OntoRBAC:基于本体的 RBAC 策略描述与集成*)

王治纲　王晓刚　卢正鼎　李瑞轩
(华中科技大学计算机学院　武汉 430074)

摘　要　随着分布式安全系统研究的不断深入和发展,基于策略的体系结构已经逐步成为了访问控制系统中的一种主流方法。然而这些策略的定义均建立在特定的语言基础上,如何在异构的系统间进行集成尚缺乏有效的机制。本文提出了一种基于本体的访问控制策略定义机制——OntoRBAC——它完全支持了 RBAC96 模型,利用本体这个工具,能够为访问控制策略的制定提供更精确、更强大的描述能力,并且可以实现从语义层次上对不同策略的集成。而借助于 SWRL(Semantic Web Rule Language),我们定义了多条适用于 OntoRBAC 的规则来进行对访问控制决策的推理。最后,介绍了这个系统的实现框架。

关键词　本体,策略,基于角色的访问控制

OntoRBAC: Specify and Integrate RBAC Policies with Ontologies

WANG Zhi-Gang　WANG Xiao-Gang　LU Zheng-Ding　LI Rui-Xuan
(College of Computer Science & Technology, Huazhong University of Science & Technology, Wuhan 430074)

Abstract　Policies are becoming an increasingly popular approach to access control systems of applications in academia and industry. As information pertaining to different policies has varying access control requirements, in this paper, we propose the definition of OntoRBAC, an ontology of a general RBAC policy. Supporting and extending the RBAC96 model, it helps to specify role-based access control policies more accurately for the powerful expressive ability. It also provides more effective ways to integrate other policies in the semantic level. Some rules in form of SWRL have been defined to reason within the OntoRBAC. At last, the architecture of implementation is discussed.

Keywords　Ontology, Policy, Role-based access control

1　概述

计算技术和网络技术的不断发展促进了各种分布式应用的增长,同时也引起了对分布式系统安全的更高要求,如何提供一种有效的、能跨多个自治域安全管理方式成为了一项国内外学者普遍关注的问题。基于策略的安全体系框架将访问控制策略与系统安全实现分离,提高了对安全域中的策略进行动态管理的能力,在当前的研究中逐渐得到了越来越多的专家和业内人士的肯定。

各种不同的安全策略定义语言和方法纷纷应运而生,虽然这些方法针对不同的应用领域提供了各自的一套访问策略的定义和使用方法,然而这些方法在处理多域集成的问题上依旧存在这样或那样的缺陷,特别是在解决策略的语义集成问题上有很大的局限性。现有的安全策略研究中主要有以下一些方法:面向对象的策略描述、基于逻辑的策略描述、基于 XML 的策略描述和基于语义网的策略描述。面向对象的策略描述方法[1]可以很好地刻画系统中各种实体之间的关系,与系统实现可以比较好的结合到一起,但是在表达成机器可以理解的语言和集成方面存在一定的局限性。基于逻辑的方法[2~4]用于解决策略的推理方面有天生的优势,但同样地,如何解决不同的逻辑语言之间的映射与集成也是一个比较困难的问题。基于 XML 的策略描述[5,6]在描述工具上使用了 XML,使策略具有良好的语法结构,在集成的过程中可以大大降低集成的难度,但是对集成中的语义问题无法很好地解决;基于语义网的方法[7,8]将语义网这种知识表达方式运用到了策略描述中,和其他几种方法相比能有效地解决分布式环境中的策略集成问题,然而它们只是对一般性的访问控制策略进行了描述,相关的基于 RBAC 模型的研究还很少。

本文在研究了各种访问控制模型的基础上,提出了使用本体的方法来描述基于角色的访问控制策略。借助本体的方式,不仅可以方便地表达访问控制中各种实体之间的关系,而且可以使多安全域的策略集成问题从语义的层面上得到解决,为分布式系统的安全管理提供一个新的思路。

2　基于本体的 RBAC 策略描述

RBAC 作为一种形式化、标准化,具有强大互操作能力的访问控制模型,成为了我们进行对访问控制策略描述的基础。同时借助了语义网这个强有力的知识表达工具,对 RBAC 建立了一系列本体,作为策略描述的基本公理系统。

Entity(实体):囊括了系统中可能涉及到的所有实体的抽象范畴,包括访问控制中的各种主体和客体对象;

Subject(主体):访问控制过程中的主体对象,是访问权限的承载者、访问请求的发出者,在 RBAC 模型中又分为 Agent 和 Role 两个子类;

Agent(代理):是每一个访问请求的发出者以及会话(Session)的创建者,可以被授予或自主地激活一定的角色,

*)本课题获国家自然科学基金项目(60403027)、湖北省自然科学基金项目(2005ABA258)、软件工程国家重点实验室开放基金项目(SKLSE05-07)资助。王治纲　博士生;王晓刚　博士生;卢正鼎　教授,博导;李瑞轩　副教授,博士。

而根据具体的应用环境不同，可以由具体的用户或者是软件代理来充当，因此我们没有直接套用 RBAC96 模型里的 User 概念，而不失一般性地代之以 Agent；

Role(角色)：RBAC 中的核心要素，依靠它将用户与权限联系起来，是 RBAC 模型中实施权限管理和进行访问控制判断的关键环节。通过 juniorRoleOf 属性可以定义父角色与子角色间的关系，从而构造出实际应用系统中的角色层次树；

Resource(资源)：访问控制系统中主要的被保护客体对象，这里是一个抽象的范畴，根据访问控制所应用的不同环境，可以有更加具体的分类或表达形式，例如一个数据库中的表、视图，或者是 Web 应用中的一个网页或 Web 服务等；

Action(行为)：定义了系统中对资源可能进行的各种操作，这里的行为既可以是针对具体资源的访问操作(如：对数据库表的“修改”操作)，也可以是通用性的安全管理方面操作(例如对某一权限的“授予”行为)；

Privilege(权限)：描述了对某些特定对象执行某些操作的能力。由行为和操作对象两部分构成，操作对象就是访问控制中的客体资源；

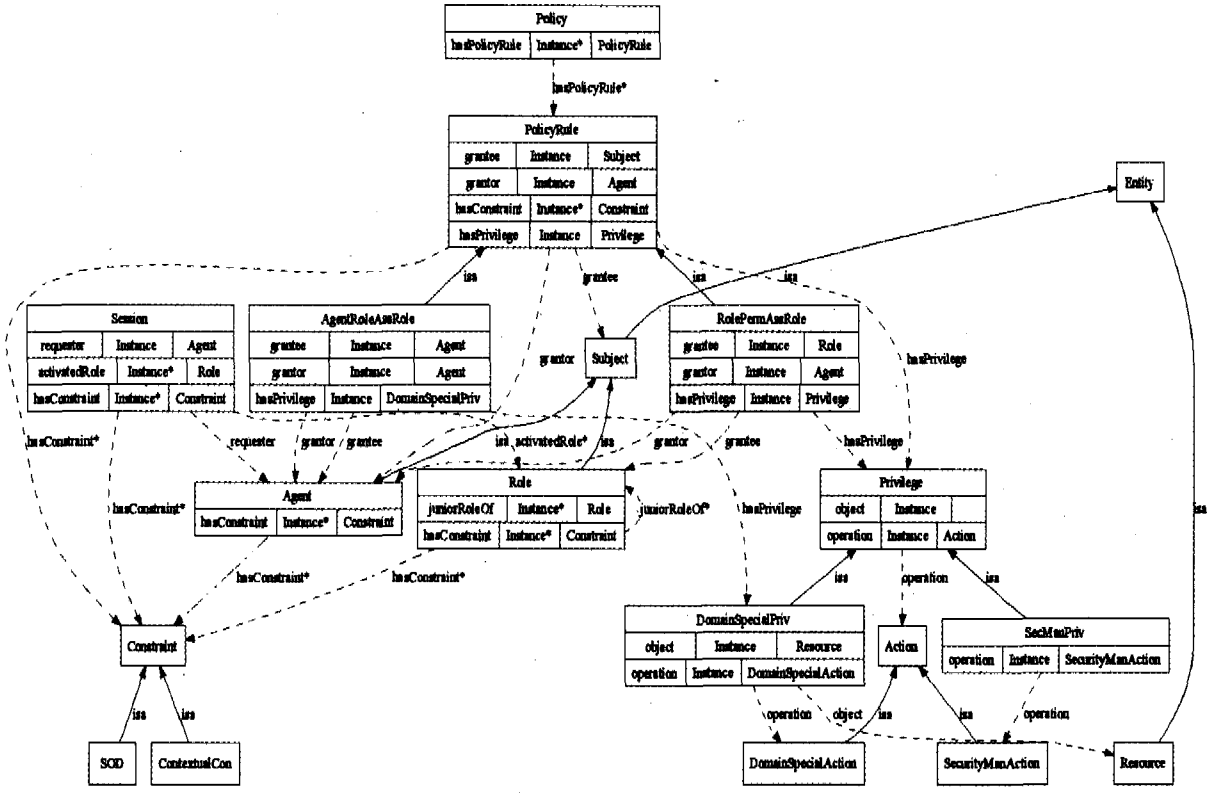

图 1　OntoRBAC 中的概念及其关系

Atom concept:

Entity, Action, Privilege, Policy, PolicyRule, Session, Constraint

Atom Property:

operation(Privilege, Action), object(Privilege, Entity ∪ Privilege),
hasPolicyRule(Policy, PolicyRule), grantor(PolicyRule, Agent),
grantee(PolicyRule, Subject), hasPrivilege(PolicyRule, Privilege),
requestor(Session, Agent) activatedRole(Session, Role),
juniorRoleOf(Role, Role), seniorRoleOf(Role, Role),
hasConstraint(Subject ∪ PolicyRule ∪ Privilege, Constraint)

Compound concept:

$$Subject \subseteq Entity, Resource \subseteq Entity, Agent \subseteq Subject, Role \subseteq Subject,$$
$$DomainSpecialAction \subseteq Action, SecManAction \subseteq Action,$$
$$RoleActivate \subseteq SecManAction,$$
$$DomainSpecialPrivilege \equiv Privilege \cap (=1 operation.DomainSpecialAction) \cap (=1 object.Resource),$$
$$SecManPrivilege \equiv Privilege \cap (=1 operation.SecManAction) \cap (=1 object.(Privilege \cup Subject)),$$
$$RoleActivatePriv \equiv SecManPrivilege \cap (=1 operation.RoleActivate) \cap (=1 object.Role),$$
$$AgentRoleAssRule \equiv PolicyRule \cap (=1 grantee.Agent) \cap (=1 hasPrivilege.RoleActivatePriv),$$
$$RolePrivAssRule \equiv PolicyRule \cap (=1 grantee.Role) \cap (=1 hasPrivilege.Privilege).$$

图 2　用 AAXN 语言对 OntoRBAC 进行的形式化描述

Session(会话):记录一个会话中的基本状态信息,如:请求者、已激活角色等;

Policy(策略):一个访问控制策略的主体部分,由多条策略规则构成;

PolicyRule(策略规则):构成策略的基本单元,反映了RBAC模型中的授权关系,根据模型的特点又可以分为用户-角色授予规则(AgentRoleAssRule)和角色-权限授予规则(RolePriAssRole)两个子类。

Constraint(限制):描述了RBAC模型中可能出现的各种限制的概念。这些限制可能涉及到角色、代理、策略规则等多种实体和对象。

图1中的方框代表了系统中的概念(Concept),实线箭头表示了类与类之间的"包含"关系(isa),虚线箭头代表的是属性(Property),表示了类与类之间的特定关系。AΛXN语言是描述逻辑中常用的一种属性描述语言(Attribute Λ anguage),它提供了对概念否定(Xomlement)和值限制(Number restrictions)的扩充。我们采用AΛXN描述逻辑语言对OntoRBAC的各种概念和关系进行了进一步的形式化描述,具体可以参见图2中所示,有关AΛXN的语法可以参见文[9]。

在OntoRBAC中,最主要的概念就是策略(Policy),每个策略又是由多个策略规则(PolicyRule)组成,策略规则可以分为两大类:AgentRoleAssRule和RolePrivAssRule,它们分别对应于RBAC中用户与角色以及角色与权限的授权关系;通过Session中包含的用户和所激活的角色信息,可以获取该会话所能包含的所有权限;通过定义角色之间的juniorRoleOf的传递性关系,可以表达角色的层次关系;通过对不同概念所包含Constraint的定义,来表达模型中的多种限制。可以看出,我们的本体可以完全实现对RBAC96模型的描述。

3 基于规则的推理

现有的描述逻辑语言和OWL语言能够比较完美地定义和抽象出现实世界中的概念以及概念间的各种关系,并实现以此为基础的本体知识库的可满足性推理。然而由于与生俱来的逻辑基础的缺陷,DL并不能很好地表达出现实世界中各种事物间普遍存在的因果关系。这种形如产生式规则的逻辑关系在实际的知识库应用中往往又是极其重要的,因此实际的本体系统中需要加入另外的机制来满足这种应用逻辑规则的定义和推理。

在定义出OntoRBAC以描述RBAC模型之后,需要将它纳入应用逻辑中,用于检验在具体策略中的用户及角色的访问权限。鉴于实践中OWL本体描述语言的使用,我们采用了SWRL[10]这个结合了OWL DL和Horn逻辑子句的规则描述语言,来定义基于角色访问控制模型中的一系列推理规则,实现本体本身所无法描述的应用逻辑的推理。

R1:$juniorRoleOf(?\ a,?\ b) \wedge juniorRoleOf(?\ b,?\ c) \rightarrow juniorRoleOf(?\ a,?\ c)$,

R2:$juniorRoleOf(?\ a,?\ a) \rightarrow conflict(?\ a)$

juniorRoleOf定义了RBAC中最基本的父角色与子角色之间的包含关系,规则R1说明了这个关系是具有传递性的,可以由它来构造策略中所包含的整个角色树。规则R2表示juniorRoleOf关系是非自反的,conflict是系统中的一个特殊谓词,它表明策略中出现了冲突。可以证明由以上两条规则也可推出juniorRoleOf关系同样是满足非对称性的。这两条规则为后续的规则和推理奠定了基础。

R3:$AgentRoleAssRule(?\ ru) \wedge grantee(?\ ru,?\ u) \wedge hasPrivilege(?\ ru,?\ p) \wedge object(?\ p,?\ r) \rightarrow canPlay(?\ u,?\ r)$

R4:$canPlay(?\ u,?\ r1) \wedge juniorRoleOf(?\ r2,?\ r1) \rightarrow canPlay(?\ u,?\ r2)$,

R5:$canPlay(?\ u,?\ r) \wedge requestor(?\ s,?\ u) \rightarrow canPlay(?\ s,?\ r)$,

规则R3解释了AgentRoleAssRule概念的本质含义,当存在一条用户-角色授权规则,它显式地将某个角色授予了特定的用户,那么这个用户就拥有了激活该角色的权力。而R4说明了这种角色激活在角色树中向下的传递关系,即若一个用户可以激活某个角色,那么它必定可以激活这个角色的子角色。R5表现了角色激活可以根据相应的用户动态地适用于该用户所创建的会话中。

R6:$RolePrivAssRule(?\ ru) \wedge grantee(?\ ru,?\ r) \wedge hasPrivilege(?\ ru,?\ p) \rightarrow canDo(?\ r,?\ p)$

R7:$canDo(?\ r1,?\ p) \wedge juniorRoleOf(?\ r1,?\ r2) \rightarrow canDo(?\ r2,?\ p)$,

R8:$canDo(?\ r,?\ p) \wedge actividRole(?\ s,?\ r) \rightarrow canDo(?\ s,?\ p)$

规则R6~R8与前面的三条规则相对应,解释了角色-权限授予规则RolePrivAssRule的涵义以及隐含的推理关系。R6说明了AgentRoleAssRule显式地将某个权限授予了特定的角色。而R7说明了这个权限可以在角色树中反向继承,即若一个权限被授予一个角色,那么该角色的所有父角色必定可以享有这个权限。R8表现了权限可以根据所激活的角色动态地绑定到一个会话中。

其他类型的规则和推理,由于篇幅的关系,我们就不在这里一一讨论了。

4 策略的集成

语义网和本体的出现,最大的贡献就是提供了一种良好的语义信息集成方法。我们利用这个优点,可以较好地实现基于OntoRBAC本体所描述的安全策略之间的语义集成,也就是一种基于实例的集成方法。

在我们已经实现的安全策略集成中,我们主要提供了两种基本的集成方法:

1. 角色层次映射

角色层次映射是RBAC模型中最主要的跨自治域集成方式,通过定义不同自治域中的角色之间的层次关系来实现策略与策略间的集成。在OntoRBAC中我们可以很方便地定义这种映射关系,例如:juniorRoleOf(p1:emploree, p2:manager),即将自治域p1中的角色emploree定义为了p2中manager的子角色,这里p1、p2分别代表了不同自治域的命名空间,运用前面所讨论的推理规则,可以很方便地推理出域p2中被授予了manager的用户可以激活p1域中的emploree及其子角色,而emploree在p1中的所有权限则可以快速地添加到p2的manager以及它的父角色中,这样就达到了不同策略间集成的目的;

2. 同一性映射

在一些自治域中会发生管理重叠的现象,也就是不同的自治域管理了相同的实体,而这些实体在不同的管理域中可能会有不同的标识。如何在策略集成时,将这些不同策略中针对同一实体的描述统一地管理起来,达到语义上的集成,这

恰恰是其他大部分策略定义语言所无法解决的。而语义网中专门对这一类的"多词同义"问题进行了讨论，并有了一套较为通用的解决方案，即同一性映射。OWL 中提供了 equivalentClass、sameAs、equivalentProperty 这三种词汇分别来声明概念、实例以及属性之间的同一性。在 OntoRBAC 中，可以方便地利用这些词汇描述不同策略中用户、角色、资源甚至权限的"同一性"，这样就可以在策略集成时将不同策略中的同一个实体合并起来，并实现在此基础上的推理。

5 系统实现原型

在已有的工作基础上，我们已经设计并实现了一个基于本体的 RBAC 访问控制系统的初步原型系统，整个系统的体系结构如图 3 所示。体系中各个主要部分的说明如下：

1)OntoRBAC Policy：即我们前面介绍的 OntoRBAC 模型的本体描述，它是知识库(KB)中所有概念和关系的公理集合(Tbox)，系统中的所有的访问控制策略都是根据它来定义和解释的；

2)Local Policy：即一个自治域中的本地访问控制策略，每个策略文件实际上就对应了 Policy 概念的一个实例，它将实际系统中的每个实体或对象，声明为以上 Tbox 中某个概念的实例，每个本地策略文件就是描述了本管理域内的一个访问控制策略实例的集合(Abox)；

3)PDP(Policy Decision Point，策略决策点)：访问控制系统的核心，所有推理工作都是在此完成，担负本体安全策略库的运作，根据策略库中的现有策略规则对访问请求作出评估；

· PolicyLoader：装载相应的策略本体库；检验策略的一致性和可满足性；集成多个策略文件；

· PolicyEngine：依据本体知识作出相关推理，对访问请求作出评判；

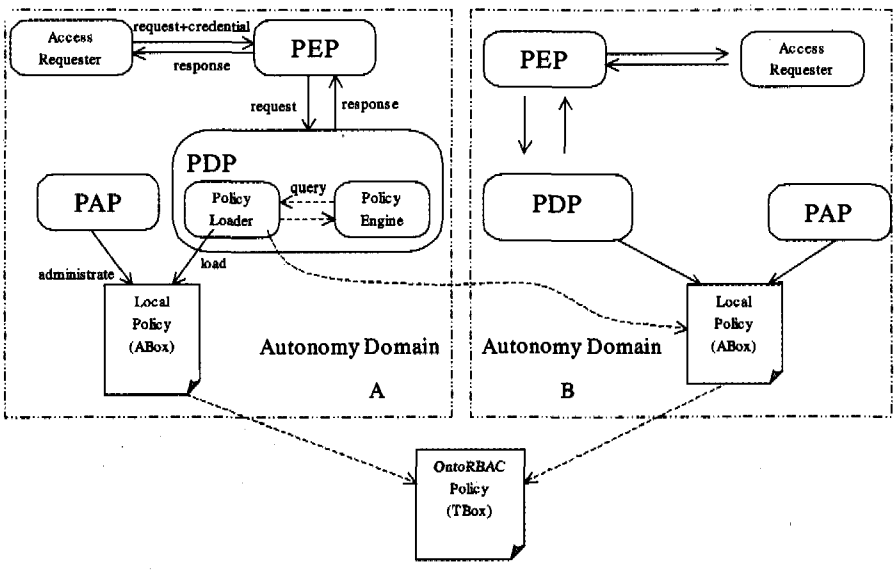

图 3 OntoRBAC 的基本实现框架

4)PEP(Policy Enforcement Point，策略实施点)：按照特定的认证机制检验访问请求者的相关信任状，并将之与相关的环境上下文内容一起，作为访问请求的参数，转交 PDP，最终对请求作出应答；

5)PAP(Policy Administration Point，策略管理点)：对特定本体安全策略库进行维护，利用可视化的工具来更新现有的本体库。

结论 RBAC 在多自治域安全控制方面已经获得了比较广泛的应用，但是不同策略之间的集成方面的研究还有待深入。本文通过定义出可以全面描述 RBAC 策略模型的本体——OntoRBAC，以及借助于 SWRL 定义的推理规则，可以比较好地实现这种访问控制策略间的语义级别上的集成。在今后的工作中我们将进一步完善现有的集成方法，并且努力将集成的领域扩大到不同的本体以及不同的访问控制模型之间的集成，向完全意义上的异构策略集成继续迈进。

参 考 文 献

1 Damianou N, et al. The Ponder Policy Specification Language. In: Proc. Policy 2001, Workshop on Policies for Distributed Systems and Networks, Bristol, UK, Jan. 2001. 18～39

2 Koch M, Mancini LV, Parisi-Presicce F. A graph-based formalism for rbac. ACM Transactions on Information and System Security (TISSEC), 2002. 332～365

3 Jajodia S, Samarati P, Sapino M, et al. Flexible support for multiple access control policies. ACM Transactions on Database Systems, 2001. 214～260

4 Bertino E, Catania B, Ferrari E, et al. A logical framework for reasoning about access control models. ACM Transactions on Information and System Security (TISSEC), 2003. 71～127

5 Joshi J B D, Bhatti R, Bertino E, et al. Access Control Language for Multidomain Environments. IEEE Internet Computing, 2004, 8(6):40～50

6 Moses T, et al. eXtensible Access Control Markup Language (XACML) Version 2.0. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf, 30 Sep. 2004

7 Kagal L, et al. A Policy Based Approach to Security for the Semantic Web. In: Proc. 2nd International Semantic Web Conference (ISWC2003), Sanibel Island, Florida, USA, 2003. 402～418

8 Uszok A, Bradshaw J, Jeffers R, et al. KAoS Policy Management for Semantic Web Services. IEEE Intelligent Systems, 2004, 19(4): 32～41

9 Baader F, Nutt W. Basic Description Logics. In: the Description Logic Handbook, F. Baader, D. Calvanese, D. L. McGuinness, et al, eds. Cambridge University Press, 2002. 47～100

10 Horrocks I, Patel-Schneider P F, Boley H, et al. SWRL: A semantic Web rule language combining owl and ruleml. W3C Member Submission, 21 May 2004. Available at: http://www.w3.org/Submission/SWRL/