基于属性和主体、操作和客体分层描述的逻辑授权语言

翟浩良 夏兰亭 李 磊
(中山大学软件研究所 广州 510275)

摘 要 安全策略是访问控制的核心,安全策略的描述、验证和执行离不开授权语言。在实际应用中,安全需求具有复杂性和动态性的特点,而现有的授权语言不能很好地适应这一特点,并不能对多种访问控制策略提供足够的支持。提出了一种基于属性和主体、操作和客体分层描述的逻辑授权语言(SOOSAL)。SOOSAL以一阶逻辑为基础,通过谓词对主体、客体和操作进行刻画,并以分层的方式通过规则对主体、操作和客体之间的关系进行描述。此外,SOOSAL从逻辑语义世界假设的角度对现实世界中的策略进行了分类:封闭性世界策略和开放性世界策略,并对这两种策略的安全性进行了讨论,给出安全性问题的简单解决方案。实例结果表明,SOOSAL具有较强的策略描述能力,能更好地实现策略的动态变化,并对不同的安全需求和授权原则提供良好的支持。

关键词 授权语言,逻辑,属性,分层
中图法分类号 TP301 **文献标识码** A

Logical Authorization Language Based on Attribute and Subject-Operation-Object Stratification

ZHAI Hao-liang XIA Lan-ting LI Lei
(Software Research Institute, Sun Yat-Sen University, Guangzhou 510275, China)

Abstract Security policy is the key of access control, and the description, authentication and execution of policy can not be realized without authorization language. In practice, the existing authorization languages are not well adapted to the complex and dynamic nature of security requirements and can not provide enough support for access control model. This paper proposed a logical authorization language based on attribute and subject-operation-object stratification (SOOSAL). Based on first-order logic, SOOSAL describes the subject, operation, and object by predication, and the relationship of these by rules. In addition, policy was classified into closed world policy and open world policy from the logic point of view of the world and a simple solution was given under security discussion of the two types of policies. Our experimental results show that SOOSAL has a strong descriptive power, and can achieve the policy of dynamic change and support different security requirements and authority principle better.

Keywords Authorization language, Logic, Attribute, Stratification

1 引言

随着互联网和信息系统的发展,用户可访问资源的结构日趋复杂,规模日益增大,且对访问灵活性的要求越来越高,因此如何保证信息系统中数据资源的安全性已经成为当前一个突出的问题。

访问控制是通过特定的规则和途径来保护系统资源安全的一种方法,它通过限制主体对客体的访问来保证客体资源安全、合法的使用。访问控制的关键是安全策略,它是用于确定主体是否拥有对客体访问能力的一套特定规则的集合[1]。对安全策略的研究主要包括安全策略的描述、执行和验证等几个方面。

安全策略最核心的内容是授权语言,对安全策略的描述、执行和验证都基于授权语言。在实际应用中,由于安全需求和策略经常会发生变化,对于安全策略的描述也变得越来越复杂,现有的授权语言很难适应策略的动态变化,而且维护代价高。

例1 假设Alice和Bob都是业务部的职员,Jack是业务部的经理,Henry是业务部的副经理。Alice和Bob都能查询和修改自己制作的业务报表,业务经理处理可以处理自己的业务外,还能查看和修改部门职员制作的业务报表,但副经理不可以查看和修改员工制作的业务报表,只负责处理业务部其他事务,同时这些事务也可以由经理处理。经理、副经理和职员之间的权限关系见图1。

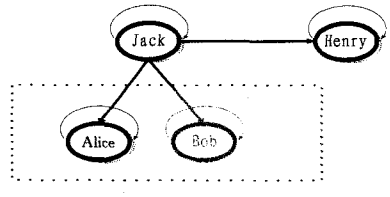


图1 权限关系图

图中的结点表示用户,结点X到结点Y的边表示用户X

本文受澳门科学技术发展基金(013/2010/A),中国博士后科学基金(20100480806)资助。
翟浩良(1983—),男,博士生,主要研究领域为人工智能、信息安全等,E-mail:zhaihliang@163.com;夏兰亭(1982—),男,博士生,主要研究领域为知识表示与推理;李 磊(1951—),男,教授,博士生导师,主要研究领域为计算机软件、数据库与知识库等。

拥有 Y 的所有权限。

此时如要描述该部门的授权情况，需要对每个职员和经理及其可以操作的业务报表以及其他资源进行逐一描述。比如可以使用访问控制矩阵或三元组以及其他结构和语言。但这些语言基本都是对用户、操作和操作的资源进行捆绑式描述的，如使用(Alice，查询，报表 1)来描述 Alice 能查询报表 1。假设发生以下情况：

(1)权限委托：假设经理 Jack 出差，此时业务部的事务可能需要副经理 Henry 处理，授权情况该怎么描述？如果 Jack 出差回来后，权限又有什么新变化？

(2)权限增加：假如业务部有新员工进入，在不影响其他人的情况下，权限需要进行什么样的变更？

(3)权限修改：假设经理也不能处理业务员制作的业务报表，该怎么描述，权限总体的变化情况是怎么样？

上述这些授权的变化可以通过修改策略和描述规则来实现。但这些规则通常是互相联系和影响的，需要进行全面统一的判断和修改，有时甚至需要重建策略和规则，代价较大，而且不易保证修改后的策略的安全性和一致性。

在实际应用中，会经常遇到与例 1 类似的情形，这给策略语言的研究提出了挑战。而现有的授权语言却不能很好地适应安全需求的各种变化，变更和维护代价较大。因此，本文提出一种基于属性和主体、操作和客体分层描述的逻辑授权语言。通过属性描述主体、操作和客体能加强语言的表达能力；而主体、操作和客体分层描述能为策略的动态变化提供足够的支持，并且策略容易修改和维护。逻辑可以为访问控制策略的描述和推理提供一个简单、固定而又一般性的理论基础，具有良好的可靠性和完全性。因此把它们结合起来是有意义的。

本文第 2 节介绍了相关的背景知识和研究工作；第 3 节是基于属性和主体、操作和客体分层描述的逻辑授权语言，给出了具体的语法和结构描述，并对应用该语言描述的策略的安全性进行了讨论；第 4 节将其与其他访问控制语言的各种特性进行了简单对比；第 5 节是一个语言具体实现的例子；最后总结全文。

2 背景知识和相关工作

本节介绍策略基本元素和相关的研究工作。

2.1 基本元素

文献[2]中把策略的结构分为三大基本元素：策略主体、策略目标和策略行为。

策略主体即操作的执行者，可以是用户或任何虚拟的主体，比如一段程序、某个系统或一个 agent。用户或虚拟主体都包含有其本身的属性，比如用户的姓名、身份、角色、部门或虚拟主体执行的角色、身份、所在的部门等。在实际应用中，主体的属性之间可能存在从属或层次关系，比如角色。而且在某些特殊情况下，比如用户出差、协同工作和权力下放时，主体需要通过权限委托即把自身权限授予其他主体来实现功能的需要[27]。

操作(operation)即策略行为，表示“做什么(动作)”。操作通常是原子的，即操作和操作之间都是独立的，不存在包含和从属关系，比如文件系统中的读操作、写操作和执行操作，数据库系统中的查询、增加、修改和删除等操作。操作也有自己的属性，这些属性可以用来表示该操作对于主体的访问约束或访问条件。

客体(object)即策略目标，是操作或执行的对象，它可以是任何的数据实体，比如目录、文件、系统、程序等。客体也有自己的属性，比如数据类型、位置、数据内容等，也有与主体和操作相关的属性，比如可执行的操作和客体的创建者等。

为了统一，在本文后面的章节中，我们用主体、操作和客体来表示策略主体、策略行为和策略目标。

2.2 相关研究

传统的授权语言主要有缺省逻辑[2]、ASL[3-5]、PDL[7]、Ponder[7-10]、Selinux 参考策略语言[12]和 Binder[14]。缺省逻辑有很强的表达能力和推理能力，但缺省逻辑语言是不可判定的或半可判定的[6]，而且可能是非确定性的，即对于一个具体的访问请求可能无法给出明确的授权结果；另外缺省逻辑不支持主体和客体含有结构性的属性，没有给出具体的冲突解决机制[5]。ASL 是基于谓词逻辑的授权语言，有较强的推理能力，并给出了一致性问题的具体解决方案，支持多种访问策略，运算能在多项式时间结束，而且运算结果是明确的。但 ASL 对访问控制模型的支持不够，不支持管理策略，而且受到谓词逻辑的限制，导致语言可读性差，不便于理解和编写。PDL 是基于事件的授权语言，它的基本格式是 Event-conditions-action，事件只能在某一条件下发生并触发动作的执行。PDL 主要应用于基于策略的网络管理，不支持信息系统的访问控制。Selinux 是针对安全操作系统领域的参考策略语言，并支持多种安全模型。但是 Selinux 缺乏结构性，对每个策略规则都需要明确声明和编写，这导致其策略规则十分复杂、庞大，不便于用户的理解和使用。Ponder 是一种声明性、结构性的策略语言，它添加了面向对象的设计思想，支持策略类型的定义和实例化。但是 Ponder 配置复杂，描述的主要是管理策略，而且针对服务质量领域，仅提供了对自主访问控制的支持，无法完整地描述其他访问控制模型如 MAC 和 RBAC 等[15]。John 的 Binder 语言是直接基于逻辑编程的，一个 Binder 程序是一个逻辑规则的集合，它的逻辑规则描述形式与 Prolog 类似，在规则描述中引入了一个独特的运算符“says”，Binder 支持授权委托。但 Binder 在规则中不支持函数符号[14]，而且不支持内嵌的命名空间。

2003 年 OASIS 制定了一种基于 XML 标准的通用授权语言和访问决策语言 XACML[16]。XACML 允许用户自定义数据类型、函数和规则，可以用来描述复杂的访问控制需求；访问策略的描述是基于属性的，因此可以提供细粒度的访问控制机制，而且支持策略集之间的代数合成；在应用中由于是基于 XML 标准的，因此它能够适应不同系统中的协同工作。但 XACML 在具体应用中，标签定义复杂可读性差，而且不能用来描述委托策略和进行可信计算，描述角色继承存在冗余信息，影响策略管理的效率，不能有效地支持职责分离原则[1]。XACML 缺乏策略分析、规则匹配、判定响应等相关的优化处理方法，随着策略规模和复杂性的增加，在很大程度上导致了 XACML 策略评估引擎在处理策略信息检索、多策略匹配等问题时的实际性能指标偏低，具体表现为系统资源开销大、访问请求应答延时长、远程通信交互多，因而无法满足商业应用的高业务吞吐量[17]。

LUC[18]是一个基于逻辑语义的使用控制授权语言框架，它以 datalog 为基础，具有强大的表达能力，并且具有逻辑上

的一致性、完备性和可行性。LUC是针对使用控制模型的，而且在实际应用中还需要进一步对其语义进行扩展。

分散授权语言SecPAL[19]是基于逻辑编程语言DATALOG的，它扩展了DATALOG的约束表达，其表达能力强，能支持多种授权策略和模型，如自主访问控制模型和强制访问控制模型，支持角色层次、职责分离和授权委托，语义简单明了。但SecPAL会产生信息泄露，不支持函数，并且没有具体实现的工具。DKAL[20]是一种分布式的授权语言，它是在SecPAL的基础上研究开发的，它基于EFPL(existential fixed-point logic)，解决了SecPAL的信息泄露问题，并且引入了"≤"来描述信息之间的偏序关系。而且DKAL能支持否定和函数。

逻辑因其强大的推理能力和良好的可靠性和完全性，在访问控制其他领域中也获得了大量的研究成果[24-29]。

文献[21]对逻辑在访问控制中的研究和应用作了基础的描述和总结，具体表现在访问控制矩阵到逻辑公式的变换、分布式系统中权限的传递与规则的建立以及基于逻辑的授权语言等问题。文献[22]中作者尝试揭露和总结逻辑和访问控制的内在关系，并在逻辑基础上分析了访问控制模型的一些基本逻辑问题如逻辑基础、可判定问题和安全性分析。文献[23-25]分别用逻辑对访问控制策略中的形式化、推理和授权进行了描述。文献[26]对基于角色的访问控制(RBAC)作了形式化的描述，并以描述逻辑(DL)为基础提出了RBAC的描述逻辑DL_{RBAC}，用描述逻辑的符号对RBAC中的主要元素和关系进行了形式化定义，并证明了描述逻辑DL_{RBAC}对于RBAC模型的忠实性。

综上所述，上述工作有些对模型的支持不够，有些不能很好地适应策略的动态变化，而且对于策略变化的维护代价一般较高。

3 基于属性和主体、操作和客体分层描述的逻辑授权语言(SOOSAL)

逻辑有着严格的理论基础和推理系统、强大的表达力和灵活性，因此适合用于描述基于规则的策略。在实际应用中，随着安全需求经常性的变化和策略规模及结构复杂性的增加，对主体、操作和客体的关系进行分层描述能为策略的动态变化提供足够的支持，因此本文提出了基于属性和主体、操作和客体分层描述的逻辑授权语言。

3.1 SOOSAL语法

定义1(符号) 基于逻辑编程的通用访问控制授权语言主要包含以下3类符号：

(1)常量符号，用小写字母开头的字符有限序列表示，例如a b manager；

(2)变量符号，是大写字母开头的字符有限序列表示。例如X Y Who；

(3)比较运算符，包括=，≠，>，<，≥，≤；

(4)谓词符号，SOOSAL语言包含的主要谓词符号见表1，第一列表示谓词符号的名称，第二列表示该谓词符号的元数，第三列说明谓词，第四列说明该谓词符号的意义；

(5)特殊符号，包括⊥，⊤，其中，⊥表示"假"，⊤表示"真"。

表1 SOOSAL谓词符号和原子公式

谓词符号	元	谓词(原子公式)	意义
sub	n+1	sub(X,X_1,X_2,…,X_n)	sub(X,X_1,X_2,…,X_n)表示主体及主体属性，其中X,X_i($1\leqslant i\leqslant n$)是项，X表示主体的标识，X_i($1\leqslant i\leqslant n$)表示主体的属性
oper	n+1	oper(Y,Y_1,Y_2,…,Y_n)	oper(Y,Y_1,Y_2,…,Y_n)表示操作及操作属性，其中Y,Y_i($1\leqslant i\leqslant n$)是项，Y表示操作的标识，Y_i($1\leqslant i\leqslant n$)表示操作的属性
obj	n+1	obj(Z,Z_1,Z_2,…,Z_n)	obj(Z,Z_1,Z_2,…,Z_n)表示客体及客体属性，其中Z,Z_i($1\leqslant i\leqslant n$)是项，Z表示客体的标识；Z_i($1\leqslant i\leqslant n$)表示客体的属性
sub_oper_permit	2	sub_oper_permit(X,Y)	sub_oper_permit(X,Y)表示主体和操作的操作许可关系，其中X,Y是项，X表示主体标识，Y表示操作标识
sub_oper_deny	2	sub_oper_deny(X,Y)	sub_oper_deny(X,Y)表示主体和操作的操作拒绝关系，其中X,Y是项，X表示主体标识，Y表示操作标识
oper_obj_permit	2	oper_obj_permit(Y,Z)	oper_obj_permit(Y,Z)表示操作和客体之间的操作许可关系，其中Y,Z是项，Y表示操作标识，Z表示客体标识
oper_obj_deny	2	oper_obj_deny(Y,Z)	oper_obj_deny(Y,Z)表示操作和客体之间的操作拒绝关系，其中Y,Z是项，Y表示操作标识，Z表示客体标识
sub_obj_permit	2	sub_obj_permit(X,Z)	sub_obj_permit(X,Z)表示主体和客体之间的操作许可关系，其中X,Z是项，X表示主体标识，Z表示客体标识
sub_obj_deny	2	sub_obj_deny(X,Z)	sub_obj_deny(X,Z)表示主体和客体之间的操作拒绝关系，其中X,Z是项，X表示主体标识，Z表示客体标识
permit	3	permit(X,Y,Z)	permit(X,Y,Z)表示主体、操作和客体之间的访问许可关系，其中X,Y,Z是项，X表示主体标识，Y表示操作标识，Z表示客体标识
deny	3	deny(X,Y,Z)	deny(X,Y,Z)表示主体、操作和客体之间的访问拒绝关系，其中X,Y,Z是项，X表示主体标识，Y表示操作标识，Z表示客体标识
in	2	in($Attr_1$,$Attr_2$)	in($Attr_1$,$Attr_2$)表示主体属性或客体属性之间的从属(包含)关系，其中$Attr_i$(i=1,2)是项，表示主体之间或客体之间存在包含关系的属性
conflict	2	conflict($Attr_1$,$Attr_2$)	conflict($Attr_1$,$Attr_2$)表示主体属性或客体属性之间的互斥关系，其中$Attr_i$(i=1,2)是项，表示主体之间或客体之间互相矛盾的属性
delegate	2	delegate(X_1,X_2)	delegate(X_1,X_2)表示主体之间的权限委托关系，其中X_1,X_2是项，表示主体的标识
error	1	error(system)	error(system)表示系统策略存在错误，其中system为系统标识
nothing	1	nothing(system)	nothing(system)表示系统不进行任何行为或操作，其中system为系统标识

说明：n为元谓词，在具体的应用时可以根据应用需要来确定n具体的数值，比如说$n=2$或$n=4$。而且对于不同的应用环境和应用需求，可以根据实际情况，对表1的谓词符号进行扩展。

定义 2(项)　常量和变量都是项。

定义 3(原子公式)　$P(t_1,t_2,\cdots,t_n)$是原子公式,当且仅当:P是n元谓词符号且$t_1,t_2,\cdots,t_n$是项。SOOSAL中主要原子公式及其含义见表2。

表2 各种策略描述语言对比

	策略支持	动态变化	推理能力	面向对象	一致性检测	完整性检测	角色层次	职责分离	结构性	授权委托	控制粒度	可读性	扩展性	分布式	函数支持
缺省逻辑			√									(			
ASL	√		√		√	√	√	√							
Ponder				√					√				√		
Binder	√		√							√		√		√	
XACML	√						√		√		√		√	√	
SecPAL	√		√				√	√		√				√	
DKAL	√		√				√	√		√				√	√
SOOSAL	√	√	√		√	√	√	√		√	√	√	√		

定义 4(文字)　原子公式或原子公式的否定是文字。

定义 5(规则和事实)　SOOSAL中的规则包含一个规则头和一个规则体,可以表示为:$a:-L_1,L_2,\cdots,L_n$,其中a是原子公式,$L_1,L_2,\cdots,L_n$是文字,:-左边的部分为规则头,:-右边的部分为规则体。

(1)如果$n=0$,即规则中没有规则体,则称此规则为事实,可以简记为a。

(2)如果规则头为空,则该规则表示一个约束,此时规则可以记为$\perp:-L_1,L_2,\cdots,L_n$。

3.2 SOOSAL事实和基本规则

SOOSAL事实和基本规则定义了利用SOOSAL来描述访问策略的核心内容。主要包括主体、操作和客体自身的事实描述和属性关系事实描述以及各种访问规则。由于策略动态性的需要,SOOSAL采用主体、操作和客体关系分层的访问规则描述方法,SOOSAL的基本规则包括主体和操作访问规则、操作和客体访问规则、主体和客体访问规则和主体、操作和客体访问规则。

定义 6(事实)　SOOSAL主要包含6种类型的事实,分别用来表示主体、操作和客体以及属性从属、矛盾关系和主体授权关系。具体的形式见表1。

使用事实可以对策略的3大元素以及一些属性关系进行描述,为了提供细粒度的访问控制,采用了基于属性的描述方法。下面通过一个例子来说明。

例2　SOOSAL中6种类型事实的具体实例:

(1)sub(s01,zhangsan,sysu,student)

(2)oper(receive,Mail,sysu)

(3)obj(obj1,notification,Mail,zhangsan,sysu,student)

(4)in(general_manager,department_manager)

(5)conflict(proposer,approver)

(6)delegate(zhangsan,lisi)

其中,(1)是一个主体事实,主体的标识为s01,属性有3个,分别表示姓名、学校和角色。(2)是一个操作事实,标识(名称)是receive,该操作还有两个属性,表示操作是学校sysu的邮件接收器。(3)是一个客体事实,标识是obj1,客体名称是notification,类型是mail,后3个属性表示收件人的姓名、学校和角色。(4)是一个角色从属事实,即总经理的角色是部门经理角色的上层角色。(5)是一个属性矛盾事实,即角色proposer和approver是互斥的角色。(6)是一个主体的授权委托事实,即张三把自己的权限委托给李四。

通过事实可以对策略的3大元素及相关事实关系进行描述,事实之间都是独立的,元素属性和状态的变化只需要对相应的事实进行修改即可。

访问规则是建立在主体、操作和客体事实上的访问关系。对于同一个系统来说,规则本身相对来说是稳定的,变化的通常是主体、操作和客体本身的属性。

定义 7(主体和操作访问规则)　主体和操作访问规则定义了主体和操作之间的可操作关系,主要有两种类型:主体和操作访问许可规则和主体和操作访问拒绝规则,也即满足什么条件的主体可操作和不能操作该操作。具体的形式如下:

(1)$sub\_oper\_permit(X,Y):-sub(X,X_1,X_2,\cdots,X_n),oper(Y,Y_1,Y_2,\cdots,Y_n),X_{i1}\nabla Y_{j1},\cdots,X_{ik}\nabla Y_{jk}$

(2)$sub\_oper\_deny(X,Y):-sub(X,X_1,X_2,\cdots,X_n),oper(Y,Y_1,Y_2,\cdots,Y_n),X_{i1}\nabla Y_{j1},\cdots,X_{ik}\nabla Y_{jk}$

其中$\{X_{i1},\cdots,X_{ik}\}\subseteq\{X_1,X_2,\cdots,X_n\},\{Y_{j1},\cdots,Y_{jk}\}\subseteq\{Y_1,Y_2,\cdots,Y_n\}(k\geqslant 1)$,(1)是主体和操作访问许可规则,(2)是主体和操作访问拒绝规则,∇是比较关系谓词符号。

定义 8(操作和客体访问规则)　操作和客体访问规则定义了客体和操作之间的可操作关系,也即一个客体可以被什么操作所操作。与主体和操作访问规则一样,操作和客体访问规则也有两种类型,具体形式如下:

(1)$oper\_obj\_permit(Y,Z):-oper(Y,Y_1,Y_2,\cdots,Y_n),obj(Z,Z_1,Z_2,\cdots,Z_n),Y_{i1}\nabla Z_{j1},\cdots,Y_{ik}\nabla Z_{jk}$

(2)$oper\_obj\_deny(Y,Z):-oper(Y,Y_1,Y_2,\cdots,Y_n),obj(Z,Z_1,Z_2,\cdots,Z_n),Y_{i1}\nabla Z_{j1},\cdots,Y_{ik}\nabla Z_{jk}$

其中$\{Y_{i1},\cdots,Y_{ik}\}\subseteq\{Y_1,Y_2,\cdots,Y_n\},\{Z_{j1},\cdots,Z_{jk}\}\subseteq\{Z_1,Z_2,\cdots,Z_n\}(k\geqslant 1)$,(1)是操作和客体访问许可规则,(2)是操作和客体访问拒绝规则,∇是比较关系谓词符号。

定义 9(主体和客体访问规则)　主体和客体访问规则定义了主体和客体之间的可访问关系,也即客体可以被满足什么条件的主体所访问,什么条件的主体不能访问。具体形式

如下：

(1)$sub\_obj\_permit(X,Z)$:- $sub(X,X_1,X_2,\cdots,X_n)$,$obj(Z,Z_1,Z_2,\cdots,Z_n)$,$X_{i1}\nabla Y_{j1},\cdots,X_{ik}\nabla Y_{jk}$

(2)$sub\_obj\_deny(X,Z)$:- $sub(X,X_1,X_2,\cdots,X_n)$,$obj(Z,Z_1,Z_2,\cdots,Z_n)$,$X_{i1}\nabla Y_{j1},\cdots,X_{ik}\nabla Y_{jk}$

其中$\{X_{i1},\cdots,X_{ik}\}\subseteq\{X_1,X_2,\cdots,X_n\}$,$\{Z_{j1},\cdots,Z_{jk}\}\subseteq\{Z_1,Z_2,\cdots,Z_n\}$($k\geqslant1$),(1)是主体和客体访问许可规则,(2)是主体和客体访问拒绝规则,∇是比较关系谓词符号。

定义 10(主体、操作和客体访问许可规则) 主体、操作和客体访问规则定义了主体、操作和客体之间的访问关系,也有两种类型的规则:许可规则和拒绝规则,即主体可以对客体进行操作和不能对该客体进行操作。许可规则的形式如下:

$permit(X,Y,Z)$:- $sub\_oper\_permit(X,Y)$,$oper\_obj\_prmit(Y,Z)$,$sub\_obj\_permit(X,Z)$

拒绝规则有两种描述方式:一种是强制拒绝,强制拒绝表示只要主体和操作、操作和客体或主体和客体单独拒绝访问,则主体就不可对客体进行操作操作;另一种是柔性拒绝,它表示只有主体和操作、操作和客体和主体和客体同时拒绝访问时,主体、操作和客体才拒绝访问。它们的形式如下:

(1)$deny(X,Y,Z)$:- $sub\_oper\_deny(X,Y)$

$deny(X,Y,Z)$:- $oper\_obj\_deny(Y,Z)$

$deny(X,Y,Z)$:- $sub\_obj\_deny(X,Z)$

(2)$deny(X,Y,Z)$:- $sub\_oper\_deny(X,Y)$,$oper\_obj\_deny(Y,Z)$,$sub\_obj\_deny(X,Z)$

(1)是强制拒绝规则,如主体 $u1$ 不能进行 *read* 操作,则 $u1$ 对任何客体 *obj* 都不可进行 *read* 操作;(2)是柔性拒绝规则,只有 $sub\_oper\_deny(u1,read)$,$oper\_obj\_deny(read,obj)$,$sub_obj_deny(u1,obj)$同时成立时,$u1$ 对客体 *obj* 才不可进行 *read* 操作。

3.3 SOOSAL 扩展规则

SOOSAL 基本规则描述了主体、操作和客体之间的简单访问关系,主体和主体以及客体和客体之间都是独立的,没有联系。扩展规则描述了主体、操作和客体之间的扩展访问关系,即主体之间以及客体之间的属性存在从属关系或主体之间存在权限委托关系,比如主体的角色之间或部门之间、客体所在的位置之间。

定义 11(主体、操作和客体访问许可扩展规则) 主体、操作和客体访问许可扩展规则定义了主体、操作和客体之间的扩展访问关系,即主体属性之间和客体属性存在从属关系,比如角色层次、文件位置等。主体、操作和客体访问扩展规则有 3 种扩展形式:主体属性扩展、客体属性扩展以及主体和客体属性同时扩展,这 3 种形式的扩展规则可以用如下两条规则来描述:

(1)$permit(X,Y,Z)$:- $permit(XX,Y,Z)$,$sub(X,X_1,X_2,\cdots,X_n)$,$sub(XX,XX_1,XX_2,\cdots,XX_n)$,$in(X_1,XX_1),\cdots,in(X_n,XX_n)$

(2)$permit(X,Y,Z)$:- $permit(X,Y,ZZ)$,$obj(Z,Z_1,Z_2,\cdots,Z_n)$,$obj(ZZ,ZZ_1,ZZ_2,\cdots,ZZ_n)$,$in(Z_1,ZZ_1')$,$\cdots$,$in(Z_n,ZZ_n)$

主体属性扩展可以单独用规则(1)来描述,客体属性扩展可以单独用规则(2)来描述,主体和客体属性同时扩展可以用规则(1)和规则(2)共同来描述。

定义 12(权限委托规则) 权限委托规则定义了主体和主体之间的权限传递关系,即如果一个主体由于某种原因不能执行权限对应的事务,则可以把自己的权限委托给另一个主体来进行操作。权限委托规则有两种形式,一种是全部权限委托,即委托人把自己的权限全部委托给另一主体。另一种是部分权限委托,即主体只能把一部分权限委托给另一主体,具体的规则形式如下:

(1)$permit(X,Y,Z)$:- $permit(XX,Y,Z)$,$delegate(XX,X)$

(2)$permit(X,Y,Z)$:- $permit(XX,Y,Z)$,$delegate(XX,X)$,$oper(Y,Y_1,Y_2,\cdots,Y_n)$,$obj(Z,Z_1,Z_2,\cdots,Z_n)$,$Y_1\nabla y_1,\cdots,Y_n\nabla y_n,Z_1\nabla z_1,\cdots,Z_n\nabla z_n$

主体全部权限委托可以用规则(1)来描述,部分权限委托可以用规则(2)来描述。

3.4 SOOSAL 语义

SOOSAL 以一阶逻辑为基础,其中的符号、项、原子公式、事实与规则描述基本与 Prolog 逻辑程序设计语言的语法相同。因此,SOOSAL 中的符号、项、原子公式、事实与规则的语义解释 Prolog 语言的语义解释相同。具体见文献[29]。

3.5 SOOSAL 策略

文献[28]把策略分成了两种类型:Closed Policy 和 Open policy,分别用来描述许可规则和拒绝规则。从逻辑语义的世界假设看,这两种描述方式都是基于封闭性世界假设的。SOOSAL 是基于逻辑的授权语言,从逻辑语义的世界假设出发,本文把 SOOSAL 策略也分为两种,分别是封闭性世界策略和开放性世界策略。

定义 13(封闭性世界策略) 封闭性世界策略是一个只有许可规则或拒绝规则的集合,它有两种描述形式:许可规则即许可的访问全部被描述,没有被描述的访问全被拒绝;另一种描述形式是拒绝规则,即被拒绝的访问全被描述,没有被描述的访问都许可。但封闭性世界策略不能同时包含这两种规则。

封闭性世界策略两种类型的描述形式如下:

封闭性世界许可策略={事实,许可规则,权限委托规则,拒绝默认规则},许可规则可以是主体和操作许可、操作和客体许可、主体和客体许可、主体和操作扩展许可、操作和客体扩展许可、主体和客体扩展许可和主体、操作和客体许可规则。拒绝默认规则的形式如下:

$deny(X,Y,Z)$:-$\rightarrow permit(X,Y,Z)$

封闭性世界拒绝策略={拒绝规则,许可默认规则},拒绝规则可以是主体和操作拒绝、操作和客体拒绝、主体和客体拒绝、主体和操作扩展拒绝、操作和客体扩展拒绝、主体和客体扩展拒绝和主体、操作和客体拒绝规则。许可默认规则的形式如下:

$permit(X,Y,Z)$:-$\rightarrow deny(X,Y,Z)$

定义 14(开放性世界策略) 开放性世界策略是一个许可规则和拒绝规则的集合。

开放性世界策略的形式如下:

开放性世界策略={事实,访问规则},访问规则可以是许可规则,也可以是拒绝规则,主要包括主体和操作访问、操作和客体访问、主体和客体访问、主体和操作扩展访问、操作和客体扩展访问、主体和客体扩展访问和主体、操作和客体访问

规则。

3.6 策略安全性讨论及其解决方案

在实际应用中，访问策略在应用时必须要保证是安全的，安全性主要包括3个方面的内容：主体属性约束、一致性约束和完备性约束。主体属性约束指同一个主体拥有互斥的属性，比如角色；一致性约束指在策略中同一个访问请求不能得到互斥的回答；完备性约束指对于任何一个访问请求，策略都能给出确定的回答。

3.6.1 主体属性约束

主体属性约束定义了在策略编写和策略执行过程中主体属性之间所需要满足的约束条件，如一个主体不能同时拥有互斥或冲突的属性(如角色，组织等)。典型的主体属性约束有职责分离约束、中国墙策略等。我们可以通过如下形式的规则来对策略进行主体属性约束检测：

$error(system)$:- $sub(X_1, X_{11}, X_{12}, \cdots, X_{1n}), sub(X_2, X_{21}, X_{22}, \cdots, X_{2n}), conflict(X_{1i}, X_{2i})$

其中，$1 \leqslant i \leqslant n$。

主体属性约束的解决方案可以通过删除主体属性事实来解决，即在同一策略中不能存在两条互斥的主体属性事实。

3.6.2 一致性约束

由于封闭性世界策略不存在一致性问题，因此只对开放性世界策略进行一致性讨论。

定义15(一致性) 如果开放性世界策略中对于任一请求(X,Y,Z)，$permit(X,Y,Z)$和$deny(X,Y,Z)$不同时成立，则称策略是一致的。

一致性约束可以通过以下规则来进行检测：

$error(system)$:- $permit(X,Y,Z), deny(X,Y,Z)$

策略一致性问题有如下4种解决方案：拒绝优先、许可优先、返回错误和请求无效，可以分别通过以下规则来实现：

$deny(X,Y,Z)$:- $permit(X,Y,Z), deny(X,Y,Z)$

$permit(X,Y,Z)$:- $permit(X,Y,Z), deny(X,Y,Z)$

$error(system)$:- $permit(X,Y,Z), deny(X,Y,Z)$

$nothing(system)$:- $permit(X,Y,Z), deny(X,Y,Z)$

3.6.3 完备性约束

封闭性世界策略一定是满足完备性约束的，因此完备性约束也只在开放性世界策略中讨论。

定义16(完备性) 如果开放性世界策略对于任一(X,Y,Z)，$permit(X,Y,Z)$和$deny(X,Y,Z)$至少有一个成立，则称策略是完备的。完备性约束可以通过如下规则来检测，规则的形式如下：

$error(system)$:-$\neg permit(X,Y,Z), \neg deny(X,Y,Z)$

策略完备性约束也有四种解决方案，分别是：默认许可、默认拒绝、返回错误、请求无效。分别可以通过以下规则来实现：

$permit(X,Y,Z)$:-$\neg permit(X,Y,Z), \neg deny(X,Y,Z)$

$deny(X,Y,Z)$:-$\neg permit(X,Y,Z), \neg deny(X,Y,Z)$

$error(system)$:-$\neg permit(X,Y,Z), \neg deny(X,Y,Z)$

$nothing(system)$:-$\neg permit(X,Y,Z), \neg deny(X,Y,Z)$

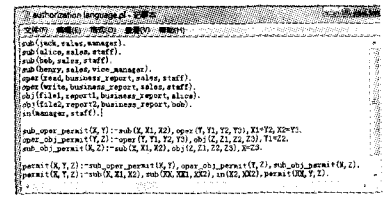
4 与其他访问控制授权语言的对比

基于第2节对已有授权语言的分析，本节对一些常用的访问控制授权语言与SOOSAL进行了简单对比，按照一些常规考查指标，可以用表格简要描述，见表2。表格中的符号“(”表示支持该项功能或该项性能较好，为空则表示不支持或性能不好。

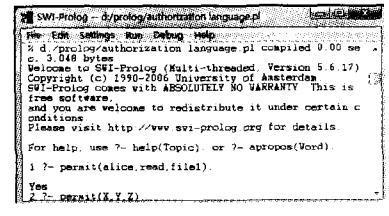
5 实例及分析

在例1中可以发现对于安全需求的变化，描述的安全策略很难适应这种动态变化，而且对于策略的变更和维护代价较高。本节应用SOOSAL授权语言对例1给出的情况进行具体的描述。描述的具体事实和规则如下：

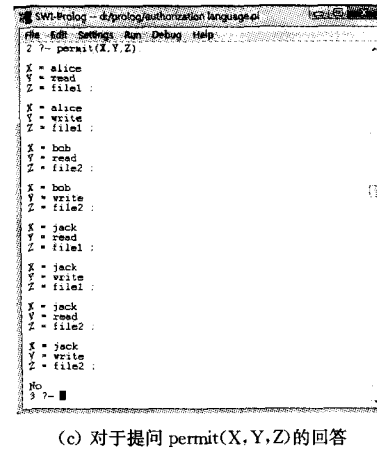
```
sub(jack,sales,manager)
sub(alice,sales,staff)
sub(bob,sales,staff)
sub(henry,sales,vice_manager)
oper(read,business_report,sales,staff)
oper(write,business_report,sales,staff)
obj(file1,report1,business_report,alice)
obj(file2,report2,business_report,bob)
in(manager,staff)
sub_oper_permit(X,Y):- sub(X,X1,X2),oper(Y,Y1,Y2,Y3),X1=Y2,X2=Y3
oper_obj_permit(Y,Z):- oper(Y,Y1,Y2,Y3),obj(Z,Z1,Z2,Z3),Y1=Z2
sub_obj_permit(X,Z):- sub(X,X1,X2),obj(Z,Z1,Z2,Z3),X=Z3
permit(X,Y,Z):- sub_oper_permit(X,Y),oper_obj_permit(Y,Z),sub_obj_permit(X,Z)
permit(X,Y,Z):- sub(X,X1,X2),sub(XX,XX1,XX2),in(X2,XX2),permit(XX,Y,Z)
```



(a)实例程序



(b) 对于提问 permit(alice,read,file1)的回答



(c) 对于提问 permit(X,Y,Z)的回答

图2

还可以添加 3.6 节中的约束规则来对上述策略进行安全性检测。如在策略中可以添加规则不一致性规则 error(system):- permit(X,Y,Z),deny(X,Y,Z),然后可以通过提问?—error(system)来判断策略是否存在不一致性问题,如回答 yes 则存在不一致性,如回答 no 则策略是满足一致性的。

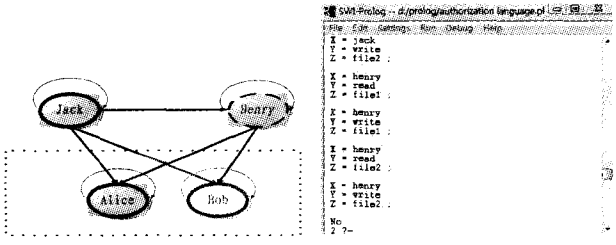
策略定义好后,可以运用下述类似的请求来进行授权的判断。

? —permit(alice,read,file1)

? —permit(X,Y,Z)

该例在阿姆斯特丹大学研发的 SWI—Prolog 编译器中的运行效果见图 2。

(1)权限委托问题:假设经理有事外出,但经理的工作不能停止,此时可以把经理的权限委托给副经理。此时的权限关系见图 3(a)。



(a) 委托后的权限关系图　　(b) 权限委托后的运行结果

图 3

从图 1 到图 3 的权限变化只需增加一条主体委托事实和一条扩展访问规则即可。具体形式如下:

delegate(jack,henry)

permit(X,Y,Z):- permit(XX,Y,Z),delegate(XX,X)

具体的运行结果见图 3(b)。此时可观察到 Henry 也具有 Jack 委托的权限。

而且如果有后续的委托,只需增加相应的委托事实即可。

如果要收回权限的委托,则只需删除委托的事实即可。可保留委托规则,防止有新的委托请求。

(2)权限增加:假设有新的员工 Mary 加入业务部,此时的权限关系见图 4。

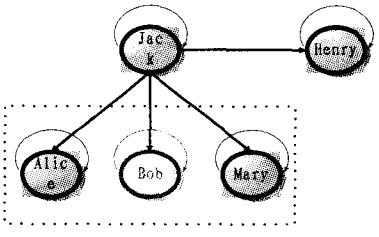


图 4　新员工 Mary 加入后的权限关系图

此时从图 1 到图 4 的权限变化只需增加相应的员工主体事实及其制作的相应业务报表事实,而不需要对规则进行任何修改。

(3)权限修改:假设经理也不能查看和修改员工制作的业务报表,也即经理和员工之间的权限是独立的,此时的权限关系见图 5。

实现从图 1 到图 5 的权限变化只需删除事实 in(manager,staff),处理简单而方便。其他类型的权限修改只需要修改相应的主体、操作和客体事实。

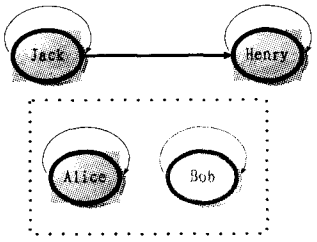


图 5　经理和员工权限独立后的权限关系图

通过对以上实例的分析,使用 SOOSAL 对策略进行描述时,主要分为两部分:事实和规则,事实主要包括各主体、操作和客体的属性描述,以及各相关的其他属性关系事实,比如属性从属事实、委托事实及属性矛盾事实。规则描述主要分为主体和操作访问规则、操作和客体访问规则、主体和客体访问规则、主体、操作和客体访问规则以及其他扩展规则。而在具体应用时,对于同一个系统,规则本身基本不会发生变化。大多的策略需求变化都可以通过事实即各主体、操作和客体的属性变化来实现。应用 SOOSAL 语言,可以很方便地对安全策略进行简单而又有效的描述。而且描述的策略能很好地适应安全需求的动态变化,维护代价低。在实际应用中,还可以根据需要对规则进行扩展以适应具体应用的需求。

结束语　授权语言是对系统安全策略进行描述、验证和执行的基础。在实际应用时,授权语言必须要对各种不同的安全策略、访问模型和授权原则提供足够的支持,并且能够很好地适应系统运行环境的动态变化,比如安全需求的变化、主体、操作和客体各属性的变化甚至是安全策略的变更。基于属性和主体、操作和客体分层描述的逻辑授权语言(SOOSAL)以逻辑为基础,在策略描述中通过谓词和规则对策略 3 大元素和元素之间的关系进行了分层描述,能较好地适应环境的动态变化,维护代价低,并且可以通过规则的增加来实现不同的安全策略和授权原则,以满足实际应用复杂的安全需求。对授权语言本身规则的扩展和在实际应用中的问题分析是今后进一步的工作。

参 考 文 献

[1] 韩道军,高洁,翟浩良,等. 访问控制模型研究进展[J]. 计算机科学,2010,37(11):29-33

[2] Sloman M S. Policy Driven Management for Distributed Systems [J]. Journal of Network and Systems Management,1994,2(4):333-360

[3] Woo T Y C,Lam S S. Authorizations in distributed systems:A new approach[J]. Journal of Computer Security,1993,2(2/3):107-136

[4] Sushil J,Samarati P. Flexible support for multiple access control Policies[J]. ACM Transactions on Database Systems,2001,26(2):214-260

[5] Sushil J,Samarati P,Los Alamitos,et al. A unified framework for enforcing multiple access control policies[C]// Proceedings of the 1997 ACM SIGMOD international conference on Management of Data. New York,USA,1997:474-485

(下转第 364 页)

[M]. Addison Wesley Professional, 2004

[2] 柴晓路,梁宇奇. Web Services 技术、架构和应用[M]. 北京:电子工业出版社,2002

[3] 喻坚,韩燕波. 面向服务的计算—原理与应用[M]. 北京:清华大学出版社,2006

[4] Ryu S H, Casati F, Skogsrud H, et al. Supporting the dynamic evolution of Web service protocols in service-oriented architectures[J]. ACM Transactions on the Web, 2008, 2(2): 1-45

[5] Papazoglou M P. The challenges of service evolution[C]// Proceedings of the International Conference of Advanced Information Systems Engineering. Montpellier, France, 2008: 1-15

[6] 宋巍,马晓星,吕建. Web 服务组合动态演化的实例可迁移性[J]. 计算机学报,2009,32(9):1816-1831

[7] 曾晋,孙海龙,刘旭东,等. 基于服务组合的可信软件动态演化机制[J]. 软件学报,2010,21(2):261-276

[8] Von de Aalst W M P, Jablonski S. Dealing][with workflow change: Identification of issues and solutions[J]. Int'l Journal of Computer System Science & Engineering, 2000, 15(5): 267-276

[9] Andrikopoulos V, Benbernou S, Papazoglou M P. Managing the evolution of service specifications[C]// Proc of the 20th Int'l Conf. on Advanced Information Systems Engineering. 2008: 359-374

[10] Lohmann N. A Feature-Complete Petri Net Semantics for WS-BPEL 2.0[J]. Lecture Notes in Computer Science, 2008, 4937: 77-91

[11] Van der Aalst W M P. Workflow Management Models, Methods, and Systems[M]. The MIT Press Cambridge, Massachusetts London, England, 2002

[12] 李景霞,程久军. Web 服务组合的层次颜色 Petri 网描述模型[J]. 计算机工程,2010,24(12):39-44

(上接第 349 页)

[6] Sushit J, Samarati P, Subrahmanian V. A Logical Language for Expressing Authorizations[C]// Proceedings of IEEE Symposium on Security and Privacy. Oakland, Calif, USA, 1997: 94-107

[7] Lobo J, Bhatia R, Naqvi S. A Policy Description Language[C]// Proceedings of the Sixteenth National Conference on Artificial Intelligence Eleventh Innovative Applications of AI Conference. Orlando, Florida, USA, 1999

[8] Damianou N. Tools for domain-based policy management of distributed systems[C]// Network Operations and Management Symposium. NOMS 2002. IEEE/IFIP, 2002

[9] Damianou N, Dulay N, Sloman M, et al. Ponder: A Language for Specifying Security and Management Policies for Distributed Systems[S]. The language Specification Version 2. 3. Imperial College of Science Technology and Medicine, Department of Computing, London, 2001

[10] Damianou N, Dulay N, Sloman M, et al. The Ponder Policy Specification Language[C]// Proceedings of Policies for Distributed Systems and Networks. HP Labs, Bristol, UK, Springer-Verlag, 2001: 18-38

[11] Damianou N, Dulay N, Sloman M, et al. A policy deployment model for the Ponder language[C]// Proceedings of IEEE/IFIP International Symposium on Integrated Network Management (IM'2001). Seattle, IEEE Press, 2001: 529-543

[12] Hicks B, Rueda S, Clair L S, et al. A logical specification and analysis for SELinux MLS policy[C]// Proceedings of the ACM Symposium on Access Control Models and Technologies. 2007: 91-100

[13] Ahn G J, Xu Wen-juan, Zhang Xin-wen. Systematic Policy Analysis for High-Assurance Services in SELinux[C]// the 9m IEEE Workshop on Policies for Distributed Systems and Networks (POLICY). NY, USA, 2008

[14] Binder D J. A logic-based security language[C]// IEEE Symposium on Security and Privacy. 2002: 105-113

[15] 李星. 访问控制策略语言的研究与设计[D]. 合肥:中国科学技术大学,2009

[16] OASIS. eXtensible Access Control Markup Language(XACML) Version 2. 0 core specification[EB/OL]. www. oasis-open. org/committees/xacml/, 2005

[17] 王雅哲,冯登国,张立武,等. 基于多层次优化技术的 XACML 策略评估引擎[J]. 软件学报,2011,22(2):323-338

[18] 钟勇,秦小麟,郑吉平,等. 一种灵活的使用控制授权语言框架研究[J]. 计算机学报,2006,29(8):1408-1418

[19] Becker M Y, Fournet C, Gordon A D. SecPAL: Design and semantics of a decentralized authorization language[J]. Journal of Computer Security, 2010, 18(4): 619-665

[20] Gurevich Y, Neeman I. DKAL: Distributed-Knowledge Authorization Language[C]// 21st IEEE Computer Security Foundations Symposium. 2008: 149-162

[21] Abadi M. Logic in access control [C]// Proceeding of the Eighteenth Annual IEEE Symposium on Logic in Computer Science. Ottawa, Canada, 2003: 228-233

[22] 颜学雄,王清贤. 基于逻辑的访问控制研究[J]. 计算机科学,2009,36(4):42-46

[23] Crampton J, Loizou G, O'Shea G. A logic of access control[J]. The Computer Journal, 2001, 44(2): 137-149

[24] Dougherty D J, Fisler K, Krishnamurthi S. Specifying and reasoning about dynamic access-control policies[C]// International Joint Conference on Automated Reasoning(IJCAR'06). 2006

[25] Chaudhuri A, Naldurg P, Rajamani S K, et al. EON: Modeling and analyzing dynamic access control systems with logic programs[C]// Proceedings of the 15th ACM Conference on Computer and Communications Security. 2008: 381-390

[26] 马丽,马世龙,眭跃飞,等. 一种 RBAC 的描述逻辑表示方法[J]. 计算机科学,2010,37(3):29-35

[27] 徐震,李斓,冯登国. 基于角色的受限委托模型[J]. 软件学报,2005,16(5):970-978

[28] Lunt T F. Access control policies for database systems[M]. Database Security II: Status and Prospects, North-Holland, Amsterdam, 1989: 41-52

[29] Geri S, Gottlob G, Tanca L. Logic Programming and Databases [M]. Beijing: Springer-Verlag, 1992