Active Authorization Management for Multi-domain Cooperation

Yuqing SUN[1], Bin GONG[1], Xiangxu MENG[1], and Zongkai LIN[2]
[1]*School of Computer Science and Technology, Shandong University, China*
{sun_yuqing,gb,mxx }@sdu.edu.cn
[2]*Institue of Computing Technology, Chinese Academy of Sciences, Beijing, China*
lzk@ict.ac.cn

Abstract

In a multi-domain collaboration environment, an enterprise should authorize different access rights for sensitive information to partners according to its security policies and relationships with them, which may be changed dynamically with the development of transaction and business rules. So, it is emerging as one of the major concerns to effectively manage the authorizations while supporting flexible multi-level collaboration. In this work, we propose an active authorization model for multi-domain cooperation, which introduces the notions of business rules and context parameters to update security policies automatically and satisfy the dynamic context requirements. The algorithms of handling authorization queries and roles mapping are also presented. The system architecture is discussed in detail to implement this model and support interoperation among heterogeneous platforms.

Keywords: cooperation, access control, RBAC

1. Introduction and Motivation

System interoperation and services sharing are becoming new paradigms for multi-domain collaboration. Generally, an enterprise will authorize different access rights for sensitive information to their partners according to its security policies and the relationships with them. Furthermore, the security policies and relationships may be changed dynamically according to business rules and transactions of underlying system. For example, the relationship of *partner* will be updated to *VIP partner* if their trade amount exceeds 50 million dollar within one year and thereby the partner will be assigned more privileges. Also the accessing policies are possibly changed with the business development, like the qualification threshold of *VIP partner* may be elevated to 100 million dollar of trade amount. So, it is emerging as one of the major concerns for enterprises to effectively manage the authorizations in supporting flexible multi-level collaboration.

However, the traditional access control models are suitable for predefined regulation of access to resources and are adjusted manually with policy changes, which is time consuming and error-prone. In this paper, we propose a novel model to well adapt the dynamically changed policies while reducing the complexity of security management. It focuses on active authorization management for multi-domain cooperation, which introduces the notions of business rule and context parameter into RBAC [1] to enforce the dynamic update policy considering the underlying transaction context. Also, we present the algorithms to handle the authorization queries and role mappings precondition of satisfying security constraints. The system architecture is discussed in detail to implement this model and support interoperation among heterogeneous platforms.

The remainder of this paper is as follows. Next section presents related works. The proposed model is introduced in section 3. Section 4 gives several algorithms and discusses the security constraints. In the following section the system architecture is given together with the processes. At last, we draw some conclusions and future wok.

2. Related Works

Role based access control (RBAC) model is widely adopted to secure the resources of information system [1]. RBAC associate permissions with roles, and users are associated with roles, thereby acquiring permissions of roles. Roles within an organization typically have overlapping permissions and as such can be organized in role hierarchies. Constraints are used to reflect the security policies of an organization, like Separation of duty (SoD) that formulates multi-person control policy to discourage fraud. Although RBAC provides a powerful mechanism to secure large systems, manual adjustment should be processed to enforce security policies changes.

Rule based authorization is regarded as a prospective candidate to carry out change and to provide access

1-4244-0963-2/07/$25.00 ©2007 IEEE.

control [2]. Based on the attributes of users, a family of models called RB-RBAC allows the automatic user-role assignments and provides the specification needed to administrate users' attributes and authorization rules [3,4]. However they focus on a single domain and did not consider the variability of permissions. Bacom et al. propose Open Architecture for Secure Inter-working Services to enable the self-management domains that specify their own access control policies to interoperate using service level agreement [5]. The drawback is lack of considering the influence of permission change. Based on certificates issued by third parties, the Trust Establishment system is presented to define the mappings of strangers to predefined business roles [6]. But it is without considering the relationship among different rules.

Using assignment policies and a trustworthiness threshold specified by system administrator, user could be automatically assigned roles on the Web [7]. But trustworthiness is calculated based on the performance of user; so there exist the possibility to do harm on the system in an extended period of time. Kern et al. present the Enterprise RBAC (ERBAC) model, which is implemented as a Security Administration Management (SAM) Jupiter [8,9]. SAM Jupiter relies on the automatic process of assigning users to roles. However no formal model is given to describe this process. The layered access control model in [10] exploits the SPKI/SDSI to implement and enforce policy specification. But it limits the scenario to the existence of a trusted assigner to declare policies. Shafiq et al. [11] propose a policy integration framework for merging heterogeneous RBAC policies to resolve the conflicts arising from the cooperation. But the complexity is a major problem for multi domains. Although researches in [12,13,14] also discuss the topic of multi-domain cooperation, they are immature with consideration of policy update and context.

3. Active Authorization Management Model for Multi-domain Cooperation

The proposed active authorization management model for multi-domain cooperation is depicted in figure 1. It extends RBAC with the notions of business rules and context parameters. This model is defined in terms of a set of elements and relations among them. The notations of U, R and CT are adopted in this paper to respectively denote the set of users, roles, and constraints.

Users: Users are generally human beings who are assigned responsibilities to perform certain job functions in cooperative business process.

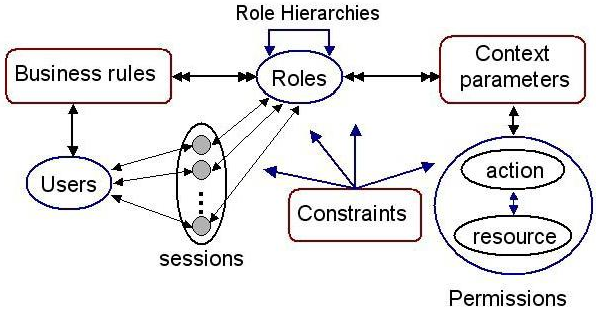


Figure 1. The active authorization management model for multi-domain cooperation

Permissions: Permissions are the approvals of users to access sensitive resources.

Session: A session is an execution instance of a user like "login" that is identified by a system generated ID. A user can invoke multi sessions.

Constraints: Constraints are used to express enterprise security policies and enforced on components of roles, users, sessions and etc.

Roles: a role is a job function with some associated semantics regarding the authority and responsibility. Role hierarchy is defined as below.

Definition 1 (*inheritance relation*) The inheritance relation of two roles *r1* and *r2* is denoted as $r_1 \geq r_2$, in which r_1 inherits all the permissions owned by r_2 while all users of r_1 are the users of r_2.

Definition 2 (*role hierarchy*) Role hierarchy $RH \subseteq R \times R$ is a partial order on R of inheritance relation, with the properties of antisymmetric and transmissible.

3.1. Business Rule

The business rules regulate the relationships among different collaborative partners, which are associated with the underlying transaction system and enterprise business requirements. They are used to dynamically restrict the implicit user-role assignments so as to reflect the flexible policy and support multi-level collaborations.

Definition 3 (*logic operator*) a logic operator is an element of the set $\{\wedge, \vee, \neg\}$, which denote *and, or, not* logic operation respectively. It satisfies the commutative law, associative law and distributive law.

Definition 4 (*relation expression*) A relation expression *ex* is associated with attributes of user and is connected by signs of $\leqslant$, $<$, $=$, $\geqslant$, $>$ etc.

Definition 5 (*business logic expression, abbr.BLE*) A *BLE* is defined as a combination of relation expression with logic operators, $BLE = ex_1 \mid ex_2 \wedge ex_3 \mid ex_4 \vee ex_5 \mid \neg ex_6$, where ex_i is a relation expression.

Definition 6 (*Basic business rule, abbr.BBR*) A basic business rule is defined as the form of $(r_i, ble) \rightarrow (r_j)$, where r_i and r_j are roles, *ble* is a business logic expression. We say that the user who is assigned role r_i can take on role r_j when *ble* is satisfied, that is *ble* yields r_i to r_j.

Here is an example of *Basic business rule* with R={*ne_partner, partner, VIP_partner, supplier, senior_spplier, distributor, senior_distributor, audit*} and *BBR*= (*distributor, (sale>1000)* $\vee$ *(quantity>* 100000)) $\rightarrow$ (*senior_ distributor*).

This means if the trade amount of a distributor company exceeds 1000 million dollar or the sale quantity is larger than 100000 pieces, it would be elevated from the role of *distributor* to *senior_distributor*. Consequently he will take the associated privileges like querying the detailed information about new products.

Above definitions reflect that more attributes can simultaneously influence user role assignments in one rule. However, it cannot express that several business rules with different impact factors simultaneously influence the assignments. So, the compound business rule is introduced to manipulate this case.

Definition 7 (*compound business rule*) A compound business rule is defined as the form of $(r_i, Com\_ble, A, threshold) \rightarrow (r_j)$, where

- r_i and r_j are roles,
- *Com_ble* is in form of vector $(E_1, E_2, \ldots, E_m)$ and each E_i is a business logic expression,
- A is in form of vector $(\alpha_1, \alpha_2, \ldots, \alpha_m)$ and α_i is an impact factor in the business rule with $0<\alpha_i<1$, $\sum_{i=1}^{m} a_i = 1$, and
- *threshold* is the threshold for this business rule to be hold with 0<*threshold*<1.

We say that if the following evaluation comes into existence, this compound business rule will hold and the user who has been assigned the role r_i is changed to take on role r_j:

$$\sum_{i=1}^{m} \alpha_i * \mathrm{E}_i > threshold$$

Furthermore, we should consider the transaction history for business rules. For example, although a partner has a good record in past years and satisfies the qualification of *VIP partner*, he still cannot get good evaluation if no progress in recent years. So we introduce the *TRACE* mechanism to store the history information about each partner so as to maintain the dynamic relationships. The considered history is divided into several periods and each one is called an interval. Considering different impacts of intervals, they are assigned with different influenced factors. The *TRACE* data will be updated periodically so as to reflect the "*fresh*" progress. This is called the historical business rule.

Definition 8 (*historical business rule*) A historical business rule is defined as the form of $(r_i, Com\_ble, A, B, threshold) \rightarrow (r_j)$, where

- r_i and r_j are roles,
- *Com_ble* is in form of vector $(E_1, E_2, \ldots, E_m)$, and each E_i is in form of business logic expression vector $(E_{i1}, E_{i2}, \ldots, E_{im})$,
- A is in form of vector $(\alpha_1, \alpha_2, \ldots, \alpha_m)$ with $0<\alpha_i<1$, $\sum_{i=1}^{m} a_i = 1$, which denote the impact factors of different business logic expression in *Com_ble,*
- B is in form of interval weight vector $(\beta_1, \beta_2, \ldots, \beta_k)$ with $0<\beta_i<1$, $\sum_{i=1}^{k} \beta = 1$, and
- $threshold \in [0,1]$ is the threshold for this business rule to be hold.

We say that if the following evaluation come into existence, this historical business rule will hold and the user who has been assigned the role r_i is permitted to take on role r_j.

$$\begin{pmatrix} E_{11}, E_{12}, \ldots, E_{1m} \\ E_{21}, E_{22}, \ldots, E_{2m} \\ \ldots\ldots \\ E_{k1}, E_{k2}, \ldots, E_{km} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \ldots \\ \alpha_m \end{pmatrix} (\beta_1, \beta_2, \ldots, \beta_k) > threshold$$

The above determinants are easy to calculate because each logical expression E_{ij} is either *TRUE* or *FALSE*, respectively denoted as 1 or 0. The threshold is set according to the business requirements of an enterprise. It can be easily adjusted with the business development.

When an organization defines the business rules for security policy, we make the following two assumptions. A new registered web user will be issued an elementary role according to the predefined qualification. The business rules also should subject to security constraints. For example, suppose that a SoD constrait is required for the roles of *VIP_partner* and *senior_distributor*, any company is not permitted to assign the new role of *VIP_partner* if it has taken on *senior_distributor*. This will be discussed in section 4.

3.2. Context Parameter

Active authorization also means that access control decision should take into account the context. Supposing in a bid scenario, although a user is assigned to the role of *partner* and is authorized to submit its proposal, he is limited within an effective period before deadline. So, it is necessary to consider the system environment. In this model, we introduce the notion of context parameter for permission to consider environment so as to ensure doing right thing in right time. Generally, the context includes system restraints, object constraints and security constraints, which are defined as conditions below.

Definition 9 (*system condition, abbr. yc*) A *yc* is defined as a logic expression concerning system attributes like system time, IP address etc.

Definition 10 (*object condition, abbr. oc*) An *oc* is defined as a logic expression concerning object attributes like the items or table in database, which is used to reflect the restraints of business transaction data.

Definition 11 (*security condition, abbr. sc*) A *sc* is associated with the security constraints like SoD.

Definition12 (*context parameter*) Context parameters are enforced on permissions and defined as a predicate with the form of $per(yc,oc,sc) \rightarrow boolean$, where *yc, oc,sc* are of the environment condition, object condition and security condition for permission *per* respectively. They are the prerequisites of permission invocation. We say that permission *per* can be invoked only if all conditions are hold.

The follow example is given to illustrate the usage of context parameter.

Modify_bid_info("*sys.data* < *deadline*", "*u.app_bid* = *bid.serialno*" , "*role*!=*third_party*")

This means if a user is authorized the permission of *Modify_bid_info* to modify the bid detailed information via the role of *partner*, he can invoke it only when all the assumptions hold: the time is with the effective period before deadline, the bid that he wants to modify is that he submitted with the identity serial number, also the role he is taking on is not the *third-party*. If all the parameters restraints are satisfied, the predicate will return *TRUE* and the permission can be invocated.

In above definition, if one or two types of conditions are without consideration, the corresponding parameters in the permission predicate can be set as the value of *True* to ensure the effectiveness and efficiency of performance.

4. Authorization Query and Role Mapping

In a multi-domain collaborative environment, an organization should exactly know what users are authorized to access its concrete sensitive resources at a concrete time and it is crucial to ensure the authorizations follow the predefined security constraints. On the other hand, he should get knowledge of its privileges of accessing the sensitive resources in another organization when cooperating with partners. This section will firstly investigate these two problems for the proposed active authorization management model and then give the analyses on security constraints.

Authorizations query: The key point is to decide whether a user's (*u*) request to activate permission *p* should be granted. We should firstly verify whether the requested permission *p* satisfies all the context parameters. And then to calculate the roles set that are authorized the specific permission *p* and check whether there exist a user role assignment for the specific user *u* to be authorized one of the above roles.

We adopt Role Graph (RG) to express the role hierarchies, in which the nodes denote roles while the directed edges denote the inheritance relations. There should not exist circle in RG because the role hierarchies are not self-inherited, which can be easily verified by topological sort algorithm. So for a specific permission *p*, the set of roles that are authorized permission *p* include the directly assigned roles and their senior roles who inherited *p* from his juniors. The algorithm of *authorization_query* is given below to perform authorization query.

Name: *authorization_query* (*u, p*)
Input: a specific user *u* and the requested permission *p* in a given system
Output: *True* if permitted; *False,* otherwise

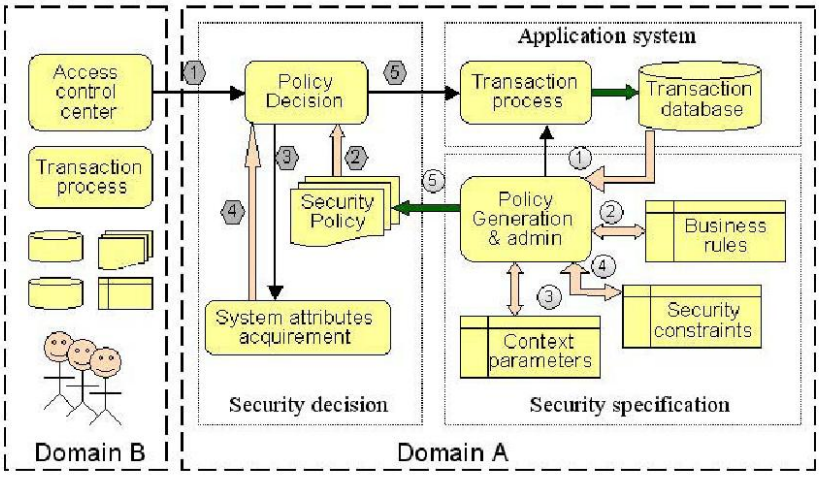
1. if not *p*(*yc,oc,se*) then return *False*
2. Establish the role graph. RG= (R, E), where E={(r_i, r_j)| $r_i \geq r_j$}.
3. For each permission role assignment (*p,r*) do step4-step5
4. *Set_role* =*Set_ role* ∪ { *r* }
5. From node *r*, adopt DFS with the edge input direction to enumerate each ***senior*** node *r'* of *r* and add it to the *Set_role*
   Set_role= *Set_role* ∪ { *r'* }
6. For each role *r*∈*Set_role* do step7-step8
7. if there exist the user role assignment (*u,r*)
8. then Return *True*
9. Return *False*

Role mapping: When a system need to allow a previously unknown system to access its resources, mechanisms should be in place to ensure that the accesses granted are limited to pre-defined sharing requirements. So it is important to map a user request to a set of roles that are authorized to the user, especially in the presence of role hierarchies, and further to calculate the set of permissions that have been assigned to the roles.

For a particular user *u*, the set of acquired permissions is the union set of permissions authorized to roles that associate with *u* in URA. And for a particular role, the acquired permissions set include the direct assignments and those inherited via role hierarchies. The mapping algorithm is given.

Name: *URole_mapping* (*u*)
Input: a specific user *u* in a given system
Output: the set of roles that are authorized to *u*

1. Establish the role graph. RG=(R,E), where E={(r_i, r_j)| $r_i \geq r_j$}.
2. For each user role assignment (*u,r*) do step3
3. *Uroles*= *Uroles* ∪ { *r* }
4. For each *r*∈*Uroles* do step5- step6
5. From node *r* and considering the edge output direction, adopt depth first search (DFS) to enumerate each ***junior*** node *r'*
6. *Uroles* = *Uroles* ∪ { *r'* }
7. Return *Uroles*



Phase 1 policy generation

① Inquire the transaction database periodically
② Get the business rules to update the user role assignments
③ Get the context parameters and bind to permissions
④ Verify the consistency of update assignments with security constraints
⑤ Generate and update XACML-base security policies

Phase 2 authorization decision

① A XACML-based authorization query
② Match authorization with the conditions of the required authorization in security policies
③ Invoke the module of system attributes acquirement if need
④ Get the system attributes
⑤ Transfer the query to the legacy application system

Figure 2. System Architecture and Remark.

Security constraints are crucial for an enterprise to enforce security policies and authorizations should satisfy all the security constraints. As an example, the Separation of Duty (*SoD*) constraint is discussed here, which formulates multi-person control policy by expressing as mutually exclusive permissions. We adopt the predicate of *role_user_mapping*(*role*) to enumerate the set of users who acquire the given role, which is similar with above algorithms and omitted here.

SOD constraint: If *SOD constraint* $rs=(n, r_1, r_2, \ldots, r_n)$ is required for a set of roles $r_1, r_2, \ldots$ and r_n, then these roles are not assigned to the same user. Formally: for $1<i<n$, $\forall\ rs=(n, r_1, r_2, \ldots, r_n) \in SoD$, $r_1, r_2, \ldots r_n \in R$

$$\Rightarrow \left(\bigcap_{i=1}^{n} \text{role\_user\_mapping}(r_i)\right) = \emptyset$$

5. System Overview

5.1. Architecture

The system architecture is designed to implement the proposed model and depicted in figure 2. Black thin arrow lines denote the commands while the thick arrow lines denote date flow.

For simplicity, we adopt two collaborative domains of A and B for illustration. From the aspect of functionality, each domain consists of three parts: operational application system, security specification and security decision.

In each domain, the part of application system denotes the legacy system to handle operational business transactions, like the ERP system in an enterprise. The security specification part is the platform to define business rules, context parameters and administrate security constraints etc. The part of security decision is to make the judgment of an authorization query. To support the interoperation among heterogeneous platform, we adopt XACML to define the security policies, which is the extensible access control marked language that is approved as OASIS Standards.

5.2. Enforcement of Security Policies and Authorization Management

The security management is narrated in the following two phases, which are depicted as the circle and diamond nodes in figure 2, respectively:

Policy generation (phase 1): The module of *policy generation and administration* inquires the transaction database periodically and update the security policies as XACML-based authorization according to the business rules and security constraints. Five steps are given to describe this process by the circle nodes of figure 2.

Authorization decision (phase 2): When a system receives an authorization query, the module of *policy decision* matches the parameters of query with the conditions of permission. If the conditions requirements are associated with system attributes, the module of *system attributes acquirement* will be invoked. In the case of satisfying all conditions, the module of *policy decision* will transfer the query to the application system and otherwise refuse. The corresponding steps are indexed by the diamond nodes of figure 2.

As an example, part of our research results has been applied into a practical system of "Property Rights

Exchange System". Details are omitted here and can be found in literature [15].

6. Conclusions and Future work

In this paper we propose the active authorization management model for multi-domain cooperation. It extends RBAC by the notions of business rules and context parameters to solve the problem of flexible access control in a dynamic environment. Several algorithms are presented to handle the authorization queries and roles mapping. The system architecture is provided to implement this model. For future works, we are investigating to combine the trust and delegation with this model so as to meet the special authorizations. Also we plan to consider the semantic authorization to support the universal access control on web.

7. Acknowledgment

This work is supported by the Nature Science Foundation of Shandong Province (Y2004G08), the National High Technology Research and Development Program (863 Program) of China （2006AA01A113）, and the Science Development Plan Program of Shandong province of China (2004GG2201131).

8. References

[1] R.S.Sandhu and D.Ferraiolo. "The NIST Model for role-based access control: towards a unified standard [S]", Proc. of the 5th ACM workshop on RBAC, 2000. pp.47-63

[2] S.Busch, B.Muschall, G.Pernul and T.Priebe. "Authrule: A Generic Rule-Based Authorization Module", Proceedings of the 20th Annual IFIP WG 11.3 Working Conference on Data and Application Security, 2006. pp.267-281

[3]M.A. AI-Kahtani and R.S.Sandhu, "A Model for Attribute-Based User-Role Assignment". Proc. of 18th Annual Computer Security Applications Conference, 2002. pp.353-362

[4]M.A.AI-Kahtani and R.Sandhu. "Induced Role Hierarchies with Attribute-Based RBAC". Proceeding of ACM SACMAT, Jun.2003, Como Italy, pp142-148

[5] J.Bacon, K.Moody and W.Yao, "A Model of OASIS Role-Based Control and its Support for Active Security", In the ACM Transaction on Information and System Security (TISSEC), Vol.5, No.4, pp.492-540, Nov. 2002.

[6] A.Herzberg, Y.Mass, and J.Mihaeli. "Access Control Meets Public Key Infrastructure, or: Assigning Roles to Strangers". Proceedings of the IEEE Symposium on Security and Privacy, 2000.pp. 2-14

[7] Y. Zhong, B. Bhargava, and M. Mahoui. "Trustworthiness based authorization on WWW", in IEEE workshop on Security in Distributed Data Warehousing, New Orlean, 2001

[8]A.Kern, A.Schaad and J.Moffett. "An Administration Concept for the Enterprise Role-based Access Control Model", SACMAT'03, June 1-4, Como, Italy, 2003, pp.3-11

[9]A Kern. "Advanced Features for Enterprise-Wide Role-Based Access Control". In Proceedings of the 18th Annual Computer Security Applications Conference, Las Vegas, Nevada, USA, Dec. 2002, pp333-342

[10]J.Biskup and S.Wortmann. "Towards a Credential-Based Implementation of Compound Access Control Policies". Proceedings of the ninth ACM symposium on Access control models and technologies, New York, USA ACM SACMAT'04 (2004) pp.31-40

[11]B.Shafiq, J.B.D.Joshi, E.Bertino and A.Ghafoor. "Secure Interoperation in a Multidomain Environment Employing RBAC Policies". IEEE Transaction on Knowledge and Data Engineering, Vol.17, No.11 (2005) pp.1557-1577

[12] J.S.Park, K.P.Costello, T.M.Neven and J.A.Diosomito. "A Composite RBAC Approach for Large, Complex Organization". Proc. of ACM SACMAT, 2004. pp.163-171

[13]E.Song, R.Reddy, R.France, I.Ray, G.Georg and R.Alexander. "Verifiable Composition of Access Control and Application Features". Proceedings of SACMAT, Stockholm (2005) pp.120-129,

[14]Ferraiolo, D. F., Gavrila, S., Hu, V., Kuhn and D. R.: Composing and Combining Policies under the Policy Machine. Proc. of ACM SACMAT, Stockholm (2005) pp.11-20

[15]Y.Q.SUN, P.PAN. "PRES—A Practical Flexible RBAC Workflow System", proceeding of The 7th International Conference on Electronic Commerce, Xi'an, China, Aug.2005. pp.653-658