Inheritance hierarchies in the Or-BAC model and application in a network environment

Frédéric Cuppens¹ Nora Cuppens-Boulahia¹ Alexandre Miège¹,²

¹GET/ENST Bretagne, BP 78, 2 rue de la Châtaigneraie, 35512 Cesson Sévigné Cedex, France
²GET/ENST, 46 rue Barrault, 75634 Paris Cedex 13, France

Abstract

Role hierarchy was first introduced in the Role Based Access Control (RBAC) model. Inheritance of permissions is associated with this hierarchy. This is useful to design security policies in a modular way. In this paper, we extend this approach in the context of the Organization Based Access Control (Or-BAC) model. We first define hierarchies of roles, views and activities and formally model inheritance mechanism associated with each hierarchy. We then define hierarchy of organizations. We show that this provides efficient means to derive policies of security components from corporate security policies specification. We illustrate our approach in the context of network security policy, in particular to configure firewalls.

1 Introduction

The inheritance mechanism was suggested in object oriented programming as an efficient way to design an application in a modular way. A similar mechanism is used in RBAC [14] when a hierarchy of roles is defined and associated with inheritance of permissions. The role hierarchy is a useful mean to structuring the security policy specification.

However, the concept of role hierarchy is not free of ambiguity [11, 12, 5]. Some of these ambiguities are directly related to the concept of role itself. If we consider the examples suggested in [14], there are actually several different interpretations of roles. Basically, a role allows a subject who is assigned to this role to perform some particular activities. This is the case of roles such as physician, nurse or medical secretary. However, in some examples, a role is related to a given organization [13]. For instance, we may consider roles such as nurse in a cardiological department or nurse in a reanimation team. This is interesting because permissions assigned to a given role may change from one organization to another. For example, a nurse may have different permissions if she performs her activities in a cardiological department or in a reanimation team. A role may also correspond to the activity of leading a given organization. Examples of such roles may be director of an hospital or head of a cardiological department.

As we shall see in the following, these different interpretations of the role concept do not behave similarly with respect to permission inheritance. This is why it is important to have a model that provides means to make explicit such differences.

We shall analyze this problem in the context of the Or-BAC model [10]. This model is centered on the concept of organization. In Or-BAC, an organization corresponds to any entity that is in charge of managing a set of security rules (permissions or prohibitions). For instance, a given hospital is an organization. A concrete security component, such as a firewall, may be also viewed as an organization since it manages a set of security rules.

Role definition in Or-BAC is always related to a given organization. This is useful to avoid some ambiguities when defining role hierarchies and associating them with permission inheritance.

The Or-BAC model considers two other concepts, namely activity and view. A security policy assigned to a given organization is defined as permissions (or prohibitions) for roles to perform activities on views. In the following, we shall suggest defining hierarchies of activities and views and model inheritance mechanism associated with these hierarchies.

Finally, in Or-BAC, we can also define hierarchy of organizations. This possibility provides a very efficient mean to structuring the security policy specification, starting with high level organization such as an hospital and finishing with concrete security components such as a firewall.

In this paper, we aim to analyze and formally model inheritance of permissions and prohibitions through these different hierarchies. The remainder of this paper is organized as follows. Section 2 recalls main concepts of Or-BAC. Section 3 presents the various hierarchies respectively associated with roles, activities and views. Section 4 studies organization hierarchies. In section 5, we summarize how to specify a security policy in Or-BAC when hierarchies are used. Section 6 shows how our approach applies to network se-
curity policy specification. Finally, section 7 concludes and suggests several issues to this work.

2 Or-BAC

2.1 Basic concepts of Or-BAC

Or-BAC [10] is an access control model based on the organization concept. In Or-BAC, different organizations can specify their own access control policy using eight basic sets of entities: Organization, Role, Activity, View, Subject, Action, Object and Context. Basic predicates used in Or-BAC to model relationships between these eight entities are summarized in table 1.

As mentioned in the introduction, an organization is any entity that manages a set of security rules. A subject is an active entity that may be assigned to a role. We shall assume that $\text{Org} \subseteq \text{Subject}$ so that a role may be assigned to an organization. For instance, roles “casualty department” or “rescue team” may be assigned to some organizations.

By means of the entity Role, we are able to structure the subjects and to update easily security policies when new subjects are added to the system. Since we have also to structure the objects and to add new objects to the system, a similar entity regarding objects is needed: the entity View. Roughly speaking, as in relational databases, a view corresponds to a set of objects that satisfy a common property.

Another entity is used to abstract actions: the entity Activity. Seeing that roles associate subjects that fulfill the same functions and views correspond to sets of objects that satisfy a common property, activities will join actions that share the same principles.

Subjects, objects and actions may have attributes. This is modeled by a set of binary predicates having the form $\text{attr}(\text{ent}, \text{val})$ where ent is a subject, an object or an action and val is the value of attribute attr. For instance, if med27 is a medical record, then name(med27, John) means that med27 is John’s medical record.

We assume that $\text{Subject} \subseteq \text{Object}$ so that we can define views of subjects that we call groups. In Or-BAC, there is a clear difference between a role and a group. Permissions are assigned to roles whereas a group is simply a set of subjects that have some common properties. However, it is sometimes useful to assign the same role to every subject belonging to a given group. For this purpose, we can use the predicate $G_{\text{Empower}}(\text{org}, \text{group}, \text{role})$ and specify the following rule:

- GE: $\forall \text{org}, \forall \text{group}, \forall \text{role}, \forall \text{subject}$,
  $\text{Use}(\text{org}, \text{subject}, \text{group}) \land$ $G_{\text{Empower}}(\text{org}, \text{group}, \text{role})$ $\rightarrow \text{Empower}(\text{org}, \text{subject}, \text{role})$

Since the Or-BAC model allows the administrator to specify that some permission or prohibition only applies in specific contexts, we also introduce the entity Context. Contexts are defined by logical rules whose conclusion is the predicate Define (see table 1 that gives the example of the working_hours context). We say that a context $\text{c}$ in organization $\text{org}$ is defined by condition cond when there is a rule 1 having the form: $\text{Define}($org, s, a, o, c$) \leftarrow \text{cond}$. Specifying contexts in Or-BAC is further analyzed in [7].

2.2 Permission and prohibition

Permissions and prohibitions in Or-BAC are defined with predicates defined in figure 2. The access control policy is specified at two different levels: an abstract level that specifies permissions and prohibitions between role, activity and view, and a concrete level where permissions and prohibitions between subject, action and object are derived.

These two levels are related as follows. In a given organization $\text{org}$, a subject $\text{s}$ is permitted to perform an action $\text{a}$ on an object $\text{o}$ if (1) $\text{s}$ is empowered to play a given role $\text{r}$ in $\text{org}$ and (2) $\alpha$ implements a given activity $\text{a}$ in $\text{org}$ and (3) $\text{o}$ is used in a given view $\text{v}$ by $\text{org}$. If these three conditions are satisfied and if (4) the organization $\text{org}$ grants to role $\text{r}$ the permission to perform the activity $\text{a}$ on the view $\text{v}$, then the request by the subject $\text{s}$ to perform the action $\text{a}$ on the object $\text{o}$ is accepted. Deriving concrete permissions from abstract permissions is modelled by the following rule:

- $\text{RG}_1$: $\forall \text{org}, \forall \text{r}, \forall \text{a}, \forall \text{v}, \forall \text{s}, \forall \text{o}, \forall \text{c}$,
  $\text{Permission}(\text{org}, \text{r}, \text{a}, \text{v}, \text{c}) \land$
  $\text{Empower}(\text{org}, \text{s}, \text{r}) \land$
  $\text{Consider}(\text{org}, \text{a}, \text{r}) \land$
  $\text{Use}(\text{org}, \text{a}, \text{v}) \land$
  $\text{Define}(\text{org}, \text{s}, \text{a}, \text{o}, \text{c})$ $\rightarrow \text{Is permitted}(\text{s}, \text{a}, \text{o})$

Another similar rule (called $\text{RG}_2$) is used to derive concrete prohibitions from abstract prohibitions.

2.3 Constraints

Constraints that apply to an access control policy was first suggested in the RBAC model (more precisely, in the RBAC2 sub-model [8]) and further analyzed in [1]. To specify constraints in the Or-BAC model, we introduce a predicate error(). A constraint is then modelled as a rule whose conclusion is error() (as suggested in [3, 9]).

For instance, we may specify that, in hospital $\text{H}$, a subject cannot be empowered in both roles anesthetist and surgeon:

- $\text{C}_1$: $\forall \text{s},$
  $\text{Empower}(\text{H}, \text{s}, \text{anesthetist}) \land$
  $\text{Empower}(\text{H}, \text{s}, \text{surgeon})$
  $\rightarrow \text{error}()$

1: We can assume that each context $\text{c}$ is defined by a unique rule by using the fact that $\text{Define}(\text{org}, \text{s}, \text{a}, \text{o}, \text{c}) \leftarrow \text{cond}_1, \ldots, \text{Define}(\text{org}, \text{s}, \text{a}, \text{o}, \text{c}) \leftarrow \text{cond}_n$ is equivalent to $\text{Define}(\text{org}, \text{s}, \text{a}, \text{o}, \text{c}) \leftarrow (\text{cond}_1 \lor \ldots \lor \text{cond}_n)$. 
<table>
<thead>
<tr>
<th>Predicate name</th>
<th>Domain</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relevant_role</td>
<td>Org × Role</td>
<td>If $org$ is an organization and $r$ a role, then $\text{Relevant_role}(org, r)$ means that playing role $r$ is defined in organization $org$. Ex: $\text{Relevant_role}(H, \text{physician})$</td>
</tr>
<tr>
<td>Relevant_activity</td>
<td>Org × Activity</td>
<td>If $org$ is an organization and $a$ an activity, then $\text{Relevant_activity}(org, a)$ means that performing activity $a$ is defined in organization $org$. Ex: $\text{Relevant_activity}(H, \text{consult})$</td>
</tr>
<tr>
<td>Relevant_view</td>
<td>Org × View</td>
<td>If $org$ is an organization and $v$ is a view, then $\text{Relevant_view}(org, o, v)$ means that using view $v$ is defined in organization $org$. Ex: $\text{Relevant_view}(H, \text{medical_record})$</td>
</tr>
<tr>
<td>Empower</td>
<td>Org × Subject × Role</td>
<td>If $org$ is an organization, $s$ a subject and $r$ a role, then $\text{Empower}(org, s, r)$ means that $org$ empowers subject $s$ in role $r$. Ex: $\text{Empower}(H, \text{John}, \text{physician})$</td>
</tr>
<tr>
<td>Consider</td>
<td>Org × Action × Activity</td>
<td>If $org$ is an organization, $a$ an action and $a$ an activity, then $\text{Consider}(org, a, a)$ means that $org$ considers that action $a$ falls within the activity $a$. Ex: $\text{Consider}(H, \text{&quot;SELECT&quot;}, \text{consult})$</td>
</tr>
<tr>
<td>Use</td>
<td>Org × Object × View</td>
<td>If $org$ is an organization, $o$ an object and $v$ is a view, then $\text{Use}(org, o, v)$ means that $org$ uses object $o$ in view $v$. Ex: $\text{Use}(H, \text{med_27}, \text{medical_record})$</td>
</tr>
<tr>
<td>Define</td>
<td>Org × Subject × Action × Object × Context</td>
<td>If $org$ is an organization, $s$ a subject, $a$ an action, $o$ an object and $c$ a context, then $\text{Define}(org, s, a, o, c)$ means that within organization $org$, context $c$ holds between subject $s$, action $a$ and object $o$. Ex: $\forall s, \forall a, \forall o, \text{Define}(H, s, a, o, \text{working_hours})$</td>
</tr>
</tbody>
</table>

In the following, we shall consider the following constraint that apply to any organization:

- $C_2$: $\forall org, \forall s, \forall r, \neg \text{Empower}(org, s, r) \land \neg \text{Relevant\_role}(org, r) \rightarrow \text{error}()$

Rule $C_2$ says that an organization $org$ should not empower a subject $s$ in role $r$ if role $r$ is not relevant in organization $org$.

There are other rules similar to $C_2$ but for activities (called $C_3$), views (called $C_4$), permissions (called $C_5$) and prohibitions (called $C_6$).

### 3 Hierarchy within an organization

In Or-BAC, it is possible to consider hierarchies of roles (as suggested in [8]) but also of views and activities. Every hierarchy respectively defines a partial order relation over the set of roles, views and activities. We present general inheritance rules of permissions and prohibitions associated with these different hierarchies.

#### 3.1 Role hierarchy

Let us first address the case of inheritance between roles. In every organization, it is possible to associate a set of roles with a hierarchy. For this purpose, we introduce the predicate $\text{sub\_role}(org, r_1, r_2)$: in organization $org$, role $r_1$ is a sub-role of $r_2$.

Notice that the role hierarchy depends on the organization. This means that the hierarchy may vary from one organization to another. Let us now model inheritance principles associated with this hierarchy.

Permission inheritance through the role hierarchy is modelled by the following rule:

- $RH_1: \forall org, \forall r_1, \forall r_2, \forall a, \forall v, \forall c, \exists e, \text{sub\_role}(org, r_1, r_2) \land \text{Permission}(org, r_2, a, v, c) \rightarrow \text{Permission}(org, r_1, a, v, c)$

This rule says that if role $r_1$ is a sub-role of role $r_2$ in organization $org$, then every permission assigned to role $r_2$ in organization $org$ is also assigned to role $r_1$.  

---

**FIG. 1** – Basic predicates of Or-BAC
Regarding prohibition inheritance, things are more complex. It is necessary to recognize that the relationships between roles in the hierarchy may be semantically different. We actually identify two different relationships:

- Relationship of specialization/generalization. For instance, role surgeon is a specialization of role physician. To model this first relationship we shall use the following predicate:
  \[
  \text{specialized\_role}(org;r_1;r_2): \text{in organization}\ org, \text{role}\ r_1 \text{ is a specialized role of role}\ r_2.
  \]

- Relationship of organizational hierarchy. For instance, role department director may be defined as hierarchically higher than role team head. This second relationship is modelled by the following predicate:
  \[
  \text{senior\_role}(org;r_1;r_2): \text{in organization}\ org, \text{role}\ r_1 \text{ is a senior role of role}\ r_2.
  \]

We consider that relationship specialized\_role is included in sub\_role:

- RH\(_2\):
  \[
  \forall org, \forall r_1, \forall r_2, \\
  \text{specialized\_role}(org;r_1;r_2) \\
  \rightarrow \text{sub\_role}(org;r_1;r_2)
  \]

The consequence is that rule RH\(_2\) applies to the specialization role hierarchy and thus permissions are inherited through this hierarchy. We also consider that prohibitions are inherited through the specialization role hierarchy:

- RH\(_3\):
  \[
  \forall org, \forall r_1, \forall r_2, \forall a, \forall v, \forall c, \\
  \text{specialized\_role}(org;r_1;r_2) \\
  \\
  \text{prohibition}(org,r_2,a,v,c) \\
  \rightarrow \text{prohibition}(org,r_1,a,v,c)
  \]

For instance, every prohibition of the role physician is inherited by the role surgeon. This is compatible with the intuition that a surgeon is a special case of physician.

By contrast, the relationship senior\_role is generally not included in sub\_role. This means that we may have:

- \(\exists org, \exists r_1, \exists r_2, \\
  \text{senior\_role}(org;r_1;r_2) \land \neg \text{sub\_role}(org;r_1;r_2)\)

For instance, we can consider that role hospital director is hierarchically higher than physician. However, in some hospitals, role hospital director is a purely administrative role that is not assigned to a physician. In this case, there is no reason to conclude that hospital director is a sub role of physician.

Now let us assume that role r1 is a senior role of r2 and that r1 is also a sub-role of r2. In this case, the idea is to consider that r1 is “more powerful” than r2. This is compatible with rule RH\(_1\) above that specifies that r1 inherits the permissions assigned to r2. However, if we assume that r1 inherits the prohibitions assigned to r2, this will not make r1 more powerful than r2. This is why it would be better to consider that, in case of organizational hierarchy, prohibitions are inherited “upward”. This is modelled by the following rule:

- RH\(_4\):
  \[
  \forall org, \forall r_1, \forall r_2, \forall a, \forall v, \forall c, \\
  \text{sub\_role}(org;r_1;r_2) \land \text{senior\_role}(org;r_1;r_2) \land
  \]

<table>
<thead>
<tr>
<th>Predicate name</th>
<th>Domain</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permission</td>
<td>Org × Role × Activity × View × Context</td>
<td>If org is an organization, r a role, α an activity, v a view and c a context, then Permission(org,r,α,v,c) means that organization org grants to role r the permission to perform activity α on view v in context c. Ex: Permission(H,physician,consult,medical_record,working_hours)</td>
</tr>
<tr>
<td>Prohibition</td>
<td>Org × Role × Activity × View × Context</td>
<td>If org is an organization, r a role, α an activity, v a view and c a context, then Prohibition(org,r,α,v,c) means that organization org prohibits role r from performing activity α on view v in context c. Ex: Prohibition(H,nurse,consult,medical_record,night)</td>
</tr>
<tr>
<td>Is_permitted</td>
<td>Subject × Action × Object</td>
<td>If s is a subject, α an action, o an object, then Is_permitted(s,α,o) means that s is concretely permitted to perform action α on object o. Ex: Is_permitted(John,&quot;SELECT&quot;,med,27)</td>
</tr>
<tr>
<td>Is_prohibited</td>
<td>Subject × Action × Object</td>
<td>If s is a subject, α an action, o an object, then Is_prohibited(s,α,o) means that s is concretely prohibited to perform action α on object o. Ex: Is_prohibited(Mary,&quot;DELETE&quot;,med,27)</td>
</tr>
</tbody>
</table>
We consider that a similar rule (called AH$_2$) applies to inheritance of prohibitions. For instance, let us assume that, in hospital $H$, nurses are prohibited to manage medical records. Applying rule AH$_2$ we can derive that nurses are also prohibited to create, consult and update medical records.

### 3.3 View hierarchy

Using a similar approach, the set of views is associated with a hierarchy that depends on the organization. This is modelled by the predicate *sub_view*($org,v_1,v_2$): in organization $org$, view $v_1$ is a sub-view of $v_2$.

Our interpretation is that, in $org$, view $v_1$ is a specialization of view $v_2$. This is actually close to class inheritance hierarchy used in object-oriented hierarchy (Isa hierarchy).

In the context of Or-BAC, view hierarchies are associated with the permission inheritance that is modelled by the following rules:

- $\text{VH}_1: \forall org,\forall r,\forall a,\forall v_1,\forall v_2,\forall c,\exists\text{ Permission}(org,r,a,v_1,c) \land \text{ sub_view}(org,v_1,v_2) \rightarrow \text{ Permission}(org,r,a,v_2,c)$

A similar rule (called VH$_e$) applies to inheritance of prohibitions. For instance, in a hospital, we may consider that the view surgeon_record is a sub-view of the view medical_record. In this case, a role who is permitted or prohibited to perform a given activity on the view medical_record will be permitted or prohibited to perform the same activity on the view surgeon_record.

We can also define the concept of derived view as a special case of view specialization. Hence, we can define that a view $v_1$ is derived from view $v_2$ if there is a rule having the following form:

- $\forall org,\forall obj,\forall v_1,\forall v_2,\forall c,\exists\text{ Permission}(org,\text{ Use}(org,\text{ obj},v_2,c)) \land \text{ Condition} \rightarrow \text{ Use}(org,\text{ obj},v_1)$

where Condition is a logical condition used to specialize view $v_2$ into view $v_1$.

### 4 Organization hierarchy

Previous section showed how to define hierarchies between roles, activities and views within a given organization. In this section, we study how to define hierarchies of organizations. For this purpose, we introduce the predicate *sub_organization*($org_1,org_2$): organization $org_1$ is a sub-organization of organization $org_2$. We assume that this predicate defines a partial order relation on the set of organizations.

For instance, if $H$ is an hospital and dept8 is the casualty department of this hospital, then we have: $\text{sub_organization}(dept8,H)$
We may actually require that every sub-organization \( \text{org}_1 \) of a given organization \( \text{org}_2 \) is assigned to a role. This requirement is modelled by the following constraint:

\[-C_7 \land \neg \text{Empower}(\text{org}_2, \text{org}_1, r) \rightarrow \text{error}() \]

For instance, in the above example, constraint \( C_7 \) is satisfied if we have: Empower(\( H, \text{dept}8, \text{casualty}_\text{dept}) \).

Notice that some roles may be defined in a given organization but not in some of its sub-organizations. For instance if \( \text{dept}7 \) is the management department of the hospital \( H \), then the role nurse may be not defined in \( \text{dept}7 \) (we have \( \neg \text{Relevant}\_\text{role}(\text{dept}7, \text{nurse}) \)) whereas it is defined in \( H \) (we have \( \text{Relevant}\_\text{role}(H, \text{nurse}) \)).

Conversely, if \( \text{org}_1 \) is a sub-organization of \( \text{org}_2 \), then some roles may be defined in \( \text{org}_1 \) whereas they are not defined in \( \text{org}_2 \).

Similar comments apply to views and activities: if \( \text{org}_1 \) is a sub-organization of \( \text{org}_2 \), then the views (resp. activities) defined in \( \text{org}_1 \) may be disjoint from the views (resp. activities) defined in \( \text{org}_2 \).

### 4.1 Hierarchy inheritance

Let us assume that \( \text{org}_1 \) is a sub-organization of \( \text{org}_2 \). For those roles of \( \text{org}_2 \) that are relevant in \( \text{org}_1 \), we consider that the role hierarchy defined in \( \text{org}_2 \) also applies in \( \text{org}_1 \).

This is modelled by the following rule:

\[-\text{HH}_1 \land \neg \text{Relevant}\_\text{role}(\text{org}_2, r) \land \neg \text{relevant}\_\text{role}(\text{org}_1, r) \rightarrow \text{error}() \]

Similar principles apply to inheritance of specialized role hierarchy and also of activity and view hierarchies through the organization hierarchy. Thus, we obtain three other rules (respectively called \( \text{HH}_2, \text{HH}_3 \) and \( \text{HH}_4 \)) by replacing the \( \text{sub}\_\text{role} \) predicate in rule \( \text{HH}_1 \) by the \( \text{specialized}\_\text{role} \) predicate (resp. the \( \text{sub}\_\text{activity} \) and \( \text{sub}\_\text{view} \) predicates).

### 4.2 Permission and prohibition inheritance

We accept similar principles for inheritance of permissions and prohibitions through the organization hierarchy provided that the role, activity and view in the scope of the permission or prohibition are relevant in the sub-organization. This is modelled by the following rule:

\[-\text{OH}_1 \land \neg \text{Relevant}\_\text{activity}(\text{org}_2, r) \land \neg \text{relevant}\_\text{activity}(\text{org}_1, r) \land \neg \text{Permission}(\text{org}_2, r, a, v, c) \rightarrow \text{error}() \]

A similar rule applies to inheritance of prohibition (rule called \( \text{OH}_2 \)).

### 5 Specifying a security policy in Or-BAC

#### 5.1 Policy theory

To summarize, a security policy that includes inheritance hierarchies is modelled as a logical theory corresponding to the following definition.

**Definition 1:** In the Or-BAC model, a security policy \( \text{pol} \) is modelled as a logical theory \( T_{\text{pol}} \) defined as follows:

- Sets of facts using predicates \( \text{Relevant}\_\text{role} \), \( \text{Relevant}\_\text{activity} \) and \( \text{Relevant}\_\text{view} \)
- Sets of facts using predicates \( \text{Empower}, \text{Use}, \text{Consider}, \text{Permission} \) and \( \text{Prohibition} \)
- Rule \( GE \) and facts using predicate \( G\_\text{Empower} \)
- A set of rules for derived view definition (section 3.3)
- A set of facts using attribute binary predicates for describing attribute values of subjects, actions and objects
- A set of context definition rules, i.e. rules whose conclusion is the predicate \( \text{Define}(\text{org},\alpha,\alpha,\alpha) \)
- Sets of facts (inheritance hierarchies) using predicates \( \text{sub}\_\text{role}, \text{specialized}\_\text{role}, \text{sub}\_\text{activity} \) and \( \text{sub}\_\text{view} \)
- Rules \( \text{RG}_1 \) and \( \text{RG}_2 \) for deriving concrete permissions and prohibitions (section 2.1)
- Rules \( \text{RH}_1 \) to \( \text{RH}_4 \) (role inheritance rules), \( \text{AH}_1 \) and \( \text{AH}_2 \) (activity inheritance rules) and \( \text{VH}_1 \) and \( \text{VH}_2 \) (view inheritance rules)
- Rules \( \text{HH}_1 \) to \( \text{HH}_4 \) (hierarchy inheritance rules)
- Rules \( \text{OH}_1 \) and \( \text{OH}_2 \) (organization inheritance rules)
- A set of constraints, i.e. rules whose conclusion is the predicate \( \text{error}() \).

**Definition 2:** The security policy \( \text{pol} \) violates a constraint if it is possible to derive \( \text{error}() \) from \( T_{\text{pol}}: T_{\text{pol}} \vdash \text{error}() \)

#### 5.2 Conflicts

Since the Or-BAC model provides means to specify both permissions and prohibitions, it is possible that some conflicts arise. This occurs when a given user is both permitted and prohibited to perform a given action on a given object. To model such a situation, we introduce a predicate called \( \text{conflict}() \). The following rule specifies a situation of conflict:

\[-\text{RC} \land \neg \text{Is}\_\text{permitted}(s,\alpha,\alpha) \land \neg \text{Is}\_\text{prohibited}(s,\alpha,\alpha) \rightarrow \text{conflict}() \]

**Definition 3:** There is a conflict in the security policy \( \text{pol} \) if it is possible to derive \( \text{conflict}() \) from \( T_{\text{pol}}: T_{\text{pol}} \vdash \text{conflict}() \)
Notice that exceptions in the inheritance rules may lead to conflicts. For instance, in the example presented in section 3.1, a surgeon may be both prohibited to consult a given medical record of someone who is not his or her patient (prohibition inherited from role physician) and permitted to do so (explicit permission assigned to role surgeon).

In [6], we suggest managing such a conflict by assigning priority to permissions and prohibitions. In our previous example, prohibition inherited from physician should have lower priority than explicit permission assigned to surgeon and thus this surgeon should be finally permitted to consult the medical record. However, this is not the purpose of this paper to further discuss how to manage conflicts in the Or-BAC model (see [6] for a detailed presentation).

6 Application

In this section, we model a local area network, its security architecture and its connectivity to the Internet. We choose to take back the example used in Firmato [2] so as to bring out how Or-BAC provides a natural statement of various entities and concepts used in the security architecture. Furthermore, we show that hierarchy notions of extended Or-Bac applied to the central entities, say organization, role, activity and view, avoid the use of artifices like "open" (prohibition inherited from role physician) and "closed" groups suggested in [2].

6.1 Organization hierarchy

We want to model the access control policy of a corporate network used in an organization $H$. $H$ has a two-firewall network configuration, as shown in Figure 3. As presented in [2], the external firewall guards the corporation’s Internet connection. Behind it is the DMZ, which contains the corporation’s externally visible servers. In our case these servers provide http/https (web), ftp, smtp (e-mail), and dns services. The corporation actually only uses two hosts to provide these services, one for dns and the other (called Multi_server) for all the other services. Behind the DMZ is the internal firewall which guards the corporation’s intranet. This firewall actually has three interfaces: one for the DMZ, one for the private network zone, and a separate interface connecting to the firewall administration host.

Within the private network zone, there is one distinguished host, Admin_serv, which provides the administration for the servers in the DMZ.

In Or-BAC, we can introduce several organizations to model such a configuration. First, there is an organization $H$ and to simplify, we shall actually identify $H$ with its corporate network. $H$ has two sub-organizations denoted $H_{fw1}$ corresponding to the external firewall and $H_{fw2}$ corresponding to the internal firewall. We may actually introduce other organizations, such as $H_{private-net}$ if one would like to specify the policy to be enforced within $H$ private network. For instance, if $H$ is an hospital, we might introduce roles such as physician, nurse, etc., to model this part of the policy. However, for the sake of simplicity, we shall not further refine this part. Notice that we could also use an organization called internet if we had to specify an explicit policy to be enforced by the Internet.

6.2 Subject

In this example, subjects correspond to hosts identified by their IP address. So if $h$ is an host, then predicate $address(h,a)$ means that the IP address of $h$ is $a$. Roles are assigned to hosts as suggested in section 6.3 below. For this purpose, predicate Empower enables us to assign a role to a given host. However, it would easier to cluster hosts into groups (also called zone in Firmato) and use $G_{Empower}$ to assign the same role to every host belonging to the same group. For instance, we can define the group $Private_net$ as follows:

$$\neg \exists h, Use(H,h,Private_net) \land Use(H,h,Host) \land address(h,a) \land a \in 111.222.2.*$$

$$\land \neg Use(H,h,Firewall_interface)$$

6.3 Role

Hosts may be assigned to roles presented in figure 4. All these roles are relevant to $H$. Figure 4 specifies those roles that are respectively relevant to organizations $H_{fw1}$ and $H_{fw2}$. Notice that role Firewall is relevant to $H_{fw1}$ (for administration purpose) but not to $H_{fw2}$. For each role, figure 4 also presents the sub-roles of this role. In this example, the sub-role hierarchy actually corresponds to a specialization role hierarchy.

6.4 Activity

Activities correspond to various services available in corporate network $H$. We define a first activity all_tcp with different tcp activities (such as smtp, ssh and https) as sub-activities. Similarly, we define an activity all_icmp with different icmp activities (such as ping) as sub-activities. We also define two other activities, admin_to_gw and gwy_to_admin has also two sub-activities: ssh and https. All these activities are relevant in organizations $H$, $H_{fw1}$ and $H_{fw2}$.

The main difference here with the approach suggested in [2] is that we use hierarchies of activities whereas Firmato defines elementary services and groups of services. Our approach is more generic since permissions and prohibitions will all apply to a unique entity, the activity.

6.5 View

Views are used to structure objects on which network services apply. Thus, we define a view called target having
two attributes: content that corresponds to messages transmitted when using the service and dest that corresponds to the destination host of the service. The destination host is identified by its role.

Actually, the content attribute is not used in the example because we shall only consider filtering rules on the destination host. However, it would be useful to filter messages depending on their content.

We can then define sub-views derived from view target according to the role assigned to the destination host. For instance, we can define sub-view to dns as follows:

\[
\forall o. Use(H,o, to_dns) \leftarrow Use(H,o, target) \land dest(o, dns)
\]

This would lead to define as many views as there are roles. This would be quite fastidious. Instead, we suggest defining a function to target from roles into views. Views created by function to target are defined as follows:

\[
\forall o, r. Use(H,o, to_target(r)) \leftarrow Use(H,o, target) \land dest(o, r)
\]

We consider that a view to target(r) is relevant in one of the organization of our example if r is a role relevant in this organization. We also consider that if role r1 is a sub-role of role r2, then view to target(r1) is a sub-view of view to target(r2).

### 6.6 Security policy

We can now specify different permissions that apply to organization H. These permissions correspond to the security policy presented in [2]. Figure 5 lists how these permissions are modelled in Or-BAC. For the sake of simplicity, we do not specify prohibitions and all permissions are supposed to apply in every context (corresponding to default context that is always evaluated to true).

Compared to Firmato, one significant advantage of our approach is that it enables us to automatically derive permissions that respectively apply to H_fw1 and H_fw2 (using rule OH1 for deriving permissions in sub-organizations of H). The results we obtain for H_fw1 is presented in figure 6. To illustrate this derivation process.

<table>
<thead>
<tr>
<th>Role name</th>
<th>Description</th>
<th>sub_role</th>
<th>Relevant to H_fw1</th>
<th>Relevant to H_fw2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public_host</td>
<td>Role assigned to hosts in view Public_Net</td>
<td></td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Private_host</td>
<td>Role assigned to hosts in view Private_Net</td>
<td></td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Firewall</td>
<td>Role assigned to firewall interfaces</td>
<td></td>
<td>External_firewall</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Internal_firewall</td>
<td></td>
</tr>
<tr>
<td>External_firewall</td>
<td>Role assigned to external firewall interfaces</td>
<td></td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Internal_firewall</td>
<td>Role assigned to internal firewall interfaces</td>
<td></td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>DNS_server</td>
<td>Role assigned to the DNS server</td>
<td></td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Ftp_server</td>
<td>Role assigned to ftp server</td>
<td></td>
<td>Multi_server</td>
<td>X</td>
</tr>
<tr>
<td>Mail_server</td>
<td>Role assigned to mail server</td>
<td></td>
<td>Multi_server</td>
<td>X</td>
</tr>
<tr>
<td>Web_server</td>
<td>Role assigned to web server</td>
<td></td>
<td>Multi_server</td>
<td>X</td>
</tr>
<tr>
<td>Multi_server</td>
<td>Role assigned to multi-server</td>
<td></td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Adm_fw_host</td>
<td>Role assigned to hosts in view Admin_gtw</td>
<td></td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Adm_serv_host</td>
<td>Role assigned to hosts in view Admin_serv</td>
<td></td>
<td>-</td>
<td>X</td>
</tr>
</tbody>
</table>
let us consider the following permission:

\[ \text{Permission}(H;\text{adm}_1;\text{fw}_1;\text{host}, \text{admin}_1;\to;\gtwy, \\
\quad \text{to;target;}(\text{firewall}), \text{default}) \]

Since role \( \text{adm}_2;\text{fw}_2;\text{host} \), activity \( \text{admin}_2;\to;\gtwy \) and view \( \text{to;target;}(\text{firewall}) \) are relevant in \( H;\text{fw}_2 \), we can apply rule \( \text{OH}_1 \) to derive:

\[ \text{Permission}(H;\text{fw}_2;\text{adm}_2;\text{fw}_2;\text{host}, \text{admin}_2;\to;\gtwy, \\
\quad \text{to;target;}(\text{firewall}), \text{default}) \]

However, since view \( \text{to;target;}(\text{firewall}) \) is not relevant in \( H;\text{fw}_1 \), we cannot derive a similar permission for \( H;\text{fw}_1 \). But, view \( \text{to;target;}(\text{external;firewall}) \) is a sub-view of \( \text{to;target;}(\text{firewall}) \). Since \( \text{to;target;}(\text{external;firewall}) \) is a relevant view in \( H;\text{fw}_1 \), we can apply rules \( \text{VH}_1 \) and \( \text{OH}_1 \) to derive:

\[ \text{Permission}(H;\text{fw}_1;\text{adm}_1;\text{fw}_1;\text{host}, \text{admin}_2;\to;\gtwy, \\
\quad \text{to;target;}(\text{external;firewall}), \text{default}) \]

Notice that one permission, namely:

\[ \text{Permission}(H;\text{private;host}, \text{all}_1;\text{tcp}, \\
\quad \text{to;target;}(\text{public;host}), \text{default}) \]

is not inherited by \( H;\text{fw}_1 \) nor \( H;\text{fw}_2 \). This is because role \( \text{private;host} \) is only relevant to \( H;\text{fw}_2 \) whereas view \( \text{target;}(\text{public;host}) \) is only relevant to \( H;\text{fw}_1 \). So, no firewall alone can manage this permission. In this case, our proposal is to use this permission to configure both firewalls.

Our approach is actually based on fewer concepts than Firmato. In particular, we do not need to use the notion of “closed” groups. A closed group does not inherit from higher groups in the hierarchy. The example suggested in Firmato is the firewall group that should not inherit from private_host. We guess that the notion of closed group is complex to manage and actually not necessary. In our approach, we have simply to specify that private-net does not include firewall-interface (see the definition suggested for private-net in section 6.2). Our approach can also be used to handle more complex applications that include prohibitions, requirements on message contents or contextual rules.

7 Conclusion

In this paper we show how to model inheritance hierarchies in the Or-BAC model. Previous works related to inheritance hierarchies only considered role hierarchies (as suggested in the RBAC model). By contrast, we define role, activity, view and organization hierarchies and analyze inheritance of both permissions and prohibitions through these hierarchies.

Regarding the role hierarchy, we show that it is useful to distinguish between two different hierarchies: the specialization/generalization role hierarchy and the senior/junior role hierarchy. Permissions are inherited “downward” in both hierarchies (the more specialized role inherits from the less specialized role and the senior role inherits from the junior role). However, we suggest that prohibitions are inhe-

References


Permission(H,adm_fw_host,admin_to_gtwy_to_target(firewall),default)
Permission(H,firewall,gtwy_to_admin_to_target(adm_fw_host),default)
Permission(H,private_host,all_tcp_to_target(public_host),default)
Permission(H,adm_server_host,all_tcp_to_target(dns_server),default)
Permission(H,adm_server_host,all_tcp_to_target(multi_server),default)
Permission(H,public_host,smtp_to_target(mail_server),default)
Permission(H,public_host,ftp_to_target(ftp_server),default)
Permission(H,public_host,https_to_target(web_server),default)
Permission(H,private_host,smtp_to_target(mail_server),default)
Permission(H,private_host,ftp_to_target(ftp_server),default)
Permission(H,private_host,https_to_target(web_server),default)
Permission(H,dns_server,dns_to_target(public_host),default)
Permission(H,ftp_server,ftp_to_target(public_host),default)
Permission(H,dns_server,dns_to_target(private_host),default)
Permission(H,ftp_server,ftp_to_target(private_host),default)

Fig. 5 — Permissions in organization H

Permission(H_fw,adm_fw_host,admin_to_gtwy_to_target(external_firewall),default)
Permission(H_fw,external firewall,gtwy_to_admin_to_target(adm_fw_host),default)
Permission(H_fw,public_host,smtp_to_target(mail_server),default)
Permission(H_fw,public_host,dns_to_target(dns_server),default)
Permission(H_fw,public_host,ftp_to_target(ftp_server),default)
Permission(H_fw,public_host,https_to_target(web_server),default)
Permission(H_fw,ftp_server,ftp_to_target(public_host),default)
Permission(H_fw,dns_server,dns_to_target(public_host),default)
Permission(H_fw,ftp_server,ftp_to_target(public_host),default)

Fig. 6 — Permissions in organization H_fw


