The NIST Model for Role-Based Access Control: Towards A Unified Standard

Ravi Sandhu[*], *David Ferraiolo*[†] *and Richard Kuhn*[†]

Abstract

This paper describes a unified model for role-based access control (RBAC). RBAC is a proven technology for large-scale authorization. However, lack of a standard model results in uncertainty and confusion about its utility and meaning. The NIST model seeks to resolve this situation by unifying ideas from prior RBAC models, commercial products and research prototypes. It is intended to serve as a foundation for developing future standards. RBAC is a rich and open-ended technology which is evolving as users, researchers and vendors gain experience with it. The NIST model focuses on those aspects of RBAC for which consensus is available. It is organized into four levels of increasing functional capabilities called flat RBAC, hierarchical RBAC, constrained RBAC and symmetric RBAC. These levels are cumulative and each adds exactly one new requirement. An alternate approach comprising flat and hierarchical RBAC in an ordered sequence and two unordered features—constraints and symmetry—is also presented. The paper furthermore identifies important attributes of RBAC not included in the NIST model. Some are not suitable for inclusion in a consensus document. Others require further work and agreement before standardization is feasible.

[*]Laboratory for Information Security Technology (LIST), George Mason Univ., `sandhu@gmu.edu`, `www.list.gmu.edu`. The work of Ravi Sandhu is partially supported by NIST and by NSF.

[†]Information Technology Lab., National Institute of Standards and Technology (NIST), `dferriaolo@nist.gov`, `kuhn@nist.gov`, `www.itl.nist.gov`

1 INTRODUCTION

The paper proposes a standard reference model for role-based access control (RBAC). RBAC is a technology that is both new and old. Although rigorous RBAC models have only recently appeared, the basic concept of roles has been used for decades as a means of managing privileges. The lack of standards for RBAC has led to roles being implemented in different ways, impeding the advance of RBAC technology. The term RBAC itself does not have a generally accepted meaning, and is used in different ways by different vendors and users. The goal of this paper is to provide a standard in this arena.

RBAC provides a valuable level of abstraction to promote security administration at a business enterprise level rather than at the user identity level. The basic role concept is simple: establish permissions based on the functional roles in the enterprise, and then appropriately assign users to a role or set of roles. With RBAC, access decisions are based on the roles individual users have as part of an enterprise. Roles could represent the tasks, responsibilities, and qualifications associated with an enterprise. Because the roles within an enterprise are relatively persistent with respect to user turnover and task re-assignment, RBAC provides a powerful mechanism for reducing the complexity, cost and potential for error in assigning user permissions within the enterprise. Because roles within an enterprise typically have overlapping permissions, RBAC models often include features to establish role hierarchies, where a given role can include all permissions of another role.

RBAC allows for specification and enforcement of a variety of protection policies which can be tailored on an enterprise-by-enterprise basis. The policies enforced within a particular system are the result of the configuration of various components of RBAC. This approach to access control and authorization management is a dramatic departure from existing access control standards—such as classical discretionary and mandatory access control—where policy is essentially "hard-wired" into the access control model.[1] Because permissions are organized into enterprise functions through roles, conflict of interest relationships are more evident than if dealing with permissions on an individual basis. As such, many RBAC models support the establishment of separation of duty constraints among roles. This provides administrators with enhanced capabilities to specify and enforce enterprise policy as compared to existing access control standards.

Because of customer demand for RBAC, vendors have incorporated RBAC features into their database, system management, and operating system products. These development efforts continue without any general agreement as to what actually constitutes RBAC features. As an attempt at rigorously defining RBAC features, a number of RBAC models have been proposed and implemented [FK92, FCK95, FBK99, Gui95, NO99, RS98, SCFY96, San98b, San98a, TDH92]. These models have been independently proposed without any attempt at standardizing salient RBAC features. As such, RBAC remains an amorphous concept. One means to further the development and use of RBAC technology is to develop standards. The NIST RBAC model is a first step in this direction.

RBAC is a rich and open-ended concept which ranges from very simple at one extreme to fairly complex and sophisticated at the other. It has been recognized that a single definitive model for RBAC is therefore unrealistic. Such a model would either include or exclude too much, and would represent one point along a spectrum of choices. The NIST RBAC model is consequently organized in a four step sequence of increasing functional capabilities given below. These levels are cumulative in that each includes the requirements of the previous ones in the sequence.[2]

- **Flat RBAC**
- **Hierarchical RBAC**
- **Constrained RBAC**
- **Symmetric RBAC**

The rest of this paper is organized as follows. Section 2 gives an overview of the NIST RBAC model and its four levels. Sections 3 through 6 describe each of the four levels in sequence, along with rationale for the features packaged in each level. The NIST model focuses on those aspects of RBAC for which consensus is available. Section 7 discusses important attributes of RBAC that are not included in the NIST model. Some of these are not suitable for inclusion in a consensus document. Others require further work and agreement in the RBAC community before their inclusion is warranted. Section 8 concludes the document.

2 MODEL OVERVIEW

This section provides an overview of the NIST RBAC model as summarized in table 1. A rationale for each of the four levels of the model is also given. Readers familiar with RBAC concepts and terminology should be able to understand the model largely by reading this section. Readers relatively new to RBAC can skim this section and revisit it after reading the description of the four levels of the model in the following sections.

The NIST RBAC model is organized in a four step sequence of increasing functional capabilities given below. These levels are cumulative in that each includes the requirements of the previous ones in the sequence. Each level adds exactly one new requirement. Rationale for specifying this sequence, and for the choice of added features at each level, is given below as each level is described. Additional rationale is given in subsequent sections that discuss each level in turn.

2.1 Flat RBAC

Flat RBAC embodies the essential aspects of RBAC. The basic concept of RBAC is that users are as-

[1] A convincing testimony to the flexibility of RBAC is its ability to enforce mandatory and discretionary access controls [OSM00].

[2] An alternate approach is presented in Appendix A. This alternative recognizes Flat RBAC and Hierarchical RBAC as an ordered sequence but treats Constrained and Symmetric RBAC features as independent unordered requirements. At this point we feel that the alternate approach is preferable because it doe not force an ordering amongst features that can be independently implemented. We have refrained from changing the main body of the paper since it has previously been circulated for comments.

signed to roles, permissions are assigned to roles and users acquire permissions by being members of roles. The NIST RBAC model requires that user-role and permission-role assignment can be many-to-many. Thus the same user can be assigned to many roles and a single role can have many users. Similarly, for permissions. Flat RBAC has a requirement for user-role review whereby the roles assigned to a specific user can be determined as well as users assigned to a specific role. (A similar requirement for permission-role review is imposed in symmetric RBAC.) Finally, flat RBAC requires that users can simultaneously exercise permissions of multiple roles. This precludes products that restrict users to activation of only one role at a time.

Rationale. Flat RBAC captures the features of traditional group-based access control as implemented in operating systems through the current generation. As such it is a widely deployed and familiar technology. The features required of flat RBAC are obligatory for any form of RBAC and are almost obvious. The main issue in defining flat RBAC is to determine which features to exclude. The NIST RBAC model has deliberately kept a very minimal set of features in flat RBAC. In particular, these features accommodate traditional but robust group-based access control. Not every group-based mechanism qualifies because of the requirements given above.

2.2 Hierarchical RBAC

Hierarchical RBAC adds a requirement for supporting role hierarchies. A hierarchy is mathematically a partial order defining a seniority relation between roles, whereby senior roles acquire the permissions of their juniors. The NIST model recognizes two sub-levels in this respect.

- **General Hierarchical RBAC**

  In this case there is support for an arbitrary partial order to serve as the role hierarchy.

- **Restricted Hierarchical RBAC**

  Some systems may impose restrictions on the role hierarchy. Most commonly, hierarchies are limited to simple structures such as trees or inverted trees.

These sub-levels also apply to subsequent forms of RBAC as indicated in table 1.

Role hierarchies can be inheritance hierarchies (whereby activation of a role implies activation of all junior roles) or activation hierarchies (whereby there is no such implication) or both. The precise nature of a role hierarchy is left open.

Rationale. Role hierarchies in the form of arbitrary partial orders are the single most desirable feature in addition to flat RBAC. This feature has often been mentioned in the literature and is available in a number of existing products. Justification for requiring the transitive, reflexive and anti-symmetric properties of a partial order have been amply discussed in the literature [SCFY96]. There is strong consensus on this aspect. There is also strong consensus regarding the benefits of supporting arbitrary partial orders. Nevertheless there are products which support only restricted hierarchies, which provide substantially improved capabilities beyond the flat model. Hence, the recognition of two sub-levels in this context.

2.3 Constrained RBAC

Constrained RBAC adds a requirement for enforcing separation of duties (SOD). SOD is a time-honored technique for reducing the possibility of fraud and accidental damage, known and practiced long before the existence of computers. SOD spreads responsibility and authority for an action or task over multiple users, thereby raising the risk involved in committing a fraudulent act by requiring the involvement of more than one individual. Many different SOD requirements have been identified in the literature. These include static SOD (based on user-role assignment) and dynamic SOD (based on role activation). The exact form of SOD that is supported is left open by the NIST RBAC model.

Rationale. SOD is often mentioned as one of the driving motivations of RBAC. It is practiced routinely in organizations and should be supported by sophisticated access control products. It is introduced after hierarchies principally because existing products more often support hierarchies than SOD.

2.4 Symmetric RBAC

Symmetric RBAC adds a requirement for permission-role review similar to user-role review introduced in level 1. Thus the roles to which a particular permission is assigned can be determined as well as

Level	Name	RBAC Functional Capabilities
1	**Flat RBAC** (see Figure 1)	• users acquire permissions through roles • must support many-to-many user-role assignment • must support many-to-many permission-role assignment • must support user-role assignment review • users can use permissions of multiple roles simultaneously
2	**Hierarchical RBAC** (see Figure 2)	Flat RBAC + • must support role hierarchy (partial order) • **level 2a** requires support for arbitrary hierarchies • **level 2b** denotes support for limited hierarchies
3	**Constrained RBAC** (see Figures 6 and 7)	Hierarchical RBAC + • must enforce separation of duties (SOD) • **level 3a** requires support for arbitrary hierarchies • **level 3b** denotes support for limited hierarchies
4	**Symmetric RBAC** (see Figures 9 and 10)	Constrained RBAC + • must support permission-role review with performance effectively comparable to user-role review • **level 4a** requires support for arbitrary hierarchies • **level 4b** denotes support for limited hierarchies

Table 1: RBAC Variations Organized as Levels

permissions assigned to a specific role. The performance of permission-role review must be effectively comparable to that of user-role review.

Rationale. The requirement for permission-role review has been deferred until level 4 because it can be intrinsically difficult to implement in large-scale distributed systems. It is sometimes mentioned as an intrinsic aspect of RBAC that distinguishes RBAC from group-based access control. The NIST model takes a pragmatic approach in this regard so this feature that is hard to implement in certain environments is reserved for higher levels of RBAC.

3 FLAT RBAC

Flat RBAC is illustrated in figure 1. The features required of flat RBAC are obligatory for any form of RBAC and are almost obvious. The main issue with flat RBAC is the features that have been excluded.

Flat RBAC captures the features of traditional group-based access control as implemented in operating systems through the current generation. Some might argue that these features are not sufficient to merit the designation of RBAC. The NIST RBAC model recognizes traditional group-based access control as the first level of RBAC because it is widely deployed and familiar technology that serves well as the starting point for RBAC. This approach bypasses the usual fruitless debate about the difference between roles and groups [San97] by recognizing the underlying commonality. At the same time by allowing additional more sophisticated levels of RBAC, the NIST model recognizes that RBAC is more than just another name for traditional but robust group-based access controls.

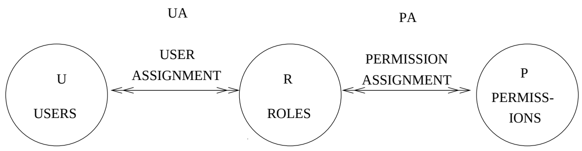

Figure 1: Flat RBAC

An important aspect of flat RBAC is the ability to support many-to-many user assignment relation. Clearly every practical system will have some limit on the number of roles to which a user can belong. At the same time products should have some scalability in this respect. Some current systems impose a very small limit, such as 16 or 32, on the number of roles a user be assigned to. Others allow larger numbers in the 100's and even 1000's. The NIST model does not legislate a quantitative minimum that must be supported in order to satisfy the flat requirement. An operational model would need to specify numerical requirements in this regard. There are other aspects of RBAC where similar scalability issues arise. Scalability is further discussed in section 7.

The requirement that users acquire permissions through roles is the essence of RBAC. Flat RBAC does not exclude other means by which users can acquire permissions such as by direct assignment to the user or by means of security labels in lattice-based access control [San93].

Figure 1 shows three sets of entities called users (U), roles (R), and permissions (P). A user in this model is a human being or other autonomous agent such as a process or a computer. 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. A permission is an approval of a particular mode of access to one or more objects in the system. The terms authorization, access right and privilege are also used in the literature to denote a permission. Permissions are always positive and confer the ability to the holder of the permission to perform some action(s) in the system.[3] The nature of a permission depends greatly on the implementation details of a system and the kind of system that it is. A general model for access control must therefore treat permissions as uninterpreted symbols to some extent. The exact nature of permissions in a system is left open by flat RBAC.

Flat RBAC requires that user-role assignment (UA) and permission-role assignment (PA) are many-to-many relations. This is an essential aspect of RBAC. The concept of a session is not explicitly a part of flat RBAC. A session corresponds to a particular occasion when a user signs on to the system to carry out some activity. The semantics of a session vary widely from system to system. In some cases all roles of a user (as assigned in the UA relation) are activated in every session of the user. In other cases the user is given a choice to activate and deactivate roles in a given session at the user's discretion. This enables the user to activate sessions with roles appropriate to the task that the user is pursuing. In particular powerful roles can be kept dormant until they are needed to provide an element of least privilege and safety. Some systems limit users to activation of only a single role in a session. The NIST model does not require support for sessions with discretionary role activation. It does require the ability to activate multiple roles simultaneously and in a single session. This precludes products that limit users to activation of a single role in a session. This feature is considered overly restrictive.

Flat RBAC requires support for user-role review whereby it can be efficiently determined which roles a given user belongs to and which users a given role is assigned to. It is often felt that a similar requirement for permission-role review is an intrinsic part of RBAC. Permission-role review enables efficient answers to questions about which permissions are assigned to a role and which roles a permission is assigned to. Some have argued that support for permission-role review is the essential feature that distinguishes roles from groups [San97]. In the NIST model requirement for permission-role review is deferred until level 4 in recognition of its intrinsic difficulty in large-scale distributed systems.

The flat RBAC model leaves open many important issues of RBAC that must be addressed in an implementation. There are no scalability requirements on the numbers of roles, users, permissions, etc., that should supported. The nature of permissions and support for discretionary role activation is not fully specified. The behavior of revocation is not specified. Revocation can occur when a user is removed from a role or a permission is removed from a role. How quickly the revocation actually takes place, particularly with respect to activity which is already under way, is left unspecified. In some ways this is an assurance issue. The important issue of role administration is not specified. Role administration is concerned with who gets to assign users to roles and permissions to roles. There are two reasons why these features are not specified in flat RBAC. In some cases it is not appropriate to legislate the details of a particular feature in a standard model. Since many choices are available it is upto vendors and the market to determine which combinations turn out to be most useful. In other

[3]Flat RBAC does not rule out the use of so-called negative permissions which deny access. In general, the NIST model prescribes features required at a particular level of RBAC, but vendors are free to include additional features as they see fit.

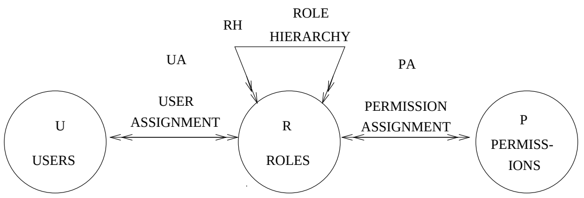

Figure 2: Hierarchical RBAC

cases there is insufficient consensus in the community to justify making a specific choice as part of a standard. These issues are further discussed in section 7.

4 HIERARCHICAL RBAC

Hierarchical RBAC is illustrated in figure 2. It differs from figure 1 only in introduction of the role hierarchy relation RH. Role hierarchies are often included whenever roles are discussed. They are also commonly implemented in systems that provide roles.

Role hierarchies are a natural means for structuring roles to reflect an organization's lines of authority and responsibility. Examples of role hierarchies are shown in figure 3. Mathematically, these hierarchies are partial orders. A partial order is a reflexive, transitive and anti-symmetric relation. By convention more powerful (or senior) roles are shown toward the top of role-hierarchy diagrams, and less powerful (or junior) roles toward the bottom. Justification for requiring the transitive, reflexive and anti-symmetric properties of a partial order have been amply discussed in the literature [SCFY96]. There is strong consensus on this aspect. There is also strong consensus regarding the benefits of supporting arbitrary partial orders. Nevertheless there are products which support only limited hierarchies, but nevertheless provide substantially improved capabilities beyond a flat model. Hence, the recognition of two sub-levels in this context as follows.

- **General Hierarchical RBAC**

  In this case there is support for an arbitrary partial order to serve as the role hierarchy.

- **Limited Hierarchical RBAC**

  If any restriction is imposed on the structure of the role hierarchy then we are in this case. Most commonly, hierarchies are limited to simple structures such as trees or inverted trees.

4.1 Limited vs. General Hierarchies

Figure 3(a) shows an inverted tree hierarchy that might exist in a hypothetical engineering department. In these diagrams senior roles are shown towards the top with edges connecting them to junior roles. Transitive edges, such as from PE1 to ED are omitted to avoid clutter. There is a junior-most role ED to which all employees in the department belong. Senior to this role are roles for two projects within the department, project 1 on the left and project 2 on the right. Each project has an engineer role and, senior to it, production and quality engineer roles. An inverted tree facilitates sharing of resources. Resources made available to the ED role are also available to senior roles. However, an inverted tree does not allow aggregation of resources from more than one role.

Figure 3(b) shows a tree hierarchy in which senior roles aggregate the permissions of junior roles. Thus PL1 acquires the permissions of PE1 and QE1, and may have additional permissions of its own. Trees are good for aggregation but do not support sharing. In this hierarchy there can be no sharing of resources between the project 1 roles on the left and project 2 roles on the right.

Figure 3(c) shows a general hierarchy that facilitates both sharing and aggregation. Within the engineering department there is a junior-most role ED and senior-most role DIR. In between there are roles for two projects. Each project has a senior-most project lead role (PL1 and PL2) and a junior-most engineer role (E1 and E2). In between each project has two incomparable roles, production engineer (PE1 and PE2) and quality engineer (QE1 and QE2). This structure can, of course, be extended to dozens and even hundreds of projects within the engineering department. Moreover, each project could have a different structure for its roles. The example can also be extended to multiple departments with different structure and policies applied to each department. Practical hierarchies will typically have an irregular structure rather than the highly symmetrical construction of this example.

We emphasize there is no requirement that there be a seniormost role such as DIR in this example. Similarly, there is no requirement that there be a juniormost role such as ED. For example, the

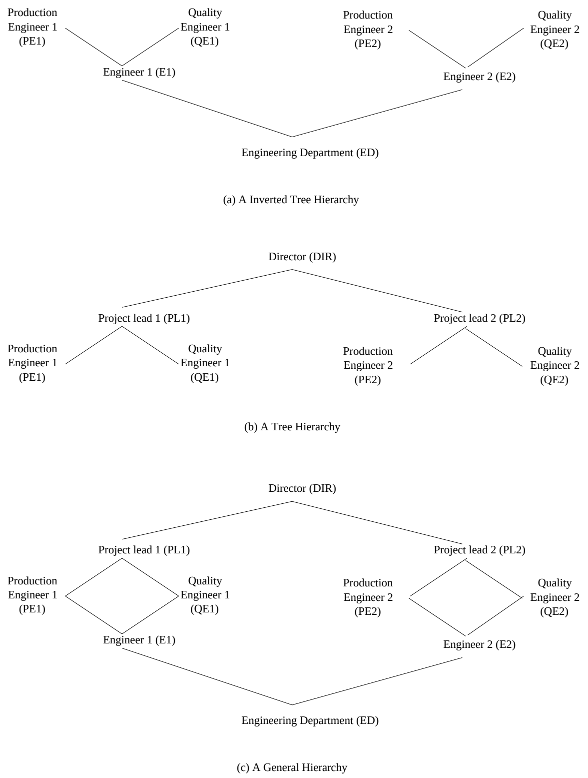

Figure 3: Example Role Hierarchies

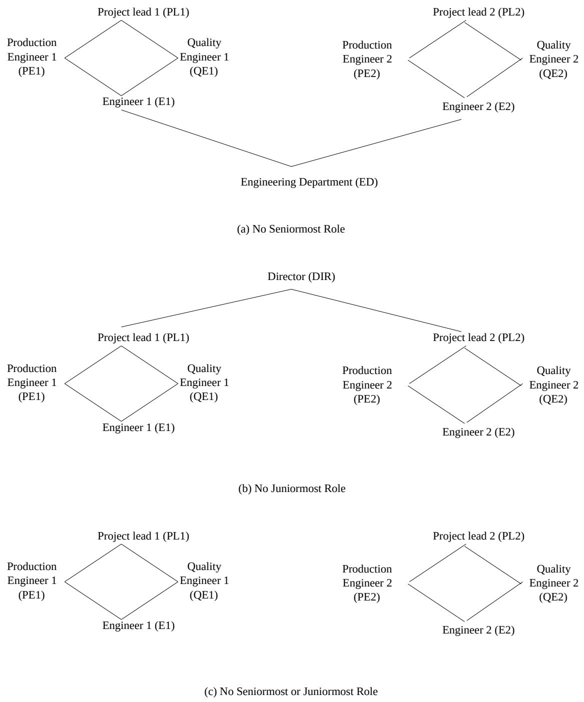

Figure 4: Example Role Hierarchies Without Seniormost or Juniormost Roles

hierarchies of figure 4 are all acceptable. The design of a suitable hierarchy is a matter of policy. The requirement is to support general hierarchies in level 2a and limited ones in level 2b.

4.2 Limited Inheritance

Senior roles such as DIR in figure 3(c) are often considered dangerous because they aggregate too much power. Even if users in these roles are very trustworthy they have the potential to make major mistakes as well as being susceptible to malicious software. It is possible to limit inheritance in role hierarchies as illustrated in figure 5. Figure 5(a) shows that the Project Supervisor role inherits all permissions of the project. Figure 5(b) on the other hand allows Test Engineers to have permissions in the Test Engineer' role that are not inherited by Project Supervisor. Test engineers are assigned to the Test Engineer' role whereas the Test Engineer role is simply a placeholder for those permissions of the Test Engineer' role that need to be inherited upwards. Roles such as Test Engineer' are called private roles [SCFY96]. A similar situation holds with respect to Programmer' role.

4.3 Inheritance vs. Activation Hierarchies

There are two distinct interpretations of a role hierarchy that have been discussed in the literature. In one interpretation members of a senior role in the hierarchy are regarded as inheriting permissions from juniors. This is called the permission-inheritance interpretation and the hierarchy is called an inheritance hierarchy. Interpreting figure 3(c) as a inheritance hierarchy, when role PL1 is activated the permissions assigned to PL1, PE1, QE1, E1, ED and E are all available for use. In the alternate interpretation, activation of a senior role does not automatically activate permissions of junior roles. This is called the activation interpretation and the hierarchy is called an activation hierarchy. In this case activation of role PL1 does not activate permissions of the junior roles. Each junior role must be explicitly activated to enable its permissions in a session. It is possible to have both interpretations simultaneously applied. In such cases the activation hierarchy may extend the inheritance hierarchy or be separate and independent of it [San98a]. The NIST model leaves open the exact meaning of role hierarchies since multiple interpretations are possible.

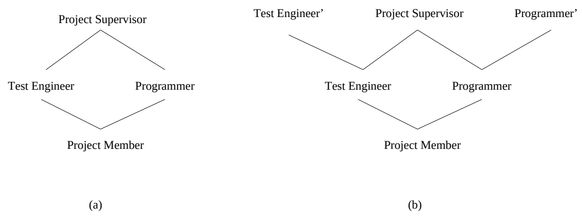

Figure 5: Example of Limited Inheritance

5 CONSTRAINED RBAC

Constrained RBAC, shown in Figures 6 and 7, adds constraints to the hierarchical RBAC model. Constraints may be associated with the user-role assignment (static, Figure 6), or with the activation of roles within user sessions (dynamic, Figure 7). Separation requirements are used to enforce conflict of interest policies that organizations may employ to prevent users from exceeding a reasonable level of authority for their positions.

Separation of duty refers to the partitioning of tasks and associated privileges among different roles so as to prevent a single user from garnering too much authority. The motivation is to ensure that fraud and major errors cannot occur without deliberate collusion of multiple users to this end. Within an RBAC system separation concepts are supported by the principle of least privilege.

Least privilege is the time honored administrative practice of selectively assigning privileges to users such that the user is given no more privilege than is necessary to perform his/her job function. The principle of least privilege avoids the problem of an individual having the ability to perform unnecessary and potentially harmful actions merely as a side effect of granting the ability to perform desired functions. Permissions (or privileges) are rights granted to an individual, or subject acting on behalf of a user, that enable the holder of those rights to act in the system within the bounds of those rights. Least privilege provides a rationale for where to install the separation boundaries that are to be provided by RBAC protection mechanisms.

The NIST model allows for both static and dynamic separation of duty, but leaves open which of these should be supported and exactly in what form.

5.1 Static Separation of Duty

Conflict of interest in a role based system may arise as a result of a user gaining authorization for permissions associated with conflicting roles. One means of preventing this form of conflict of interest is through static separation of duty (SSD), that is, to enforce constraints on the assignment of users to roles. This means that if a user is authorized as a member of one role, the user is prohibited from being a member of a second role. For example, a user who is authorized for the role Billing Clerk may not be authorized for the role Accounts Receivable (AR) Clerk (see Figure 8). That is, the roles Billing Clerk and Accounts Receivable Clerk are mutually exclusive. The SSD policy can be centrally specified and then be uniformly imposed on specific roles.

Constraints are inherited within a role hierarchy. For example, if the role Accounts Receivable Supervisor inherits Accounts Receivable Clerk, and Accounts Receivable Clerk has an SSD relationship with Billing Clerk, then Accounts Receivable Supervisor also has an SSD relationship with Billing Clerk. Another way of thinking about this is that any instance of AR Supervisor can be treated as an instance of Accounts Receivable Clerk. Therefore, the SSD constraint that Billing Clerk has with Accounts Receivable Clerk must also apply to AR Supervisor.

Because a containing role is effectively an instance of its contained roles, no SSD relationship can exist between them. In the previous example, it would not make sense to have an SSD relationship between AR Supervisor and AR Clerk, since by definition there cannot be any conflict of interest. Otherwise, a containment relationship should not have been used to inherit implicit properties that conflict with explicit properties being defined.

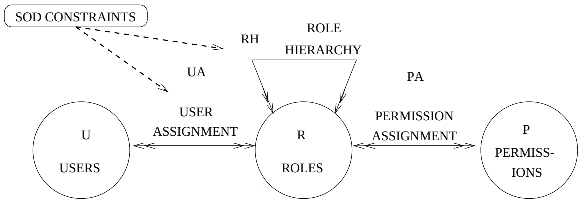

Figure 6: Constrained RBAC—Static SOD

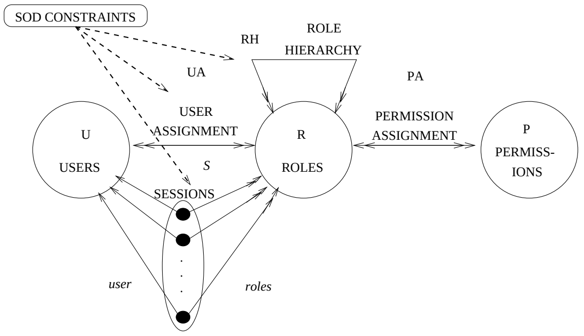

Figure 7: Constrained RBAC—Dynamic SOD

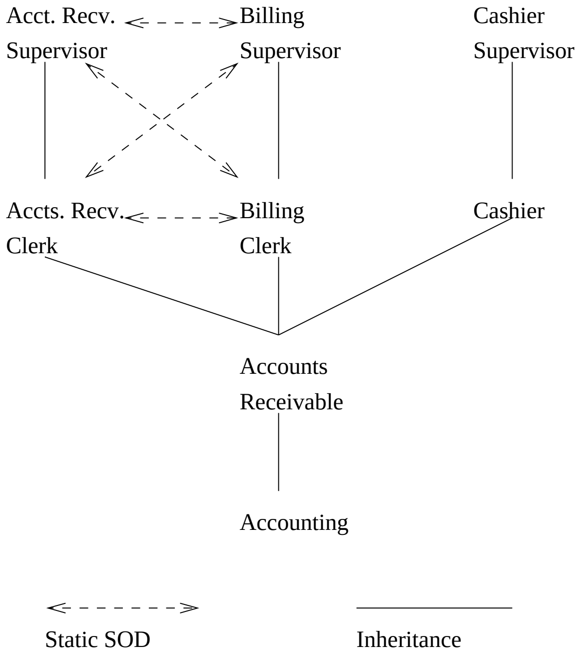

Figure 8: Constrained RBAC—SOD Example

5.2 Dynamic Separation of Duty

RBAC also provides administrators with the capability to enforce an organization specific policy of dynamic separation of duty (DSD). SSD provides an organization with the capability to address potential conflict-of-interest issues at the time a user's membership is authorized for a role. With DSD it is permissible for a user to be authorized as a member of a set of roles which do not constitute a conflict of interest when acted in independently, but produce policy concerns when allowed to be acted in simultaneously. For example, a user may be authorized for both the roles of Cashier and Cashier Supervisor, where the supervisor is allowed to acknowledge corrections to a Cashier's open cash drawer. If the individual acting in the role Cashier attempted to switch to the role Cashier Supervisor, RBAC would require the user to drop his or her Cashier role, and thereby force the closure of the cash drawer before assuming the role Cashier Supervisor. As long as the same user is not allowed to assume both of these roles at the same time, a conflict of interest situation will not arise. Although this effect could be achieved through the establishment of an SSD relationship, DSD relationships generally provide the enterprise with greater operational flexibility.

Note that unlike roles in an SSD relation, roles in a DSD relation can be hierarchically related through the containment relation. This is consistent with the DSD property of restricting simultaneous activation of roles and that of a role hierarchy as a representation of a user's implicit and explicit authorizations for a role. As such, authorization and activation can be treated as independent notions.

Some aspects of separation of duty can be implemented with flat RBAC or simple group-based mechanisms. The NIST model requires role hierarchies as a prerequisite in systems providing separation of duty because most of the benefits of RBAC are tightly integrated with the provision of hierarchies. Separation of duty mechanisms implemented without hierarchies have serious limitations on flexibility and functionality.

Many researchers have proposed separation of duty policies for RBAC [AS99, CW87, FBK99,

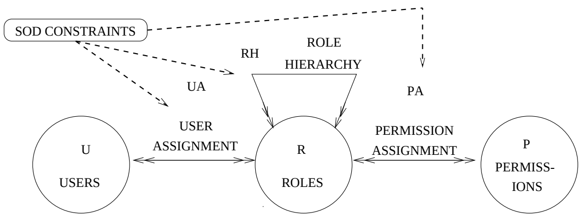

Figure 9: Symmetric RBAC—Static SOD

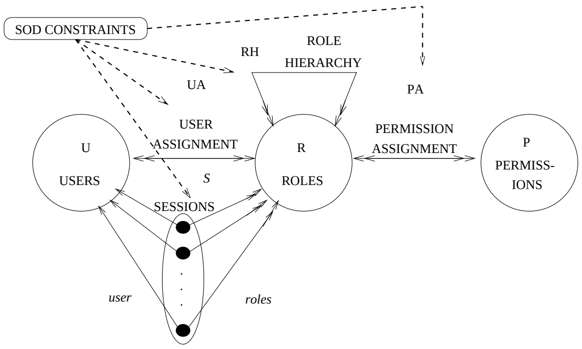

Figure 10: Symmetric RBAC—Dynamic SOD

GGF98, Kuh97, NO99, NP90, San88, SZ97]. The separation of duty features discussed here are most closely related to those proposed by Ferraiolo et al [FCK95].

6 SYMMETRIC RBAC

Maintaining appropriate and accurate permission-role assignments is an essential component of any authorization management scheme. Permission-role assignments of the past may become inappropriate as situations change within an enterprise and may become detrimental to the policy objectives of an organization. Maintaining these relations can be especially problematic in situations where user and role permissions are established over distributed administrative boundaries where the coordinated efforts of multiple administrators become a necessity.

To effectively maintain permission assignments an organization must be provided with the ability to identify and review the assignment of permissions to roles regardless of where they might reside within the organization. When maintaining permission assignments, special attention is taken to abide by the principle of least privilege. The challenge then becomes how to maintain appropriate permission assignments, among the aggregate of system objects, that correspond to the user's authorized functions or duties within the enterprise.

The need to review permission assignments can arise due to a variety of administrative situations. When a user departs from the enterprise, changes jobs or responsibilities within the enterprise, or existing permissions become obsolete great care must be applied in reviewing and selectively deleting permissions that are no longer necessary for the operations of the enterprise. In the case where the user departs from the enterprise all of the user's permissions need to be effectively revoked. One approach to this problem might be to simply delete all of the user's existing accounts within the enterprise. However, this would leave garbage in the system, that might lead to potentially damaging accesses. In the case where the employee changes responsibilities within the enterprise, the administrator must take great care in selectively revoking permissions. Deleting permissions that are necessary for the performance of the user's new responsibilities would inhibit the user's ability to effectively perform his/her job. Not deleting permissions that are no longer necessary in performing the user's new responsibilities would be a violation of the principle of least privilege, and thereby provide the potential for abuse.

Starting at Level 1 RBAC systems are required to establish and maintain many-to-many relationships among user-role and permission-role assignments. Among these relations level 1 and level 2 RBAC systems require an interface for the review of user-role assignments. Level 1 requirements include the establishment of the set of roles that are directly assigned to a user. Level 2 RBAC extends coverage of user-role review to include not only the roles that are assigned to a user but also the roles that are inherited by the roles that are assigned to the user.

Level 4 RBAC further extends these requirements to include an interface for permission-role review with respect to a defined user or role. These requirements pertain to the type of data that is returned to the administrator as a result of a review, the ability to select direct or indirect permission assignments, and for distributed systems the ability to select the target systems in which the permission review will be applied.

Level 4 or symmetric RBAC requires that the permission-role review interface provide the capability to return any one of two types of results. These results include the complete set of objects that are associated with the permissions assigned to a particular user or role, or the complete set of operation and object pairs that are associated with the permissions that are assigned to a particular user or role. As an option on this query symmetric RBAC requirements further include the ability to selectively define direct and indirect permission assignment. Direct permission assignment pertains to the set of permissions that are assigned to the user and/or to the role(s) for which the user is assigned. Indirect permission assignments pertain to the set of permissions that are included in the direct permission assignment in addition to the permissions that are assigned to the roles that are inherited by the roles assigned to the user. As a further option on a permission query symmetric RBAC requires the ability to select the target systems for which the review will be conducted.

7 OTHER RBAC ATTRIBUTES

This section discusses attributes of RBAC products that are not covered or only partially covered in the NIST RBAC model. RBAC is a rich and open-ended technology. As such it would not be appropriate to pin down all aspects of RBAC in a standard model. Some of the issues raised in this section are not suitable for standardization. On others there is lack of consensus to justify their standardization at the moment.

7.1 Scalability

Scalability is an important requirement for modern systems, especially with the tremendous growth of the Internet. Some of the current products can meet level 1 RBAC requirements but provide support for only a small number of roles such as 16 or 32. Others are more scalable providing support for 100's of roles. Clearly this is an important attribute in product selection.

The notion of scalability is multi-dimensional. In RBAC we can have scalability with respect to number of roles, number of permissions, size of role hierarchy, limits on user-role assignments, etcetera. A given product may be scalable in some dimensions but not in others.

As a general guideline we might adopt scale measure as follows.

- Small scale: 10's
- Medium scale: 100's
- Large scale: 1000's

However, a product can be small scale in, say, number of roles and large scale in number of users.

The NIST RBAC model does not incorporate a scalability attribute but this is an important issue in choosing a product.

7.2 Authentication

The NIST RBAC model does not address the issue of authentication. How are individual users authenticated and associated with the roles to which they belong. This is an important attribute which can have significant impact on the usability of a product in a specific environment. This issue is outside the scope of an access control model and is part of system architecture and mechanism considerations.

7.3 Negative permissions

The NIST model is based on positive permissions that confer the ability to do something on holders of the permission. The NIST model does not rule out the use of so-called negative permissions which deny access. Thus vendors are free to add this feature. Nevertheless vendors and users are cautioned that use of negative permissions can be very confusing, especially in presence of general hierarchies. The uses to which they are put can often be better achieved by judicious use of constraints.

7.4 Nature of permissions

The nature of permissions is not specified in the NIST RBAC model. Permissions can be fine-grained (e.g., at the level of individual objects) or coarse-grained (e.g., at the level of entire subsystems). They can be defined in terms of primitive operations such as read and write, or abstract operations, such as credit and debit. Permissions can also be customized. For example, a Physician roles can be granted permission to read medical records but only for those patients assigned to the individual doctor in question. The exact nature of permissions is determined by the nature of the product. Operating systems, database management systems, workflow systems, network management systems will all support different kinds of permissions. Standardization of permissions is beyond the scope of a general-purpose access control model.

7.5 Discretionary role activation

The NIST RBAC model does not specify the ability of a user to select which roles are activated in a particular session. The only requirement is that it should be possible to allow a user to activate multiple roles simultaneously. This rules out products that only allow use of one role at a time. Some existing products give no choice to the user and activate all the users roles in a session. Other products activate a default set of roles and leave it up to the user's discretion to add and subtract roles from this set. The NIST RBAC does not impose a requirement in this arena. It is a feature that vendors can use to distinguish their products as they see fit.

7.6 Role engineering

The NIST RBAC model does not provide guidelines for deigning roles and assigning permissions and users to roles. This activity is called role engineering. Effective use of RBAC in large-scale is strongly dependent on effective role engineering. However, this issue is outside the scope of the NIST RBAC model.

7.7 Constraints

The NIST RBAC model recognizes separation of duty (SOD) constraints. Support for SOD is required at level 3. The exact forms of that that need to be supported are not legislated in the NIST RBAC model. The NIST model distinguishes between static and dynamic SOD. However, there are many other forms of SOD that can be distinguished. For instance, concepts of role-centric, permission-centric and user-centric SOD have been recently introduced in the literature [AS99]

SOD is an example of a prohibition constraint which prevent something from happening. RBAC can also incorporate obligation constraints which require something to happen [AS99]. The concept of obligation constraints is considered too new to incorporate into a standard model at this point.

7.8 RBAC administration

The NIST RBAC model does not specify the authorization for assigning users to roles, permissions to roles and roles to roles (in a role hierarchy), and for revoking these assignments. Several models for this purpose have been proposed in the literature. Some of these are rooted in traditional discretionary access control where the owner of a roles is allowed full control over that role. Others centralize administrative authority in a single security officer role. A decentralized administrative model based on administrative roles has been recently published [SBM99]. Due to lack of consensus in this arena the NIST RBAC model does not incorporate an administrative component.

7.9 Role revocation

The semantics of role revocation is not specified in the NIST RBAC model. The main issue is the immediacy of revocation. When a user is revoked from a role what happens to existing sessions where the user has activated that role? Should the user be allowed to complete the session or should the revoked role be immediately deactivated from that session? This is a difficult issue, particularly in distributed systems where the notion of doing some action immediately is itself hard to pin down. The NIST RBAC model does not specify revocation behavior, but it is an important issue to which vendors and users of RBAC products must pay careful attention.

8 CONCLUSION

The driving motivation for RBAC is to simplify security policy administration while facilitating the definition of flexible, customized policies. Over the past nine years significant advancements have been made in both the theoretical modeling and practical implementation of RBAC features. Today RBAC is becoming expected among large users and the number of vendors that are offering RBAC features is growing rapidly. This development continues without any general agreement as to what constitutes RBAC features. This paper is the first attempt to develop an authoritative definition of core RBAC features for use in authorization management systems. Although RBAC continues to be an evolving technology, the RBAC features that were chosen to be included within this paper represent a stable and well accepted set of features described in RBAC literature and/or are beginning included within a wide breath of commercial implementations.

Standardization over a core set of RBAC features is expected to provide a multitude of benefits. These benefits include a common set of benchmarks for vendors, who are already developing RBAC mechanisms, to use in their product specifications. It will give IT consumers, who are the principal beneficiary of RBAC technology, a basis for the creation of uniform acquisition specification. In addition, this standardized RBAC model will promote the subsequent development of a standard RBAC API, that would in turn promote the development of new and innovative authorization management tools by guaranteeing interoperability and portability.

Although RBAC is often considered a single access control and authorization model, RBAC is in fact composed of a number of models each fit for a specific security management application. As such, the NIST RBAC model has been organized into four separate levels of increasing functional capability, each with a specific rationale for its deployment. The first level, flat RBAC defines features that are minimally required of all RBAC systems. flat RBAC requires that user-role and permission-role assignment to consist of a many-to-many relationship. Although this basic requirement can be achieved using a simple group mechanism, not all group mechanisms today are capable of meeting this requirement. In addition level 1 has a requirement for a user-role review feature. For a system to meet flat RBAC requirements it must provide the capability to identify the users that are assigned to any role as well as the roles that are assigned to any user. Level 2, hierarchical RBAC adds requirements for role hierarchies. Among RBAC features role hierarchies are considered to be the most beneficial from an administrative efficiency point of view and are included within a number of commercial implementations. Level 2, recognizes two types of role hierarchies - 2a, general hierarchies, to support the arbitrary partial ordering of roles to serve as the hierarchy and limited hierarchies, 2b, to allow for simpler hierarchical structures such as trees or inverted trees. Although hierarchies composed of arbitrary partial orders are more powerful and flexible, trees and inverted tree structures are the most common implementation and are included for that purpose. Level 3, constrained RBAC, adds requirements for the enforcement of separation of duties (SOD) policies. The exact form of SOD is left open by the RBAC model. Two sub-levels 3a and 3b are defined similar to levels 2a and 2b requiring support for arbitrary and limited hierarchies respectively. Level 4 further extends these requirements to include an interface for permission-role review with respect to a defined user or role. These requirements pertain to the type of data that is returned to the administrator, the ability to select direct or indirect permission assignment and for distributed systems the ability to select the target systems in which the permission review will be applied.

References

[AS99] Gail-Joon Ahn and Ravi Sandhu. The RSL99 language for role-based separation of duty constraints. In *Proceedings of 4th ACM Workshop on Role-Based Access Control*, pages 43–54, Fairfax, VA, October 28-29 1999. ACM.

[CW87] D.D. Clark and D.R. Wilson. A comparison of commercial and military computer security policies. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 184–194, Oakland, CA, May 1987.

[FBK99] David F. Ferraiolo, John F. Barkley, and D. Richard Kuhn. A role based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information and System Security*, 2(1), February 1999.

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

[FK92] David Ferraiolo and Richard Kuhn. Role-based access controls. In *Proceedings of 15th NIST-NCSC National Computer Security Conference*, pages

554–563, Baltimore, MD, October 13-16 1992.

[GGF98] Virgil D. Gligor, Serban I. Gavrila, and David Ferraiolo. On the formal definition of separation-of-duty policies and their composition. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, pages 172–183, Oakland, CA, May 1998.

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

[Kuh97] D. Richard Kuhn. Mutual exclusion of roles as a means of implementing separation of duty in role-based access control systems. In *Proceedings of 2nd ACM Workshop on Role-Based Access Control*, pages 23–30. ACM, Fairfax, VA, November 6-7 1997.

[NO99] Matunda Nyanchama and Sylvia Osborn. The role graph model and conflict of interest. *ACM Transactions on Information and System Security*, 2(1), February 1999.

[NP90] M.N. Nash and K.R. Poland. Some conundrums concerning separation of duty. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 201–207, Oakland, CA, May 1990.

[OSM00] 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*, 3(2), May 2000.

[RS98] Chandramouli Ramaswamy and Ravi Sandhu. Role-based access control features in commercial database management systems. In *Proceedings of 21st NIST-NCSC National Information Systems Security Conference*, pages 503–511, Arlington, VA, October 5-8 1998.

[San88] Ravi Sandhu. Transaction control expressions for separation of duties. In *Proceedings of 4th Annual Computer Security Application Conference*, pages 282–286, Orlando, FL, December 1988.

[San93] Ravi Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, November 1993.

[San97] Ravi Sandhu. Roles versus groups. In *Proceedings of the 1st ACM Workshop on Role-Based Access Control*. ACM, 1997.

[San98a] Ravi Sandhu. Role activation hierarchies. In *Proceedings of 3rd ACM Workshop on Role-Based Access Control*, pages 33–40, Fairfax, VA, October 22-23 1998. ACM.

[San98b] Ravi Sandhu. Role-based access control. In Zelkowitz, editor, *Advances in Computers, Volume: 46*. Academic Press, 1998.

[SBM99] Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and System Security*, 2(1):105–135, February 1999.

[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.

[SZ97] R. Simon and M. Zurko. Separation of duty in role-based environments. In *Proceedings of 10th IEEE Computer Security Foundations Workshop*, pages 183–194, Rockport, Mass., June 1997.

[TDH92] T.C. Ting, S.A. Demurjian, and M.Y. Hu. Requirements, capabilities, and functionalities of user-role based security for an object-oriented design model. In C.E Landwehr and S. Jajodia, editors, *Database Security V: Status and Prospects*. North-Holland, 1992.

Appendix A. The Alternate Model

In this appendix we present an alternate model which recognizes that the four step sequence presented in the main body of the paper may not always be applied in practice. Features of later steps in that sequence may be adopted prior to adopting features of earlier steps. Based on comments received so far and our own reflections we feel this alternate model presents a superior approach. It reflects the initial approach we had taken but then abandoned in an attempt to formulate a strict ordering of RBAC levels. We have thus come around full circle on this issue. For the moment we have refrained from changing the main body of the paper since it has previously been circulated for comments. It would be confusing at the workshop to change the paper so drastically.

The alternate model does not require any new concepts as such. It is essentially a restructuring of what we have already presented. The alternate model is summarized below.

RBAC Functional Capabilities		
Role Structure	**Role Constraints**	**Role Symmetry**
Flat	No	No
Limited Hierarchy	Yes	Yes
General Hierarchy		

Table 2: The Alternate Model

Although not explicitly shown above the requirements of Flat RBAC as shown in Table 1 are required in all cases. In addition the three RBAC functional capabilities of role structure, constraints and symmetry are recognized. Role structure can be Flat, Limited Hierarchy or General Hierarchy in sequence of increasing power. Role constraints and role symmetry have a simple No/Yes choice with Yes being the stronger one.

Altogether the alternate model allows for 12 possibilities. In the original model only 7 of these 12 are recognized. There is a straightforward mapping between the levels of the original model and the RBAC functional capabilities of the alternate model as shown below. In the tables below these capabilites are presented in the same sequence as in the table above.

Level	**RBAC Functional Capabilities**		
1	Flat	No	No
2a	General Hierarchy	No	No
2b	Limited Hierarchy	No	No
3a	General Hierarchy	Yes	No
3b	Limited Hierarchy	Yes	No
4a	General Hierarchy	Yes	Yes
4b	Limited Hierarchy	Yes	Yes

Table 3: Relationship Between the Original and Alternate Models

The alternate model also allows the following 5 possibilities.

RBAC Functional Capabilities		
Flat	Yes	No
Flat	No	Yes
Flat	Yes	Yes
General Hierarchy	No	Yes
Limited Hierarchy	No	Yes

Table 4: Additional Possibilities in the Alternate Model

These 5 possibilities as such are not excluded in the original model but at the same time are not given any extra value. In particular there is only one Flat RBAC in the original model but in the alternate model Flat RBAC can occur in four variations.

The alternate model is also more extensible. As consensus develops we can move from a simple No/Yes scale to a more discriminating scale for role constraints and role symmetry. Moreover additional RBAC functional capabilities can be recognized. Some of these are mentioned in Section while others may emerge later.

The alternate model is more complex than the original but captures the reality of RBAC more accurately.

Appendix B. What Next?

Paths Toward a Voluntary Industry Consensus Standard for RBAC

by Richard Kuhn and David Ferraiolo

From the level of interest in Role-Based Access Control shown by the research community and in the marketplace, it is clear that RBAC is becoming an integral part of the global information infrastructure. With the growing number of RBAC products on the market has come interest in a consensus specification, either as a de facto or de jure standard. Such a standard would provide both vendors and buyers with better means to compare RBAC products, and to specify the features they offer. As employees of the Computer Security Division (CSD) of the National Institute of Standards and Technology (NIST) we are excited at the prospect of working with Professor Ravi Sandhu, of George Mason University and other interested parties towards a voluntary industry consensus standard for RBAC.

NIST is the United States' measurement and test laboratory, whose mission is to strengthen the U.S. economy and improve the quality of life by working with industry, universities, and government organizations to develop and apply technology, measurements, and standards for information technology. NIST carries out this mission by working with industry and government partners to develop and demonstrate tests, test methods, reference implementations and data, proof of concept implementations, and other infrastructure technologies that are usable, secure, scalable, and interoperable. Within the area of computer security, NIST has specific statutory responsibilities for developing security standards and technology to assist Federal agencies in the protection of sensitive unclassified systems. This is in addition to NIST's mission of strengthening the U.S. economy – including improving the competitiveness of America's information technology (IT) through performing security research, standards development and providing sound guidance. The solutions that NIST develops are made available to both public and private users and are often complemented by an active Technology Transfer program.

NIST has long been active in developing Federal standards and working in cooperation with private sector standards organizations and universities in preparing our nation for the future. For example, we are leading a public process to develop the Advanced Encryption Standard (AES), and developing a government wide Public Key Infrastructure standard to serve 21st century security needs for both government and industry.

We are particularly excited about this proposed RBAC consensus standard because the work would be consistent with and build on our own research efforts in developing and transferring to industry Role-Based Access Control and other authorization management technologies. Today RBAC is becoming increasingly popular within commercially available Database Management, Enterprise Management, and Network Operating Systems. A number of benefits could result from a uniform, widely accepted model for RBAC. Standardization is one means of establishing such a model. Although advancements have been made at the platform and network operating system level, access control mechanisms do not interoperate with one another, are ineffective at serving the policy need of all sectors, and are difficult to manage, and prone to human error.

The need for the capabilities provided by RBAC is highlighted by a growing recognition of the threat of insider attacks and the potential for these attacks to undermine the global information infrastructure. Insider attacks already account for over 70computer crime and cyberterrorism.[4] Insiders have wider access to sensitive resources, deeper knowledge of internal systems and greater opportunity to carry out their plans. The majority of insider attacks do not hinge on flawed protocols or sophisticated cryptanalysis. Instead, they take advantage of gaps in enterprise security policies. A consensus RBAC standard would represent a culmination of the research efforts of many and is meant to represent leading edge technology in addressing the most likely target of insider attacks - enterprise security policies.

Standard RBAC provides an opportunity for a common representation for access control models and policies, making it a suitable foundation for a policy coordination system. Embedding RBAC in our network and database architectures is an innovative approach to managing security policies in open environments.

[4] F. Schneider (Ed.) "Trust in Cyberspace". National Academy Press, Washington, D.C., 1999.

A variety of options are available for achieving a standard definition for RBAC, including:

National and International Standards Bodies: The International Organization for Standardization (ISO) provides mechanisms for developing consensus standards, typically through adoption of a national standard. National bodies within ISO, such as ANSI (USA), DIN (Germany), and AFNOR (France), may develop standards that are then forwarded to ISO for international adoption.

Professional Society Standards Bodies: The same path is available for standards developed through bodies such as the Institute of Electrical and Electronics Engineers. IEEE standards are typically developed with the participation of interested parties from many nations. IEEE standards typically begin with a base document proposed by one or more organizations. Working groups that meet several times a year then modify the document before it is voted on by interested parties. After approval as an IEEE standard, they may be put on track for adoption as an ISO standard.

Industry Consortia: Within the information technology industry, a number of consortia have developed consensus specifications that have such widespread adoption that they have become de facto standards. Consortia specifications are developed by working groups consisting of parties from member companies. Although not recognized by official standards bodies, they are often developed more quickly and achieve the same level of market recognition as international standards.

Federal Information Processing Standards: Developed by NIST, FIPS are U.S. government standards that are often adopted widely within industry. The Data Encryption Standard (DES) is probably the most widely recognized example. FIPS are proposed by NIST, then progress through a comment and revision before approval.