Proc. of 17th NIST-NCSC National Computer Security Conference, Baltimore, MD, October 11-14, 1994, pages 138-149

A THREE TIER ARCHITECTURE FOR ROLE-BASED ACCESS CONTROL

Ravi S. Sandhu and Hal Feinstein*

SETA Corporation
6858 Old Dominion Road, Suite 200
McLean, VA 22101

This paper presents a reference architecture (or conceptual framework) for the specification and enforcement of role-based access control (RBAC). The architecture has three tiers in loose analogy to the well-known ANSI/SPARC architecture for database systems. (Although we take our inspiration from the database domain, we emphasize that our proposed RBAC architecture is germane to applications and systems in general and is not limited to databases per se.) The three tiers of the reference architecture consist of (i) multiple external or user views concerned with the utilization of RBAC in a specific context within the organization, (ii) a single conceptual or community view which amalgamates diverse external views into a consistent and unified composite suitable for overall security administration, and (iii) multiple internal or implementation views concerned with enforcement of RBAC in various subsystems of the enterprise information system. This paper discusses these three tiers and their interrelationships. We demonstrate the usefulness of this conceptual approach, and identify issues which need further research to make this framework a reality.

1 INTRODUCTION

Role-based access control (RBAC) is an idea whose time has come. A consensus has developed in recent years that the traditional discretionary and mandatory access controls (DAC and MAC, respectively) embodied in DoD's landmark Orange Book [Dep85] are inadequate for the information security needs of many commercial and civilian Government organizations (as well as single-level military systems, for that matter). Orange Book DAC is too weak for effective control of information assets, whereas Orange Book MAC is focused on US policy for confidentiality of classified information. RBAC has therefore emerged as a third form of access control to fill this urgent need.

Although RBAC is perceived to be a good match for the information security needs of a wide spectrum of organizations (which are not being currently served by Orange Book DAC and MAC) there remains a lack of consensus about exactly what RBAC means. For example, participants at the recent Federal Criteria Workshop felt that while "RBACs were needed in the commercial/civilian sector," at the same time "roles are a new concept and not yet well understood" [Nat93b].

It is beyond the scope of this paper to give a complete definition of RBAC, let alone one on which wide consensus has been achieved. Such an attempt would be premature. Our purpose here is to present a conceptual framework, or reference architecture, for specifying and enforcing RBAC. Our framework has three tiers in loose analogy to the well-known ANSI/SPARC architecture for database systems [Te78]. Although we take our inspiration from the database domain, we emphasize that our

*Ravi Sandhu is also affiliated with the Department of Information and Software Systems Engineering at George Mason University in Fairfax, VA.

proposed RBAC architecture is germane to applications and systems in general and is not limited to databases per se. Much as the ANSI/SPARC framework is useful independent of the particular data model employed, our proposed RBAC framework is useful whatever the final consensus definition of RBAC turns out to be.

Our reference architecture is motivated by two main considerations. Firstly, a number of proposals incorporating one form or another of RBAC have been published in recent years. Some of these have been incorporated in commercial products, and more such products can be expected to appear in the near future. Vendors tend to integrate RBAC facilities in products in different ways, because of the economics of integrating such features into existing product lines. Over time the emergence of standards will impose some order in this arena, but the near term is likely to display a divergence of approaches. Even as standards emerge, we can expect a diversity of support for RBAC due to the longevity of legacy systems.

Secondly, in large organizations there will be a large number of roles and complex relationships between the roles and permissions authorized by them. In most contexts it would be appropriate to take a simplified view appropriate for the task at hand. For example, in some situations all members of a particular department can be treated as belonging to a single role; whereas in other situations more refined roles such as managers, technical staff and administrative staff need to be distinguished.

The central tier of our architecture resides in a single community view of RBAC as it applies to the entire organization in question. This community view will typically be large and complex reflecting the reality of modern organizations. The specialized context-specific views of RBAC tailored to particular applications and situations are accommodated in multiple user views that reside above the central tier. The views of RBAC embodied in different products are embodied in multiple implementation views residing below the implementation tier. Figure 2 illustrates these three tiers. The central tier serves as the focal point for mapping the external user views to the internal implementation views.

The rest of this paper is organized as follows. Section 2 briefly reviews prior work on RBAC. Section 3 presents our three-tiered reference architecture for RBAC. Section 4 discusses issues pertaining to the all important central tier. Sections 5 and 6 respectively discuss relationships between the top two tiers and the bottom two tiers of our architecture. Section 7 gives our conclusions.

2 BACKGROUND

The roots of RBAC can be traced back to the earliest access control systems. RBAC has a superficial resemblance to the long-standing use of user groups in access control systems. There are, however, two very important differences between groups and roles; as articulated by Ferraiolo and Kuhn [FK92].

Firstly, groups are essentially a discretionary mechanism whereas roles are non-discretionary. The ability to assign permissions to a group is usually discretionary (although the authority to assign members to a group is usually non-discretionary, and reserved for the security administrator). Thus, the owner of a file can decide what access a particular group has to that file. On the other hand, the allocation of permissions to a role, as well as determination of membership in a role, are both intended to be non-discretionary.* In the simplest case, these decisions are made solely by the security administrator. More generally, the security administrator can selectively delegate this

*Not all proposals for RBAC agree with this position. For example, relations in Oracle [Ora92] can be owned by individuals who the have discretionary authority regarding how to assign permissions for these relations to users and roles. In our opinion the non-discretionary aspect of roles is very important. In systems such as Oracle, it is possible to achieve a de facto non-discretionary behavior by strict control of ownership of relations which contain corporate data.

authority to other users or roles in the system (as recognized in the CS-3 profile of the Draft Federal Criteria [Nat92]).

Secondly, the nature of permissions allocated to a role is significantly different than the usual read, write, execute, etc., supported by typical Operating Systems (OSs). Ferraiolo and Kuhn define the notion of a transaction as a program (or transformation procedure) plus a set of associated data items. The operation authorized is therefore to execute the specified program on this set of data items. This very important notion allows authorization in terms of abstract operations embodied in transformation procedures. For example, the bank teller role can be allocated the authorization to execute credit and debit operations on accounts rather than to general read and write operations. This enables RBAC to address security for applications in terms of the application's operations, as opposed to generic read and write operations in a general-purpose OS.

Roles have been employed in several mainstream access control products of the 1970s and 80s, such as IBM's RACF and Computer Associates' CA-ACF2 and CA-TOP SECRET. These products typically include roles for administrative purposes. For example, RACF provides an Operator role with access to all resources but no ability to change access permissions, a Special role with ability to change permissions but no access to resources, and an Auditor role with access to audit trails (including events generated by Operator and Special, who have no access to the audit trail) [Mur93]. The use of roles for administrative purposes also appears in context of cryptographic modules [Nat93a]. Here User, Crypto-Officer and Maintenance roles are distinguished.

Recent proposals for RBAC, such as Ferraiolo and Kuhn [FK92], go beyond this traditional use of roles by providing them at the application level to control access to application data. This is an important innovation which makes RBAC a service to be used by applications. RBAC offers the opportunity to realize benefits in securing an organization's information assets, similar to the benefits of employing databases instead of files as the data repository. Instead of scattering security in application code, RBAC will consolidate security in a unified service which can be better managed while providing the flexibility and customization required by individual applications. It should be noted that access control similar to RBAC has often been embedded in application code. The point is to move this functionality out of application code into a common set of services.

Over the past five years or so, several proposals for RBAC have been published. Some of these, such as [Bal90, Ste92, Tho91], have proposed extensions to existing access control systems to incorporate roles. Commercial products, such as ORACLE [Ora92], have incorporated roles. Roles are also being considered as part of the emerging SQL3 standard [PB93]. Proposals for incorporating roles in object-oriented systems have been published [LW88, Tin88]. More recently Ferraiolo and Kuhn [FK92] of NIST have given an abstract and unifying description of the essential characteristics of RBAC. Their ideas have been incorporated in the CS-3 protection profile of the Draft Federal Criteria [Nat92]. The application of roles for enforcing static and dynamic separation of duties has also been recognized [CW87, San88b, San91].

The formulations of RBAC mentioned above have been motivated by different considerations. Not surprisingly they differ in important aspects. At present there is no unified model with respect to which these different formulations can be viewed as special cases. Development of such a model, and a taxonomy of its special cases, would be a significant contribution to this area. This task is beyond the scope of this paper. Our concerns here are independent of the unified RBAC model that may eventually emerge.

3 THE THREE TIER FRAMEWORK

In the late 1970s, an ANSI/SPARC study group published a report [Te78] which has had an enduring impact on database systems. This report described a three-tier "architecture" for a database, consisting of:

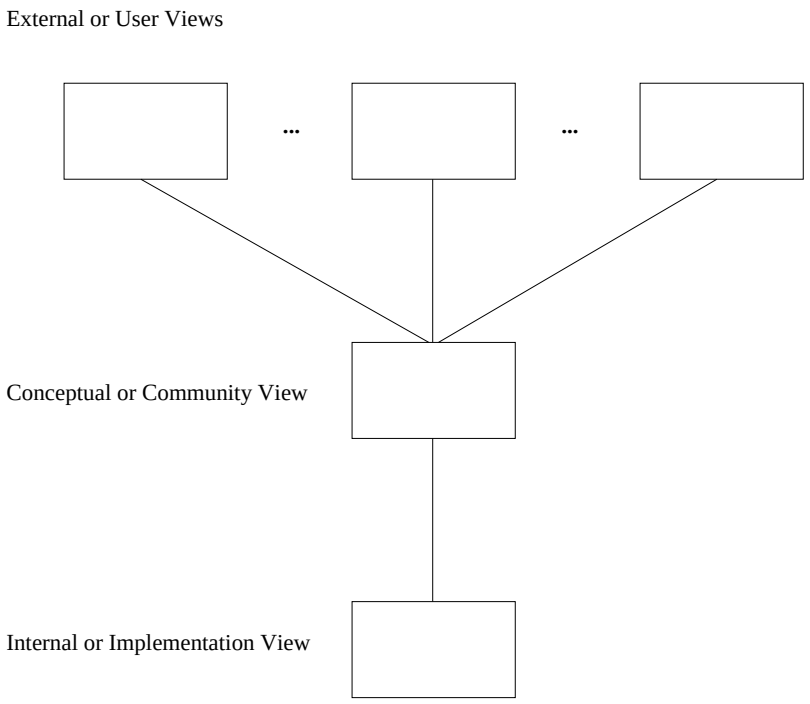

Figure 1: ANSI/SPARC Database Architecture

1. the external or user view which is concerned with the way data is viewed by end users,
2. the conceptual or community view which amalgamates diverse external views into a consistent and unified composite, and
3. the internal or implementation view which is concerned with the way that data is actually stored.

This database architecture is shown in figure 1.

Note that there are multiple external views, but only a single conceptual and a single internal view. This three-tier approach to database systems has stood the test of time, and is remarkably independent of the particular data model being used.

We believe a similar approach is suitable for developing a common framework or reference architecture for RBAC. RBAC is concerned with the meaning and control of access control data (i.e., data used to control access to the actual data of the organization). In other words we are concerned with a special purpose database system. It is therefore sensible to adapt the approach used for general-purpose database systems. However, there is one significant difference. In database systems, it is intended that the implementation will eventually be on a particular database management platform. Consequently, the internal or implementation view is closely tied to the particular platform that is selected. With RBAC we do not have the luxury of assuming a homogeneous implementation environment. Instead we must confront the reality of heterogeneous implementations up front. This leads us to modify the three-tier ANSI/SPARC architecture by introducing multiple internal views, corresponding to different platforms on which the implementation is done. This RBAC reference architecture is shown in figure 2.

Our three-tiered approach to RBAC therefore consists of

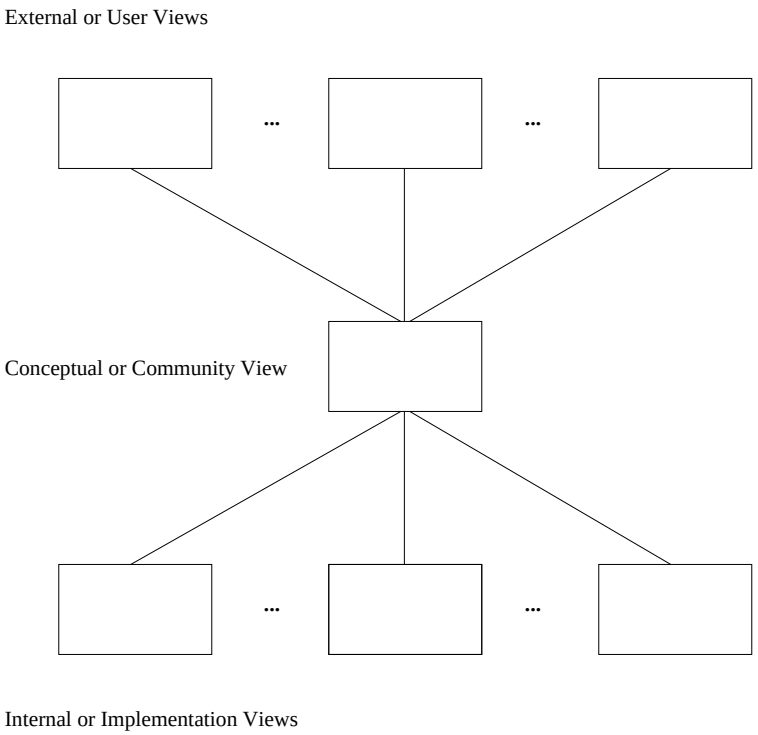

Figure 2: A Three Tier Architecture for RBAC

1. multiple external views,
2. a single conceptual view, and
3. multiple implementation views

Next, let us consider the appropriate model for each of these tiers. We again turn to the ANSI/SPARC architecture for inspiration. There is a conspicuous difference between the models used at the implementation and conceptual tiers. We expect a similar difference in our RBAC reference architecture. Why is this so? We expect the model used at the conceptual level to have richer constructs and primitives, because it is intended to express a composite system-wide view of RBAC. Practical considerations will inevitably dictate that not all these features can be directly supported in an implementation. Hence the implementation models will be simpler and less user-friendly. Moreover, we expect a range of sophistication from rather primitive mechanisms (say on a vanilla UNIX platform) at one end to very elaborate ones (say on an object-oriented database management system) at the other. Note that this viewpoint lets us accommodate legacy systems co-existing with newer ones. It should also be clear that the effort required to translate a conceptual view will be less or greater depending upon the sophistication of the implementation platform being targeted. In some cases, a translation may not even be feasible (or practical) without enhancement of the target platform.

The difference between the conceptual and external tiers is less marked. Whether or not there should be any difference is open to debate. For relational databases, both tiers are often identical and directly based on the relational data model. However, sometimes a richer model such as the entity-relationship model is used for the external view while a relational model is used at the conceptual

view. We anticipate a similar situation in the RBAC reference architecture. Based on the historical experience with the ANSI/SPARC architecture, it might well happen that initially the same RBAC model is used at both tiers, but over time richer models are developed for the external view.

In subsequent section we first discuss the all important central tier of our RBAC reference architecture. This is followed by discussion regarding the top two tiers and their relationship. Finally we discuss the relationship between the bottom two tiers.

4 THE CENTRAL TIER

The central tier of our reference architecture consists of a single community view of RBAC applicable to the entire information system and its myriad applications. This community view is the essential conceptual vehicle for effective deployment of enterprise-wide RBAC. Development of a suitable model of RBAC for this tier is an all important task, but beyond the scope of this paper. Here we discuss some issues in constructing such a model, and describe some desirable characteristics that this model should have. We should say at the outset that an RBAC model suitable for this tier must be rigorous, have a formal foundation and yet be intuitively comprehensible and useful to practitioners. This is a daunting task, but one which we feel can be accomplished in future work.

RBAC is intended to be a flexible and customizable vehicle for application security. A recent NIST study [FGL93] of "current and future information technology security needs of the commercial, civil, and military sectors" concluded that, "Each organization viewed its access control needs as unique. Access control mechanisms need to be applied on a case-by-case basis in meeting individual computer security threats." Ferraiolo and Kuhn [FK92] similarly state that, "A wide gamut of security policies and needs exist within civilian government and private organizations. An organizational meaning of security cannot be presupposed."

In order to achieve flexibility, it is important to resist imposing a particular form of RBAC in all situations. A general RBAC model must instead accommodate a variety of alternatives that can be selected on a case by case basis. At the same time it is not very useful to enumerate a long menu of alternatives as a general RBAC model. The goal of flexibility and customization must be reconciled with the need for simplicity and minimality of concepts in the model.

To illustrate this conflict, consider the question of whether or not a user can simultaneously take on more than one role. In many situations it can be argued that limiting the user to one role at any time is beneficial for purpose of least privilege. For example, the role of being an employee and the role of being a stockholder of an enterprise are two independent attributes of a user yielding different access rights which should be separately exercised by an individual. Similarly, the role of being a physician and being a patient should be regarded as mutually exclusive. At the same time, there are many situations when it is beneficial to let a user exercise multiple roles simultaneously. This is particularly so when the roles are based on competence or skill. Thus an attorney can take on the role of a specialist in, say, tax and criminal law. A single individual, cleared to both specialist roles, can then be assigned to a case requiring both kinds of attorneys. A system which insists on users taking on only one role at a time would require two individuals to process the case, or perhaps a single individual who is required to switch back and forth between roles. Requiring two individuals, where one would do, is clearly inefficient. Requiring frequent switching back and forth of roles, in this situation, is the sort of thing that gets users frustrated with security.

This example demonstrates that any system which enforces one alternative to the exclusion of the other, is going to be awkward to use when the situation at hand does not match the alternative hardwired in the system. One approach to resolving the particular conflict of this example is to recognize two kinds of roles: those that can be simultaneously held and those that cannot. More generally, one could imagine disjoint sets of roles which can be mixed under permissible combinations specified in some formal language. If one is not careful, this kind of thinking can lead to models

which are extremely general and open-ended, and thereby lose their value. Rather than customizing such a general model, it may be more useful for the practitioner to construct a model more directly applicable to the need at hand.

The issue of how many and which roles can be simultaneously exercised by a user, is but one of many such issues which need to be addressed in constructing a RBAC model. Perhaps, the most fundamental issue that needs to be addressed is what exactly is meant by a role. Ferraiolo and Kuhn [FK92] define a role as follows: "A role can be thought of as a set of transactions that a user or set of users can perform within the context of an organization. Transactions are allocated to roles by a system administrator. ... Membership in a role is also granted and revoked by a system administrator."

The question then arises as to what is a transaction. Ferraiolo and Kuhn provide two definitions. In the first definition, a transaction is defined as "a transformation procedure, plus a set of data items accessed by the transformation procedure." In this case Ferraiolo and Kuhn observe that access control is very simple, because it "does not require any checks on the user's rights to access a data object, or on the transformation procedure's right to access a data item, since the data accesses are built into the transaction." All that needs to be checked is whether or not the user is authorized to run the transaction in question (via some role). Ferraiolo and Kuhn also offer a more sophisticated definition of transaction by redefining it to "refer only to the transformation procedure, without including a binding to objects." Access control enforcement must then check 5-tuples of the form (u,r,t,o,x) to ascertain whether or not a user u in role r can access object o in mode x using transaction t (x is one of read, write, append, etc.). The need for such fine-grained access control has been supported by comments from an IRS representative at the NIST-NSA Federal Criteria Workshop [Nat93b, page 47].

So even the definition of a transaction has several important variations. The foregoing aspect concerns the nature of privileges that are associated with roles. There is also significant variation concerning the manner by which privileges and users are assigned to roles. On one hand this assignment should be non-discretionary, and perhaps done only by the system administrator. On the other hand, in large systems this will be an onerous responsibility to impose on a single individual or office. To facilitate security administration it should be possible for the system administrator to delegate pieces of this authority to other users or roles. The need for such delegation is recognized in the Commercial Security profiles of the draft Federal Criteria [Nat92], as well as in [FK92]. There is, however, a great deal of variation in how this delegation can be accomplished, especially if delegation of such administrative privileges can be further delegated. For example, the manager of a department may be given some administrative control over roles pertaining to that department. However, we would like to impose some non-discretionary controls on the manager so that, for example, the manager can delegate his authority to certain roles but not others. A general RBAC model must allow variation here without stipulating the universal use of one approach.

Another important aspect of RBAC in which there is significant variation concerns inheritance of privileges across roles. In general, roles can be composed of other roles [FK92]. To take an example from [FK92], the Intern role can be assigned to the Healer role, so that members of the Intern role automatically obtain membership in the Healer role. Taking this one step further, a Doctor role can be assigned to the Intern role. In this manner members of the Doctor role become members of the Intern role, and transitively members of the Healer role. There are significant policy issues that arise in this context. In this particular example transitive propagation of membership appears to be justified. On the other hand, transitive propagation may not always be desirable. It may also be useful to distinguish the privileges of a role that may be inherited through other roles, from privileges that are private to a role and cannot be inherited. In a truly general model we may also wish to consider denials (or negative privileges), in addition to permissions (or positive privileges). This is a useful facility, particularly when there are multiple administrative authorities in a system. The exact semantics of inheritance of privileges in such cases can become extremely murky [Lun88]. It is

also important to develop systematic methodologies for designing and maintaining such hierarchies of roles. The techniques described in [San88a] for construction of such hierarchies in the context of protection groups could be employed here.

The point of the preceding discussion is that there are many variations to be considered in a model for RBAC. The dimensions that were considered above are summarized below.

- What, and how many, roles can a user exercise simultaneously?
- What is the granularity of privileges that can be assigned to roles?
- How do privileges granted to roles interact with privileges granted to users as individual?
- How is security administration of assignment of users and privileges to roles accomplished?
- How are privileges inherited when roles are composed of other roles?

We need a common approach towards modeling RBAC wherein variations along the dimensions identified above (and possibly others which emerge).

Let us now consider the nature of an RBAC model suitable for the central tier of the RBAC reference architecture. In abstract terms any access control model has to address the following issues.

1. What is a protection state?
2. What does it mean?
3. How is it changed?

To illustrate this let us consider some classical access control models, and see how they address these issues. The most widely used model to date is perhaps the Bell-LaPadula or BLP model [BL75]. In BLP a protection state consists of a set of subjects SUB, a set of objects OBJ, a discretionary access matrix D, a current access matrix M, and a function SECURITY-LEVEL which maps each subject and object to a label from the given security lattice. The meaning of the protection state is that M specifies which accesses are currently authorized. The D and M components of the protection state are changed in different ways. D is changed at the discretion of subjects. A subject who owns an object controls access to that object. M is changed when an access is actually attempted. If D authorizes the access, and the simple-security and star-properties permit it, the relevant right is entered in the appropriate cell of M. Only the security officer can change the sets SUB and OBJ.

For another example of how these three issues are addressed consider the typed access matrix (TAM) model of Sandhu [San92] (which is obtained by adding strong typing of subjects and objects to the classical HRU model [HRU76]). In TAM a protection state consists of a set of subjects SUB, a set of objects OBJ, an access matrix AM whose cells contain entries from a set of rights R and a function TYPE which maps each subject and object to a type in the specified set of types. The meaning of the protection state is that AM specifies which accesses are currently authorized, as well how AM can be currently modified. The sets SUB and OBJ, and the access matrix AM are changed by executing one of a given collection of commands. A command will execute only if AM authorizes its execution.

A conceptual RBAC model will follow this established paradigm. Each of the three issues identified above need to be formally defined in some appropriate notation. Some of the components of such a model have been identified by Ferraiolo and Kuhn [FK92]. There is, however, much that remains to be done. As we have argued there are many variations regarding the precise behavior of RBAC. A vital component of this subtask is the development of a common framework that can accommodate these variations. In practice many of the "advanced" features of RBAC may not be needed in all applications, and may not be supported in all products. Nevertheless a complete RBAC

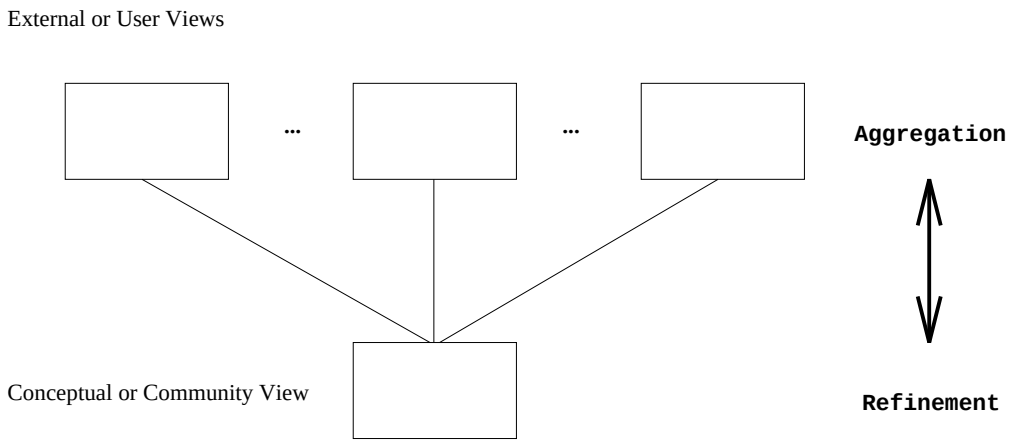

Figure 3: Harmonizing the Top Two Tiers

model must address the problem in its full generality. Restricted versions of the model can then be identified as needed. This approach is consistent with the ordered ranking of protection profiles in the draft Federal Criteria [Nat92]. It is important to analyze which aspects of RBAC add significant expressive power, and which are just dispensable conveniences.

In conclusion the RBAC model for the central tier must be a flexible and general model. It should be rigorously defined, have a solid formal foundation and yet be intuitively comprehensible and useful to practitioners. Although this is a challenging task, we feel it can be accomplished relatively soon.

5 HARMONIZING THE TOP TWO TIERS

Let us now consider the relationship between the top two tiers of the reference architecture, reproduced in figure 3. Each external view gives one perspective on the common community view, relevant to the particular context at hand. The relationship between the the top two tiers is one of aggregation and refinement as indicated in the figure.

Aggregation is a process by which several distinct roles are combined into a single role, because the distinction is not relevant in the given context. For example, the community view might have distinct roles for, say, Undergraduate Students, Master's Students and Doctoral Students. In an application where are all students are treated alike, these roles could be collapsed (i.e., aggregated) into a single Student role. In other applications, which confer different privileges to the various student roles, this distinction is significant. Refinement is simply the opposite operation to aggregation.

Different external views will aggregate different collections of roles from the community view. Some external views may aggregate the student roles into a single one. Others may keep the distinction between student roles but aggregate distinct faculty roles into one. Still others may aggregate both or none of the student and faculty roles. Our expectation is that a relatively small portion of the overall role set from the community view will be needed more or less intact in a particular external view. Most of the roles will, however, be aggregated. In other words each external view will see only a small part of the roles set in all its detail.

So long as entire roles are being aggregated or refined, the mapping between the top two tiers is relatively simple. There may be situations where the role relevant to the external view does not

come about so cleanly by aggregation. For example, suppose the community view has roles A and B, whereas the external view requires a role which has some (but not all) members of A and some (but not all) members of B. We identify below some techniques for accommodating such an external view.

- One could modify the community view to create a new role C and explicitly assign those members of A and B who should belong to this role. This treats A, B and C as unrelated roles.
- One could modify the community view to partition A into A_1 and A_2 (with $A_1 \cap A_2 = \phi$), and B into B_1 and B_2 (with $B_1 \cap B_2 = \phi$) so that $C = A_1 \cup A_2$ can be defined in the desired external view. This would require external views which use A to now treat A as an aggregate of A_1 and A_2, instead of being a role form the community view. Similarly, for external views which use role B.
- We could allow aggregation which can select the appropriate subsets of A and B, based on some condition for identifying members who should belong to the aggregated role C. This will complicate the aggregation operation and might dilute the central role of the conceptual view.

This list is not intended to be exhaustive. The point is that various alternatives are available as the community and external views adapt to the ever changing demands of the applications. One needs a systematic methodology for dealing with such changes.

6 HARMONIZING THE BOTTOM TWO TIERS

Now consider harmonization of the bottom two tiers, shown in figure 4. Each of the implementation views will aggregate roles from the community view. The aggregation done here will constrain which external views can be hosted on which implementation views. An implementation view that aggregates distinct student roles into a single role obviously cannot support an external view that requires this distinction to be maintained. In an ideal situation the implementation view may do no aggregation, in which case it could support all the external views. In practice, however, one would expect considerable aggregation to occur; if only because of legacy systems which have directly built in the external view without consideration of the common community view. Performance considerations may also require such aggregation to occur. Note that in both figures 3 and 4 aggregation is in the direction away from the central community view, and refinement is directed towards this view.

The second mapping shown in figure 4 is between implicit and explicit mechanisms. This mapping recognizes that the implementation platform may not support all the features of RBAC in the community view. For example, role hierarchies may not be supported. Suppose there are two roles Faculty and Staff such that every member of the Faculty role is automatically a member of the Staff role (but not vice versa). Thus a new faculty member need only be enrolled in the Faculty role, and will automatically be enrolled in the Staff role. This facility is often called role inheritance in the literature. Support for role inheritance in the community view is highly desirable, but such support will not be available on every implementation platform. To continue our example, at the community view it suffices to enroll a new faculty member into the Faculty role. However, in the implementation view the new faculty member will need to be enrolled in both Faculty and Staff roles. Similarly, a departing faculty member needs to be removed from the Faculty role in the community view; but in the implementation view requires removal from both Faculty and Staff roles.

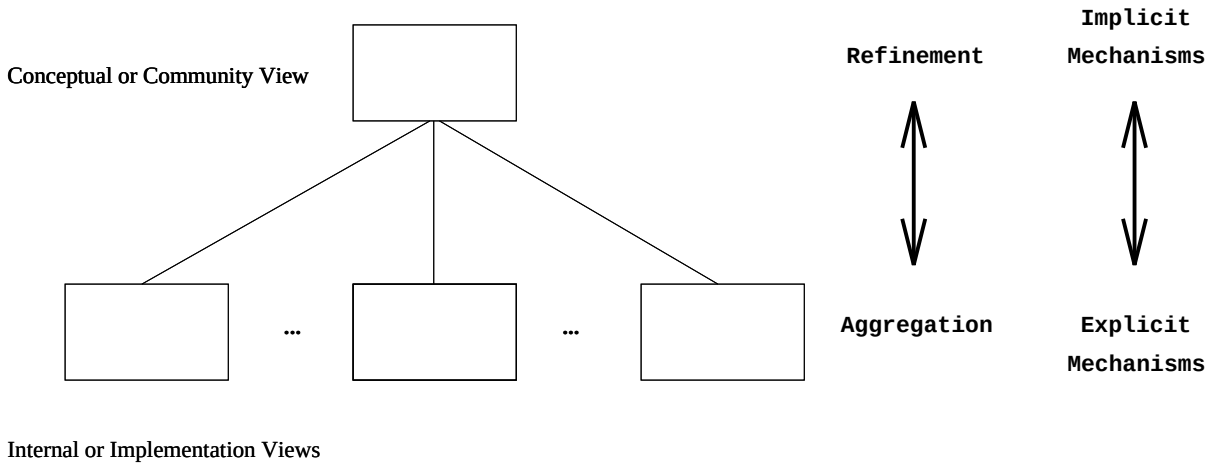

Figure 4: Harmonizing the Bottom Two Tiers

7 CONCLUSION

In this paper we have proposed a three-tiered reference architecture for role-based access control (RBAC), and have identified some of the issues that need to be addressed in making this framework a reality. Our reference architecture provides a perspective within which ongoing work on RBAC can be synthesized into a common framework.

In conclusion, we note that the appeal of RBAC is in the simplification of the management of authorizations. For example, maintaining cognizance of the permission set of an individual and the consequence of assigning particular role sets to a user is vital. It is also important for a security administrator to know exactly what authorization is implied by a role. This is particularly so when roles can be composed of other roles. Moreover, as new roles and transactions are introduced the security administrator needs tools to assist in their integration into the existing system. Future work in RBAC should identify useful tools for security administration and point the way toward designing these. We feel the central role of the community view in our reference architecture will greatly assist in this objective.

References

[Bal90] Robert W. Baldwin. Naming and grouping privileges to simplify security management in large database. In *Proceedings IEEE Computer Society Symposium on Research in Security and Privacy*, pages 61–70, Oakland, CA, April 1990.

[BL75] D.E. Bell and L.J. LaPadula. Secure computer systems: Unified exposition and Multics interpretation. Technical Report ESD-TR-75-306, The Mitre Corporation, Bedford, MA, March 1975.

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

[Dep85] Department of Defense National Computer Security Center. *Department of Defense Trusted Computer Systems Evaluation Criteria*, December 1985. DoD 5200.28-STD.

[FGL93] David F. Ferraiolo, Dennis M. Gilbert, and Nickilyn Lynch. An examination of federal and commercial access control policy needs. In *NIST-NCSC National Computer Security Conference*, pages 107–116, Baltimore, MD, September 20-23 1993.

[FK92] David Ferraiolo and Richard Kuhn. Role-based access controls. In *15th NIST-NCSC National Computer Security Conference*, pages 554–563, Baltimore, MD, October 13-16 1992.

[HRU76] M.H. Harrison, W.L. Ruzzo, and J.D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, 1976.

[Lun88] Teresa Lunt. Access control policies: Some unanswered question. In *IEEE Computer Security Foundations Workshop II*, pages 227–245, Franconia, NH, June 1988.

[LW88] Frederick H. Lochovsky and Carson C. Woo. Role-based security in data base management systems. In C.E Landwehr, editor, *Database Security: Status and Prospects*, pages 209–222. North-Holland, 1988.

[Mur93] William H. Murray. Introduction to access controls. In Hal F. Tipton and Zella A. Ruthberg, editors, *Handbook of Information Security Management*, pages 515–523. Auerbach Publishers, 1993.

[Nat92] National Institute of Standards and Technology, and National Security Agency. *Federal Criteria for Information Technology Security, Volumes I and II*, December 1992. Version 1.0, Draft.

[Nat93a] National Institute of Standards and Technology. *General Security Requirements for Cryptographic Module*, May 24 1993. Draft.

[Nat93b] National Institute of Standards and Technology, and National Security Agency. *Federal Criteria for Information Technology Security Workshop Proceedings*, July 1993. Issue 1.0.

[Ora92] Oracle Corporation. *ORACLE7 Server SQL Language Reference Manual*, December 1992. 778-70-1292.

[PB93] W. Timothy Polk and Lawrence E. Bassham. Security issues in the database language sql. Technical report, National Institute of Standards and Technology, July 30 1993.

[San88a] R.S. Sandhu. The NTree: A two dimension partial order for protection groups. *ACM Transactions on Computer Systems*, 6(2):197–222, May 1988.

[San88b] R.S. Sandhu. Transaction control expressions for separation of duties. In *Fourth Annual Computer Security Application Conference*, pages 282–286, Orlando, FL, December 1988.

[San91] R.S. Sandhu. Separation of duties in computerized information systems. In S. Jajodia and C.E. Landwehr, editors, *Database Security IV: Status and Prospects*, pages 179–189. North-Holland, 1991.

[San92] R.S. Sandhu. The typed access matrix model. In *Proceedings IEEE Computer Society Symposium on Research in Security and Privacy*, pages 122–136, Oakland, CA, May 1992.

[Ste92] Daniel F. Sterne. A TCB subset for integrity and role-based access control. In *15th NIST-NCSC National Computer Security Conference*, pages 680–696, Baltimore, MD, October 13-16 1992.

[Te78] D.C. Tsichritizis and A. Klug (editors). The ANSI/X3/SPARC DBMS framework: Report of the study group on data base management system. *Information Systems*, 3, 1978.

[Tho91] D.J. Thomsen. Role-based application design and enforcement. In S. Jajodia and C.E. Landwehr, editors, *Database Security IV: Status and Prospects*, pages 151–168. North-Holland, 1991.

[Tin88] T.C. Ting. A user-role based data security approach. In C.E Landwehr, editor, *Database Security: Status and Prospects*, pages 187–208. North-Holland, 1988.