# (6) Issues in RBAC

Ravi Sandhu

George Mason University and SETA Corporation
sandhu@isse.gmu.edu

## 1.0 Discussion

During the course of the workshop, a number of issues were identified by various attendees. Roshan Thomas volunteered to maintain a list of these as the workshop progressed. Toward the end of the workshop, this list was presented to the attendees who then had the opportunity to suggest additional items. The objective was not so much to actually discuss these issues in detail but rather to explicitly identify them, particularly those that merit future discussion. Given the nature of this exercise, we know we have probably not identified all the important issues but we certainly have identified quite a few. At the workshop we had time to extensively discuss only one of these issues concerning the difference between roles and groups. This discussion is summarized elsewhere in Part I of this proceedings.

The issues identified by the workshop attendees are enumerated below along with a brief explanation as to the nature of the question and the discussion that took place.

***1. What is a role?***

This is perhaps the most fundamental question that can be raised for role-based access control (RBAC). The concept of a role originated in organizational theory much before the advent of computerized information systems. Even in the context of modern information systems, roles have significance beyond their application in security and access control. From the perspective of RBAC, it is therefore important to distinguish the concept and scope of a role for access control purposes as opposed to the more general organizational context in which roles arise. It was argued that roles have much bigger significance than access control but we should not be tempted to expand the access control arena as a consequence. Instead we should focus on aspects of roles that are relevant from the access control perspective. This question does impact activities such as role engineering, because the design of roles may need to take the bigger picture into account even though the immediate focus is on roles for access control purposes.

***2. How are roles different from groups for access control purposes?***

Early timesharing operating systems introduced the notion of a group of users which can occur as a single entity in access control lists, thereby conferring the associated permissions to all members of the group. Groups have since become commonplace in operating systems, including the most recent ones. It is often perceived that groups can serve the same purpose as roles, and perhaps RBAC is simply inventing a new term for an old



ACM RBAC Workshop, MD, USA
© 1996 ACM 0-89791-759-6/95/0011 $3.50

idea. There was extensive discussion of this issue at the workshop. It is discussed separately in Part I.

***3. Are negative permissions useful for RBAC?***

Permissions are usually positive in that a permission confers the ability to do something. Negative permissions (or denials), on the other hand, deny abilities. Negative permissions are sometimes used in combination with positive permissions. A typical example is that a group of users is given a positive permission but some members of the group are given a corresponding negative permission so these users do not inherit the positive permission from the group. Negative permissions introduce several complications in determining whether or not a particular access is permitted. RBAC models can accommodate the useful aspects of negative permissions by means of constraints. The question is whether negative permissions are useful in RBAC as an alternative to constraints, or perhaps for some other purpose.

***4. Should duties or obligations be considered part of RBAC?***

Access control is concerned with whether or not operations invoked by active entities such as users should be permitted to occur. Duties (or obligations) on the other hand require a user (or other active agent) to perform some operations in the system. Duties have not been part of traditional access control, but perhaps in context of RBAC they need to be considered. The general opinion at the workshop was in favor of keeping non-access control issues such as duties outside the scope of RBAC. This is consistent with the discussion on question 1 above.

***5. What is the relationship of RBAC to enterprise models?***

It is anticipated that roles will arise naturally in enterprise models. However, the scope of roles in enterprise models extends beyond their use for access control (as discussed in context of questions 1 and 4 above). An enterprise model that incorporates roles would be useful in designing roles for access control purposes, but RBAC models have a much narrower scope than enterprise models.

***6. What is the nature of permissions and operations mediated by RBAC?***

The nature of permissions and operations mediated by RBAC depends on the nature of the system in which RBAC is embedded. Operating systems usually control permissions such as read, write and execute on files. RBAC is useful at the level of these primitive permissions. However, RBAC has potentially even greater benefit if it can be applied at the level of abstract application-oriented operations such as credit and debit operations on an account. Credit and debit both require read and write access to the account balance, so they cannot be distinguished if protection is provided only in terms of read and write. On the other hand, the credit and debit operations themselves must be programmed as part of the application. There are several techniques which allow the program implementing a credit (or debit) operation to execute with different permissions than directly available to the user who invokes that program. Operating systems and database management systems should provide facilities for application programmers to effectively build such application programs that embody abstract application-level operations and to protect these by means of RBAC.

***7. What is the relationship between roles and stored procedures?***

Stored procedures are one technique by which application programs that implement abstract operations, such as credit or debit, can be executed with different permissions than the user invoking the operation (see discussion for question 6). Stored procedures usually execute with the permissions of the owner of the procedure. Would it be useful to allow stored procedures to be assigned roles from which they inherit permissions? How can a stored procedure base its access decisions on roles that the invoking user has activated?

***8. Should a user be allowed to take on multiple roles in a single session and if so, how?***

Some systems allow a user to simultaneously take on multiple roles in a session, while others allow the user to assume only one role at a time. If multiple simultaneous roles are allowed, some systems turn on all roles of the user while others allow the user to select which roles are turned on in a particular session. (There is an analogous situation with respect to groups in operating systems.) Systems implementations tend to hard-wire one of these alternatives. Sometimes a limited choice is provided between these options which must be selected when the system is installed. It was argued that systems should permit flexibility in this matter and allow constraints to be imposed as desired by the system owners.

***9. Should a user be allowed to take on multiple simultaneous sessions?***

Some systems allow a user to establish multiple simultaneous sessions, others do not. Multiple sessions are useful in modern computing systems where each session can be mapped to a different window on the user's display. From an access-control viewpoint, multiple sessions with different roles activated by the same user in different sessions support the principle of least privilege. As for question 8, it was felt that systems should be flexible and allow constraints to be imposed if appropriate.

***10. How should roles be delegated?***

A user may choose to delegate one or more roles to another user. This might be for a limited period of time, such as a vacation, or under specified circumstances, such as when the former user is unavailable. Information systems should provide for flexibility in such matters otherwise users will perceive security to be a hindrance and bypass it. There may also be a need for a program to delegate roles to another program in order to obtain appropriate services on behalf of a user.

***11. How do we enforce RBAC in information systems involving multiple organizations?***

There are numerous issues that need to be addressed in this context. Most of the discussion at the workshop focussed on the context of a single system. Inter-organizational issues will need to be considered as RBAC matures.

***12. What is the relationship between RBAC and users, principals, subjects, and automated agents (all of which can potentially be members of a role)?***

The access control literature distinguishes the notion of a user from that of a subject. An individual human user may be manifested in the system as multiple subjects each having different permissions. For example, a top-secret user may have a top-secret subject as well as an unclassified subject acting on the user's behalf. Similarly, an individual user may have different subjects depending upon the project the user is working on in different sessions. The term principal has been used in access control to include users and other intelligent agents. Can roles be assigned to each of these access control entities? Do we need to distinguish human users from automated agents for the purpose of RBAC? How do we ensure that an individual human being does not appear as multiple users (so as to enforce strict separation of privileges)?

***13. How do we use role classes, object classes, object instances, and role affiliations in designing effective RBAC?***

This question addresses the issue of role engineering, that is, how do we define an appropriate set of roles for an organization. For example, the concept of a bank teller is a generic one that applies to many branches of a bank. Bank teller could be a role class and a role affiliation could be the assignment of a teller to a particular branch. Similarly, a particular kind of loan could be an object class with various instances corresponding to individual loan accounts.

***14. Should RBAC encompass temporal and other dependencies between authorizations?***

Temporal dependencies between authorizations arise when one step must be completed before the next can begin. For example, an employment verification is required prior to approval of a loan. Certain roles are authorized to perform employment verification and others to perform the loan approval. Nevertheless, for each loan these two steps must occur in sequence. Temporal dependencies of this nature are a common feature in business procedures. Are such dependencies part of RBAC, or are they outside the scope of RBAC? How far can we or should we stretch the concepts of RBAC to include such dependencies?