International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307, Volume-2 Issue-1, March 2012

A Study of Role based Access Control policies and Constraints

Miriyala Markandeyulu, Bussa V.R.R.Nagarjuna, Akula Ratna Babu, A.S.K.Ratnam

***Abstract*— Access control policies are constraints that protect computer-based information resources from unauthorized access. Role-Based Access Control (RBAC) is used by many organizations to protect their information resources from unauthorized access. RBAC policies are defined in terms of permissions that are associated with roles assigned to users. A permission determines what operations a user assigned to a role can perform on information resources. Role-based access control (RBAC) is also a powerful means for laying out higher-level organizational policies such as separation of duty, and for simplifying the security management process. One of the important aspects of RBAC is authorization constraints that express such organizational policies. This paper presents an overview of Role- based access control policies and constraints.**

***Index Terms*—Constraints, RBAC, Policies, UML.**

I. INTRODUCTION

Employing access control mechanisms in medium to large scale organizations always has been crucial. One of the challenging jobs for security-critical organizations, such as financial institutes, hospitals, and military is to control access to system resources at the highest level without violating the underlying access control policies. The research in recent years has brought role-based access control (RBAC) [1,2] as an efficient and flexible model for controlling access to computer resources and enforcing the organizational policies.

Access control policies are constraints that protect computer-based information resources from unauthorized access. Role-Based Access Control (RBAC) is used by many organizations to protect their information resources from unauthorized access. RBAC policies are defined in terms of permissions that are associated with roles assigned to users. A permission determines what operations a user assigned to a role can perform on information resources. In addition, various kinds of constraints can be specified in RBAC. These includes separation of duty constrains, prerequisite constraints, and cardinality constraints. Separation of duty constraints are used to enforce conflict of interest policies. The concept of prerequisite roles is based on competency and appropriateness. Prerequisite constraints require that a user can be assigned to a role only if the user is already assigned to the role's prerequisites. Cardinality constraints can be used to restrict, for example, the number of users that can be assigned to a role, the number of roles a user can play, the number of roles a permission can be assigned to, or the number of sessions a user is allowed to activate at the same time.Work on formalization of RBAC policies has resulted in the development of new specification notations , but there is still a need for policy specification approaches that can be integrated with design techniques used in industry.

UML(Unified Modeling Language) is the de facto modeling language used in the software industry. UML is easy to use and understand and is also amenable to analysis. Other researchers [3] have also advocated the use of UML for specifying RBAC policies. For instance, Ahn and Shin [3] show how RBAC constraints can be expressed in UML using the Object Constraint Language (OCL) [4]. However, they do not provide a systematic modeling approach that can be used by developers to create applications with RBAC features. This paper presents an overview of RBAC policies and the related constraints.

II. ROLE BASED ACCESS CONTROL POLICIES

RBAC has gained much attention as an alternative to traditional discretionary and mandatory access control. It is an access control model in which the security administration can be simplified by the use of roles to organize the access privileges [5]. Early work on role-based access control goes back to 1988, when Lochovsky and Woo defined roles and organised them into a hierarchy . The basic idea of role-based access control is to include another level of indirection between the user to permission (or privileges) mapping. Roles thus break this mapping into two, the first part maps users to roles, while the second maps roles to privileges. This is illustrated in Figure 1.

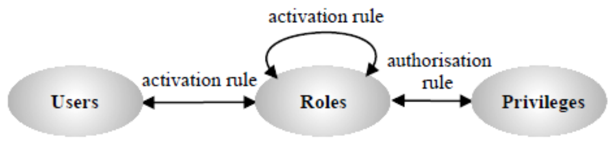

Figure 1: The basic RBAC model.

The indirection introduced by roles is very similar to one that can be expressed by groups, but while groups have only users as members, roles can form collections of users, permissions, and other roles . Also, unlike the case for groups, users can act in specific roles upon request, i.e. a user activates a role only when she needs the privileges associated with the role. Due to this dynamic behaviour RBAC can support the concept of least privilege , which requires the users to hold a minimum set of privileges that is

Miriyala Markandeyulu, M.Tech(CSE) Vignan's Lara Institute Of Technology & Science.Vadlamudi, Guntur, A.P., I ndia.
Bussa V.R.R.Nagarjuna, M.Tech(CSE) Vignan's Lara Institute Of Technology & Science.Vadlamudi, Guntur, A.P., I ndia.
Akula Ratna Babu, M.Tech(CSE) Vignan's Lara Institute Of Technology & Science.Vadlamudi, Guntur, A.P., I ndia.
A.S.K.Ratnam, Head, Dept.of Computer Science Engg,Vignan's Lara Institute Of Engineering & Technology,Vadlamudi,Guntur, A.P.,India.

necessary only for their current task, thus avoiding unnecessary, accidental, or malicious resource access.

A widely cited document [SCFY96] in the world of role-based access control distinguishes four kinds of RBAC models. The first model – called RBAC0 – is the simplest, it serves as a basis for the other three models. RBAC1 extends the basic model with role hierarchies. RBAC2 adds constraints to the basic model, while the consolidated RBAC3 model combines both RBAC1 and RBAC2, supporting both role hierarchies and role activation constraints. The relation between these four models is shown in Figure 2.

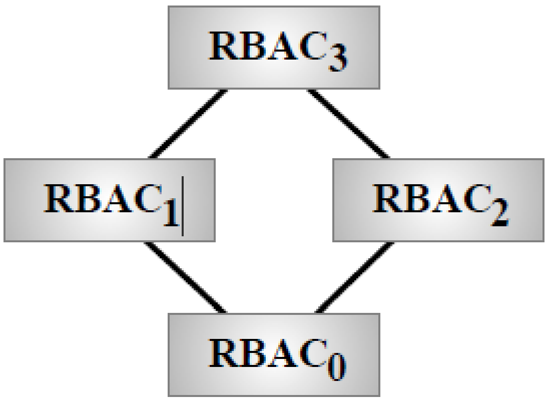

Figure 2: Relationship of the RBAC models.

RBAC is used to protect information objects (henceforth referred to as objects) from unauthorized users. To achieve this goal, RBAC specifies and enforces different kinds of constraints. Fig. 3 describes the general model of RBAC. RBAC has three components: base model, role hierarchies, and constraints. The base model embodies the essential aspects of RBAC. The base model requires that users (human) be assigned to roles (job function), roles be associated with permissions (approval to perform an operation on an object), and users acquire permissions by being members of roles. The base model also includes the notion of user sessions. A user establishes a session during which he activates a subset of the roles assigned to him. Each user can activate multiple sessions; however, each session is associated with only one user. The operations that a user can perform in a session depend on the roles activated in that session and the permissions associated with those roles.

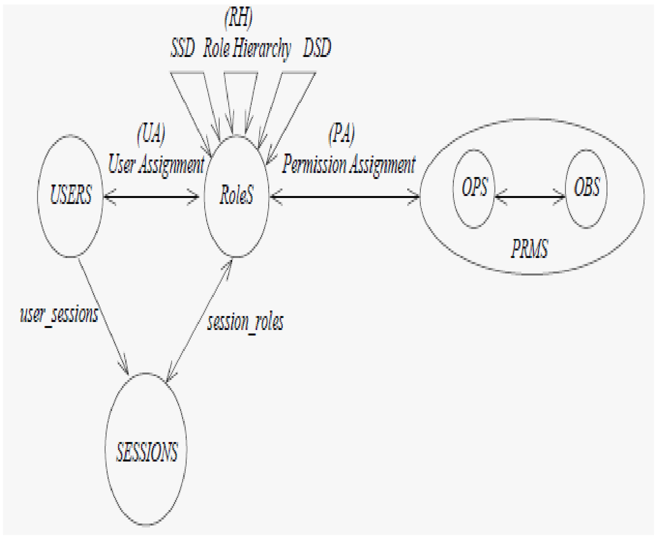

Figure 3: Different components of RBAC

The basic components of RBAC0 are users, roles, permissions, and sessions. RBAC0 includes a many-to-many permission to role assignment and a many-to-many user to role assignment. RBAC1 extends the basic model with role hierarchies. Role hierarchies were thought to be a natural way to describe role relations reflecting organizational structure. The permissions assigned to a junior role are inherited transitively by the more powerful senior roles. Thus role hierarchies define an inheritance relation among the roles in terms of permissions and user assignments. In other words, role *r*1 inherits role *r*2 only if all permissions of *r*2 are also permissions of *r*1 and all users of *r*1 are also users of *r*2.

III. ROLE BASED ACCESS CONTROL CONSTRAINTS

The RBAC2 model extends RBAC0 by adding constraints, which help to specify preconditions to role entry. Constraints are an important aspect of RBAC and are sometimes argued to be the principal motivation for RBAC. The common examples of RBAC constraints include static separation of duty, dynamic separation of duty, prerequisite roles, and cardinality constraints.

A. Separation of Duty Constraints

Separation of duty constraints are used to enforce conflict of interest policies. Static separation of duty (SSD) constraints aim to prevent conflict of interests that arise when a user gains permissions associated with conflicting roles (roles that cannot be assigned to the same user). SSD constraints are specified for any pair of roles that conflict. SSD constraints place constraint on the assignment of users to roles, that is, membership in one role that takes part in an SSD constraint prevents the user from being a member of the other conflicting role. We refer to this type of constraints as SSD Role constraints. SSD-Role constraints may exist in the absence of role hierarchies or in the presence of role hierarchies. A role hierarchy defines inheritance relationships between roles. Through the inheritance relationship, a senior role inherits the permissions of its junior roles and any user assigned to the senior role is also assigned to the junior roles.The presence of role hierarchies complicates the enforcement of the SSD-Role constraints: before assigning users to roles not only should one check the direct user assignment but also the indirect user assignment that occurs due to the presence of the role hierarchies.

Dynamic separation of duty (DSD) constraints aim to prevent conflict of interests as well. DSD constraints place restrictions on the roles that can be activated within the same user session. If one role that takes part in a DSD constraint is activated, the user cannot activate the other conflicting role in the same session. We refer to these types of constraints as DSD constraints.

B. Prerequisite Constraints

Prerequisite constraint specifies a set of prerequisites a user must satisfy upon entering a role. One such prerequisite can be the requirement for a user to hold a particular role. An example would be to require someone to have the university student role before allowing him or her to enter the computer science student role. Other prerequisites can include more complex predicate evaluations, such as ones that consider the time of the evaluation.

The concept of prerequisite roles is based on competency and appropriateness. Prerequisite constraints require that a user can be assigned to a role only if the user is already assigned to the role's prerequisites. We refer to such constraints as Prerequisite- Role constraints. This notion of prerequisite can also be applied to other elements such as permission in RBAC. The concept prerequisite permissions requires that a permission can be assigned to a role only if the role already possesses the permission's prerequisites. We refer to such constraints as Prerequisite-Permission constraints.

C. Cardinality Constraints

Another constraint type is cardinality constraints. Cardinality constraints can be used to restrict, for example, the number of users that can be assigned to a role, the number of roles a user can play, the number of roles a permission can be assigned to, or the number of sessions a user is allowed to activate at the same time. Cardinality constraints place constraint on the relationship between elements. They restrict the number of elements that can be related to each other. This numerical limitation may vary depending upon organizational policies. For example, we may have one type of organizational policies stating that there is at most one person in a role.

IV. UNIFIED MODELING LANGUAGE

UML is a general-purpose modeling language in which we can specify, visualize, and document the components of software systems [6]. UML has become a standard modeling language in the field of software engineering, and allows one to describe static, functional, and dynamic models of software systems. Here, we concentrate on the static UML models. A static model provides a structural view of information in a system. Classes are defined in terms of their attributes and relationships. The relationships include specifically associations between classes. In Figure 4, the static UML model for RBAC consisting of the RBAC classes and associations is depicted (UML class diagram).

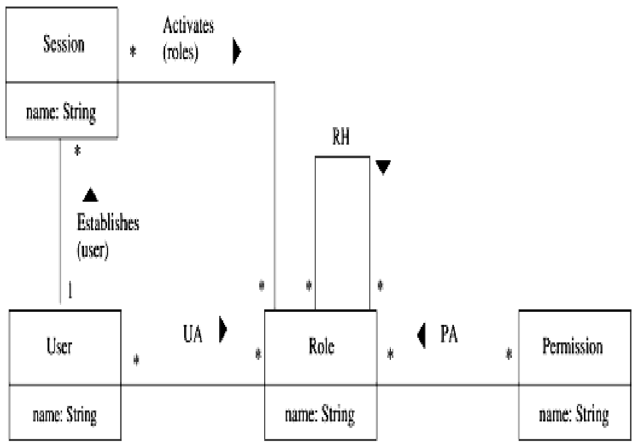

Figure 4: Class model for RBAC-entity classes.

A user can be a member of many roles and a role can have many users. Similarly, a role can have many permissions and the same permissions can be assigned to many roles. A user may activate a subset of roles he or she is assigned to in a *session*. The permissions available to the users are the union of permissions from all roles activated in that session. Role hierarchies can be formed by the *RH* relation. Senior roles inherit permissions from junior roles through *RH* (e.g., a *chief physician* inherits all permissions from the *physician*).

V. CONCLUSION

Organizations use Role-Based Access Control (RBAC) to protect information resources from unauthorized access. RBAC policies are defined in terms of permissions that are associated with roles assigned to users. A permission determines what operations a user assigned to a role can perform on information resources. Role-based access control (RBAC) is also a powerful means for laying out higher-level organizational policies such as separation of duty, and for simplifying the security management process. One of the important aspects of RBAC is authorization constraints that express such organizational policies. In this paper we present the policies and constraints of the RBAC model.

REFERENCES

1. R. Sandhu, E. Coyne, H. Feinstein, C. Youman. Rolebased access control models, IEEE Computer, vol. 29, no. 2, pp. 38–47, Feb. 1996.
2. American National Standards Institute Inc. Role Based Access Control, ANSI-INCITS 359-2004, 2004.
3. G.J. Ahn and M. E. Shin. Role-based authorization constraints specification using object constraint language. In Proceedings of the 10th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '01), pages 157–162, Cambridge, Massachusetts, June 2001.
4. J. Warmer and A. Kleppe. The Object Constraint Language, Second Edition. Addison-Wesley, 2003.
5. D.F. Ferraiolo, D.R. Kuhn, R. Chandramouli, Role-based access control, Artec House, Boston, 2003.
6. J. Rumbaugh, I. Jacobson, G. Booch. The Unified Modeling Language
7. Reference Manual, Second Edition. Reading, Mass., Addison Wesley Longman, 2004.

AUTHORS PROFILE

Miriyala Markandeyulu, Pursuing M.Tech(CSE) from Vignan's Lara Institute of Technology & Science, Vadlamudi, Guntur, A.P., India. My research Interests are computer networks and Mobile computing.

Bussa V.R.R.Nagarjuna, Pursuing M.Tech(CSE) from Vignan's Lara Institute of Technology & Science, Vadlamudi, Guntur, A.P., India. My research Interests are computer networks and Mobile computing.

Akula Ratna Babu, Pursuing M.Tech(CSE) from Vignan's Lara Institute of Technology & Science, Vadlamudi, Guntur, A.P., India. My research Interests are computer networks and Mobile computing.

A.S.K.Ratnam, Assoc.Professor & Head, Department of Computer Science Engineering at Vignan's Lara Institute of Technology & Science, Vadlamudi, Guntur, A.P., India..My research Interests includes Data Mining, Mobile Computing, Network Security.