PBDM: A Flexible Delegation Model in RBAC

Xinwen Zhang
George Mason University
xzhang6@gmu.edu

Sejong Oh
George Mason University
soh2@gmu.edu

Ravi Sandhu
George Mason University
NSD Security
sandhu@gmu.edu

ABSTRACT

Role-based access control (RBAC) is recognized as an efficient access control model for large organizations. Most organizations have some business rules related to access control policy. Delegation of authority is among these rules. RBDM0 and RDM2000 models are recently published models for role-based delegation. They deal with user-to-user delegation. The unit of delegation in them is a role. But in many cases users may want to delegate a piece of permission from a role. This paper proposes a flexible delegation model named Permission-based Delegation Model (PBDM), which is built on the well-known RBAC96 model. PBDM supports user-to-user and role-to-role delegations with features of multi-step delegation and multi-option revocation. It also supports both role and permission level delegation, which provides great flexibility in authority management. In PBDM, a security administrator specify the permissions that a user (delegator) has authority to delegate to others (delegatee), then the delegator creates one or more temporary delegation roles and assigns delegatees to particular roles. This gives us clear separation of security administration and delegation.

Categories and Subject Descriptors

D.4.6 [**Operating System**]: Security and Protection—*Access Control*

General Terms

Security

Keywords

Access control, RBAC, Delegation

1. INTRODUCTION

Access control is an important security issue in large organizations such as commercial companies, hospitals, government organizations, and colleges. Role-based access control (RBAC) is a proven and increasingly commonplace technology for these organizations. In RBAC, access rights are associated with roles, and users are assigned appropriate roles thereby acquiring the corresponding permissions. The notion of role is an enterprise or organizational concept. RBAC allows us to model security from the perspective of the organization, because we can align security modelling to the roles and responsibilities in the organization. Most large organizations have some business rules related to access control policy such as need-to-know, separation of duty, rotation of sensitive job position, and so on. Delegation of authority is an important one of these. Delegation means that a person gives all or part of his authority to somebody. There are three types of situations in which delegation takes place.

1. ***Backup of role***. When an individual is on a business trip or long-term absence, the job functions need to be maintained by others. This requires that somebody be delegated the authority to do the absent individual's job.

2. ***Decentralization of authority***. When an organization needs to setup initially or reorganize subsequently, job functions are distributed from higher job positions to lower job positions in the organization structure.

3. ***Collaboration of work***. Oftentimes people need to collaborate with others in the same organization or other organizations. In this case, we need to grant some access authority to share information.

The first and third cases need temporary delegation of authority rather than the second case which needs durable delegation. Temporary delegation means that the term of delegation is short; after the term ends, delegated authority is revoked or expired.

A number of models dealing with various aspects of delegation have been published, including [5-10]. The RBDM0 [2,3] and RDM2000 [1] models, in particular, are primarily based on roles. In this paper we propose a flexible delegation model based on RBAC, named PBDM (Permission-based Delegation Model). The PBDM model covers all three cases of "backup of role", "collaboration of work", and "decentralization or distribution of authority".

The rest of this paper is organized as follows. Section 2 presents related work and our motivation. We briefly review RBDM0 and RDM2000 model. Section 3 then presents PBDM0 model. Section 4 presents PBDM1 model, which is extended from PBDM0. Section 5 presents PBDM2, which is a role-to-role delegation model extended from PBDM1. Section 6 compares our model with RBDM0 and RBDM2000. Conclusions and future work are presented in Section 7.

2. RELATED WORKS AND MOTIVATION

There has been considerable work on different aspects of delegation, including the following. Gasser and McDermott addressed

SACMAT'03, June 2–3, 2003, Como, Italy.

user-to-machine delegation [7]. Stein explored delegation and inheritance on the object-oriented environment [8]. Nagaratnam and Lea introduce process-to-process delegation in the distributed object environment [9]. Sandhu et al addressed delegation among the role administrators in the ARBAC97 model [5]. Goh and Baldwin dealt with delegation as an attribute of role [10].

RBDM0 and RDM2000 are closely related to our model. RBDM0 [2,3] addresses human-to-human delegation, whereby a user in a role (delegator role) delegates his role membership to another user in another role (delegatee role). RBDM0 is the first attempt to model delegation involving user-to-user based on roles. It formalized the delegation model with total delegation and flat roles. It also deals with revocation, delegation with hierarchical roles, and multi-step delegation. The last two features are discussed very shortly.

RDM2000 [1] is an extension of RBDM0. It supports regular role delegation in role hierarchy and multi-step delegation. A rule-based declarative language has been proposed to specify and enforce policies. It uses $can\_delegate$ condition with prerequisite roles to restrict the scope of delegation. The unit of delegation in RBDM0 or RDM2000 is "role". Delegation means that "delegator" assigns "delegatee" to some role. No delegator can delegate a piece of role; they cannot break a role. But in many situations, such as "backup of role" or "collaboration of work", a delegator wants to delegate a piece of a role. Figure 1 shows the cases of a partial delegation. Case 1 and case 2 require that unit of delegation should be permission rather than role. Case 3 requires that unit of delegation should be both permission and role. The RBDM0 and RDM2000 models cannot cover these cases. Therefore a new delegation model is needed.

3. PBDM0

As discussed in section 2, permission level delegation is required in the real world. In this section we develop PBDM0 to address this requirement. An intuitive overview of this model is described first then a formal definition will be presented.

3.1 Overview of PBDM0

PBDM0 is designed with the following considerations in mind. It is based on the RBAC96 model. As such it extends RBAC96 to include user-to-user delegation. It provides for single-step as well as multi-step delegation. The focus is on temporary delegation rather than durable delegation. A delegator can delegate both permissions and roles to delegatees. To facilitate control of delegation by security administrators, we adapt control concepts from ARBAC97.

Before addressing formal PBDM0, we will show an example for intuitive understanding. Let us consider Case 3 in Figure 1. John wants to delegate a permission "$change\_schedule$" and a role "PE" to Jenny. In PBDM0 model, John can delegate according to following 3 phases.

P1 John creates a temporary delegation role "$D1$".

P2 John assigns the permission "$change\_schedule$" to $D1$ with permission-role assignment and role "PE" to $D1$ with role-role assignment.

P3 John assigns Jenny to $D1$ with user-role assignment.

As a result of delegation, the access control information in Figure 1 is changed to Figure 2. Jenny acquires new permissions: "$change\_schedule$" and "$req\_program$". The central idea of PBDM0 is to create one or more temporary delegation roles (DTR), and assign permissions or roles to them. In PBDM0, roles in DTR are different and distinct from regular roles (RR). DTR cannot be assigned to any other roles because PBDM0 does not allow role-to-role delegation, because role-to-role delegation with PBDM0 will generate invalid permission inheritance in role hierarchy. Suppose $D1$ is assigned to QE in Figure 2, then all users of QE and senior of QE will get the permissions of $D1$, which is not permitted by security administration.

Revocation is the reverse process of delegation. A delegator can remove his/her own delegation at any time. Revocation includes three cases:

1. Remove user from delegatees, that is, revoke the user-delegation role assignment.
2. Remove one or more pieces of permissions from delegation role.
3. Revoke delegation role.

PBDM0 supports multi-step delegation and revocation. In Figure 2, Jenny has all permissions assigned to $D1$. Based on this delegation role, Jenny can create another temporary role $D2$ and assign any permissions of $D1$ to $D2$, then assign user to $D2$. This process is very similar to the delegation from John to Jenny, except the new delegation role is generated from an existing delegation role. This multi-step delegation process results in a multi-step revocation.

3.2 Formal Definition of PBDM0

Figure 3 shows basic components of PBDM0 based on RBAC96 model. PBDM0 differs from RBAC96 with respect to the components of roles, user-role assignment (UA), permission-role assignment (PA), and role-role assignment (RRA). In PBDM0, roles are partitioned into regular roles (RR) and delegation roles (DTR). This partition induces a parallel partition of UA and PA. UA is separated into user-regular role assignment (UAR) and user-delegation role assignment (UAD). PA is similarly separated into permission-regular role assignment (PAR) and permission-delegation role assignment (PAD). Delegation role can be placed in the regular role hierarchy when a delegator delegates a regular role or roles to a delegatee, otherwise it is isolated from the hierarchy. A delegation role cannot have any senior regular role if it is placed in the role hierarchy, since delegated permissions cannot be inherited through role-role assignment. In general, definition of RR, UAR, PAR, and the regular role hierarchy (RRH) is responsibility of security administrators. Definition of DTR, UAD, PAD and delegation role hierarchy ($DTRH$) is responsibility of general users which essentially adds a form of discretionary access control (DAC) to the RBAC96 model.

PBDM0 Model:

Sets: U, S, P, RR, DTR, PA, PAR, PAD, UA, UAR, UAD

$RRH \subseteq RR \times RR$: regular role hierarchy

$DTRH_u \subseteq DTR \times DTR$: delegation role hierarchy owned by a user u

$R = RR \cup DTR$

$RR \cap DTR = \emptyset$

$UAR \subseteq U \times RR$

$UAD \subseteq U \times DTR$

$UA = UAR \cup UAD$

$PAR \subseteq P \times RR$

$PAD \subseteq P \times DTR$

$PA = PAR \cup PAD$

$senior(r) : R \rightarrow 2^R$: a function mapping a role to all its senior roles in role hierarchy.

$\forall dtr \in DTR \cdot senior(dtr) \cap RR = \emptyset$: for each delegation role

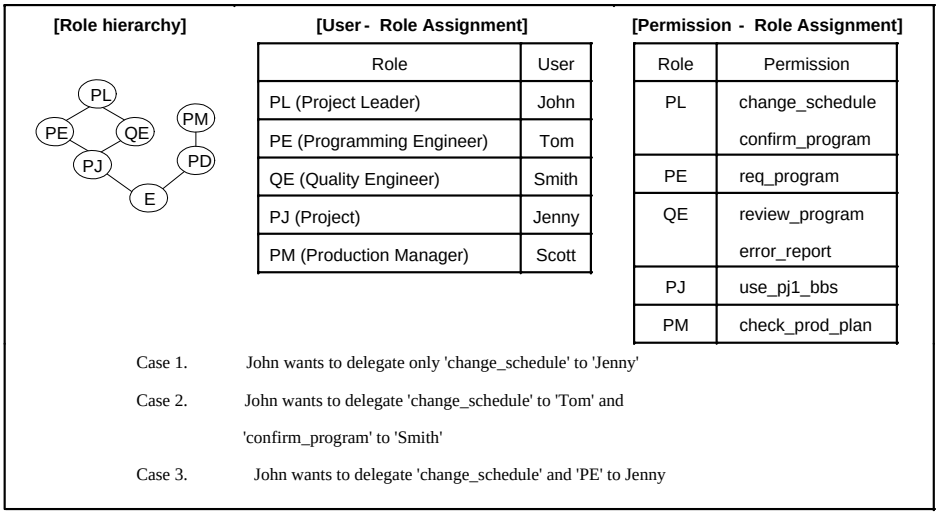

Figure 1: Cases of Permission Level Delegation

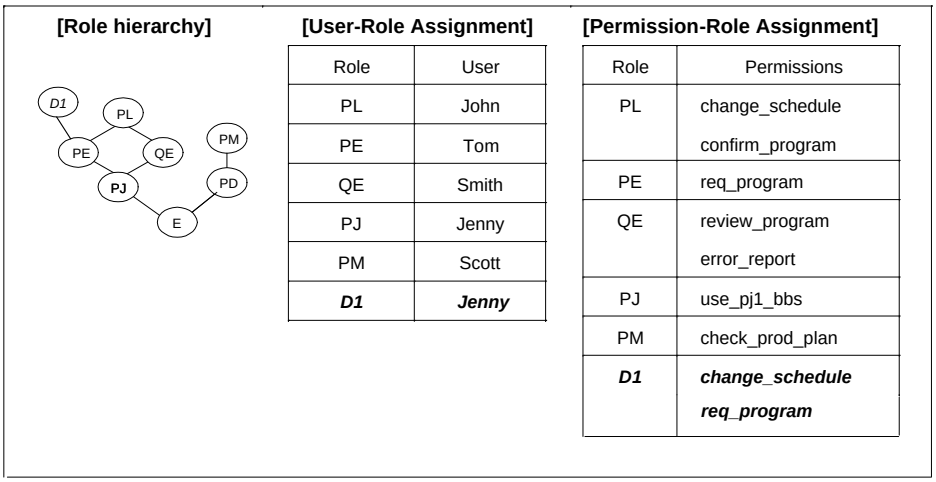

Figure 2: Example of PBDM0

there is no senior regular role.
$own(u) : U \rightarrow 2^{DTR}$ and $\nexists(u_1, u_2 \in U, dtr \in DTR) \cdot (u_1 \neq u_2) \wedge \big(dtr \in own(u_1) \wedge dtr \in own(u_2)\big)$: a function mapping a user to a set of delegation roles which he/she created.
$permissions\_r(r) : RR \rightarrow 2^P$, a function mapping a regular role to a set of permissions.
$permissions\_d(r) : DTR \rightarrow 2^P$, a function mapping a delegation role to a set of permissions.
$permission^*(u)$: a function mapping a user to a set of permission with UAR and UAD (when multi-step delegation is allowed).
$permissions\_r(r) = \{p : P \mid \exists r' \leq r \cdot (r', p) \in PAR\}$
$permissions\_d(r) = \{p : P \mid \exists r' \leq r \cdot (r', p) \in PAD\}$
$permission^*(u) = \{p : P \mid \exists r \in RR \cdot (u, r) \in UAR \wedge (r, p) \in PAR\} \cup \{p : P \mid \exists r \in DTR \cdot (u, r) \in UAD \wedge (r, p) \in PAD\}$
$\forall dtr \in DTR, (\exists u \in U \cdot \big(dtr \in own(u)\big) \wedge \big(permissions\_d(dtr) \subseteq permission^*(u)\big)$: the permissions pool for a delegation role owned by a user is the permission set that assigned to this user by UAR and UAD (when multi-step delegation is allowed).
$can\_delegate \subseteq RR \times Pre\_con \times P\_range \times M$ where Pre_con: prerequisite condition, P_range: delegation range, M: maximum delegation depth: a relation to to mapping a user to his/her delegation range with prerequisite conditions.

$can\_delegate$ is a constraint on UAD and PAD. For example, $can\_delegate(PL, PJ, \{change\_schedule, PE\}, 1)$ means that a user who has PL can delegate $\{change\_schedule, PE\}$ to others who have PJ role. In addition, the delegated role or permission cannot be re-delegated to other users with the maximum depth of delegation is 1. Table 1 shows an example of $can\_delegate$.

In PBDM0, the UAR and PAR are managed by organization security administrator, while UAD and PAD is managed by individual user with the ownership relation between U and DTR. Only owner can assign permissions or users to his/her delegation roles. A delegation role can be deleted only by owner.

4. PBDM1

As we defined in above section, PBDM0 solves the problem of

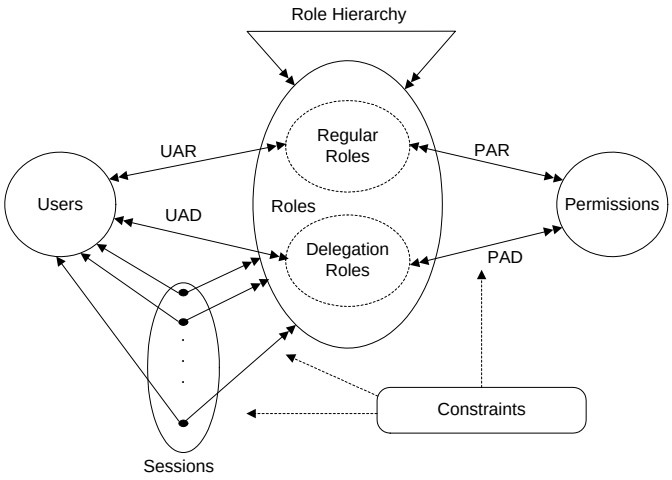

Figure 3: PBDM0 Model

Rule No	users assigned regular role	Pre_con	P_range	M
$R1$	PL	PE	$\{confirm\_program\}$	1
$R2$	PL	$PJ \vee PM$	$\{change\_schedule, PE\}$	3
$R3$	QE	PJ	$\{error\_report\}$	2
$R4$	PM	PD	$\{check\_prod\_plan\}$	3

Table 1: Example of $can\_delegate$

permission level delegation and revocation. But there are still two main shortcomings with PBDM0:

1. In PBDM0, a user can assign any permission to a delegation role, and assign any user to this delegation role. This will result in a risk that a malicious user can delegate high level permissions to a low level user without notice of the security administrator. The reason for this is that a security administrator has no permission to manage or monitor the UAD and PAD owned by individual user.

2. PBDM0 only supports user-to-user delegation as we showed in Section 3. In the real world, there are many cases that role-to-role delegation is expected. For example, in a department, if the project leader (PL) is out of work, part of the PL's permissions can be delegated to QE, and other permissions can be delegated to the project leader in another department. This delegation is determined based on role, not user.

In this section we extend PBDM0 to address the first problem. The new model is named PBDM1 which supports security administrator involved delegation and revocation.

4.1 Overview of PBDM1

In this model, there are three different layers of roles: regular roles (RR), delegatable roles (DBR), and delegation roles (DTR). Permissions assigned to regular roles cannot be delegated to other roles or users. A delegatable role is the role whose permission can be delegated to other roles or users by creating delegation role. For each delegatable role there is a regular role on which it is based. There is a one-to-one mapping between regular roles and delegatable roles. The users that assigned to a delegatable role are exactly the same as that assigned to a regular role it based on. That is, a pair of (regular role, delegatable role) will be used as a single role in user-role assignment. It follows that a pair of regular role and delegatable role in PBDM1 is exactly the same as the regular role in PBDM0. The reason that we divide a role into two parts is that we want to facilitate the involvement of security administrator. In PBDM1, each delegatable role is created by a security administrator. The permission-delegatable role assignment (PAB) is also managed by security administrator. Therefore the security administrator can control the permission flow by assigning different permission to delegatable role. From this the first problem in PBDM0 is solved.

There is ownership between U and DTR which is similar to the relationship between U and DTR in PBDM0. Each user owns a set of delegation roles and has no common ones with others. Same as PBDM0, the delegation roles owned by a user can form a role hierarchy determined by this user.

In PBDM1, RR and DBR are durable roles while DTR are temporary at owner's discretion. The permission-regular role assignment (PAR), user-regular role assignment (UAR), permission-delegatable role assignment (PAB), and user-delegatable role assignment (UAB) are managed by security administrators, while permission-delegation role assignment (PAD) and user-delegation role assignment (UAD) are managed by individual users.

Figure 4 and 5 show the roles, permissions and user assignments. When John wants to delegate some of his delegatable permissions to others, it will follow the three phases similar to that in Section 3, except that the permissions only can be taken from PL':

P1 John creates a temporary delegation role "$D2$".

P2 John assigns the permission "$change\_schedule$" to $D2$ from PL' with permission-role assignment and role "PE'" to $D2$ with role-role assignment.

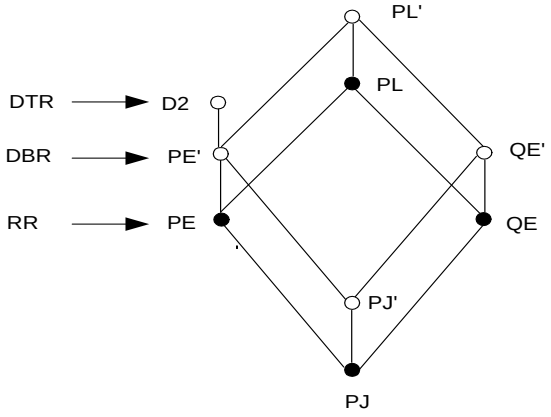

Figure 4: Role and Role Hierarchy in PBDM1

P3 John assigns Jenny to $D2$ with user-role assignment.

4.2 Formal Definition of PBDM1

Figure 6 shows basic components of PBDM1 extended from PBDM0. Similar to PBDM0, a delegation role can be placed in the delegatable role hierarchy when a user delegates a whole delegatable role or roles to a delegatee, otherwise it is isolated from the hierarchy. Same as PBDM0, a delegation role cannot have any senior delegatable role or regular role since delegated permissions cannot be inherited.

PBDM1 Model:
Sets: U, S, P, R, RR, DBR, DTR, PAR, PAB, PAD, UAR, UAB, UAD
$RRH \subseteq RR \times RR$: regular role hierarchy
$DBRH \subseteq DBR \times DBR$: delegatable role hierarchy
$DTRH_u \subseteq DTR \times DTR$: delegation role hierarchy owned by a user u
$R = RR \cup DBR \cup DTR$
$RR \cap DBR = \emptyset$
$RR \cap DTR = \emptyset$
$DBR \cap DTR = \emptyset$
$UAR \subseteq U \times RR$
$UAB \subseteq U \times DBR$
$UAD \subseteq U \times DTR$
$UA = UAR \cup UAB \cup UAD$
$PAR \subseteq P \times RR$
$PAB \subseteq P \times DBR$
$PAD \subseteq P \times DTR$
$PA = PAR \cup PAB \cup PAD$
$user\_r(r) : RR \rightarrow 2^U$: a function mapping a regular role to a set of users that assigned to this role.
$user\_b(r) : DBR \rightarrow 2^U$: a function mapping a delegatable role to a set of users that assigned to this role.
$own\_b(r) : DBR \rightarrow RR$: a function mapping each delegatable role to a single regular role on which it is based.
$\forall rr \in RR, \exists u : U, dbr : DBR \cdot (u, rr) \in URA \wedge rr = own\_b(dbr) \Rightarrow user\_r(rr) = user\_b(dbr)$: all users that assigned to a regular role must be the assigned to the corresponding delegatable role, and there is no other users assigned to this role.
$own\_d(r) : DBR \rightarrow 2^{DTR}$ and $\nexists(dbr_1, dbr_2 \in DBR, dtr \in DTR) \cdot (dbr_1 \neq dbr_2) \wedge \big(dtr \in own\_d(dbr_1) \wedge dtr \in own\_d(dbr_2)\big)$: a function mapping a delegatable role to a set of delegation roles.
$permissions\_r(r) : RR \rightarrow 2^P$, a function mapping a regular role to a set of permissions.
$permission\_b(r) : DBR \rightarrow 2^P$, a function mapping a delegatable role to a set of permissions.
$permissions\_d(r) : DTR \rightarrow 2^P$, a function mapping a delegation role to a set of permissions.
$permission^*(u)$: a function mapping a user to a set of delegatable permissions with UAB and UAD (when multi-step delegation is allowed).
$permissions\_r(r) = \{p : P \mid \exists r' \leq r \cdot (r', p) \in PAR\}$
$permissions\_b(r) = \{p : P \mid \exists r' \leq r \cdot (r', p) \in PAB\}$
$permissions\_d(r) = \{p : P \mid \exists r' \leq r \cdot (r', p) \in PAD\}$
$permission^*(u) = \{p : P \mid \exists r \in DBR \cdot (u, r) \in UAB \wedge (r, p) \in PAB\} \cup \{p : P \mid \exists r \in DTR \cdot (u, r) \in UAD \wedge (r, p) \in PAD\}$
$\forall dtr \in DTR, \exists u \in U \cdot \big(dtr \in own\_d(u)\big) \wedge \big(permissions\_d(dtr) \subseteq permission^*(u)\big)$: the permissions pool to create a delegation role owned by a user is the delegatable permissions that assigned to this user by UAB and UAD (when multi-step delegation is allowed).
$can\_delegate \subseteq DBR \times Pre\_con \times P\_range \times M$ where Pre_con: prerequisite condition, P_range: delegation range, M: maximum delegation depth: a relation to to mapping a delegatable role to its delegation range.
$can\_delegate$ is a constraint on UAD and PAD. For example, $can_delegate(PL', PJ', \{change\_schedule, PE\}, 1)$ means that a user having PL' (also has PL) can delegate $\{change\_schedule, PE\}$ to others who have PJ' (also has PJ) role. In addition, the delegated role or permission cannot be re-delegated to other users with the maximum depth of delegation is 1.

In PBDM1, the UAR, UAB, PAR, and PAB are managed by organization security administrator, while UAD and PAD is managed by individual user with the $owner\_d$ relation between U and DTR. Therefore a form of discretionary access control (DAC) is added to RBAC.

4.3 Delegation Revocation in PBDM1

PBDM1 provides more options than PBDM0 for the delegation revocation. With the separation of delegatable permissions and non-delegatable permissions by a security administrator, the delegation revocation can be security administrator involved. Possible revocation mechanisms by individual user are:

1. Remove a user from delegatees, that is, revoke the user-delegation role assignment.
2. Remove one or more pieces of permissions from delegation role.
3. Revoke delegation role.

Above options are exactly the same as that in PBDM0. With PBDM1, possible revocation mechanisms by a security administrator can be:

1. Remove one or more pieces of permission from a delegatable role to its regular role.
2. Revoke a user from regular role and delegatable role.

5. PBDM2

Both PBDM0 and PBDM1 cannot support role-to-role delegation. A role-to-role delegation is expected in many cases as we

PAR

RR	Permissions
PL	confirm_program
PE	
QE	error_report
PJ	use_pj1_bbs

PAB

DBR	Permissions
PL'	change_schedule
PE'	req_program
QE'	review_program
PJ'	

PAD

DTR	Permissions
D2	***change_schedule***
	req_program

UA

U	Roles
John	PL, PL'
Tom	PE, PE'
Smith	QE, QE'
Jenny	***PJ, PJ', D2***

Figure 5: PA and UA in PBDM1

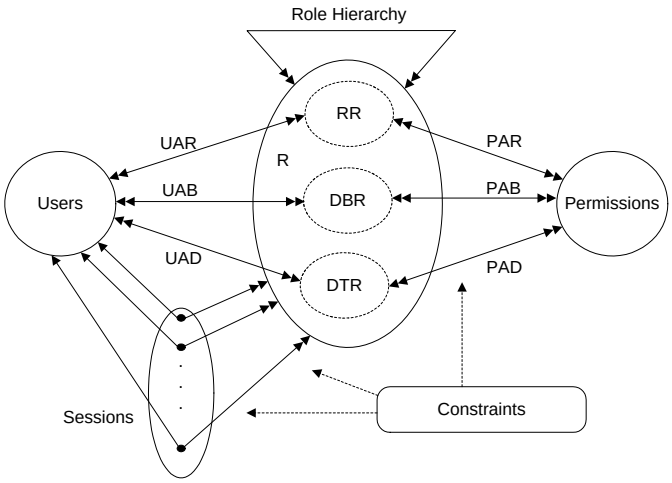

Figure 6: PBDM1 Model

mentioned in Section 4. In PBDM0 and PBDM1, the delegation role cannot be assigned to another regular role or delegatable role, since the permission inheritance along the role hierarchy will result in invalid permission flow. In this section, we will present a new model extended from PBDM1, named PBDM2, which supports role-to-role delegation with multi-step delegation and multi-option revocation features.

5.1 Overview of PBDM2

PBDM2 is a role-to-role delegation model. Figure 7 shows the roles and hierarchies in this model. Similar to PBDM1, the roles are divided into different layers. In PBDM2, there are four different layers: regular roles (RR), fixed delegatable roles ($FDBR$), temporal delegatable roles ($TDBR$), and delegation roles (DTR). RR and $FDBR$ are exactly the same as RR and DBR in PBDM1. The delegation roles are similar to that in PBDM0 and PBDM1, but the owner of a delegation role is a fixed delegatable role, not a user. In PBDM2, a delegator is a fixed delegatable role. A temporal delegatable role has the permissions that it receives from delegator with role-role assignment. There is a one-to-one mapping between RR, FDBR and TDBR. The users that assigned to a temporal delegatable role, a fixed delegatable role and a regular role are exactly the same if these three roles are in the one-to-one relation. That is, a pair of (regular role, fixed delegatable role, temporal delegatable role) will be used as a single role in user-role assignment. It follows that a pair of (regular role, fixed delegatable role) is exactly the same as a pair of (regular role, delegatable role) in PBDM1 and the same as the regular role in PBDM0. By dividing the delegatable role into fixed and temporal delegatable roles, a temporal role can receive permissions delegated by a fixed delegatable role. Since there is no role hierarchy for TDBR, invalid permission flow will not happen, then a role-to-role delegation can be achieved.

There are role hierarchies in RR and $FDBR$ layers. The three-layer roles are distinguished by the delegation and inheritance properties of their permissions assigned. On the user-role assignment side, there is only one layer, that is, set of (regular role, fixed delegatable role, temporal delegatable role) will be used as a single role in user-role assignment. Similar to PBDM1, a fixed delegatable role is created by a security administrator. There is a ownership between FDBR and DTR. Each fixed delegatable role owns a set of delegation roles and has no common ones with others. The delegation roles owned by a fixed delegatable role can form a role hierarchy.

The most important difference between PBDM2 and PBDM1 is that PBDM2 model is for role-to-role delegation. Therefore, all the delegation authority will be managed by a security administrator. There is no individual user can own any delegation roles and permissions. Suppose a delegation role $D3$ will be created based on PL' and delegated to QE'', the following three phases will be performed by a security administrator:

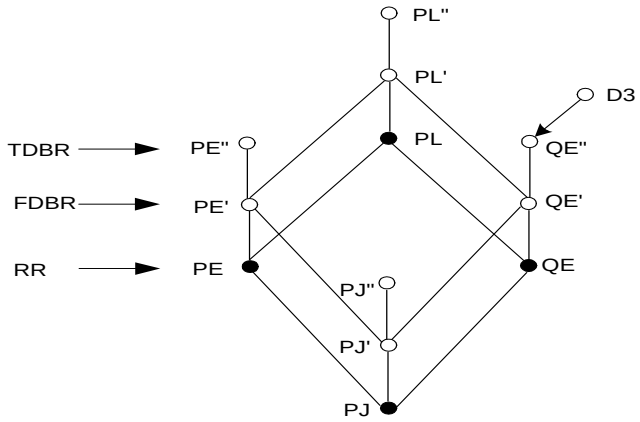

Figure 7: Roles and Role Hierarchy in PBDM2

P1 Create a delegation role $D3$.

P2 Assign permission $change\_schedule$ to $D3$, and assign PE' to $D3$.

P3 Assign $D3$ to QE''.

Since the delegation role $D3$ is assigned to a temporal delegatable roles QE'', the relation between $D3$ and PE' are not shown in the role hierarchy. The arrow line shows the temporal delegatable role-delegation role assignment (RAD). Figure 8 shows the permission-role assignments in this example. PL'' receives the permission use_pj2_bbs delegated by some role in $project2$ department. The permission directly assigned to QE'' is empty, but it inherits all permissions from $D3$ by RAD.

5.2 Formal Definition of PBDM2

Figure 9 shows basic components of PBDM2.

PBDM2 Model:

Sets: U, S, P, R, RR, $FDBR$, $TDBR$, DTR, PAR, $PAFB$, $PATB$, PAD, UAR, $UAFB$, $UATB$, UAD, RAD

$RRH \subseteq RR \times RR$: regular role hierarchy

$FDBRH \subseteq FDBR \times FDDBR$: fixed delegatable role hierarchy

$DTRH_r \subseteq DTR \times DTR$: delegation role hierarchy owned by a role r

$DBR = FDBR \cup TDBR$: delegatable roles

$R = RR \cup DBR \cup DTR$

$RR \cap DBR = \emptyset$

$RR \cap DTR = \emptyset$

$DBR \cap DTR = \emptyset$

$FDBR \cap TDBR = \emptyset$

$UAR \subseteq U \times RR$

$UAFB \subseteq U \times FDBR$

$UATB \subseteq U \times TDBR$

$UA = UAR \cup UAFB \cup UATB$

$PAR \subseteq P \times RR$

$PAFB \subseteq P \times FDBR$

$PAD \subseteq P \times DTR$

$PA = PAR \cup PAFB \cup PAD$

$RAD = TDBR \times DTR$

$user\_r(r) : RR \rightarrow 2^U$: a function mapping a regular role to a set of users that assigned to this role.

$user\_fb(r) : FDBR \rightarrow 2^U$: a function mapping a fixed delegatable role to a set of users that assigned to this role.

$user\_tb(r) : TDBR \rightarrow 2^U$: a function mapping a temporal delegatable role to a set of users that assigned to this role.

$own\_fb(r) : FDBR \rightarrow RR$: a function mapping each fixed delegatable role to a single regular role on which it is based.

$own\_tb(r) : TDBR \rightarrow FDBR$: a function mapping each temporal delegatable role to a single fixed delegatable role on which it is based.

$\forall rr \in RR, \exists u : U, fdbr : FDBR, tdbr : TDBR \cdot (u, rr) \in URA \wedge rr = own\_fb(fdbr) \wedge fdbr = own\_tb(tdbr) \Rightarrow user\_r(rr) = user\_fb(fdbr) \wedge user\_fb(fdbr) = user\_tb(tdbr)$: all users that assigned to a regular role must be the assigned to the corresponding fixed and temporal delegatable role, and there is no other users assigned to these roles.

$own\_d(r) : FDBR \rightarrow 2^{DTR}$ and $\nexists(fdbr_1, fdbr_2 \in FDBR, dtr \in DTR) \cdot (fdbr_1 \neq fdbr_2) \wedge \big(dtr \in own\_d(fdbr_1) \wedge dtr \in own\_d(fdbr_2)\big)$: a function mapping a fixed delegatable role to a set of delegation roles.

$rad(r) : TDBR \rightarrow 2^{DTR}$: a function mapping a temporal delegatable role to a set of delegation roles.

$permissions\_r(r) : RR \rightarrow 2^P$, a function mapping a regular role to a set of permissions.

$permission\_fb(r) : FDBR \rightarrow 2^P$, a function mapping a fixed delegatable role to a set of permissions.

$permissions\_d(r) : DTR \rightarrow 2^P$, a function mapping a delegation role to a set of permissions.

$permissions\_t^*(r) : TDBR \rightarrow 2^P$: a function mapping a temporal delegatable role to a set of permissions inherited from RAD.

$permissions\_f^*(r) : FDBR \rightarrow 2^P$: a function mapping a fixed delegatable role to a set of delegatable permissions with $PAFB$ and RAD (when multi-step delegation is allowed)

$permissions\_r(r) = \{p : P \mid \exists r' \leq r \cdot (r', p) \in PAR\}$

$permissions\_fb(r) = \{p : P \mid \exists r' \leq r \cdot (r', p) \in PAFB\}$

$permissions\_d(r) = \{p : P \mid \exists r' \leq r \cdot (r', p) \in PAD\}$

$permission\_t^*(r) = \{p : P \mid \exists r' \in DTR \cdot (r', p) \in PAD \wedge r' \in rad(r')\}$

$permissions\_f^*(r) = \{p : P \mid (r, p) \in PAFB\} \cup \{p : P \mid \exists r' \in TDBR \cdot p \in permissions\_t^*(r') \wedge r = own\_tb(r')\}$

$\forall dtr \in DTR, \exists fdbr \in FDBR \cdot \big(dtr \in own\_d(fdbr)\big) \wedge$

PAR

RR	Permissions
PL	confirm_program
PE	
QE	error_report
PJ	use_pj1_bbs

PAFB

FDBR	Permissions
PL'	change_schedule
PE'	req_program
QE'	review_program
PJ'	

PATB

TDBR	Permissions
PL"	use_pj2_bbs
PE"	
QE"	
PJ"	

PAD

DTR	Permissions
D3	***change_schedule***
	req_program

Figure 8: PA in PBDM2

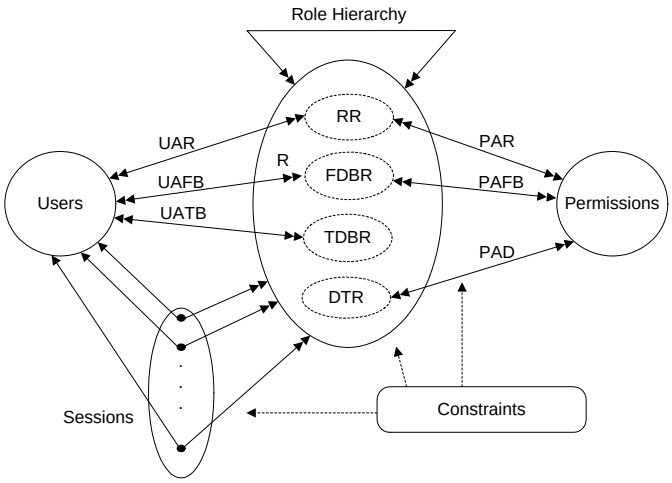

Figure 9: PBDM2 Model

$\big(permissions\_d(dtr) \subseteq permission\_f^*(fdbr)\big)$: the permissions pool to create a delegation role owned by a role is the delegatable permissions that assigned to this role by $PAFB$ and RAD (when multi-step delegation is allowed).
$can\_delegate \subseteq FDBR \times Pre\_con \times P\_range \times M$ where Pre_con: prerequisite condition, P_range: delegation range, M: maximum delegation depth: a relation to to mapping a fixed delegatable role to its delegation range.

5.3 Delegation Revocation in PBDM2

Similar to PBDM1, PBDM2 has the involvement of a security administrator in delegation and revocation. Furthermore, PBDM2 provides revocation options in role layer with the separation of original delegatable permissions and multi-step delegatable permissions. Possible revocation mechanisms are:

1. Remove one or more pieces of permissions from delegation role.
2. Revoke delegation role owned by a fixed delegatable role
3. Remove one or more pieces of permission from a fixed delegatable role to its regular role.

6. DISCUSSION

All models in PBDM support flexible role and permission level delegation. A delegator can delegate his/her entire or partial permissions to others. Partial revocation is also possible. The key idea for this flexibility is to separate delegation role (DTR) from regular role (RR) and delegatable role (DBR). In PBDM2, temporal permissions delegated from other roles are separated from its original delegatable permissions. In RBDM0 and RDM2000 model, there is no difference between them. Figure 10 illustrates this point. In RBDM0 and RDM2000 model, PA is unique and a delegator cannot touch PA. As a result permission level delegation is not available. In the PBDM models a delegator can touch PA through PAD function, and permission level delegation is possible. Moreover the two previous delegation models have ambiguous user-to-role assignment. A role has both originally assigned users through UAO and delegated users through UAD. This induces ambiguity between security administration scope and delegation scope. In PBDM1 model UA is clearly separated into UAR, UAB and UAD. Delegator cannot touch security administration scope of UAR and UAB. Therefore our model matches more closely with administrative RBAC model such as ARBAC97 [5]. Clearly PBDM models have more powerful modelling features. RBDM0 and RDM2000 model can be interpreted as special cases of PBDM. The advantages of these two models are thereby also available in PBDM.

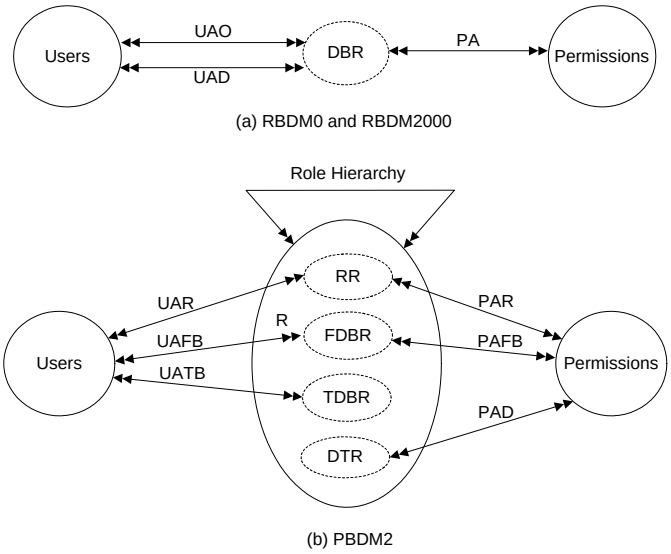

Figure 10: RBDM0, RBDM2000 and PBDM1

7. CONCLUSION AND FUTURE WORK

We propose a flexible delegation model named PBDM, which includes three sub-models: PBDM0, PBDM1, and PBDM2. PBDM0 and PBDM1 models support user-to-user while PBDM2 supports role-to-role delegation. All models support multi-step delegation and revocation in role and permission level. In PBDM0, a user can delegate all of permission from his/her roles to delegation roles. In PBDM1, only delegatable permissions in a role are available for a user to delegate. In PBDM2, delegatable permissions received from other roles are separated from original delegatable permission, from which role-to-role delegation is support without illegal permission flow. In PBDM1 and PBDM2 a security administrator can control the permission flow by defining delegatable roles. Both delegation and revocation are controlled by a security administrator. PBDM is flexible and useful for management of delegation authority in role-based access control environment. It is motivated by temporary delegation situations.

The future work includes the study of constraints in RBAC delegation models, such as separation of duty in user-to-user and role-to-role delegation. Also, the delegation problems in distributed environment will be studied in the future.

8. REFERENCES

[1] Longhua Zhang, Gail-Joon Ahn, and Bei-Tseng Chu, A rule-based Framework for Role-Based Delegation, Proc. 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001), May, 2002.

[2] Ezedin Barka and Ravi Sandhu, Framework for Role-Based Delegation Models, Proc of 16th Annual Computer Security Application Conference (ACSAC 2000). December, 2000.

[3] Ezedin Barka and Ravi Sandhu, A Role-Based Delegation Model and Some Extensions, Proc. of 23rd National Information Systems Security Conference (NISSC 2000). December, 2000.

[4] Ravi Sandhu, Edward Coyne, Hal Feinstein and Charles Youman, Role-Based Access Control Models, IEEE Computer, Volume 29, Number 2, February, 1996.

[5] Ravi Sandhu, Venkata Bhamidipati and Qamar Munawer, The ARBAC97 Model for Role-Based Administration of Roles, ACM Transactions on Information and System Security, Volume 2, Number 1, February, 1999.

[6] Moffett, J.D., Delegation of Authority Using Domain Based Access Rules, PhD Thesis. Dept of Computing, Imperial College, University of London. 1990.

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

[8] Lynn Andrea Stein, Delegation Is Inheritance, Proc. of Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '87). October, 1987.

[9] Nataraj Nagaratnam, Doug Lea, Secure Delegation for Distributed Object Environments, USENIX Conference on Object Oriented Technologies and Systems. April, 1998.

[10] Cheh Goh and Adrian Baldwin, Towards a more Complete Model of Role, Proc. of 3rd ACM Workshop on Role-Based Access Control. October, 1998.