Improvement on Role Based Access Control model

Wei Guan

1. Introduction:

Role Based Access Control (RBAC) has become a well-accepted and well-known approach for authorization and access control in modern systems. It regulates the access of users to the information on the basis of activities the users execute in the system. But instead of specifying all the access each user is allowed to execute, access authorizations on objects are specifies as roles. A role is usually a function using to categorize users within an organization, and is assigned to an appropriated set of permissions by the security administrator; while a permission can be thought as authority to perform an operation on one of the objects in the system. So when a user attempts to perform a operation on a target object, the access control system only allows it to proceed provided that this person is authorized to "play" the roles that include the necessary permissions for that operation. Main benefit of RBAC is the ease of administration on security policies, and its scalability; if a user moves to a new function within the organization, there is no need to revoke the authorizations he/she had in the previous function and grant the authorizations he/she needs in the new function; the security administrator simply needs to revoke and grant the appropriate role membership.

RBAC is capable of expressing policies especially suited for commercial applications.

- First, RBAC models include features to establish role hierarchies (where a given role can include all of permissions of another role), because roles within an organization typically have overlapping permissions, thus can easily mirror an organization's structure and encourages well-structured access control policies that make sense in the context of the organization.
- Second, RBAC policies change very little over time, because transaction permissions are associated with roles, not users and roles in an organization are relatively persistent with respect to user turn over and task re-assignment, thus providing a powerful mechanism for reducing the complexity, cost and the potential for error of assigning users permissions within the organization
- Third, it naturally supports delegation of access permissions. For example, if a physician is taking a vacation, he can temporarily delegate part of the permissions on patient information and treatment decision to an assistant physician by simply delegating the corresponded role to that person before leaving for vacation and revoking the role after his return to work.

In this proposal, I'll give a brief description on base RBAC model and Specification Language for it (section 2); and summary on the developments on this model (section 3) and in the last section, possible improvement or integration upon current RBAC with application in Healthcare field, and design of the corresponded specification language.

2. Preliminaries

2.1 base RBAC model:

Sandhu[2] gives formal Definition of base RBAC model as:

- U, R, P, and S represent the set of users, roles, permissions, and sessions, respectively
- PA: R→ P, the permission assignment function that assigns to roles the permissions needed to complete their jobs.
- UA: U→ R, the user assignment function that assigns users to roles.
- user: S→ U, which maps each session to a single user.
- role: S→ 2R, that maps each session to a set roles.

It can be shown that cost of administering RBAC is a factor of **U+P** while the cost of associating users directly with permissions is a factor of **U*P** where U is the number of individuals in a role and P is number of permission to perform that role.

Sandhu[2] also provides a characterization of RBAC models as follows:

- RBAC0: the basic model described above with users associated with roles and roles associated with permissions
- RBAC1: RBAC0 with role hierarchies, including limited inheritance which can be achieved by introducing private roles.
- RBAC2: RBAC0 with constraints on user/role, role/role, and/or role, permission associations.
- RBAC3: consolidation model which provides both role hierarchies and constraints, as it combines RBAC1 and RBAC2, thus arising issue in applying constraints to role hierarchy such as: the model should accommodate both possibilities when a senior role with two conflicting junior roles may or may not be acceptable, and cardinality constraints are applied only to direct membership or including inherited memberships.

2.2 Specification Language

Specification Language is formal language aimed to identify RBAC model and specify security policies and conflicts of interest policies in the role-based system by using a restricted form of first order predicate logic (RFOPL).

Separation of duty (SOD) is a fundamental technique for preventing fraud and errors using in RBAC. *RBAC96* did significant work on SOD analysis, but didn't support role hierarchies and missed the concept like session-based SOD, while *RSL99* (role based separation of duty language 1999) can automatically identify most SOD properties.

- **Static SOD (SSOD):**

Property 1: no user can be assigned to two conflicting roles.

Property 2: a user can have at most one conflicting permission acquired through role assigned to this user. however, this may generate roles which cannot be used at all.

Property 3: each role can have at most one conflicting permission without consideration of user-role assignment which eliminates the possibility of useless roles.

Property 4: conflicting permissions can only be assigned as conflicting roles, thus a user can have at most one conflicting permission via role assignment.

Property 5: two conflicting users cannot be assigned to roles in the same conflicting role set.

- **Dynamic SOD**

Property 1: conflicting roles may have common users but user can not simultaneously activate conflicting roles.

Property 2: session-based dynamic SOD by which no user can activate two conflicting roles in a single session

Another important aspect of RBAC is constraints which constrain role definitions in order to avoid conflicting access policies, promote separation of duties. *RCL2000* (role based constraint language) can specify these following constraints:

- **Prohibition constraints**: forbid the RBAC components from doing (or being) something not allowed based on organization policy. A common example is SOD, which requires that the same individual cannot be assigned to both roles which are declared mutually exclusive, e.g. if a user is assigned to cashier in a bank, he/she cannot be assigned to an accountant of this bank.
- **Obligation Constraints**: force RBAC component to do (or be) something allowed or required based on organizational policy. For example, in lattice-based access control, a user may be required to have certain combinations of roles in user-role assignment.
- **Cardinality constraints**: a numerical limitation for the number of users, roles and sessions. For example, only one person can fill the role of department chair; similarly, the number of roles (sessions) a user can belong to (activate) could be constrained.

3. Current Development on RBAC model:

Although RBAC is very useful for modeling access control in a variety of applications, its roles are inherently subject-centric, and it provides no support for distinctions between various objects or environment states based on their properties. Thus, it cannot capture security-relevant context from the environment which could have an impact on access decisions. Therefore researchers make efforts to modify RBAC to make it able to handle these problems while remain the simplicity of the basic model.

3.1 Temporal RBAC

Temporal-RBAC (TRBAC) introduced in [5] has formal semantics for specification language to support *periodic activations/deactivations of roles* (e.g. roles to access object O is only active between 9:00 am and 5:00 pm on weekdays), and *temporal dependencies among actions*, upon which actions may be either executed immediately, or be deferred by a specified amount of time (e.g. a nurse-on-night-duty role is active whenever the doctor-on-night-duty is active), expressed by ***role triggers***. While triggers and periodic activations/ deactivations both have a priority associated with them, in order to resolve conflicting actions. However, the time factors are only defined on roles, or role activation events, so this model cannot express timing constraints in terms of protected objects, or relationships between roles and protected sources.

3.2 Content-based access control

Since the RBAC model does not specify whether a permission is applicable to a particular target object or to all instances of a class of objects, this is usually left to the application to decide and enforce. In most practical system, the number of objects requiring access control is huge, making it impractical to define permissions for accessing each of them. An alternative therefore is to define permissions in terms of operations on classes of objects (e.g. a permission may correspond to the Read operation on the Patient Record class). However, some ambiguity still remains, regarding which

instances of the class are made accessible to a user by a permission. For example, a patient role has the permission to read a patient record, but implicitly, only his/her own, whereas a doctor only has access to some records - those of his/her own patients. So implementation of RBAC needs to limit user's access to these implicit subsets of objects in application-specific manner. Mechanisms proposed can be categorized into:

1) **Enumeration:** involves a manual (and often static) specification of the subset of instances. In the Patient Record example, for each member of the Attending Physician role, the access control system may maintain a list of his/her patients.

- The model represents in [6] is able to activate and control permissions on individual users and object instance by maintaining a list of valid value for each security relevant property of the object classes in the system. Central concept of the system is the notion ***team***, which is a set of users in various roles need for a task. The user context and object context are defined for a team rather than a role. And team context, expressed in terms of ranges of values for certain security-relevant attributes, is used to restrict the object instances over which the permissions of a user apply.
- Barkley et al. [7] use concept of ***relationships*** to determine whether a user U's role R, which applies over an object type O, has an active relation with the particular instance of O. The requested operation is allowed only if the set of active relations between U and O contain the one required by the access control policy. However, in their implementation, they still use enumeration as means to capture the relationships information as each instance maintains a list of authorized users.

2) **Object grouping:** involves creating groups of instances of the same class. A role-member acquires the associated permissions only for a specific set of such groups. This technique is more efficient and scalable than Enumeration, since a list of group is maintained rather than a list of individual instances. Grouping criteria may be arbitrary, and the grouping process is usually done manually, or outside the scope of the access control system.

- Generalized RBAC model [8] employs notions of ***subject roles*** (traditional RBAC roles for users), ***object roles*** (groups of instances computed algorithmically on the basis of certain security-related properties of target objects by procedure *transducer*) and ***environment roles*** (access control relevant information from the environment like time, etc). But policy administrator may need to specify on some properties of target, thus having to manually define some object roles, resulting in loss of scalability. Furthermore, this model supports separate policies for users and objects, which lead to potential conflicts, and requires extra support that adds overhead.

3) **Constraints:** conditions that an object must satisfy in order to perform an operation, which involve security-relevant parameters of the attempted operation, including information gleaned from environment (such as time, whether it's holiday), or state contained in the target object (e.g. bank accounts' overdrawn status, etc.). These constraints are distinct from those defined in the base RBAC.

- Constraints can be in the form of environment roles, which are dependent on external properties rather than properties of the objects/subjects involved in the operation. [9]

.

Recently, Context Sensitivity RBAC (CS-RBAC) model proposed in [10] is intended to provides role context to decides whether a role's permissions are valid for a given user-object pair. The access control system captures security information about a particular user and the target object respectively, and then compose role context from user and object contexts by ***Context Filter*** (a Boolean constraint expression whose operands are the attributes available in the user and object contexts while operators include the standard comparison and logical operators). However, system administrator needs to specify the attributes and role context at the time of role creation.

4. Propose and future work

As I listed in the previous sections, the base RBAC model and its current existing extensions have limitations which prevent them from being able to fully specify all the security requirements in the healthcare domain. We want to extent the RBAC model with a single construct that is powerful enough to encompass the current developments by adding context as one of parameters in deciding role access permissions. This new context dependent RBAC model will consist of five basic types of elements [1]:

- *Entity,* including users, roles, objects (in form of (protected data, time duration, creator u1, owner u2) or their combinations.
- *Relationship* is defined on entities, such as role hierarchy, static separation of duty, dynamic separation of duty, and cardinality constraints in base model.
- *Context* is a collection of state information that may impact security decisions, in form of (location, time duration, relationship between entities).
- *Permissions* are approvals to perform an operation on one or more protected entities.
- *Assignment Functions* contain role assignment function: AR: U → R and permission assignment function: AP: R*C*O → P.

In order to control the complex contextual information, we limit context information to the health-care domain. However, we believe that this model is generic and can be applied in other application domains since the relationships between roles, users and controlled objects can be defined in an adjustable granularity. Also it can be shown that, examples such as affiliation of user, location and time constraints, relationship among entities in this field, can be clearly represented by the model. Furthermore, this model degenerates to the base RBAC model when not concerning engaging context.

We also plan to provide a specification language based on RCL2000, which isolate the security concerns from other function requirements, and allow security specification be independent from system architecture and application's functional requirements. Such separation will not only mitigate the complexity of building security systems, but also provide more software reusability in both security and other functional aspects.

- **Current work need to be done:**

1. We may extend the model to deal with circumstances in Healthcare domain which the current variations of RBAC didn't support (in case we can find some examples in this period), while keep covering the current developments on RBAC, like time-dependent access control, role context obtained by context filter from user and object context.

2. We may do some modifications on current definition of the model after compare the increase on administrator cost, since new parameters are added into the model, with its capability in describing access control policies and detecting inconsistency.

3. Our model should include ability to check a set of specifications to detect inconsistent and infeasible security policies before deploying specification or during running time, and we are investigating on giving a formal semantics for it.

Reference

[1] Shangping Ren. "a Model and a Specification Language for Context Dependent information Access Control in Healthcare domain", proposal at IIT, Mar. 2004

[2] R. S. Sandhu, E.J.Coyne, H.L.Feistein, and C.E. Youman. "Role Based Access Control Models". IEEE Computer, Vol.29, No.2, page 38-47, Feb. 1996

[3] Gail-Joon Ahn, Ravi Sandhu. "the RSL99 Language for Role-based Separation of Duty Constraints", ACM, 1999

[4] Gail-Joon Ahn. "Specification and Classification of Role-based Authorization Policies", In proceedings of the 12th IEEE international Workshops on Enabling Technology, 2003

[5] Bertino E., Bonatti P., Ferrari E. "TRBAC: A Temporal Role-based Access Control Model". ACM Transactions on Information and System Security, 4(3), 2001

[6] C.K. Georgiadis, I. Mavridis, G. Pangalos, and R.K.Thomas. "Flexible Team Based Access Control using Contexts", In proceedings of ACM RBAC97, 1997

[7] J. Barkley, K. Beznosov, and J. Uppal. "Supporting Relationships in Access Control Using Role Based Access Control" In proceedings of 4th ACM workshop on Role-based access control, Oct. 1999

[8] M.J.Moyer and M. Ahamad. "Generalized role based access control", In Proceedings of the International Conference on Distributed Computing Systems, Oct. 1997

[9] M. J. Covington, W.Long, S. Srinivasan, A.K. Dey, M.Ahamad, and G.D.Abowd. "Securing Context-Aware Application Using Environment Roles" In proceeding of SACMAT, May 2001

[10] Arun Kumar, Neeran Karnik, Girish Chafle, "Context Sensitivity in Role-based Access Control", IBM India Research Laboratory.

[11]Marc Wilkens, Simone Feriti, Alberto Sanna, Marcelo Masera, "A Context-related Authorization and Access Control Method based on RBAC: a case study from health care domain". SACMAT, Jun. 2002