(1) Rationale for the RBAC96 Family of Access Control Models

Ravi Sandhu

George Mason University and SETA Corporation
ISSE Department, MS 4A4, George Mason University, Fairfax, VA 22030, USA
E-mail: sandhu@isse.gmu.edu

Abstract

A family of role-based access control (RBAC) models, referred to here as the RBAC96 models, was recently published by the author and his colleagues. This paper gives our rationale for the major decisions in developing these models and discusses alternatives that were considered.

1.0 Introduction

The RBAC96 family of RBAC models was recently defined by the author and his colleagues [SAND96b]. The scope and nature of our original paper did not accommodate detailed discussion of the issues and alternatives that were considered while developing these models. The objective of this paper is to describe the rationale for the major design decisions and to discuss alternate approaches that could have been taken.

The paper begins with a brief review of the RBAC96 models in Section 2.0. This review is intended as a refresher and readers should be familiar with the original paper [SAND96b] to establish the background and context. In Section 3.0, we discuss various issues that arose in the process of defining these models. Section 4.0 concludes the paper.

2.0 The RBAC Models

The family of RBAC96 models is summarized in Figure 1-1, *The RBAC96 Model*. This figure actually shows the most general model in this family. For simplicity, we overload the term RBAC96 to refer to the family of models as well as its most general member.

The top half of the figure shows roles and permissions in the system that regulate access to the data and resources. The bottom half shows administrative roles and administrative permissions. RBAC96 is based on five sets of entities called users (*U*), roles (*R*), and permissions (*P*), and their administrative counterparts called administrative roles (*AR*) and administrative permissions (*AP*). It is required that administrative roles and administrative permissions be respectively disjoint from the regular (i.e., non-administrative) roles and permissions. Moreover regular permissions can only be assigned to regular roles and administrative permissions can only be assigned to administrative roles.

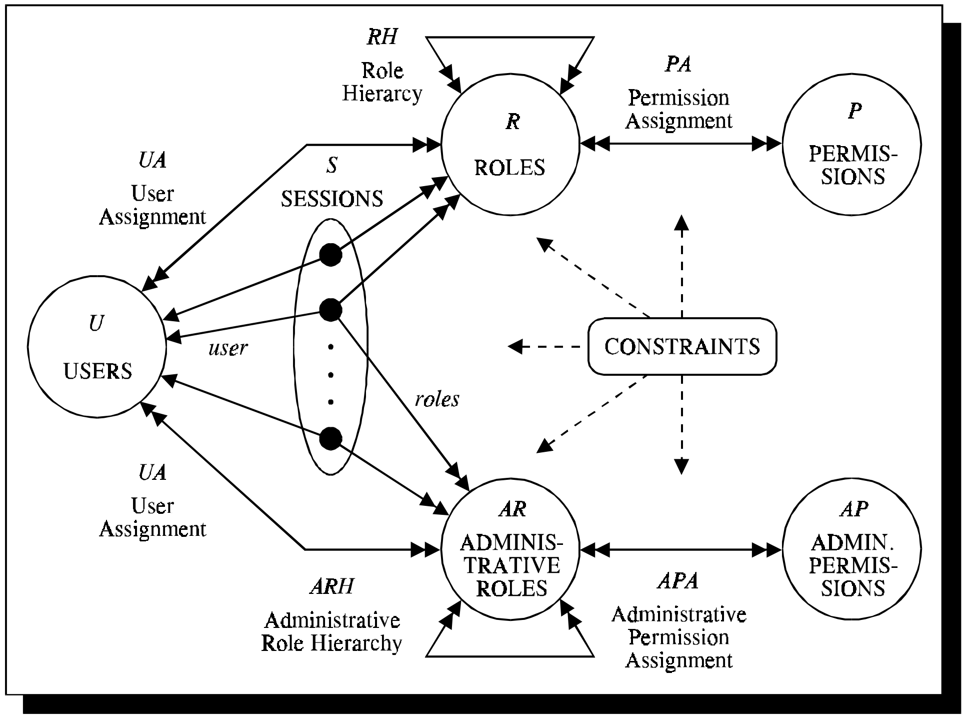

Figure 1-1. The RBAC96 Model

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. Administrative permissions control operations which modify the components of RBAC, such as adding new users and roles and modifying the user assignment and permission assignment relations. Regular permissions on the other hand control operations on the data and resources and do not permit administrative operations. We loosely use the term role to include both regular and administrative roles while making this distinction precise whenever appropriate. Similarly for the term permission.

The user assignment (*UA*) and permission assignment (*PA* and *APA*) relations of Figure 1-1 are many-to-many. A user can be a member of many roles, and a role can have many users. Similarly, a role can have many permissions, and the same permission can be assigned to many roles. There is a partially ordered role hierarchy *RH*, also written as $\geq$, where $x \geq y$ signifies that role x inherits the permissions assigned to role y. Inheritance along the role hierarchy is transitive and multiple inheritance is allowed in partial orders. There is similarly a partially ordered administrative role hierarchy *ARH*.

Each session in Figure 1-1 relates one user to possibly many roles. Intuitively, a user establishes a session during which the user activates some subset of roles that he or she is a member of (directly or indirectly by means of the role hierarchy). The double-headed arrows from a session to *R* and *AR* indicates that multiple roles and administrative roles can be simultaneously activated. The permissions available to the user are the union of permissions from all roles activated in that session. Each session is associated with a single user, as indicated by the single-headed arrow from the session to U. This association remains constant for the life of a session. A user may have multiple sessions open at the same time, each in a different window on the workstation screen for instance. Each session may have a different combination of active roles. The concept of a session equates to the traditional notion of a subject in access control. A subject (or session) is a unit of access control, and a user may have multiple subjects (or sessions) with different permissions active at the same time.

Finally, Figure 1-1 shows a collection of constraints. Constraints can apply to any of the preceding components. An example of constraints is mutually disjoint roles, such as purchasing manager and accounts payable manager, where the same user is not permitted to be a member of both roles.

The following definition formalizes the above discussion.

Definition 1. The RBAC96 model has the following components:

- U is a set of users;
- R and AR are disjoint sets of roles and administrative roles respectively;
- P and AP are disjoint sets of permissions and administrative permissions;
- $UA \subseteq U \times (R \cup AR)$, is a many-to-many user to role, and administrative role, assignment relation;
- $PA \subseteq P \times R$ and $APA \subseteq \mathrm{AP} \times AR$, are respectively many-to-many permission to role assignment and administrative permission to administrative role assignment relations;
- $RH \subseteq R \times R$ and $ARH \subseteq AR \times AR$, are respectively partially ordered role and administrative role hierarchies (written as $\geq$ in infix notation);
- S is a set of sessions;
- $user : S \rightarrow U$, is a function mapping each session s_i to the single user $user(s_i)$ and is constant for the session's lifetime;
- $roles : S \rightarrow 2^{R \cup AR}$ is a function mapping each session s_i to a set of roles $roles(s_i) \subseteq \{r \mid (\exists r' \geq r)[(user(s_i), r') \in UA]\}$ (which can change with time) so that session s_i has the permissions $\cup_{r \in roles(si)}\{p \mid (\exists r'' \leq r)[(p, r'') \in PA \cup APA]\}$; and
- There is a collection of constraints stipulating which values of various components of the RBAC model are allowed or forbidden.

3.0 Rationale for the RBAC96 Models

This section describes our rationale for resolving various issues that arose during development of the RBAC96 models. We also discuss alternatives that were considered.

3.1 A Family of Models

The decision to develop a family of models rather than a single all-encompassing model was made early in the project. It is evident that the RBAC96 model described in the previous section is complex and has multiple facets. The end result shown in Figure 1-1 was actually developed incrementally and would be difficult to construct in one single step. Our initial efforts at characterizing the multidimensional aspects of RBAC are discussed in [SAND94b].

The RBAC96 family shown in Figure 1-2, *The RBAC96 Family*, consists of RBAC with respect to regular roles and permissions on the left and RBAC with respect to administrative roles and permissions on the right. The left and right components of Figure 1-2 respectively relate to the top and bottom halves of Figure 1-1, and are similarly mirror images of each other. Looking at the left half of Figure 1-2, we have $RBAC_0$, the base model, at the bottom, indicating that it is the minimum requirement for RBAC. $RBAC_1$ and $RBAC_2$ both include $RBAC_0$, but add independent features to it. $RBAC_1$ adds the concept of role hierarchies (situations where roles can inherit permissions from other roles). $RBAC_2$ adds constraints (which impose restrictions on acceptable configurations of the different components of RBAC). $RBAC_1$ and $RBAC_2$ are incomparable to one another. The consolidated model, $RBAC_3$, includes $RBAC_1$ and $RBAC_2$ and, by transitivity, $RBAC_0$.

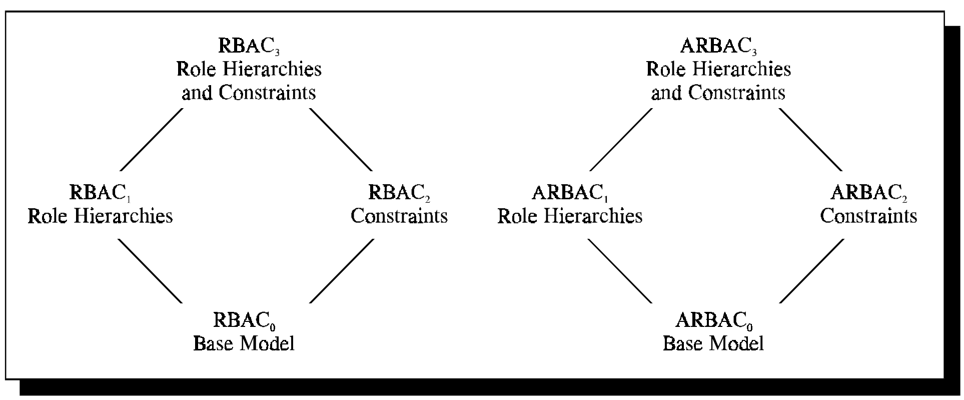

Figure 1-2. The RBAC96 Family

While developing RBAC96, we were driven by the fact that the main motivation for RBAC, and its main advantage, is to facilitate administration of permissions. This led us to ask how RBAC can be used to manage RBAC itself. We feel that the use of RBAC for managing RBAC will be an important factor in the success of RBAC. It seemed natural to us to structure the administrative RBAC models in the same way as the regular RBAC models. The top half of Figure 1-1 can range in sophistication across $RBAC_0$, $RBAC_1$, $RBAC_2$, and $RBAC_3$. The bottom half can similarly range in sophistication across $ARBAC_0$, $ARBAC_1$, $ARBAC_2$, and $ARBAC_3$, as shown in the right-hand side of Figure 1-2 (the *A* denotes administrative). In general, we would expect the administrative model to be simpler than the RBAC model itself. Thus, $ARBAC_0$ can be used to manage $RBAC_3$, but there seems to be no point in using $ARBAC_3$ to manage $RBAC_0$.

In the RBAC96, family we treat role hierarchies and constraints as being two independent extensions to $RBAC_0$. Strictly speaking, a role hierarchy can be considered as a constraint. The constraint is that a permission assigned to a junior role must also be assigned to all senior roles. Or equivalently, the constraint is that a user assigned to a senior role must also be assigned to all junior roles. So in some sense, $RBAC_1$ is redundant and is subsumed by $RBAC_2$. However, we felt it is appropriate to recognize the existence of role hierarchies in their own right. The concept of role hierarchies occurs very frequently in the literature and is natural to simplifying administration.

3.2 Users and Sessions

The distinction between a user and a session is a fundamental aspect of RBAC and consequently arises in $RBAC_0$. A user is a human being, or other intelligent agent, capable of autonomous activity in the system. To support the principle of least privilege a user should be allowed to login to a system with only those roles appropriate for a given occasion.

Many systems will turn on all permissions of a user irrespective of what the user wishes to accomplish in a particular session. Thus, a user who has powerful permissions (or roles) that are used only rarely when needed finds that these permissions are turned on all the time. It is possible to set up separate accounts, one in which the usual permissions are turned on and another in which the powerful permissions are turned on. Assigning multiple accounts to the same user introduces problems with respect to auditing, accountability, and constraints such as separation of duties. It is not a desirable general-purpose solution but can be used in the short term to simulate RBAC on existing platforms.

In $RBAC_0$, the distinction between users and sessions is useful only if users exercise discipline regarding the roles they normally invoke. With constraints, it may not be possible for a user to activate all their roles simultaneously. Consider a constraint that stipulates two roles which can be assigned to the same user but cannot be simultaneously activated in a session. For instance, a user may be qualified to be a pilot and a navigator but at any time can activate at most one of these roles. In presence of such constraints, a user cannot establish a single session with all the user's roles activated. Changing the roles activated in a session is a security-sensitive act and should be acknowledged to the security system via a so-called trusted path which guarantees that the user is making the request rather than some program acting on the user's behalf. Such changes can be regulated by constraints in $RBAC_1$. For instance, certain roles may not be dynamically added but can only be acquired when a session is created. $RBAC_0$ allows dynamic changing of roles in a session because of two reasons. From a conceptual viewpoint, constraints belong in $RBAC_1$ and higher, and should not be present in $RBAC_0$. We could still define $RBAC_0$ to disallow all changes in a session's roles. We felt this is impractical and too restrictive for a base model.

An important property of a session is that the user associated with a session cannot change. In many applications, there are long-lived sessions where one user hands over to another without a logout and login. This preserves the integrity of the computing activity being performed in the session. We feel this problem is an artifact of existing system architectures. Continuity of activity across multiple security sessions should be possible in properly engineered systems. Also our models are conceptual models seeking to capture what needs to be achieved. In implementations on specific platforms, we will need to simulate the requirements with the mechanisms available.

The RBAC96 models do not address the issues of idle session termination and lockout. In practice, this is an important issue. In our conceptual framework, termination and lockout is most easily modeled as a constraint and belongs in $RBAC_1$. As a practical matter, it would be hard to effectively do $RBAC_0$ without bringing in at least a small number of constraints of this nature.

Although we did not anticipate this in our construction of RBAC96, the distinction between users and sessions and the ability to constrain roles that can be simultaneously activated in a single session turns out to be critical for simulating lattice-based access controls by means of roles [SAND96b].

3.3 Permissions

It is difficult to identify the nature of permissions precisely in an abstract general purpose model such as RBAC96. Permissions tend to be implementation dependent. In lattice-based access control models [SAND93], it is possible to abstract the essential operations into read and write. This is because these models are focussed on one-directional information flow in a lattice of security labels.

RBAC models are policy neutral. Hence, the nature of permissions has to be open ended. In applying RBAC to a particular system, the interpretation of permissions is among the most important steps to be performed.

We deliberately decided to exclude so-called negative permissions from RBAC96. Negative permissions deny rather than confer access. They are used in some discretionary access control models to disallow a user from obtaining a permission from some alternate source. The use of constraints in RBAC is a much more useful mechanism to achieve the same result. The literature on negative permissions is fraught with problems concerning their interaction and relative strength with respect to positive permissions. In the presence of role hierarchies, this could become very complicated and arcane. We would be very reluctant to add negative permissions into a complex model such as RBAC96.

The scope of RBAC is also consciously limited to classical permissions. Sequencing or temporal dependencies between permissions are important in emerging applications such as workflow [THOM94]. We decided to limit the scope of RBAC to exclude these for two reasons. Firstly, these are not yet well understood and much further basic research is required for this purpose. Secondly, RBAC must have a well-delineated scope otherwise it will be an amorphous concept which can be taken to include all kinds of security and authorization issues.

3.4 Administrative Model

In large systems, the number of roles can be in the hundreds or thousands. Managing these roles and their interrelationships is a formidable task that often is highly centralized and delegated to a small team of security administrators. Because the main advantage of RBAC is to facilitate administration of permissions, it is natural to ask how RBAC can be used to manage RBAC itself. We believe that the use of RBAC for managing RBAC will be an important factor in the success of RBAC.

RBAC96 makes a clear distinction between permissions and administrative permissions and likewise between roles and administrative roles. In the philosophy of RBAC, the administrative model itself is policy neutral but does facilitate formulation and articulation of administrative policy. This is an important area for research and for the future of RBAC. Effective decentralized management of permissions within parameters established by central authority will be required to implement enterprise-wide information systems.

3.5 Model Conformance

What does it mean for a system to conform to RBAC96? RBAC96 is best viewed as a family of reference models which play a dual role. On one hand, RBAC96 provides a framework for analyzing the capabilities of existing systems to assess how well and how extensively they can support RBAC. RBAC96 also provides guidance to vendors and developers regarding access controls to be implemented in future systems. It is not necessary for a system to completely conform to $RBAC_0$ before it includes features of $RBAC_1$ or $RBAC_2$. Many existing systems do not distinguish between users and sessions. We would say these systems have aspects of $RBAC_0$, $RBAC_1$, and $RBAC_2$, but are also missing other aspects of $RBAC_0$. Other systems have hard-wired constraints, such as a session can only have one role at a time. Such systems cannot accommodate $RBAC_0$, because they do too much without any choice in the matter.

4.0 Conclusion

In this paper we have discussed design decisions made by the author and his colleagues in developing the RBAC96 models. Additional discussion is contained in the original paper [SAND96a] and we have focussed on issues which were not adequately discussed there. We feel that other access control modeling efforts can benefit from our approach of developing a family of models. We found it very useful to think about $RBAC_0$ without the complications of constraints and hierarchies.

Similarly, it was useful to think about hierarchies without considering constraints and vice versa. Finally, there is need to perform further research within the framework of RBAC96 to refine and develop this family of models, but we do not expect the framework itself to change very much.

Acknowledgments

This research is partly supported by contract 50-DKNB-5-00188 from the National Institute of Standards and Technology at SETA Corporation and by grant CCR-9503560 from the National Science Foundation at George Mason University.

References

[SAND93] Ravi S. Sandhu, "Lattice-based access control models," IEEE Computer 26(11):9-19, November 1993.

[SAND94b] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman, "Role-based access control: A multi-dimensional view," In Tenth Annual Computer Security Application Conference, pages 54-62, Orlando, FL, 5-9 December 1994.

[SAND96a] 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.

[SAND96b] Ravi S. Sandhu, "Role hierarchies and constraints for lattice-based access controls," In Elisa Bertino, editor, Proc. European Symposium on Research in Computer Security, Springer-Verlag, Rome, Italy, 1996. To appear as Lecture Notes in Computer Science, Computer Security - ESORICS96.

[THOM94] Roshan Thomas and Ravi S. Sandhu, "Conceptual foundations for a model of task-based authorizations," In IEEE Computer Security Foundations Workshop 7, pages 66-79, Franconia, NH, June 1994.