2010 International Conference on Availability, Reliability and Security

Owner-Based Role-Based Access Control OB-RBAC

Mohsen Saffarian
Faculty of EEMCS University of Twente
Enschede, Netherlands
saffarianm@cs.utwente.nl

Babak Sadighi
Axiomatics AB Electrum 223
Kista, Sweden
babak@axiomatics.com

***Abstract*—Administration of an access control model deals with the question of who is authorized to update policies defined on the basis of that model. One of the models whose administration has absorbed relatively large research is the Role-Based Access Control (RBAC) model. All the existing role-based administrative models fall into the category of administrator-based decentralized approach. In such an approach, a group of administrators are given firstly, the authority of updating authorizations for operative roles and secondly, the authority of delegating the previous right to other lower-level administrators. However, in organizations with informal and flexible structure, like academic and research-oriented organizations such a sharp distinction between administrative roles and operative roles might not exist. Here, each role may take part in both operative and administrative decisions such that more mission-oriented decisions are made by senior roles and more specialized-level decisions are made by junior roles. In this paper, we study a new class of access control model called Owner-Based Role-Based Access Control (OB-RBAC) which is suitable for such environments. The OB-RBAC model utilizes the advantages of both Discretionary Access Control (DAC) and RBAC. In particular, the OB-RBAC model builds a policy model which not only fulfills the organizational restrictions but enjoys the flexible administration of the DAC model.**

I. Introduction

Access control policies can be divided into two types: access policies and administrative policies. While access policies answer the question of who is permitted to do what, administrative policies answer the question of who is authorized to update policies. There are three different approaches for the administration of policies: centralized, owner-based, and decentralized [1]. In the centralized approach, a group of administrators are given the full authority to update access policies. In the owner-based approach, it is the owner of an object who has the authority to update access policies for his object. In the decentralized approach, the owner or administrator is allowed to delegate the right of updating both access and administrative policies to other subjects. The decentralized approach requires complex administrative policies controlling who can delegate what to whom.

Administration of RBAC has absorbed large research resulting in several different models, such as the ARBAC97 [2], ARBAC02 [3] and the administrative scope model [4]. Existing role-based administrative models share two characteristics: firstly, they administer RBAC using RBAC itself and secondly, they use two different types of roles: regular (operative) roles and administrative roles. Using RBAC mechanisms to administer RBAC means that role-based administrative models use the same principles and assumptions as those of the basic and hierarchical RBAC. In particular, this means that both operative and administrative role hierarchies are created by ordering roles according to a generic-specialized relation, indication that specific roles inherit permissions from generic roles. Separation of operative and administrative roles is based on the assumption that both tasks and required permissions to satisfactorily execute them are separated into two different classes: access-level or operative and administrative-level or simply administrative. The two mentioned characteristics and their underlying assumptions lead to inconvenient consequences which are briefly described below.

Organizing a role hierarchy according to a generic-specialized relation leads to several problematic consequences [5], [6], [7], among which the most important one is that senior roles acquire all permissions assigned to junior roles. In many organizations, senior roles are not qualified enough to undertake the activities of junior roles [6].

Distinguishing between operative and administrative roles is not always realistic. Based on such an assumption, in a typical organization operative roles are associated with operative tasks and permissions to perform activities resulting in satisfying the goals of that organization. The purpose of the existence of administrative roles is to manage operative roles. To do so, administrative roles are given administrative permissions, namely role creation/deletion, edge insertion/deletion, permission-role assignment/revocation and user-role assignment/revocation. However, In many organizations an operative role can also be associated with administrative tasks. The existing role-based administrative models can deal with such a situation by assigning users of the operative role to an appropriate administrative role. Such an assignment should be performed with respect to the least privilege principle which requires that a member of the operative role should only be given access to administrative permissions that are necessary to complete his tasks. Therefore, when assigning users of the operative role to the existing administrative roles does not satisfy the least privilege principle, an appropriate administrative role should

978-0-7695-3965-2/10 $26.00 © 2010 IEEE
DOI 10.1109/ARES.2010.94

be created. Consequently, the number of administrative roles increases in the direct proportion to the number of operative roles associated with different set of administrative tasks.

To overcome the above-mentioned inconveniences, we present the Owner-Based Role-Based Access Control model (OB-RBAC). At the core of the OB-RBAC is the concept of timed-ownership delegation through which resources are assigned to roles for specific period of time. Similar to the traditional Discretionary Access Control (DAC) models, members of a role which is the owner of a resource have the authority to update both access and administrative policies regarding that resource. The OB-RBAC model comes with an intrinsic administrative model which falls into the owner-based decentralized approach. In particular, the OB-RBAC model can be well adapted to administrative procedures in organizations with flexible and informal structure.

The rest of the paper is organized as follows. In section 2 we briefly review the background research work which has influenced the OB-RBAC model. In section 3 we highlight the motivation behind the design of the OB-RBAC model. Semantics of the model are given in section 4. Section 5 discusses the related work and section 6 concludes the paper.

II. Motivation

In this section, in the context of the business-IT alignment principle [8], [9], [10], we highlight the motivation of proposing the OB-RBAC model. This principle is concerned with the importance of having IT systems correspond to the organizational structure and organization's goals. In this regard, we describe two unrealistic assumptions in the context of RBAC, namely permission and membership inheritance and separation of operative and administrative roles. These assumptions result in a policy model which does not necessarily capture the reality in organizations.

A. Permission and Membership Inheritance

The hierarchical RBAC assumes that the relation between roles in the hierarchy is of the *"isa"* type. In such a hierarchy, a member of a senior role is a member of its junior roles, thereby acquires permissions assigned to those junior roles. Furthermore, members of a more senior role can activate any subset of its junior roles. Moffet [11] argues that in addition to the generalization hierarchy, there are at least two other role hierarchies, namely activity hierarchy and supervision hierarchy, which are built based on the aggregation and supervision relations between roles, respectively. In a typical organization, most of the time, the reason for which a role is created is that it is impossible for the members of the already existing roles to satisfactorily perform all the activities of that organization. This is due to the fact that either among the existing roles some expertise is missing or the load of the activities assigned to each role is in such a way that some of them cannot be satisfactorily performed. In both cases, some activities are delegated from a delegator role to a newly created delegatee role. When such a delegation occurs, the delegator loses its authority to perform the delegated activities. This is due to the fact that if the delegator could perform the delegated activities satisfactorily, there would be no reason to delegate them. Moffet, in the same paper, also addresses the need for the supervision relation between the delegator and delegatee so that delegation works properly. As a conclusion, generalization hierarchy does not always reflect the relation between roles precisely.

B. Separation of Operative and Administrative Roles

Existing role-based administrative models assume that role-based policies can only be updated by the members of administrative roles. However, we argue that each role in an organization can be associated with both operative and administrative tasks. In particular, we argue that each role in the hierarchy can supervise and manage its immediate children through administrative permissions.

III. The OB-RBAC Model

In this section, we formally describe the OB-RBAC model. Particularly, we show how the role hierarchy is organized and administered. The role hierarchy in OB-RBAC is the activity hierarchy based on the aggregation relation between roles. Central to the activity role hierarchy is the delegation of subsets of activities down in the hierarchy to either newly created roles or the existing roles. When such a delegation occurs, the delegator no longer would be able to perform the delegated activities, but the delegatee becomes responsible to do so. However, this delegation works satisfactorily only if firstly, the delegatee is given the required permissions and resources to perform the delegated activities and secondly, the delegator is given the right of supervising the delegatee. We leave the investigation of models and mechanisms to support the supervision relation between the delegator and delegatee for the future work.

A. Delegation of Resources and Permissions in the OB-RBAC

The concept of delegation is at the core of the administration of authorizations. In the OB-RBAC model two different types of delegations, namely timed-ownership delegation and timed-permission delegation are used.

1) Timed-Ownership Delegation: Traditionally, the concept of role connects a group of subjects to a set of permissions. However, we believe a set of resources can also be associated to each role. In a typical organization, an individual subject is recognized as the owner of a resource, since he is a member of a role to which that resource has been assigned. In this way, subjects assigned to a role not only acquire the role's permissions but also become the owner of the role's resources. Here, the concept of ownership is different than that of used in the DAC model. Similar to the traditional RBAC models, in the OB-RBAC

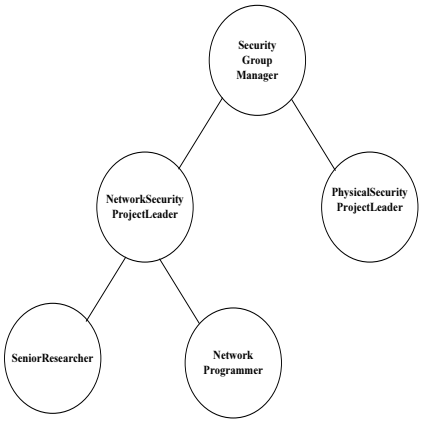

Figure 1. Role hierarchy in a security group.

model the system owns resources. However, by associating resource re to role r certain degree of control over re is given to the members of r. The system can restrict the degree of control by imposing set of constraints on each resource. When a specific type of resource is associated to a role, either each member of the role becomes the owner of his own specific instance of that resource, or all members of the role are collectively considered as the owner of instances of that resource. For instance, in Figure 1 assume that role *SeniorResearcher* is associated with 10GB disk space to be given to each member. Consequently, when subject s_1 is assigned to this role he becomes the owner of his own 10GB disk space and thus, can define policies regarding his disk space. Now, assume that printers p_1 and p_2 are also given to the same role, but to be shared between all members of the role. In this case, a general approach for approval of policies regarding p_1 and p_2 defined by members of role *SeniorResearcher* is to have a consensus protocol which approves a policy only if at least m out of n members accept that policy. For the sake of simplicity and also to be consistent with the previous case, we assume m is always equivalent to 1.

Definition 1: Let T denote a time interval of type $[t_1, t_2]$, where $t_1, t_2 \in \mathbb{R}$ and $t_1 \leq t_2$. We also define the two following relations:

- $t_i \in [t_1, t_2]$ if $t_1 \leq t_i$ and $t_i \leq t_2$
- $[t_1, t_2] \subseteq [t_3, t_4]$ if $t_3 \leq t_1$ and $t_2 \leq t_4$

Definition 2: We define the OWNS database as a set of expressions of the form $owns(r, u, re)[T]$, where $r \in R$ is a role, $u \in U$ is a user which can be null, $re \in RE$ is a resource and T is a time interval.

This means that for each resource within an organization, there is a corresponding record in the OWNS database specifying the role and maybe one of its members (in case that the resource is not shared between all of the members) who is the owner of that resource for a specific period of time. This will raise the issue that the system has to be initiated by a root role e.g. CEO which is the first owner in the system. Table 1 shows the OWNS database for the root role in our example depicted in Figure 1 at time point t_0.

Table I
OWNS DATABASE FOR THE ROOT ROLE IN FIGURE 1 AT TIME POINT t_0

Role	User	Resource	t_{start}	t_{end}
$Sec.Gro.Man.$	$null$	$Net.Sec.Pro.Lea.$	t_0	t_n
$Sec.Gro.Man.$	$null$	$Phy.Sec.Pro.Lea.$	t_0	t_n
$Sec.Gro.Man.$	$alice$	$laptop_1$	t_0	t_n
$Sec.Gro.Man.$	$alice$	$laptop_2$	t_0	t_n
$Sec.Gro.Man.$	$alice$	$disk_1 : 500GB$	t_0	t_n

In the OB-RBAC model, each role in the system is also considered as a specific type of resource whose ownership is shared between the members of its parent role. Over each role eight predetermined administrative actions can be performed: role creation/deletion, user-role assignment/revocation, permission-role assignment/revocation and resource-role assignment/revocation. Such administrative actions can only be performed by either the owner of a role or delegatees who receive such permissions from the owner. While the resource-role assignment permission is performed through timed-ownership delegation, role creation, user-role assignment and permission-role assignment actions are performed through timed-permission delegation. We leave the investigation of revocation mechanisms for the future work.

Definition 3: We define the set of timed-ownership delegation statements T-OD as follows:
own-delegate$(u_i, u_j, r, re, m, t)[T] \in$ T-OD if $u_i, u_j \in U$, $r \in R, re \in RE, m \in \{s, i\}$ which indicates the mode of the delegation such that if $m = s, u_j$ is null, t is a time point, and $[T]$ is a time interval.

own-delegate$(u_1, null, r_1, re_1, s, t_1)[T_1] \in$ T-OD means that user u_1 at time point t_1 delegates the ownership of resource re_1 to role r_1 such that the ownership is *shared* between all members of r_1 for the time interval T_1. The meaning of *own-delegate*$(u_1, u_2, r_1, re_1, i, t_1)[T_1] \in$ T-OD is that u_1 at time point t_1 delegates the ownership of re_1 to u_2, an *individual* member of role r_1, for the time interval T_1. *own-delegate*$(u_1, u_2, r_1, re_1, i, t_1)[T_1]$ operation succeeds if and only if the following constraints are satisfied (which are the same for mode s):

1) *own-delegate*$(u_1, u_2, r_1, re_1, i, t_1)[t_2, t_3]$, $t_1 \leq t_2$.
2) $((u_1, r_2) \in UR)^1 \wedge (r_2 \neq r_1) \wedge$

[1] UR is the user-role assignment relation.

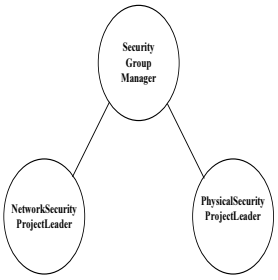

Figure 2. Role hierarchy in the security group at t_0

$(((r_2, u_1, re_1)[T_2] \in \text{OWNS}) \vee ((r_2, null, re_1)[T_2] \in \text{OWNS})$ such that $[T_1] \subseteq [T_2]$).

3) $\neg\exists(((r_i, u_j, re_1)[T_i] \in \text{OWNS}) \vee ((r_i, null, re_1)[T_i] \in \text{OWNS})$ such that $\exists\, t_j \in [T_1] \wedge t_j \in [T_i]$).
4) $(r_2, null, r_1)[T_3] \in \text{OWNS} \wedge [T_1] \subseteq [T_3]$.

The first constraint indicates that the owner of a resource can only delegate the ownership of his resource for a time interval in the future. The second and third sets of constraints indicate that the delegator user u_1 must be a member of role r_2 which is the owner of resource re_1 for at least the same time interval for which re_1 is to be delegated to the delegatee role r_1. The reason for which the third constraint is also required is given soon. The fourth constraint indicates that the delegator role r_2 can delegate the ownership of its own resources only to its immediate children. The motivation behind this is that each role in the system can be associated with both operative and administrative tasks and thus, should have freedom to administer its own resources. As a consequence, when the right of creating new roles is delegated to a role, this is the delegatee role who becomes authorized to administer its own roles and not the delegator or any other role in the system. If *own-delegate*$(u_1, u_2, r_1, re_1, i, t_1)[T_1]$ succeeds, the OWNS database must be accordingly updated. This is accomplished by inserting tuple $(r_1, u_2, re_1)[T_1]$ into the OWNS database (if the mode is s tuple $(r_1, null, re_1)[T_1]$ is inserted into the OWNS database). Moreover, the time interval for which the delegator role is the owner of re_1 must be changed. We address this issue by the third constraint. This constraint requires the authorization function to search through tuples in the OWNS database regarding resource re_1 to see if the ownership of re_1 has already been delegated for some time points within T_1.

Using the concept of timed-ownership delegation, the owner of each resource within the system may change time to time. This mainly affects trusted policies (policies from which authorities originate), due to the fact that each trusted policy must now be associated with the time interval for which it is considered trusted. Consequently, in the OB-RBAC model, when at time point t_1 subject u_1 requests to issue a policy for time interval $[T_1]$ about resource re_1 and for target subjects who are members of role r_1, the authorization function checks the constraints described above to see if the policy is trusted. This is the case only if all those constraints are held.

Table 3 shows the OWNS database for the existing roles in our example at time point t_1, after enforcing the authorized timed-ownership delegation instances given in Table 2. We can see that $alice$ delegates $laptop_1$ to bob who is a member of role $Net.Sec.Pro.Man.$. In addition, bob delegates 10GB of his own disk storage $disk_2$ to $carol$ who is a member of role $Sen.Res.$.

2) Timed-Permission Delegation: As aforementioned, in order for delegation of activities down in the role hierarchy works satisfactorily, each delegatee role should be given the required resources and permissions. In the previous section we described delegation of resources. However, in some cases a delegator role instead of delegating the ownership of its own resources, may prefer to only delegate required permissions over its own resources. This is specially the case when a resource should be shared between more than one delegatee role. In the following, we describe delegation of permissions in the OB-RBAC.

Definition 4: We define the set of permissions P as follows:

- $p_{acc}(a, re) \in P$, if $a \in A$ (set of actions), and $re \in RE$.
- $p_{adm}(r, p) \in P$, if $r \in R$, and $p \in P$.

permissions of the form $p_{acc}(a, re)$ denote access permissions, while permissions of the form $p_{adm}(r, p)$ denote administrative permissions. $p_{acc}(a, re)$ is the right of performing action a on resource re. As we mentioned earlier, we leave the investigation of revocation mechanisms for the future work. Therefore, $p_{adm}(r_1, p_1)$ is the right of delegation of permission p_1 to role r_1.

While access delegation is a mechanism of assigning access permissions to subjects, Administrative delegation is used to distribute administrative permissions, through which decentralized administration of authorizations is achieved. In the OB-RBAC model, an authorized role can delegate to its immediate children firstly, the right of accessing its own resources and secondly, the right of issuing policies regarding such resources, using access and administrative delegation operations, respectively.

Definition 5: We define the set of timed-permission delegation statements T-PD as follows:
perm-delegate$(u, p, t)[T] \in$ T-PD if $u \in U$, p is an administrative permission, t is a time point, and $[T]$ is a time interval.

perm-delegate$(u_1, p_{adm}(r_1, p_1), t_1)[T_1] \in$ T-PD means that user u_1 at time point t_1 performs administrative permission $p_{adm}(r_1, p_1)$ which results in the delegation of

Table II
T-OD: AUTHORIZED TIMED-OWNERSHIP DELEGATION INSTANCES

Instance of Timed-Ownership Delegation	Delegator Role
own-delegate$(alice, bob, net.sec.pro.lea., laptop_1, i, t_1)[t_2, t_6]$	$alice \in sec.gro.man.$
own-delegate$(alice, null, net.sec.pro.lea., disk_2 : 50GBofdisk_1, s, t_2)[t_3, t_6]$	$alice \in sec.gro.man.$
own-delegate$(bob, carol, Sen.Res, disk_3 : 10GBofdisk_2, i, t_3)[t_3, t_6]$	$bob \in Net.Sec.Pro.Lea.$

Table III
OWNS DATABASE AFTER ENFORCING AUTHORIZED *own-delegate* OPERATIONS, $t_0 \leq t_i \leq t_n \ \ (1 \leq i \leq n)$

Role	User	Resource	t_{start}	t_{end}
$sec.gro.man.$	$null$	$net.sec.pro.lea.$	t_0	t_n
$sec.gro.man.$	$null$	$phy.sec.pro.lea.$	t_0	t_n
$sec.gro.man.$	$alice$	$laptop_1$	t_0	t_n
$sec.gro.man.$	$alice$	$laptop_2$	t_0	t_n
$sec.gro.man.$	$alice$	$disk_1$	t_0	t_n
$net.sec.pro.lea.$	$null$	$sen.res.$	t_0	t_n
$net.sec.pro.lea.$	bob	$laptop_1$	t_2	t_6
$net.sec.pro.lea.$	$null$	$disk_2$	t_3	t_6
$sen.res.$	$carol$	$disk_3$	t_3	t_6

permission p_1 for the time interval $[T_1]$ to role r_1. It should be noted that p_1 can assume two forms: $p_{acc}(a_1, re_1)$ and $p_{adm}(\check{r_1}, p_{acc}(a_1, re_1))$. In the first case, members of r_1 are authorized to perform action a_1 on resource re_1. In the second case, members of r_1 are authorized to delegate to the immediate children of role r_1 denoted by $\check{r_1}$ firstly, the right of performing a_1 on re_1 and secondly, the right of further delegation of the previous right to the immediate children of $\check{r_1}$. This is similar to the mechanism of delegation of administrative rights applied by the delegation profile [12] of the eXtensible Access Control Markup Language [13]. In the following, we describe under which circumstances operation *perm-delegate*$(u_1, p_{adm}(r_1, p_1), t_1)[T_1]$ succeeds.

Definition 6: Let $\vdash$ be the validates relation between two timed-permission delegation statements, where
perm-delegate$(u_1, p_{adm}(r_1, p_1), t_1)[T_1] \vdash$
perm-delegate$(u_2, p_{adm}(r_2, p_2), t_2)[T_2]$, if:
$(p_1 = p_{adm}(\check{r_1}, p_{acc}(a_1, re_1))) \ \wedge ((u_2, r_1) \in UR) \ \wedge (t_2 \in [T_1]) \wedge (p_2 = p_{acc}(a_1, re_1) \vee p_{adm}(\check{r_2}, p_{acc}(a_1, re_1))) \wedge (r_2 \in \check{r_1})$.

Each request for a timed-permission delegation operation *perm-delegate*$(u_n, p_{adm}(r_n, p_n), t_n)[T_n]$ is evaluated by the backward chaining process [12], in which a chain of permission delegation statements must be found, such that each delegation statement is validated by the next one. Such a process is successfully terminated only if the chain can be traced back to a permission delegation statement *perm-delegate*$(u_1, p_{adm}(r_1, p_1), t_1)[T_1]$ such that:

- $((r_0, null, re_1)[T_0] \in$ OWNS$) \vee ((r_0, u_1, re_1)[T_0] \in$ OWNS), where re_1 is a resource that all permissions $p_i \ 1 \leq i \leq n$ are about, and r_0 is a role that u_1 is a member of.
- $[T_i] \subseteq [T_0] \ (1 \leq i \leq n)$.

Each access request of the form $(u_n, p_{acc}(a_n, re_n), t_n)$ is also authorized in a similar way described in the following (such an authorization is required when u_n is not the owner of re_n at time point t_n):

- $\exists \, r_n$ such that $(u_n, r_n) \in UR \wedge$ $p_{acc}(a_n, re_n)$ has been assigned to r_n.
- The backward chaining process similar to the one described above is used to evaluate the access request. However, in this case, the process begins with a permission delegation statement of the following form: *perm-delegate*$(u_i, p_{adm}(r_n, p_n), t_i)[T_i]$ in which $p_n = p_{acc}(a_n, re_n)$.
- $t_n \in [T_0]$, where $[T_0]$ is the time interval for which the root delegator is the owner of resource re_n.

B. Discussion about Resource Creation Permission

As we mentioned earlier, in the OB-RBAC model there must be a top senior role, e.g. CEO, who is the first owner of the resources in the system. Among such resources are also roles created by this top senior role, to which it delegates some activities based on the competencies of each delegatee role. However, besides the top senior role, other roles may also require to introduce new resources such as roles, documents, and services into the system. In general, we argue that each role in the hierarchy is allowed to introduce new resources of specific types into the system, only if it has been delegated the required permission. For instance, in our example depicted in Figure 2, a member of role $NetworkSecurityProjectLeader$ is authorized to create roles $SeniorResearcher$ and $NetworkProgrammer$ only if it has been delegated permission $p_{acc}(create, role)$ through corresponding delegation permission. It should be noted that though resource creation is an administrative permission, we consider it as an access permission $p_{acc}(create, re)$, in order to keep it consistent with definition 4. This is also the case for user-role assignment and permission-role assignment actions.

IV. RELATED WORK

Administration and delegation in RBAC have been research tracks within the active area of RBAC. In the existing role-based administrative models, the actual administration of the administrative roles is centralized and it is assumed that there is a central unit of administration that has all the authority for creating and modifying the administrative roles and their relations. Thus, the authority to administrate the

administrative roles is not captured in the administrative or extended role hierarchy and requires yet another administrative level of role hierarchy [1]. This is in contrast to our proposed model, in which only one role hierarchy is built in such a way that all roles can actively participate in both operative and administrative tasks.

Delegation in RBAC is about delegation of rights available to a role to another role. This means that a user assigned to a specific role may let another user act on his behalf in activating that specific role. This is not the same as delegating administrative rights either in terms of assigning new users, resources and permissions to the role or introducing new resources into the system. In delegation models for RBAC [14], [15], [16], it is assumed that there are designated administrative roles or security officers controlling such administrative rights. This is an assumption that we don't have in the OB-RBAC model, as the administrative tasks are distributed among all roles.

V. Conclusion

In this paper we present a new class of RBAC model called OB-RBAC, which utilizes the administrative approach applied by DAC models. The main advantage of the OB-RBAC model is that it builds a policy model which fits more naturally with the structure of an organization and the activities it performs to satisfy its missions. This is particularly the case for organizations with flexible and informal structure and fully decentralized administrative procedures. Consequently, in our model, each role in the system is given a certain degree of control over resources and permissions in such a way that more mission-oriented decisions are made by senior roles and more specialized-level decisions are made by junior roles.

Future work will include the development of a prototype for the OB-RBAC model which can be used in different application scenarios. Furthermore, we plan to extend the OB-RBAC model in two directions. Firstly, incorporating models and mechanisms to support the supervision relation between the delegator and delegatee. Secondly, investigating different revocation schemes which can be applied to the model.

References

[1] B. Sadighi, "Decentralised privilege management for access control," Ph.D. dissertation, 2005.

[2] R. Sandhu, V. Bhamidipati, and Q. Munawer, "The ARBAC97 model for role-based administration of roles," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 1, pp. 105–135, 1999.

[3] S. Oh and R. Sandhu, "A model for role administration using organization structure," in *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*. New York, NY, USA: ACM, 2002, pp. 155–162.

[4] J. Crampton and G. Loizou, "Administrative scope: A foundation for role-based administrative models," *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 2, pp. 201–231, 2003.

[5] J. Crampton, "On permissions, inheritance and role hierarchies," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2003, pp. 85–92.

[6] C. Goh and A. Baldwin, "Towards a more complete model of role," in *RBAC '98: Proceedings of the third ACM workshop on Role-based access control*. New York, NY, USA: ACM, 1998, pp. 55–62.

[7] J. D. Moffett and E. C. Lupu, "The uses of role hierarchies in access control," in *RBAC '99: Proceedings of the fourth ACM workshop on Role-based access control*. New York, NY, USA: ACM, 1999, pp. 153–160.

[8] A. J. G. Silvius, "Business & IT alignment in theory and practice," in *HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences*. Washington, DC, USA: IEEE Computer Society, 2007, p. 211b.

[9] F. B. Tan and R. B. Gallupe, "A framework for research into business-IT alignment: a cognitive emphasis," pp. 50–73, 2003.

[10] R. J. Wieringa, J. Gordijn, and P. A. T. van Eck, "Value-based business-IT alignment in networked constellations of enterprises," in *1st International Workshop on Requirements Engineering for Business Need and IT Alignment (REBNITA 2005), Paris, France*, K. Cox, E. Dubois, Y. Pigneur, S. J. Bleistein, J. Verner, A. M. Davis, and R. J. Wieringa, Eds. University of New South Wales Press, August 2005, pp. 38–43.

[11] J. D. Moffett, "Control principles and role hierarchies," in *RBAC '98: Proceedings of the third ACM workshop on Role-based access control*. New York, NY, USA: ACM, 1998, pp. 63–69.

[12] E. Rissanen, "XACML v3.0 administration and delegation profile version 1.0," 2008, http://docs.oasis-open.org/[tc-short-name]/[additionalpath/filename].pdf.

[13] T. Moses, "eXtensible Access Control Markup Language," 2005, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

[14] E. Barka and R. Sandhu, "Framework for role-based delegation models," *Computer Security Applications, 2000. ACSAC '00. 16th Annual Conference*, pp. 168–176, Dec 2000.

[15] J. Crampton and H. Khambhammettu, "Delegation in role-based access control," *International Journal of Information Security (IJIS)*, vol. 7, no. 2, pp. 123–136, April 2008. [Online]. Available: http://dx.doi.org/10.1007/s10207-007-0044-8

[16] L. Zhang, G.-J. Ahn, and B.-T. Chu, "A rule-based framework for role-based delegation and revocation," *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 3, pp. 404–441, 2003.