SA_j$, 如果规则类型 $type_i = type_j$, 则原子标签 tag_i 对 tag_j 完全冗余但无冲突, tag_j 对 tag_i 部分冗余但无冲突, 如果规则类型 $type_i \neq type_j$, 则原子标签 tag_i 对 tag_j 无冗余但完全冲突, tag_j 对 tag_i 无冗余但部分冲突。

4) 当两个原子标签主体属性集合相交时, 即 $(SA_i \cap SA_j) \neq SA_i \neq SA_j \neq \emptyset$, 如果规则类型 $type_i = type_j$, 则原子标签 tag_i 和 tag_j 部分冗余但无冲突, 如果规则类型 $type_i \neq type_j$, 则原子标签 tag_i 和 tag_j 无冗余但部分冲突。

规则 2 若环境属性 $CA_i \neq CA_j$, 按环境属性集

合相关性分情况进行冗余与冲突检测。

1) 当两个原子标签环境属性集合不相交时, 即 $CA_i \cap CA_j = \emptyset$, 此时无论主体属性集合相关性如何, 原子标签 tag_i 和 tag_j 无冗余且无冲突。

2) 当两个原子标签环境属性集合为包含关系时, 即 $CA_i \subseteq CA_j$ (或 $CA_j \subseteq CA_i$), 则将环境属性集合更大的原子标签进一步拆分, 即 $tag_j = tag_{j1} + tag_{j2} = \langle op_j, type_j, SA_j, CA_j \rangle + \langle op_j, type_j, SA_j, CA_j - CA_i \rangle$, 依据规则 2 中的 1), tag_{j2} 与 tag_i 无冗余和冲突。 tag_{j1} 与 tag_i 环境属性相同, 则依据规则 1 按主体属性集合相关性进一步检测得到结果。

3) 当两个原子标签环境属性集合相交时, 即 $(CA_i \cap CA_j) \neq CA_i \neq CA_j \neq \emptyset$, 则对两个原子标签进一步拆分, 即 $tag_j = tag_{j1} + tag_{j2} = \langle op_j, type_j, SA_j, CA_j - CA_i \cap CA_j \rangle + \langle op_j, type_j, SA_j, CA_i \cap CA_j \rangle$, $tag_i = tag_{i1} + tag_{i2} = \langle op_i, type_i, SA_i, CA_i - CA_i \cap CA_j \rangle + \langle op_i, type_i, SA_i, CA_i \cap CA_j \rangle$, 其中 tag_{j2} 与 tag_{i2} 环境属性相同, 则依据规则 1 按主体属性集合相关性进一步检测得到结果; 根据规则 2 中 1), tag_{j1} 和 tag_{i1} 、 tag_{j1} 和 tag_{i2} 、 tag_{j2} 和 tag_{i1} , 均无冗余和冲突。

基于规则 1 和规则 2, 本文提出了同操作间策略冗余冲突检测算法, 如算法 1 所示。若两个原子标签的环境属性相同, 则基于原子标签间主体属性集合相关性、规则类型进行冗余与冲突检测; 若两个原子标签的环境属性不同, 则根据原子标签中的环境属性集合相关性分别讨论, 若环境属性集合不相交, 则这两个原子标签没有关系, 既不冗余也不冲突; 若环境属性集合为包含和相交关系, 则根据两个原子标签环境属性的关系, 将原子标签再细分成环境属性相同和不相交的两个原子标签, 环境属性不相交的原子标签既不冗余也不冲突, 环境属性相同的原子标签按照规则 1 进行分析。

算法 1 同操作间策略冗余冲突检测

输入 同操作的原子标签 tag_i 和 tag_j ($op_i = op_j$)

输出 $List_redundancies_A$, 原子标签冗余列表或者 $List_conflicts_A$, 原子标签冲突列表

1) **if** $CA_i = CA_j$ //两个标签的环境属性相同

2) **if** $SA_i = SA_j$ //两个标签的主体属性相同

3) **if** $type_i = type_j$ //两个标签类型相同, 标签完全冗余

4) $\text{list_redundancy_A}_q = \langle \text{tag}_i, \text{complete}, \text{tag}_j, \text{complete}, \text{SA} \rangle;$
 //两个标签类型不同, 标签完全冲突

5) **else** $\text{list_conflict_A}_q = \langle \text{tag}_i, \text{complete}, \text{tag}_j, \text{complete}, \text{SA} \rangle;$
 //环境属性相同, 主体属性为包含关系

6) **else if** $\text{SA}_i \subseteq \text{SA}_j$
 //如果类型相同, 标签 tag_i 相对 tag_j 完全冗余,
 标签 tag_i 相对 tag_j 部分冗余

7) **if** $\text{type}_i = \text{type}_j$

8) $\text{list_redundancy_A}_q = \langle \text{tag}_i, \text{complete}, \text{tag}_j, \text{part}, \text{SA} \rangle;$
 //如果类型不同, 标签 tag_i 相对 tag_j 完全冲突,
 标签 tag_i 相对 tag_j 部分冲突

9) **else** $\text{list_conflict_A}_q = \langle \text{tag}_i, \text{complete}, \text{tag}_j, \text{part}, \text{SA} \rangle;$
 //环境属性相同, 主体属性为相交关系

10) **else if** $(\text{SA}_i \cap \text{SA}_j) \neq \text{SA}_i \neq \text{SA}_j \neq \emptyset$
 //如果类型相同, 部分冗余

11) **if** $\text{type}_i = \text{type}_j$

12) $\text{list_redundancy_A}_q = \langle \text{tag}_i, \text{part}, \text{tag}_j, \text{part}, \text{SA} \rangle;$
 //如果类型不同, 部分冲突

13) **else** $\text{list_conflict_A}_q = \langle \text{tag}_i, \text{part}, \text{tag}_j, \text{part}, \text{SA} \rangle;$
 //如果环境属性为包含关系, 将标签拆分成环境属性相同部分和不同部分分别进行判断

14) **else if** $\text{CA}_i \subseteq \text{CA}_j$

15) $\text{tag}_j = \text{tag}_{j1} + \text{tag}_{j2} = \langle \text{op}_j, \text{type}_j, \text{SA}_j, \text{CA}_j \rangle + \langle \text{op}_j, \text{type}_j, \text{SA}_j, \text{CA}_j - \text{CA}_i \rangle;$
 //如果环境属性为相交关系, 同样将标签拆分成环境属性相同部分和不同部分进行判断

16) **else if** $(\text{CA}_i \cap \text{CA}_j) \neq \text{CA}_i \neq \text{CA}_j \neq \emptyset$

17) $\text{tag}_j = \text{tag}_{j1} + \text{tag}_{j2} = \langle \text{op}_j, \text{type}_j, \text{SA}_j, \text{CA}_j - \text{CA}_i \cap \text{CA}_j \rangle + \langle \text{op}_j, \text{type}_j, \text{SA}_j, \text{CA}_i \cap \text{CA}_j \rangle;$

18) $\text{tag}_i = \text{tag}_{i1} + \text{tag}_{i2} = \langle \text{op}_i, \text{type}_i, \text{SA}_i, \text{CA}_i - \text{CA}_i \cap \text{CA}_j \rangle + \langle \text{op}_i, \text{type}_i, \text{SA}_i, \text{CA}_i \cap \text{CA}_j \rangle$

假设针对同一客体, 现有 3 个原子标签 $\text{tag}_1 = \langle \text{read}, Y, g_1 \cup g_2 \cup g_3, [8:00, 18:00] \rangle$, $\text{tag}_2 = \langle \text{read}, N, g_1, [8:00, 18:00] \rangle$, $\text{tag}_3 = \langle \text{read}, Y, g_2 \cup g_3, [10:00,$

$20:00] \rangle$, 由于 3 个原子标签均针对读操作设置, 按照算法 1 进行冗余冲突检测。

对于 tag_1 和 tag_2 , 由于 $\text{CA}_1 = \text{CA}_2$, $g_1 \subseteq g_1 \cup g_2 \cup g_3$, 即 $\text{tag}_2.\text{SA}_2 \subseteq \text{tag}_1.\text{SA}_1$, 且 $\text{tag}_1.\text{type}_1 \neq \text{tag}_2.\text{type}_2$, tag_2 对于 tag_1 来说无冗余但完全冲突, tag_1 对于 tag_2 无冗余但部分冲突, 冲突部分为主体属性 g_1 。对于 tag_1 和 tag_3 , $\text{CA}_1 \neq \text{CA}_3$, 将 tag_1 和 tag_3 进行拆分, $\text{tag}_1 = \text{tag}_{1.1} + \text{tag}_{1.2} = \langle \text{read}, Y, g_1 \cup g_2 \cup g_3, [8:00, 10:00] \rangle + \langle \text{read}, Y, g_1 \cup g_2 \cup g_3, [10:00, 18:00] \rangle$, $\text{tag}_3 = \text{tag}_{3.1} + \text{tag}_{3.2} = \langle \text{read}, Y, g_2 \cup g_3, [10:00, 18:00] \rangle + \langle \text{read}, Y, g_2 \cup g_3, [18:00, 20:00] \rangle$ 。 $\text{tag}_{1.1}$ 和 $\text{tag}_{3.1}$ 环境属性不相交, 只需对环境属性相同的 $\text{tag}_{1.2}$ 和 $\text{tag}_{3.1}$ 进行判断, 由于主体属性 $(g_2 \cup g_3) \subseteq (g_1 \cup g_2 \cup g_3)$, 且规则类型相同, 因此 $\text{tag}_{3.1}$ 对于 $\text{tag}_{1.2}$ 完全冗余但无冲突, $\text{tag}_{1.2}$ 对于 $\text{tag}_{3.1}$ 部分冗余但无冲突, 冗余部分为主体属性 g_2 和 g_3 。综上, tag_1 和 tag_3 部分冗余但无冲突, 冗余内容为 $\langle \text{read}, Y, g_2 \cup g_3, [10:00, 18:00] \rangle$ 。对于 tag_2 和 tag_3 可以采用类似的方法进行判断。

3.3 不同操作的原子标签的冗余与冲突检测

对于包含不同操作的原子标签, 如果两个操作间没有偏序关系, 那么这两个原子标签不存在冗余和冲突。如果不同操作存在偏序关系, 那么这两个原子标签可能存在冗余或冲突。因此, 在对包含不同操作原子标签进行冗余与冲突检测时, 只需要检测具有偏序关系的不同操作的原子标签之间的关系。下面详细介绍具有偏序关系的不同操作的原子标签冗余与冲突检测方法。

对于 $\text{tag}_i = \langle \text{op}_i, \text{type}_i, \text{SA}_i, \text{CA}_i \rangle$, $\text{tag}_j = \langle \text{op}_j, \text{type}_j, \text{SA}_j, \text{CA}_j \rangle$ 且 $\text{op}_j < \text{op}_i$, tag_i 与 tag_j 之间的冗余冲突通过规则 3 和规则 2 进行检测。

规则 3 若环境属性 $\text{CA}_i = \text{CA}_j$, 则根据主体属性集合相关性进行冗余与冲突检测。

- 1) 当两个原子标签主体属性集合不相交时, 即 $\text{SA}_i \cap \text{SA}_j = \emptyset$, tag_i 和 tag_j 无冗余和冲突。
- 2) 当两个原子标签主体属性集合相等时, 即 $\text{SA}_i = \text{SA}_j$, 如果规则类型 $\text{type}_i = \text{type}_j$, 则 tag_i 对于 tag_j 部分冗余且无冲突, 原子标签 tag_j 对于 tag_i 完全冗余; 如果规则类型 $\text{type}_i \neq \text{type}_j$, 当 $\text{type}_i = Y$ 且 $\text{type}_j = N$ 时, 标签 tag_i 和 tag_j 完全冲突, 当 $\text{type}_i = N$

且 $type_j=Y$ 时, tag_i 和 tag_j 无冗余和冲突。

3) 当两个原子标签主体属性集合为包含关系时, 即 $SA_i \subseteq SA_j$, 如果类型 $type_i=type_j$, 则原子标签 tag_i 对于 tag_j 部分冗余但无冲突, 如果规则类型 $type_i \neq type_j$, 则当 $type_i=Y$ 且 $type_j=N$ 时, 原子标签 tag_i 对于 tag_j 无冗余但完全冲突, tag_j 对于 tag_i 无冗余但部分冲突, 当 $type_i=N$ 且 $type_j=Y$ 时, tag_i 和 tag_j 无冗余和冲突。

4) 当两个原子标签主体属性集合相交时, 即 $(SA_i \cap SA_j) \neq SA_i \neq SA_j \neq \emptyset$, 如果规则类型 $type_i=type_j$, 则原子标签 tag_i 和 tag_j 部分冗余但无冲突, 如果规则类型 $type_i \neq type_j$, 则当 $type_i=Y$ 且 $type_j=N$ 时, tag_i 和 tag_j 无冗余但部分冲突, 当 $type_i=N$ 且 $type_j=Y$ 时, tag_i 和 tag_j 无冗余和冲突。

若环境属性 $CA_i \neq CA_j$, 则按环境属性集合相关性分情况进行冗余与冲突检测。此时参考规则 2, 依据环境属性的关系拆分成环境属性相同部分和不同部分, 对于环境属性相同部分按照规则 3 进行检测, 对于环境属性不同部分, 两个标签无冗余与冲突。

基于规则 3 和规则 2, 本文提出了偏序操作间策略冗余冲突检测算法, 如算法 2 所示。与算法 1 基本思想类似, 同样按照环境属性相同和环境属性不同分别展开检测。若两个原子标签的环境属性相同, 基于原子标签间主体属性集合相关、规则类型以及原子标签操作间的偏序关系进行冗余与冲突检测; 若两个原子标签的环境属性不同, 根据原子标签中的环境属性集合相关性分别讨论, 如果环境属性集合不相交, 那么这两个原子标签没有关系, 既不冗余也不冲突; 如果环境属性集合为包含和相交关系, 那么根据两个原子标签环境属性的关系, 将原子标签再细分成环境属性相同和不相交的两个原子标签, 环境属性不相交的原子标签既不冗余也不冲突, 环境属性相同的原子标签按照规则 3 进行分析。

算法 2 偏序操作间策略冗余与冲突检测

输入 不同操作的原子标签 tag_i 和 tag_j
($op_j \prec op_i$)

输出 List_redundancies_A, 原子标签冗余列表;
List_conflicts_A, 原子标签冲突列表

```

1)  if  $CA_i=CA_j$  //两个标签的环境属性相同
2)      if  $SA_i=SA_j$  //两个标签的主体属性相同
3)          if  $type_i=type_j$ 
                //如果类型相同, 标签  $tag_i$  相对  $tag_j$  部分冗余,
                标签  $tag_i$  相对  $tag_j$  完全冗余
4)              list_redundancy_Aq = <tagi, part,
                tagj, complete, SA>;
                //如果类型不同, tagi 类型为 Y, tagj 类型为 N,
                完全冲突
5)          else if (typei=Y) && (typej=N)
6)              list_conflict_Bq = <tagi, com-
                plete, tagj, complete, SA>;
                //环境属性相同, 主体属性为包含关系
7)          else if  $SA_i \subseteq SA_j$ 
                //如果类型相同, 标签  $tag_i$  对于  $tag_j$  部分冗余
                但无冲突,  $tag_j$  对于  $tag_i$  部分冗余但无冲突
8)              if  $type_i=type_j$ 
9)                  list_redundancy_Aq = <tagi,
                part, tagj, part, SA>;
                //如果类型不同, tagi 类型为 Y, tagj 类型为 N,
                完全冲突
10)             else if (typei=Y) && (typej=N)
11)                 list_conflict_Bq = <tagi,
                complete, tagj, part, SA>;
                //环境属性相同, 主体属性为相交关系
12)             else if  $(SA_i \cap SA_j) \neq SA_i \neq SA_j \neq \emptyset$ 
                //如果类型相同, 部分冗余
13)                 if  $type_i=type_j$ 
14)                     list_redundancy_Aq = <tagi,
                part, tagj, part, SA>;
                //如果类型不同, tagi 类型为 Y, tagj 类型为 N,
                部分冲突
15)                 if (typei=Y) && (typej=N)
16)                     list_conflict_Bq = <tagi, part,
                tagj, part, SA>;
                //如果环境属性为包含关系, 将标签拆分成环
                境属性相同部分和不同部分, 分别进行判断
17)             else if  $CA_i \subseteq CA_j$ 
18)                 tagj = tagj1+tagj2 = <opj, typej, SAj,
                CAi> + <opj, typej, SAj, CAj-CAi>;
                //如果环境属性为相交关系, 同样将标签拆分

```

成环境属性相同部分和不同部分进行判断

19) **else if** $(CA_i \cap CA_j) \neq CA_i \neq CA_j \neq \emptyset$

20) $tag_j = tag_{j1} + tag_{j2} = \langle op_j, type_j, SA_j, CA_j - CA_i \cap CA_j \rangle + \langle op_j, type_j, SA_j, CA_i \cap CA_j \rangle;$

21) $tag_i = tag_{i1} + tag_{i2} = \langle op_i, type_i, SA_i, CA_i - CA_i \cap CA_j \rangle + \langle op_i, type_i, SA_i, CA_i \cap CA_j \rangle;$

假设针对同一客体, 现有两个原子标签 $tag_4 = \langle read, N, g_1 \cup g_2, [8:00, 18:00] \rangle$ 和 $tag_5 = \langle write, Y, g_1, [10:00, 20:00] \rangle$, 由于 $read \prec write$, $[8:00, 18:00] \cap [10:00, 20:00] = [10:00, 18:00]$, 即 $(CA_4 \cap CA_5) \neq CA_4 \neq CA_5 \neq \emptyset$, 将 tag_4 和 tag_5 进行拆分, $tag_4 = tag_{4.1} + tag_{4.2} = \langle read, N, g_1 \cup g_2, [8:00, 10:00] \rangle + \langle read, N, g_1 \cup g_2, [10:00, 18:00] \rangle$, $tag_5 = tag_{5.1} + tag_{5.2} = \langle write, Y, g_1, [10:00, 18:00] \rangle + \langle write, Y, g_1, [18:00, 20:00] \rangle$. 对环境属性相同的 $tag_{4.2}$ 和 $tag_{5.1}$ 进一步判断, 由于主体属性 $g_1 \subseteq (g_1 \cup g_2)$, 且规则类型 $tag_4.type_4 = N$, $tag_5.type_5 = Y$, 因此根据规则 3, $tag_{4.2}$ 对于 $tag_{5.1}$ 无冗余但部分冲突, $tag_{5.1}$ 对于 $tag_{4.2}$ 无冗余但完全冲突, 冲突部分为主体属性 g_1 . 综上, tag_4 和 tag_5 无冗余但部分冲突, 冲突内容分别为 $\langle read, N, g_1, [10:00, 18:00] \rangle$ 和 $\langle write, Y, g_1, [10:00, 18:00] \rangle$.

基于标签的数据流转控制机制描述了各种不同的流转控制策略。客体标签冗余与冲突消解基于客体标签包括的原子标签的检测结果、原子标签的优先级以及不同原子标签的逻辑关系实现。在实际应用过程中, 原子标签的优先级可以根据原子标签设置的顺序、设置原子标签主体的优先级确定。例如, 按照主体的等级确定原子标签的优先级, 主体等级高, 其设置的原子标签对应的优先级也相对比较高。原子标签的逻辑关系依据不同的安全需求由主体进行设置。

4 实验与分析

本节从理论和实验两个角度对 LBDLC 策略冗余与冲突检测方法的性能进行分析。

4.1 理论分析

本文提出的基于标签的数据流转控制策略由客体标签描述, 客体标签通过原子标签的逻辑组合生成。在对客体标签进行冗余与冲突检测时, 首先分析客体标签中不同原子标签是否存在冗余

与冲突, 基于原子标签的检测结果以及原子标签的逻辑关系, 得到不同客体标签是否存在冗余与冲突。对不同原子标签进行检测时, 从原子标签包含的操作出发, 按照相同操作和不同操作展开, 通过分析不同原子标签中操作间的偏序关系、主体属性、环境属性以及规则类型之间的关系进行冗余与冲突检测, 并且将属性相关性分析归结为数学中的集合间关系判定问题, 有效地降低了检测难度。本文提出的策略冗余与冲突检测最终归结为判断集合间关系这种简单的数学问题, 大大降低了策略冗余与冲突检测的复杂度。

基于标签的策略冗余与冲突检测主要的时间消耗在于原子标签的检测。假设客体标签 OL_1 和 OL_2 分别包含 n_1 和 n_2 个原子标签, 要分析 OL_1 和 OL_2 是否存在冗余与冲突, 首先分析 OL_1 中的 n_1 个原子标签和 OL_2 中的 n_2 个原子标签是否存在冗余与冲突。最简单的情况是, OL_1 中的 n_1 个原子标签包含的操作和 OL_2 中的 n_2 个原子标签包含的操作不相同也不存在偏序关系, 那么原子标签不存在冗余与冲突, 客体标签 OL_1 和 OL_2 也不存在冗余与冲突。最复杂的情况是, OL_1 中的原子标签与 OL_2 中的原子标签包含的操作, 要么相同要么具有偏序关系, 需要将 OL_1 中的 n_1 个原子标签与 OL_2 中的 n_2 个原子标签一一进行检测分析, 也就是要做 $(n_1 n_2) / 2$ 次检测。原子标签的检测调用算法 1 或算法 2, 算法 1 或算法 2 从原子标签中包含的主体属性、环境属性以及规则类型的关系进行分析, 也就是说, 进行一次原子标签冗余与冲突检测, 要进行 3 次集合关系的分析。根据上面的分析, 客体标签 OL_1 和 OL_2 冗余与冲突检测最复杂的情况是进行 $1.5n_1 n_2$ 次集合关系的分析。

4.2 实验仿真

基于标签的数据流转控制机制通过标签来描述流转控制需求, 标签具有轻量级的特点, 冗余与冲突检测的性能相比其他策略描述方法具有明显优势。为进一步验证本文提出的标签描述方法以及检测方法的性能优势, 下面通过实验将本文方法与基于标签的访问控制 (LBAC, label based access control) 策略的冗余冲突检测性能进行对比分析。其中, LBAC 通过 XACML 语言描述访问控制策略, 本文通过原子标签的方式描述控制

策略。实验时,两类标签表达相同的安全需求,通过调整原子标签数量、原子标签复杂度进行分析和对比。本文在 Windows 10 操作系统上利用 C 语言编写测试代码进行实验,硬件环境为 Intel Core i5-6200U CPU 2.40 GHz, 8 GB RAM。实验分别测试了同操作、具有偏序关系的不同操作原子标签冗余冲突检测性能,对检测方法性能进行验证。

下面通过实验测试原子标签的数量和标签复杂度对策略冗余与冲突检测效率的影响。原子标签复杂度主要受属性类型和属性值影响,下面实验以标签中属性类型的数量表示标签的复杂度,记为 λ 。为便于对比,在实验时假设环境属性类型的数量不变, λ 的变化体现在主体属性类型的数量上,原子标签总数记为 Num。实验 1 测试同操作原子标签的冗余与冲突检测方法的性能。实验采取控制变量法进行两组实验。第一组实验在保证原子标签复杂度不变的条件下,也就是在原子标签包含的属性类型数量不变的情况下,通过改变原子标签的数量,将 LBAC 的策略冗余与冲突检测方法的性能与本文所提方法进行对比,分析原子标签数量对冗余与冲突检测方法性能的影响。第二组实验在原子标签数量一定的条件下,通过调整原子标签中属性的复杂度进行,也就是通过调整原子标签类型的数量进行,将本文所提方法与 LBAC 的策略冗余与冲突检测方法的性能进行对比,分析原子标签复杂度对冗余与冲突检测性能的影响。

实验 1 测试同操作下标签数量对冗余与冲突检测效率的影响。第一组实验取原子标签复杂度 $\lambda=1$ 不变,也就是说主体属性中只包含一种类型,对原子标签总数 Num=50、100、150、200 分别进行测试,分析标签中属性的数量对冗余与冲突检测效率的影响。第二组实验取原子标签总数 Num=50 不变,分别对复杂度 $\lambda=1、2、3、4、5$ 进行测试,也就是说主体属性中分别包含 1、2、3、4、5 种类型的属性。第二组实验测试分析标签中属性类型数量对冗余与冲突检测效率的影响。实验结果如图 3 和图 4 所示。

实验 2 的思路与实验 1 基本相同,区别在于实验 2 中原子标签的操作不同,不同操作之间存在偏序关系。实验结果如图 5 和图 6 所示。

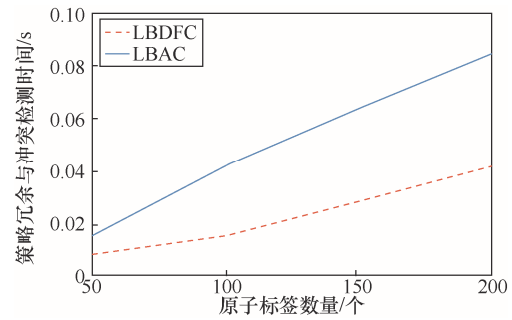


图 3 同操作原子标签冗余冲突检测时间与原子标签总数间的关系
Figure 3 The relationship between the redundant conflict detection time and the number of atomic tags containing same operation

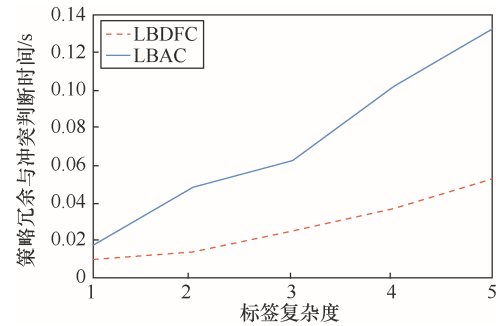


图 4 同操作原子标签冗余冲突检测时间与标签复杂度间的关系
Figure 4 The relationship between the redundant conflict detection time and the complexity of atomic tags containing same

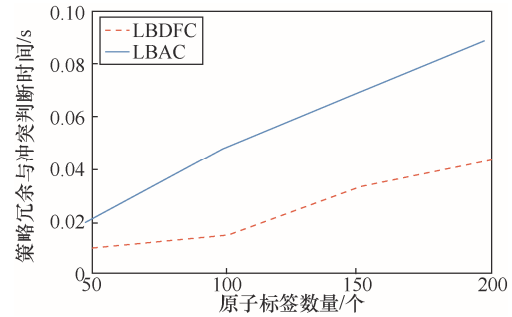


图 5 不同操作原子标签冗余冲突检测时间与原子标签总数间的关系
Figure 5 The relationship between the redundant conflict detection time and the number of atomic tags containing different operations

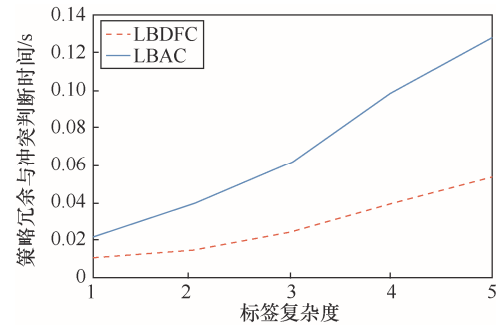


图 6 不同操作原子标签冗余冲突检测时间与标签复杂度间的关系
Figure 6 The relationship between the redundant conflict detection time and the complexity of atomic tags containing different operations

从实验 1 和实验 2 的结果来看,本文提出的

基于标签的数据流转控制策略冗余与冲突检测方法消耗的时间与原子标签数量和复杂度呈线性关系, 消耗时间远低于 LBAC 策略冗余与冲突检测方法, 而且随着标签复杂度和数量增加, 这种优势越发明显。实验 1 和实验 2 测试的是最复杂的情况, 所有原子标签包含的操作要么相同, 要么具有偏序关系, 实际中检测的性能会比实验高。这种性能优势最关键在于 LBDFC 将客体标签划分为单一操作的原子标签, 通过集合相关性对原子标签进行冗余与冲突检测。而其他 LBAC 策略冗余与冲突检测方法, 标签描述的复杂度与原子标签不是同一个数量级, 性能会远低于 LBDFC 策略冗余与冲突检测方法。验证了原子标签数量和复杂度对检测性能的影响。未来将针对物联网、社交网络等不同场景下的不同安全需求对本文方法进行仿真验证。

5 结束语

基于标签的数据流转控制机制通过主客体标签实现数据流转控制, 具有轻量级、延伸控制的优势, 受到广泛关注。在客体流转中, 标签修改、增加不可避免, 不同标签之间不可避免地出现冗余与冲突, 如何快速高效检测不同标签之间是否存在冗余与冲突, 是基于标签的数据流转控制中急需解决的问题。针对上述问题, 本文从标签描述方法入手, 提出了基于原子操作的标签描述方法。该方法基于操作生成多个原子标签, 原子标签用于描述最小的安全需求, 客体标签是多个原子标签的逻辑组合, 解决了标签描述简洁性和丰富性问题。在策略冗余与冲突检测时, 基于原子标签检测结果和逻辑关系实现对客体标签的检测。原子标签分为同操作和不同操作两类, 通过分析原子标签中各要素的关系对原子标签进行冗余与冲突检测, 提高了检测效率。从理论和实验两个角度对本文方法进行分析, 验证了原子标签数量和复杂度对检测性能的影响。验证了原子标签数量和复杂度对检测性能的影响。未来将针对物联网、社交网络等不同场景下的不同安全需求对本文方法进行仿真验证。

参考文献:

- [1] XIE R N, FAN X N, ZHU J Y, et al. Research on label based data flow control mechanism[C]//2021 IEEE Sixth International Conference on Data Science in Cyberspace (DSC). 2021: 233-239.
- [2] 刘敖迪, 王娜, 刘磊. 面向云服务组合的策略冲突检测机制[J]. 计算机应用研究, 2017, 34(4): 1145-1150.
LIU A D, WANG N, LIU L. Policy conflict detection mechanism for cloud service composition[J]. Computer Application Research, 2017, 34(4): 1145-1150.
- [3] 戚湧, 陈俊, 李千目. 基于冗余消除和属性数值化的 XACML 策略优化方法[J]. 计算机科学, 2016, 43(2): 163-168.
QI Y, CHEN J, LI Q M. XACML policy optimization method based on redundancy elimination and attribute numeralization[J]. Computer Science, 2016, 43(2): 163-168.
- [4] 吴泽智, 陈性元, 杨智, 等. 信息流控制研究进展[J]. 软件学报, 2017, 28(1): 135-159.
WU Z Z, CHEN X Y, YANG Z, et al. Research progress on information flow control[J]. Journal of Software, 2017, 28(1): 135-159.
- [5] 诸天逸, 李风华, 成林, 等. 跨域访问控制技术研究[J]. 网络与信息安全学报, 2021, 7(1): 20-27.
ZHU T Y, LI F H, CHENG L, et al. Research on cross-domain access control technology[J]. Journal of Network and Information Security, 2021, 7(1): 20-27.
- [6] 李风华, 谢绒娜, 李晖, 等. 一种信息流转方法、装置及系统: 中国, CN109347845B[P]. 2020-08-07.
LI F H, XIE R N, LI H, et al. An information transfer method, device and system: China, CN109347845B[P]. 2020-08-07.
- [7] 王媛, 孙宇清, 马乐乐. 面向社交网络的个性化隐私策略定义与实施[J]. 通信学报, 2012, 33(S1): 239-249.
WANG Y, SUN Y Q, MA L L. Definition and implementation of personalized privacy policies for social networks[J]. Journal on Communication, 2012, 33(S1): 239-249.
- [8] LIU G, FANG L, WANG Q, et al. Resource and attribute based access control model for system with huge amounts of resources[C]//International Conference on Green, Pervasive and Cloud Computing. 2018.
- [9] LI L Q, HAI L. An access control model based on matrix domain security label[J]. IOP Conference Series: Materials Science and Engineering, 2021, 1043(4).
- [10] WANG Y T. A cross-domain authorization access control model based on security label attribute[J]. Journal of Physics: Conference Series, 2021, 1955(1).
- [11] 吴迎红, 黄皓, 周靖康, 等. 分布式应用访问控制策略精化冲突分析[J]. 计算机应用, 2014, 34(2): 421-427.
WU Y H, HUANG H, ZHOU J K, et al. Refinement conflict analysis of distributed application access control policy[J]. Computer Applications, 2014, 34(2): 421-427.
- [12] CHEN M, HONG J, XIAO Y, et al. A method of conflict detection and resolution for security policy based on matrix description[C]//2017 7th IEEE International Conference on Electronics Information and Emergency Communication(ICEIEC). 2017: 548-551.
- [13] 江泽涛, 谢朕, 王琦, 等. 一种基于屏蔽码的 ABAC 静态策略冲突与冗余检测算法[J]. 计算机科学, 2018, 45(2): 197-202.
JIANG Z T, XIE Z, WANG Q, et al. An ABAC static policy conflict and redundancy detection algorithm based on masking code[J].

Computer Science, 2018, 45(2): 197-202.

[14] LIU G, PEIW, TIAN Y, et al. A novel conflict detection method for ABAC security policies[J]. Journal of Industrial Information Integration, 2021, 22: 100200.

[15] CHOWDHARY A, SABUR A, HUANG D, et al. Object oriented policy conflict checking framework in cloud networks (OOPC)[J]. IEEE Transactions on Dependable and Secure Computing, 2021.

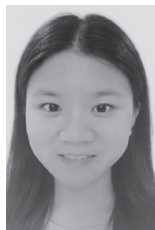
[作者简介]



谢绒娜（1976—），女，山西永济人，博士，北京电子科技学院教授，主要研究方向为网络与系统安全、访问控制、密码工程。



李苏浙（1999—），女，浙江绍兴人，北京电子科技学院硕士生，主要研究方向为访问控制。



黄宇欣（1999—），女，湖南长沙人，北京电子科技学院硕士生，主要研究方向为信息安全、访问控制。



范晓楠（1997—），女，河北邢台人，北京电子科技学院硕士生，主要研究方向为信息安全、访问控制。



史国振（1974—），男，河南济源人，北京电子科技学院教授、博士生导师，主要研究方向为密码信息安全、信息论与编码理论。