面向云服务组合的策略冲突检测机制*

刘敖迪[1,2]，王 娜[1,2]，刘 磊[3]
(1. 解放军信息工程大学，郑州 450001；2. 数学工程与先进计算国家重点实验室，郑州 450001；3. 信息保障技术重点实验室，北京 100000)

摘 要：针对云服务组合的策略冲突问题，研究了云服务中属性间关系及组件服务间组合关系的特点，提出了基本类型冲突、层次关系冲突、互斥关系冲突、组合关系约束冲突四种冲突类型；设计了能够直观表达策略中多种关系的策略生成图模型，该模型具有结构灵活、易于更新和动态扩展的优点；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，实现了对云环境下大规模策略集中冲突策略的高效检测。最后，开展仿真实验，验证、评估了检测机制的有效性和检测效率。

关键词：云服务；服务组合；访问控制；ABAC；策略冲突检测
中图分类号：TP393 **文献标志码**：A **文章编号**：1001-3695(2017)04-1145-06
doi:10.3969/j.issn.1001-3695.2017.04.043

Policy conflict detection mechanism for cloud services composition

Liu Aodi[1,2], Wang Na[1,2], Liu Lei[3]
(1. *PLA Information Engineering University*, *Zhengzhou* 450001, *China*; 2. *State Key Laboratory of Mathematical Engineering & Advanced Computing*, *Zhengzhou* 450001, *China*; 3. *Science & Technology on Information Assurance Laboratory*, *Beijing* 100000, *China*)

Abstract: To solve the problem of policy conflict under cloud services composition environment, according to the attributes relationship of cloud composite service and the invoking relationship of the component services, this paper proposed the basic conflict, hierarchical conflict, exclusive conflict, composite constraint conflict. It designed a policy generated graph model to express access control policy set of cloud services composition. This model can show the attributes relationship and the invoking relationship of the component services well. It was flexible, easy to update and extensible. The conflict problem was transformed to the connectivity problem of graphs. This research designed a policy generated graph model based conflict detection mechanism and achieved efficient conflict detection. It can meet the needs of large-scale policy set conflict detection under cloud environment. Finally, simulation verified and evaluated the effectiveness and performance of the method.

Key words: cloud services; service composition; access control; ABAC; policy conflict detection

0 引言

作为云计算[1]的主要服务形式，云服务得到了非常广泛的应用[2]。单一的云服务只能提供有限的功能，通过云组合服务[3]实现不同云服务功能的集成，能够灵活、有效地满足不同的应用需求。但是，由此也带来了很多新的安全问题，特别是云组件服务间访问控制策略的冲突问题，将直接影响云组合服务的有效性与安全性。针对 Web 服务组合过程中的访问控制策略冲突问题，Yan 等人[4]通过深入分析组合 Web 服务访问控制模型 CWS-RBAC 中的角色传播、继承关系及服务组合过程中约束关系对授权策略的影响，提出了一种面向组合 Web 服务访问控制策略冲突检测的形式化方法。但是，该方法基于 RBAC 模型，组合服务与组件服务间通过全局角色与局部角色的映射关系来进行访问控制，当某组件服务退出、更新或其策略发生改变时，对所有存在映射关系的组件服务策略都要进行同步的更新，增加了检测的开销。Wu 等人[5]采用时序逻辑方法对 Web 组合服务的冲突进行动态分析，能够有效检测出一定时间内因策略动态更新导致的冲突。但该方法没有考虑组件服务之间的时序调用关系，并且由于缺少对策略内实体属性的分析，丢失了实体之间的层次关系，无法检测出因实体间层次关系和组件服务时序调用关系导致的冲突。Kamoda 等人[6]针对 Web 服务组合的策略冲突问题，提出一种基于 Free Variable Tableaux 的检测机制。该机制采用谓词逻辑表达策略，将逻辑运算符转换为树型关系，同时提出了冲突策略的修正方案，以有效检测模态冲突与静态职责分离冲突。Turkmen 等人[7]通过对主体用户、用户组进行规则的划分，利用二叉决策树进行冲突策略的检测。王雅哲等人[8]、吴迎红等人[9]在资源语义树、策略精化树的策略索引基础上，提出了多种冲突类型并利用状态相关性给出了相应规则的冲突检测算法，但该方法忽略了环境约束对策略冲突的影响。同时，由于其冲突检测模型基于树进行构建，策略的每次更新都将导致多个节点关联规则的更新，提高了检测成本。

云服务不同于 Web 服务[10]。根据提供服务的层次，云服务包括 IaaS 服务、PaaS 服务和 SaaS 服务；不同层次服务的用户群体不同，PaaS 和 IaaS 适用于企业级用户，SaaS 与 Web 服务相似，多应用于企业级与个体级用户；付费模式不同，云服务提供更灵活的按需使用计费，Web 服务采用固定收费模式。

收稿日期：2016-03-08；**修回日期**：2016-05-05 **基金项目**：国家“863”计划基金资助项目(2015AA011705,2012AA012704)
作者简介：刘敖迪(1992-)，男，黑龙江伊春人，硕士研究生，主要研究方向为云计算、网络信息安全(ladyexue@163.com)；王娜(1980-)，女，副教授，博士，主要研究方向为复杂系统环境下的信任管理；刘磊(1979-)，男，工程师，硕士研究生，主要研究方向为信息安全.

可见,与 Web 服务相比,云组合服务策略冲突检测在灵活性、效率上需要满足更高的要求。

本文研究采用 ABAC 模型(基于属性的访问控制模型:attribute based access control)。该模型可满足云服务动态、细粒度的访问控制需求,且兼容其他访问控制模型(如 DAC、MAC、RBAC 等)[11,12]。在深入分析策略内属性间关系及云组合服务内各组件服务间组合关系的基础上,本文提出了基本类型冲突、层次关系冲突、互斥关系冲突、组合关系约束冲突四种策略冲突类型;设计了一种基于策略生成图的冲突检测方法,该方法将冲突检测问题转换为图模型的连通检测问题,可直观表达、有效检测上述四种策略冲突类型;同时,利用生成图内属性节点灵活、可扩展的特性,易于实现策略的动态更新;在检测过程中可根据不同冲突类型的特点灵活选取初始节点开始检测,实现了对策略集的分布式并行处理;通过去除冗余策略,有效提高了检测效率,能够满足云环境下对大规模策略冲突检测的需要。

需要说明的是虽然已有学者提出基于图模型的冲突检测方法,例如姚键等人[13]提出了一种基于有向图模型的冲突检测机制,但对实体间关系及约束考虑不足;Prakash 等人[14]利用图模型实现合成策略的表述和冲突策略的自动消解,但是,其研究对象是不同层次的网络策略。

1 策略冲突类型

1.1 术语与符号定义

ABAC 模型的基本元素,包括访问请求者、被访问的资源、访问的动作及约束条件,统一采用属性表示,即主体属性(用 s 表示)、资源属性(用 r 表示)、动作属性(用 a 表示)、环境属性(用 e 表示)。属性由属性名来唯一标志,具有与之对应的属性值及其属性范围。为便于下文叙述,给出以下定义。

定义 1 属性项 它是表示属性的基本单元,用(xATTR, sub_ser$_i$),$x \in \{s,r,a,e\}$ 表示。其中,xATTR 表示该属性的属性名及其属性值,sub_ser$_i$ 表示拥有此属性的组件服务标志集合。

(sATTR, sub_ser$_i$),(rATTR, sub_ser$_i$),(aATTR, sub_ser$_i$),(eATTR, sub_ser$_i$)分别代表主体属性项、资源属性项、动作属性项、环境属性项。每种类型的属性可分别存在多个,因此引入属性元组的概念。

定义 2 属性元组 它是同类型属性项的集合,用 xATTR_SET,$x \in \{s,r,a,e\}$ 表示。即 xATTR_SET = {(xATTR$_1$, sub_ser$_1$),(xATTR$_2$, sub_ser$_2$),…,(xATTR$_i$, sub_ser$_i$)}。

定义 3 属性授权项 属性授权项 policy 指一条访问控制策略,且 policy = {⟨sATTR_SET⟩,⟨rATTR_SET⟩,⟨aATTR_SET⟩,⟨eATTR_SET⟩,sign},其中 xATTR_SET 不能为空集,$x \in \{s,r,a\}$,sign 表示授权结果,且 sign ∈ {" +"," -"},其中," +"表示授权结果为允许(正向授权)," -"表示授权结果为拒绝(负向授权)。由于同一属性授权项中,各属性项是针对同一组件服务进行授权管理,其组件服务标记必然相同。

定义 4 访问控制策略集:由一个或多个属性授权项所组成的集合。

1.2 属性间关系

1)分配关系 该关系存在于主体属性、资源属性、动作属性、环境属性之间,表述了属性授权项内属性之间的对应关系,使用符号⇔表示。

如属性授权项 policy = [⟨sATTR_SET⟩,⟨rATTR_SET⟩,⟨aATTR_SET⟩,⟨eATTR_SET⟩,sign]。

存在分配关系如下:⟨sATTR_SET⟩⇔⟨rATTR_SET⟩,⟨sATTR_SET⟩⇔⟨aATTR_SET⟩和⟨rATTR_SET⟩⇔⟨aATTR_SET⟩,⟨aATTR_SET⟩⇔⟨eATTR_SET⟩。

2)层次关系 该关系分为聚合层次关系与继承层次关系。聚合层次关系存在于资源属性间,若某一资源由多个子资源聚合而成,则该资源与子资源间存在聚合层次关系;继承层次关系存在于主体属性之间,主体存在用户、用户组、角色等属性,主体的角色属性之间可能存在继承层次关系。使用符号↦表示层次关系。

如 $r\text{ATTR}_1 \mapsto r\text{ATTR}_2$、$r\text{ATTR}_1 \mapsto r\text{ATTR}_3$ 表示 $r\text{ATTR}_1$ 由 $r\text{ATTR}_2$ 和 $r\text{ATTR}_3$ 聚合而成;$s\text{ATTR}_{\text{role2}} \mapsto s\text{ATTR}_{\text{role1}}$ 表示 $s\text{ATTR}_{\text{role2}}$ 是 $s\text{ATTR}_{\text{role1}}$ 的上层角色,继承了 $s\text{ATTR}_{\text{role1}}$ 角色的权限。

3)互斥关系 存在于资源属性之间,使用符号↔进行表示。若某些资源不可同时被同一主体操作,则称这些资源属性之间存在互斥关系;互斥关系为访问控制的职责分离原则提供了保障。如 $r\text{ATTR}_1 \leftrightarrow r\text{ATTR}_2$ 表示属性 $r\text{ATTR}_1$ 与 $r\text{ATTR}_2$ 之间存在互斥关系。

4)约束关系 是环境属性对主体、资源、动作属性的约束,使用符号⇒进行表示。如 $e\text{ATTR}_1 \Rightarrow s\text{ATTR}_1$、$e\text{ATTR}_2 \Rightarrow r\text{ATTR}_1$、$e\text{ATTR}_3 \Rightarrow a\text{ATTR}_1$ 分别表示了 $e\text{ATTR}_1$、$e\text{ATTR}_2$、$e\text{ATTR}_3$ 对 $s\text{ATTR}_1$、$r\text{ATTR}_1$、$a\text{ATTR}_1$ 的约束。

1.3 云服务组合关系

组合云服务间主要存在以下四种组合关系,如图 1 所示。

1)时序控制关系 表示组合服务内部各组件服务必须按照时序关系顺序调用执行,其结构如图 1(a)所示。

2)并发控制关系 表示组合服务内部各组件服务能够并行、同步执行,相互之间不存在直接的依赖关系,其结构如图 1(b)所示。

3)选择控制关系 根据组合服务的选择约束条件,执行相对应的组件服务。若所有条件都未满足,则执行默认的组件服务,其结构如图 1(c)所示。

4)循环控制关系 当循环执行约束条件满足时执行循环体内部包含的各组件服务;当循环执行条件不满足时,执行相对应的组件服务,其结构如图 1(d)所示。

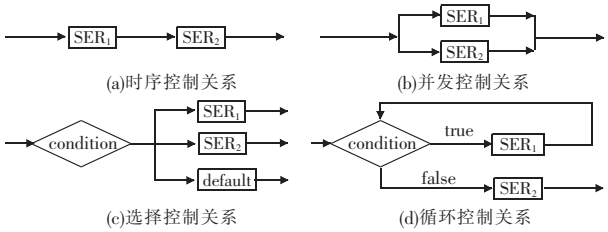


图 1 服务组合关系图

1.4 面向云服务组合的策略冲突类型

根据 1.3 节定义的云服务属性间关系和服务组合关系,本文提出以下四种策略冲突类型:

a)基本类型冲突。基本类型冲突指存在 $n(n \geq 2)$ 个属性授权项 {policy$_i$},其中 policy$_i$ = {⟨sATTR_SET$_i$⟩,⟨rATTR_SET$_i$⟩,⟨aATTR_SET$_i$⟩,⟨eATTR_SET$_i$⟩,sign$_i$},$(1 \leq i \leq n)$,如果这 n 个属性授权项的主体属性元组、资源属性元组与动作属性元组相交,而策略授权结果不相同,那么这 n 个属性授权项之间产生基本类型冲突。

b)层次关系冲突。包括主体层次冲突、资源层次冲突和

混合层次冲突三类。

(a)主体层次冲突指策略内的具有继承层次关系的主体针对同一资源进行相同动作,授权结果不同时产生的冲突。(b)资源层次冲突指主体对于存在聚合层次关系的客体资源进行相同动作,授权结果不同时产生的冲突。(c)混合层次冲突指具有继承层次关系主体对存在聚合层次关系的客体资源进行相同动作,授权结果不同时产生的冲突。

对于属性授权项 $\text{policy}_i = \{\langle s\text{ATTR\_SET}_i\rangle, \langle r\text{ATTR\_SET}_i\rangle, \langle a\text{ATTR\_SET}_i\rangle, \langle e\text{ATTR\_SET}_i\rangle, \text{sign}_i\}\ (1 \leqslant i \leqslant 2)$。其中,$(s\text{ATTR}_{\text{role1}}, \text{sub\_ser}_1) \in s\text{ATTR\_SET}_1$,$(s\text{ATTR}_{\text{role2}}, \text{sub\_ser}_1) \in s\text{ATTR\_SET}_2$,$(r\text{ATTR}_1, \text{sub\_ser}_1) \in r\text{ATTR\_SET}_1$,$(r\text{ATTR}_2, \text{sub\_ser}_1) \in r\text{ATTR\_SET}_2$,若存在 $s\text{ATTR}_{\text{role1}} \mapsto s\text{ATTR}_{\text{role2}}$,$r\text{ATTR\_SET}_1 \cap r\text{ATTR\_SET}_2 \neq \varnothing$,$a\text{ATTR\_SET}_1 \cap a\text{ATTR\_SET}_2 \neq \varnothing$,且 $\text{sign}_1 \neq \text{sign}_2$,则说明 policy_1 策略与 policy_2 策略发生主体层次冲突。若 $r\text{ATTR}_1 \mapsto r\text{ATTR}_2$,$s\text{ATTR\_SET}_1 \cap s\text{ATTR\_SET}_2 \neq \varnothing$,$a\text{ATTR\_SET}_1 \cap a\text{ATTR\_SET}_2 \neq \varnothing$,且 $\text{sign}_1 \neq \text{sign}_2$,则说明 policy_1 与 policy_2 发生资源层次冲突。若 $s\text{ATTR}_{\text{role1}} \mapsto s\text{ATTR}_{\text{role2}}$,$r\text{ATTR}_1 \mapsto r\text{ATTR}_2$,$a\text{ATTR\_SET}_1 \cap a\text{ATTR\_SET}_2 \neq \varnothing$,且 $\text{sign}_1 \neq \text{sign}_2$,则说明产生混合层次冲突。

c)互斥关系冲突。它是指主体属性,针对互斥的资源属性进行相同动作时,授权结果同时为“+”所产生的冲突。该冲突导致主体权限过大,违背了职责分离原则,对系统安全产生威胁。

对于属性授权项 $\text{policy}_i = \{\langle s\text{ATTR\_SET}_i\rangle, \langle r\text{ATTR\_SET}_i\rangle, \langle a\text{ATTR\_SET}_i\rangle, \langle e\text{ATTR\_SET}_i\rangle, \text{sign}_i\}, (1 \leqslant i \leqslant 2)$,若存在属性关系 $r\text{ATTR}_1 \in r\text{ATTR\_SET}_1$,$r\text{ATTR}_2 \in r\text{ATTR\_SET}_2$,$s\text{ATTR\_SET}_1 \cap s\text{ATTR\_SET}_2 \neq \varnothing$,$a\text{ATTR\_SET}_1 \cap a\text{ATTR\_SET}_2 \neq \varnothing$,$\text{sign}_i = \text{"+"}$,且 $r\text{ATTR}_1 \leftrightarrow r\text{ATTR}_2$,则说明产生互斥关系冲突。

d)组合关系约束冲突。包括时序调用约束冲突和上下文约束冲突两类。

时序调用约束冲突是指策略的服务运行时间与组件服务间时序控制关系不一致所产生的冲突。

上下文约束冲突是指策略的环境属性与组件服务间选择或循环控制关系中约束条件 condition 不一致所产生的冲突。

属性授权项 $\text{policy}_i = \{\langle s\text{ATTR\_SET}_i\rangle, \langle r\text{ATTR\_SET}_i\rangle, \langle a\text{ATTR\_SET}_i\rangle, \langle e\text{ATTR\_SET}_i\rangle, \text{sign}_i\}, (1 \leqslant i \leqslant 2)$。每一个属性项都具有服务标记,使用 c_ser 表示组件服务,其中 $c\_\text{ser}_1 \in x\text{ATTR\_SET}_1$,$c\_\text{ser}_2 \in x\text{ATTR\_SET}_2$,$x \in \{s,r,a,e\}$。$c\_\text{ser}_1$ = 生产服务,c_ser_2 = 送货服务,若 $(e\text{ATTR}_{\text{time1}}, \text{sub\_ser}_1) \in e\text{ATTR\_SET}_1$,$(e\text{ATTR}_{\text{time2}}, \text{sub\_ser}_2) \in e\text{ATTR\_SET}_2$,$c\_\text{ser}_1$ 的服务时间 $e\text{ATTR}_{\text{time1}} = \{10{:}00 - 14{:}00\}$,$c\_\text{ser}_2$ 的服务时间 $e\text{ATTR}_{\text{time2}} = \{10{:}00 - 11{:}00\}$,根据组合服务内调用时序关系得出,送货服务必须在生产服务完成之后进行,而对于 policy_1 与 policy_2 来说,送货服务在生产服务之前被调用是允许的,产生时序调用约束冲突。

若选择或循环控制关系中约束条件 condition 中规定 c_ser_1 的运行时间 $\text{TIME}_{c\_\text{ser}_1} = \{13{:}00 - 17{:}00\}$,由于 $e\text{ATTR}_{\text{time1}} = \{10{:}00 - 14{:}00\}$,则在约束条件 condition 外调用服务是允许的,产生上下文约束冲突。

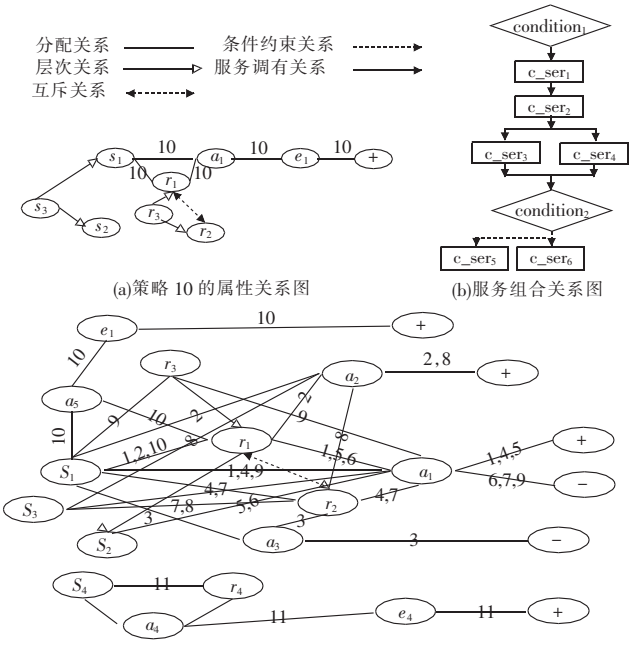
2　基于策略生成图的冲突检测机制

2.1　访问控制策略的策略生成图模型

使用图 $G(V,E)$ 表示访问控制策略属性间及服务组合间的关系,称其为策略生成图,包括属性关系图与服务组合关系图。其中 V 表示图中节点集,节点类型包括属性项节点、授权结果节点、组件服务节点与约束条件节点,E 表示生成图的边集合,边的类型包含分配关系、层次关系、互斥关系、时序调用关系和条件约束关系。节点之间通过边集合内边元素进行连接,分配关系边连接访问控制策略内部的主体属性、资源属性、动作属性、环境属性节点,用边上的权值来记录该边所属的策略编号;层次关系边连接存在聚合层次关系的资源节点和继承层次关系的主体节点;互斥关系边连接互斥属性节点;时序调用关系边连接组件服务节点表示组合关系;条件约束关系边将组件服务节点与约束条件节点相连。为了表述方便,在图表中用符号 s、r、a、e 分别表示主体属性节点、资源属性节点、动作属性节点、环境属性节点,$x.\ x\text{ATTR}(x \in \{s,r,a,e\})$ 表示属性节点属性项中的属性,c_ser 表示组件服务节点,condition 表示约束条件节点,符号“+”“-”表示正向授权结果与负向授权结果节点。

在构建策略生成图时,将一个策略内的 s、r、a 三节点之间都用分配关系边进行连接,使模型结构变得灵活,可根据冲突特点以不同属性节点为起点进行冲突检测,有效提高检测效率。将待检测策略集转换为策略生成图,就是将策略内部各属性及授权结果拆分成节点,通过属性内的关系边将节点连接形成属性关系图,以及根据组合服务之间的组合关系及条件约束构建服务组合关系图的过程。

在策略生成图模型中,节点与边的概念与前文中定义的属性、属性间关系以及云服务组合关系内容一一对应,能够实现没有策略语义丢失的同步转换,即通过策略生成图的形式来对访问控制策略进行表达,图模型能够一致性地满足策略的相关定义要求。图 2 给出了表 1 所示访问控制策略集合的策略生成图。其中,图 2(a)表示了策略编号为 10 的属性授权项 $\{s_1, r_1, a_5, e_1, +\}$ 的属性关系图。



(a)策略 10 的属性关系图　(b)服务组合关系图

(c)策略集的属性关系图

图 2　策略生成图

2.2　冲突在策略生成图中特点分析

以图 3 为例对冲突在策略生成图中的特点进行分析。

1)基本类型冲突　表 1 策略(5)(6)存在基本类型冲突,

即主体 s_2 对资源 r_1 的动作 a_1 正向授权与负向授权同时存在，如图 3(a)所示。可见，当主体、资源、动作属性之间存在有相同策略编号的边相连通，且授权结果“+”“-”同时存在时，发生基本类型冲突。

表 1 访问控制策略集

策略编号	策略内容	策略编号	策略内容
①	$\{s_1, r_1, a_1, +\}$	②	$\{s_1, r_1, a_2, +\}$
③	$\{s_1, r_2, a_3, -\}$	④	$\{s_1, r_2, a_1, +\}$
⑤	$\{s_2, r_1, a_1, +\}$	⑥	$\{s_2, r_1, a_1, -\}$
⑦	$\{s_3, r_2, a_1, -\}$	⑧	$\{s_3, r_2, a_2, +\}$
⑨	$\{s_1, r_3, a_1, -\}$	⑩	$\{s_1, r_1, a_5, e_1, +\}$
⑪	$\{s_4, r_4, a_4, e_4, +\}$		

属性关系：$s_3 \mapsto s_1, s_3 \mapsto s_2, r_3 \mapsto r_1, r_3 \mapsto r_2, r_1 \leftrightarrow r_2$；
属性与组件服务关系：$c\_ser_1 \in \{s_1, s_2, s_3, r_1, r_2, r_3, a_1, a_2, a_3, a_5, e_2\}$，$c\_ser_2 \in \{s_4, r_4, a_4, e_1\}$；
属性值：$e_1 = \{10:00-11:00\}$，$e_4 = \{10:00-14:00\}$；
约束条件 $condition_1$：$TIME_{c\_ser_1} = \{7:00-10:00\}$。

2)层次关系冲突 层次关系包括面向资源的聚合层次关系与面向主体的继承层次关系。表 1 策略(1)(9)中 $r_3 \mapsto r_1$，主体 s_1 对资源 r_3、r_1 的动作 a_1 同时存在正向授权与负向授权，产生聚合层次冲突；表 1 策略(4)(7)中 $s_3 \mapsto s_1$ 存在继承层次关系，对资源 r_2 的动作 a_1 同时存在正向授权与负向授权，产生继承层次冲突，如图 3(b)(c)所示。若将存在层次关系的属性节点合并为一个“节点”，层次关系冲突将转换为基本类型冲突。

3)互斥关系冲突 表 1 策略(1)(4)构造的策略生成图如图 3(d)所示，$r_1 \leftrightarrow r_2$，主体 s_1 同时对互斥资源 r_1、r_2 的动作 a_1 存在正向授权，产生互斥关系冲突。

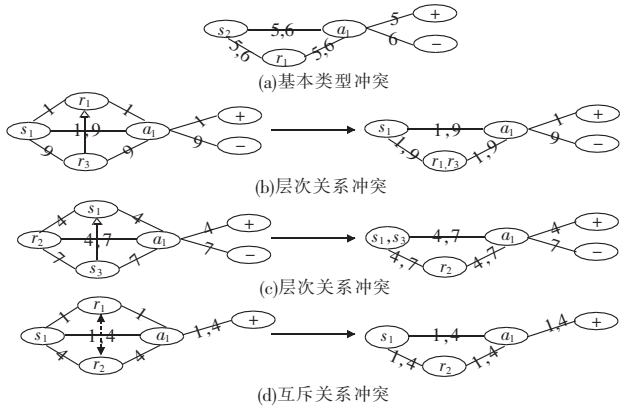


图 3 冲突关系图

将互斥资源属性节点合并为一个“资源节点”，主体、资源、动作属性之间存在有相同策略编号的边互相连通，且授权结果为“+”时，即为互斥关系冲突。

4)组合关系约束冲突 表 1 中策略(10)(11)存在 $c\_ser_1 \in \{s_1, r_1, a_5, e_1\}$，$c\_ser_2 \in \{s_4, r_4, a_4, e_4\}$，若 $s_1.sATTR = s_4.sATTR, r_1.rATTR = r_4.rATTR, a_4.aATTR = a_5.aATTR, e_1 = \{10:00-11:00\}$ 与 $e_4 = \{10:00-14:00\}$ 表示策略执行时间。可见，策略(10)(11)存在相同主体、资源、动作属性并且具有一致的授权结果，但分属于两个不同的组件服务，这时查看如图 2(b)的服务组合关系图，发现 c_ser_2 需在 c_ser_1 之后进行调用，但 $e_4 \wedge e_1 \neq \varnothing$，允许 c_ser_2 在 c_ser_1 之前被调用，违背了服务组合关系，产生时序调用约束冲突。由于全局服务存在约束条件 $condition_1$ 为 $TIME_{c\_ser_1} = \{7:00-10:00\}$，$e_1 \wedge TIME_{c\_ser_1} \neq \varnothing$，在约束条件 $condition_1$ 外调用服务是允许的，产生上下文约束冲突。

2.3 基于策略生成图的冲突检测算法

2.3.1 冲突检测算法

算法 1：基本类型冲突检测算法

算法流程：从策略生成图的主体属性节点出发，若通过具有相同策略编号的边，存在与该节点构成全连通关系的资源与动作属性节点，且同时存在“+”和“-”，则产生的冲突为基本类型冲突，标记冲突类型为 1，如图 4 所示。

算法 2：层次关系冲突检测算法

算法流程：从策略生成图的主体属性节点出发，若存在与当前属性节点相连具有层次关系的主体属性节点，将这两个节点合并当做策略生成图的一个临时主体属性节点，若通过具有相同策略编号的边，存在与该临时节点构成全连通关系资源属性节点与动作属性节点的同时，存在“+”和“-”，则产生继承层次冲突，标记冲突类型为 2，如图 4 所示。

从策略生成图的资源属性节点出发，若当前资源属性节点具有存在层次关系的相连资源节点，将两个节点合并为图中临时资源节点，若通过具有相同策略编号的边，存在与该临时节点构成全连通关系的主体属性节点与动作属性节点的同时，存在“+”和“-”，则产生聚合层次冲突，标记冲突类型为 2，如图 4 所示。

算法 3：互斥关系冲突检测算法

算法流程：从策略生成图的资源属性节点出发，若当前资源属性节点存在互斥资源节点，将两个节点合并作为图中临时资源节点进行冲突检测，若通过具有相同策略编号的边，存在与该临时节点构成全连通关系的主体属性节点与动作属性节点的同时，存在“+”，产生互斥关系冲突，标记冲突类型为 3，如图 4 所示。

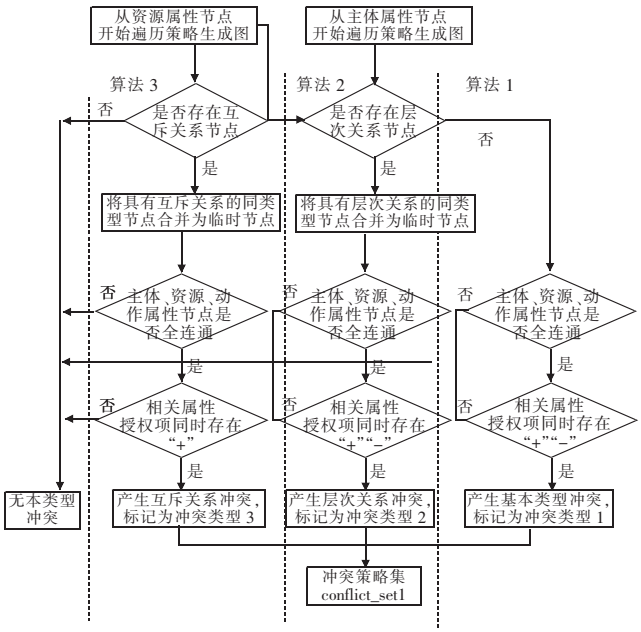


图 4 算法 1~3 流程

算法 4：组合关系约束冲突检测算法

对于服务组合关系的冲突检测主要分两部分进行。第一步，确定服务时序调用关系是否存在冲突，从策略生成图的主体属性节点出发，查看节点属性项中的组件服务标记，确定当前节点属于哪一组件服务，通过查询组合服务调用关系图中服务之间的调用关系，与不同策略环境属性中时序约束条件进行对比，不一致时产生时序调用约束冲突，标记冲突类型为 4-1；

第二步,比较策略中环境属性节点与其约束条件 condition 是否一致,若不一致,产生上下文约束冲突,标记冲突类型为 4-2。冲突检测算法流程如图 5 所示。

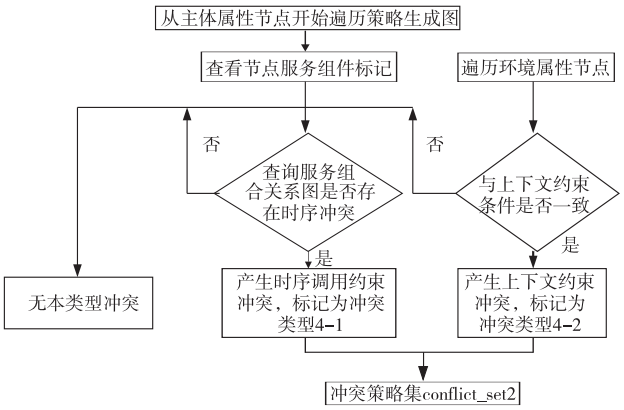


图5　算法4流程

2.3.2　冲突的并行检测

由图 4、5 可知,算法流程中基本类型冲突、继承层次冲突与组合关系约束冲突都是从主体属性节点出发进行检测,聚合层次关系冲突与互斥关系冲突都是从资源属性节点出发开始检测。两种检测过程相对独立,可将检测过程进行拆分,分别作为两个计算单元进行检测,实现冲突的并行检测,提高检测效率。

2.3.3　策略去冗余处理

在冲突检测过程中对结果不产生判决影响的策略,为冗余策略。由于在策略集合中可能存在大量的冗余策略,这些策略将严重影响冲突检测效率,通过策略的去冗余,能够提高检测算法的处理性能,减少响应时延。

1)基本类型冗余

对于属性授权项 $\text{policy}_i = \{\langle s\text{ATTR\_SET}_i\rangle, \langle r\text{ATTR\_SET}_i\rangle, \langle a\text{ATTR\_SET}_i\rangle, \langle e\text{ATTR\_SET}_i\rangle, \text{sign}_i\}$,$(1 \leqslant i \leqslant 2)$,若存在 $x\text{ATTR\_SET}_1 = x\text{ATTR\_SET}_2$,$x \in \{s, r, a, e\}$,且 $\text{sign}_1 = \text{sign}_2$,即 policy_1 与 policy_2 互为基本类型冗余策略。

2)层次类型冗余

由主体属性的继承关系和资源聚合关系能够导致层次类型冗余。对于 $\text{policy}_i = \{\langle s\text{ATTR\_SET}_i\rangle, \langle r\text{ATTR\_SET}_i\rangle, \langle a\text{ATTR\_SET}_i\rangle, \langle e\text{ATTR\_SET}_i\rangle, \text{sign}_i\}$,$(1 \leqslant i \leqslant 2)$,如果存在 $s\text{ATTR}_{\text{role1}} \in s\text{ATTR\_SET}_1$,$s\text{ATTR}_{\text{role2}} \in s\text{ATTR\_SET}_2$,$x\text{ATTR\_SET}_1 = x\text{ATTR\_SET}_2$,$x \in \{r, a, e\}$,且 $\text{sign}_1 = \text{sign}_2$,若 $s\text{ATTR}_{\text{role1}} \mapsto s\text{ATTR}_{\text{role2}}$,则策略 policy_2 为策略 policy_1 的主体层次冗余策略。若 $r\text{ATTR}_1 \mapsto r\text{ATTR}_2$,$r\text{ATTR}_1 \in r\text{ATTR\_SET}_1$,$r\text{ATTR}_2 \in r\text{ATTR\_SET}_2$,$x\text{ATTR\_SET}_1 = x\text{ATTR\_SET}_2$,$x \in \{s, a, e\}$,则 policy_2 为 policy_1 的资源层次冗余策略。

冗余处理过程伴随策略生成图的构建进行,算法流程如图 6 所示。

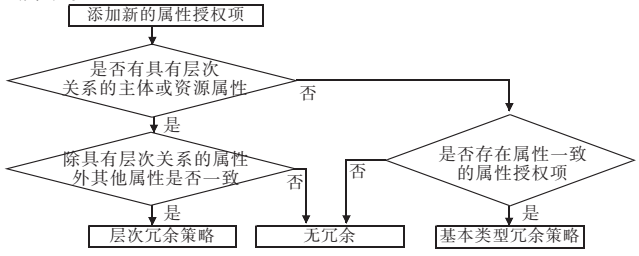


图 6　冗余处理算法流程

在策略生成图模型中,每条访问控制策略,都能转换为与之对应的属性授权项,属性授权项由主体、资源、动作、环境属性结合而成,每一个属性在图中都存在一个与之对应的节点,节点之间的边直观表达了策略中属性间的多种关系,从而,将冲突检测问题转换为图模型下的连通性检测问题;另一方面,策略生成图的结构不同于策略逻辑语义中固定的结构关系,策略内属性节点 s、r、a 互相存在关系边,检测初始点可以动态设定,每条策略能够根据冲突特点从主体属性或资源属性出发进行检测,结构灵活,可实现冲突的分布并行检测。当系统对策略进行更新操作时,只需要对相应策略下的相关节点进行更新即可。同时,通过对策略进行去冗余处理,删除策略集中的冗余策略,能够有效降低检测时延。

3　实验与分析

本章通过仿真实验来验证基于策略生成图的策略冲突检测算法的准确性及其效率。选择在英特尔 Intel® Core™ i7-4710MQ CPU @2.50 GHz 四核处理器、8.00 GB 内存、运行 64 位 Windows7 系统的 PC 机上进行仿真实验。

实验模拟了对云组合服务的冲突检测过程。随机构建由 100、200、500、1 000、2 000 条策略组成的待检测策略集,主体属性集中存在 100 项主体属性,其中 20 项属性存在层次关系;资源属性集由 200 项资源属性构成,其中 40 项属性存在层次、互斥属性关系;定义五种动作属性以供测试;所有属性分属于六项不同组件服务,组件服务的调用关系如图 2(b)所示。使用一元数组表示策略,如[1,1,1,1,0]可以表示$\{s_1, r_1, a_1, e_1, -\}$。基本类型冲突、层次关系冲突、互斥关系冲突、组合关系约束冲突分别对应冲突类型 1 ~ 4。

本文选取三组对比实验,实验 1 对待检测策略集进行冲突检测,验证算法是否能够有效检测出四种类型冲突,通过对比四种不同冲突类型策略的数量,分析得到在不同策略规模下不同类型冲突对检测结果的影响。实验 2 用于评估不同规模策略集的检测效率以及冗余处理对算法效率的影响。实验 3 评估分布式检测对算法效率的影响。

实验 1:检测算法的有效性

对待检测的策略集进行冲突检测,分析本文算法对不同冲突类型检测结果的有效性。如图 7(a)所示算法能够有效地检测出四种不同类型的冲突,随着策略数量的增加,其中类型 1、3 和 4 的曲线陡增程度不是很高,走向较为平缓,因为这三种类型的冲突都由单一因素导致,而类型 2 冲突由层次关系产生,策略规模越大,使得本不存在冲突的策略具有了产生冲突的可能,冲突规模成倍增加。如图 7(b)所示,单独类型 1 冲突的曲线平缓,而加入类型 2 后曲线有明显增幅,说明类型 2 在总冲突中占有较高的比重。

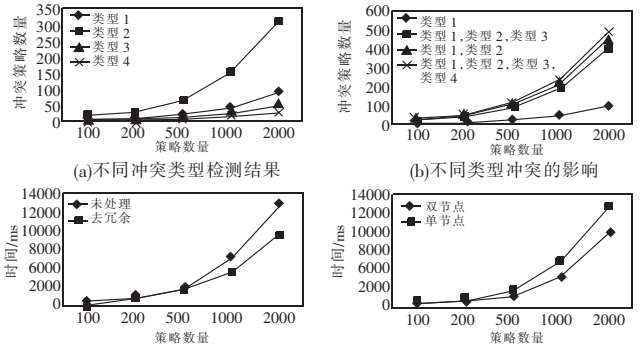
实验 2:策略规模及冗余处理对检测效率的影响

分析不同策略规模对算法性能的影响,比较冲突检测所需时延;对比去冗余处理后的待检测策略集与未进行处理的策略集对不同数量规模的策略检测所需时延,分析冗余策略对系统检测性能的影响。如图 7(c)所示,随着策略规模的增长,检测所需时间不断增加,但未呈爆发式增加。当策略数量在较低水平时,经过去冗余处理检测所需时间比未处理所需时间更多,但随着策略规模的增长,采用去冗余处理能够有效降低系统时延,提高检测效率。这是因为在低策略数量下,相较于较少的

冲突检测总时延,去冗余处理带来了较大的系统损耗,而当策略规模激增,存在大量的冗余策略时,冗余处理才能更有效地提高检测效率。在实际的检测过程中,可分情况判断是否需要进行冗余处理。

实验 3:分布式检测对效率的影响

通过对比单节点与双节点分布式检测所需时延,分析分布式检测对系统效率的影响。如图 7(d)所示,在低策略规模情况下,双节点检测对系统效率提升不明显;但随着策略规模的增长,双节点分布式检测比单节点检测节约的时间也越来越多,能够达到提升检测效率的要求。



(a)不同冲突类型检测结果 (b)不同类型冲突的影响

(c)策略规模及冗余处理的影响 (d)多节点分布处理的影响

图 7 实验结果

4 结束语

本文针对云服务组合提出了一种基于策略生成图的访问控制策略冲突检测机制,其检测过程动态、灵活、高效。实验表明该机制能够对基本类型冲突、层次关系冲突、互斥关系冲突、组合关系约束冲突四种策略冲突类型进行有效的检测。通过冗余与分布式处理,在较大规模策略检测过程中,能够有效提高检测效率。能够满足云计算环境下对大规模策略集进行冲突检测的需要,为安全人员制定正确的安全策略,提供重要参考。由于本文只针对策略冲突的检测过程,下一步将对冲突策略的消解问题开始进一步的研究。

参考文献:

[1] Armbrust M, Fox A, Griffith R, *et al*. A view of cloud computing[J]. Communications of the ACM, 2010, 53(4): 50-58.

[2] Dikaiakos M D, Katsaros D, Mehra P, *et al*. Cloud computing: distributed internet computing for IT and scientific research[J]. IEEE Internet Computing, 2009, 13(5): 10-13.

[3] Jula A, Sundararajan E, Othman Z. Cloud computing service composition: a systematic literature review[J]. Expert Systems with Applications, 2014, 41(8): 3809-3824.

[4] Yan Danfeng, Huang Junlin, Tian Yuan, *et al*. Policy conflict detection in composite Web services with RBAC[C]//Proc of IEEE International Conference on Web Services. [S. l.]: IEEE Press, 2014: 534-541.

[5] Wu Zhengping, Liu Yuanyao. Dynamic policy conflict analysis for collaborative web services[C]// Proc of the 6th International Conference on Network and Service Management. [S. l.]: IEEE Press, 2010: 338-341.

[6] Kamoda H, Yamaoka M, Matsuda S, *et al*. Policy conflict analysis using free variable tableaux for access control in Web services environments[C]//Proc of Policy Management for the Web Workshop. [S. l.]: IEEE Press, 2005: 5-12.

[7] Turkmen F, Foley S, O'Sullivan B, *et al*. Explanations and relaxations for policy conflicts in physical access control[C]//Proc of the 25th IEEE International Conference on Tools with Artificial Intelligence. [S. l.]: IEEE Press, 2013: 330-336.

[8] 王雅哲,冯登国. 一种 XACML 规则冲突及冗余分析方法[J]. 计算机学报, 2009, 32(3): 516-530.

[9] 吴迎红,黄皓,吕庆伟,等. 基于开放逻辑 R 反驳计算的访问控制策略精化[J]. 软件学报, 2015, 26(6): 1534-1556.

[10] Sun Le, Dong Hai, Hussain F K, *et al*. Cloud service selection: state-of-the-art and future research directions[J]. Journal of Network and Computer Applications, 2014, 45(10): 134-150.

[11] Yuan E, Tong Jin. Attributed based access control (ABAC) for Web services[C]//Proc of IEEE International Conference on Web Services. [S. l.]: IEEE Press, 2005: 561-569.

[12] 王小明,付红,张立臣. 基于属性的访问控制研究进展[J]. 电子学报, 2010, 38(7): 1660-1667.

[13] 姚键,茅兵,谢立. 一种基于有向图模型的安全策略冲突检测方法[J]. 计算机研究与发展, 2005, 42(7): 1108-1114.

[14] Prakash C, Lee J, Turner Y, *et al*. PGA: using graphs to express and automatically reconcile network policies[C]//Proc of ACM Conference on Special Interest Group on Data Communication. [S. l.]: ACM Press, 2015: 29-42.

(上接第 页)

[11] 徐小琳,云晓春,周勇林,等. 基于特征聚类的海量恶意代码在线自动分析模型[J]. 通信学报, 2013, 189(8): 146-153.

[12] Narudin F A, Feizollah A, Anuar N B, *et al*. Evaluation of machine learning classifiers for mobile malware detection[J]. Soft Computing, 2016, 20(1): 343-357.

[13] Tahan G, Rokach L, Shahar Y. Mal-ID: automatic malware detection using common segment analysis and meta-features[J]. Journal of Machine Learning Research, 2012, 13(1): 949-979.

[14] Zhang Wei, Ren Huan, Jiang Qingshan, *et al*. Exploring feature extraction and ELM in malware detection for Android devices[M]//Advances in Neural Networks. Switzerland: Springer International Publishing, 2015: 489-498.

[15] James M K, Kalyan V. Deep feature synthesis: towards automating data science endeavors[C]//Proc of IEEE International Conference on Data Science & Advanced Analytics. [S. l.]: IEEE Press, 2015: 1-10.

[16] Ahmadi M, Giacinto G, Ulyanov D, *et al*. Novel feature extraction, selection and fusion for effective malware family classification[C]// Proc of the 6th ACM Conference on Data and Application Security and Privacy. New York: ACM Press, 2016: 183-194.

[17] 王蕊,苏璞睿,杨轶,等. 一种抗混淆的恶意代码变种识别系统[J]. 电子学报, 2011, 39(10): 2322-2330.

[18] Xu Weilin, Qi Yanjun, Evans D. Automatically evading classifiers, a case study on PDF malware classifiers[C]//Proc of Network and Distributed System Security Symposium. 2016: 1-15.

[19] 李挺,董航,袁春阳,等. 基于 Dalvik 指令的 Android 恶意代码特征描述及验证[J]. 计算机研究与发展, 2014, 51(7): 1458-1466.

[20] Krawczyk B, Woźniak M. Evolutionary cost-sensitive ensemble for malware detection[J]. Advances in Intelligent Systems & Computing, 2014, 299(1): 433-442.

[21] 孔德光,谭小彬,奚宏生,等. 提升多维特征检测迷惑恶意代码[J]. 软件学报, 2011, 22(3): 522-533.

[22] Kolter J Z, Maloof M A. Learning to detect malicious executables in the wild[J]. Journal of Machine Learning Research, 2004, 7(4): 470-478.

[23] Holland B, Deering T, Kothari S, *et al*. Security toolbox for detecting novel and sophisticated Android malware[C]//Proc of the 37th International Conference on Software Engineering. [S. l.]: IEEE Press, 2015: 733-736.

[24] Manku G S, Jain A, Das Sarma A. Detecting near-duplicates for Web crawling[C]//Proc of the 16th International Conference on World Wide Web. New York: ACM Press, 2007: 141-150.

[25] 张华伟,王明文,甘丽新. 基于随机森林的文本分类模型研究[J]. 山东大学学报:理学版, 2006, 41(3): 5-9.