DATA ACCESS CONTROL IN VIRTUAL ORGANISATIONS

Role-Based Access Control Patterns

Peter Bertok, Saluka R. Kodituwakku
Department of Computer Science
RMIT University, Australia
{ pbertok, sakoditu}@cs.rmit.edu.au
Tel: 61 3 9925 6124
Fax: 61 3 9925 6139

Abstract: Virtual enterprises bring together different companies under one umbrella, and the organizational structure is tailored to the common project rather than reflecting the participating companies' structure. A virtual organization is also dynamic in nature, as jobs/positions can be created or abolished as the project progresses. Access to information needs to be very flexible in such environment. On one hand, people should have access to all information they need to perform their duties, and those duties may change as time progresses. On the other hand, providing access to data other than needed for a particular job can lead to information overload and also poses a security risk, as it can lead to information leak or accidental modification of data. Role Based Access Control (RBAC) offers a solution to this problem by associating roles with jobs and assigning access privileges to roles.

This paper examines Role Based Access Control and proposes some modifications to conventional RBAC to make it suitable for virtual enterprises.

1. INTRODUCTION

Controlling access to data is one of the core issues in system security. Within a single company access to data is very closely related to the immediate work environment, e.g. designers have full access to their own design data while can not modify other designs. It is quite possible, however, that all designs can be freely viewed by all designers, as this can promote interaction between designers. In a virtual organisation, however, any type of

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35492-7_50
G. L. Kovács et al. (eds.), *Digital Enterprise Challenges*

access to any data is a potential risk. While members in a virtual organisation are working on a common project, each member company has its own set of product, design etc. data, and only a small subset of these data, that belongs to the common project, should be accessible by partner organizations. [1]

Virtual organizations are also dynamic in structure. Responsibilities are defined and re-defined as the project progresses, necessitating different types of access to data at various stages of the project.

Access control in general is concerned with restricting the activity of legitimate users in a system. It relies on and coexists with other security services, such as authentication that establishes the correct identity of the user.

Several access control models have been developed, such as discretionary access control (DAC), mandatory access control (MAC) and role based access control (RBAC). In the following we give a short overview of the different access control methods, and then we describe RBAC patterns that are particularly suitable for the complex organisation of virtual enterprises.

2. ACCESS CONTROL METHODS

2.1 Approaches

Access control consists of two main components. Policies govern user access to information, and mechanisms are used to implement access control. The same set of mechanisms can be used in different ways with different access control policies. With discretionary access control the decision to allow or deny access to information is based on user identity. Every user or every user group has access privileges on individual objects. Each request is validated against those privileges and only requests passing this authorisation test are granted access. DAC can provide great flexibility via access policies, e.g. users can grant or revoke access rights on objects under their control, and this flexibility can be very useful in many cases. However, it may also pose a security risk, as users can open up confidential documents to other users; e.g. setting wrong access privileges enables access to product, design etc. data by partners in a virtual organisation.

Mandatory access control overcomes this by nominating a security administrator who solely has the responsibility of granting and revoking access rights. MAC policies are considerably less flexible than DAC policies, and can enforce security hierarchies straightforwardly. On the other hand,

this rigidity can be a significant obstacle in a virtual enterprise, where partners need to respond to frequent changes swiftly.

Role based access control policies are based on activities of users. A role is a collection of tasks performed and responsibilities assumed in the organisation. A user acting in a certain role has to have all the privileges and access rights needed to perform the tasks attached to that role. With the increasingly widespread use of object-oriented technologies, the information accessed is often stored in the form of objects. [2][3][4]

When setting up the framework for RBAC, first roles are identified within the organisation, and then privileges are assigned to the roles. A user acting in a particular role is granted all privileges (also known as access rights) assigned to that role. Users may be able to assume several roles at the same time, or can be restricted to one role at a time. The structure of a RBAC system is shown in figure 1. RBAC implements two levels of association. First, access privileges are associated with roles, and second, roles are associated with users. A subject is a user acting in a particular role, and a user in multiple roles is represented by multiple subjects.

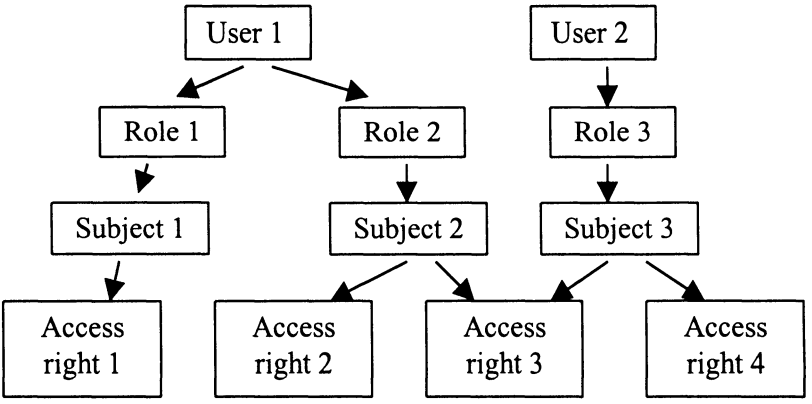

Figure 1. Associations between users and access rights

There can be mutually exclusive roles, which means that a user is not allowed to act in both roles; e.g. designers may not countersign their own designs. This is also known as separation of duties. Some roles have cardinality constraints; i.e. only a limited number of users can take that role at a particular point in time. [8]

The role-based approach has several important advantages, including the following.

- It implements separation and logical independence of user-role and role-privilege assignments.
- Role hierarchies can be used to support roles with overlapping responsibilities and privileges. A role hierarchy can contain non-overlapping sub-roles at a lower level, and upper-level roles can contain different, sometimes overlapping combinations of lower-level roles.
- It can easily implement the principle of least privileges, i.e. a subject acting in a role has no more privileges than those needed to act in that role.

2.2 Administration of privileges

Administrative policies control the allocation and modification of privileges. In DAC a security administrator assigns and withdraws user rights. Users then can control access to objects they own. In MAC privileges are based on security classification of subjects and objects. A security administrator assigns security levels to users, and the security level of an object is determined by the security level of the subject that owns it. It is the administrator only that can change the security level of subjects and objects. In RBAC the management of user privileges can be one of the roles. This administrative role, however, is different from the common roles it administers.

Delegation of responsibilities often makes administrative work easier. DAC allows this by letting users control access to objects they own. MAC is not really suited for delegation of responsibilities. RBAC can easily implement delegation of responsibilities by simply creating additional administrative roles with lower privileges.

3. ROLE BASED ACCESS CONTROL PATTERNS

3.1 Building blocks

RBAC is made up of several modules. [6] In the following we describe the building blocks that help to set up RBAC in a particular application environment.

Role-definition, privilege assignment

The first step is identification of job titles and responsibilities, and then a set of roles and privileges have to be assigned to each job title. Initially, roles are derived from job titles and privileges from responsibilities. When a user

is assigned a particular role, it means that the user is given all the privileges and allowed all the operations that belong to that particular role.

Role administration

Administrative and common roles are similar in structure, but differ fundamentally on the objects they operate on: administrative roles operate on roles and privileges while common roles operate on ordinary data. To avoid any interference that may compromise security, administration roles are distinct from user roles i.e. only administrators can assign and revoke roles and permissions, and administrators are restricted to perform only these functions. Users are expected to keep a strict separation of administrative and common roles.

In a complex system there can be several administrators, with different, sometimes overlapping responsibilities. E.g. some administrators may assign, update and revoke roles and privileges, others may only manage user-role assignments; or certain administrators can be restricted to a particular domain and manage only a particular set of roles, etc. In fact, separation of authorization, i.e. role-privilege assignment from user-role assignment results in a more easily manageable system. This type of relationships between roles can be part of an administration role hierarchy, where upper-level roles can be combinations of lower-level roles and additional responsibilities.

Administrators may administer common roles and lower-level administration roles. Notwithstanding, administrators should never be able to increase their own rights under any circumstances.

Access control

After the administrator has assigned roles to users, the system must validate user requests against privileges assigned to roles: operations allowed for any of the user's roles can proceed while requests for other operations are denied. A straightforward approach to this problem is to provide a common interface to all operations, and individual operations are accessed via this common interface; in object-oriented terminology all operations inherit the common interface. The common interface implements access control, and the actual functionality of the operation is implemented in a derivative class.

This hierarchical implementation of operations ensures that all operations use the same access control mechanisms, and significantly simplifies the set-up and maintenance of role-privilege assignment by security administrators.

Object collections

In object-oriented systems the information is stored in form of objects. There can be any number of operations defined on these objects, and each operation can be permitted or forbidden for an individual role.

As the number of objects can be very large, it is very useful to combine smaller objects into larger aggregates, and define roles and privileges that

affect the whole aggregate. This aggregation should be transparent to the user.

Roles as proxies

A consistent way of implementing RBAC is when roles access data and perform operations on behalf of the user, i.e. roles act as proxies for users. This simplifies the administration of privileges, as any role-privilege assignment is performed only once, and each instance of a role will have all privileges of the given role without any additional effort. A proxy role can be created only if the user has been assigned to that role.

A proxy role forwards any requests from the user to the actual operation. As actual operations are invoked via a common interface, an access request coming from a proxy role can easily be assessed within the common interface and allowed to proceed or denied, depending on the privileges of that role.

Role classification and access validation

The validation of a request consists of two steps. First, the privileges of the role requesting the operation are retrieved. Then the privileges are checked against the requested operation: permitted operations proceed, others are rejected.

Classifying roles and identifying common elements can be used for setting up role hierarchies and for rationalization of common functionality. [5]

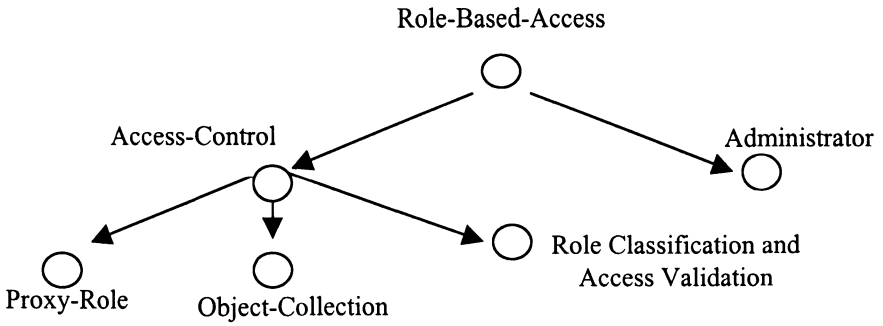

Figure 2. RBAC Components

3.2 Static and dynamic properties

Static properties form the skeleton of a RBAC system. Assignments that don't change very often, such as role-privilege assignment, and properties

related to them are considered static. They are specified at role authorization time, and maintained throughout the role activation. There can be constraints of static nature as well, such as limiting the number of users in a particular role, also called cardinality constraint, and static separation of duty that describes roles that can not be taken by the same user.

Traditionally, properties referring to rules that apply while a role is active are called dynamic properties. Most of the static properties have a dynamic equivalent; e.g. dynamic separation of duty refers to roles that cannot be active by the same user concurrently.

We also introduced a new feature, which refers to the history of the user with regards to objects. If that is applied to separation of duty, it means that a user can not assume a certain role when dealing with a particular object. This can be easily understood by using an example in the area of design: a design may not be countersigned or authorized by the same person who actually produced it. However, without the history constraint, it would be easy to promote a designer to chief designer, who can then authorize all designs including his or her own.

4. A ROLE BASED ACCESS CONTROL IN PRACTICE

4.1 Roles and rules

RBAC can be implemented in several ways. In the following we present a pattern, in which two companies, company A and company B team up in a virtual enterprise to design and sell a particular product. Company A produces a piece of hardware that includes a common-off-the-shelf (COTS) computer, and company B produces software that will run on the computer. The team consists of the software and hardware designer groups, a sales representative and the manager of the manufacturing workshop and the manager of the software development group. Each designer group has a chief designer. Additionally, there is a project manager responsible for the whole project.

We consider the design phase of the project, which includes the following operations: (1) start a new design (2) delete an existing design (3) read (look at) a design (4) write (save) a design (5) modify an existing design (6) approve / reject a design.

A sample set of rules and constraints were applied, which consisted of the following:

(1) designers can write their own design

(2) designers can have a look at any design of their own group

(3) project manager and designer are mutually exclusive roles

(4) a person can be designer and chief designer but not at the same time

(5) the chief designer can have a look at any design, and can modify them.

(6) the chief designer has to sign off any design except those that he designed when he was an ordinary designer

(7) sales representative, manufacturing manager and software development manager have read only access to design data; if they want some modifications, e.g. for manufacturability reasons or to convey some request from the customer, they have to ask the designer to introduce those changes.

4.2 System architecture

In software development it is very common that data files are stored in a central repository, and users need to check out files if they want to work with them. This method offers an obvious point for implementing access control. When a user checks out a file, the access control system checks the user's role(s), and access privileges are attached to the file. In this way a user can get a read-only file or can get a file that can be read and written, or access may be denied altogether. When a designer wants to provide access to a particular design for other people, the file is checked in, i.e. a copy of the data file is put into the repository. This repository is shared with an existing Unix file versioning system, such as CVS, which also has the feature of maintaining a full design history. A versioning system like CVS can provide additional benefits, like checking for conflicts during check-in if two users had write access to the same file, but interpretation of data can be difficult. [7] From an architectural point of view the role based access control system constitutes a middle layer between the CAD systems used by the designers and the file system of the computer in use, as shown in figure 3.

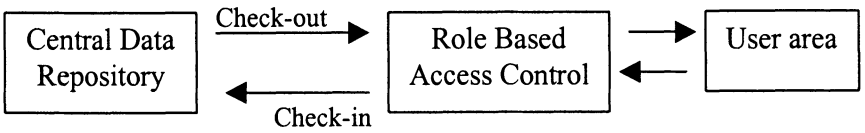

Figure 3. System Architecture

4.3 System elements

The RBAC system maintains a history of accesses to each file, and also inserts this information into each data file in form of a header. This header includes file ownership, a history of modifying authors, and the authors' roles at the time of modification. A sample header is shown in figure 4.

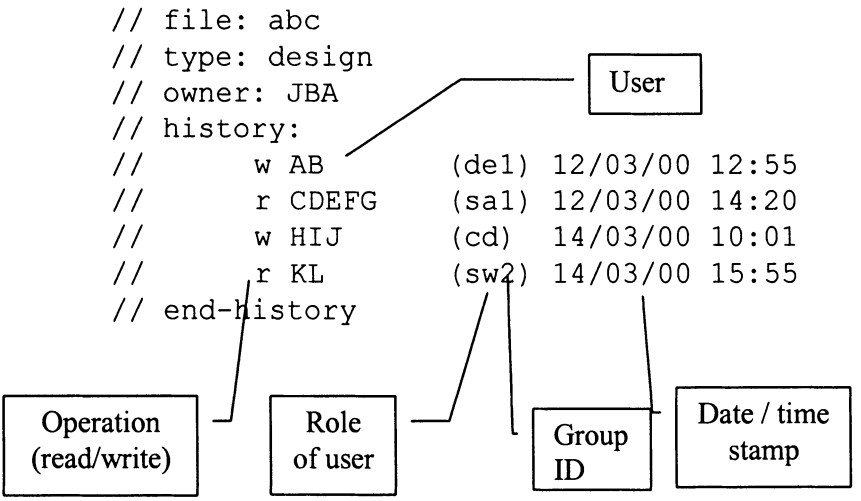

Figure 4. History Included in the File Header

The system stores the rules in the form of data, as opposed to code, which provides great flexibility as rules can easily be modified. Traditional RBAC systems implement rules in code, which means the source code needs to be modified, re-compiled etc. in order to change the rules.

The RBAC system maintains a simple user-role assignment table, and the operation – read or write – has to be specified at checkout. User ID and date/time stamp can easily be obtained from the operating system.

The rules are stored in a simple format. One group of rules describes access rights to objects, as shown in figure 5.

```
file-type: design
owner:   r w
group:   r
cdesign: r w
sales:   r
manuf:   r
sw:      r
```

Figure 5. Access Rules for a Specific Object Group

Another group of rules describes role hierarchies from the prospect of privileges; i.e. a role has all the access privileges of all other roles that are on its 'includes' list. A third group describes 'separation-of-duties' rules. Examples of these are shown in figure 6.

When a user requests a file, RBAC establishes the user's identity. Then it ascertains the user's roles, and the rules relating to those roles. Subsequently it validates the request, and if no conflict is found, the file is checked out and passed on to the user with the appropriate read/write privileges. Requests that would violate existing rules are rejected. The system also keeps a log of checked-out files, mainly for error-detecting purposes.

```
cdesign includes          design
manager includes          cdesign, sales

manager excludes(static) cdesign
design  excludes(dyn)    cdesign
cdesign excludes(hist)   design
```

Figure 6. Relationship between Roles.

4.4 Implementation

The RBAC system consists of three main modules, the Rule Management module, the User-role Management module and the Access Control module, as shown in figure 7.

RBAC maintains two databases. One contains the rules, the other contains user-role assignment data. When a request comes from a user to access a particular data file, the Access Control module (1) retrieves the relevant data from the databases, (2) evaluates the request, and (3) sends an appropriate answer to the user requesting the file. In step (1) the Access Control module determines the user's identity, retrieves the user's roles form the User-role database, then extracts the rules associated with that role from the rule data, and finally retrieves the history of the file. In step (2) it reads the rules associated with the roles, and checks them against the data in the history file. If the request passes the evaluation process, the Access Control system checks out a copy from the repository and passes it on to the user. The access privileges are set at this point, i.e. RBAC checks out a read-only copy or a read-write copy, as required. If the request fails the test, an error message is sent to the user.

When a user wants to check in a modified file into the repository, RBAC first checks if the file was checked out for writing. If it was, then it updates

the file header from the history file kept in RBAC, i.e. inserts information about the user's name and role and puts in a time stamp. Then the file with the updated header is checked into the repository.

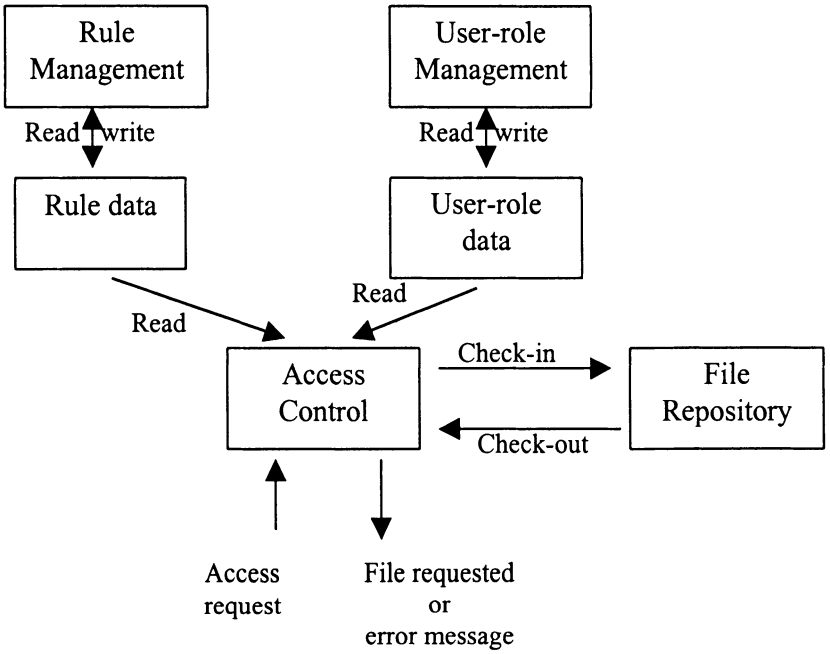

Figure 7. System Components

5. SUMMARY

This paper examined the use of role based access control in virtual enterprises. It was found that RBAC is suitable for such applications, and has some very distinctive advantages, as follows.

- There is a great flexibility in assigning roles to users, and user roles can be changed, users can be promoted or demoted, easily.
- RBAC provides a clear separation of role management and everyday task management. The roles of privilege assignment to roles and the role assignment to users are separate, and can be performed by different administrators.

- Implementing RBAC over a traditional operating system / file system is very straightforward and can easily be coupled with other, existing resources, such as CAD systems used in the enterprise.
- Features not used in conventional RBAC, such as rules relating to object history are needed in virtual enterprises.

An RBAC pattern was also examined. It was found that

- a RBAC system can be straightforwardly implemented as a middle layer between users and accessed files,
- in addition to user-role and role-rule databases a history of each file has to be maintained for assessing object-history-based rules.

6. REFERENCES

[1] W. Essmayr, E.Kapsammer, R.Wagner, G.Pernul, A.M.Tjoa: Enterprise-Wide Security Administration, Proc. of the 8th Int. Workshop on Database and Expert Systems Application, IEEE Computer Press, 1998,pp 267-272

[2] D.F.Ferraiolo, J.A.Cugini, D.R.Kuhn: Role Based Access Control: Features and Motivations, 1995, Proc. Of the 11th Annual Computer Security Applications Conference.

[3] S.I.Gavrila, J.F.Barkley: Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management, 1998, Third ACM Workshop on Role-Based Access Control

[4] W.A. Jansen: A Revised Model for Role Based Access Control, NIST-IR 6192, July 9, 1998

[5] W.A. Jansen: Inheritance Properties of Role Hierarchies, 21st National Information Systems Security Conference, October 6-9, 1998, Crystal City, Virginia

[6] S.R.Kodituwakku, P.Bertok, L.Zhao: A Pattern Language for Designing and Implementing Role-Based Access Control, KoalaPlop The Second Asian Pacific Conference on Pattern Languages of Programs, 29 May –1 June 2001, Melbourne, Australia

[7] A.Lam: Collaborative Design in a Distributed Environment and Object Versioning, Honours thesis, RMIT Australia,1996

[8] R.S.Sandhu, J.C.Edward, L.F.Hal, E.Y.Charles: Role-Based Access Control Models, IEEE Computer, Vol.29 February 1996, pp38-47