A Role-Based Delegation Model and Some Extensions

Ezedin Barka and Ravi Sandhu
Laboratory for Information Security Technology
Information and Software Engineering Department, MS 4A4
George Mason University, Fairfax, VA 22030, USA
{ebarka,sandhu}@ise.gmu.edu
www.List.gmu.edu

Abstract

In Role-based Access control (RBAC) permissions are associated with roles and users are made members of roles thereby acquiring the associated permissions. User delegation in RBAC is the ability of one user (called the delegating user) who is a member of the delegated role to authorize another user (called the delegate user) to become a member of the delegated role. This paper proposes a simple but practically useful model for delegation called RBDM0 (role-based delegation model zero). The paper also explores some extensions to RBDM0 including issues of revocation, partial delegation, multiple step delegation, and delegation with hierarchical roles.

1 Introduction

Role-based access control (RBAC) has received considerable attention as a proven alternative to traditional discretionary and mandatory access control [FCK95, SCFY96, San97]. In RBAC permissions are associated with roles. Users are made members of appropriate roles based on their responsibilities and qualifications, thereby acquiring the permissions of these roles. Users can be easily reassigned from one role to another. Roles can be granted new permissions as new applications and systems come online, and permissions can be revoked from roles as needed.

In this paper we explore the concept of delegation in context of RBAC. The basic idea of delegation is that some active entity in a system delegates authority to another active entity to carry out some functions on behalf of the former. Delegation in computer systems can take many forms: human to human, human to machine, machine to machine and perhaps even machine to human. In this paper we focus on the human to human form of delegation in computer systems. Specifically we consider the ability of a user in a role to delegate his role membership to another user who belongs to some other role.

We develop a simple but practically useful model for delegation called RBDM0 (role-based delegation model zero). We motivate the model and give a formal definition for it. We explore some possible extensions to the original model that will add some complexity. These include issues of revocation, partial delegation, multiple step delegation, and delegation with hierarchical roles.

To appreciate the motivation behind role-based delegation, consider the roles in figure 1 from a hypothetical computer science department in a University. An intuitive

scenario to illustrate delegation would be to have a professor give his key for his office to a secretary to do some filing or allowing his teaching assistant to administer an exam or to grade a homework. Another scenario is to have a guest speaker from outside of the school faculty substituting for the original assigned professor. All of these activities are considered delegation simply because in each case an original member of a role is delegating his/her role membership to someone else to perform some task on his or her behalf. This can benefit the overall interests of the organization by letting the work continue even in the absence of the original member of that role. These types of activities have to be monitored and controlled in such a manner so that the resource inside the organization can stay protected. For example, in figure 1 a professor could be permitted to delegate the professor role to a secretary or a teaching assistant but not to a student. Also, a teaching assistant who is given the key to a professor's office is not allowed to further give the key to someone else (this is called one step delegation).

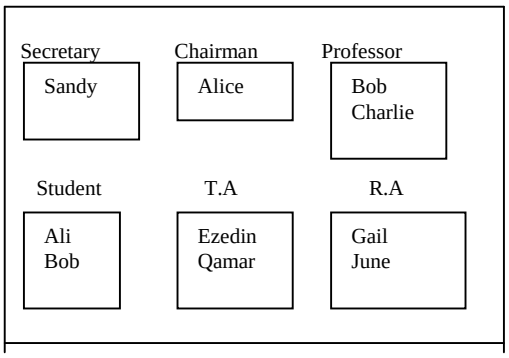

Figure 1: An example of Computer Science Department Roles

The rest of the paper is organized as following: We begin by giving a background and review of delegation and of the RBAC96 model in section 2. In section 3 we define and explain in full detail the simple role-based delegation model (RBDM0). Section 4 introduces possible extensions to RBDM0. Section 5 concludes the paper.

2 Background and review

2.1 Delegation

There are many forms and definitions of delegation in the literature. Most commonly delegation has been studied as human to machine and machine to machine delegation [Glad97], [ABLP96], [GM90], [VAS91]. Models for propagation of access rights also relate to delegation indirectly (e.g. HRU, TAM, SPM, and the Take Grant model) [HRU76], [San97], [Lamp71]. The scope of our model (RBDM0) is to address the human to human delegation, whereby a user in a role (delegating role) delegates his role membership to another user in another role (delegate role). This type of delegation has not been discussed in the literature so far. This paper is the first attempt to model delegation involving user to user based on roles.

2.1.1. Work that is directly related to delegation

Gasser and McDermott [GM90] defined user to machine delegation as "the process whereby a user in a distributed environment authorizes a system to access remote resources on his behalf." The user's authorization of his process to act on his behalf is a form of delegation of rights from the user to the process. In some cases the user may delegate the rights to one of several permissible roles or identities (e.g., by logging in using different names and/or passwords), in order to limit the actions of the process to some subset that user is authorized. Limited delegation also occurs routinely in multilevel secure systems where the user selects a single classification of his process that is a subset of the access class for which the user is authorized [GM90].

Gladny [Glad97] considered the security requirements for a digital library that emulates massive collections of paper and other physical media for clerical, engineering, and cultural applications. He proposed an access control method that mimics organizational practice by combining a subject tree with ad hoc role granting that control privileges for many operations independently. Scaling to many users is accomplished by emulating vertical delegation in organizational hierarchies, extended to permit privilege delegation from any to any other node, up, down, or across the organization tree; this provided a way to represent special administrative roles like security officers. A driving objective is that every privilege should be traceable as a sequence from a custodial user.

Varadharajan et al [VAS91] consider the essence of the delegation problem to be the verification that an object that claims to be acting on another's behalf is indeed authorized to act on its behalf. In practice this means that we need to ensure that the information is securely transferred between the objects (process to process delegation).

2.1.2. Work that is indirectly related to delegation

Propagation of permissions can also be considered as delegation. A large number of papers have been published in this area. Some of the well known models are: HRU, SPM, TAM, TG [HRU76], [San97], [Lamp71].

2.1.3 Scope

The scope of our work is to address user-to-user delegation based on RBAC. We will base our work on RBAC0 of the RBAC96 family of models [SCFY96]. This means that we will consider only flat roles. Extension to delegation with hierarchical roles is discussed as an extension in this paper. We chose this approach in order to work out a simple but useful model in complete detail and then gradually introduce extensions to add functionality in an incremental manner.

2.2 RBAC0-Flat roles

Our work is cast within the framework of the well-known RBAC96 model [San97]. We use the simplest form of this model, called RBAC0, as summarized in figure 2. A user is a human being, a role is a job function and permission is an approval of access to some objects or a privilege to carry out a particular task. The management of permissions and roles is greatly simplified by associating permissions with the roles and assigning the users to roles. In this way the users acquire the associated permission. Roles are created for various job functions in an organization. The permissions required to carry out the jobs are associated with the roles. New permissions can be granted to roles as new applications and systems are incorporated. Unnecessary permissions can be revoked from the roles. Users are assigned to the roles depending on the responsibilities and qualifications and can be reassigned from one role to another. (The session concept of RBAC96 is not used in our work and is hence omitted here).

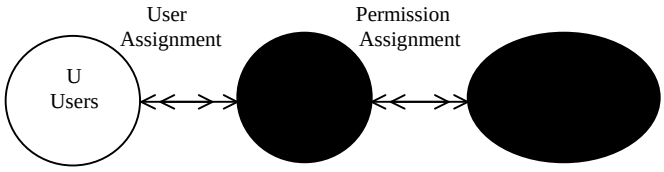

Figure 2: Simplified version of RABC96

3. RBDM0-Flat roles

This model is the simplest form of the RBDM model and is based on RBAC0 of the RBAC96 family. This means that the delegation addressed in this section is between users in flat roles (no inheritance of permissions between

roles is involved). First we give some assumptions and basic elements that we will use throughout the paper.

3.1 Assumptions and basic elements

Delegation between members in the same role is not allowed because it is meaningless. This assumption is very basic and it will not be relaxed throughout the paper.

The delegation addressed in this model is a one step delegation. This means that the delegated role can not be further delegated. Hence, only the original members can delegate. We will keep this assumption in our original model, but as part of our extended model we will relax this assumption and extend the model to allow some multi-step delegation.

The delegation is total. Each user in a delegating role delegates the total package of permissions embodied in that role or does not delegate at all. This assumption will also be relaxed later when we extend our model to include partial delegation, where we will distinguish between delegable and non-delegable permissions.

Each delegating role r has two types of members:

- Original members Users_O(r) are the members which originally assigned to the role by the system administrator

- Delegated members Users_D(r) are the members which are assigned to the role by other original members (that is assigned by delegation)

To simplify revocation we assume in our basic model that any original member in a role can revoke the delegation of any delegate member in that role. That is revocation is not related to who did the delegation. As we will see revocation is one area which presents many different policy choices, some of which will be explored in the extensions discussed in this paper.

We assume each unit of delegation has a time element associated with it called duration (T). The duration of each delegation is under the control of the delegating user. Once the assigned time for the delegation expires, the delegation is automatically revoked. Revocation of delegation can also be exercised by original members in the delegating role even if the duration of delegation is still valid.

The following definitions formalize the above discussion.

Definition 1 The following is a list of the original RBAC96 components

- U and R and P are sets of users, roles, and permissions respectively
- $UA \subseteq U \times R$ is a many to many user to role assignment relation
- $PA \subseteq P \times R$ is a many to many permission to role assignment relations
- Users: $R \rightarrow 2^U$ is a function derived from UA mapping each role r to a set of users where $Users(r) = \{U \mid (U, r) \in UA\}$
- Permissions: $R \rightarrow 2^P$ is a function derived from PA mapping each role to a set of permissions where $Permissions(p) = \{P \mid (P, r) \in PA\}$

Definition 2 The RBDM0 model adds the following components:
- $UAO \subseteq U \times R$ is a many to many original member to role assignment relation
- $UAD \subseteq U \times R$ is a many to many delegate member to role assignment relation
- $UA = UAO \cup UAD$

- UAO ∩ UAD = ∅ Original members and delegate members in the same role are disjoint
- Users_O(r) = {U | (U, r)∈ UAO}
- Users_D(r) = {U | (U, r)∈ UAD}
- All members Users_O(r) ∪ Users_D(r) in a role get all the permissions assigned to that role
- Note that users_ O(r) ∩ users_ D(r) = ∅ because UAO ∩ UAD = ∅
- T is a set of durations
- Delegate roles: UAD → T is a function mapping each delegation to a single duration

3.2 Delegation

In RBAC96, the security officer handles assignment of users to roles [San96]. In RBDM0, the delegation from one user in a delegating role to another user in a different role is actually making the delegated user a member of the delegated role. Thus, the delegating user handles this function [Glad97]. In this paper our focus is exclusively on the user-user delegation. This function is a widely decentralized task that can be taken care of by the users themselves and without continuous involvement from the security officer.

User-user delegation is authorized in RBDM0 using the following relation.

Definition 3 RBDM0 controls user-user delegation by means of the relation can-delegate ⊆ R× R.
Can-delegate is irreflexive. This means that a user in a role cannot delegate his membership to another user in the same role, since this is meaningless.

The meaning of (a, b) ∈ can-delegate is that a user (say, Alice) who is an original member of role a can delegate her role membership to any another user (say, Bob) who is an original member of another role b. For example, If Alice ∈ User_O(a) and Bob ∈ User_O(b), then Alice can delegate to Bob, so thereby (Bob, a) ∈ UAD.

3.3 Revocation

So far we have described how users in a delegating role can delegate their permissions to others users in another roles and how we can control this processes using the can-delegate relation. However, as often happens in real life, we may want to revoke rights. In the examples described above, when the department chairperson goes away, one or more other professors will be delegated the chairperson's permissions. Subsequently, when the department chairperson returns, the delegated permissions need to be removed from the delegate professor. In this section we shall look at possible ways in which a user in a delegating role can change his mind and revoke the permission that he/she delegated. We will also consider under which conditions it is not possible to revoke a previous decision and the issues that might arise as a result of revocation.

3.3.1 Types of revocations:

RBDM0 deals with the issue of revocation in two ways: by using timeouts and by allowing any original member of the delegating role to revoke the membership of any delegate member in that role (Grant-independent revocation). The following two subsections describe both approaches and discuss the pros and cons for each approach.

3.3.1.1 Revocation using time out

In using this approach we attach a time clock to every assigned delegation so that when the assigned time expires, the delegation also expires. This approach has some advantages and some disadvantages.

Using timeouts has the following advantages:

- Timeout revocation is a simple self-triggering process that ensures the revocation of delegate membership automatically.

- In attaching a timeout to the delegation we no longer have to worry about tracking the sponsoring roles (the delegator).

Using timeouts has the following disadvantages:

- Timeouts by themselves are not enough to ensure security.

- If there is no other tracking mechanism, delegate members can behave in a bad manner during the duration of the time set which can cause great harm to the system before revocation takes place by time out.

- When employing this approach, we have to choose the time carefully, because we might overset or under set the time for delegate members

3.3.1.2 Grant-independent revocation

This type of revocation allows any original member in a delegating role to revoke the membership of any delegated member in that role. This gives the power to the original members to protect the role from the temporary delegate members, which can have some advantage and disadvantage. The advantage is that in the case where the delegate member behaves badly, any original member can revoke him immediately which will minimize the damage before even the time out.

The disadvantage on the other hand is that it raises the possibility of conflicts between the original members. This can occur if someone else other than the granting original member revokes the delegate membership.

There is no need to define a can-revoke relation in order to control the revocation of the delegated roles in a role by the original role members in that role because there is only one role (the delegating role) relevant to this process.

3.4 Summary of RBDM0

To summarize, the RBDM0 model has the basic elements given in definition 1 and 2, and authorizes delegation using the can-delegate defined in definition 3. Moreover, the model deals with the issue of revocation using the notions explained in section 3.3.

4 Extensions to RBDM0

This section explores the possible ways by which the model we described can be extended to address more complicated issues. The following is a list of possible extensions and brief description of the impact that they would have on the existing model.

4.1 Grant-dependent revocation

This means that only the delegating member is allowed to revoke the role he delegated.

Adding this extension to our model means that no other member in any role can revoke the membership of a delegate member except for the user that originally delegated the role. This extension will add a great deal of complexity.
Adding grant-dependent revocation to our model also has its own advantages and disadvantages. The advantages of adding this feature will add the following:

- It makes the process of revocation more controllable

- It eliminates conflict between the original members

The following is a list of added complications as a result of adding this extension:

1. From the sponsor side

- In the case of revocation we have to keep track of who the sponsoring user is in order to do revocation. This is especially cumbersome when dealing with a large number of users.

- If the sponsoring role gets revoked from the sponsoring user, then we have to deal with issue of what to do with its delegated roles and how.

- The misconduct of the delegate member can go a long way without being revoked.

- Multiple sponsorship will be an issue that we have to deal with if we allow a member to be a delegate for more than one sponsor

2. From the supporting role's side

- We have to worry about the supporting role's prerequisite condition

- We have to deal with the question of what happens if the delegate member in the sponsoring role loses his original membership in his supporting role

- We have to deal with cascading revoke, which is an awkward thing to deal with

- The number of sponsoring roles also become a factor when dealing with revocation, because in this case all the problems of cascading revokes and prerequisite conditions will increase depending on the numbers of the supporting roles

4.2 There are two types of permissions (Delegable and Non-delegable permissions)

Adding permissions to our existing model will not have any impact on the delegation or revocation, because the only relevant element to delegation and revocation is the human. What it adds, however, is an extra control on what can and can not be delegated.

By defining permission as delegable or a non-delegable, we put the control in the hands of the administration. This will require an additional process that can predefine the set of permissions as one package.

The formal definition of the modified RBDM0 will have the components from the original model plus the following components:

- Each role has two types of permissions:

- Delegable permissions (PD) are the permissions allowed to be delegated. These types of permissions are available to both the original members as well as to the delegated members.

- Non-delegable permissions (PN) are the permissions that can not be delegated. These types of permissions are available only to the original members.

- P is a set of regular Permissions
- $PA \subseteq P \times R$ is many to many permission to role assignment relation
- $PDA \subseteq P \times R$ is many to many Delegable permission to role assignment
- $PNA \subseteq P \times R$ is many to many Non-delegable permission to role assignment

- PA = PDA $\cup$ PNA
- PDA $\cap$ PNA = $\varnothing$
- Permissions: $R \rightarrow 2^P$ is a function mapping each role to a set of permissions
  Permission (r) = {P | (P, r) $\in$ PA}
  Permission PD(r) = {P | (P, r) $\in$ PDA}
  Permission PN(r) = {P | (P, r) $\in$ PNA}
- Original members O(r) in a role get all the permissions assigned to that role
- Delegated members D(r) in a role get only the delegated permissions

4.3 Two step delegation

This type of delegation allows the delegated role memberships to be further delegated to other roles. We show how two-step delegation can be modeled. Multi-step delegation can be similarly developed.

Definition 4 The RBDM0 with two-step delegation has the following components

- U, R, P are sets of users, roles , and permissions
- UA $\subseteq$ U $\times$ R is many to many user to role assignment relation
- UAO $\subseteq$ U $\times$ R
- UAD $\subseteq$ U $\times$ R
- UADD $\subseteq$ U $\times$ R
- UA = UAO $\cup$ UAD $\cup$ UADD
- UAO $\cap$ (UAD $\cup$ UADD) = $\varnothing$
- Users: $R \rightarrow 2^U$ is a function mapping each role r to a set of users
- Users(r) = {U | (U, r)$\in$ UA}
- Users_O(r) = {U | (U, r)$\in$ UAO}
- Users_D(r) = {U | (U, r)$\in$ UAD}
- Users_DD(r) = {U | (U, r)$\in$ UADD}

Note that user_O(r) $\cap$user_D (r) $\cap$ DD_(r) = $\varnothing$ because UAO $\cap$ UAD $\cap$ UADD = $\varnothing$

4.4 Delegation in hierarchical roles

In role hierarchies, senior roles inherit the permissions of roles that are junior to them. When we extend our model to capture the user to user delegation using based on hierarchical roles, the model becomes more complicated. Here, we have to deal with different kinds of delegation. Some of these delegations are useless and some carry more risk than others do. In this section we will only give an overview of the different types of delegations using hierarchical roles and introduce some formal definitional in addition to those introduced in the original model. More detailed explanation of this requires further work. The following is a list of the different types of delegations.

4.4.1 Upward delegation

This type of delegation is useless because by the inheritance, the senior roles get all the permissions of their junior roles. Thus, there is no need for a user who is a member of a junior role to delegate he/her role membership to a user who is a member of a more senior role.

4.4.2 Downward delegation

This type of delegation works with the partial delegation only. By that we mean that we can not delegate the whole role because that will shrink the hierarchy.
This type of delegation is good for promoting a member who belongs to a junior role to be a member in a senior role.

4.4.3 Cross sectional delegation

This type of delegation is very useful. For example, a manager in a sales department can delegate his role membership to a member of the auditing department in order to conduct some auditing in the sales department.

In this type of delegation not only the original member of a role can delegate, but also, every

member in a role senior to the role of the original member can do the delegation.

Revocation issues become more complicated when we deal hierarchical roles. This is due to the involvement of many different roles.

Partial delegation can be accomplished by delegating only the relevant junior role or a combination of relevant junior roles.

5. Conclusion

In his paper we have described the motivation, intuition and outline of a new simple and a non-trivial model for user to user delegation using roles called RBDM (role-based delegation model) that is based on the Role-Based Access control (RBAC96) developed by [SCFY96]. RBDM has two main components: RBDM0 (role-based delegation model using flat role), and RBDM1 (role-based delegation model using hierarchical roles). Only the first component was described in full detail in this paper. The second component is still evolving and will be the subject of future work. Furthermore, in this paper we identified and discussed a list of some possible directions by which this model can be extended. This list includes revocation, partial delegation, multiple-step delegation, and delegation in hierarchical roles.

References

[ABLP96] Martin Abadi, Michael Burrows, Butler Lampson and Gordon Plotkin. A Calculus for Access Control in Distributed Systems. ACM Transactions on Programming Languages and Systems, Vol. 15, No 4, September 1993, pages 706-734.

[FCK95] David Ferriaolo, Janet Cugini, and Richard Kuhn. Role-based access control (RBAC): Features and motivations. In Proceedings of 11th Annual Computer Security Application Conference, pages 241-48, New Orleans, LA, December 11-15 1995.

[FK92] David Ferriaolo and Richard Kuhn. Role-based access controls. In Proceedings of 15th NIST-NCSC National Computer Security Conference, pages 554-563,Baltimor, MD, October 13-16 1992.

[Glad197] Henry M. Gladny, Access Control for Large Collections. ACM Transactions on Information Systems, Vol.15, No.2, April 1997, Pages 154-194.

[GM90] Morrie Gasser, Ellen McDermott. An Architecture for practical Delegation in a Distributed System. 1990 IEEE Computer Society Symposium on Research in Security and Privacy. Oakland, CA. May 7-9, 1990.

[HRU76] M.H. Harrison, W.L. Ruzzo, and J.D. Ullman, Protection in Operating Systems. Communications of ACM. 1976. Pages 461-471.

[Lamp71] B.W. Lampson, Protection. 5th Princeton Symposium on information science and system. Pages 437-443.

[San92] Ravi Sandhu, The Typed Access Matrix Model. Proceeding Symposium on Security and Privacy,

Oakland, CA, May 4-6, 1992, pages 122-136.

[San97] Ravi Sandhu. Rationale for the RBAC96 family of access control models. In Proceedings of the 1st ACM Workshop on Role-Based Access Control. ACM, 1997.

[SB97] Ravi Sandhu and Venkata Bhamidipati. Role-based administration of user-role assignment: The UR97 model and its Oracle implementation. In Proceedings of IFIP WG11.3 Workshop on Data Security. August, 1997.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. IEEE Computer, 29(2):38-47, February 1996.