The RRA97 Model for Role-Based Administration of Role Hierarchies

Ravi Sandhu and Qamar Munawer
Laboratory for Information Security Technology and ISE Department
George Mason University—Mail Stop 4A4, Fairfax, VA 22030
sandhu@isse.gmu.edu, www.list.gmu.edu

Abstract *Role-based access control (RBAC) has recently received a lot of attention due to its flexibility, expressive power and simplicity in administration. In RBAC permissions are associated with roles and users are made members of roles thereby acquiring the associated permissions. Centralized management of RBAC in large systems is a tedious and costly task. An appealing possibility is to use RBAC itself to facilitate decentralized administration of RBAC. The recently proposed ARBAC97 (administrative RBAC '97) model identifies components called URA97, PRA97 and RRA97 for administration of user-role, permission-role and role-role assignments respectively. URA97 and PRA97 have already been described in detail in the literature, whereas RRA97 has so far not been defined.*

The central contribution of this paper is to give a complete and formal definition of RRA97, thereby completing the ARBAC97 model. The effect of role-role assignment is to construct a role hierarchy (that is, a partial order) in which senior roles inherit permissions from junior roles. Modifications to the role hierarchy can have drastic impact on the effective distribution of permissions to roles. At the same time we would like to decentralize this aspect of RBAC administration so that, for example, it should be possible for project security officers to rearrange roles within a project without impacting other role relationships within the department in which the project exists. RRA97 shows how this goal can be achieved.

1 Introduction

Role-based access control (RBAC) has recently received considerable attention as a promising alternative to traditional discretionary and mandatory access controls. 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 losing 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 includes aspects of user-role assignment and role-permission assignment, but more significantly role-role

relationships establish broad policy. Control of these relationships would typically be relatively centralized in the hands of a few security administrators.

Sandhu et al [SBC+97] recently introduced a model for role-based administration of RBAC. This model is called ARBAC97 (administrative RBAC '97). It consists of three components called URA97, PRA97 and RRA97 for administration of user-role, permission-role and role-role assignments respectively. URA97 and PRA97 have already been described in detail in the literature [SB97, SBC+97, SB98], whereas RRA97 has so far not been formally defined. Sandhu et al [SBC+97] describe some of the requirements and intuitive goals of RRA97, but leave many issues open. Modifications to the role hierarchy can clearly have drastic impact on the effective distribution of permissions to roles. The intuitive idea of RRA97 is to decentralize role-role assignment so that, for example, project security officers can rearrange roles within a project without impacting other role relationships within the department in which the project exists. A fundamental assumption in RRA97 is that there is a single global hierarchy of roles which is known to all administrators.

The central contribution of this paper is to give a complete and formal definition of RRA97, thereby completing the ARBAC97 model. The effect of role-role assignment is to construct a role hierarchy (that is, a partial order) in which senior roles inherit permissions from junior roles. RRA97 shows how this goal can be achieved. It is the first attempt in the literature to give a comprehensive model for decentralized management of role hierarchies.

The rest of this paper is organized as follows. In section 2 we review the ARBAC97 model and, particularly, its URA97 and PRA97 components. We also briefly discuss the RBAC96 models on which ARBAC97 is based. Section 3 gives a formal definition of the RRA97 model and rationale for the design choices made here. Section 4 concludes the paper.

2 RBAC96 and ARBAC97 Models

This section gives a brief review of the RBAC96 and ARBAC97 models.

2.1 The RBAC96 Model

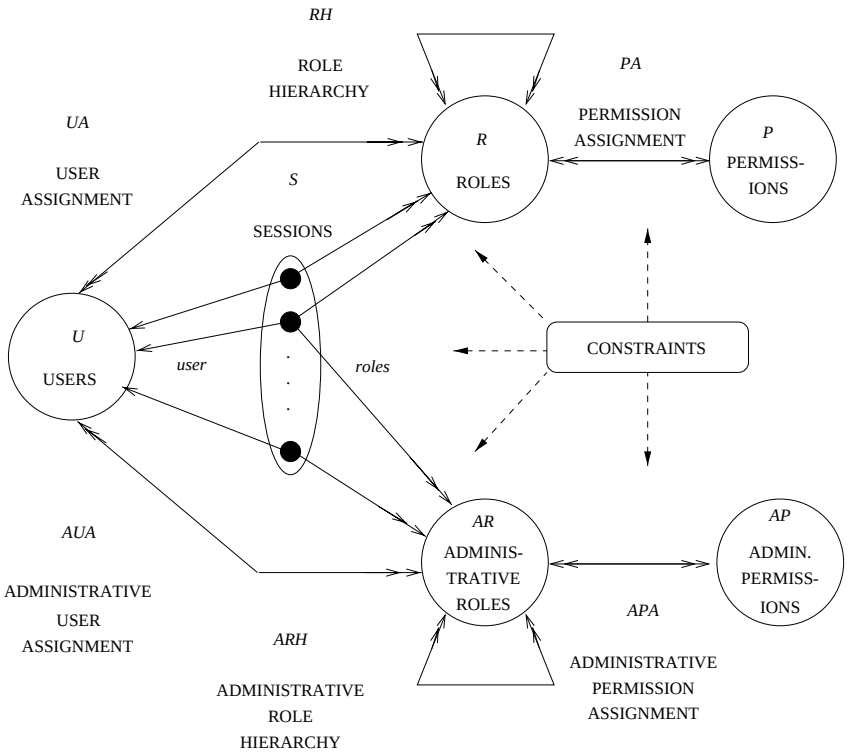
The well-known RBAC96 model [SCFY96, San97] is summarized in figure 1. The top half of the figure shows the regular roles and permissions that regulate access to data and resources. The bottom half shows the administrative roles and permissions to regulate the administration of users, roles and permissions. A user is a human being, a role is a job function and a permission is an approval of access to some objects or a privilege to carry out a particular task. Sessions are related to one user and possibly to many roles for which the user is a member. The management of permissions and roles is greatly simplified by associating permissions to the roles and assigning the users to roles. In this way the users acquire the associated permissions. Roles are created for various job functions in an organization. The permissions required to carry out the job are associated to the roles. New permissions can be granted to roles as new applications and systems are incorporated. Unnecessary permissions can be revoked from the roles. Users are assigned to the roles depending on the responsibilities and qualifications and can be reassigned from one role to another. Role-role relationships can be established to lay out broad policy objectives of the organizations.

2.2 The ARBAC97 Model

Sandhu et al [SBC+97] have outlined a comprehensive model for administration of RBAC in context of RBAC96. The model is called ARBAC97 and consists of three components which we describe briefly below. As stated earlier the RRA97 component of ARBAC97 is defined for the first time in this paper in the next section. A complete definition of the URA97 and PRA97 components, and the intuitive motivation for these models,is given in [SB97, SB98, SBC+97].

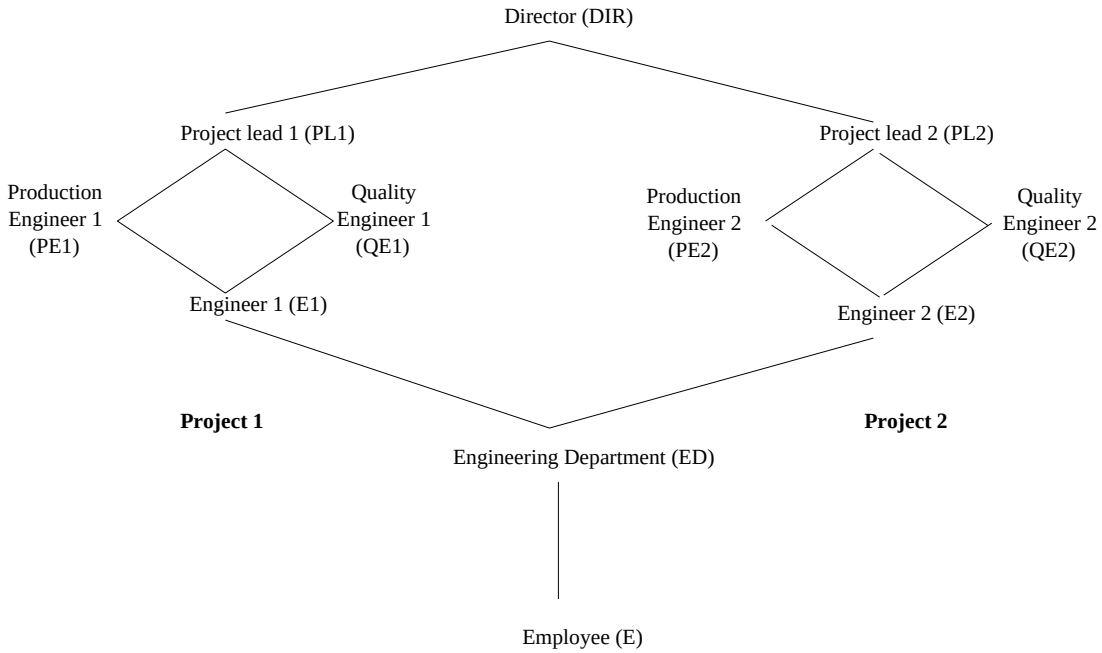
2.2.1 URA97 for user-role assignment

URA97 was developed by Sandhu and Bhamidipati [SB97]. It is concerned with the administration of user-role assignments. The basic idea is that administrative roles have authority to modify the UA relation of RBAC96. For example the administrative roles in figure 2(b) are authorized to modify the memberships of the roles in figure 2(a). In these diagrams senior roles are shown at the top and junior ones at the bottom. Permissions are inherited upwards in the hierarchy. The power of a administrative role extends over some range of regular roles—identified by giving the top and bottom of the range, and indicating whether the top and bottom are themselves included in the range. Familiar interval notation is used for this purpose. Thus [E1, PL1]={E1, PE1, QE1, PL1} and [E1, PL1)={E1, PE1, QE1}. A novel aspect of URA97

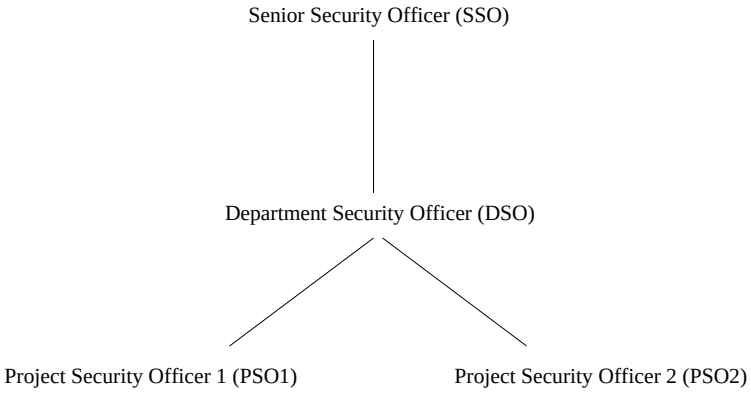


- U, a set of users
  R and AR, disjoint sets of (regular) roles and administrative roles
  P and AP, disjoint sets of (regular) permissions and administrative permissions
  S, a set of sessions
- $UA \subseteq U \times R$, user to role assignment relation
  $AUA \subseteq U \times AR$, user to administrative 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

is that the user population that can be assigned by an administrative role to such a range is specified by means of prerequisite conditions 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 $(\forall x' \geq x)(u, x') \notin UA$. □

The simplest example of a prerequisite condition is simply a prerequisite, for example, the prerequisite condition ED requires membership in ED. The more complex condition ED$\wedge\overline{\text{E2}}$ specifies users from the engineering department who are not assigned to project 2.

Definition 2 User-role assignment and revocation are respectively authorized in URA97 by the following relations, *can-assign* $\subseteq AR \times CR \times 2^R$ and *can-revoke* $\subseteq AR \times 2^R$ (with subsets of R identified by the range notation). □

For example *can-assign*(PSO1, ED, [E1,PL1)) authorizes members of the administrative role PSO1 to assign users who are already members of ED to the roles E1, PE1, QE1. Similarly, *can-revoke*(PSO1, [E1,PL1)) authorizes members of the administrative role PSO1 to revoke users from the roles E1, PE1, QE1. The exact semantics of revocation in this context is somewhat subtle and is discussed at length in [SB97, SB98, SBC+97].

2.2.2 PRA97 for permission-role assignments

This component of ARBAC97 deals with the assignments of permissions to roles. From a role-based , users and permissions have a similar character so PRA97 is an exact analog 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. Analogous to URA97 we have the following two relations to control permission-role assignment and revocation.

Definition 3 Permission-role assignment and revocation are respectively authorized in PRA97 by the following relations, *can-assignp* $\subseteq AR \times CR \times 2^R$ and *can-revokep* $\subseteq AR \times 2^R$. □

3 RRA97 Model for Role-Role Assignment

In this section we develop the RRA97 model, which is the central contribution of this paper.

3.1 Abilities, Groups and UP-Roles

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

- **Abilities** are roles that can only have permissions and other abilities as members.
- **Groups** are roles that can only have users and other groups as members.
- **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 abilities is introduced, by analogy 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* $\subseteq AR \times 2^A$. □

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* $\subseteq AR \times 2^G$. □

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. We use the term role to mean UP-roles in the rest of the paper.

3.2 The can-modify relation

Decentralization of administrative authority requires that members of different administrative roles should have authority over different parts of the hierarchy. Authority over a part of the role hierarchy implies autonomy in modifying the internal role structure of that part. That includes the creation and deletion of roles as well as the alternation of role-role relationships by adding or deleting the edges. For example in figure 2 we would like the DSO to configure changes in the role hierarchy between DIR and ED. The PSO1 would manage the hierarchy between PL1 and E1, whereas PSO2 would manage the part between PL2 and E2. This leads to the following notion of a can-modify relation.

Administrative Role	UP-Role Range
DSO	(ED, DIR)
PSO1	(E1, PL1)
PSO1	(E2, PL2)

Table 1: Example of can-modify

Definition 6 Role creation, role deletion, edge insertion and edge deletion are all authorized by the relation, *can-modify* $\subseteq AR \times 2^{UPR}$ (with subsets of R identified by the range notation but limited to open ranges that do not include the endpoints). □

Table 1 illustrates an example of can-modify relative to the hierarchies of figure 2. 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 and can modify relationships between roles in Y. The examples in the rest of the paper are all in context of figure 2 and table 1. For purpose of our example we have ignored PSO2 in this table and have instead authorized PSO1 to manage the roles of both projects. This illustrates how a single administrative role can be authorized to control multiple pieces of the role hierarchy.

The semantics of the four operations—create role, delete role, insert edge and delete edge—are described in subsequent subsections. Some of the important intuitive ideas are mentioned here in anticipation. In particular none of these operations is allowed to introduce a cycle in the hierarchy.

Creation of a new role requires the specification of its immediate parent and child in the existing hierarchy. Thus PSO1 can create a new role with immediate parent PL1 and immediate child E1, or a new role with immediate parent PL1 and immediate child PE1. Generally the immediate parent and immediate child must fall within the range or be one of the endpoints as specified in can-modify. Since creation of a role also introduces two edges in the hierarchy, it is not possible to use any two roles as the immediate parent and

immediate child. Clearly we do not want this operation to introduce a cycle in this manner. As we will see we also impose additional restrictions to prevent undesirable side effects of role creation.

Deletion of a role leaves relationships between the parents and children of the deleted role unchanged. So if DSO deletes E1, PE1 and QE1 continue to be senior to ED after deletion of E1. As such deletion does not pose a problem. However, deletion of E1 will leave dangling references in table 1, since the range (E1, Pl1) no longer exists. In general, some roles are referenced in various relations in URA97, PRA97 and RRA97. If these roles are actually deleted we will have dangling references. Our approach is to prohibit deletion that would cause a dangling reference. Roles that cannot be deleted due to this reason can be deactivated so that they can be phased out later by adjusting the references that prevent deletion. Furthermore, when a role is deleted we need to do something about the users and permissions that are directly assigned to this role.

Insertion of an edge is meaningful only between incomparable nodes. Thus insertion of an edge from PL1 to E1 has no meaning, whereas insertion of an edge from PE1 to QE1 does. As well see there are edges that should not be inserted because they can lead to anomalous side effects later.

Likewise deletion of an edge is meaningful only if that edge is not transitively implied by other edges. For example, deletion of the edge PL1 to E1 is meaningless and has no impact on the hierarchy. Deletion of the edge QE1 to E1 will change the hierarchy. Edge deletion only applies to a single edge and does not carry over to implied transitive edges. For example, deletion of the edge QE1 to E1 makes QE1 and E1 incomparable, but QE1 continues to be senior to ED.

More sophisticated forms of these operations can be constructed out of the basic ones defined here. In these basic operations roles and edges are created and destroyed one at a time. This approach is analogous to the definition of weak revocation in URA97 and PRA97 [SBC+97] from which various forms of strong revocation can be constructed. Similarly, in RRA97 more complex operations can be constructed in terms of these basic ones.

3.3 Restrictions on can-modify

The relation can-modify confers authority to administrative roles to change the role hierarchy. We would like to restrict this authority so as to maintain global consistency of authorization. The issue of dangling references has already been raised and RRA97 will not allow dangling references to occur. But this is not enough.

Consider the example of figure 3. Now if PSO1 who has authority over the range (E1, PL1) makes PE1 junior to QE1 by introducing an edge the effect is to indirectly introduce a relationship between X and Y roles. The role PSO1 does not have the authority to create this relationship, so this is an anomalous side effect. We should either restrict the authority of the administrative role (in our example DSO) that introduced X and Y roles in the first place, or PSO1 should be prevented from introducing relationships that makes PE1 junior to QE1 (and indirectly Y junior to X). In general administrative roles are given autonomy within a range but only so far as the global side effects are acceptable.

To formally state these restrictions on the authority of the administrative roles we introduce the concepts of authority range, encapsulated authority range and create range.

3.4 Concept of Range

The concept of range is very important in RRA97. It is formally defined as follows.[1]

Definition 7 A range of roles is defined by giving lower bound x and upper bound y, where y > x. Formally $(x, y) = \{z : R \mid x < z < y\}$. We say x and y are the end points of the range. □

Note that a range, as defined here, does not include the end points. In figure 2, (E1, PL1), (E2, PL2) and (ED, DIR) are different ranges. The range (ED, DIR) contains the roles which constitute ranges (E1, PL1) and (E2, PL2). We say ranges (E1, PL1) and (E2, PL2) are junior to range (ED, DIR).

Definition 8 For two ranges Y and Y' if $Y \subset Y'$, then Y is a junior range to Y' and Y' is a senior range to Y. □

Here Y is a proper subset of Y'. This eliminates the possibility of a range to be junior or senior to itself. This makes later definitions more convenient.

[1] We understand there is existing mathematical terminology for this concept and others to be introduced below. We have chosen to develop our own terminology motivated by administrative RBAC.

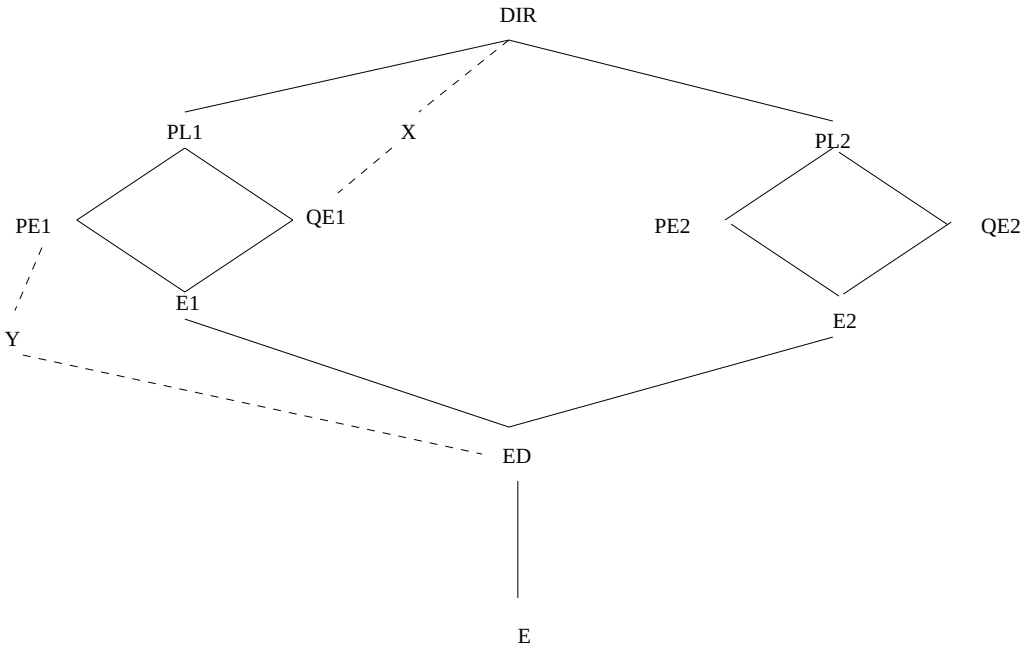


Figure 3: Out of Range Impact

If two ranges Y and Y' in the role hierarchy are such that one is not junior to the other then they are either incomparable or partially overlapping. Formal definitions of partially overlapping and incomparable ranges are as follows.

Definition 9 Ranges Y and Y' partially overlap if $Y \cap Y' \neq \phi$ and $Y \not\subset Y'$ and $Y' \not\subset Y$. Ranges Y_1 and Y_2 are said to be incomparable if $Y_1 \cap Y_2 = \phi$. □

Note that incomparable ranges may have one common end point.

3.5 Authority Range and Encapsulated Authority Range

The members of an administrative roles are authorized to modify certain range of roles in role hierarchy. These ranges are called authority ranges.

Definition 10 Any range referenced in the can-modify relation is called an authority range. □

To ensure that administrative authority over authority ranges does not overlap, we introduce the following restriction.

Definition 11 In RRA97 authority ranges do not partially overlap. □

Note that an administrative role may have more than one authority range. Table 1 shows that DSO has authority over the range (ED, DIR). In figure 2 the authority range (ED, DIR) has two junior authority ranges, (E1, PL1) and (E2, PL2). Since these junior authority ranges are completely contained within the authority range for DSO, DSO has authority over these junior authority ranges as well. In other words DSO has inherited the authority over the ranges (E1, PL1) and (E2, PL2).

Our model allows an administrative role to have authority over more than one incomparable authority range. Table 1 shows that PSO1 has authority over two incomparable authority ranges namely (E1, PL1) and (E2, PL2).

Let us consider figure 3 again. To maintain consistency we observed that either DSO should not be allowed to create roles X or Y in the role hierarchy or PSO1 should not be allowed to make PE1 junior to QE1. In the latter case the autonomy of PSO1 to manage its authority range is interfered by DSO's actions. While this is a possibility we pursue the former case here. Decentralization of authority and autonomy requires that all inward and outgoing edges from an authority range should only be directed to and from the end points of the authority range. The concept of

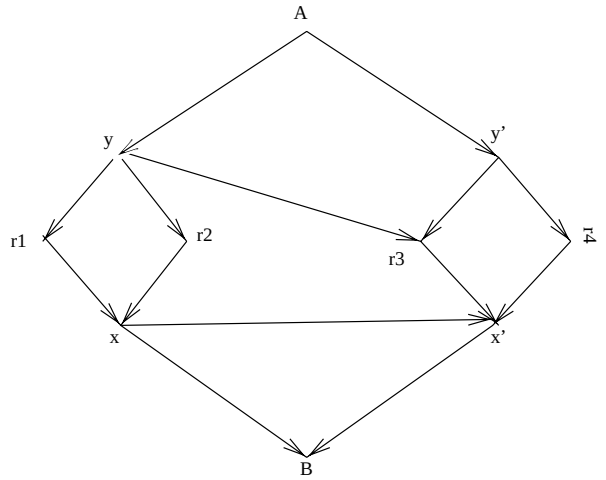


Figure 4: Encapsulated Range (x, y)

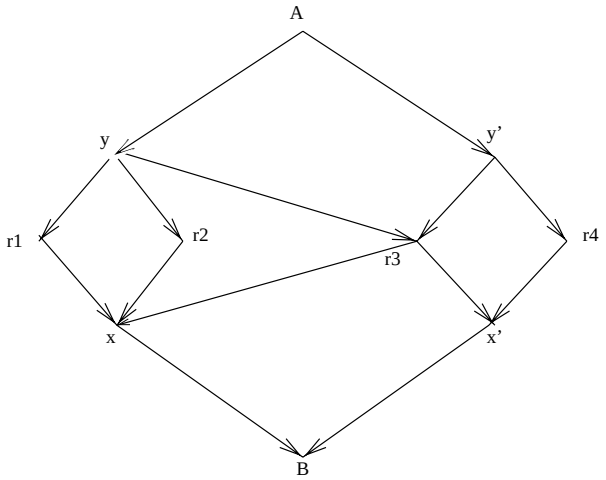


Figure 5: Non-Encapsulated Range (x, y)

encapsulation of authority range serves this purpose.

Definition 12 A range (x, y) is said to be encapsulated if $\forall r1 \in (x, y) \wedge \forall r2 \notin (x, y)$ we have $r2 > r1 \Leftrightarrow r2 > y$ and $r2 < r1 \Leftrightarrow r2 < x$. □

Intuitively an encapsulated range is one in which all roles have identical relation to roles outside of the range. The intuition in RRA97 is that an encapsulated range is the correct unit for autonomous management of role-role relationships within the range. All authority ranges in RRA97 are required to be encapsulated. Figure 4 and 5 respectively show examples of encapsulated and non-encapsulated range (x, y).

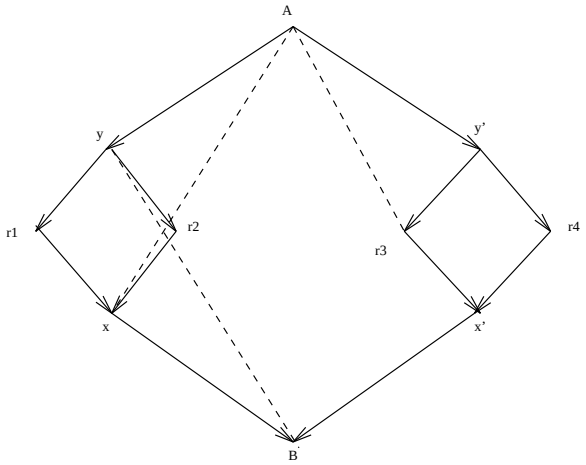


Figure 6: Create Range

3.6 Role Creation

As discussed earlier creation of a role requires specification of the new role's immediate parent and child. If the immediate parent and child are the end points of an authority range, there is no difficulty. More generally we wish to allow creation of a new role such that its immediate parent and child are within the authority range rather than being at the end points. Thus PSO1 can create a new role with parent PL1 and child PE1. However if DSO exercises this power we can end up with up the undesirable situation illustrated in figure 3. To prevent this from happening we introduce the following notions.

Definition 13 The immediate authority range of role r written $AR_{immediate}(r)$ is the authority range (x, y) such that $r \in (x, y)$ and for all authority ranges (x', y') junior to (x, y) we have $r \notin (x', y')$. □

Definition 14 The range (x, y) is a create range if $AR_{immediate}(x) = AR_{immediate}(y)$ or x is an end point of $AR_{immediate}(y)$ or y is an end point of $AR_{immediate}(x)$. □

Note that only comparable roles constitute a create-range.

Consider figure 6. Let (B, A) and (x, y) be authority ranges whereas (x', y') is not an authority range. The ranges marked by the dotted lines, i.e., (r3, A), (x, A) and (B, y) are create ranges. However (r1, A) or (r2, A) do not satisfy the conditions and thereby are not create ranges.

In RRA97 we require that the immediate parent and child of a new role must be a create range in the hierarchy prior to creation of the new role.

Roles can be created outside the authority ranges or without a parent or child only by the chief security officer. In general the chief security officer can do arbitrary modifications.

3.7 Role Deletion

Deletion of roles in a hierarchy is a complicated process. Our assumption is that a role in an authority range can be deleted by the administrator of that range. It does not matter how this role got there.

ARBAC97 defines some authorization relations such as can-assign, can-revoke and can-modify. If the roles specified as end points of the role ranges of these relationships are deleted, we will leave dangling references to non-existing roles. The ranges with these deleted end points will become meaningless. To avoid this problem RRA97 provides two alternatives.

1. Roles referred in can-assign, can-revoke and can-modify relationships cannot be deleted. Though it is a more restrictive constraint but it is required to keep the range referential integrity intact.
2. Roles referred in 1 above can be made inactive (explained in the next paragraph) whenever it is needed to delete them. The advantage of deactivating roles is that it avoids references to non-existing roles and at the same time achieves the purpose of deletion.

A role is said to be inactive if a user associated to it cannot activate it in a session. The edges to and from the inactive role, its associated permissions and assigned users remain unchanged. While a user assigned to an inactive role cannot activate it, the permissions associated with that role are still inherited by senior roles. In this way the hierarchy is not changed but at the same time a partial effect of deletion is achieved.

RRA97 allows both of the above alternatives. Regular users cannot invoke inactive roles, but administrators can revoke users and permissions from these roles. These roles can be made empty but cannot be deleted from the hierarchy until the references preventing deletion are suitably adjusted. Other roles in the role hierarchy can be deleted.

In case of deletion of a role we need to preserve the permissions and users assigned to the role. RRA97 provides two alternatives for deletion of roles.

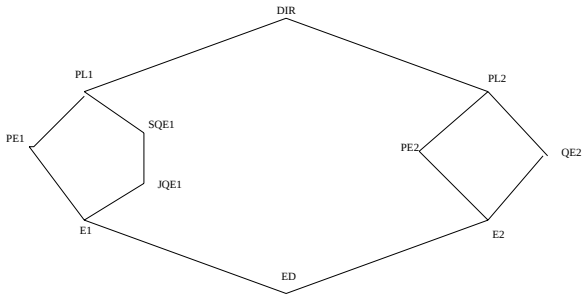


Figure 7: Before Deletion of edge from SQE1 to JQE1

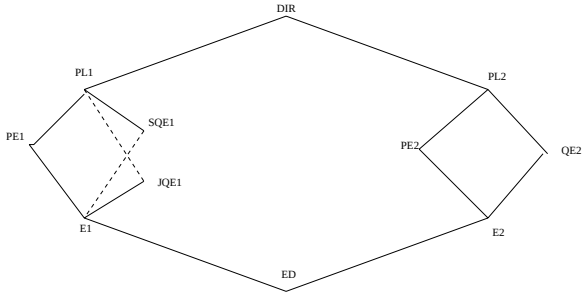


Figure 8: After Edge Deletion

1. Roles can be deleted only if they are empty.
2. Delete role but at the same time take care of the assigned permissions and associated users as follows: assign permissions to the immediate senior roles and assign users to immediate junior roles.

3.8 Edge Insertion

Now let us explain how the model deals with the insertion of edges in the role to role relationships. The insertion of transitive edges has no effect so we only consider edges inserted between incomparable roles. When an edge is inserted we must ensure that encapsulation of authority range is not violated. We have the following rules.

- The roles between which the edge is inserted must have same immediate authority range, or
- if the new edge connects a role in one authority range to a role outside the range encapsulation of the authority range must not be violated.

For example in figure 5 assume edges (y, r3) and (r3, x) are initially not present, and that (x, y) and

(B, A) are authority ranges. Insertion of the edge (y, r3) does not pose any problem. However in presence of this edge, insertion of edge (r3, x) violates encapsulation of authority range (x, y), hence it must not be allowed. Similarly in the presence of (r3, x) the edge (y, r3) would not be allowed. This leads to the following formal definition for insertion of an edge.

Definition 15 A new edge AB can be inserted between incomparable roles A and B

- if $AR_{immediate}(A) = AR_{immediate}(B)$ or
- if (x, y) is an authority range such that $(A = y \wedge B > x) \vee (B = x \wedge A < y)$ then insertion of AB must preserve encapsulation of (x, y). □

3.9 Edge Deletion

The deletion of a transitive edge does not change the hierarchy, so their deletion is meaningless. In RRA97 we consider only those edges for deletion that are in transitive reduction of the hierarchy. If edge AB is not in the transitive reduction then it is not a candidate for deletion.[2] For example in figure 7 deletion of the edge SQE1 to JQE1 will change the hierarchy. Edge deletion only applies to a single edge and does not carry over to implied transitive edges. As discussed in the general rules for edge deletion RRA97 keeps intact transitive edges after deletion. For example, deletion of the edge SQE1 to JQE1 makes SQE1 and JQE1 incomparable, but SQE1 continues to be senior to E1 and JQE1 junior to PL1 shown in figure 8.

There is one special case that needs to be considered. If the edge being deleted is between the end points of an authority range, deletion of the edge will disrupt the authority range and cause inconsistency in the model. Hence this operation is disallowed.

4 Conclusion

This paper defines the RRA97 model thus completing the definition of ARBAC97 started in [SBC+97]. RRA97 is very different from the URA97 and PRA97 components of ARBAC97 which have been previously reported in the literature. RRA97 provides for decentralized administration of role hierarchies. This desire is to give administrative roles autonomy within a range but only so far as the side effects of the resulting actions are acceptable. To do so we need to disallow some operations authorized by the authority range, thereby tempering the administrative role's authority. We have formally identified these restrictions in the paper and have provided their motivations. RRA97 is the first model to deal with these issues.

[2] Other models and applications do not have this restriction. For example, Oracle allows insertion and deletion of transitive edges [KL95].

5 Acknowledgment

This work is partially supported by grant CCR-9503560 from the National Science Foundation at the Laboratory for Information Security Technology at George Mason University.

References

[KL95] George Koch and Kevin Loney. *Oracle The Complete Reference*. Oracle Press, 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. The URA97 model for role-based administration of user-role assignment. In T. Y. Lin and Xiaolei Qian, editors, *Database Security XI: Status and Prospects*. North-Holland, 1997.

[SB98] Ravi Sandhu and Venkata Bhamidipati. Role-based administration of user-role assignment: The URA97 model and its Oracle implementation. *The Journal Of Computer Security*, 1998. in press.

[SBC+97] Ravi Sandhu, Venkata Bhamidipati, Edward Coyne, Srinivas Ganta, and Charles Youman. The ARBAC97 model for role-based administration of roles: Preliminary description and outline. In *Proceedings of 2nd ACM Workshop on Role-Based Access Control*, Fairfax, VA, November 6-7 1997. ACM.

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