# An Analysis of Flexibility on RBAC Model

**ZHANG Mingsheng[1], MA Xinqiang[2], ZHAO Min[1]**

*1. College of Economics & Management, Guizhou University for Nationalities, Guiyang, China*

*2. Chongqing University of Arts and Sciences, Chongqing, China*

*e-mail*: *gyzhangms@126.com, mxq345@sohu.com, zhminx@163.com*

**Abstract:** In an access control system, expressiveness and flexibility have been become the top requirements. On the basis of analyzing the flexible mechanisms of RBAC model, the simulation of the security policy of BLP in RBAC is formally enforced, showing how to use the RBAC framework to validate the key BLP policies. In addition, a flexible authorization framework based on logic program is proposed, and it may make good use of the flexible and powerful role mechanism in RBAC for complex authorization administration.

**Keywords:** flexibility of RBAC; simulation of BLP; flexible authorization framework; formal analysis

## 1. Introduction

Nowadays, expressiveness and flexibility have become top requirements for an access control system [1]. On the other hand, role-based access control model (RBAC) was accepted as an ANSI/INCITS standard in 2004[2]. This is absolutely a significant event in the computer security scope as same as TCSEC standard based on BLP (Bell-LaPadula Model) in 1983. Therefore, it might be clear that RBAC is more and more important in computer secure theory and practice. Actually, the number of practical access control systems that offer RBAC features is coming to be expected and growing rapidly in large organizations for the expressiveness and flexibility of authorization policies. In order to further employ RBAC model, we must go deep into it.

In this paper, the flexibilities of RBAC model will be explored based on the following two motivations:

1) Role-based access control (RBAC) is an alternative to traditional discretionary and mandatory access control policies that is attracting increasing attention, particularly for commercial applications. Different from the conventional discretionary and mandatory access controls, role-based access control attempt to merge the flexibility of explicit authorizations with additionally imposed organizational constraints. We will describe using a formalization of RBAC to model BLP policies, and it is studied how to implement lattice-based access control policy using the mechanisms of RBAC. In addition, the study can also explore some essential characteristics of RBAC, and can also help understand clearly the mechanism of RBAC.

2) It is well known that one of the driving motivations of RBAC model is to simplify security policy administration while facilitating the definition of flexible customized policies. In fact, for access control purposes in a large number of business activities, it is much more important to know what a user's organizational responsibilities are, rather than who the user is. The role mechanism in RBAC model can well realize this. Thus we will propose an authorization framework based on logic program, which can make good use of the mechanism of roles.

The rest of this paper is arranged as follows: in Section 2, we describe and summarize the flexibilities of RBAC; simulation of the security policy of BLP by RBAC and a flexible authorization framework based on logic program are stated in Section 3 and Section 4 respectively; at last, the conclusions and relevant work are discussed.

## 2. Flexibilities of RBAC

RBAC models regulate the access of users to the information resources on the basis of the organizational activities and responsibility that users have in a system. In RBAC, permissions are assigned to roles, where roles can be defined as sets of actions and responsibilities associated with a particular working activity, and users are associated with appropriate roles to acquire the roles' permissions. Users can obtain authorizations through roles. Moreover, roles may be created for the various job functions in an organization, and users can be assigned roles in terms of their responsibilities and qualifications. Also, users can be easily reassigned from one role to another, and roles can be granted new permissions in a new appli-

cation environment. The user playing a role is allowed to execute all accesses from which the role is authorized. In general, a user can take on different roles on different occasions, and the same role can be played by several users, perhaps simultaneously. Some proposals (e.g. [2,3]) allow a user to exercise multiple roles at the same time, while some proposals (e.g. [4]) limit the user to only one role at a time, or recognize some roles can be jointly exercised while others must be adopted in exclusion to one another.

This mechanism can greatly simplify management of permissions, and make RBAC very flexible and powerful. Although the administration of users and access privileges in large enterprises is a complex and challenging task, RABC has the flexible and powerful functions for simplifying access control. In fact, RBAC can not only be a promising alternative to traditional discretionary and mandatory access controls, but also articulate different policies by means of precise configurations and interactions of various RBAC components. For instance, the common forms of DAC and LBAC (Lattice Based Access Control) models can be simulated and enforced in RBAC with systematic constructions [5]. For another instance, Barker and Stuckey [6] extended the standard RBAC, and enable security administrators to define a range of access polices with the features like denials of access and temporal authorizations. These features are often useful in practice, but are not widely supported in existing access control models. It is more important that representing access policies as constraint logic programs makes it possible to support constraint checks and administrator queries, and also enables access requests and constraint checks to be efficiently evaluated. However, RBAC is only well suited for use with relatively static, closed and centralized system. When applied in distributed computing contexts, RBAC has certain shortcomings. Thus in the recent years, the researchers have proposed some models extending RBAC, such Action-Status Access Control (ASAC) [7].

## 3. Simulating the Security Policy of BLP in RBAC

Based on analysis of the flexibility of RBAC, we have proposed the problem: a formal analysis for implementing LBAC in RBAC, which refer to how to implement lattice-based access control (LBAC) policy using the mechanisms of RBAC. The problem proposed may be based on the following idea: since many existing security systems are based on the mechanisms of RBAC, can these existing systems fit to the classical and traditional access control policy BLP?

We will do it by the following definitions and theorems.

**Definition 1** (RBAC Configuration)

$S_{RBAC}=_{def}<U \times S \times R,\ UA \cup PA \cup user \cup roles,\ RH>$, a configuration of RBAC not considering administrations.

We consider authorizations in RBAC model as two kinds of access in the following: Can-Access and Current-Access.

**Definition 2** (Can-Access)

$<Ui, Oj, p>can=_{def} \exists R_k \bullet ((U_i, R_k) \in UA \wedge ((O_j, p), R_k) \in PA)$

Definition 2 represents that the user Ui have the power or right of the access to the object Oj with the mode p. {<Ui, Oj, p>can} is similar to access matrix D[i,j] in BLP model. Note that <Ui, Oj, p>can would be performed by a or many roles of the user Ui.

**Definition 3** (Current-Access)

<Si, Oj, p>current$=_{def}$

$\exists R_{Si} \exists P_{Si} \bullet \{ (R_{Si} \in \{r \mid \exists r' \geq r \bullet ((user(S_i), r') \in UA)\}) \wedge (P_{Si} \subseteq \{p' \mid \exists r'' \leq R_{si} \bullet ((p', r'') \in PA)\} \wedge (O_j, p) \in P_{hi}) \}$

Definition 3 may indicate that the session Si is currently accessing to the Oj by the role $R_{Si}$ and with the permission $P_{Si}$, however it may also imply that the task <Si, Oj, p> current can be performed by the several roles and their corresponding permissions. {<Si, Oj, p> current} is corresponding to access matrix M[i,j] in BLP model. In fact, we describe only the necessary condition of <Si, Oj, $p>_{current}$. Obviously, it is certain that <Si, Oj, p>current $\Rightarrow \exists$ Ui $\bullet$ <Ui, Oj, p>can, i.e. Ui is one user of Si.

**Definition 4** (BLP model)

BLP=(M, D, λ), there are the following elements:

SUB={S1,S2,...,Sm}, a set of subjects;

OBJ={O1,O2,…,On}, a set of objects;

P, a fixed set of rights, such as r, w;

D, an m × n discretionary access matrix with D[i,j] $\subseteq$ P;

M, an m × n current access matrix with M[i,j] $\subseteq$ {r,w};

Λ={λ1, λ2, …, λk}, confidentiality labels generally with a partially ordered;

λ: SUB ∪ OBJ→Λ, static assignment of confidentiality



978-1-935068-06-8 © 2009 SciRes.

labels;

$\geq \subseteq \Lambda \times \Lambda$, called dominate relation with a partial order;

$p \in M[i,j]=<Si,Oj,p>$, Si is accessing to Oj with the p mode, where $p \in P$.

**Definition 5** (BLP Security Policy)

In BLP model, there are the definitions about discretionary property (ds-property), simple security property (ss-property) star property (*-property) respectively as follows:

ds-property: $r/w \in M[i,j] \Rightarrow r/w \in D[i,j]$,

ss-property: $r \in M[i,j] \Rightarrow \lambda(S_i) \geq \lambda(O_j)$, and

*-property: $w \in M[i,j] \Rightarrow \lambda(S_i) \leq \lambda(O_j)$.

**Definition 6** $S_{BLP1}$=def<$U_{BLP1} \times R_{BLP1} \times S_{BLP1}$, $UA_{BLP1} \cup PA_{BLP1} \cup user_{BLP1} \cup roles_{BLP1}$, $RH_{BLP1}$>, where

$U_{BLP1}$=U, $S_{BLP1}$=S;

$R_{BLP1}$={λ1Read, λ2Read,…, λkRead}, indicates the roles of "read";

$RH_{BLP1}$=<$R_{BLP1}$,≥ >, i.e. λiRead≥λjRead iff λi≥λj;

$UA_{BLP1}$={(Ui, Rj)|Rj= $\lambda$ (Ui)Read ∧ Rj ∈ $R_{BLP1}$};

--$PA_{BLP1}$={(Oi,p),Rj)|p={r ∧ $\lambda$ (Oi)Read ≤ Rj ∧ Rj ∈ $R_{BLP1}$};

$user_{BLP1}$={(Si, Ui)|Si ∈ S, Ui ∈ U};

$roles_{BLP1}$={(Si, Rj)| ∃ r ∈ $R_{BLP1}$ • [Rj= r ∧ ($user_{BLP1}$(Si), r) ∈ $UA_{BLP1}$] ∧ Si ∈ S ∧ Rj ∈ $R_{BLP1}$};

**Theorem 1:** $S_{BLP1}$ satisfies ss-property in BLP

Proof: It is sufficient to prove $r \in M[i,j] \Rightarrow \lambda$ (Si)≥ $\lambda$ (Oj)

From Definition 4.2, $r \in M[i,j]$ corresponding <Si, Oj, r>current. By user BLP1, we get that the session Si responds to Ui. By roles BLP1 and user BLP1, we can know that the role of Si is just the $\lambda$ (Ui) Read, and then the permission of Si is just one of $\lambda$ (Ui) Read. Again by PA BLP1, we can get $\lambda$ (Oi) Read ≤ $\lambda$ (Ui) Read. Therefore, by the definition of RH BLP1, we can get that $\lambda$ (Si)≥ $\lambda$ (Oj).

**Definition 7** $S_{BLP2}$=def<$U_{BLP2} \times R_{BLP2} \times S_{BLP2}$, $UA_{BLP2} \cup PA_{BLP2} \cup user_{BLP2} \cup roles_{BLP2}$, $RH_{BLP2}$>, where

$U_{BLP2}$=U, $S_{BLP2}$=S, $user_{BLP2}$=$user_{BLP1}$;

$R_{BLP2}$={λ1Write, λ2Write,…, λkWrite }, indicates the role of "write";

$RH_{BLP2}$=< $R_{BLP2}$, ≥>, i.e. λi Write≥λj Write iff λi≥λj;

$UA_{BLP2}$={(Ui, Rj)|Rj= $\lambda$ (Ui) Write ∧ Rj ∈ $R_{BLP2}$};

$PA_{BLP2}$={(Oi, p), Rj)|p={w} ∧ $\lambda$ (Oi) Write≥Rj ∧ Rj ∈ $R_{BLP2}$};

$roles_{BLP2}$={(Si, Rj)| ∃ r ∈ $R_{BLP2}$ • [Rj= r ∧ ($user_{BLP2}$ (Si), r) ∈ $UA_{BLP2}$] ∧ Si ∈ S ∧ Rj ∈ $R_{BLP2}$};

**Theorem 2:** $S_{BLP2}$ satisfies *-property in BLP.

Proof: Proof is similar to Theorem 1

**Definition 8** $S_{BLP3}$=def<$U_{BLP3} \times R_{BLP3} \times S_{BLP3}$, $UA_{BLP3} \cup PA_{BLP3} \cup user_{BLP3} \cup roles_{BLP3}$, $RH_{BLP3}$>, where

$U_{BLP3}$=U, $S_{BLP3}$=S, $user_{BLP3}$=$user_{BLP2}$=$user_{BLP1}$;

$R_{BLP13}$={λ1Read,λ2Read,…,λkRead,λ1Write, λ2Write,…,λkWrite }, λiRead indicates the role of "read", λjWrite indicates the role of "write";

$RH_{BLP3}$=<$R_{BLP3}$, ≥>, i.e. λiRead≥λjRead iff λi≥λj, and λi Write≥λj Write iff λi≥λj;

$UA_{BLP3}$={(Ui, Rj)|[(Rj= $\lambda$ (Ui)Read ∧ Rj ∈ R BLP3] ∨ [Rj= $\lambda$ (Ui)Write ∧ Rj ∈ R BLP3]};

$PA_{BLP3}$={(Oi, p), Rj)|[(Si,Oj,r) ⇒ p={r} ∧ $\lambda$ (Oi)Read ≤ Rj ∧ Rj ∈ $R_{BLP3}$] ∨ [(Si,Oj,w) ⇒ p={w} ∧ $\lambda$ (Oi)Write≥Rj ∧ Rj ∈ $R_{BLP3}$]};

$roles_{BLP3}$={(Si, Rj)| ∃ r ∈ $R_{BLP3}$ • [Rj= r ∧ ($user_{BLP13}$(Si), r) ∈ $UA_{BLP3}$] ∧ Si ∈ S ∧ Rj ∈ $R_{BLP3}$};

**Theorem 3:** $S_{BLP3}$ satisfies ss-property and *-property in BLP.

Proof: Note that $UA_{BLP3}$ and $PA_{BLP3}$ are selective. We can select "read" when ss-property, and "write" when *-property respectively. Therefore the remaining proofs are similar to the above statement.

Through the case above, BLP model may be implemented in BRAC.

## 4. Flexible Authorization Framework

Our proposed authorization framework is composed of the programs of three main parts: PRAP module, URAP module and UR-RP authorization policy module, where PRAP means Privilege Role Assignment Program which is in charge of assigning privileges to roles; URAP expresses User Role Assignment Program whose function is the assignment of roles to users; UR-RP program combine PRAP and URAP for implementing the integration of multiple policies. Moreover, there is data system supporting the three basic modules. The three basic modules constitute a policy program which materializes authorization requirements. The policy program determines authorizations in terms of its semantics. The semantics of the policy program is based on answer sets, but is obtained by computing. That is, our authorization frame-

work can be supported by powerful computing tools. When a user make a request for performing a particular access r on an object o, that is Request (u, p, o), Permitted (u, p, o) are checked whether it belongs the answer set of the policy program. If the result of the checking is "Yes", the request (u, p, o) is granted, and denied otherwise. The completeness of our authorization framework is satisfied by using the closed policy or open closed policy.

The three components of the authorization framework are simply described in the following:

PRAP module consists of ordered logic programs for assigning privileges to roles. The syntax and semantics of PRAP are specified like literature [8].

There are four steps: firstly, the components are original; the knowledge base comes from the propagation of rules in the components according to the hierarchies related to role, privilege and object; the reduction is obtained by conflict resolution; at last, the reduction program produces stable model semantics as the semantics of the whole PRAP program.

URAP module is constructed by stratified logic programs for assigning roles to users. A URAP program is composed of the four rules, which are Explicit Role-User Assignment Authorization with can-ura, Rules for Propagation Policies with deri-ura, Rules for Conflict Revolution Policies with do-ura, and Dynamic Role-User Assignment Authorizations with ura. We can obviously know the program is stratified (i.e., the dependency graph of the program is no cyclic for negation as failure). Therefore the URAP programs are stratified ones.

UR-RP authorization policy enforces multi-policies by combining URAP module and PRAP module. The authorization policy is defined according to the form of rules:

Permitted (u, p, o) ← ura (u, r, g), (o, r). pra (p, g')

¬ Permitted (u, p, o) ← not ura (u, r, g)

¬ Permitted (u, p, o) ← (o, r). ¬ pra(p, g)

where,

1) Predicate Permitted (u, p, o) specifies that a user u has the privilege p access to object o for an access request (u, p, o), i.e. Req (u, p, o) is satisfied by Permitted (u, p, o);

2) The predicate ura (u, r, g) represents dynamic role-user assignment authorization specially designed for dynamic separation of duty;

3) pra (p, g) can be explained by the following:

It refers the meaning of a referential literal and the concept of conflicting referential literal:

The referential literal (o, r1). Pra (write, r2) represent that the role r1 is assigned (or authorized) by the administer role r2 to the privilege write to the object o.

The referential literal (o, r1). ¬ pra (write, r3) denote that administer role r3 assigned (or authorized) the role r1 not to write the object o.

(o, r1). Pra (write, r2) and (o, r1). ¬ pra (write, r3) are conflicting. Note that r2 and r3 may be unequal.

## 5. Conclusions and Relevant Work

### 5.1 About Simulation of the Security Policy of BLP

Although there have been some researches on the relationships between LBAC and RBAC [5,9,10], these researches are informal. Our investigation is based on the formal method related to the ideas of relation, homomorphism and logic etc. The research has the following main contributions:

The formal analysis shows how to use the RBAC framework to validate the key LBAC policies, suggesting that RBAC has a good role to play in unifying the formal treatment of a range of LBAC systems.

The research can conclude that several studied lattice-based access control policies can be carried out in RBAC, and that the mechanism of managing access control in RBAC can be clearly exploited.

Through using one security model as a unifying principle for studying others, we have possibility to explore a way for reasoning about combinations of security policies.

### 5.2 About the Flexible Authorization Framework

The framework can make good used of the role mechanism in RBAC, and it has also the following advantages:

As the administrative mechanism of the framework is based on RBAC, it may be easily further extended and refined.

Since the framework is specified by logic programs, it is flexible. Using RBAC to organize the rules in a specification may enhance the construction of the specification.

The component modules in the authorization can adopt the advantages of the theory above. For instance, PRAP make use of the fine-grained and structural propagation mechanism in Bertino99 Framework for roles, objects



978-1-935068-06-8 © 2009 SciRes.

and privileges; URAP employs the multiple conflict resolution and decision policies in FAF for users. UR-RP explicitly facilitates the session in the literature [6].

The component modules implement independently the policies, and interplay on base of the unifying framework. In fact, the functions of the implementations and interplay are based on the combination of logic programs.

## 6. Acknowledgments

This paper is supported by the Science and Technology Funds of Guizhou Province (Grant No. [2009]2125), and the Stadholder Foundation of Guizhou Province (Grant No. 2009-111).

## References

[1] S.Vimercati, P.Samarati and S.Jajodia. Access control policies and languages. Int. J. Computational Science and Engineering, Vol.3, No.2, 2007.

[2] R.Sandhu, D.Ferraiolo, and R.Kuhn. The NIST model for role-based access control: Towards a unified standard. In Proceedings of 4th ACM Workshop on Role-Based Access Control, 47–61. 2000.

[3] R.Sandhu, E.Coyne, H.Feinstein, and C. Youman. Role- based access control models. IEEE Computer 29, 2, 38–47. 1996.

[4] S.Jajodia, P.Samarati, M.Sapino, and V.Subrahmaninan. Flexible support for multiple access control policies. ACM TODS 26, 2, 214–260, 2001.

[5] S.Osborn, R.Sandhu and Q.Munawer. Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies. ACM Transactions on Information and System Security, Vol. 3, No. 2, May, pp85–106, 2000

[6] S.Barker and P.Stuckey. Flexible access control policy specification with constraint logic programming. ACM Trans. on Information and System Security, 6(4):501–546, 2004.

[7] S.Barker. Action-status access control. Symposium on Access Control Models and Technologies, SACMAT'07, June 20-22, 2007, Sophia Antipolis, France. Proceedings of the 12th ACM symposium on Access control models and technologies, 2007.

[8] E.Bertino, F.Buccafurri, E. Ferrari and P. Rullo. A logical framework for reasoning on data access control policies. Proc. Of the 1999 IEEE Computer Security Foundations Workshop, 1999.

[9] S. Osborn. Mandatory Access Control and Role-based Access Control Revisited. In Proceedings of the Second ACM Workshop on Role-based Access Control (RBAC '97, Fairfax, VA, Nov. 6–7), C. Youman, E. Coyne, and T. Jaeger, Chairs. ACM Press, New York, NY, 31–40, 1997.

[10] M.Nyanchama and S.Osborn. Modeling Mandatory Access Control in Role-based Security Systems. In Database Security VIII: Status and Prospects. Chapman and Hall, Ltd., London, UK, pp129–144, 1996.