Proc. of 10th Annual Computer Security Applications Conf., Orlando, Florida, December 5-9, 1994, pages 54-62.

Role-Based Access Control: A Multi-Dimensional View*

Ravi S. Sandhu†*, Edward J. Coyne, Hal L. Feinstein and Charles E. Youman*

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

Abstract

Recently there has been considerable interest in role-based access control (RBAC) as an alternative, and supplement, to the traditional discretionary and mandatory access controls (DAC and MAC) embodied in the Orange Book. The roots of RBAC can be traced back to the earliest access control systems. Roles have been used in a number of systems for segregating various aspects of security and system administration. Recent interest in RBAC has been motivated by the use of roles at the application level to control access to application data. This is an important innovation which 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. A number of proposals for RBAC have been published in the literature, but there is no consensus on precisely what is meant by RBAC. This paper lays the groundwork for developing this consensus.

In our view RBAC is a concept which has several dimensions, all of which may not be present in a given system or product. We envisage each dimension as being linearly ordered with respect to the sophistication of features provided. This leads us to the idea of a multi-dimension model for RBAC. Achieving agreement on what these dimensions are, and how the features in each dimension should be ordered, will take debate and time. Our contribution here is to lay out a vision on how to approach a common understanding of RBAC, and take a first cut at identifying the dimensions of RBAC. A major benefit of such a multi-dimensional RBAC would be to allow comparison of different products and assess their appropriateness for various system requirements.

*This paper is funded in part by a contract (50-DKNA-4-00122) from the National Oceanic and Atmospheric Administration. The views expressed herein are those of the authors and do not necessarily reflect the views of NOAA or any of its subagencies. The authors are grateful to David Ferraiolo and Janet Cugini of the National Institute of Standards and Technology for their support and encouragement in making this work possible.

†Ravi S. Sandhu is also affiliated with the Department of Information and Software Systems Engineering at George Mason University, Fairfax, VA 22030.

1 INTRODUCTION

A consensus has emerged in recent years that the traditional discretionary and mandatory access controls (DAC and MAC, respectively) embodied in DoD's landmark Orange Book [Dep85] are inappropriate 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.

Role-based access control (RBAC) has been proposed as an alternative, and supplement, to traditional DAC and MAC. Although RBAC is perceived to be a good match for the information security needs of a wide spectrum of organizations, there remains a lack of agreement 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].

The objective of this paper is to lay the groundwork for developing a consensus on the meaning of RBAC. In doing so we have attempted to unify and transcend existing literature on RBAC. In our view RBAC is a concept which has several dimensions, some of which may not be present in a given system or product. Within each dimension there is significant variation with respect to the sophistication of features provided. This leads us to the idea of a multi-dimension model for RBAC. Achieving agreement on what these dimensions are, and how the features in each dimension should be ordered, will take debate and time. Our contribution here is to lay out a vision on how to approach a common understanding of RBAC, and take a first cut at identifying the dimensions of RBAC. A major benefit of such a multi-dimensional RBAC would be to allow comparison of different products and assess their appropriateness for various system requirements.

The rest of this paper is organized as follows. Section 2 surveys previous literature on RBAC, tracing its roots to very early access control systems through its resurgence in recent times. Section 3 argues that roles are a policy component. It is therefore important to separate roles as policy from mechanisms, such as groups or compartments, that could be used to implement roles in a given access control system. Section 4 describes our vision of a multi-dimensional RBAC model. We identify some of the dimensions that such a model should have, and what features might belong in each one of these dimensions. What is reported here is the result of our initial analysis. It is presented here as a starting point for discussion of these issues with other security researchers and practitioners. Section 5 concludes the paper.

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.[1] In the simplest case, these decisions are made solely by the security administrator. More generally, the security administrator can selectively delegate this 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

[1] Not all proposals for RBAC agree with this position. For example, relations in Oracle [Ora92] can be owned by individuals who have the 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. Anticipating the discussion of section 3, we can treat Oracle roles as a mechanism which can, with suitable discipline, be used to implement the stated nondiscretionary policy for roles.

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]. Sandhu and Feinstein [SF94] have discussed a three-tier architecture for implementing RBAC on diverse platforms, which have varying amount of direct support for RBAC.

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 achievement in this area. This paper attempts to lay the groundwork for this task.

3 POLICY VERSUS MECHANISM

It is very important to distinguish roles as policy, from the mechanism that is used to implement roles in a particular access-control system. Failure to make this distinction leads to unnecessary confusion. This is reflected in the reaction that proponents of RBAC sometimes receive along the following lines.[2]

- Is there a difference between roles and groups? After all, roles can be implemented using groups.
- Similarly, for compartments or whatever somebody's favorite access control mechanism might be.

In our opinion this reaction represents a confusion between policy and mechanism. We regard roles as a policy component relating to the authority and responsibility relationships in an organization. Groups, compartments, or other mechanisms, are tools one can use to implement roles. The better aligned these tools are with the semantics of roles, the easier it will be to do the implementation. On the other hand, given a sufficiently powerful and flexible mechanism there will always be some way, however awkward and cumbersome, to implement roles using that mechanism.

It is useful to draw an analogy to programming languages to clarify this point. The concept of a **while** loop emerged only after several years of research in this arena. Since then **while** loops have been established as one of the cardinal components of structured programming. Now **while** loops can be implemented using DO loops in FORTRAN IV. Does that mean that there is no difference between **while** loops and DO loops? Is FORTRAN IV as good a language for structured programming as more modern languages such as PASCAL? Clearly, the answer to these questions is negative.

Similarly mechanisms which can be used to implement roles must be evaluated for their effectiveness in implementing roles, before asserting how suitable they are for this purpose. At the same time it is reassuring to realize that systems which do not directly support roles can still be used to implement RBAC. This is important so as to accommodate legacy systems.

To stretch the analogy further, let us ask whether use of a programming languages which has excellent constructs for support of structured programming equates to doing good structured programming. Again, the answer is clearly in the negative. Similarly, use of an access control system which has excellent support for RBAC will not equate to doing a good job of RBAC. It is possible to use good tools to do bad jobs, and mismatched tools to do good jobs. In an engineering discipline we can only hope that good tools will facilitate and make it easier to do a good job.

The policy-mechanism distinction has been elegantly incorporated in a taxonomy of security requirements given by LaPadula and Williams [LW91]. Security requirements need to be viewed at different levels of abstraction. LaPadula and Williams propose a layered taxonomy of stages, where the security requirements at higher stages are successively refined and elaborated at lower stages. Starting with the highest stage, these include:

1. *Trust Objectives:* The basic organizational security objectives to be achieved by a system.
2. *External-Interface Requirements:* This specifies the system's interface to the environment, in

[2] David Ferraiolo, personal communication.

terms of the security requirements.

3. *Internal Requirements:* Specifies requirements that must hold within the components internal to a system.

4. *Rules of Operation:* These rules explain how internal requirements are enforced.

5. *Functional Design:* This is a functional description of the behavior of system components.

Additional lower stages can be further developed going down all the way to code and hardware. At each boundary between two stages we can treat the higher stage as giving us policy and the lower one as giving us mechanism to enforce that policy.

The security requirements of a system at stages 1 and 2 above, are at a much higher level of abstraction than those at stages 3, 4, and 5. The higher stages specify *what* needs to be done, and these get refined into detailed executable specifications that deal with *how* things are to be done. The higher stages thus involve people-oriented policies and requirements while the lower ones are more computer-oriented.

Given these stages of elaboration, one can formulate security models for each of these stages, as well as classify existing models as to where they belong. In fact, it is possible to derive a related taxonomy of security models for the above stages (see figure 1). At the highest level we have models to capture organizational policy and requirements that pertain to security. These requirements are then applied to the interface between the organization and the computer system and captured by computer policy models. Computer policy models in turn are implemented by access-control models, which in turn map to implementation models, and so on.

Given such a taxonomy of security models, where would a model of RBAC fit in? We see RBAC as an attempt to formulate access-control models to bridge the gap between the internal requirements and higher stages of elaboration. We feel the proper place for an RBAC model is at stage 2, so it falls in the category of computer policy models shown in figure 1.

4 MULTI-DIMENSIONAL RBAC

In the rest of the paper we identify and discuss some dimensions of RBAC. The basic concept of a role has two aspects to it:

- users are assigned to roles, and
- privileges and permissions are assigned to roles.

A user assigned to a role thereby acquires the privileges and permissions of that role. This fundamental characteristic of roles is widely agreed upon, but there are many details and extensions on which there is little or no common agreement in the literature.

In this section we discuss the dimensions we have identified in our initial analysis. We are, of course, open to suggestions on other dimensions and modifications to the ones we have enumerated here, and expect to revise these ourselves to some extent. Within each dimension we have identified alternative approaches of addressing the issues of concern. The individual dimensions are largely independent of each other. The desire is to keep them independent, but that is sometimes not entirely possible. The individual dimensions are discussed below in a loose sequence from the more basic ones to the more sophisticated ones.

4.1 Nature of Privileges and Permissions

We use the term privileges to refer to general system wide authority. Examples of typical privileges in existing products are Operator, Auditor, System-Programmer, etc. Some of the newer access-control systems allow privileges to be customized for each installation. At any rate the nature of such privileges differs from one product to another. There may be room for standardization here. We feel that there is a need for customization of privileges for each installation. What could be standardized are elementary privileges which can be combined together in various combinations to construct compound privileges.

The term permission denotes access rights for particular data objects such as files. Most Operating Systems provide permissions such as read, write, execute, append, etc., to control access to these basic operations on files. At the application level, however, one would like to see permissions which relate to the transactions (or operations) of the application on objects which are meaningful in the application domain. For example, credit and debit operations on an account object. Note that credit and debit both require read and write access to the account balance. A user authorized to do a credit operation on an account should not be given arbitrary read and write access to the account balance. Rather, that user should be authorized to execute a program embodying the credit operation on the account. It is therefore important for application-oriented RBAC to support this requirement in some manner. With such support it is possible to authorize a bank-teller role to execute credit and debit opera-

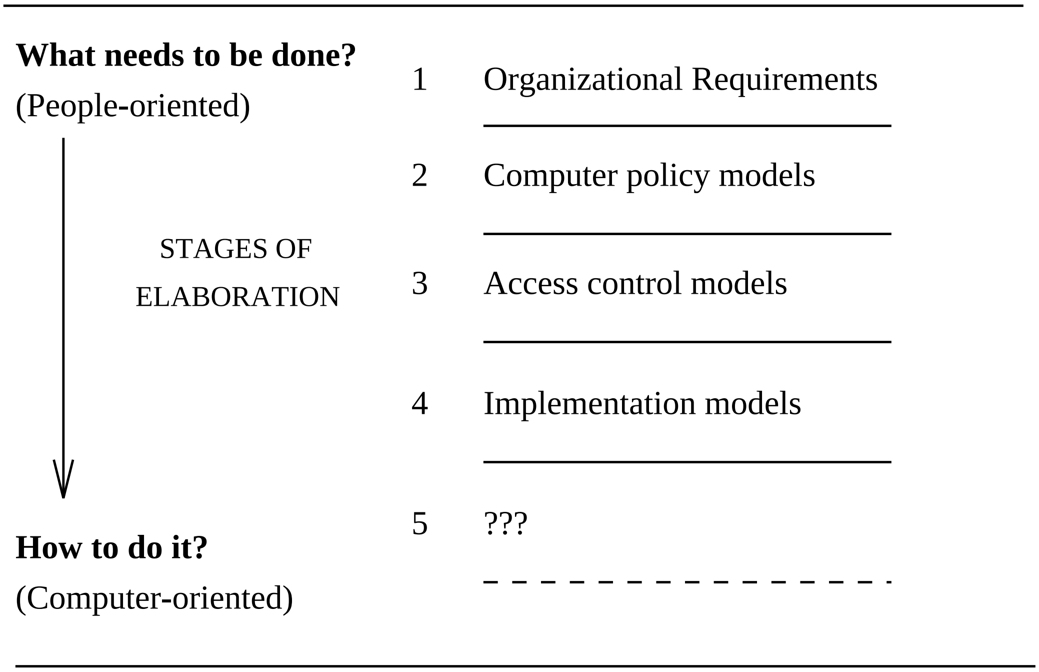

Figure 1: A taxonomy of models

tions on accounts rather than general read and write operations.

There are two different approaches to controlling permissions in an application-oriented manner. The first, and less granular, approach is to provide permissions entirely on basis of which programs (or transactions or operations or transformation procedures) a given role can execute. Thus, for example, the bank-teller role can be authorized to execute credit and debit operations. If these operations are authorized for all accounts then the credit and debit operations (more precisely, the processes which run these programs) can be authorized to perform read and write operations on the accounts.

The second approach provides for finer grained control, so the bank-teller role only gets authorization to debit and credit certain kinds of accounts. Such finer granularity can be provided directly by the access control system. Alternatively this finer granularity of control can be programmed into the application code for the credit and debit operations.

Another important question regarding RBAC is whether the privileges and permissions acquired via roles are sufficient to obtain access, or only necessary. In other words are additional permissions, for example DAC, required or not. To be concrete consider a physician role which is authorized to see medical information pertaining to patients. Does this allow a physician to see medical information for all patients, or only for those in the physician's care?

4.2 Hierarchical Roles

In many applications there is a natural hierarchy of roles, based on the familiar principles of generalization and specialization. For example, a physician role could be further specialized into, say, primary-care physician and specialist physician. In turn the role physician itself may be a specialization of a more general role called health-care provider (see figure 2(a)). A user assigned to the role of primary-care physician will also inherit privileges and permissions assigned to the more general roles of physician and health-care provider.

A tree structure is one obvious candidate for a hierarchy of roles. More generally, it can be argued that a partial order is also an appropriate structure [San88a]. An examples is shown in figure 2(b). Here the roles of hardware and software engineer are specializations of the engineer role. The role of supervising engineer inherits privileges and permissions from both of these roles. In object-oriented parlance this is an example of multiple inheritance, which is a frequent occurrence in information systems.

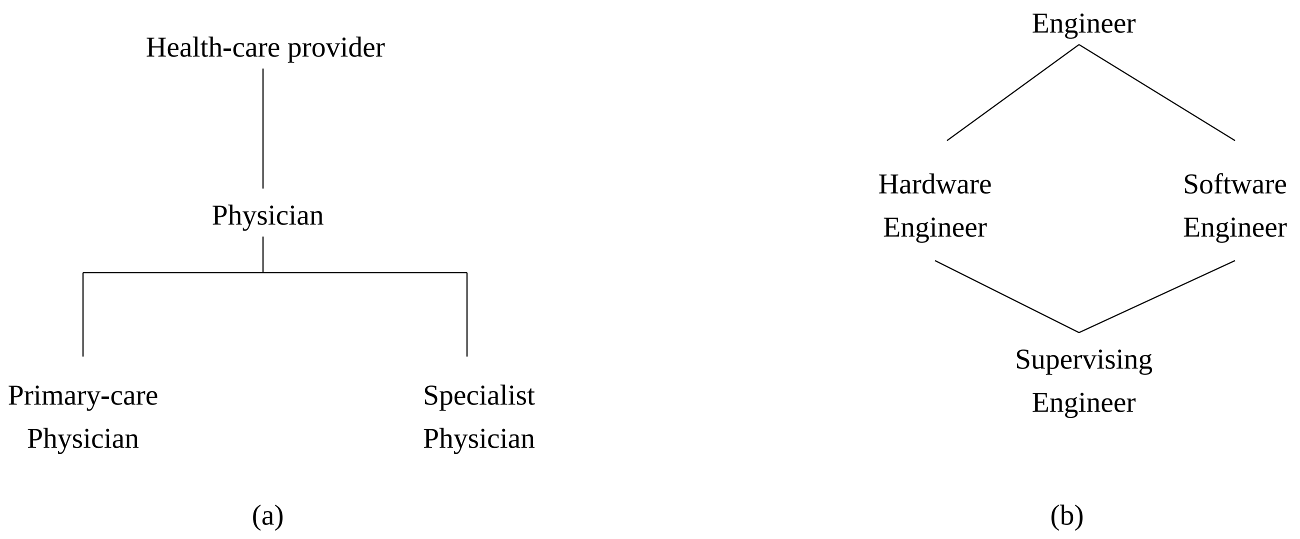

Figure 2: Examples of Role Hierarchies

There are a number of significant policy issues that arise in context of hierarchical groups. For instance, it may be useful to distinguish the privileges and permissions of a role that may be inherited through other roles, from those that are private to a role and cannot be inherited. In a truly general model we may also wish to consider denials (or negative authorization), in addition to the usual positive authorizations. 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, GSF91].

4.3 User Assignment

The dimension of user assignment is concerned with the manner by which users are assigned to roles. The primary issue here is whether user assignment to roles is entirely centralized and restricted to being done by a security officer, or whether there is some decentralization whereby certain users are authorized to do user assignment for some of the roles. The advantage of a centralized approach is the tight control it provides and its centralization of responsibility. The disadvantage is the increased administrative effort of dealing with routine manners, especially when the system becomes very large. Also, ultimately, requests for adding users to roles originate at the user end, and a good system should allow appropriate users to do this directly without a centralized point of control. Note that RBAC can be used profitably here. Roles for administration of user assignment can be created and authorized to enroll users, but only in a subset of the roles in the system.

There are two different aspects of user assignment to roles. Both of these are more critical when decentralized user assignment to roles is considered.

The first aspect is the question of whether or not there are there any constraints regarding the roles a user may belong to. In many applications some roles are considered to be mutually exclusive for purpose of separation of duties. For example, consider a supervisor role which is authorized to approve payments for vouchers and a clerk role which is authorized to issue a check. If the same user is given both roles there is an increased vulnerability to fraud due to misuse of authorized privileges by that user. To avoid this possibility, these two roles could be stipulated to be mutually exclusive, so no user can belong to both of them.[3] Note that mutually exclusive roles can be directly provided by an RBAC product, or instituted by administrative procedures outside of the computer

[3] Separation of duties based entirely on roles, as suggested here, is also known as static separation of duties. Dynamic separation of duties allows a user to belong to both roles in question, but that user can exercise only one role with respect to a particular object. For example, a user could approve payment of some vouchers and issue payment for some others, but cannot approve and issue payment for the same voucher. Roles by themselves only support static separation of duties. Support of dynamic operation of duties requires interaction between RBAC and permissions on individual objects, using transaction control expressions or similar mechanisms [San88b, San91].

system.

The second aspect of user assignment is concerned with constraints on which users can belong to a particular role. For example, consider a constraint which says that a user can be enrolled as an hardware engineer only if the user is already a member of the engineer role. In such a case the assignment of users to the engineer roles could be controlled centrally by the security officer, whereas the assignment of these users to specialized engineering functions could be delegated to appropriate users. This is an example of how discretionary authority can be delegated while still retaining some centralized constraints on it.

4.4 Privilege and Permission Assignment

Privilege and permission assignment to roles can also be centralized or decentralized, as discussed above for user assignment. Similar issues and tradeoffs arise in this case too. There are also similar considerations regarding mutually exclusive privileges and permissions. If permission assignment is delegated to users, they may go around mutually exclusive roles by means of permission assignment. Constraints on which privileges and permissions can be assigned to a particular can be very important. For example, the ability to write a prescription should be limited to the physician role.

In short this dimension is a dual of the previous one, and similar considerations would apply.

4.5 Roles Usage

The question of role usage is concerned with how a user can activate different roles in the system. One obvious issue is whether or not a user can take on multiple roles at the same time. This issue does not pertain so much to roles inherited via a hierarchy, but more so with roles that are independently assigned to the user. Thus a user in the primary-care physician role should automatically, and simultaneously, also be in the physician role. On the other hand, consider a user who has the roles of project manager and department manager. In such cases the question is whether these roles should be simultaneously held by the user. Allowing users to simultaneously exercise all their roles is convenient for users, who acquire all their privileges and permissions in a single session. On the other hand this violates the principle of least privilege, and opens vulnerabilities which need not exist. For example, as department head a user has access to confidential departmental information which can be copied into project documents by Trojan Horses. There is a clear need to limit the manner in which more powerful roles can be combined with more ordinary ones. For instance, a system programmer role should be assumed by a user only when needed and not routinely.

Other issues with role usage may be concerned with temporal constraints of how long a particular role can be held, or how often it might be exercised in a given time interval. The motivation here is to limit damage due to misuse or intrusion into powerful roles.

4.6 Role Evolution

The dimension of role evolution is a very important one which is all too often ignored. In a large organization one can expect a large number of roles which will change and evolve over time. Existing roles will be merged, split, discarded; and new ones will be created as the organizational structure evolves. It is imperative that access-control systems provide good support for such changes.

4.7 Object Attributes for RBAC

The final dimension we mention is that of object attributes which can be employed for RBAC. Roles basically provide a convenient means for grouping users together for access control, on basis of their job functions. Object attributes would provide a similar facility for grouping of object on basis of the tasks and job functions they support in an organization. Complete consideration of this issue is outside the scope of this paper. It is not even clear whether it belongs within the scope of RBAC. We have mentioned it here because of its importance.

5 CONCLUSION

In this paper we have proposed the concept of a multi-dimensional approach to role-based access control (RBAC). We have identified several dimensions, and have discussed different variations that can arise within each dimension. To a large extent the dimensions are independent of each other. The particular dimensions identified here, and the features discussed within each dimension, represent our initial cut at this task. We are, of course, open to suggestions on other dimensions and modifications to the ones we have enumerated here, and expect to revise these ourselves to some extent. Achieving a commonly accepted understanding of RBAC along the multi-dimensional vision proposed in this paper would be of significant benefit to vendors and users of access-control products.

One major benefit would be to allow comparison of different products and assess their appropriateness for various system requirements.

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.

[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.

[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.

[GSF91] Ehud Gudes, Haiyan Song, and Eduardo B. Fernandez. Evaluation of negative, predicate, and instance-based authorization in object-oriented databases. In S. Jajodia and C.E. Landwehr, editors, *Database Security IV: Status and Prospects*, pages 85–98. North-Holland, 1991.

[Lun88] Teresa Lunt. Access control policies: Some unanswered questions. 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.

[LW91] L.J. LaPadula and J.G. Williams. Toward a universal integrity model. In *IEEE Computer Security Foundations Workshop*, pages 216–218, Franconia, NH, June 1991.

[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. T. Polk and Lawrence E. Bassham. Security issues in the database language SQL. Technical report, National Institute of Standards and Technology, July 30 1993.

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

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

[San91] Ravi 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.

[SF94] Ravi S. Sandhu and Hal L. Feinstein. A three tier architecture for role-based access control. In *17th NIST-NCSC National Computer Security Conference*, Baltimore, MD, October 11-14 1994.

[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.

[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.