Specifying and Managing Role-Based Access Control within a Corporate Intranet

David Ferraiolo and John Barkley
National Institute of Standards and Technology
Gaithersburg, Maryland 20899

Abstract

In order for intranets to reach their full potential, access control and authorization management mechanisms must be in place that can regulate user access to information in a manner that is consistent with the current set of laws, regulations, and practices that face businesses today. The purpose of RBAC on the Web would be to provide this access control service, thereby enabling the use of the Web for new and more sophisticated applications -- to allow access to information and other resources that would otherwise not be possible given the existing lack of operational assurance. This paper describes an approach at providing these assurances through the use of RBAC for networked Web servers.

1 Introduction

One of the greatest obstacles in the growth of intranets as a means of enterprise computing is the inability to effectively manage authorization data. Today, authorization management is costly and prone to error. Web Server administrators usually control user access to enterprise published documents through the creation and maintenance of ACLs on a server-by-server basis.

ACLs specify, for each protected resource, a list of named individuals, or groups composed of individual users, with their respective modes of access to that object. When users are required to self-administer access rights to the objects they "own," ACLs have many advantages. At any given time it is easy to answer the question -- Who and, under what mode of access, has access to this object? Here users are provided with the capability of granting other users access to objects, and revoking existing accesses on a "need-to-know" basis.

1.1 The Existing Problem

This use of ACLs is problematic for a variety of reasons. In many enterprises within industry and civilian government, end users do not "own" the information for which they are allowed access [1][3]. For these organizations, the corporation or agency is the actual "owner" of system objects and discretion on the part of the users may not be appropriate. Although, enforcing a need-to-know policy is important where classified information is of concern, there exists a general need to support subject-based security policies, such as, access based on competency, the enforcement of conflict-of-interest rules, or permitting access based on a strict concept of least privilege. To support such policies assumes the ability to restrict access based on a user function or role within the enterprise. Here the relevant question is: "What are the current access rights for this user"? Performing such a review is difficult when authorization data is organized in an ACL structure. ACLs further complicate matters when a user takes on new responsibilities or changes roles within the enterprise. To reflect these changes would entail a through review and selective deletion of all the user's existing privileges on all servers. Without the ability to perform a per-user review an enterprise runs the risk of maintaining residual and inappropriate user access rights.

From an authorization management perspective, each Web server is treated as stand-alone device. Their underlying access control mechanisms are locally and individually administrated with little or no regard to the authorization data maintained on the other servers within the network. To attempt to administer enterprise wide subject-based policies, using ACLs alone, would require intense coordination across administrative boundaries, would be prone to error, and result in a low degree of confidence that the required control policies would be faithfully and consistently enforced.

Because of the inherent risk associated with this lack of operational security assurance, organizations have resisted publishing sensitive information on their Web servers, thereby limiting their utility, and depriving the organization of potential productivity gains.

1.2 Solution: Role-Based Access Control

To solve these and other operational assurance problems, NIST has initiated an effort to implement Role-Based Access Control (RBAC) for the WWW (RBAC/Web). RBAC is a technology that is attracting increasing attention, because of its potential for reducing the complexity and cost of authorization management in large systems [3][4][5][12]. RBAC provides administrators with a context for the specification and enforcement of complex security policies, that are often impractical or even impossible to enforce through the direct administration of lower level access control mechanisms, such as ACLs.

For Web server applications, RBAC/Web provides these administrative conveniences by composing the seemingly unrelated and incomprehensible authorization data of the lower level access control mechanisms, and other RBAC relevant data, into a single RBAC authorization database. In doing so, RBAC/Web organizes this authorization data and presents it to the intranet administrator(s) in a relational format and at a level of abstraction that is natural to the way enterprises are normally structured and conduct business. From an administrators perspective, RBAC/Web serves as a visualization and maintenance tool of the enterprises intranet access control mechanisms in terms of it's users, roles, role hierarchies, operational constraints, and privileges.

The remainder of this paper describes NIST's approach to RBAC/Web. Section 2, provides an overview of RBAC/Web including its constituent process components and services. Section 3, provides a detailed description of RBAC and its supporting security policies. Section 4, describes the RBAC/Web distributed authorization database, and the static security policies that it supports. The RBAC/Web Authorization database is distributed among some of the RBAC/Web network components described in section 2. Section 5, Role Activation, provides a description of the operation of RBAC/Web and its dependency on the RBAC Authorization database in supporting dynamic security policies through the creation of an Active Role Set (i.e., the introduction of a role or set of roles into a users active session). Section 6, Scenario of Use, provides a comprehensive scenario of use, from a client request for URL access (at the browser), through user authorization and activation, to the result provided by the Web server.

2 RBAC/Web Overview

RBAC for the World Wide Web (RBAC/Web) is an implementation of RBAC for use by World Wide Web (WEB) servers. Because RBAC/Web places no requirements on a browser, any browser that can be used with a particular Web server can be used with that server enhanced with RBAC/Web. RBAC/Web is implemented for both UNIX (e.g., Netscape, NCSA, CERN, or Apache servers) and Windows NT (e.g., Internet Information Server, Website, or Purveyor) environments.

Components of RBAC/Web are shown in Table 1. RBAC/Web for UNIX uses all of the components in Table 1. Because built-in NT security mechanisms are closely compatible with RBAC, the NT version uses only the Database, Session Manager, and Admin Tool components. RBAC/Web for NT requires no modification of Web server internals or access to source code. With RBAC/Web for UNIX, there are two ways to use RBAC/Web with a UNIX Web server.

The simplest way is by means of the RBAC/Web CGI. The RBAC/Web CGI can be used with any existing UNIX server without modifying its source code. RBAC URLs are passed through the Web server and processed by the RBAC/Web CGI. RBAC/Web configuration files map URLs to file names, while providing access control based on the user's roles. Installation of the RBAC/Web CGI is similar to the

Table 1: RBAC/Web Components

Data-base	Files that specify the relationship between users and roles, the role hierarchy, the constraints on user/role relationships, current active roles, and relationship between roles and privi-leges.
Data-base Server	Hosts the authoritative copies of the files which define relationships between users and roles, the role hierarchy, and the constraints on user/role relationships. These files are cre-ated and maintained by the Admin Tool.
API Library	A specification which may be used by Web servers and CGIs to access the RBAC/Web Database. The API is the means by which RBAC may be added to any Web server implementation. The API Library is a C and Perl library which implements the RBAC/Web API.
CGI	Implements RBAC as a CGI for use with any currently existing Web server without having to modify the server. The RBAC/Web CGI uses the RBAC/Web API.
Session Manager	Manages the RBAC Session. The RBAC/Web Session Manager cre-ates and removes a user's current active role set.
Admin. Tool	Allows server administrators to cre-ate users, roles, and permitted opera-tions; associate users with roles and roles with permitted operations; spec-ify constraints on user/role relation-ships; and maintain the RBAC Database. Administrators access the RBAC/Web Admin Tool by means of a Web browser.

installation of the Web server.

While RBAC/Web CGI is relatively simple to install anduse, it is not as efficient as performing access control directly in the Web server. The other way to use RBAC/Web is to modify the UNIX Web server to call the RBAC/Web API to determine RBAC access. A URL is configured as an RBAC controlled URL by means of the Web server configuration files that map URLs to file names.

Some Web servers for a UNIX environment, such as Netscape and Apache, divide their operation into steps and provide the capability for each step to be enhanced or replaced by means of a configuration parameter. This allows Web server operation to be modified without having to change the server's source code. For these Web servers, the RBAC/Web API can be integrated by simply providing the appropriate calling sequence and modifying configuration parameters.

RBAC is an access control mechanism that can be used in conjunction with existing WWW authentication and confidentiality services. These include username/password, Secure Socket Library (SSL), Secure HTTP (SHTTP), and Private Communication Technology Protocol (PCT). User identification information is passed to RBAC/Web by the Web server. It is the responsibility of the Web server to authenticate user identification information and provide confidential data transmission as configured by the Web server administrator.

A description of RBAC, the RBAC Authorization Database, and some RBAC/Web supported policies are describe below.

3 Administration

While RBAC can be treated as either a discretionary or non-discretionary access control method, the treatment given in this paper is the latter. That is, one or more administration roles are required that are distinct from user roles, insofar as their permissions deal solely with the policy attribute elements of the model: User-to-Role and Role-to-Permission mappings, containment relations, cardinality constraints, and separation of duty constraints. Users not assigned to administration roles are denied these permissions and must operate within the confines of the roles defined for them and assigned to them by an administrator. Conversely, users assigned to administration roles are restricted to administration of policy attribute elements when active in those roles.

Division of roles in this manner supports the principle of Attenuation of Privileges which states that subjects should not be able to increase their privilege or grant to other subjects privileges they themselves do not own. Separation of authorization aspects from the policy-attribute management is useful in practice since authorization must be relatively independent of how policy attributes, such as roles, are managed [5][8][9]. However, a circular dependency between authorization and policy attribute management exists in such models, since authorization requires defined policy attributes for controlling access, and specification of policy attributes requires control of access to that information [9].

Under RBAC, users are granted membership into roles based on their competencies and responsibilities. User membership into roles can be revoked easily and new memberships established as job assignments dictate, without having to deal with the complexity of the underlying access control mechanisms. With RBAC, users are not granted permission to perform operations on an individual basis, as is the case with conventional access control methods, but instead privileges are associated with roles and users are granted membership into those roles. Role association with new privileges can be established as well as old privileges deleted as organizational functions change and evolve. Roles can be hierarchical. For example, some roles in a hospital may be healthcare provider, intern, and doctor. The doctor role may include all privileges available to the intern role, which in turn includes all the privileges available to the health care provider role.

RBAC is administered through the use of roles and role hierarchies that mirror an enterprise's job positions and organizational structure. Users are assigned membership into roles in a manner that is consistent with a user's duties, competency and responsibility. Constraints are imposed on user membership into roles and on a user's ability to activate a role to address conflict of interest issues. Complexities that are introduced by simultaneously supporting mutually exclusive roles and role hierarchies are handled by the RBAC software, making security administration easier. It is the roles, role hierarchies, and constraints that provide the context by which the intranet administrators can specify, and RBAC/Web servers can enforce, the specifics of a large variety of laws, regulations and business practices that can pertain to an organization.

RBAC has been shown to support several well-known security principles and policies that are important to commercial and government enterprises that process unclassified but sensitive information [1][12]. These include: the specification of competency to perform specific tasks; the enforcement of Least Privilege for administrators and general users; and the specification, as well as the enforcement, of conflicts of interest rules, which may entail duty assignment and dynamic and static separation of duties. For RBAC/Web these policies can be enforced at the time that users are authorized as members of a role, at the time of role activation (e.g., when a role is established as part of a user's active session), or at the time the user attempts to access a URL.

In addition to RBAC's commercial relevance, RBAC has the potential to support policies that are essential within classified environments. Such policies can include one-directional information flow and provide the same effects as the well accepted Simple Security property and the Star-property[1][6] of the Bell and Lapadula security model[11].

4 RBAC/Web Database

The RBAC/Web Database includes roles, and role hierarchies; relational association of users with roles and roles with permitted operations on objects; and relational constraints on role membership, role activation, and object access.

Within the RBAC/Web Database, a *user* is a person that is represented by a unique identifier, a *role* is a collection of job functions, and a *privilege* represents a particular method of access to a set of one or more protected RBAC *objects*. When authorizing user membership into a role, the user is implicitly provided with the potential to exercise the privileges that are associated with the role. Privileges in RBAC/Web are the HTTP methods that the end-user can perform on RBAC controlled URLs.

Roles can have overlapping responsibilities and privileges, that is, users belonging to different roles may need to perform common operations. Furthermore, within many organizations there are a number of general privileges that pertain to all employees. As such, it would prove inefficient and administratively cumbersome to specify repeatedly these general privileges for each role that gets created. To improve administrative efficiency and support the natural structure of an enterprise, RBAC and therefore RBAC/Web includes the concept of *role hierarchies*. A role hierarchy defines roles that have unique attributes and that may "contain" other roles, that is, one role may implicitly include the set of privileges that are associated with another role. Role hierarchies are a natural way of organizing roles to reflect authority and responsibility, and competency.

1. The Simple Security Property states that a subject (i.e., a process executing on a user's behalf) must not be allowed to read from storage repositories that are at a higher sensitivity level than the subject's current sensitivity level. The Star Property states that a subject must not be allowed to write to storage repositories that are at a lower sensitivity level than the subject's maximum sensitivity level allowed for reading.

Role hierarchies within the RBAC/Web authorization database are represented as ancestor relationships. The immediate parent relationship can be represented as an ordered pair $((R_{i+1}, R_i), >)$, where R_{i+1} is the immediate parent and R_i the child and ">" is a transitive relation "contains," which induces a hierarchical structure on the role set. Thus, $R_{i+1} > R_i$ implies R_{i+1} contains R_j.

Role hierarchies are an ideal structure for ensuring adherence to the principle of Least Privilege which applies to administrators as well as general users. The principle of Least Privilege requires that a user be given no more privileges than necessary to perform his/her job function. Ensuring least privilege requires identifying the users's job functions, determining the minimum set of privileges required to perform that function, and restricting the user to a domain with those privileges and nothing more. In non-RBAC implementations, where privileges are organized on a per user or per object basis, least privilege is often difficult or costly to achieve and maintain.

From a policy perspective, the capability within RBAC/Web to administratively impose constraints on user membership into roles provides a powerful means of enforcing conflict of interest and cardinality rules for roles as they uniquely apply to an enterprise. For example, to address conflict of interest issues, RBAC/Web can enforce a rule of static separation of duty (SSD) when defined within the authorization database. This means that a user may be authorized as a member of a role only if that role is not mutually exclusive with any of the other roles for which the user already possesses membership. For example, a user that is authorized as a member of the role Derivative Trader may not be allowed to be a member of the role Derivative Settler for the same securities group. Another type of constraint imposed on the RBAC/Web authorization database is the cardinality of a role. Some roles in a organization may only be occupied by a certain number of employees at any give time. For example consider the role of a department head. Although over time a number of individuals may assume this role, only one individual may assume the responsibilities of the department head at a given point in time. Cardinality constraints could also be used as a means of enforcing licencing agreements.

In general, constraints provide confidence as to the adherence of enterprise wide policies. In theory, similar effects can be achieved through the establishment of procedures and the sedulous actions of administrators. For example, administrators can maintain and share a list of role pairs that are known to be mutually exclusive and ensure that an individual user never gains membership to both role. However, the reality is that procedures break down and administrators get reassigned over time. The constraints imposed by RBAC/Web provide management and regulators with the confidence that critical security policies are uniformly and consistently enforced within the network, and as such, contributes to the networks operational assurance.

To further promote operational assurance, RBAC/Web provides security administrators with a complete and consistent view of the entire RBAC Database. This is important because of the manner in which authorization data is distributed among the RBAC/Web servers. The RBAC/Web Database include data elements that pertain to the ACLs that reside with each Web server where RBAC/Web is installed. An ACL is organized as a list of roles, where for each role, there is a list of HTTP methods under which a user acting in the role is permitted to access an associated URL. The collection of ACLs are organized and managed as the collection of the role-privilege relationships within the RBAC/Web Database. The user-role relationships, role-role relationships, and constraint relationships (for user membership and role activation) are maintained on the RBAC/Web database Server.

5 RBAC/Web Role Activation

In the context of RBAC/Web, each subject represents a user active in one or possibly many roles. As shown in Figure 3, a *subject* represents an active user process with the single and double arrow denoting a one-to-many relationship. A user establishes a session during which the user is associated with a subset of the roles for which the user has membership (i.e., the user's ARS). A user's authorization (which is a consequence of role membership) is a necessary but not always sufficient condition for a user to be permitted to execute a privilege. Other organizational policy considerations or constraints may need to be taken into consideration that pertain to authorizing users to execute privileges.

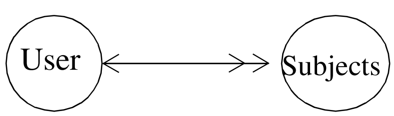

Figure 1. User and subjects

RBAC/Web requires a user to first be authorized as active in a role before a user is permitted to perform an operation or access a URL. This provides the context for other policy checks to be imposed. As such, RBAC/Web provides administrators with the capability to enforce an organization-specific policy of Dynamic Separation of Duty (DSD). DSD places constraints on the simultaneous activation of roles. For example, an individual 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/Web would require the user to drop his or her Cashier role, and thereby forcing the closer of the cash drawer before assuming the role Cashier Supervisor. As long as one individual is not allowed to assume both of these roles at the same time a conflict of interest situation will not arise. Although the same effect could be achieved through the establishment of an SSD relationship, DSD relationships generally provide the enterprise with greater operational flexibility.

6 Scenario of Use

From the users perspective, the end-user interaction with a WWW server enhanced with RBAC/Web is similar to requesting URLs whose access is not controlled by RBAC/Web. However, before access to a URL controlled by RBAC is permitted, end-users must establish an RBAC session. In establishing the RBAC session, end-users choose and/or are assigned a current active role set (ARS). The ARS determines the HTTP methods that the end-user can perform on RBAC controlled URLs. The ARS remains in effect until the end-user establishes a new ARS. It is the ARS which constitutes the RBAC session. An end-user has only one RBAC session at any given time.

A user may be assigned roles which have DSD relationships. If this is the case, the Session Manager enables users to choose the subset of their assigned role set that they would like to use in the session. Users are presented with a list of subsets which do not violate any DSD relationships and asked to choose. In order to minimize the number of choices, the subsets in the list, taken from the set of all possible subsets of a user's assigned roles, contains the largest subsets which do not violate any DSD relationships. Once the choice is made, the RBAC session is established with an ARS consisting of all assigned roles in the chosen subset and all roles which the assigned roles inherit. If there are no DSD relationships among the roles assigned to a user, then the RBAC session is automatically established with all authorized roles in the ARS.

7 Conclusion

Although intranets can offer great benefits to a company or government agency, security problems remain. For intranets to reach their full potential as a means for enterprise computing, access control mechanisms must be in place that can conveniently, and cost effectively regulate user access to information, while providing management with a confidence that their critical policies are faithfully and consistently enforced across administrative boundaries. To solve these and other authorization problems, NIST has initiated an effort to provide and promote the use of Role-Based Access Control (RBAC) for the WWW (RBAC/Web). RBAC is particularly attractive for intranet applications because of its ability to reduce the complexity and cost of authorization management. In addition, RBAC provides a context for the specification and enforcement of complex security policies that are often impractical or even impossible to enforce through the direct use of conventional access control mechanisms. Under RBAC, intranet administrators are provided with a single view of the RBAC authorization database which is at a level of abstraction that is intuitive and consistent with the way the enterprise is structured and conducts business. RBAC/Web thereby bridges the huge gap between the enterprise's laws, regulations, and business practices and the details of the underlying access control mechanisms of the Web servers.

References

[1] David F. Ferraiolo, Dennis M. Gilbert, Nickilyn Lynch, "An Examination of Federal and Commercial Access Control Policy Needs," *Proceedings of the 16th NIST-NSA National Computer Security Conference*, Baltimore, MD, 20-23 September 1993.

[2] Imtiaz Mohammed and David M. Ditts, "Design for Dynamic User Role-Based Security," *Computers and Security,* 1994.

[3] David F. Ferraiolo and Richard Kuhn, "Role-Based Access Control," *Proceedings of the 15th NIST-NSA National Computer Security Conference*, Baltimore, MD, 13-16 October 1992.

[4] David F. Ferraiolo, Janet A. Cugini, D. Richard Kuhn, "Role-Based Access Control: Features and Motivations," Proceedings 11th Annual Computer Security Applications Conference, New Orleans, LA, December 1995.

[5] R. Sandu, E.J. Coyne, H.L. Feinstein, and C.E. Youman, "Role Based Access Control Models," IEEE Computer, 29(2), February 1996.

[6] Department of Defense, *Trusted Computer Security Evaluation Criteria*, DoD 5200.28-STD, 1985.

[7] National Computer Security Center, *A Guide to Understanding Discretionary Access Control in Trusted Systems*, NCSC-TG-003, September 1987.

[8] Hal L. Feinstein, et al., *Final Report: Small Business Innovation Research (SBIR): Role-Based Access Control: Phase 1*, McLean, VA, SETA Corporation, January 20, 1995.

[9] Virgil Gligor, *RBAC Security Policy Model, Preliminary Draft Report,* R23 Research and Development Department of the National Security Agency, April 1995.

[10] Virgil Gligor, J. Huskamp, S. Welke, C. Linn, and W. T. Mayfield, *Traditional Capability-Based Systems: An Analysis of Their Ability to Meet the Trusted Computer Security Evaluation Criteria,* IDA Paper P-1935, October 1986.

[11] D. E. Bell and L. J. LaPadula, *Secure Computer Systems: Mathematical Foundations*, Technical Report. ESD-TR-73-278, Volume 1, The MITRE Corporation, Bedford, MA, March 1973.

[12] S. H. von Solms and Isak VanderMerve, "The Management of Computer Security Profiles Using a Role-Oriented Approach," *Computers and Security,* 1994.