基于描述逻辑的 RB-RBAC 授权规则冲突检测方法*)

于海波　车海燕　金淳兆
(吉林大学计算机科学与技术学院　长春 130012)

摘　要　RB-RBAC(Rule-Based RBAC)模型克服了 RBAC 模型的一些局限,提供了基于用户属性自动指派角色的机制。为了检测 RB-RBAC 模型的策略冲突,提出了一种基于描述逻辑的 RB-RBAC 模型的形式化方法,在此基础上提出了一种检测有关规则间冲突的方法、一种发现无关规则间冲突的方法和在授权规则集合中检测不同类型冲突的方法,可以根据具体情况选择不同的方法以提高效率。并给出了一种简单的冲突消解方法。

关键词　RB-RBAC,描述逻辑,授权规则,策略冲突,冲突检测

Research on Description Logic Based Conflict Detection Methods for RB-RBAC Model

YU Hai-Bo　CHE Hai-Yan　JIN Chun-Zhao
(College of Computer Science and Technology, Jilin University, Changchun 130012)

Abstract　RB-RBAC(Rule-Based RBAC) provides the mechanism to dynamically assign users to roles based on a finite set of authorization rules defined by the enterprise's security policy. These rules may have conflict due to negative authorization. We propose a formalization of RB-RBAC by description logic language ALC, and then represent conflict detection method based on knowledge base consistency. Some different methods are suggested to detect conflict among related rules and that among unrelated rules, and they may cooperatively work in one system to provide more efficient detecting service. We also give a simple method to rewrite conflict rules for eliminating policy conflict.

Keywords　RB-RBAC, Description logic, Authorization rule, Policy conflict, Conflict detection

1　引言

近年来,随着网络应用的逐步深入,安全问题日益受到关注。基于角色的访问控制(RBAC)的出现越来越显示出替代传统的 DAC 和 MAC 模型的优势[1,2]。但是,传统的 RBAC 模型中,用户-角色指派通常由管理员手工完成,故在用户数量相当大的系统中,为用户指派角色的工作将成为极沉重的负担,而且正确性难以保证。文[3~5]中定义的 RB-RBAC(Rule-Based RBAC)模型克服了 RBAC 模型的一些局限,提供了一种基于授权规则为用户动态指派角色授权机制。授权规则考虑用户的属性和一些安全约束,使得系统可以自动地根据规则指派和撤消用户的角色,极大地减轻了系统管理员的工作负担,并保证用户-角色指派结果的正确性。

RB-RBAC-ve 模型[5]扩展了 RB-RBAC 基本模型,形成了 RB-RBAC 模型族,允许定义否定授权规则,这为系统提供了额外的安全保证,但是这种方式也可能在进行授权过程中导致冲突。文[5]分析了 RB-RBAC 模型规则间存在的冲突类型,并提出了几种冲突消解策略,但是没有给出冲突的检测方法。文[6~8]提出了一些逻辑方法进行策略冲突的检测,但是一般都不具备完整的推理引擎,而且不支持 RB-RBAC 中复杂的属性表达式定义,无法有效进行 RB-RBAC 中策略冲突的检测和发现。

本文研究了基于描述逻辑(Description Logic, DL)[9]对 RB-RBAC 模型进行表示和推理的方法,能够描述 RB-RBAC 模型中的主要元素和关系,除了能够进行访问控制推理外,利用描述逻辑提供的一致性推理服务,提出了几种检测冲突的方法,并给出了一种简单的策略冲突消解方案。

2　RB-RBAC 模型

RB-RBAC 模型[3,5]中的用户、角色和许可是直接从 RBAC96[1]引入的。RB-RBAC 模型在修改了 RBAC 模型的基础上以授权规则的形式自动进行用户-角色指派。授权规则具有如下形式:$rule_k: ae_k \Rightarrow r_1, \cdots, r_n$,其中 ae_k 是一个属性表达式,$r_1, \cdots, r_n$ 是可分配角色。授权规则表示,如果用户满足 ae_k,那么该用户就会被自动分配角色 $r_1, \cdots, r_n$。属性表达式实际确定了一个满足具体条件的用户集合。RB-RBAC 模型允许定义否定形式的授权规则,如 $ae_k \Rightarrow \neg r_i$,表示如果用户满足 ae_k,那么禁止该用户获得角色 r_i。

为了比较两个授权规则,以判定这两个授权规则间存在的关系,文[3]中引入了优先级(Seniority Levels)的概念,基于属性表达式比较来描述授权规则间的关系。用符号"$\geqslant$"(读作"优先于")来表示授权规则间的优先级:

$$rule_j \geqslant rule_k \leftrightarrow (ae_j \rightarrow ae_k)$$

其中 ae_j,ae_k 分别是对应于 $rule_j$,$rule_k$ 的属性表达式。这种规则间的优先级,如果用户满足规则 $rule_j$,那么也将满足规则 $rule_k$,因此该用户也被授予规则 $rule_k$ 所产生的角色。

引入否定授权规则将会引起相关规则间冲突和无关规则间冲突[5]。在图 1 中,冲突可能发生在不相关的规则间,如

*)国家自然科学基金项目(60173006)、国家高技术研究发展计划项目(2003AA118020)、"吉林大学'985'工程"项目。**于海波**　讲师,博士生,主要研究方向:软件工程、信息安全;**车海燕**　助教,博士生,主要研究方向:自动推理、信息安全;**金淳兆**　教授,博士导师,主要研究方向:软件工程。

$rule_1$ 和 $rule_2$。如果用户 u 同时满足 $rule_1$ 和 $rule_2$，那么 u 被同时要求分配角色 r_1 和禁止分配角色 r_1，这种情况称之为无关规则间冲突。冲突也可能发生在相关的规则间，如 $rule_2$ 和 $rule_4$。如果用户 u 满足 $rule_2$，那么 u 禁止分配角色 r_1。但根据 $rule_2 \geqslant rule_4$，用户 u 又被要求分配 $rule_4$ 中的角色 r_1，这种情况称之为有关规则间冲突。

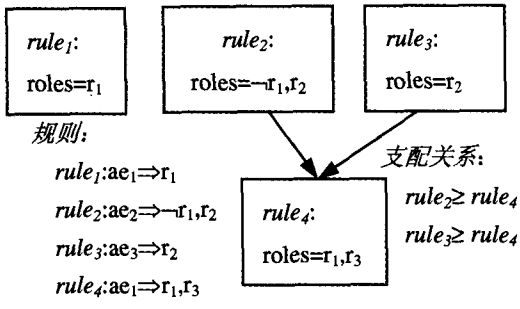

图 1　冲突的例子

3　描述逻辑语言

描述逻辑[9]是一阶谓词逻辑的一个可判定子集，具有强大的推理能力，能够提供完备、高效的知识推理机制，而且描述逻辑的表达能力和结构化语法，非常适合抽象 RB-RBAC 中的元素和关系。描述逻辑的基本元素是概念(Concepts)和角色(Roles)。概念表示一些个体的集合，角色表示个体之间的二元关系。任意的概念和角色都是由基本的原子概念(Atomic Concepts，记为 A)和原子角色(Atomic Roles，记为 R)通过构造符(Constructors)形成的。根据 RB-RBAC 模型的特点和本文的主要目的，本文选用 ALC[9]进行表示。

ALC 的概念(用 C,D 表示)具有如下的语法规则：

$C,D \rightarrow A \mid \top \mid \neg C \mid C \sqcap D \mid \exists R.C$

其中，A 表示原子概念，R 表示原子角色。描述逻辑中，$\top$ 定义了全称概念(Universal Concept)，$\bot$ 定义为底概念(Bottom Concept)，并且 $\top = \neg\bot$。可以使用如下的简写形式：$C \sqcup D$ 表示 $\neg(\neg C \sqcap \neg D)$，$\forall R.C$ 表示 $\neg\exists R.\neg C$，$CxorD$ 表示 $(C \sqcup D) \sqcap \neg(C \sqcap D)$。

为了定义 ALC 概念的形式化语义，考虑由一个非空的解释域 Δ^I 和解释函数 $\cdot^I$ 构成的解释 $I=(\Delta^I, \cdot^I)$。解释函数 $\cdot^I$ 把每个原子概念 A 映射到一个集合 $A^I \subseteq \Delta^I$，把每个原子角色 R 映射到一个二元关系 $R^I \subseteq \Delta^I \times \Delta^I$。$ALC$ 的语法和语义见表 1。

表 1　ALC 的语法和语义

构造符	语法	语义
全称概念	$\top$	Δ^I
原子概念	A	A^I
合取	$C \sqcap D$	$C^I \cap D^I$
非	$\neg C$	$\Delta^I \backslash C^I$
存在量词	$\exists R.C$	$\{a \in \Delta^I \mid \exists b.(a,b) \in R^I \wedge b \in C^I\}$

描述逻辑中的知识库 K 由 $TBox$ T 和 $ABox$ A 两部分组成，$K=\langle T,A\rangle$。

$TBox$ 定义了概念术语，描述了问题域中一般性知识，是领域结构中公理的集合。在 $TBox$ 中，用概念表示个体的集合，用角色表示个体间的二元关系。$TBox$ 中的公理具有如下形式：

$C \sqsubseteq D(R \sqsubseteq S)$ or $C \equiv D(R \equiv S)$

其中 C,D 是概念(R,S 是角色)。前一种形式叫做包含公理(inclusions)，声明对任何解释都有 $C^I \subseteq D^I$；后一种形式是等价公理(equalities)，声明对任何解释都有 $C^I = D^I$。一个解释 I 满足 $TBox$ T，当且仅当解释 I 满足 I 中的每个公理。如果解释 I 满足一个公理(或公理集合)，那么称它是该公理(或公理集合)的一个模型(model)。如果两个公理(或公理集合)有同样的模型，那么它们相等。

$ABox$ 描述问题域的具体知识，是关于具体个体和关系的断言的集合。$ABox$ 中包括概念断言(concept assertion)和角色断言(role assertion)，分别具有形式 $C(a)$，$R(a,b)$，C 是概念，R 是角色。一个解释 I 满足概念断言 $C(a)$，当且仅当 $a^I \in C^I$。一个解释 I 满足 $ABox$ A，当且仅当它满足 A 中的每个公理。

一个解释 I 满足知识库 K，当且仅当它满足 $TBox$ T 和 $ABox$ A。

描述逻辑的基本推理任务是可满足性(Satisfiablity)和包容(Subsumption)。如果存在 T 的一个模型使得概念 C 的解释非空，则称概念 C 关于 T 是可满足的。如果对 T 的任意模型 I，$C^I \subseteq D^I$ 成立，则称概念 C 包容概念 D，记为 $T \models C \sqsubseteq D$。概念包容很容易转化成可满足性判定：$C \sqsubseteq D$ 当且仅当 $C \sqcap \neg D$ 是不可满足的。如果存在一个解释是 T 和 A 的模型，那么知识库 K 就是一致的，也称为 A 关于 T 是一致的，简称为 A 是一致的。

4　RB-RBAC 模型的表示和推理

RB-RBAC 模型没有给出属性值间的优先级定义方法。为支持属性表达式间的优先级推理，本文的模型需要明确指明属性值间存在的优先关系。例如，在职位属性中，定义属性值“部门经理≥项目经理”，表明部门经理也可以作为项目经理。

为了支持 RBAC 的某些特性，允许在 RB-RBAC 模型中定义角色继承关系。

对于给定的 RB-RBAC 应用系统，定义描述逻辑知识库 K。在 K 中按照如下方法进行概念术语和断言的定义。

在 K 中定义如下原子概念：

原子概念 $CUser$，$CRole$ 和 $CPermission$，分别表示用户、角色、许可和用户属性；

对系统内的每个角色 r_i，定义一个原子概念 $Role_i$；

对每个属性表达式 ae_i，定义原子概念 AE_i；

对每种属性 A_i，定义一个原子概念 CA_i；对属性 A_i 的每个属性值 v_i^j，定义一个原子概念 $CAval_i^j$；

对每种属性 A_i，定义一个原子角色 $hasA_i$，表示用户具有的属性 A_i 的属性值；

原子角色 $assignRole$ 表示根据规则用户被自动分配某角色；

原子角色 $holdPermission$ 表示角色具有某种许可；

原子角色 $authorize$ 表示用户通过角色被授予某种许可。

在 K 的 $TBox$ 中，包含如下 5 种公理。

属性包含公理：表示了属性及属性值间的关系。每个属性值都是所在属性的子集，因此对每个属性值定义公理：$CAval_i^j \sqsubseteq CA_i$。对于属性值之间存在的偏序关系，例如部门经理也可以是项目经理，定义如下公理：$CAval_i^j \sqsubseteq CAval_i^k$，其中 $CAval_i^j$，$CAval_i^k$ 分别是对应属性值的原子概念。

概念 $\exists hasA_i.CAval_i^j$ 解释为属性 A_i 的属性值为 $CAval_i^j$

的用户集合。例如，职位(Position)是一种属性，属性值为{部门经理(DM)，项目经理(PM)}。那么，定义属性概念 $CPosition$、原子角色 $hasPosition$ 以及相应的属性值概念 DM，PM。概念 $\exists hasPosition.DM$ 解释为职位为部门经理的用户集合。

角色层次公理：表达了角色间的层次关系。每个 $Role_i$ 都是 $CRole$ 的子集，所以为每个角色定义公理：$Role_i \sqsubseteq CRole$。对任意角色 r_i 继承了角色 r_j 的许可，定义如下公理：$Role_i \sqsubseteq Role_j$，其中 $Role_i$，$Role_j$ 分别是对应于 r_i，r_j 的原子概念。

属性表达式公理：确定了具有特定属性值的用户集合。对每个属性表达式 ae_i，给出如下概念定义：

$AE_i \equiv \exists hasA_1.CAval_{i1}^1 \sqcap \cdots \sqcap \exists hasA_n.CAval_{in}^k$ 当属性表达式中没有要求某个属性 A_j 的取值时，在概念 AE_i 中可以省略关于属性 A_j 的表达，而不必记为 $\exists hasA_j.\top$。当允许对某属性取多个值时，可以记为 $\exists hasA_j.(CAval_j^{k1} \sqcup \cdots \sqcup CAval_j^{km})$。

角色分配公理：表示根据授权规则自动分配的角色。对任意授权规则定义具有如下形式的公理：$AE_i \sqsubseteq \exists assignRole.(Role_i^1 \sqcap \cdots \sqcap Role_i^m)$。概念 $\exists assignRole.Role_k$ 解释为分配了某个角色的用户的集合。该公理表明，如果用户属性值满足属性表达式 AE_i，那么用户将被自动分配此授权规则中定义的角色。例如，图 1 例子中的规则 $rule_2$ 可以表示为：

$$AE_2 \sqsubseteq \exists assignRole.\neg(Role_1 \sqcap Role_2)$$

当然，也可以定义形式如下的公理：

$$AE_i \sqsubseteq \exists assignRole.(Role_1 \ xor \ Role_2).$$

表示规则不能同时分配对应的角色 r_1 和 r_2。

授权公理：表示用户通过分配的角色获取许可。公理如下：

$$\exists assignRole.(\exists holdPermission.CPermissions) \sqsubseteq \exists authorize.CPermissions$$

概念 $\exists authorize.CPermissions$ 解释为被授予某个许可的用户的集合，概念 $\exists assignRole.(\exists holdPermission.CPermissions)$ 解释为根据规则被自动分配了某角色且该角色具有某特定许可的用户的集合。该包含公理表明一个用户可以获得某个许可，如果他能够被自动分配持有该许可的角色。

在 K 的 $ABox$ 中包含如下断言：

角色概念断言，形式为 $Role_i(r_i)$，用来定义每个系统内的角色；

用户概念断言，形式为 $CUser(u)$，用来定义系统内的用户；

属性值概念断言，形式为 $CAval_i^j(v_i^j)$，定义系统内的每个属性值；

许可概念断言，形式为 $CPermission(p)$，用来定义系统内的许可；

角色许可关系断言，形式为 $holdPermission(r,p)$，表示某个角色 r 具有许可 p；

用户属性关系断言，形式为 $hasA_i(u,av)$，表示用户 u 在属性 A_i 上的属性值为 av。

通过描述逻辑提供的推理机制，可以进行授权推理。根据角色分配公理和授权公理可知，如果一个用户满足某个授权规则的属性表达式，那么该用户就被分配了该授权规则中指定的角色。如果上述分配的角色持有许可 p，那么用户 u 最终将被授予许可 p。

对任意用户 u 和角色 r_i，如果 $K \models (\exists assignRole.Role_i)(u)$（$Role_i$ 是角色 r_i 对应的原子概念），说明用户 u 可以分配角色 r_i。对于用户 u 和许可 p，如果 $K \models authorize(u,p)$，说明用户 u 获得了许可 p。

通过 $TBox$ 推理机制也能够进行属性表达式间的优先级判定。任意属性表达式概念 AE_i，AE_j，如果 $T \models AE_i \sqsubseteq AE_j$，说明任意符合 ae_i 的用户也符合 ae_i，所以有 $ae_i \geqslant ae_j$。如果 $T \not\models AE_i \sqsubseteq AE_j$ 且 $T \not\models AE_j \sqsubseteq AE_i$，说明 ae_i 和 ae_j 间不存在任何支配关系。

5 策略冲突检测

在 $TBox$ 中定义每个属性表达式概念 AE_i 时，必须对 AE_i 进行可满足性检查，防止属性表达式本身的错误，保证在 $TBox$ 中添加概念定义 AE_i 后，$TBox$ 仍然是一致的。例如定义一个用户必须是部门经理但是不能是项目经理，由于部门经理和项目经理间存在偏序关系，因此这是一个无法满足的概念。

在 RB-RBAC 中，主要存在无关规则间冲突和有关规则间冲突。

5.1 有关规则间冲突检测

有关规则间冲突发生情况如下：对授权规则 $rule_i: ae_i \Rightarrow roleSet_i$，$rule_j: ae_j \Rightarrow roleSet_j$，如果 $ae_i \geqslant ae_j$ 且 $r_k \in roleSet_i$，$\neg r_k \in roleSet_j$（或 $\neg r_k \in roleSet_i$，$r_k \in roleSet_j$），那么规则 $rule_i$ 与 $rule_j$ 间存在冲突。

在 $TBox$ 中，如果存在支配关系 $ae_i \geqslant ae_j$，那么有 $T \models AE_i \sqsubseteq AE_j$。只要 AE_i 和 AE_j 分别具有如下形式的角色分配定理：

$AE_i \sqsubseteq \exists assignrole.(Role_k \sqcap \cdots)$，

$AE_j \sqsubseteq \exists assignrole.(\neg Role_k \sqcap \cdots)$。

综上，可推理得到 $AE_i \sqsubseteq \exists assignrole.(\neg \exists Role_k \sqcap \cdots)$，这显然是矛盾的，所以 AE_i 是不可满足的，进而 $TBox$ 是不一致的。因此，使用 $TBox$ 一致性检查能够有效地发现有关规则间冲突。以上推理也表明，授权策略中处于支配地位的用户受到了更多的约束条件，更具有特殊性。

可以通过 $TBox$ 一致性检查来检测有关规则间冲突。首先，保证在定义概念 AE_i 后 $TBox$ 仍然是一致的。接着，如果在添加 AE_i 的角色分配公理后 $TBox$ 是不一致的，说明授权规则间出现了冲突，那么获得不可满足的属性表达式原子概念集合 $CulpritAESet$，并删除 AE_i 的角色分配公理。对于 $\forall AE_j \in CulpritAESet$，如果 $AE_j \neq AE_i$，可以确定对应的授权规则 $rule_i$ 与 $rule_j$ 间存在冲突。如果 $AE_j = AE_i$，利用 $TBox$ 检索获得包含 AE_i 的所有属性表达式概念集合 $AncestorAESet$。对 $\forall AE_k \in AncestorAESet$，如果至少存在一个角色 r 在 AE_i 和 AE_k 分配的角色中分别以 r 和 $\neg r$ 否定形式出现，可以确定对应的授权规则 $rule_i$ 与 $rule_k$ 间存在冲突，其中授权规则分配的角色应考虑角色继承的影响。

对图 1 的例子，有如下角色分配公理

$AE_2 \doteq \exists assignRole.(\neg Role_1 \sqcap Role_2)$

$AE_4 \doteq \exists assignRole.(Role_1 \sqcap Role_3)$

删除以上两条角色分配公理，改写为

$AE_2 \doteq \exists assignRole.(Role_2 \sqcap Role_3)$

$AE_4' \equiv AE_4 \sqcap \neg AE_2$

$AE_4' \doteq \exists assignRole.(Role_1 \sqcap Role_3)$

图 2 改写方法一

对检测到的有关规则间冲突，在保留原有语意的原则下，可以采用破坏冲突规则间优先关系的方法，改写相应的公理，来消除系统中的冲突。例如，如图 2 所示，按照否定优先策略，对符合 AE_i 的用户及符合 AE_j 但不属于 AE_i 的用户重新进行角色分配。但是这种简单改写方法主要适用于 $rule_i$ 支配的所有规则间不存在其它的冲突。例如，如果存在 AE_k，$AE_i \sqsubseteq AE_k$，$AE_k \sqsubseteq AE_j$，$rule_i$，$rule_j$ 间冲突且 $rule_k$，$rule_j$ 间也有冲突。按照这种改写方式能够消除 $rule_i$，$rule_j$ 间的冲突，但可能将 $rule_k$，$rule_j$ 的有关规则间冲突转化为 $rule_k$，$rule_j'$ 的无关规则间冲突。可定义更复杂的改写算法，解决在支配关系链中同时存在两个以上的冲突问题。

5.2 无关规则间冲突检测

无关规则间冲突发生的情况如下：对授权规则 $rule_i: ae_i \Rightarrow roleSet_i$，$rule_j: ae_j \Rightarrow roleSet_j$，且 $r \in roleSet_i$，$\neg r \in roleSet_j$，如果用户 u 同时满足 $rule_i$ 和 $rule_j$，则引起冲突。无关规则间冲突，可以使用两种方法进行检测。

如果 $TBox$ 是一致的，$TBox$ 中 AE_i 和 AE_j 角色分配包含定理具有如下形式：$AE_i \sqsubseteq \exists assignrole.(Role_k \sqcap \cdots)$，$AE_j \sqsubseteq \exists assignrole.(\neg Role_k \sqcap \cdots)$。在 $ABox$ 加入用户概念断言 $CUser(u)$，只要用户 u 同时满足概念 AE_i 和 AE_j，则 u 也同时满足概念 $\exists assignrole.Role_k$ 和 $\exists assignrole.\neg Role_k$，但是 $\exists assignrole.Role_k$ 和 $\exists assignrole.\neg Role_k$ 不相交，所以断言 $CUser(u)$ 关于 $TBox$ 是不一致的。

根据 $ABox$ 一致性的检测方法，并不能保证发现所有的无关规则间冲突。因为，从 $ABox$ 的角度来看，无关规则间的冲突不是始终能反映在 $ABox$ 的不一致性中，而是因为加入了特定用户才引起 $ABox$ 的不一致性。所以，通过 $ABox$ 一致性检查来发现无关规则间冲突的前提是存在一个断言 $CUser(u)$ 导致 $ABox$ 是不一致的。但在用户规模相对小时，这种方法仍然是一种好的选择。当 $ABox$ 是一致的时候，即使 $TBox$ 中存在无关规则间冲突，也不影响推理结果的正确性；当 $ABox$ 不一致时，通过结合 $TBox$ 的概念可满足性查询，能够立即找出发生冲突的规则。

使用 $ABox$ 一致性检查检测无关规则间冲突前，应保证 $TBox$ 是一致的。首先，检查 $ABox$ 的一致性。如果 $ABox$ 是不一致的，说明系统出现了冲突，获取不可满足的用户集合 $CulpritUserSet$。其次，对 $\forall u \in CulpritUserSet$，使用 $ABox$ 检索，获得 u 作为实例的所有属性表达式原子概念集合 $AESet$。对 $\forall AE_i, AE_j \in AESet$，如果 $T \not\models AE_i \sqcap AE_j$，可以确定对应的授权规则 $rule_i$ 与 $rule_j$ 间存在冲突。

对检测到的有关规则间冲突，在保留原有语意的原则下，可以采用对用户重新划分的方法，消除系统中的冲突。例如，如图 3 所示，按照否定优先策略，对符合 AE_i 但不符合 AE_i 的用户、同时符合 AE_i 和 AE_i 的用户，以及符合 AE_j 但不符合 AE_i 的用户重新进行角色分配。

对图 1 的例子，有如下角色定义包含公理：

$AE_1 \sqsubseteq \exists assignRole.Role_1$

$AE_2 \sqsubseteq \exists assignRole.(\neg Role_1 \cap Role_2)$

删除以上两条角色公理，改写为

$AE_1' \equiv AE_1 \cap \neg AE_2 \quad AE_1' \sqsubseteq \exists assignRole.Role_1$

$AE_2' \equiv \neg AE_1 \cap AE_2$

$AE_2' \sqsubseteq \exists assignRole.(\neg Role_1 \sqsubseteq Role_2)$

$AE_{12} \equiv AE_1 \cap AE_2 \quad AE_{12} \sqsubseteq \exists assignRole.Role_2$

图 3 改写方法二

5.3 基于交集可满足性的冲突检测方法

有关规则间冲突在某种意义上是无关规则间冲突的一种特例，高优先级规则的用户恰好等于两个规则用户的交集，因此也可以对这两种冲突不加区分而进行冲突检测。定义 T^{all} 是所有的公理的集合，定义 T^{part} 为包含除角色分配定理之外的所有公理的集合。对任意的 AE_i 和 AE_j，如果有 $T^{part} \models AE_i \sqcap AE_j$，且 $T^{all} \not\models AE_i \sqcap AE_j$，那么对应的规则 $rule_i$，$rule_j$ 间必然存在冲突。概念 AE_i，AE_j 的交集在不包含角色分配公理时是可满足的，在加入了角色分配定理后无法满足，说明这两个概念的角色分配定理间存在着冲突。因此，可以在一次冲突检测中同时发现有关规则间冲突和无关规则间冲突。

基于交集可满足性的方法进行冲突检测需要将角色分配公理单独存储。首先，在 $TBox$ 中载入除了角色分配公理之外的所有公理，并保证 $TBox$ 是一致的。对 $TBox$ 中的任意 AE_i 和 AE_j，检查 $AE_i \sqcap AE_j$ 的可满足性。如果有 $T \models AE_i \sqcap AE_j$，将 (AE_i, AE_j) 加入列表 $OverlapedAEs$。其次，将所有的角色公理也添加到该 $TBox$ 中。对任意 $(AE_i, AE_j) \in OverlapedAEs$，检查 $AE_i \sqcap AE_j$ 的可满足性。如果有 $T \not\models AE_i \sqcap AE_j$，可以确定 $rule_i$ 与 $rule_j$ 间存在冲突。该方法能够在一次这样的检测过程中发现所有的有关规则间冲突和无关规则间冲突。在具有 n 个授权规则的系统中，最坏的情况下，需要做 $2\binom{2}{n}$ 次可满足性检查。即使没有冲突发生，也要做 $\binom{2}{n}$ 次可满足检查。

系统可以灵活选择采用的检测方法。例如，对已经定义了授权规则集合和用户的系统，如果对于每条规则和每个加入的用户都分别进行基于 $TBox$ 一致性和 $ABox$ 一致性的检测来发现冲突，将会成为沉重的负担，此时仅对授权规则应用检查交集可满足性的方法将能极大提高工作效率。在授权规则较多，但用户数量相对少的系统中，如果每次添加新的授权规则都使用基于检查交集可满足性的方法，将消耗较多的系统时间，而且显然没有必要重新计算其它规则间的冲突。此时利用基于 $TBox$ 一致性和 $ABox$ 一致性的检测方法将十分有助于提高系统的效率。

结论 本文提出了一种基于描述逻辑对 RB-RBAC 模型进行表示和推理的方法，该方法表达自然，便于进行推理，能够进行访问控制推理，在此基础上提出了几种冲突检测方法。根据 $TBox$ 一致性检测方法能够在逐条加入授权规则的过程中发现有关规则间冲突；$ABox$ 一致性检测方法能够根据不可满足的用户发现无关规则间冲突；基于交集可满足性的方法能在授权规则集合中检测出不同类型的冲突。在不同的情况下，综合使用这些方法，能够有效地解决实际的 RB-RBAC 系统中的冲突检测问题。

本文只给出了简单的策略改写方法以消除冲突，更复杂的策略改写方法，能够避免有关规则间冲突变成无关规则间冲突。

参 考 文 献

1 Sandhu R, Coyne E, Feinstein H, et al. Role-Based Access Control Model. IEEE Computer, 1996, 29(2): 38～47

2 Ferraiolo D, Sandhu R, Gavrila S, et al. Proposed NIST Standard for role-based access control: towards a unified standard. ACM Transaction on Information and System Security (TIS-

SEC),2001,4(3):224～274
3 Al-Kahtani M, Sandhu R. A Model for Attribute-Based User-Role Assignment. In: Proceedings of the 18th Annual Computer Security Applications Conference, Las Vegas, Nevada, December 2002
4 Al-Kahtani M, Sandhu R. Induced Role Hierarchies with Attribute-Based RBAC. In: Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT), Villa Gallia, Como, Italy, June 2003
5 Al-Kahtani M A, Sandhu R. Rule-Based RBAC with Negative Authorization. In: Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04), December 2004. 405～415
6 Benferhat S, Baida R E, Cuppens F. A Stratification-based Approach for Handling Conflicts in Access Control. In: SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies, Como, Italy, ACM Press, June 2003. 189～195
7 Moffett J D, Sloman M S. Policy Conflict Analysis in Distributed System Management. Journal of Organisational Computing, 1994,4(1):1～22
8 Lupu E, Sloman M. Conflict Analysis for Management Policies. Fifth IFIP/IEEE International Symposium on Integrated Network Management IM'97, San-Diego, May 1997
9 Baader F, Calvanese D, et al. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press ,2003

(上接第 96 页)

的入口点，通过发送未经授权的、容量庞大的邮件，以洪水攻击的方式，攻击接收者的电子邮件信箱或者网站的邮件服务器系统和目标网络的带宽资源，达到摧毁接收者的电子邮件信箱的邮件信息、使邮件服务器性能下降或大量消耗网络带宽资源的攻击意图，造成破坏信息的完整性和可用性的攻击后果。

Attack 5：网络监听攻击

网络监听攻击属于信息收集攻击类别。攻击者一般来自本地网络，攻击者以网络协议与操作系统的接口为攻击的入口点，通过监听程序，使用网络监听的攻击方法，获取网络上传输的数据并分析解码，达到收集敏感信息的攻击意图，造成系统信息的保密性被破坏的攻击后果。

攻击 6：后门攻击

后门攻击属于信息利用攻击类别。直接利用事先已植入的并已运行的后门程序进行攻击时，也不存在漏洞的利用问题，且所攻击的平台是由后门程序决定的。

分析：Attack 4 由于不需要利用现存的漏洞进行攻击，因而对漏洞的利用不加描述。Attack 5、Attack 6 所攻击的平台不固定，并且没有利用漏洞，因而在描述时，忽略了漏洞利用属性，而且未对攻击的平台做说明。当使用具体的攻击软件时，再描述攻击的平台。

结论 本文提出了一个系统化的攻击行为描述方案，并在此基础上讨论了对它进行裁剪的思想和方法，为分析和描述攻击提供了参考方案。通过对大量攻击实例的分析，说明了本文分析方法和裁剪规则的有效性和普遍适用性。当然，本文提出的攻击行为分析方法也存在着不足之处，特别是对于某些攻击来说，此方法适用于在较高层次抽象分析和描述，这时只具有最基本的描述能力。应用这种攻击分析描述方法能达到比较好的分析效果，可以系统地分析攻击的多方面特征，揭示攻击的本质特性，简化对攻击的理解，还可以用来分析攻击行为之间存在的关联性，构造完整的攻击过程。

参 考 文 献

1 Stallings W. Network and Internet Work Security Principles and Practice. NJ: Prentice Hall, 1995
2 Li M, Jia W, Zhao W. Decision Analysis of Network-Based Intrusion Detection Systems for Denial of Service Attacks. 0-7803-7010-4/01, IEEE, 2001
3 Kumar S. Classification and detection of computer intrusions: [Ph D Thesis]. Purdue University, 1995
4 刘欣然. 网络攻击分类技术综述. 通信学报，2004(7):30～36
5 Conen F. Information system attacks: a preliminary classification scheme [J]. Computers and Security, 1997, 16(1):29～46
6 Landwehr C E, Bull A R, McDermott J P, et al. A taxonomy of computer program security flaws. ACM Computing Surveys, 1994, 26(3):211～254
7 Bishop M. A Taxonomy of (Unix) System and Network Vulnerabilities: [Technical Report]. CSE-9510. Department of Computer Science, University of California at Davis, May 1995
8 Bishop M. Vulnerabilities analysis [A]. Second International Symposium on Recent Advances in Intrusion Detection [C]. USA, 1999
9 Lindqvist U, Jonsson E. How to Systematically Classify Computer Security Intrusions. In: IEEE Symposium on Security and Privacy. Oakland, California, USA, 1997. 154～163
10 Howard J D. An Analysis of Security Incidents on the Internet: [Ph D dissertation]. West Lafayette, USA: Carnegie Mellon University, 1997
11 A Taxonomy of Network and Computer Attack Methodologies: [Honours thesis]. University of Canterburry, November 7, 2003. http://www.cosc.canterbury.ac.nz/research/reports/HonsReps/2003/hons_0306.pdf
12 张涛，董占球. 网络攻击行为分类技术的研究. 计算机应用，2004，24(4):115～118
13 郭林，严芬，黄皓. 基于多维角度的攻击分类方法. 计算机工程，2005