Specification and Classification of Role-based Authorization Policies

Gail-Joon Ahn
University of North Carolina at Charlotte
gahn@uncc.edu

Abstract

Constraints are an important aspect of role-based access control (RBAC). Although the importance of constraints in RBAC has been recognized for a long time, they have not received much attention. In this paper we classify RBAC constraints into two major classes called prohibition constraints *and* obligation constraints*. To specify these constraints, we utilize a formal language, named* RCL2000*. In this paper we show that prohibition, cardinality, and obligation constraints can be also represented in* RCL2000*.*

1. Introduction

RBAC has become a well-accepted and well-known approach for authorization and access control in modern systems. RBAC regulates the access of users to the information on the basis of the activities the users execute in the system. It requires the identification of roles in the system. A role can be defined as a set of actions and responsibilities associated with a particular working activity. Then, instead of specifying all the access each user is allowed to execute, access authorizations on objects are specified for roles. Since roles in an organization are relatively persistent with respect to user turn over and task re-assignment, RBAC provides a powerful mechanism for reducing the complexity, cost, and the potential for error of assigning users permissions within the organization. Because the roles within an organization typically have overlapping permissions, RBAC models include features to establish role hierarchies, where a given role can include all of permissions of another role. Another fundamental aspect of RBAC is constraints. Although the importance of constraints in RBAC has been recognized for a long time, they have not received much attention. This paper focuses on constraints in RBAC, and addresses how we can specify constraints in role-based systems, using the specification language *RCL2000*.

1.1. Constraints

Constraints are an important aspect of access control and are a powerful mechanism for laying out higher level organizational policy. Consequently the specification of constraints needs to be considered. So far this issue has not received enough attention in the area of role-based access control. In this paper we identify the major classes of constraints in RBAC such as *Prohibition Constraints* and *Obligation Constraints*, including *Cardinality Constraints*. We also propose a framework and formal language for specification of these identified classes of constraints in role-based systems.

1.1.1. Prohibition Constraints

In organizations, we need to prevent a user from doing (or being) something that he is not allowed to do (or be) based on organizational policy. *Prohibition Constraints* are constraints that forbid the RBAC component from doing (or being) something which it is not allowed to do (or be). A common example of prohibition constraints is separation of duty (SOD). SOD is a fundamental technique for preventing fraud and errors, known and practiced long before the existence of computers. We can consider the following statement as an example of this type of constraint: if a user is assigned to purchasing manager, he cannot be assigned to accounts payable manager. This statement requires that the same individual cannot be assigned to both roles which are declared mutually exclusive.

1.1.2. Obligation Constraints

We also need to force a user to do (or be) something that he is allowed to do (or be) based on organizational policy. We derived another class of constraints from this motivation. *Obligation Constraints* are constraints that force the RBAC component to do (or be) something. The motivation of this constraints is from the simulation of lattice-based access control in

Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03)
1080-1383/03 $17.00 © 2003 IEEE

RBAC [San96]. In [San96], there is a constraint which requires that certain roles should be simultaneously active in the same session. There is another constraint which requires a user to have certain combinations of roles in user-role assignment. We classify this kind of constraints as obligation constraints.

1.1.3. Cardinality Constraints

Another constraint is a numerical limitation for the number of users, roles, and sessions. For example, only one person can fill the role of department chair; similarly, the number of roles (sessions) an individual user can belong to (activate) could be constrained.

Our objective in this paper is to study constraints in context of role-based access control (RBAC) [SCFY96], i.e, on how constraints can be expressed. Constraints can be expressed in natural languages, such as English, or in more formal languages. Natural language specification has the advantage of ease of comprehension by human beings, but may be prone to ambiguities. Natural language specifications do not lend themselves to the analysis of properties of the set of constraints. For example, one may want to check if there are conflicting constraints in the set of access constraints for an organization. We opted for a formal language approach to specify constraints. The advantages of a formal approach include a formal way of reasoning about constraints, a framework for identifying new types of constraints, a classification scheme for types of constraints (e.g., prohibition constraints and obligation constraints), and a basis for supporting optimization and specification techniques on sets of constraints.

To specify these constraints we utilize the specification language *RCL2000* which is the specification language for role-based authorization constraints [AS99, AS00]. For brevity, we omit formal syntax and semantics of *RCL2000*. Who would be the user of *RCL2000*? The first reaction might be to say the security officer or the security administrator. However, we feel there is room for a security policy designer distinct from security administrator. The policy designer has to understand organizational objectives and articulate major policy decisions to support these objectives. The security officer or security administrator is more concerned with day to day operations. Policy in the large is specified by the security policy designer and the actions of the security administrator should be subject to this policy. Thus policy in the large might stipulate what is the meaning of conflicting roles and what roles are in conflict. *RCL2000* is also useful for security researchers to think and reason about role-based authorization constraints.

The rest of the paper is organized as follows. Section reviews previous work on constraints in RBAC. Section discusses how role-based authorization constraints can be expressed in *RCL2000*. In section we characterize the classes of constraints in RBAC. Section concludes the paper.

2. Related Work

Constraints are an important aspect of access control and are a powerful mechanism for laying out a higher level organizational policy. Consequently the specification of constraints needs to be considered. This issue has received surprisingly little attention in the research literature. There is some work such as [CS95, GI96] that deal with constraints in the context of role-based access control. This work, however, is preliminary and tentative, and need substantial further development. Most prior work has focused on separation of duty constraints. Chen and Sandhu [CS95] presented the initial description which *RCL2000* builds on. Even though their description was preliminary, it suggested how constraints can be specified. Giuri and Iglio [GI96] defined a new model to provide the capability of defining constraints on roles. In their model, a role is defined as a *named set of constrained protection domains* (NSCPD) that is activatable only if the corresponding constraint is satisfied. Their description focused on the activation of roles. But we should also consider that constraints can be applied to other components in RBAC.

3. Expressive Power of *RCL2000*

Now, we show how we can specify constraints in *RCL2000*. Particularly we show specification of SOD which is an example of prohibition constraints. Also, we specify cardinality constraints and obligation constraints using *RCL2000*.

3.1. Prohibition Constraints: SOD Constraints

SOD is a well-known principle for preventing fraud by identifying conflicting roles—such as Purchasing Manager and Accounts Payable Manager—and ensuring that the same individual can belong to at most

one conflicting role. Static SOD applies to the user-assignment relation and dynamic SOD applies to the activated roles in session(s). In this section, we show how *RCL2000* can be used to specify the various separation of duty properties.

Static SOD (SSOD) is the simplest variation of SOD. In Table 1 we show our expression of several forms of SSOD. These include new forms of SSOD which have not previously been identified in the literature. This demonstrates how *RCL2000* helps us in understanding SOD and discovering new basic form of it.

Property 1 is the most straightforward property. The SSOD requirement is that no user should be assigned to two roles which are conflicting each other. *RCL2000* can clearly express this property. This property is the classic formulation of SSOD which is identified by several papers including [GGF98, Kuh97, SCFY96]. It is a role-centric property.

Property 2 follows the same intuition as property 1, but is permission-centric. Property 2 says that a user can have at most one conflicting permission acquired through roles assigned to the user. Property 2 is a stronger formulation than property 1 which prevents mistakes in role-permission assignment. This kind of property has not been previously mentioned in the literature. *RCL2000* helps us discover such omissions in previous work. In retrospect property 2 is an "obvious property" but there is no mention of this property in over a decade of SOD literature.

Property 3, another form of SOD, ensures that each role can have at most one conflicting permission without consideration of user-role assignment.

With the condition in property, we can extend property 1 in presence of conflicting permissions as property 4. In property 4 we have another additional condition that conflicting permissions can only be assigned to conflicting roles. The net effect is that a user can have one conflicting permissions via roles assigned to the user. The viewpoint of property 4 is that conflicting roles are designated in advance and conflicting permissions must be restricted to conflicting roles.

Property 5 is a very different property. With a notion of conflicting users, we identify new forms of SSOD. Property 5 says that two conflicting users cannot be assigned to roles in same conflicting role set. This property is useful because it is much easier to commit fraud if two conflicting users can have different conflicting roles in same conflicting role set. This property prevents this kind of situation in role-based systems. A collection of conflicting users is less trustworthy than a collection of non-conflicting users, and therefore should not be mixed up in the same conflict role set. This property has not been previously identified in the literature.

We may also identify a composite property which includes conflicting users, roles and permissions from these properties.

3.2. Cardinality Constraints

Another constraint is a numerical limitation for the number of users, roles, and sessions. For example, only one person can fill the role of department chair; similarly, the number of roles (sessions) an individual user can belong to (activate) could be limited. This numerical limitation may vary depending upon the organizational policy. *RCL2000* can specify these constraints without any extension of language. Along with our expression, we reiterate the requirement of an example constraint as below.

- Numerical limitation $\mathcal{N}$ that exists for the number of roles an individual user can belong to cannot be exceeded.

  $| \texttt{roles}^*(\texttt{OE}(\texttt{U})) | \leq \mathcal{N}$

We have briefly shown that we could express cardinality constraints in *RCL2000* without any extension provided they apply uniformly all roles and all users.

3.3. Obligation Constraints

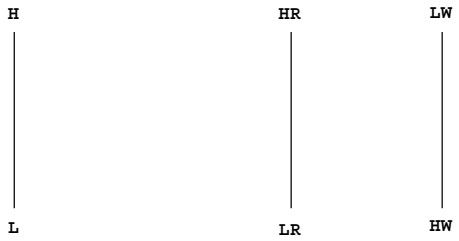
Role-based access control is a promising alternative to traditional discretionary access control (DAC) and mandatory access control (MAC). Sandhu has earlier shown how to simulate several variations of MAC in RBAC, and that Chinese Wall policy is just another Lattice-based information policy. Sandhu and Munawer have recently shown how to simulate a variety of DAC policies in RBAC [SM98]. These results were of theoretical interests because it relates RBAC to the most dominant form of access control. In this section we show how to express these simulations with *RCL2000*, particularly based on Sandhu's LBAC-RBAC simulation [San96].

3.3.1. Lattice-Based Access Control

Lattice-based access control is concerned with enforcing one directional information flow in a lattice of security labels [San96]. LBAC is also known as mandatory access control or multilevel security.

Properties	Expressions
1. SSOD-CR	$\mid \texttt{roles}^*(\texttt{OE(U)}) \cap \texttt{OE(CR)} \mid \leq 1$
2. SSOD-CP	$\mid \texttt{permissions}(\texttt{roles}^*(\texttt{OE(U)})) \cap \texttt{OE(CP)} \mid \leq 1$
3. Variation of 2	$(2) \wedge \mid \texttt{permissions}^*(\texttt{OE(R)}) \cap \texttt{OE(CP)} \mid \leq 1$
4. Variation of 1	$(1) \wedge \mid \texttt{permissions}^*(\texttt{OE(R)}) \cap \texttt{OE(CP)} \mid \leq 1$ $\wedge\ \texttt{permissions}(\texttt{OE(R)}) \cap \texttt{OE(CP)} \neq \phi \Longrightarrow \texttt{OE(R)} \cap \texttt{OE(CR)} \neq \phi$
5. SSOD-CU	$(1) \wedge \mid \texttt{user}(\texttt{OE(CR)}) \cap \texttt{OE(CU)} \mid \leq 1$

Table 1: Static Separation of Duty



(a)A partially ordered Lattice

(b) Roles in RBAC Simulation

Figure 1: A Partially Ordered Lattice

The mandatory access control policy is expressed in terms of security labels attached to subjects and objects. The security labels form a lattice structure as defined below.

Definition 1 (Security Lattice) There is a finite lattice of security labels $\mathcal{SC}$ with a partially ordered dominance relation $\succeq$ and a least upper bound operator. □

A simple example of a security lattice is shown in Figure 1(a) with H > L. Information is only permitted to flow upward in the lattice. In this example, H and L, respectively, denote high and low. This is a typical confidentiality lattice where information can flow from low to high but not vice versa.

The specific mandatory access rules usually specified for a lattice, where λ signifies the security label of the indicated subject or object, are as follows:

Definition 2 (Simple Security) Subject s can read object o only if $\lambda(s) \geq \lambda(o)$. □

Definition 3 (Liberal *-property) Subject s can write object o only if $\lambda(s) \leq \lambda(o)$. □

Definition 4 (Strict *-property) Subject s can write object o only if $\lambda(s) = \lambda(o)$. □

We begin by considering the example lattice of Figure 1(a) with the liberal *-property.

Sandhu [San96] introduced how LBAC can be reconstructed in the context of RBAC. LBAC is enforced in terms of read and write operations. In RBAC this means our permissions are reads and writes on individual objects written as (o,r) and (o,w), respectively. An LBAC object has a single sensitivity label associated with it. This is expressed in RBAC by requiring that each pair of permissions (o,r) and (o,w) be assigned to exactly one matching pair of xR and xW roles, respectively. By assigning permissions (o,r) and (o,w) to roles xR and xW respectively, we are implicitly setting the sensitivity label of object o to x. *RH* is expressed as shown in Figure 1(b)

With *RCL2000* we need additional administrative sets to specify these constraints because these constraints require that each user should have exactly two roles xR and LW and each session should have exactly two roles yR and yW. From this observation, we introduce the following new administrative sets: active role sets (AR), assignment role sets (ASR), write roles (WR), and read roles (RR). Active roles mean that those roles should be active at the same session and assignment roles are roles which a user should have during user-role assignment. With these sets, we can specify the formalization mentioned above as below.

RCL 2000 Specification: (*Liberal *-Property*)

- R = {HR, HW, LR, LW}
- OBJ = $\{obj_1, obj_2, ..., obj_n\}$
- OP = $\{read, write\}$
- P = $\{rp, wp\}$
  rp=$\{(read, obj_1), (read, obj_2), ..., (read, obj_n)\}$

Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03)
1080-1383/03 $17.00 © 2003 IEEE

and
wp={$(write, obj_1), (write, obj_2), ..., (write, obj_n)$}

- RR = {HR, LR}
- WR = {HW, LW}
- AR={ar_1, ar_2}
  ar_1 = {HR, HW}
  ar_2 = {LR, LW}
- ASR={asr_1, asr_2}
  asr_1 = {HR, LW}
  asr_2 = {LR, LW}

Given these sets, such as R, OBJ, OP, P, RR, WR, AR, and ASR, we specify the above constraints described as below.

- Constraint on *UA*:
  roles(OE(U)) = OE(ASR)
- Constraint on sessions:
  roles(OE(sessions(OE(U)))) = OE(AR)
- Constraints on *PA*:
  objects(OE(wp)) = objects(OE(rp)) $\Longrightarrow$
  roles(OE(wp)) $\cup$ roles(OE(rp)) = OE(AR) $\wedge$
  roles(OE(rp)) $\cap$ WR = ϕ $\wedge$
  $\mid$ roles(OE(rp)) $\cap$ RR $\mid = 1$

□

From the initial construction, we can have two sets, ar_1 and ar_2 because both HR and HW roles should be activated, and both LR and LW roles should also be in a session. Also asr_1 and asr_2 are required because each user should have HR and LW (or, LR and LW) in user-role assignment. Therefore, we can express UA constraint and session constraint using ASR and AR sets, respectively. In expression of UA constraints, roles(OE(U)) = OE(OE(ASR)) ensures that a user should have roles exactly equal to one of the ASR sets. In expression of session constraints, roles(OE(sessions(OE(U)))) denotes all roles which a user activates in a single session. The constraint ensures that the roles activated by a user in a session consist of exactly one of the AR sets. Constraint specification of PA ensures that each permission should be assigned to a single object and roles which have read-write permissions on a single object can be invoked in a session. Unlike SOD constraints, we have just shown that we need additional administrative sets of *RCL2000* to specify LBAC constraints and that these constraints need to be forced to activate the predefined roles at the same time.

4. Characterization

We characterize the classes of constraints in RBAC. This characterization is based on *RCL2000* specifications described in this paper such as SOD constraints and LBAC simulation. Also, we refer a sampling of RCL expression in [AS00] to characterize the classes of constraints. From these specifications, we try to characterize each class of constraints with an intuitive and a sharp distinction rather than with exhaustive analysis.

Even though we believe that *RCL2000* helps us discover useful constraints, this paper cannot list all of the existing constraints. We assert that our specifications in this paper cover a subset of each class of constraints. We simply call our characterizations *Simple Prohibition Constraints* class and *Simple Obligation Constraint* class. That is, a small set represents a group of constraints which we identified in this paper.

4.1. Simple Prohibition Constraints

The identified constraints which are classified as prohibition constraints are generally expressed in several similar forms. We generalize these forms to characterize prohibition constraints.

Simple prohibition constraints include:

- Type 1:
  $\mid expr \mid \leq 1$
- Type 2:
  $expr = \phi$ or $\mid expr \mid = 0$
- Type 3:
  $\mid expr \mid < \mid expr \mid$

Most of these forms are identified in the SOD constraints and some can also be found in other case studies.

4.2. Simple Obligation Constraints

Obligation constraints which are identified in our work have several unique forms. We generalize these forms to characterize prohibition constraints.

Simple obligation constraints include:

- Type 1:
  $expr \neq \phi$ or $\mid expr \mid > 0$
- Type 2:
  $set\ X = set\ Y$

- Type 3:
  obligation constraints $\Longrightarrow$ *obligation constraints*
- Type 4:
  $\mid expr \mid = 1$
  This form can be re-expressed with the forms which are identified in simple prohibition constraints and simple obligation constraints as below.
  $\mid expr \mid = 1 \equiv \mid expr \mid \leq 1 \wedge \mid expr \mid > 0$

As we mentioned earlier, most of the constraints which are identified in the simulation of LBAC in RBAC have these types of forms.

5. Conclusion

Role hierarchies and constraints are two fundamental aspects of role-based access control. Although the importance of constraints in RBAC has been recognized for a long time, they have not received much attention in research literature. In this paper we have identified the major classes of constraints in role-based access control such as prohibition constraints and obligation constraints. Such classification of constraints is the first attempt in role-based security. We have described a framework to specify these classes of constraints using *RCL2000*. We also characterized these classes of constraints based on the *RCL2000* specifications. Our distinction is a pragmatic, but it is not theoretically complete. We believe that this distinction will help system security officers to think and to design a system more practically. We are currently exploring a systematic way which can be adopted to utilize this language in security policy architect arena.

Acknowledgment

This work was partially supported at the Laboratory of Information of Integration, Security and Privacy at the University of North Carolina at Charlotte by the grants from National Science Foundation (NSF-IIS-0242393).

References

[AS99] Gail J. Ahn and Ravi S. Sandhu. The RSL99 Language for Role-Based Separation of Duty Constraints. In *Proceedings of 4th ACM Workshop on Role-based access control*, ACM, 1999.

[AS00] Gail J. Ahn and Ravi S. Sandhu. Role-based Authorization Constraints Specification. *ACM Transactions on Information and System Security*, pages 207-226, Vol. 3, No. 4, ACM, November 2000

[CS95] Fang Chen and Ravi Sandhu. Constraints for role based access control. In *Proceedings of 1st ACM Workshop on Role-Based Access Control*, pages 39–46, Gaithersburg, MD, November 1995.

[GGF98] Virgil D. Gligor, Serban I. Gavrila, and David Ferraiolo. On the formal definition of separation-of-duty policies and their composition. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, pages 172–183, Oakland, CA, May 1998.

[GI96] Luigi Giuri and Pietro Iglio. A formal model for role-based access control with constraints. In *Proceedings of IEEE Computer Security Foundations Workshop*, pages 136–145, Kenmare, Ireland, June 1996.

[Kuh97] D. Richard Kuhn. Mutual exclusion of roles as a means of implementing separation of duty in role-based access control systems. In *Proceedings of 2nd ACM Workshop on Role-Based Access Control*, Fairfax, VA, October 1997.

[San96] Ravi S. Sandhu. Role hierarchies and constraints for lattice-based access controls. In Elisa Bertino, editor, *Proc. Fourth European Symposium on Research in Computer Security*. Springer-Verlag, Rome, Italy, 1996. Published as *Lecture Notes in Computer Science, Computer Security–ESORICS96*.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.

[SM98] Ravi S. Sandhu and Qamar Munawer. How to do Discretionary Access Control Using Role. In *Proceedings of 3rd ACM Workshop on Role-Based Access Control*. ACM, 1998.