Role Based Access Control for the World Wide Web

John F. Barkley, Anthony V. Cincotta,
David F. Ferraiolo, Serban Gavrilla, and D. Richard Kuhn
National Institute of Standards and Technology
Gaithersburg, Maryland 20899
301-975-3290 [voice], 301-926-3696 [fax] (Rick Kuhn, point of contact)
kuhn@nist.gov

April 8, 1997

Abstract

One of the most challenging problems in managing large networked systems is the complexity of security administration. This is particularly true for organizations that are attempting to manage security in distributed multimedia environments such as those using world Wide Web (WWW) servers. Today, security administration is costly and prone to error because administrators usually specify access control lists for each user on the system individually.

Role based access control (RBAC) is a technology that is attracting increasing attention, particularly for commercial applications, because of its potential for reducing the complexity and cost of security administration in large networked applications. This paper describes software components that provide RBAC for networked servers using WWW protocols. The RBAC components can be linked with commercially available web servers, and require no modification of the server software.

1 Introduction

Establishing and maintaining a presence on the World Wide Web (WWW), once a sideline for US industry, has become a key strategic aspect of marketing and sales. Many companies have demonstrated that a well designed Web site can have a positive effect on their profitability. Enabling customers to answer their own questions by clicking their way through Web pages, instead of dealing with operators and voice response systems, increases the efficency of the customer interface.

More recently companies have begun using web technology on private networks for service to internal clients. Web sites are now running inside the company, most created for and by employees. Corporations are seizing the Web as a swift way to streamline – even transform – their organizations. These private nets, or "intranets," use the infrastructure and standards of the Internet and the World Wide Web but are cordoned off from the public Internet through firewalls.

The Web can be used as an inexpensive yet powerful alternative to other forms of internal communications. Due to the fact that Web browsers run on any type of computer, electronic information can be accessed consistently and concurrently by all employees. A plethora of corporate information (e.g., procedures, training materials, directories, forms) can be converted to electronic form and made available via the Web. With a single source for these materials the cost of maintenance is significantly reduced, while greatly simplifying the task of ensuring currency. Thus an objective of enterprise computing, creation of a company wide system irrespective of the underlying information technology infrastructure, can be fulfilled.

Although intranets can offer great benefits to a company or government agency, security threats remain. To date net enthusiasts tend to focus on how to link people and businesses, not on using the network as a way to run and manage businesses securely. As was the case before intranets, not all users are allowed access to all information. Although existing Web servers can effectively provide all or nothing access to a particular Web site and a number of popular Web servers can even provide fairly fine grained access control, they provide very primitive tools to administer these controls from the perspective of a single enterprise.

This paper describes the benefits of RBAC and an implementation of RBAC on the Web (RBAC/Web), and in particular as RBAC applies to an intranet computing environment. This will provide intranet administrators with a capability for the first time to centrally administer and regulate user access to information in a manner that is consistent with the current set of laws, regulations, and practices that face their business today.

2 RBAC Description

Role based access control (RBAC) [1], [2], [3], [4], [5] is an alternative to traditional discretionary (DAC) and mandatory access control (MAC) policies that is attracting increasing attention [6], particularly for commercial applications. The principle motivation behind RBAC is the desire to specify and enforce enterprise-specific security policies in a way that maps naturally to an organization's structure. Traditionally, managing security has required mapping an organization's security policy to a relatively low-level set of controls, typically access control lists.

With RBAC, security is managed at a level that corresponds closely to the organization's structure. Each user is assigned one or more *roles*, and each *role* is assigned one or more *privileges* that are permitted to users in that role (see Figure 1). Roles can be hierarchical. For example, some roles in a hospital may be health care provider, nurse, and doctor. The doctor role may include all privileges available to the nurse role, which in turn includes all the privileges available to the health care provider role. Security administration with RBAC consists of determining the operations that must be executed by persons

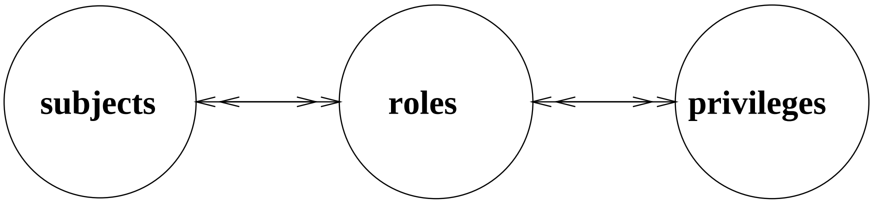

Figure 1: RBAC Relations

in particular jobs, and assigning employees to the proper roles. Complexities introduced by mutually exclusive roles or role hierarchies are handled by the RBAC software, making security administration easier. A formal description of RBAC is provided in the Appendix.

2.1 RBAC Example

Consider the branch office of a bank. In this environment, there are roles such as branch manager, teller, and account representative, as illustrated in Figure 2.

The graph structure shows role hierarchy. The role $financial\_advisor$ inherits the role *account_rep*. An individual authorized for the role $financial\_advisor$ is permitted to perform all of the operations permitted to an individual authorized for the role *account_rep*. Thus, an individual in the role of $financial\_advisor$ is able to create and remove accounts. Because account representatives, branch managers, internal auditors, and tellers are all employees of the bank, their corresponding roles inherit the employee role.

In Figure 2, the role *account_rep* is highlighed in order to show the other role relationships for *account_rep*. The roles *teller* and *account_holder* are shown in rectangles to indicate that these roles have a "Dynamic Separation of Duties" (DSD) relationship with the role *account_rep*. This relationship is a conflict in interest relationship indicating that an individual acting in the role of *account_rep* cannot also be acting in either of the roles of *account_holder* or *teller*. The policy of the bank is that an account representative, an employee of the bank, can have an account in the bank but such an individual may not simultaneously process their personal account while processing accounts of others. Likewise, because a teller has an open cash drawer that must balance when closed, an individual acting in the role of *account_rep* and sitting at a desk away from a teller's window is not permitted to simultaneously act in the role of *teller* even if authorized for that role.

The role internal_auditor is shown in an octagonal box to indicate that this role has a "Static Separation of Duties" (SSD) relationship with the role *account_rep*. The SSD relationship is also a confict of interest relationship like the DSD relationship but much stronger. If two roles have a DSD relationship, then they may both be authorized for an individual but that individual may not act in both roles simultaneously. If two roles have a SSD relationship, then they may not even be authorized for the same individual. In this example, the policy of the bank is that there is a fundamental conflict of interest between the roles of internal_auditor and *account_rep*. Thus, these two roles may never be authorized for the same individual.

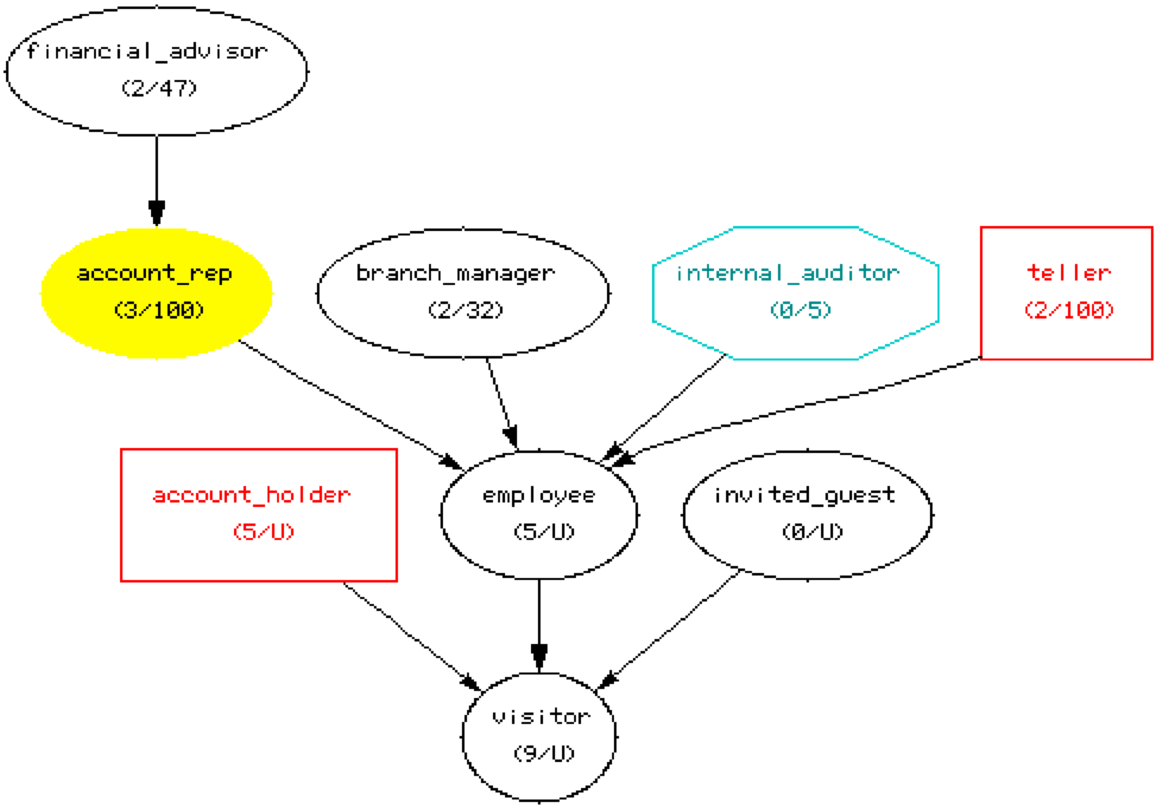

Figure 2: Bank Example

3 RBAC for World Wide Web Applications

Role Based Access Control (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., for Netscape, NCSA, CERN, or Apache servers) and Windows NT (e.g., for 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 installation of the Web server.

While the RBAC/Web CGI is relatively simple to install and use, 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 paramenters. 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.

3.1 Authentication

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.

Database	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 operations.
Database 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 created and maintained by the Admin Tool. When changes are made these files, the Database Server notifies the Web Servers to update their cached copies.
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 creates and removes a user's current active role set (ARS).
Admin Tool	Allows server administrators to create users, roles, and permitted operations; associate users with roles and roles with permitted operations; specify constraints on user/role relationships; and maintain the RBAC Database. Administrators access the RBAC/Web Admin Tool by means of a Web browser.

Table 1. RBAC/Web Components

3.2 End-User Use Scenario

End-user interaction with a Web server enhanced with RBAC/Web is basically the same when requesting URLs whose access is not controlled by RBAC/Web (see Figure 3). 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 permitted operations 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.

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 ask 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, contain the largest subsets which do not violate any DSD relationships. Once the choice is made, the

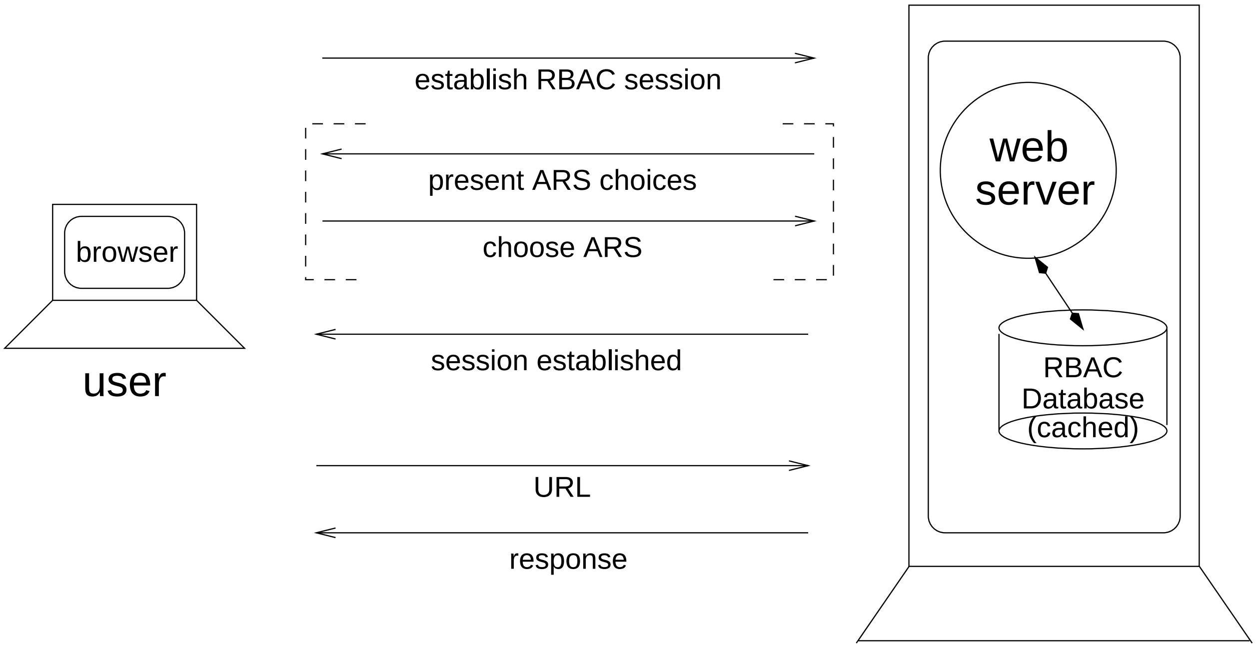

Figure 3: RBAC/Web Use

RBAC session is established with all authorized roles (i.e., assigned roles along with all roles which the assigned roles inherit) being placed in the ARS. 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.

Generally, an RBAC session requires an authenticated end-user. If authentication is removed from an end-user, access to RBAC controlled URLs is denied. However, end-user authentication and the establishment of an RBAC session are completely separate operations. This is so that RBAC Web can work with any authentication mechanism.

3.3 Consistency Principle

The RBAC model presented in the paper does not preclude the development of inheritance relationships and dynamic separation of duty relationships among roles where the two criteria presented above are in conflict. For example, R inherits R', and R and R' also share a dynamic separation of duty relationship. In this example, the question arises as to what should be placed in the ARS when a user who is assigned R establishes an RBAC session. Both relationships cannot simultaneously hold in the ARS.

The implementation answers this question by not permitting the role relationships in this example to be defined. In the implementation, the following consistency principle holds in all cases and at all times:

The inheritance, static separation of duty, and dynamic separation of duty relationships among roles are always consistent.

There are no scenarios or instances in the operation of RBAC/Web where one relationship overrides another. This principle assures the RBAC administrator that there are no "hidden" rules present in the implementation. At all times in the implementation's operation, the RBAC/Web Database is consistent and all role relationships always behave according to their definition in the RBAC Database.

The permitted operations implemented by RBAC/Web are the "methods" defined in the HTTP protocol definition. These methods are GET, HEAD, PUT, POST, etc. RBAC/Web controls the ability of a user acting in a role to perform an HTTP method on a URL.

4 Conclusions

For intranets to reach their full potential as a means for enterprise computing, access control 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/Web is to provide this access control service. This makes it possible to use 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 and emerging threat environment.

One of RBAC's greatest virtues is the administrative capabilities it supports. The administration of authorization data is widely acknowledged as an onerous process with a large and reoccurring expense. 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. With RBAC, users are not granted permission to perform operations on an individual basis, rather, operations are associated with roles. Role association with new operations can be established as well as old operations deleted as organizational functions change and evolve. This basic concept has the advantage of simplifying the understanding and management of privileges: roles can be updated without having to directly update the privileges for every user on an individual basis.

RBAC/Web provides the advantages of role based access control for intranet environments, and can be incorporated into existing systems with no modification to server code, making it portable to virtually all web servers. For more information on RBAC and RBAC/Web, see http://hissa.nist.gov/rbac.

Disclaimer

Because of the nature of this report, it is necessary to mention vendors and commercial products. The presence or absence of a particular trade name product does not imply criticism or endorsement by the National Institute of Standards and Technology, nor does it imply that the products identified are necessarily the best available.

References

[1] D. Ferraiolo and D.R. Kuhn. Role based access control. In *15th National Computer Security Conference*. NIST/NSA, 1992.

[2] D.F. Sterne. A tcb subset for integrity and role-based access control. In *15th National Computer Security Conference*. NIST/NSA, 1992.

[3] R. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role based access control models. *IEEE Computer*, 29(2), February 1996.

[4] S.H. von Solms and I. van der Merve. The management of computer security profiles using a role oriented approach. *Computers and Security*, 13(8), 1994.

[5] D. Ferraiolo, J. Cugini, and D.R. Kuhn. Role based access control: Features and motivations. In *Annual Computer Security Applications Conference*. IEEE Computer Society Press, 1995.

[6] R. Sandhu, E.J. Coyne, and C.E. Youman, editors. *Proceedings of the First ACM Workshop on Role Based Access Control*. ACM, 1996.

Appendix - RBAC Formal Description

This section summarizes the fundamental rules of RBAC (based on those in [5]) as used in RBAC/Web. RBAC is a mechanism that can implement a variety of policies, but separation of duty policies are often closely tied to RBAC models, because separation of duty is critical in most commercial applications, and because RBAC is a natural mechanism for implementing separation of duty. Two forms of separation of duty are described in this appendix.

Variables used are shown with their types below:

$s : subject$
$i, j : role$
$p : privilege$
$u : user$

The following definitions are used:

Subjects:
$U[s]$ = the user u associated with subject s
$R[s]$ = the set of roles for which subject s is authorized
$A[s]$ = the current list of active roles for subject s

Roles:

$M[i]$ = the users authorized for role i
$P[i]$ = the privileges that are authorized for role i
E = the set of role pairs (i, j) that are mutually exclusive with each other

Access to privileges:
$X[s, p]$ = true if and only if subject s can execute privilege p

The following invariants must be maintained by the RBAC system.
Consistent subject: relates human users to subjects executing on the users' behalf.

$$(\forall s, u, i) | U[s] = u : u \in M[i] \Leftrightarrow i \in R[s] \tag{1}$$

Role assignment: a subject can execute a privilege only if the subject has selected or been assigned an active role:

$$(\forall s, p) : X[s, p] \Rightarrow A[s] \neq \emptyset \tag{2}$$

Role authorization: a subject's active role must be authorized for the subject:

$$(\forall s) : i \in A[s] \Rightarrow i \in R[s] \tag{3}$$

Privilege authorization: a subject can execute a privilege only if the privilege is authorized for a role in which the subject is currently active:

$$(\forall s, p)(\exists i) : X[s, p] \Rightarrow i \in A[s] \land p \in P[i] \tag{4}$$

With (2) and (3), this rule guarantees that a subject can execute a privilege only if the privilege is authorized for that active role.

Role Hierarchy: Roles are organized into a partially ordered set (poset) so that if a role is included in the authorized or active role sets, roles below it in the poset are included also:

$$(\forall i, j, s) : (i \in A[s] \land i \succeq j \Rightarrow j \in A[s]) \land (i \in R[s] \land i \succeq j \Rightarrow j \in R[s]) \tag{5}$$

Separation of Duty

We define static separation of duty to mean that roles which have been specified as mutually exclusive cannot both be included in a user's set of authorized roles. With dynamic separation of duty, users may be authorized for two roles that are mutually exclusive, but cannot have both roles active at the same time. In other words, static separation of duty enforces the mutual exclusion rule at the time an administrator sets up role authorizations, while dynamic separation of duty enforces the rule at the time a user selects roles for a session.

Static separation of duty:

$$(\forall u, i, j) | i \neq j : u \in M[i] \wedge u \in M[j] \Rightarrow (i, j) \notin E \tag{6}$$

Dynamic separation of duty:

$$(\forall u, s, i, j) | i \neq j : u \in M[i] \wedge u \in M[j] \Rightarrow i \in A[s] \wedge j \in A[s] \Rightarrow (i, j) \notin E \tag{7}$$