Journal of Southeast University (English Edition) Vol. 23, No. 3, pp. 394 – 398 Sept. 2007 ISSN 1003—7985

Novel scheme to specify and integrate RBAC policy based on ontology

Sun Xiaolin Lu Zhengding Li Ruixuan Wang Zhigang Wen Kunmei

(College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: To describe and integrate various policies applied in different domains, the definition of the family of OntoRBAC based on the ontology of a general role-based access control (RBAC) policy is proposed, which can support and extend the RBAC96 model. The uniform ontology-based description mechanism of secure policies is applied in OntoRBAC, which can be used to describe different secure policies in distributed systems and integrate policies in semantic level with upper concepts. In addition, some rules have been defined to reason within the OntoRBAC to extend the inference algorithms in ontology, which makes the system accommodate itself to RBAC policies better.

Key words: ontology; policy; role-based access control

Access control is concerned with permitting only authorized users (subjects) to access services and resources. Policies, which constrain the behavior of system components, are becoming an increasingly popular approach to access control systems of applications in academia and industry. Policies guide the behavior of entities within the policy domain and have been used extensively in security, management and even network routing. And they are often used in systems where flexibility is required as agent, services and access rights change frequently. Authorization or access control policies define the high-level rules specifying the conditions under which subjects are permitted to access targets. The policy-based management has become a promising solution for managing enterprise-wide applications and distributed systems.

Multiple approaches for policy specification have been proposed, which is described formal policy languages that can be processed and interpreted easily and directly by a computer. Furthermore, the heterogeneity of security mechanisms used to implement access control still makes security management an important and difficult task.

The RBAC model is characterized by the notion that permissions are assigned to "roles", and not directly to users. Users are assigned appropriate "roles" according to their job functions, and, hence, indirectly acquire the permissions associated with those roles. RBAC policies are particularly well-suited for large-scale computing systems, because they reduce the administrative complexity of associating users with permissions by decoupling the following two cases: users are authorized for roles, and permissions are assigned to roles.

Received 2007-05-18.

Foundation item: The National Natural Science Foundation of China (No. 60403027).

Biographies: Sun Xiaolin(1980—), male, graduate; Lu Zhengding(corresponding author), male, professor, zdlu@hust.edu.cn.

1 Related Work

1.1 RBAC model

Role-based access-control (RBAC) models show clear advantages over traditional discretionary and mandatory access-control models with regard to these requirements. In particular, an RBAC approach allows uniform representation of diverse security policies and supports efficient access management. We develop the ontology in the context of RBAC96, the most widely known role-based access control model[1-2]. In fact, RBAC96 consists of four different models, RBAC0, RBAC1, RBAC2 and RBAC3. RBAC0 mentions the core model of RBAC, and RBAC1 describes the basic model and role hierarchy, RBAC2 adds constraints to the basic model, RBAC3 is the universal model, besides the basic model, role hierarchy and constraints.

1.2 Policy specification language

There has been much research on various ways of specifying policy in the last decade. We can classify them into logic-based approaches, object-oriented approaches, XML-based approaches and ontology-based approaches.

As logic-based languages always have well-understood formalisms, they have been proved attractive for the specification of security policy. Woo and Lam[3] proposed an authorization language based on default

logic. This approach involves computing extensions of arbitrary default theories, which is NP-complete in the propositional case. Jajodia et al. [4] proposed a formal language ASL based on locally stratified logic, which is known to be decidable. However all of them are difficult to use and are not efficient to implement.

The languages above focused on how to specify the policy. They did not consider of the integration and inter-operation of a modern distributed environment. The use of XML for policy expression has been developed, such as XACML (eXtensible access control markup language)[5], X-RBAC (XML role-based access control)[6]. Both XACML and X-RBAC are based on XML, and they have had broad applications in some enterprise environments for the straightforward extensibility of XML. However, the problem with mere XML is that its semantics are mostly implicit. It is difficult to deal with the inter-operation at the level of semantics, because the entities and relationships defined by XML are lack of formal meaning and they require extra manual work that can be eliminated by a richer representation.

Semantic web-based policy representations can be mapped to semantic level representations, which XML cannot. Some initial efforts in the use of semantic web representations for basic security applications (authentication, access control, data integrity, encryption) of policy have begun. KAoS[7] and Rei[8] are semantic policy languages represented in OWL[9] to specify security policy supporting positive and negative authorization and obligation. The reasoning of KAoS policy is based on DL (description logic), which limits the expressive power of policy, as DL does not support the rule now. As to Rei, it does not support the model of RBAC96 explicitly. In addition, they cannot intuitively specify the important security principle, separation of duty.

2 Family of OntoRBAC Ontology

Directly inspired by Rei and KAoS, we introduce an ontology family, OntoRBAC, to define role-based access control policies extending to the RBAC96 model based on a set of authorization rules. This ontology also provides specification for derived hierarchies and constraints, as well as allows modality authorizations.

The OntoRBAC ontology family is shown in Fig. 1. Model A is the most basic among the family. Other models extend Model A in different ways. This model allows the specification of concepts that can be used in RBAC0 of RBAC96. Model B extends Model A to allow specifying hierarchies of various entities and privileges. Model C extends Model A to allow constraints specification. Modality authorizations are introduced in Model D to describe authorization rules with different modalities, which are seldom mentioned in the RBAC model but useful in actual applications. In the end, Model E is the universal ontology of the family.

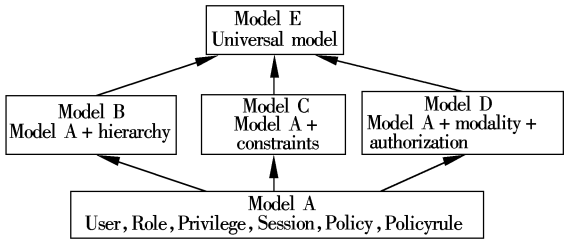

Fig. 1 The family of OntoRBAC

2.1 Model B (hierarchal model)

In Model A, some essential concepts are defined to construct the basis of the OntoRBAC model. Model A and all of the concepts and their relationships are shown in Ref. [10]. So we discuss Model B here as the beginning.

A hierarchy is mathematically a partial order defining a seniority relationship among elements. Whereas only the role hierarchy is noted in RBAC96. The hierarchy is widely used to define the relationships among all kinds of elements in many access-control languages. For conveniently specifying authorization and integrating with other policy models, we introduce several properties in Model B of OntoRBAC to specify not only the role hierarchy, but also the hierarchies among entities, actions and privileges. This model recognizes two types of hierarchies.

- Pure hierarchical RBAC Some systems may focus on the role hierarchy as RBAC96. Most commonly, role hierarchies are limited to simple structures such as trees or inverted trees.
- General hierarchical RBAC In this case, there is support for an arbitrary partial order to serve as the hierarchy to more elements, including the concept of multiple kinds of inheritances of privileges among authorizations.

For pure hierarchical RBAC, two mutual-inverse properties, "seniorRoleOf" and "juniorRoleOf", are defined. They are both transitive, non-reflexive and antisymmetric relations, used to constructure the role hierarchy. If seniorRoleOf(r_i, r_j) is denoted, then r_i is senior to r_j. As well, juniorRoleOf(r_j, r_i) mentions the junior relation between r_j and r_i.

For general hierarchical RBAC, we use the property "dominate" to denote the hierarchies between

more different elements, which range from entity and action to privilege. From these hierarchies, more privileges can be deduced by inheritance. The concept and property added to Model B can be seen as follows:

seniorRoleOf$^+$ (role, Role), juniorRoleOf$^+$ (Role, Role), dominate$^+$ (Entity $\cup$ Action $\cup$ Privilege, Entity $\cup$ Action $\cup$ Privilege), seniorRoleOf $\subseteq$ dominate

2.2 Model C (constraint model)

Constraint specifies conditions over the agent, role, action and any other context entity that must be true at the time of the invocation. Constraints specification is an extensively discussed issue both in general, and within the context of RBAC in specific. We introduce the constraints in Model C, which extends Model A by allowing the specification of constraints in the authorization rules using different methods. Our goal is to provide a general method of constraints specification to express as many kinds of constraints as possible. In contrast, the decidability of these constraints, which may differ from each other, will not be discussed in this paper. There are mainly four types of specifying constraints in RBAC.

- Separation of duty (SoD) constraints These are the most common class of role-based constraints to prevent fraud. The existence of a role hierarchy facilitates the specification of separation of duty constraints because of its ability to model organizational structures. More specially, SoD can be organized as SSoD (static separation of duty) and DSoD (dynamic separation of duty).
- Prerequisite constraints The prerequisite role is discussed widely in ARBAC99. They are based on competency and appropriateness, whereby a user can be assigned to role r_i only if the user is already a member of role r_j.
- Cardinality constraints These constraints specify the maximum number of members in a role in a static or dynamic state. It is useful when some roles can only be assigned to a certain number of users like a manager of a branch in a bank, a chairman of a department, etc. Also, this type is useful in enforcing licensing agreements.
- Context constraints They are defined as dynamic RBAC constraints that check the actual values of one or more contextual attributes. These general attributes rely on environment information like location, time, process-state or access history, etc. and may change dynamically according to real-world.

The constraints above are all classified into SimpleConstraints, for each constraint expresses one kind of condition to be satisfied. Then CompoundConstraint is defined to combine arbitrary constraints using boolean operators of And, Or and Not. Each Boolean constraint expression is a subclass of CompoundConstraint. The concept and property added to Model C can be seen as follows:

SimpleConstraint $\subseteq$ Constraint, CompoundConstraint $\subseteq$ Constraint, SoD $\subseteq$ SimpleConstraint, SSoD $\subseteq$ SoD, DSoD $\subseteq$ Sod, Cardinality $\subseteq$ SimpleConstraint, Prerequisite $\subseteq$ SimpleConstraint, ContextConstraint $\subseteq$ SimpleConstraint, NegativeConstraint $\subseteq$ CompoundConstraint, DisjunctiveConstraint $\subseteq$ CompoundConstraint, ConjunctiveConstraint $\subseteq$ CompoundConstraint, hasConstraint(Entity $\cup$ Privilege $\cup$ PolicyRule, Constraint)

2.3 Model D (modality authorization model)

Modality authorization is rarely mentioned in RBAC literature, mainly because RBAC models such as RBAC96 and the proposed NIST standard model are based on positive privileges that confer the ability to do something on holders. While modal logic expresses many normative notions that can be useful in policy specification. Model D of OntoRBAC extends Model A to allow the specification of modality authorizations. We focus on the set of modal operators of obligation, permission and prohibition, as defined in standard deontic logic and widely accepted. These policy modes can be described as follows:

Permission $\subseteq$ Modality, Obligation $\subseteq$ Permission, Prohibition $\equiv$ Modality $\cap$ $\neg$ Permission, hasMod (Privilege, Modality), rulePirior (PolicyRule, int), policyPirior (Policy, int), defaultMod (Policy, Modality), conflictedMod(Policy, Modality)

- Permission is the action of permitting an allowance, a liberty or a licence granted to do something. Permission policies are designed to give permissions to actors to run some actions on other system objects. They are conceptually enforced near the actor instead of target objects to be affected by the operations the actor performed.
- Prohibition is an unambiguous statement or rule, regulating disallowed behaviour in a system, while a prohibition privilege refers to a prohibition to execute some action.
- Obligation defines what actions an actor must do on a target object. The obligation policies do not depend on permission policies.

Modality authorization is typically discussed in the context of access control systems that adopt open policy. There is an extensive amount of work in this regard. The introduction of modality authorization implies the possibility of conflict in authorization, an issue that needs to be resolved in order for the access control model to give a conclusive result. The types of conflicts brought by the modality authorization and conflict resolution polices are discussed in abundance out-

side RBAC literature. However, it is more complicated in the RBAC model, especially, when Model D is combined to Model B and Model C.

There are many methods to resolve the modality conflict in the RBAC literatures, such as negative/positive policy precedence, assigning explicit priorities to policies, specific overrides general, configure meta-policy, etc. Much research has revealed that any one of the conflict resolution methods is insufficient to resolve all conflict types. In OntoRBAC, the priorities and meta-policies are defined to conclude which kind of authorization can be achieved when conflicts are detected. Due to the limitation of length, the details are omitted.

2.4 Model E (universal model)

Model E is the combination of Models B, C and D, obviously a superset of RBAC3 of RBAC96. Role hierarchy, as well as hierarchies of entities, actions, privileges, can be specified for Model B. Any constraint expressions, which include SoD, cardinality, prerequisite and context constraints, can be asserted. More specially, modality authorizations are introduced to support the RBAC model as in many real-world applications. Therefore, OntoRBAC has such a powerful expressivity, that it not only can specify the policies in respect of RBAC96, but also can express many other policies out of range of the classic model.

3 Rule Based Specification and Integration of Policy

A knowledge base (KB) comprises three components, the Tbox, the Abox and the Rbox. The Tbox introduces the terminology, i. e., the vocabulary of an application domain, while the Abox contains assertions about named individuals in terms of this vocabulary. A set of rules are defined in the Rbox to reason with the KB in application environments. The ontology family above is considered as the Tbox. We can then define any access control policy of special autonomy domain as individuals in Abox by importing that Tbox in OWL files.

It is convenient to integrate two or more policies within the same ontology, OntoRBAC. For the traditional RBAC model, the integration or interoperation means exactly the mapping between roles of different autonomy domains. Based on OntoRBAC, there are many methods to achieve the integration at the semantic level.

According to the mechanism of the mapping, we provide hierarchic mapping and identical mapping.

- Hierarchic mapping Assert the relationships between different parts of hierarchies in distinct policies, so that privileges can be inherited crossing the boundary of domains.
- Identical mapping It is the classic semantic integration that makes the semantic web superior to XML and other languages. It resolves the problem that the same object has different identifies in different autonomy domains. Then this object can be dealt with in both policies.

With Tbox and Abox, all the relationships can be caught between the concepts and individuals. However, there are still some applications in which logic cannot be expressed. Accordingly, we define some rules under the RBAC model corresponding to submodels of OntoRBAC as follows:

R_1: AgentRoleAssRule(?ru) ∧ grantee(?ru, ?u) ∧ hasPrivilege(?ru, ?p) ∧ object(?p, ?r) → canplay(?u, ?r)

R_2: RolePrivAssRule(?ru) ∧ grantee(?ru, ?r) ∧ hasPrivilege(?ru, ?p) → canDo(?r, ?p)

R_3: dominate(?a, ?b) ∧ dominate(?b, ?c) → dominate(?a, ?c)

R_4: seniorRoleOf(?r1, ?r2) → juniorRoleOf(?r2, ?r1)

R5: seniorRoleOf(?r1, r2) → dominate(r1, r2)

R_6: seniorRoleOf(?a, b) ∧ seniorRoleOf(?b, ?c) → seniorRoleOf(?a, ?c)

R_7: operation(?p1, ?a) ∧ operation(?p2, ?a) ∧ object(?p1, ?o) ∧ object(?p2, ?o2) ∧ dominate(?o1, ?o2) → dominate(?p1, ?p2)

R_8: operation(?p1, ?a1) ∧ operation(?p2, ?a2) ∧ object(?p1, ?o) ∧ object(?p2, ?o) ∧ dominate(?a1, ?a2) → dominate(?p1, ?p2)

R_9: canPlay(?u, ?r1) ∧ seniorRoleOf(?r1, ?r2) → canPlay(?u, ?r2)

R_{10}: canDo(?r1, ?p) ∧ seniorRoleOf(?r2, ?r1) → canPlay(?r2, ?p)

R_{11}: canDo(?r, ?p1) ∧ dominate(?p1, ?p2) → canDo(?r1, ?p2)

R_{12}: dominate(?o, ?o) → conflict()

As shown above, R_1 is defined in Model A. It expresses that an agent can play a role in a session, only if there is a rule of AgentRoleAssRule, which has the agent as the grantee, and has the privilege of activating the role. R_2 is another basic rule in the RBAC model. It defines that a role can perform an action on an object, only if there is a rule of RolePrivAssRule, which has the role as the grantee, and has a privilege with the action on the object. R_3 to R_6 define the relationships among properties of Model B. R_7 and R_8 define how to find domination between Privileges. R_9 to R_{11} are reasoning with hierarchies. R_9 says an agent can play any roles of the junior of the role having been assigned to him. R_{10} denotes a role has all the privileges that his junior role has. R_{11} defines a role has all the privileges dominated by the privilege that has been assigned to it. Also we can define the conflict with hierarchies as R_{12}. "dominate" is a transitive, non-reflexive and anti-symmetric property. But if the hierarchy has been constructed as a cycle, then it must lead to conflict. Role hierarchy conflict is just one kind of this conflict.

4 Conclusion and Future Work

Our work includes the definition of family of OntoRBAC, the ontology of a general RBAC policy, to support the basic model, the hierarchy model, the constraint model and the modality authorization model. The most obvious feature of our work different from earlier work is that we support and extend the submodel of RBAC96 and the NIST RBAC model. It helps users to express their role-based access control policies more accurately because of the powerful expressive ability of OntoRBAC. As well, it provides convenient approaches to integration different policies in semantic level.

While in some aspect the ontology of OntoRBAC is still domain dependent, our future researches intend to develop a more general ontology to integrate with other policy languages. As discussed earlier, the reasoning engine is also an important work under consideration. The rules and reasoning are just a part of the work to make the access control decision. There are still many problems such as the resolution of policy conflict and the problem of trust to be resolved especially in the universal model of OntoRBAC. They are all on schedule.

References

[1] Sandhu Ravi S, Coynek Edward J, Feinsteink Hal L, et al. Role-based access control models [J]. *IEEE Computer*, 1996, **29**(2): 38 - 47.

[2] Jajodia S, Samarati P, Sapino M, et al. Flexible support for multiple access control policies [J]. *ACM Transactions on Database Systems*, 2001, **26**(2): 214 - 260.

[3] Baader F, Calvanese D, McGuinness D, et al. *The description logic handbook*: *theory*, *implementation and applications* [M]. London: Cambridge University Press, 2003: 47 - 100.

[4] Jajodia S, Samarati P, Subrahmanian V S. A Logical Language for Expressing Authorizations [C]//*Proceedings of the IEEE Symposium on Security and Privacy*. Washington, DC: IEEE Computer Society Press, 1997: 31 - 42.

[5] Moses T. eXtensible access control markup language (XACML) [EB/OL]. (2003-02-18) [2007-04-20]. http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf.

[6] Joshi J B D. Access-control language for multidomain environments[J]. *IEEE Internet Computing*, 2004, **8**(6): 40 - 50.

[7] Uszok A, Bradshaw J, Jeffers R, et al. KAoS policy and domain services: toward a description-logic approach to policy representation, deconfliction, and enforcement[C]//*Proc of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*. Washington, DC: IEEE Computer Society, 2003: 93 - 96.

[8] Kagal L, Finin T, Joshi A. A policy language for pervasive systems[C]//*Proc of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*. Washington, DC: IEEE Computer Society, 2003: 63 - 71.

[9] Mike D. OWL web ontology language reference. W3C Recommendation 10 February 2004 [EB/OL]. (2004-02-10) [2007-04-20]. http://www.w3.org/TR/owl-ref/.

[10] Wang Zhigang, Wang Xiaogang, Lu Zhengding, et al. OntoRBAC: specify and integrate RBAC policies with ontologies[J]. *Computer Science*, 2007, **34**(2): 82 - 85. (in Chinese)

一种基于本体的 RBAC 策略与集成方法

孙小林　卢正鼎　李瑞轩　王治刚　文坤梅

(华中科技大学计算机科学与技术学院,武汉 430074)

摘要:为了实现不同异构自治域之间安全策略的统一描述与集成,以本体为基础,提出了一种支持 RBAC96 模型的基于角色访问控制策略定义机制 OntoRBAC. 利用本体的通用性,对不同异构系统的安全策略进行统一描述,并能够利用本体的上层概念描述从语义层次上实现对不同策略的集成,并以此为基础,提出了一套用于策略描述的本体模型族. 为了实现访问控制决策的推理,OntoRBAC 以规则的定义为基础,扩展本体推理算法,使其更适用于描述与推理不同自治域的 RBAC 策略.

关键词:本体;策略;基于角色访问控制

中图分类号:TP301