Proceedings of Second ACM Workshop on Role-Based Access Control, Fairfax, Virginia, November 6-7, 1997

The ARBAC97 Model for Role-Based Administration of Roles: Preliminary Description and Outline*

Ravi Sandhu
George Mason University and
SETA Corporation

Venkata Bhamidipati
George Mason University and
Aspen Systems Corporation

Edward Coyne
SETA Corporation

Srinivas Ganta
Cygnacom Solutions

Charles Youman
SETA Corporation

Abstract

In role-based access control (RBAC) permissions are associated with roles, and users are made members of roles thereby acquiring the roles' permissions. The motivation behind RBAC is to simplify administration. An appealing possibility is to use RBAC itself to manage RBAC, to further provide administrative convenience, especially in decentralizing administrative authority, responsibility and chores. This paper describes the motivation, intuition and outline of a new model for RBAC administration called ARBAC97 (administrative RBAC '97). ARBAC97 has three components: URA97 (user-role assignment '97), PRA97 (permission-role assignment '97) and RRA97 (role-role assignment '97). URA97 was recently defined by Sandhu and Bhamidipati [SB97]. ARBAC97 incorporates URA97, builds upon it to define PRA97 and some components of RRA97, and introduces additional concepts in developing RRA97.

*This work is partially supported by the National Science Foundation at the Laboratory for Information Security Technology at George Mason University and the National Institute of Standards and Technology at SETA Corporation.

All correspondence should be addressed to Ravi Sandhu, ISSE Department, Mail Stop 4A4, George Mason University, Fairfax, VA 22030, sandhu@isse.gmu.edu, www.list.gmu.edu.

1 INTRODUCTION

Role-based access control (RBAC) has recently received considerable attention as a promising alternative to traditional discretionary and mandatory access controls (see, for example, [FK92, FCK95, Gui95, GI96, MD94, HDT95, NO95, SCFY96, vSvdM94, YCS97]). In RBAC permissions are associated with roles, and users are made members of appropriate roles thereby acquiring the roles' permissions. This greatly simplifies management of permissions. Roles are created for the various job functions in an organization and users are assigned roles based on their responsibilities and qualifications. Users can be easily reassigned from one role to another. Roles can be granted new permissions as new applications and systems are incorporated, and permissions can be revoked from roles as needed. Role-role relationships can be established to lay out broad policy objectives.

In large enterprise-wide systems the number of roles can be in the hundreds or thousands, and users can be in the tens or hundreds of thousands. Managing these roles and users, and their interrelationships is a formidable task that often is highly centralized in a small team of security administrators. Because the main advantage of RBAC is to facilitate administration, it is natural to ask how RBAC itself can be used to manage RBAC. We believe the use of RBAC for managing RBAC will be an important factor in the long-term success of RBAC. Decentralizing the details of RBAC administration without loosing central control over broad policy is a challenging goal for system designers and architects.

There are many components to RBAC [SCFY96]. RBAC administration is therefore multi-faceted. In particular we can separate the issues of assigning users to roles, assigning permissions to roles, and assigning roles to roles to define a role hierarchy. These activities are all required to bring users and permissions together. However, in many cases, they are best done by different administrators or administrative roles. Assigning permissions to roles is typically the province of application administrators. Thus a banking application can be implemented so credit and debit operations are assigned to a teller role, whereas approval of a loan is assigned to a managerial role. Assignment of actual individuals to the teller and managerial roles is a personnel management function. Assigning roles to roles has aspects of user-role assignment and role-permission assignment. Role-role relationships establish broad policy. Control of these relationships would typically be relatively centralized in the hands of a few security administrators.

Sandhu and Bhamidipati [SB97] recently introduced a model for user-role assignment called URA97 (user-role assignment '97). URA97 is constructed in context of the RBAC96 model [SCFY96, San97], summarized in figure 1. In this paper we build upon URA97 to develop a comprehensive model for role-based administration of RBAC. Our model is called ARBAC97 (administrative RBAC '97). It has three components as follows.

1. The **user-role assignment** component of ARBAC97 is essentially identical to URA97 and carries the same name.

2. The **permission-role assignment** component of ARBAC97 is a dual[1] of URA97 and is called PRA97 (permission-role assignment '97).

3. The **role-role assignment** component of ARBAC97 itself has several components which are determined by the kind of roles that are involved. We defer discussion of the role-role assignment model till section 4.

The rest of this paper is organized as follows. We begin by reviewing the URA97 model in section 2. In section 3 we define the dual administrative model PRA97. Section 4 describes the motivation, intuition and outlines of the RRA97 model for role-role assignment. A complete definition of RRA97 is outside the scope of this paper. Our focus here is on developing the motivation and intuition which will underlie future formalization of RRA97. The RRA97 model is still evolving, so our discussion here is a snapshot of work in progress. Section 5 concludes the paper.

[1] In our work we have often observed a duality between user-role and permission-role relationships. For example, every constraint on user-role relationships has a dual counterpart with respect to permission-role relationships, and vice versa [SCFY96]. We see this duality exhibited in ARBAC97, where the permission-role assignment model is a dual of URA97.

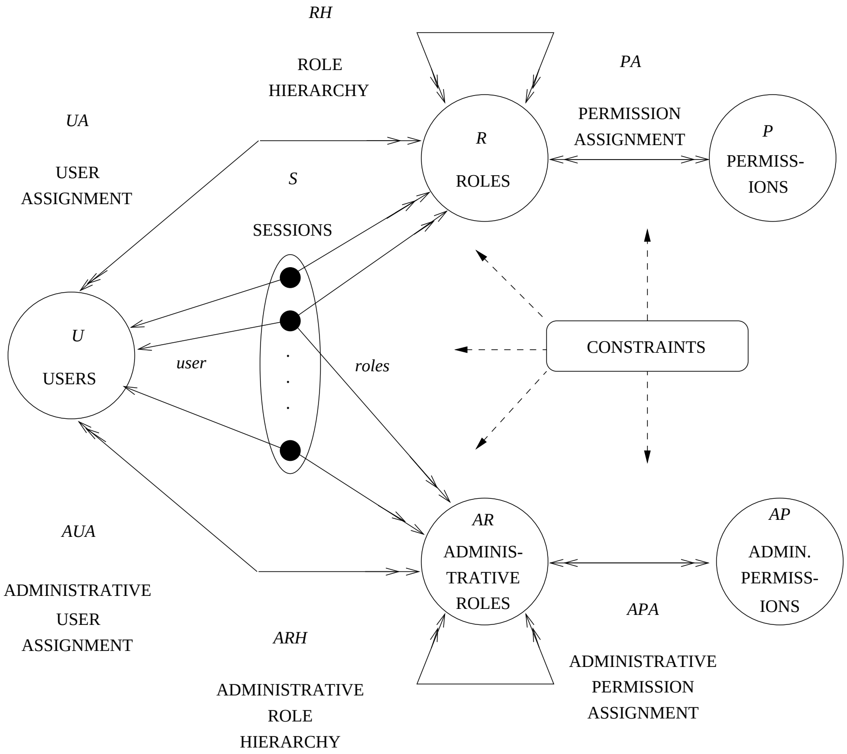
2 URA97 FOR USER-ROLE ASSIGNMENT

The URA97 model was recently defined by Sandhu and Bhamidipati [SB97]. It is constructed in context of the RBAC96 model [SCFY96] shown in figure 1. The top half of the figure shows (regular) roles and permissions that regulate access to data and resources. The bottom half shows administrative roles and permissions. Intuitively, a user is a human being or an autonomous agent, a role is a job function or job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role, and a permission is an approval of a particular mode of access to one or more objects in the system or some privilege to carry out specified actions. Each session relates one user to possibly many roles. The idea is that a user establishes a session and activates some subset of roles that he or she is a member of (directly or indirectly by means of the role hierarchy).

We use the hierarchies of figures 2(a) and 2(b) in our running example through this paper. Senior roles are shown towards the top and junior ones towards the bottom. Senior roles inherit permissions from junior ones and are said to dominate them. Figure 2(a) shows the regular roles in an engineering department. There is a junior-most role E to which all employees belong. The engineering department has a junior-most role ED and senior-most role DIR. In between there are roles for two projects within the department, project 1 on the left and project 2 on the right. Each project has a senior-most project lead role (PL1 and PL2), a junior-most engineer role (E1 and E2), and in between two incomparable roles, production engineer (PE1 and PE2) and quality engineer (QE1 and QE2). Figure 2(b) shows the administrative role hierarchy with the senior security officer (SSO) role at the top, and two project security officer roles (PSO1 and PSO2) and a department security officer (DSO) role.

URA97 is concerned with administration of the user-assignment relation UA which relates users to roles. Authorization to modify this relation is controlled by administrative roles. Thus members of the administrative roles in figure 2(b) are authorized to modify membership in the roles of figure 2(a). Assignment of users to administrative roles is outside the scope of URA97 and is assumed to be done by the chief security officer.

There are two aspects to decentralization of user-role assignment. We need to specify the roles whose membership can be modified by an administrative role. We also need to specify a population of users eligible for membership. For example, URA97 will let us specify that the administrative role PSO1 can assign users to



- U, a set of users
  R and AR, disjoint sets of (regular) roles and administrative roles
  P and AP, disjoint sets of (regular) permissions and administrative permissions
  S, a set of sessions

- $UA \subseteq U \times R$, user to role assignment relation
  $AUA \subseteq U \times AR$, user to administrative role assignment relation

- $PA \subseteq P \times R$, permission to role assignment relation
  $APA \subseteq AP \times AR$, permission to administrative role assignment relation

- $RH \subseteq R \times R$, partially ordered role hierarchy
  $ARH \subseteq AR \times AR$, partially ordered administrative role hierarchy
  (both hierarchies are written as $\geq$ in infix notation)

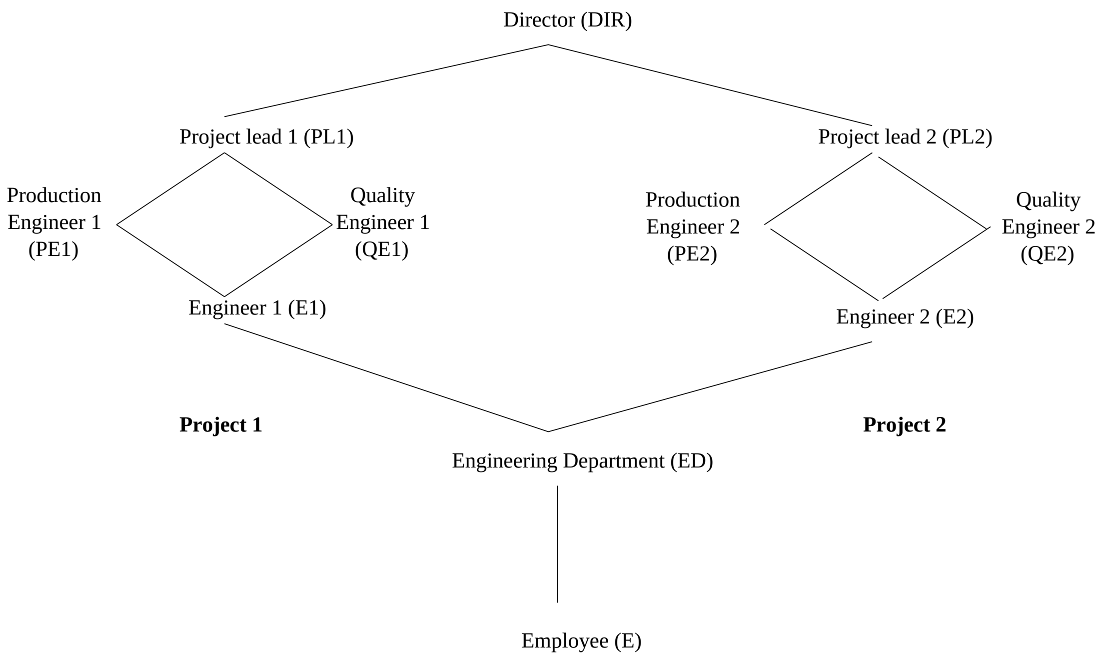
- $user : S \rightarrow U$, maps each session to a single user (which does not change)

  $roles : S \rightarrow 2^{R \cup AR}$ maps each session s_i to a set of roles and administrative roles $roles(s_i) \subseteq \{r \mid (\exists r' \geq r)[(user(s_i), r') \in UA \cup AUA]\}$ (which can change with time)

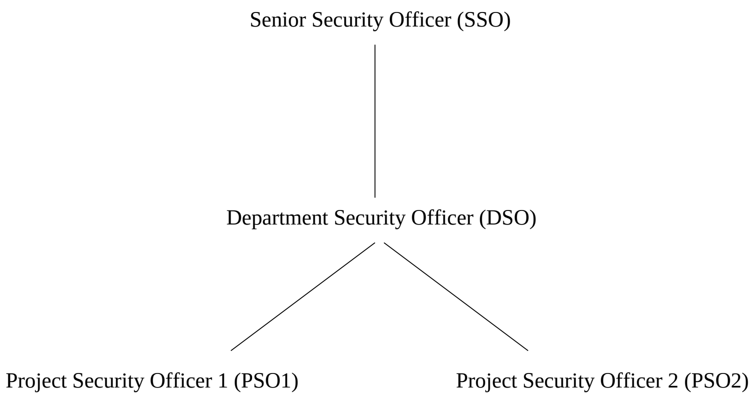
  session s_i has the permissions $\cup_{r \in roles(s_i)}\{p \mid (\exists r'' \leq r)[(p, r'') \in PA \cup APA]\}$

- there is a collection of constraints stipulating which values of the various components enumerated above are allowed or forbidden.

Figure 1: Summary of the RBAC96 Model



(a) Roles



(b) Administrative Roles

Figure 2: Example Role and Administrative Role Hierarchies

Administrative Role	Prerequisite Condition	Role Range
PSO1	ED	[E1, PL1)
PSO2	ED	[E2, PL2)
DSO	ED $\wedge$ $\overline{\text{PL1}}$	[PL2, PL2]
DSO	ED $\wedge$ $\overline{\text{PL2}}$	[PL1, PL1]

(a) can-assign

Administrative Role	Role Range
PSO1	[E1, PL1)
PSO2	[E2, PL2)
DSO	(ED, DIR)

(b) can-revoke

Table 1: Example of can-assign and can-revoke

the roles PE1, QE1 and E1, but these users must previously be members of the role ED. The idea is that PSO1 has freedom to assign users to roles in project 1 (excepting the senior-most role PL1) but these users must already be members of the engineering department. This is an example of a **prerequisite role**. More generally URA97 allows for a **prerequisite condition** as follows.

Definition 1 A prerequisite condition is a boolean expression using the usual $\wedge$ and $\vee$ operators on terms of the form x and $\overline{x}$ where x is a regular role (i.e., $x \in R$). For a given set of roles R let CR denotes all possible prerequisite conditions that can be formed using the roles in R. A prerequisite condition is evaluated for a user u by interpreting x to be true if $(\exists x' \geq x)(u, x') \in UA$ and $\overline{x}$ to be true if $(\forall x' \geq x)(u, x') \notin UA$. 2

Definition 2 User-role assignment and revocation are respectively authorized in URA97 by the following relations, $\textit{can-assign} \subseteq AR \times CR \times 2^R$ and $\textit{can-revoke} \subseteq AR \times 2^R$. 2

The meaning of $\textit{can-assign}(x, y, Z)$ is that a member of the administrative role x (or a member of an administrative role that is senior to x) can assign a user whose current membership, or non-membership, in regular roles satisfies the prerequisite condition y to be a member of regular roles in range Z.[2] The meaning of $\textit{can-revoke}(x, Y)$ is that a member of the administrative role x (or a member of an administrative role that is senior to x) can revoke membership of a user from any regular role $y \in Y$.

Table 1 illustrates these relations. Role sets are specified in URA97 by the following **range notation**.

$$
\begin{array}{rcl}
[x, y] & = & \{r \in R \mid x \geq r \wedge r \geq y\} \\
(x, y] & = & \{r \in R \mid x > r \wedge r \geq y\} \\
[x, y) & = & \{r \in R \mid x \geq r \wedge r > y\} \\
(x, y) & = & \{r \in R \mid x > r \wedge r > y\}
\end{array}
$$

By table 1(a) PSO1 can assign users in ED to the roles E1, PE1 and QE1, and similarly for PSO2 with respect to E2, PE2 and QE2. DSO can assign a user in ED to PL1 provided that user is not already in PL2, and similarly for PL2 with respect to PL1.

A notable aspect of revocation in URA97 is that revocation is independent of assignment. If Alice, by means of some administrative role, can revoke Bob's membership in a regular role the revocation takes effect independent of how Bob came to be a member of that regular role. This is consistent with RBAC philosophy where granting and revoking of membership is done for organizational reasons and not merely at the discretion of individual administrators.

Weak and Strong Revocation

The revocation operation in URA97 is said to be **weak** because it applies only to the role that is directly revoked. Suppose Bob is a member of PE1 and E1. If Alice revokes Bob's membership from E1, he continues to be a member of the senior role PE1 and therefore can use the permissions of E1.

Various forms of **strong** revocation can be considered as embellishments to URA97. Strong revocation

[2] User-role assignment is subject to additional constraints, such as mutually exclusive roles or maximum cardinality, that may be imposed. The assignment will succeed if and only if it is authorized by *can-assign* and it satisfies all relevant constraints.

Administrative Role	Prerequisite Condition	Role Range
DSO	DIR	[PL1, PL1]
DSO	DIR	[PL2, PL2]
PSO1	PL1 $\wedge$ $\overline{\text{QE1}}$	[PE1, PE1]
PSO1	PL1 $\wedge$ $\overline{\text{PE1}}$	[QE1, QE1]
PSO2	PL2 $\wedge$ $\overline{\text{QE2}}$	[PE2, PE2]
PSO2	PL2 $\wedge$ $\overline{\text{PE2}}$	[QE2, QE2]

(a) can-assignp

Administrative Role	Role Range
DSO	(ED, DIR)
PSO1	[QE1, QE1]
PSO1	[PE1, PE1]
PSO2	[QE2, QE2]
PSO2	[PE2, PE1]

(b) can-revokep

Table 2: Example of can-assignp and can-revokep

cascades upwards in the role hierarchy. If Alice has administrative role PSO1 and she strongly revokes Bob's membership from E1 as per table 1, his membership in PE1 is also revoked. However, if Charles is a member of E1 and PL1, and Alice strongly revokes Charles' membership in E1 the cascaded revoke is outside of Alice's range and is disallowed. The question remains whether or not Charles' membership in E1 and PE1 should be revoked even though the cascaded revoke from PL1 failed? It seems appropriate to allow both options depending upon Alice's choice.

In general URA97 treats strong revocation as a series of weak revocations each of which must be individually authorized by can-revoke. In this way we keep the basic URA97 model simple while allowing for more complex revocation operations to be defined in terms of weak revocation. At the same time we feel it is important to support strong revocation.

3 PRA97 FOR PERMISSION-ROLE ASSIGNMENT

PRA97 is concerned with role-permission assignment and revocation. From the perspective of a role, users and permissions have a similar character. They are essentially entities that are brought together by a role. Hence, we propose PRA97 to be a dual of URA97. The notion of a **prerequisite condition** is identical to that in URA97 except the boolean expression is now evaluated for membership and non-membership of a permission in specified roles.

Definition 3 Permission-role assignment and revocation are respectively authorized by the following relations, *can-assignp* $\subseteq AR \times CR \times 2^R$ and *can-revokep* $\subseteq AR \times 2^R$. □

The meaning of *can-assignp*(x, y, Z) is that a member of the administrative role x (or a member of an administrative role that is senior to x) can assign a permission whose current membership, or non-membership, in regular roles satisfies the prerequisite condition y to regular roles in range Z.[3] The meaning of *can-revokep*(x, Y) is that a member of the administrative role x (or a member of an administrative role that is senior to x) can revoke membership of a permission from any regular role $y \in Y$.

Table 2 shows examples of these relations. The DSO is authorized to take any permission assigned to the DIR role and make it available to PL1 or PL2. Thus a permission can be delegated downward in the hierarchy. PSO1 can assign permissions from PL1 either PE1 or QE1, but not to both. The remaining rows in table 2(a) are similarly interpreted.

Table 2(b) authorizes DSO to revoke permissions from any role between ED and DIR. PSO1 can revoke permissions from PE1 and QE2, and similarly for PSO2.

[3] Permission-role assignment may be subject to additional constraints.

Revocation in PRA97 is weak so permissions may still be inherited after revocation. Strong revocation can be defined in terms of weak revocation as in URA97. Strong revocation of a permissions cascades down the role hierarchy, in contrast to cascading up of revocation of user membership.

4 RRA97 FOR ROLE-ROLE ASSIGNMENT

In this section we consider the issue of role-role assignment. Our treatment is informal and preliminary at this point because the model is still evolving. Our focus is on the general direction and intuition.

For role-role assignment we distinguish three kinds of roles, roughly speaking as follows.

1. **Abilities** are roles that can only have permissions and other abilities as members.

2. **Groups** are roles that can only have users and other groups as members.

3. **UP-Roles** are roles that have no restriction on membership, i.e., their membership can include users, permissions, groups, abilities and other UP-roles.

The term UP-roles signifies user and permission roles. We use the term role to mean all three kinds of roles or to mean UP-roles only, as determined by context. The three kinds of roles are mutually disjoint and are identified respectively as A, G, and UPR.

The main reason to distinguish these three kinds of roles is that different administrative models apply to establishing relationships between them. The distinction was motivated in the first place by abilities. An ability is a collection of permissions that should be assigned as a single unit to a role. For example the ability to open an account in a banking application will encompass many different individual permissions. It does not make sense to assign only some of these permissions to a role because the entire set is needed to do the task properly. The idea is that application developers package permissions into collections called abilities which must be assigned together as a unit to a role. The function of an ability is to collect permissions together so that administrators can treat these as a single unit. Assigning abilities to roles is therefore very much like assigning permissions to roles. For convenience it is useful to organize abilities into a hierarchy (i.e., partial order). Hence the PRA97 model can be adapted to produce the very similar ARA97 model for ability-role assignment.

Once the notion of notion of abilities is introduced, by duality there should be a similar concept on the user side. A group is a collection of users who are assigned as a single unit to a role. Such a group can be viewed as a team which is a unit even though its membership may change over time. Groups can also be organized in a hierarchy. For group-role assignment we adapt the URA97 model to produce the GRA97 model for group-role assignment.

This leads to the following models.

Definition 4 Ability-role assignment and revocation are respectively authorized in ARA97 by *can-assigna* $\subseteq AR \times CR \times 2^A$ and *can-revokea* $\subseteq AR \times 2^A$. 2

Definition 5 Group-role assignment and revocation are respectively authorized in GRA97 by *can-assigng* $\subseteq AR \times CR \times 2^G$ and *can-revokeg* $\subseteq AR \times 2^G$. 2

For these models CR is interpreted as the collection of prerequisite conditions formed using roles in UPR, and the prerequisite conditions are interpreted with respect to abilities and groups respectively. Membership of an ability in a UP-role is true if the UP-role dominates the ability and false otherwise. Conversely, membership of a group in a UP-role is true if the UP-role is dominated by the group and false otherwise.

Assigning an ability to an UP-role is mathematically equivalent to making the UP-role an immediate senior of the ability in the role-role hierarchy. Abilities can only have UP-roles or abilities as immediate seniors and can only have abilities as immediate juniors. In a dual manner, assigning a group to an UP-role is mathematically equivalent to making the UP-role an immediate junior of the group in the role-role hierarchy. Groups can only have UP-roles or groups as immediate juniors and can only have groups as immediate seniors. With these constraints the ARA97 and GRA97 models are essentially identical to the PRA97 and URA97 models respectively.

This leaves us with the problem of managing relationships between UP-roles.[4] Consider figure 2(a) again. We would like the DSO to configure and change the hierarchy between DIR and ED. Similarly, we would like PSO1 to manage the hierarchy between PL1 and E1, and likewise for PSO2 with respect to PL2 and E2. The idea is that each department and each project has autonomy in constructing its internal role structure.

Definition 6 Role-role creation, deletion, edge insertion, edge deletion are all authorized in UP-RRA97 by *can-modify* : $AR \rightarrow 2^{UPR}$. 2

[4]Strictly speaking we also have to deal with administration of group-group and ability-ability relationships. These can be handled in the same way as relationships between UP-roles to analogously give us the G-RRA97 and A-RRA97 models.

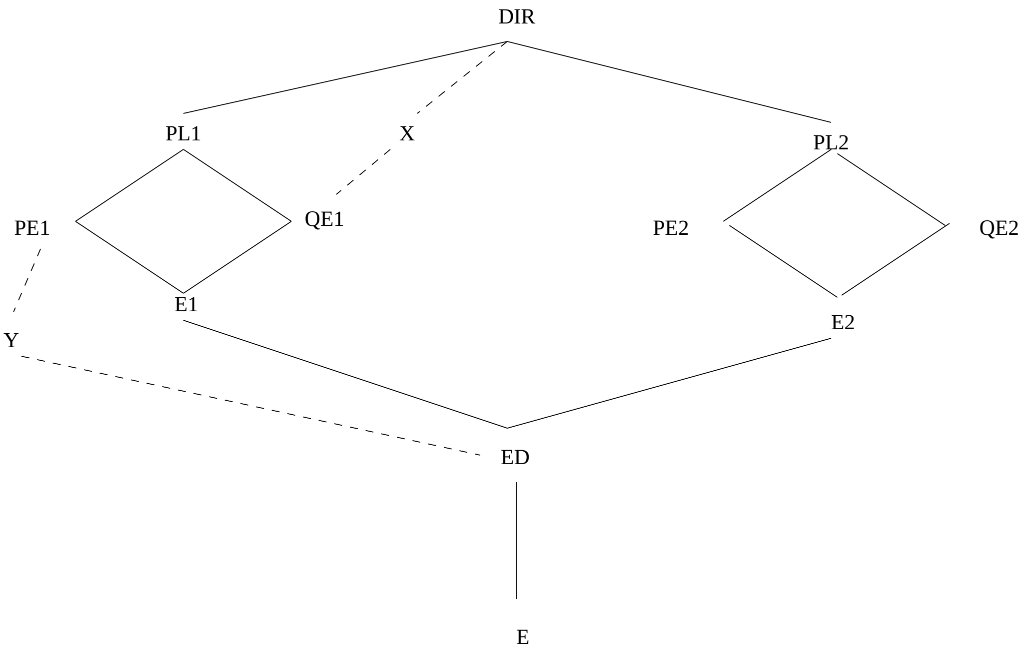


Figure 3: Out of Range Impact

Administrative Role	UP-Role Range
PSO1	(E1, PL1)
DSO	(ED, DIR)

Table 3: Example of can-modify

The meaning of *can-modify*(x, Y) is that a member of the administrative role x (or a member of an administrative role that is senior to x) can create and delete roles in the range Y, except for the endpoints of Y, and can modify relationships between roles in the range Y. This authority is, however, tempered by constraints that we discuss below.

Table 3 illustrates an example of can-modify relative to the hierarchies of figure 2. By convention the UP-role ranges are shown as open intervals since the endpoints are not included. DSO can create, delete and alter relationships between all roles in the engineering department (except the endpoints ED and DIR). When a DSO creates a new role it will be senior to ED and junior to DIR, and will remain unless some more senior administrator changes this relationship. PSO1 has similar authority with respect to roles in project 1.

Restrictions on can-modify

The authority conferred by can-modify is constrained by global consistency requirements. It is not possible to change pieces of the role hierarchy in arbitrary ways without impact larger relationships. Here we identify two conflicts that arise and explain how RRA97 deals with them.

Suppose DSO is given the authority to create and delete edges and roles in the hierarchy between DIR and ED. If PL1 gets deleted tables 1 and 2 will be left with dangling references to a non-existent role. To avoid this situation we require that roles that are referenced in any can-assign or can-revoke relation cannot be deleted. In this way the DSO's power to delete roles is restricted to maintain global consistency of the authorizations.

The second problem arises if the DSO introduces roles X and Y as shown in figure 3. Now suppose PSO1 has authority to create and delete edges and roles in the hierarchy between PL1 and E1. If PSO1 makes PE1 junior to QE1 the effect is to indirectly make Y junior to X. Now PSO1 was given authority in the range (PL1,E1) but has effectively introduced a relationship between X and Y. There are several approaches to resolving this issue. We can prevent the DSO from introducing X and Y as shown, because this violates the range integrity of (PL1,E1) with respect to PSO1. We can allow figure 3 to happen and prevent PSO1 from later making PE1 junior to QE1. Or we can tolerate the possibility of PSO1 affecting UP-role to UP-role relationships that are outside the authorized range of (E1, PL1). RRA97 allows all three possibilities.

There may be other issues that will arise as we evolve this model. Our principle for decentralized administration of role-role relationships is a sound one. We wish to give administrative roles autonomy within a range but only so far as the global consequences of the resulting actions are acceptable. To do so we need to disallow some operations authorized by the range, thereby tempering the administrative role's authority.

5 CONCLUSION

In this paper we have described the motivation, intuition and outline of a new model for RBAC administration called ARBAC97 (administrative RBAC '97). ARBAC97 has three components: URA97 (user-role assignment '97), PRA97 (permission-role assignment '97) and RRA97 (role-role assignment '97). URA97 was recently defined by Sandhu and Bhamidipati [SB97]. ARBAC97 incorporates URA97, builds upon it to define PRA97 and some components of RRA97, and introduces additional concepts in developing RRA97.

RRA97 itself consists of three components. ARA97 and GRA97 deal with ability-role assignment and group-role assignment respectively, and are very similar to PRA97 and URA97 respectively. The component dealing with role-role assignment is still evolving but we have identified the basic intuition and two important issues that need to be dealt with.

Acknowledgment

We thank John Barkley, Dave Ferraiolo and Serban Gavrila for useful discussions and comments.

References

[FCK95] David Ferraiolo, Janet Cugini, and Richard Kuhn. Role-based access control (RBAC): Features and motivations. In *Proceedings of 11th Annual Computer Security Application Conference*, pages 241–48, New Orleans, LA, December 11-15 1995.

[FK92] David Ferraiolo and Richard Kuhn. Role-based access controls. In *Proceedings of 15th NIST-NCSC National Computer Security Conference*, pages 554–563, Baltimore, MD, October 13-16 1992.

[GI96] Luigi Guiri and Pietro Iglio. A formal model for role-based access control with constraints. In *Proceedings of IEEE Computer Security Foundations Workshop 9*, pages 136–145, Kenmare, Ireland, June 1996.

[Gui95] Luigi Guiri. A new model for role-based access control. In *Proceedings of 11th Annual Computer Security Application Conference*, pages 249–255, New Orleans, LA, December 11-15 1995.

[HDT95] M.-Y. Hu, S.A. Demurjian, and T.C. Ting. User-role based security in the ADAM object-oriented design and analyses environment. In J. Biskup, M. Morgernstern, and C. Landwehr, editors, *Database Security VIII: Status and Prospects*. North-Holland, 1995.

[MD94] Imtiaz Mohammed and David M. Dilts. Design for dynamic user-role-based security. *Computers & Security*, 13(8):661–671, 1994.

[NO95] Matunda Nyanchama and Sylvia Osborn. Access rights administration in role-based security systems. In J. Biskup, M. Morgernstern, and C. Landwehr, editors, *Database Security VIII: Status and Prospects*. North-Holland, 1995.

[San97] Ravi Sandhu. Rationale for the RBAC96 family of access control models. In *Proceedings of the 1st ACM Workshop on Role-Based Access Control*. ACM, 1997.

[SB97] Ravi Sandhu and Venkata Bhamidipati. Role-based administration of user-role assignment: The URA97 model and its Oracle implementation. In T. Y. Lin and Xiaolei Qian, editors, *Database Security XI: Status and Prospects*. North-Holland, to appear.

[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.

[vSvdM94] S. H. von Solms and Isak van der Merwe. The management of computer security profiles using a role-oriented approach. *Computers & Security*, 13(8):673–680, 1994.

[YCS97] Charles Youman, Ed Coyne, and Ravi Sandhu, editors. *Proceedings of the 1st ACM Workshop on Role-Based Access Control, Nov 31-Dec. 1, 1995*. ACM, 1997.