A Model for Role Administration Using Organization Structure

Sejong Oh
Lab. for Information Security Technology
George Mason University
4400 University Drive,
Fairfax, Virginia 22030
soh2@gmu.edu

Ravi Sandhu
SingleSignOn.Net and
George Mason University
4400 University Drive,
Fairfax, Virginia 22030
sandhu@gmu.edu, www.list.gmu.edu

ABSTRACT

Role-based access control (RBAC) is recognized as an excellent model for access control in an enterprise environment. In large enterprises, effective RBAC administration is a major issue. ARBAC97 is a well-known solution for decentralized RBAC administration. ARBAC97 authorizes administrative roles by means of 'role ranges' and 'prerequisite conditions'. Although attractive and elegant in their own right, we will see that these mechanisms have significant shortcomings.

We propose an improved role administration model named ARBAC02 to overcome the weaknesses of ARBAC97. ARBAC02 adopts the organization unit for new user and permission pools independent of role or role hierarchy. It uses a refined prerequisite condition. In addition, we present a bottom-up approach to permission-role administration in contrast to the top-down approach of ARBAC97.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection – *access controls*

General Terms

Security

Keywords

Access control, RBAC, Role administration

1. INTRODUCTION

Access control is a central concern for information security in enterprises. Role-based access control (RBAC) is a proven and increasingly commonplace technology for this purpose. In RBAC, access rights are associated with roles, and users are assigned appropriate roles thereby acquiring the corresponding permissions. The notion of role is an enterprise or organizational concept. Therefore RBAC allows us to model security from the perspective of the enterprise, because we can align security modeling to the roles and responsibilities in the company. The RBAC model has been shown to be "policy-neutral" in the sense that using hierarchies and constraints, a wide range of security policies can be expressed, including discretionary access control (DAC), mandatory access control (MAC), and user-specific access control [6, 10].

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 or thousands. Managing these roles, users, and their interrelationships is a formidable task that is often highly centralized in a small team of security administrators. The motivation behind RBAC is to simplify the administration. An appealing possibility is to use RBAC itself to manage RBAC, to provide further administrative convenience, especially in decentralizing administrative authority, responsibility, and tasks [2]. ARBAC97 (administrative RBAC '97), which is based on the RBAC96 model in Figure 1 [7], allows decentralized administration of user-role assignment (URA97), permission-role assignment (PRA97), and role-role assignment (RRA97).

In spite of the advantages and elegance of the ARBAC97 model, it also has some significant shortcomings, or undesirable side effects. The main point of decentralized RBAC administration is the control and scoping of the administration domain (or boundary) of each administrative role. For this purpose, ARBAC97 uses role ranges and prerequisite conditions. In particular, prerequisite roles are used as user and permission pools for administration roles. As we will see, this approach has some weaknesses due to undesirable couplings.

The objective of this paper is to analyze the weaknesses of the ARBAC97 model, and to propose an improved administration model named ARBAC02. ARBAC02 retains the main features of ARBAC97, and adds the concept of the organization unit as new user and permission pools. We modify the URA97 and PRA97 components of ARBAC97. RRA97 is not modified in this work.

The rest of this paper is organized as follows. Section 2 presents our motivation. We briefly review ARBAC97 and describe its weaknesses. Section 3 presents the ARBAC02 model. We introduce organization structure as a candidate for user and permission pools, describe our modification of URA97 and PRA97, and show the advantages of the ARBAC02 model. Section 4 presents our conclusions.

SACMAT'02, June 3-4, 2002, Monterey, California, USA.

2. MOTIVATION

2.1 Summary of ARBAC97 Model

ARBAC97 has three components: URA97 is concerned with user-role administration, PRA97 is concerned with permission-role administration, and RRA97 deals with role-role administration. We focus on URA97 and PRA97. Detailed motivations and rationales for URA97 and PRA97 are given in other papers [1-3, 7, 11]. Here we explain these models by an example using the regular role hierarchy and administrative role hierarchy of Figures 2 and 3.

■ URA97 Model

URA97 has two components, one dealing with the assignment of users to roles (the grant model) and the other with revocation of user membership (the revocation model). User-role assignment is controlled in URA97 by the *can-assign* relation such as:

can-assign(x, y, z) // x: administrative role,
y: prerequisite condition, z: role range

For example, *can-assign*(PSO1, ED, {E1}) means that a member of the administrative role PSO1 (or a member of an administrative role senior to PSO1) can assign a user who has current membership in ED to be a member of the regular role E1. The *prerequisite condition* is a Boolean expression of the *prerequisite role* and/or *constraint*. For example, in the prerequisite condition 'E1 ∧ $\overline{\text{QE1}}$', 'E1' is a prerequisite role and '$\overline{\text{QE1}}$' is a constraint. The prerequisite condition 'E1 ∧ $\overline{\text{QE1}}$' indicates users who belong to E1 and do not belong to QE1. User revocation in URA97 is

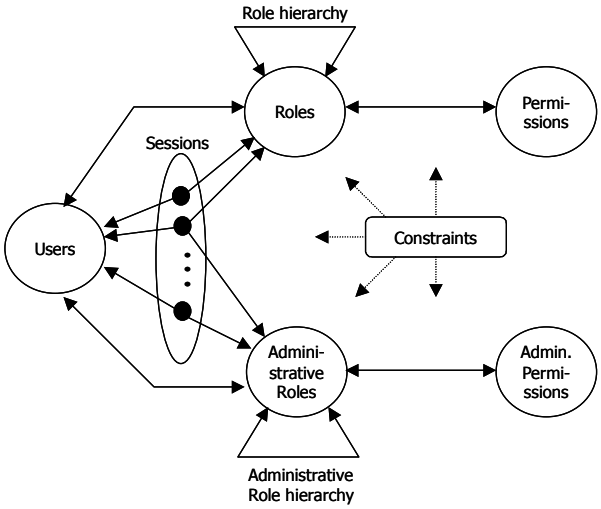


Figure 1. Summary of the RBAC96 Model

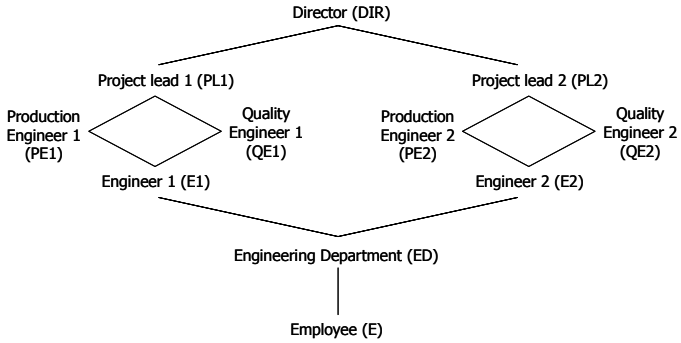


Figure 2. Example of Regular Role Hierarchy

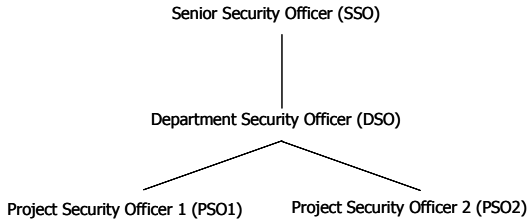


Figure 3. Example of Administrative Role Hierarchy

controlled by the *can-revoke* relation, such as:

can-revoke(x, z) // x: administrative role,
z: role range

For example, *can-revoke*(PSO1, {PE1, QE1}) means that a member of the administrative role PSO1 (or a member of an administrative role senior to PSO1) can revoke a user whose current membership is in PE1 or QE1. Table 1 and Table 2 show examples of *can-assign* and *can-revoke* in URA97. Here the role ranges are expressed by identifying lower and upper boundary points in a role hierarchy.

In the role range a '(' or ')' mean that the range does not include a boundary value. '[' or ']' means that the range includes a boundary value. For example, '[E1, PL1)' is equivalent to {E1, PE1, QE1} in Figure 2.

Table 1. Example of *can-assign* in URA97

Admin. Role	Prereq. Condition	Role Range
PSO1	ED	[E1, E1]
PSO1	$E1 \wedge \overline{QE1}$	[PE1, PE1]
PSO1	$E1 \wedge \overline{PE1}$	[QE1, QE1]
PSO2	ED	[E2, E2]
PSO2	$E2 \wedge \overline{QE2}$	[PE2, PE2]
PSO2	$E2 \wedge \overline{PE2}$	[QE2, QE2]
DSO	$ED \wedge \overline{PL2}$	[PL1, PL1]
DSO	$ED \wedge \overline{PL1}$	[PL2, PL2]
DSO	ED	(ED, DIR)
SSO	E	[ED, ED]
SSO	ED	(ED, DIR]

Table 2. Example of *can-revoke* in URA97

Admin. Role	Role Range
PSO1	[E1, PL1)
PSO2	[E2, PL2)
DSO	(ED, DIR)
SSO	[ED, DIR]

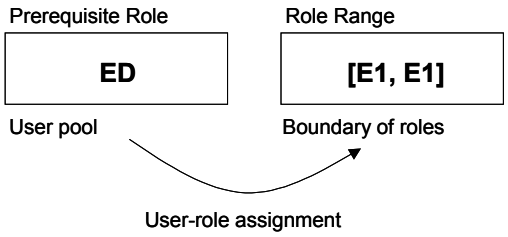


Figure 4. Relationship between Prerequisite role and Role Range in URA97

What is the point of URA97? In the URA97 model, the role range and prerequisite condition (or role) are used to restrict each administrative role. The role range is used as the boundary of target roles to assign users, and the prerequisite role is used as a domain to pick users. Therefore we can replace the term 'prerequisite role' in URA97 with '**user pool**' (*See Figure 4*). User-role administration can be decentralized by assigning the proper user pool and role range to administrators.

■ PRA97 Model

PRA97 has similar features to URA97. PRA97 has two components, one dealing with assignment of permissions to roles and the other with revocation of permissions; these two components are controlled by the *can-assignp* and *can-revokep* relations, such as:

can-assignp(x, y, z) // x: administrative role,
y: prerequisite condition, z: role range

can-revokep(x, z) // x: administrative role,
z: role range

For example, *can-assignp*(DSO, DIR, [PL1,PL1]) means that a member of the administrative role DSO (or a member of an administrative role senior to DSO) can take any permission assigned to the DIR role and make it available to the regular role PL1. Tables 3 and 4 show examples of *can-assignp* and *can-revokep* in the PRA97 model.

Table 3. Example of *can-assignp* in PRA97

Admin. Role	Prereq. Condition	Role Range
DSO	DIR	[PL1, PL1]
DSO	DIR	[PL2, PL2]
PSO1	$PL1 \wedge \overline{QE1}$	[PE1, PE1]
PSO1	$PL1 \wedge \overline{PE1}$	[QE1, QE1]
PSO2	$PL2 \wedge \overline{QE2}$	[PE2, PE2]
PSO2	$PL2 \wedge \overline{PE2}$	[QE2, QE2]

Table 4. Example of *can-revokep* in PRA97

Admin. Role	Role Range
PSO1	(E1, PL1)
PSO2	(E2, PL2)
DSO	(ED, DIR)
SSO	[ED, DIR]

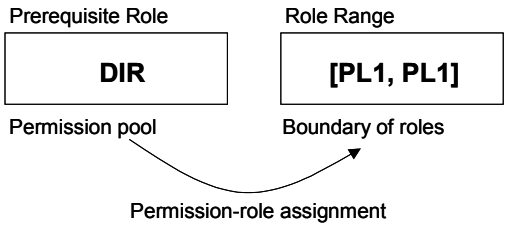


Figure 5. Relationship between Prerequisite Role and Role Range in PRA97

As can be seen, the prerequisite role in PRA97 is used as a domain for selecting permissions. Therefore we can replace the

term 'prerequisite role' in PRA97 with '**permission pool**' (See Figure 5). Permission–role administration can be decentralized by assigning the proper permission pools and role ranges to administrators.

2.2 Shortcomings of URA97 and PRA97

The ARBAC97 model supports simple and decentralized security administration. However, from a practicality viewpoint, it has some significant shortcomings. In this section, we describe some weaknesses of URA97 and PRA97 with respect to the prerequisite roles.

■ **Weaknesses in URA97**

UA1. Multi-step user assignment

Suppose that a newly employed engineer 'John' will be assigned to role 'QE1' role in the environment of Figure 2, Figure 3, and Table 1. To do so, John should be a member of prerequisite role 'E1'. Before John can become a member of 'E1', he must be a member of the prerequisite role 'ED'. Similarly, before John can be a member of 'ED', he should be a member of prerequisite role 'E'. To summarize, John's role assignments must follow the order:

assign John to E → assign John to ED → assign John to E1 → assign John to QE1

This example shows that URA97 requires multi-step user assignments. Roles higher in the role hierarchy may require more assignment steps. This may require the work of two or more security officers.

UA2. Duplicated user-role assignment (UA) information

Suppose that 'Tom' is a member of the QE1 role. Tom is therefore an explicit member of 'E', 'ED', 'E1' and 'QE1', and the corresponding information exists in the URA information as shown in Table 5, as a result of the multi-step user assignment. In Table 5, tuples 1, 3, and 5 do not affect Tom's access rights because 'QE1' inherits the access rights of 'E', 'ED', and 'E1'. From the point of view of Tom's access rights, the three tuples are redundant. They are required only for administrative purposes. One thousand users and four-step user-role assignment require 4000 tuples in the URA information, although only 1000 tuples are required for access control.

Table 5. User-Role Assignment Information

No	Role	Assigned user
1	E	Tom
..	..	..
3	ED	Tom
..	..	..
5	E1	Tom
..	..	..
7	QE1	Tom
..	..	..

UA3. Restricted composition of user pool

Suppose the company in this example wants to maintain human resource pools H1, H2, and H3. Suppose also a new policy requires that a 'Production Engineer' should be selected from H1 and a 'Quality Engineer' should be selected from H2. It is impossible to realize this new policy without changing the role hierarchy. In the URA97 model, the user pool is based on the prerequisite roles, and the prerequisite roles are part of the role hierarchy. This example shows that thereby the user pool is restricted by the structure of roles or the role hierarchy. Sometimes the real world requires a more flexible user pool, and this causes a more complicated role hierarchy in URA97. URA97 has an unnecessary coupling between two distinct concepts.

■ **Weaknesses in PRA97**

PA1. Multi-step permission assignment
PA2. Duplicated permission-role assignment (PA) information
PA3. Restricted composition of permission pool

We omit the explanation of problems PA1, PA2, and PA3 because they are similar to UA1, UA2, and UA3.

PA4. No restriction for permission pool

Suppose there exists a *can-assignp*(SO1, R2, [R1,R1]). Then SO1 can assign any permission of R2 to R1. There is no restriction. How is it possible to specify some permission for R2 only? This cannot be directly expressed in PRA97. In the PRA99 model [4] this requirement is expressed by the immobile membership concept. However, this approach requires additional information about the permission pool.

PA5. Undesirable side effect

PA5 is a corollary of PA4. Consider the role hierarchy and role range of Figure 6. If *can-assignp*(PSO1, PL1, [QE1, QE1] exists, then 'PSO1' can assign any permissions of 'PL1' to 'QE1'. Consequently, 'QL' inherits all permissions of 'QE1' because 'QL' is a parent role of 'QE1'. It means that 'PSO1' can move some permissions of 'PL1' to 'QL'. However, as can be seen, 'QL' is outside the role range of 'PSO1', so this permission flow is illegal.

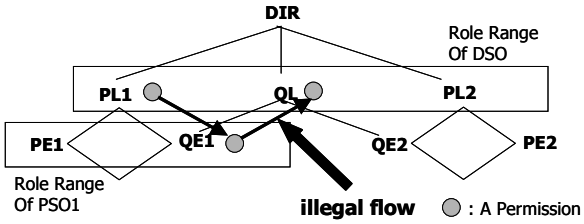


Figure 6. Undesirable Side Effect in PRA97 Model

What are the origins of these shortcomings of ARBAC97? First, the user pool and permission pool are dependent on the structure of the role or role hierarchy. A prerequisite role is dependent on its lower or higher prerequisite roles. (As described above, a prerequisite role functions as a user pool or a permission pool.) As a result, all prerequisite roles form a dependency chain along the role hierarchy. Figure 7 shows this situation. This dependency is a strong restriction for constructing the user pool or permission pool (UA3/PA3). In addition, it causes duplicate administrative work (UA1/PA1) and redundant data (UA2/PA2).

Second, there is the top-down nature of permission-role administration, allowing security administrators to select any permission from their prerequisite roles. This leads to undesirable behavior (PA4/PA5).

If we want to apply the ARBAC97 model in the real world, we should resolve these problems.

3. THE ARBAC02 MODEL

To overcome the weaknesses of ARBAC97 model, we chose two strategies. First, we use the organization structure as new user and permission pools instead of prerequisite roles in a role hierarchy. Second, based on the organization structure, we propose a bottom-up approach to permission-role assignment. Before we describe the ARBAC02 model, we introduce the concept of organization structure.

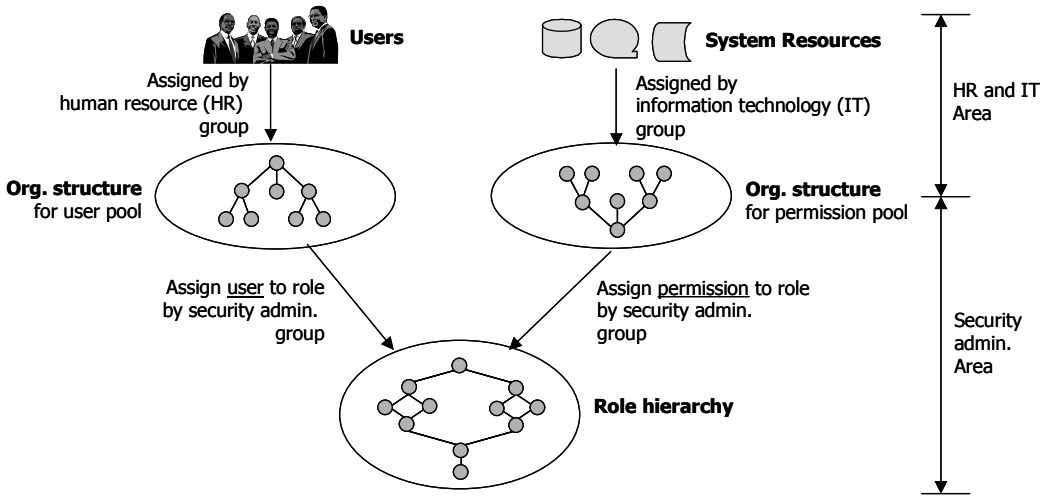
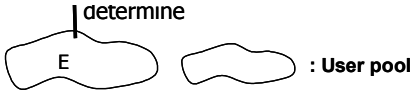
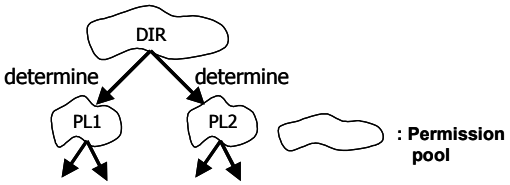


Figure 8. Role Administration Concept in ARBAC02



(a) Dependency among user pools in URA97



(b) Dependency among permission pools in PRA97

Figure 7. Dependencies in URA97/PRA97
(Above is inferred from Table 1 and Table 3)

3.1 Organization Structure as a User / Permission Pool

For information systems development the organization is a good concept as a domain for analysis of business functions and activities. Generally, organization structure is a tree structure and has the characteristic of inheritance. An organization structure is composed of organization units, each encompassing the relevant people who work to achieve the mission of the organization unit. To achieve the given mission, each organization unit has a set of job functions or tasks. To perform the job functions or tasks, users need to access information resources. In other words, job functions or tasks are related to permissions. We can redefine the organization unit as 'a group of people and functions (permissions) to achieve the given mission'. Moffett discussed the meaning of organization and role hierarchy [8, 9]. Perwaiz and Sommerville showed how to manage permission–role relationship using organization units [5]. Therefore the organization unit is a suitable container for the user pool and permission pool. For the purpose of role administration, we can use different organization structures for the user pool and the permission pool. There is no conflict because these organization structures are only used for the purpose of user pool or permission pool administration, respectively.

Figure 8 shows the role administration concept in the ARBAC02 model. First, users and permissions are assigned to proper organization units by the Human Resources (HR) and Information Technology (IT) groups. Then the security administration group assigns the users and permissions in organization units to regular roles. We do not elaborate the functions of the HR and IT groups because they are outside the scope of role-based security administration. We assume the activities of these groups are somehow accomplished in the system.

3.2 Description of ARBAC02 Model

In this section, we describe the central notions of ARBAC02 model: to adopt new user and permission pools independent of the role or role hierarchy and a bottom-up method of permission-role administration.

■ **New user and permission pools**

Definition 1. OS-U is an organization structure represented as a user pool. It contains users who are pre-assigned by the HR group. Figure 9 shows an example of the OS-U. Any organization unit can have users. If 'Tom' is a member of 'PJ1', it may mean that he has a job position in project 1. If 'John' is a member of 'ED', it may mean that he is a director of an engineering department. OS-U has a tree structure and the characteristic of inheritance. Therefore Tom as a member of 'PJ1' is also a member of 'ED' and 'PRD'.

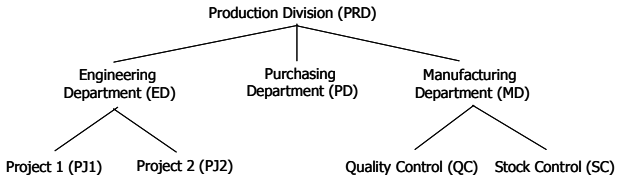


Figure 9. An Example of OS-U (user pool)

Definition 2. OS-P is an organization structure represented as a permission pool. It contains permissions that are pre-assigned by the IT group. OS-P has an inverted tree structure as shown in Figure 10. In OS-P, common permissions are assigned to the lower sections of the organization structure and special permissions are assigned to the higher sections of the organization structure. For example, access permissions for all members of the production division are assigned to 'PRD' and special permissions for members of project 1 are assigned to 'PJ1'.

From Figure 10, we can expect that users belonging to 'PJ1' inherit permissions in 'ED' and 'PRD'. However, it is important that the permission membership is inherited downward in OS-P. For example, the entire set of permissions for 'ED' is {permissions assigned to 'ED'} ∪ {permissions assigned to 'PJ1'} ∪ {permissions assigned to 'PJ2'}.

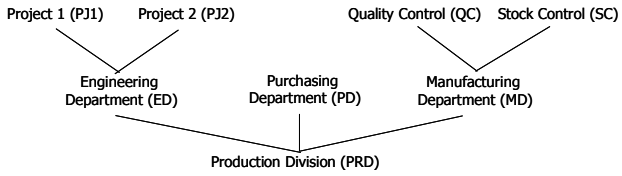


Figure 10. An Example of OS-P (permission pool)

Assumption 1. We assume that simple and basic versions of OS-U and OS-P are given, and users and permissions are pre-assigned to the proper positions of the given organization structures.

There are various policies for maintaining OS-U and OS-P. It is obvious that maintaining OS-U and OS-P requires the cooperation of the security administration group, the HR group, and the IT group. Figure 11 shows the ARBAC02 model including the two new components, OS-U and OS-P.

Now we redefine the prerequisite condition in the ARBAC97 model.

Definition 3. A **Prerequisite condition of URA** is a Boolean expression using the usual ∧ and ∨ operators on terms of the form

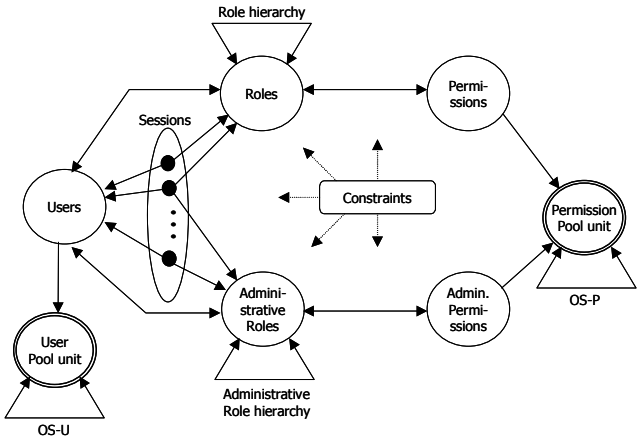


Figure 11. Components of ARBAC02 Model

x and $\bar{x}$, where x is a *regular role* or *organization unit in OS-U*. A prerequisite condition is evaluated for a user u by interpreting x to be true if

Case 1. $x \in$ *role*: $(\exists x' \geq x)(u, x') \in URA$
Case 2. $x \in$ *org. unit of OS-U*: $(\exists x' \leq x)(u, x')\} \in UUA$

and $\bar{x}$ to be true if

Case 1. $x \in$ *role*: $\neg(\exists x' \geq x)(u, x') \in URA$
Case 2. $x \in$ *org. unit of OS-U*: $\neg(\exists x' \leq x)(u, x')\} \in UUA$

Definition 4. A Prerequisite condition of PRA is a Boolean expression using the usual ∧ and ∨ operators on terms of the form x and $\bar{x}$ where x is a *regular role* or *organization unit in OS-P*. A prerequisite condition is evaluated for a permission p by interpreting x to be true if

Case 1. $x \in$ *role*: $(\exists x' \leq x)(p, x') \in PRA$
Case 2. $x \in$ *org. unit of OS-P*: $(\exists x' \geq x)(p, x')\} \in PPA$

and $\bar{x}$ to be true if

Case 1. $x \in$ *role*: $\neg(\exists x' \leq x)(p, x') \in PRA$
Case 2. $x \in$ *org. unit of OS-P*: $\neg(\exists x' \geq x)(p, x')\} \in PPA$.

(Note. *URA*: user-role assignment, *UUA*: user-organization assignment on OS-U, *PRA*: permission–role assignment, *PPA*: permission-organization assignment on OS-P. To distinguish role and organization unit names, we use an '@' in the head of organization unit names.)

ARBAC02 adopts the same notation of *can-assign*, *can-revoke*, *can-assignp*, and *can-revokep* from ARBAC97. The difference in definition of the prerequisite condition between the ARBAC97 and ARBAC02 models is that prerequisite roles are replaced by organization units. The effect is described in the next section. Following the redefinition of the prerequisite condition, an example of *can-assign* in ARBAC97:

can-assign(PSO1, E1 ∧ $\overline{\text{QE1}}$, [PE1,PE1])

can be described in the ARBAC02 model as:

can-assign'(PSO1, @PJ1 ∧ $\overline{QE1}$, [PE1,PE1]).

Table 6 shows refined *can-assign* equivalent to Table 1.

Table 6. Refined *can-assign* Equivalent to Table 1

Admin. Role	Prereq. Condition	Role Range
PSO1	@PJ1 ∧ $\overline{QE1}$	[PE1, PE1]
PSO1	@PJ1 ∧ $\overline{PE1}$	[QE1, QE1]
PSO2	@PJ2 ∧ $\overline{QE2}$	[PE2, PE2]
PSO2	@PJ2 ∧ $\overline{PE2}$	[QE2, QE2]
DSO	@ED ∧ $\overline{PL2}$	[PL1, PL1]
DSO	@ED ∧ $\overline{PL1}$	[PL2, PL2]
DSO	@ED	(ED, DIR)
SSO	@ED	(ED, DIR]

■ Bottom-up approach to permission-role administration

One of the weaknesses of ARBAC97 is the top-down approach to permission–role administration. The ARBAC02 model adopts a bottom-up approach. In the ARBAC02 model, common permissions are assigned to lower roles in the role hierarchy, and higher roles inherit common permissions, while special permissions are assigned to higher roles. For example, common permissions for all users are assigned to role 'E', common permissions for engineering department members are assigned to 'ED', and common permissions for Project 1 members are assigned to 'E1'. The remaining special permissions are assigned to appropriate higher roles of 'E1'. Table 7 shows an example of the refined *can-assignp*.

Table 7. An Example of Refined *can-assignp*

Admin. Role	Prereq. Condition	Role Range
SSO	@ED	[E, DIR]
DSO	@ED	[ED, DIR)
PSO1	@PJ1	[E1, PL1)
PSO2	@PJ2	[E2, PL2)
PSO1	@PJ1 ∧ $\overline{QE1}$	[PE1, PE1]
PSO1	@PJ1 ∧ $\overline{PE1}$	[QE1, QE1]
PSO2	@PJ2 ∧ $\overline{QE2}$	[PE2, PE2]
PSO2	@PJ2 ∧ $\overline{PE2}$	[QE2, QE2]

One advantage of this approach is that we avoid duplicate assignments of the same permission through the inheritance line of the role hierarchy. For example, a permission that is assigned to 'E' is not required by 'ED', 'E1', and so on, therefore eliminating redundancy.

3.3 Advantages of the ARBAC02 Model

■ The effects of user and permission pools are independent of role or role hierarchy

As described above, the user pool in the ARBAC97 model is implemented by prerequisite roles, and a prerequisite role depends on its prerequisite role in turn. As a result, the ARBAC97 model induces multi-step user assignment (UA1) and redundant user-role assignment information (UA2). Furthermore, composition of the user pool is strongly restricted by the role hierarchy (UA3). In the ARBAC02 model the user pool is implemented by the organization unit independently of the role or role hierarchy. A new user can be registered into the proper user pool in one step, and be assigned to the proper role from the user pool in one step. It is important that **assigning a user to a user pool is separate from assigning a user to a regular role** in the ARBAC02 model. As a result, the user assignment becomes simple and there is no redundant user-role assignment information. (UA1 and UA2 are resolved.) Figure 12 shows the comparison of the user-role

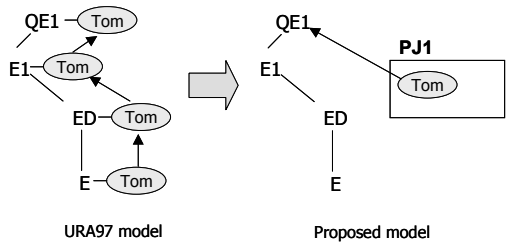


Figure 12. Comparison of User-Role Administration

administration in the URA97 and ARBAC02 models.

Let us recall the situation in UA3. A company wants to maintain human resource pools H1, H2, and H3. A new policy requires that a 'Production Engineer' should be selected from H1 and a 'Quality Engineer' should be selected from H2. In the ARBAC02 model, new organization units H1, H2, and H3 can be added at the proper positions in the organization structure. Then we change a prerequisite condition such as:

can-assign(PSO1, PJ1 ∧ $\overline{QE1}$, [PE1, PE1])

to:

can-assign'(PSO1, @H1, [PE1, PE1])

This requires no change of role hierarchy because the user pool is independent of the role hierarchy. (UA3 is solved.) PA1, PA2, and PA3 are solved similarly.

■ The effects of bottom-up permission–role administration

In the ARBAC02 model, common permissions for many roles are assigned to lower positions in the role hierarchy while non-common roles are assigned higher positions. Common permissions are inherited by senior roles through the role hierarchy. Permissions do not propagate downward in the role hierarchy (as in ARBAC97). As a result, PL1 is not a prerequisite role for PSO1 in Figure 13, and PSO1 cannot assign PL1's permissions to his/her role range. The undesirable side effect of PA5 does not occur. PA4 is solved naturally because we do not

adopt top-down permission–role administration, so the prerequisite role is not restricted as a permission pool.

The ARBAC02 model overcomes the identified shortcomings of URA97 and PRA97. It supports flexible composition of the user and permission pools. The ARBAC02 model must maintain additional components, namely the organization structure, but this is not an extensive overhead. Moreover, the organization structure is a natural notion for organizations.

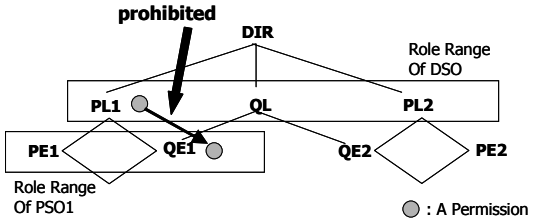


Figure 13. Prohibition of Downward Permission Flow

■ **The possibility of applying ARBAC02 to other areas**

The ARBAC02 model is suitable for any areas requiring the RBAC model. Furthermore the concept of the user and permission pools can be separated from the ARBAC02 model, and can be

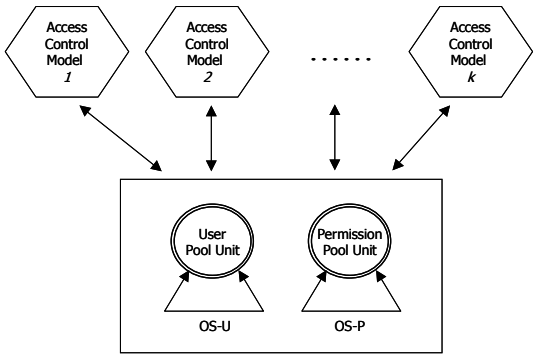


Figure 14. Applying User/Permission Pool to Other Models

applied to non-RBAC environments, as shown in Figure 14. This may be an interesting research topic.

4. CONCLUSION

The ARBAC97 model is simple and easily understood. ARBAC97 shows that RBAC itself can be used to manage RBAC. However, from a practical viewpoint ARBAC97 has some serious shortcomings caused by unnecessary integration of the user and permission pools and the role hierarchy. In this paper, we introduce the organization structure as new user and permission pools independent of role or role hierarchy. In addition, we introduce a bottom-up permission–role administration, in contrast to the top-down manner of ARBAC97. Independent user and permission pools give flexibility in constructing the user and permission pools, and overcome the identified weaknesses of the ARBAC97 model. We define ARBAC02 as an improved version of ARBAC97 and show how the weaknesses of ARBAC97 are eliminated.

We believe that managing the user and permission pools and managing the user-role and permission–role assignments are related, but they are different areas. We do not describe in detail the method of managing the user and permission pools, which is delegated to the HR and IT groups. Developing an integrated model that explicitly incorporates the activities of these groups is a future research topic.

5. ACKNOWLEDGMENTS

This work is partially supported by the National Science Foundation.

6. REFERENCES

[1] Ravi Sandhu and Venkata Bhamidipati, “The URA97 model for role-based user-role assignment”, In Proceedings of IFIP WG 11.3 Workshop on Database Security, August 1997.

[2] Ravi Sandhu and Venkata Bhamidipati, “The ARBAC97 model for Role-based administration of Roles: Preliminary Description and Outline”, In Proceedings of second ACM Workshop on Role-Based Access Control. November 1997.

[3] Ravi Sandhu and Venkata Bhamidipati, “Role-based administration of user-role assignment: The URA97 model and its Oracle implementation”, The Journal of Computer Security, Vol.7, 1999

[4] Ravi Sandhu and Qamar Munawer, “The ARBAC99 model for administration of roles”, In Proceedings of the Annual Computer Security Applications Conference. 1999.

[5] Najam Perwaiz and Ian Sommerville, “Structured management of role-permission relationships”, In Proceedings of 6th ACM Symposium on Access Control Models and Technologies (SACMAT2001), May 2001.

[6] James B.D. Joshi, Walid G. Aref, Arif Ghafoor, and Eugene H. Spafford, “Security models for web-based applications”, Communications of the ACM, Vol. 44, No.2, February 2001.

[7] Ravi Sandhu, Edward Coyne, Hal Feinstein and Charles Youman, “Role-Based Access Control Models.” *IEEE Computer*, Volume 29, Number 2, February 1996, pages 38-47.

[8] Jonathan D. Moffett, “Control Principles and Role Hierarchies”, In Proceedings of the 3rd ACM Workshop on Role-Based Access Control. October 1998.

[9] Jonathan D. Moffett and Emil C. Lupu, “The use of role hierarchies in access control”, In Proceedings of the 4th ACM Workshop on Role-Based Access Control. October 1999.

[10] Sylvia Osborn, Ravi Sandhu and Qamar Munawer, “Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies”, ACM Transactions on Information and System Security, Volume 3, Number 2, May 2000, pages 85-106.

[11] Ravi Sandhu, Venkata Bhamidipati and Qamar Munawer, “The ARBAC97 Model for Role-Based Administration of Roles”, ACM Transactions on Information and System Security, Volume 2, Number 1, February 1999, pages 105-135.

[12] Matunda Nyanchama and Sylvia Osborn, “The Role Graph Model and Conflict of Interest”, ACM Transactions on Information and System Security, Vol. 2, No. 1, February 1999, pages 3-33.