Héritage de privilèges dans le modèle Or-BAC : application dans un environnement réseau

Frédéric Cuppens[1], Nora Cuppens-Boulahia[1], and Alexandre Miège[1,2]

[1] GET/ENST Bretagne/Département RSM,
BP 78, 2 rue de la Châtaigneraie, 35512 Cesson Sévigné Cedex, France
[2] GET/ENST/Département INFRES,
46, rue Barrault, 75634 Paris Cedex 13, France
{frederic.cuppens,nora.cuppens,alexandre.miege}@enst-bretagne.fr

Résumé Le concept de hiérarchie de rôles fut introduit pour la première fois dans le modèle RBAC (*Role Based Access Control*). L'héritage des permissions est associée à cette hiérarchie ce qui permet de spécifier des politiques de sécurité de façon modulaire. Cet article propose de généraliser cette approche dans le contexte du modèle Or-BAC (*Organization Based Access Control*). Nous définissons d'abord des hiérarchies de rôles, de vues et d'activités et formalisons un mécanisme d'héritage pour chaque hiérarchie. Nous définissons ensuite une hiérarchie d'organisations. Nous montrons que ce concept fournit un moyen efficace pour dériver, de la spécification d'une politique de sécurité organisationnelle, les politiques de sécurité technique de composants de sécurité. Nous illustrons notre approche dans le cadre des politiques de sécurité réseau, en particulier pour configurer des pare-feux (firewall).

1 Introduction

Dans le domaine de la programmation orientée objet, l'héritage est un moyen efficace pour concevoir une application de façon modulaire. Un concept similaire est utilisé dans le modèle RBAC [14] lorsqu'une hiérarchie de rôles est définie et associée à un mécanisme d'héritage des permissions. Cette hiérarchie de rôles est utile pour structurer la spécification d'une politique de sécurité.

Cependant, le concept de hiérarchie de rôles n'est pas exempt d'ambiguïtés [11,12,5]. Certaines de ces ambiguïtés sont directement liées au concept de rôle lui-même. Si l'on examine les exemples proposés dans [14], plusieurs interprétations du concept de rôles sont possibles. A la base, un rôle permet à un sujet qui est affecté à ce rôle d'effectuer certaines activités. C'est le cas de rôles tels que *médecin*, *infirmier* ou *secrétaire médicale*. Cependant, dans certains exemples, un rôle est indissociable d'une *organisation* particulière [13]. Par exemple, on peut considérer des rôles tels que *infirmier dans un département de cardiologie* ou *infirmier dans une équipe de réanimation*. Cette distinction est importante car les permissions affectées à un rôle peuvent dépendre de l'organisation. Ainsi, un infirmier peut avoir des permissions différentes s'il effectue son activité dans un département de cardiologie ou dans une équipe de réanimation. Un rôle peut également être associé à l'activité d'administration de l'organisation. *Directeur d'hôpital* ou *chef du département de cardiologie* sont des exemples de tels rôles.

Comme nous le verrons dans la suite, ces différentes interprétations du concept de rôle ne se comportent pas de la même façon vis-à-vis de l'héritage des permissions. C'est la raison pour laquelle il est important de disposer d'un modèle qui permet de rendre explicite ces différences d'interprétation.

Nous proposons d'analyser ce problème dans le contexte du modèle Or-BAC [10]. Dans Or-BAC, une organisation correspond à n'importe quelle entité qui a la responsabilité de gérer un ensemble de règles de sécurité (permissions ou interdictions). Par exemple, un hôpital particulier est une organisation. Un composant de sécurité, tel qu'un firewall, peut également être assimilé à une organisation dans la mesure où il gère un ensemble de règles de sécurité.

La définition d'un rôle dans le modèle Or-BAC est toujours associée à une organisation particulière. Ceci permet d'éviter certaines ambiguïtés lorsqu'on définit des hiérarchies de rôles et qu'on leur associe un mécanisme d'héritage de permissions.

Le modèle Or-BAC introduit deux autres concepts : les *activités* et les *vues*. La politique de sécurité associée à une organisation particulière est définie par des permissions (ou interdictions) pour les rôles de réaliser des activités sur des vues. Dans la suite, nous proposons de définir des hiérarchies d'activités et de vues et modélisons les mécanismes d'héritage associés à ces hiérarchies.

Enfin, dans Or-BAC, on peut également définir des hiérarchies d'*organisations*. Il s'agit peut-être de la contribution principale de cet article car ces hiérarchies fournissent un moyen très efficace pour structurer la spécification de la politique de sécurité. On peut ainsi commencer par la spécification de la politique de sécurité d'une organisation de "haut niveau" telle qu'un hôpital. On peut ensuite décomposer cette organisation en faisant apparaître les différentes sous-organisations de cet hôpital et de proche en proche, arriver à la spécification de la politique de sécurité technique des composants de sécurité tels qu'un firewall.

Dans cet article, nous proposons d'analyser et de formaliser l'héritage des permissions et des interdictions dans ces différentes hiérarchies. Le reste de l'article est organisé de la façon suivante. La section 2 rappelle les principaux concepts de Or-BAC. La section 3 présente les différentes hiérarchies associées respectivement aux rôles, activités et vues. La section 4 étudie les hiérarchies d'organisations. Dans la section 5, nous récapitulons comment spécifier une politique de sécurité dans Or-BAC lorsque les hiérarchies sont utilisées. La section 6 montre comment notre approche s'applique à la spécification de politiques de sécurité réseau. Enfin, la section 7 conclut et suggère plusieurs extensions à ce travail.

2 Or-BAC

2.1 Concepts de base de Or-BAC

Or-BAC [10] est un modèle de contrôle d'accès fondé sur le concept d'organisation. Dans Or-BAC, plusieurs organisations peuvent simultanément spécifier leur propre politique de contrôle d'accès en utilisant huit entités de base (voir figure 1) : Org(organization), *Role*, *Activity*, $View$, $Subject$, *Action*, $Object$ et

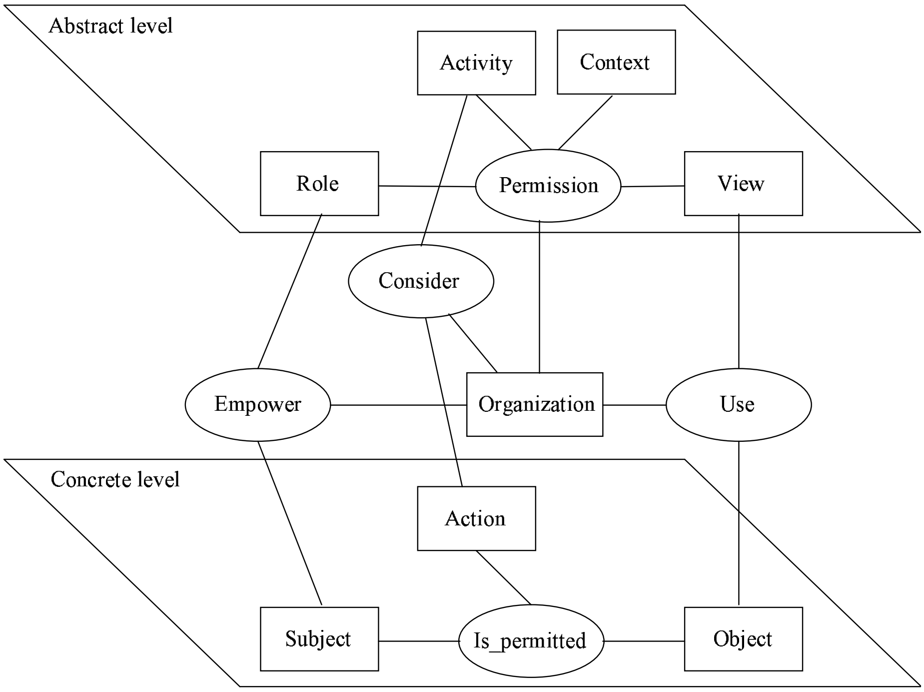

Fig. 1 –. *Structure du modèle Or-BAC*

Context. Les prédicats utilisés dans Or-BAC pour modéliser les associations entre ces huit entités sont récapitulés dans la table 2.

Comme nous l'avons mentionné dans l'introduction, une organisation est n'importe quelle entité qui gère un ensemble de règles de sécurité. Un sujet est une entité à qui l'on peut affecter un rôle. Nous supposons que $Org \subseteq Subject$ de sorte qu'il est possible d'affecter un rôle à une organisation. Par exemple, les rôles "département de cardiologie" ou "service des urgences" peuvent être affectés à certaines organisations.

En utilisant l'entité *Role*, nous disposons d'un moyen efficace pour structurer les sujets et mettre à jour les politiques de sécurité lorsque de nouveaux sujets sont insérés dans le système. Une démarche similaire nous conduit à introduire une nouvelle entité pour structurer les objets : l'entité $View$. Intuitivement, une vue correspond à un ensemble d'objets qui satisfont une même propriété[3].

Une autre entité est utilisée pour structurer les actions : l'entité *Activity*. En considérant que les rôles regroupent les sujets qui remplisssent les mêmes fonctions et les vues correspondent à des ensembles d'objets qui satisfont une même propriété, les activités associent les actions qui partagent les mêmes principes.

Les sujets, les objets et les actions peuvent avoir des attributs. Ces attributs sont modélisés par un ensemble de prédicats binaires ayant la forme $att(ent,val)$

3. Il existe clairement une analogie entre cette notion de vue et celle utilisée dans les bases de données relationnelles

Nom du Prédicat	Domaine	Description
Relevant_role	$Org \times Role$	Si org est une organisation et r est un rôle, alors $Relevant\_role(org,r)$ signifie que jouer le rôle r a un sens dans l'organisation org. Ex : $Relevant\_role(H,physician)$
Relevant_activity	$Org \times Activity$	Si org est une organisation et a est une activité, alors $Relevant\_activity(org,a)$ signifie que réaliser l'activité a a un sens dans l'organisation org. Ex : $Relevant\_activity(H,consult)$
Relevant_view	$Org \times View$	Si org est une organisation et v est une vue, alors $Relevant\_view(org,v)$ signifie que l'utilisation de la vue v a un sens dans l'organisation org. Ex : $Relevant\_view(H,medical\_record)$
Empower	$Org \times Subject \times Role$	Si org est une organisation, s est un sujet et r est un rôle, alors $Empower(org,s,r)$ signifie que org habilite le sujet s dans le rôle r. Ex : $Empower(H,John,physician)$
Consider	$Org \times Action \times Activity$	Si org est une organisation, α est une action et a est une activité, alors $Consider(org,\alpha,a)$ signifie que org considère que l'action α implante l'activité a. Ex : $Consider(H,"SELECT",consult)$
Use	$Org \times Object \times View$	Si org est une organisation, o est un objet et v est une vue, alors $Use(org,o,v)$ signifie que org utilise l'objet o dans la vue v. Ex : $Use(H,med\_27,medical\_record)$
Hold	$Org \times Subject \times Action \times Object \times Context$	Si org est une organisation, s est un sujet, α une action, o un objet et c un contexte, alors $Hold(org,s,\alpha,o,c)$ signifie que dans l'organisation org, le contexte c est satisfait entre le sujet s, l'action α et l'objet o. Ex : $\forall s,\forall \alpha,\forall o,$ $Hold(H,s,\alpha,o,working\_hours)$ $\leftarrow (08:00 \leq time(GLOBAL\_CLOCK) \wedge$ $time(GLOBAL\_CLOCK) \leq 19:00)$

Fig. 2 –. *Prédicats de base de Or-BAC*

où ent est un sujet, un objet ou une action et val représente la valeur de l'attribut att de l'entité ent. Par exemple, si med_27 est un dossier médical, alors $name(med\_27{,}John)$ indique que med_27 est le dossier médical de John.

Nous supposons que $Subject \subseteq Object$ de sorte que l'on peut définir des vues sur les sujets que nous appelons *groupes*. Dans Or-BAC, il y a une distinction claire entre un rôle et un groupe. Des permissions peuvent être affectées à des rôles alors qu'un groupe est seulement un ensemble de sujets qui ont des propriétés communes[4]. Cependant, il est parfois commode d'affecter le même rôle à chacun des sujets appartenant à un certain groupe. Dans ce cas, nous proposons d'utiliser le prédicat $G\_Empower(org{,}group{,}role)$ et de spécifier la règle suivante :

– GE : $\forall org{,}\forall group{,}\forall role{,}\forall subject{,}$
$Use(org{,}subject{,}group) \wedge$
$G\_Empower(org{,}group{,}role)$
$\rightarrow Empower(org{,}subject{,}role)$

Le modèle Or-BAC permet à l'administrateur de sécurité de spécifier que certaines permissions et interdictions ne s'appliquent que dans certains *contextes*. Pour cela, nous introduisons l'entité $Context$. Les contextes sont définis par des règles logiques qui concluent sur le prédicat $Hold$ (voir la table 2 qui donne l'exemple de contexte *working_hours*). Nous disons que, dans l'organisation org, le sujet s réalise l'action α sur l'objet o dans le contexte c quand la condition *cond* est satisfaite, c'est-à-dire lorsqu'il existe une règle[5] ayant la forme suivante : $Hold(org{,}s{,}\alpha{,}o{,}c) \leftarrow cond$. Pour une présentation plus détaillée de la notion de contexte dans Or-BAC, voir [7].

2.2 Permissions et interdictions

Les permissions et les interdictions dans Or-BAC sont définies à l'aide des prédicats présentés dans la figure 3. Une politique de contrôle d'accès est modélisée à deux niveaux différents : un niveau abstrait qui spécifie les permissions et les interdictions entre les rôles, les activités et les vues, et un niveau concret qui permet de dériver les permissions et les interdictions entre les sujets, les actions et les objets (voir figure 1).

Ces deux niveaux sont reliés de la façon suivante. Dans une organisation org, un sujet s a la permission d'effectuer une action α sur un objet o si (1) s est habilité dans un certain rôle r dans org et (2) α implante une certaine activité a dans org et (3) o est utilisé dans une certaine vue v dans org. Si ces trois conditions sont satisfaites et si (4) l'organisation org accorde au rôle r la permission de réaliser l'activité a sur la vue v dans le contexte c et si (5) dans

4. Certains modèles proposent d'affecter des permissions à des groupes de sujets. Cette démarche, qui contribue grandement à la confusion entre les notions de rôle et de groupe, est incorrecte et doit être évitée.
5. On peut supposer que chaque contexte c est défini par une règle *unique* en utilisant le fait que $Hold(org{,}s{,}\alpha{,}o{,}c) \leftarrow cond_1$, ..., $Hold(org{,}s{,}\alpha{,}o{,}c) \leftarrow cond_n$ est équivalent à $Hold(org{,}s{,}\alpha{,}o{,}c) \leftarrow (cond_1 \vee ... \vee cond_n)$.

l'organisation org, le contexte c est satisfait entre le sujet s, l'action α et l'objet o, alors une requête par le sujet s de réaliser l'action α portant sur l'objet o est acceptée. La dérivation des permissions concrètes à partir des permissions abstraites est donc modélisée par la règle suivante :

- $\mathrm{RG}_1 : \forall org, \forall r, \forall a, \forall v, \forall s, \forall \alpha, \forall o,$
  $Permission(org,r,a,v,c) \wedge$
  $Empower(org,s,r) \wedge$
  $Consider(org,\alpha,a) \wedge$
  $Use(org,o,v) \wedge$
  $Hold(org,s,o,\alpha,c)$
  $\rightarrow Is\_permitted(s,\alpha,o)$

Une autre règle similaire (appelée RG_2) est utilisée pour dériver les *interdictions* concrètes à partir des interdictions abstraites.

Nom de Prédicat	Domaine	Description
Permission	$Org \times Role \times Activity \times View \times Context$	Si org est une organisation, r un rôle, a une activité, v une vue et c un contexte, alors $Permission(org,r,a,v,c)$ signifie que l'organisation org accorde au rôle r la permission de réaliser l'activité a sur la vue v dans le contexte c. Ex : $Permission(H,physician, consult,medical\_record,working\_hours)$
Prohibition	$Org \times Role \times Activity \times View \times Context$	Si org est une organisation, r un rôle, a une activité, v une vue et c un contexte, alors $Prohibition(org,r,a,v,c)$ signifie que l'organisation org interdit au rôle r de réaliser l'activité a sur la vue v dans le contexte c. Ex : $Prohibition(H,nurse, consult,medical\_record,night)$
Is_permitted	$Subject \times Action \times Object$	Si s est un sujet, α une action et o un objet, alors $Is\_permitted(s,\alpha,o)$ signifie que s a la permission concrète de réaliser l'action α sur l'objet o. Ex : $Is\_permitted(John,"SELECT",med\_27)$
Is_prohibited	$Subject \times Action \times Object$	Si s est un sujet, α une action et o un objet, alors $Is\_prohibited(s,\alpha,o)$ signifie que s a concrètement l'interdiction de réaliser l'action α sur l'objet o. Ex : $Is\_prohibited(Mary,"DELETE",med\_27)$

Fig. 3 –. *Spécification des permissions et des interdictions dans Or-BAC*

2.3 Contraintes

Exprimer des contraintes qui s'appliquent à une politique de contrôle d'accès fut proposé pour la première fois dans le modèle RBAC (plus précisément, dans le sous-modèle RBAC_2 [8]) et ensuite davantage analysé dans [1]. Pour spécifier des contraintes dans le modèle Or-BAC, nous introduisons un prédicat $error()$. Une contrainte est modélisée par une règle ayant $error()$ pour conclusion (comme le suggère [3,9]).

Par exemple, nous pouvons spécifier que, dans un hôpital H, un sujet ne peut pas être habilité à la fois dans les rôles *anesthésiste* et *chirurgien* :

– C_1 : $\forall s$,
$Empower(H,s,anesthetist) \wedge Empower(H,s,surgeon)$
$\rightarrow error()$

Dans la suite, nous considérons la contrainte suivante qui s'applique à toutes les organisations :

– C_2 : $\forall org,\forall s,\forall r$,
$Empower(org,s,r) \wedge \neg Relevant\_role(org,r)$
$\rightarrow error()$

La règle C_2 indique qu'une organisation org ne peut pas habiliter un sujet s dans un rôle r si le rôle r n'est pas défini dans l'organisation org.

Il y a d'autres règles semblables à C_2 mais concernant les activités (règle C_3), les vues (règle C_4), les permissions (règle C_5) et les interdictions (règle C_6).

3 Les hiérarchies dans une organisation

Dans le modèle Or-BAC, il est possible de définir des hiérarchies de rôles (comme suggéré dans [8]) mais aussi des hiérarchies de vues et d'activités. Chaque hiérarchie définit respectivement une relation d'ordre partiel sur l'ensemble des rôles, des vues et des activités. Afin de modéliser ces différentes hiérarchies, nous présentons dans cette section les règles générales d'héritage des permissions et des interdictions qui leur sont associées.

3.1 La hiérarchie de rôles

Nous nous attachons dans un premier temps à étudier la hiérarchie de rôles. Pour cela nous introduisons le prédicat $sub\_role(org,r_1,r_2)$ qui signifie : dans l'organisation org, le rôle r_1 est un sous-rôle du rôle r_2.

Remarquons que la hiérarchie de rôles dépend de l'organisation. Ainsi, chaque organisation peut définir sa propre hiérarchie de rôles. L'héritage des permissions à travers la hiérarchie de rôles est modélisé par la règle suivante :

– RH_1 : $\forall org,\forall r_1,\forall r_2,\forall a,\forall v,\forall c$,
$sub\_role(org,r_1,r_2) \wedge$
$Permission(org,r_2,a,v,c)$
$\rightarrow Permission(org,r_1,a,v,c)$

Cette règle signifie que si le rôle r_1 est un sous-rôle du rôle r_2 dans l'organisation org, alors toutes les permissions attribuées au rôle r_2 dans org sont également attribuées au rôle r_1.

L'héritage des interdictions est plus complexe. En effet, la relation qui lie les rôles et les sous-rôles dans une hiérarchie peut être interprétée de différentes façons. Nous considérons pour le moment deux types de relation :

- La relation de spécialisation/généralisation. Par exemple, le rôle *chirurgien* est une spécialisation du rôle *médecin*. Afin de modéliser ce type de relation, nous introduisons le prédicat $specialized\_role(org,r_1,r_2)$ qui signifie : dans l'organisation org, le rôle r_1 est une spécialisation du rôle r_2.
- La relation de hiérarchie organisationnelle. Par exemple, le rôle *directeur de département* peut être défini comme hiérarchiquement supérieur au rôle *chef d'équipe*. Ce second type de relation est modélisé par le prédicat $senior\_role(org,r_1,r_2)$ qui signifie : dans l'organisation org, le rôle r_1 est un rôle senior du rôle r_2, c'est-à-dire un rôle supérieur dans la hiérarchie organisationnelle.

Nous considérons que la relation *specialized_role* est incluse dans la relation *sub_role* :

- RH_2 : $\forall org, \forall r_1, \forall r_2,$
  $specialized\_role(org,r_1,r_2)$
  $\rightarrow sub\_role(org,r_1,r_2)$

Par conséquent, la règle RH_1 s'applique à la hiérarchie de spécialisation des rôles. Les permissions sont alors héritées à travers cette hiérarchie. Les interdictions sont également héritées à travers la hiérarchie de spécialisation des rôles conformément à la règle d'héritage suivante :

- RH_3 : $\forall org, \forall r_1, \forall r_2, \forall a, \forall v, \forall c,$
  $specialized\_role(org,r_1,r_2) \wedge$
  $Prohibition(org,r_2,a,v,c)$
  $\rightarrow Prohibition(org,r_1,a,v,c)$

Par exemple, le rôle *chirurgien* hérite de toutes les interdictions du rôle *médecin*. Cet héritage est tout à fait justifié dans la mesure où le rôle *chirurgien* est un cas particulier du rôle *médecin*.

En revanche, la relation *senior_role* n'est généralement pas incluse dans la relation *sub_role*. Par conséquent, il est possible d'avoir :

- $\exists org, \exists r_1, \exists r_2,$
  $senior\_role(org,r_1,r_2) \wedge \neg sub\_role(org,r_1,r_2)$

Considérons par exemple le rôle *directeur d'hôpital*, hiérarchiquement supérieur au rôle *médecin*. Dans certains hôpitaux, le rôle *directeur d'hôpital* correspond uniquement à une fonction administrative qui n'est pas réalisée par un médecin. Dans ce cas, il n'y a aucune raison de conclure que le rôle *directeur d'hôpital* est un sous-rôle du rôle *médecin*.

Supposons à présent que le rôle r_1 est un rôle senior du rôle r_2 et que r_1 est aussi un sous-rôle de r_2. Supposons également que le rôle r_1 a plus de pouvoir que le rôle r_2. Alors, la règle RH_1 est tout à fait justifiée car elle permet à r_1 d'hériter de toutes les permissions de r_2. En revanche, si r_1 hérite aussi des interdictions de r_2, nous sommes en contradiction avec le fait que le rôle r_1 a plus de pouvoir que r_2. C'est pourquoi nous considérons que dans le cas d'une hiérarchie organisationnelle, l'héritage des interdictions se fait dans le sens inverse de l'héritage des permissions :

– $\mathrm{RH}_4 : \forall org, \forall r_1, \forall r_2, \forall a, \forall v, \forall c,$
  $sub\_role(org,r_1,r_2) \wedge senior\_role(org,r_1,r_2) \wedge$
  $Prohibition(org,r_1,a,v,c) \wedge$
  $\rightarrow Prohibition(org,r_2,a,v,c)$

Considérons par exemple que le rôle *directeur de département* est un rôle senior et également un sous-rôle du rôle *chef d'équipe*. Alors, le rôle *directeur de département* hérite des permissions accordées au rôle *chef d'équipe* (règle RH_1) et ce dernier hérite des interdictions attribuées au rôle *directeur de département* (règle RH_4).

Ainsi, nous avons défini trois hiérarchies de rôle : *sub_role*, *specialized_role* (inclus dans *sub_role*) et *senior_role* (généralement non inclus dans *sub_role*). Si nous nous intéressons uniquement à l'héritage des permissions et des interdictions, nous pouvons nous limiter aux deux hiérarchies suivantes : *sub_role* et *specialized_role*. Les règles RH_1, RH_2 et RH_3 s'appliquent conformément à la hiérarchie *specialized_role*, et les règles RH_1 et RH_4 s'appliquent selon la hiérarchie constituée par la différence entre la relation *sub_role* et la relation *specialized_role*. Pour cela, nous remplaçons simplement dans la prémisse de la règle RH_4 la condition : $sub\_role(org,r_1,r_2) \wedge senior\_role(org,r_1,r_2)$ par la condition : $sub\_role(org,r_1,r_2) \wedge \neg specialized\_role(org,r_1,r_2)$.

Ajoutons tout de même que toutes les règles d'héritage présentées dans cette section peuvent être accompagnées d'exceptions. Par exemple, nous pouvons considérer que dans l'hôpital H, les médecins ont l'interdiction de consulter les dossiers médicaux des patients qui ne sont pas les leurs. En appliquant la règle RH_3, nous obtenons que les chirurgiens héritent de cette même interdiction. Néanmoins, il est possible de spécifier explicitement, et ainsi définir une exception, que dans l'hôpital H, les chirurgiens ont la permission de consulter tous les dossiers médicaux, quel que soit le patient. La gestion des exceptions est traitée à la section 5.2.

3.2 La hiérarchie d'activités

Nous définissons dans cette section l'héritage entre les activités. Dans chaque organisation, les activités sont structurées sous forme de hiérarchies. La modélisation de ce type de relation hiérarchique est faite au moyen du prédicat $sub\_activity(org,a_1,a_2)$ qui signifie : dans l'organisation org, l'activité a_1 est une sous-activité de a_2.

La sémantique attribuée à cette hiérarchisation est la spécialisation. Ainsi, dans l'organisation hôpital H, l'activité *gestion* (des dossiers médicaux par exemple) est spécialisée en trois activités *création*, *consultation* et *mise-à-jour*. La structure hiérarchique associée à l'activité *gestion* est exprimée au moyen du prédicat *sub_activity* défini ci-dessus : *sub_activity(H,création,gestion)*, *sub_activity(H,consultation,gestion)* et *sub_activity(H,mise-à-jour,gestion)*.

A cette hiérarchie est associé un héritage des permissions, modélisé par la règle suivante :

– AH_1 : $\forall org, \forall r, \forall a_1, \forall a_2, \forall v, \forall c,$
  $Permission(org,r,a_2,v,c) \wedge$
  $sub\_activity(org,a_1,a_2)$
  $\rightarrow Permission(org,r,a_1,v,c)$

Supposons par exemple que, dans l'hôpital H, les médecins ont la permission de gérer les dossiers médicaux de leurs patients. En appliquant la règle AH_1 nous pouvons conclure que les médecins ont également la permission de créer, de consulter et de mettre à jour les dossiers médicaux de leurs patients.

De plus, nous considérons qu'une règle similaire, appelée AH_2, s'applique à l'héritage des interdictions. Si dans l'hôpital H les infirmiers ont l'interdiction de gérer les dossiers médicaux, alors, par application de la règle AH_2, nous obtenons que les infirmiers ont également l'interdiction de créer, de consulter et de mettre à jour ces dossiers médicaux.

3.3 La hiérarchie de vues

Comme pour les rôles et les activités, l'ensemble des vues est structuré par des hiérarchies dépendant de l'organisation. Cette hiérarchisation est modélisée par le prédicat $sub\_view(org,v_1,v_2)$ qui signifie : dans l'organisation org, la vue v_1 est une sous-vue de la vue v_2.

La sémantique que nous attribuons à cette relation hiérarchique est la spécialisation. Cette structuration est en fait proche de la hiérarchie d'héritage de classes utilisée dans les approches orientées objet (la relation Isa).

Dans le contexte du modèle Or-BAC, on associe l'héritage des permissions aux hiérarchies de vues. La règle suivante modélise cet héritage :

– VH_1 : $\forall org, \forall r, \forall a, \forall v_1, \forall v_2, \forall c,$
  $Permission(org,r,a,v_2,c) \wedge$
  $sub\_view(org,v_1,v_2)$
  $\rightarrow Permission(org,r,a,v_1,c)$

Une règle similaire, appelée VH_2, définit l'héritage des interdictions. Supposons par exemple que la vue *dossier chirurgical* est une sous-vue de la vue *dossier médical*. Dans ce cas, si un rôle a la permission ou l'interdiction de réaliser une certaine activité sur la vue *dossier médical*, alors ce rôle aura la permission ou l'interdiction de réaliser cette activité sur la vue *dossier chirurgical*.

Nous pouvons également définir le concept de vue *dérivée* comme un cas particulier de spécialisation de vue. Ainsi, nous considérons qu'une vue v_1 est dérivée d'une vue v_2 s'il existe une règle ayant la forme suivante :

– $\forall org, \forall obj, \forall v_1, \forall v_2,$
  $(Use(org,obj,v_2) \wedge Condition) \rightarrow Use(org,obj,v_1)$
  où $Condition$ est une condition logique utilisée pour spécialiser la vue v_2 en la vue v_1.

4 La hiérarchie d'organisations

Dans la section précédente, nous avons présenté comment définir dans une organisation des hiérarchies de rôles, d'activités et de vues. Nous étudions dans cette section comment construire des hiérarchies d'organisations. Nous introduisons le prédicat $sub\_organization(org_1,org_2)$ qui signifie : l'organisation org_1 est une sous-organisation de l'organisation org_2. Nous supposons que ce prédicat définit une relation d'ordre partiel sur l'ensemble des organisations.

Ainsi, par exemple, si l'organisation H est un hôpital et $dept8$ son département des urgences, alors nous avons : $sub\_organization(dept8,H)$

Il est possible de spécifier que toutes les sous-organisations org_1 d'une certaine organisation org_2 sont affectées à un rôle donné. Cette spécification est modélisée par la contrainte suivante :

– C_7 : $\forall org_1,\forall org_2,\forall r,$
  $sub\_organization(org_1,org_2) \wedge \neg Empower(org_2,org_1,r)$
  $\rightarrow error()$

Dans l'exemple précédent, la contrainte C_7 est satisfaite si l'on a *Empower*-(H,*dept*8,*casualty_dept*).

Remarquons qu'un rôle défini dans une certaine organisation n'est pas nécessairement défini dans toutes ses sous-organisations. Soit, par exemple, *dept*7 le département administratif de l'hôpital H et *infirmier* un rôle donné. Le rôle *infirmier* peut ne pas être défini dans le département *dept*7 (dans ce cas nous n'avons pas le fait $Relevant\_role(dept7,infirmier)$) alors que ce rôle est défini dans H ($Relevant\_role(H,infirmier)$).

Inversement, si org_1 est une sous-organisation de org_2, alors certains rôles peuvent être définis dans org_1 sans l'être dans org_2.

Les mêmes remarques sont valables dans le cas des vues et des activités. Ainsi, si org_1 est une sous-organisation de org_2, alors l'ensemble des vues, respectivement des activités, définies dans org_1 peut être disjoint de l'ensemble des vues, respectivement des activités, définies dans org_2.

4.1 L'héritage des hiérarchies

Supposons que org_1 est une sous-organisation de org_2. Le sous-ensemble des rôles définis dans org_2 qui sont également définis dans org_1, se voit appliquer dans $org1$ la même structure hiérarchique que celle définie dans org_2. La modélisation de cet état de faits est donnée par la règle suivante :

– HH_1 : $\forall org_1,\forall org_2,\forall r_1,\forall r_2,$
  $sub\_organization(org_2,org_1)$
  $sub\_role(org_1,r_1,r_2)\wedge$
  $Relevant\_role(org_2,r_1) \wedge Relevant\_role(org_2,r_2)$
  $\rightarrow sub\_role(org_2,r_1,r_2)$

Le même principe s'applique à l'héritage des privilèges dans la hiérarchie de spécialisation des rôles et également à l'héritage dans les hiérarchies de spécialisation des activités et des vues. Nous obtenons ainsi trois nouvelles règles, HH_2,

HH_3 et HH_4, en remplaçant le prédicat *sub_role* dans la règle HH_1 respectivement par les prédicats *specialized_role*, *sub_activity* et *sub_view*.

4.2 L'héritage des permissions et des interdictions

Nous appliquons les mêmes principes d'héritage des permissions et des interdictions à travers la hiérarchie d'organisations lorsque les rôles, les activités et les vues concernés dans les permissions et les interdictions sont pertinents pour la sous-organisation. Ce fait est modélisé par la règle suivante :

– OH_1 : $\forall org_1, \forall org_2, \forall r, \forall a, \forall v, \forall c,$
  $sub\_organization(org_2, org_1)$
  $Permission(org_1, r, a, v, c) \wedge$
  $Relevant\_role(org_2, r) \wedge$
  $Relevant\_activity(org_2, a) \wedge$
  $Relevant\_view(org_2, v)$
  $\rightarrow Permission(org_2, r, a, v, c)$

Une règle similaire, appelée OH_2, s'applique dans le cas de l'héritage des interdictions.

5 Spécification d'une politique de sécurité dans Or-BAC

5.1 Formalisation d'une politique de sécurité

En résumé, nous pouvons modéliser une politique de sécurité comportant des hiérarchies comme une théorie logique, conformément à la définition ci-après.
Définition 1 : Une politique de sécurité *pol* est modélisée dans Or-BAC comme une théorie logique T_{pol} définie de la façon suivante :

– Des ensembles de faits utilisant les prédicats *Relevant_role*, *Relevant_view* et *Relevant_activity*
– Des ensembles de faits utilisant les prédicats *Empower*, *Use*, *Consider*, *Permission* et *Prohibition*
– La règle *GE* et des faits utilisant le prédicat *G_Empower*
– Un ensemble de règles pour la définition des vues dérivées (section 3.3)
– Un ensemble de faits utilisant des prédicats binaires pour décrire les valeurs des attributs des sujets, des actions et des objets.
– Un ensemble des règles de définition de contextes, *i.e.* des règles dont la conclusion est le prédicat $Hold(org, s, \alpha, o, c)$
– Des ensembles de faits (hiérarchies d'héritage) utilisant les prédicats *sub_role*, *specialized_role*, *sub_activity* et *sub_view*
– Les règles RG_1 et RG_2 pour dériver des permissions et des interdictions concrètes (section 2.1)
– Les règles RH_1 à RH_4 (règles d'héritage de rôles), AH_1 et AH_2 (règles d'héritage des activités) et VH_1 et VH_2 (règles d'héritages des vues)

- Les règles HH_1 à HH_4 (règles d'héritage des hiérarchies)
- Les règles OH_1 et OH_2 (règles d'héritage des organisations)
- Un ensemble de contraintes, *i.e.* des règles dont la conclusion est le prédicat $error()$.

Définition 2 : Une politique de sécurité *pol* viole une contrainte s'il est possible de dériver $error()$ à partir de T_{pol} : $T_{pol} \vdash error()$

5.2 Les Conflits

Le modèle Or-BAC permet de spécifier des permissions ainsi que des interdictions. Cette possibilité offerte par le modèle peut malheureusement conduire à la génération de conflits. Une situation de conflits correspond à une situation où un sujet a à la fois la permission et l'interdiction d'effectuer une action sur un objet donné.

Signalons que les exceptions dans les règles d'héritage peuvent générer des conflits. Ainsi, dans l'exemple présenté dans la section 3.1, un chirurgien peut avoir à la fois l'interdiction de consulter le dossier médical d'un malade qui ne fait pas partie de ses patients (une interdiction héritée du rôle *médecin*) et la permission de le faire (cas où il existe une permission explicite affectée au rôle *chirurgien*).

Nous avons étudié ce type de problème dans [6]. La solution que nous proposons pour gérer de tels conflits consiste à définir des priorités entre les permissions et les interdictions. Dans l'exemple ci-dessus, l'interdiction héritée de *médecin* peut avoir une priorité plus faible qu'une permission explicite attribuée à un *chirurgien*. Ainsi, le chirurgien peut consulter le dossier médical d'un malade même si celui-ci ne fait pas partie de ses patients (ce qui peut arriver par exemple dans les cas d'urgences ou d'échanges de diagnostics). Cependant, comme les aspects relatifs à la gestion des conflits dans le modèle Or-BAC sortent du cadre du sujet traité dans cet article, nous ne les développons pas davantage, de plus amples informations pouvant être trouvées dans [6].

6 Application

Afin d'illustrer l'intérêt de la hiérarchisation des différentes entités définies dans le modèle Or-BAC étendu et son applicabilité à des cas concrets qui ne se limitent pas au niveau organisationnel, nous présentons dans cette section un modèle d'un réseau local d'entreprise (RLE), de son architecture de sécurité et de sa connectivité à Internet. A cet effet, nous avons choisi de reprendre l'exemple utilisé dans Firmato [2] de façon à montrer comment les diverses entités et concepts utilisés dans l'architecture de sécurité de ce RLE sont naturellement exprimés avec le modèle Or-BAC. En particulier, nous montrons que le concept de hiérarchie du modèle Or-BAC étendu appliqué aux entités centrales que sont l'organisation, le rôle, l'activité et la vue permet d'éviter l'utilisation d'artifices tels que les groupes "open" et "closed" proposés dans [2].

6.1 Hiérarchie d'organisations

Nous souhaitons modéliser la politique de contrôle d'accès d'un réseau d'entreprise utilisé par une organisation H. H s'est dotée d'une configuration réseau basée sur deux pare-feux (firewalls) comme l'illustre la figure 4. Conformément à la présentation faite dans [2], le firewall externe est chargé de surveiller les connexions Internet de l'entreprise. Il est suivi par une DMZ (zone tampon). La DMZ comporte les serveurs de l'entreprise visibles de l'extérieur. Dans notre cas, ces serveurs fournissent les services HTTP/HTTPS (Web), FTP, SMTP (e-mail) et DNS. L'entreprise utilise deux hôtes pour permettre l'utilisation de ces services, un premier hôte dédié au DNS et l'autre (appelé multi_server) pour disposer de tous les autres services restants. Quant au firewall interne, il contrôle les connexions intranet de l'entreprise. Ce firewall possède en fait trois interfaces : une pour la DMZ, une autre pour la zone privée de l'entreprise et la troisième pour séparer les hôtes utilisés pour l'administration des firewalls. Enfin, dans la zone privée de l'entreprise, un hôte spécifique, *Admin_serv*, permet de gérer les serveurs de la DMZ.

Plusieurs organisations peuvent être introduites dans Or-BAC pour modéliser une telle architecture de sécurité réseau. D'abord, nous pouvons définir l'organisation H qui souhaite déployer une politique d'accès au RLE. Par souci de simplification, nous allons assimiler H à son réseau d'entreprise. H possède deux sous-organisations notées H_fw_1 et H_fw_2 correspondant respectivement au firewall externe et au firewall interne. Pour spécifier des politiques devant être mis en œuvre dans une zone particulière du RLE, d'autres organisations peuvent être définies. Ainsi, pour spécifier la politique en vigueur dans le réseau privé de H, l'organisation *H_private_net* peut être introduite. Toutes les entités Or-BAC relatives à cette organisation peuvent alors être spécifiées. Si nous considérons, par exemple, que H est un hôpital, pour modéliser la politique de sécurité de cette partie du RLE on peut introduire des rôles tels que *medecin*, $infirmier$, etc. Cependant, pour simplifier, nous ne raffinons pas davantage cette partie. Signalons au passage qu'une organisation *internet* peut être définie lorsque la politique de sécurité à mettre en œuvre par l'organisation Internet est explicitement spécifiée (ce qui n'est pas le cas pour le moment).

6.2 Les sujets

Dans cet exemple, les sujets correspondent aux hôtes identifiés par leurs adresses IP. Ainsi, si h est un hôte, le prédicat *address*(h,a) signifie que l'adresse IP de h est a. Les rôles sont affectés aux hôtes conformément à la stratégie décrite dans la section 6.3. Pour ce faire, le prédicat *Empower* permet d'attribuer un rôle à un hôte donné. Toutefois, il est parfois plus simple de former des groupes d'hôtes (appelés *zone* dans Firmato) et d'utiliser le prédicat *G_Empower* pour affecter le même rôle à chacun des hôtes appartenant à un groupe. Ainsi, on peut par exemple définir le groupe *Private_net* de la façon suivante :

- $\forall h, Use(H,h,Private\_net) \leftarrow$
$$Use(H,h,Host) \wedge address(h,a) \wedge a \in 111.222.2.* \wedge \neg Use(H,h,Firewall\_interface)$$

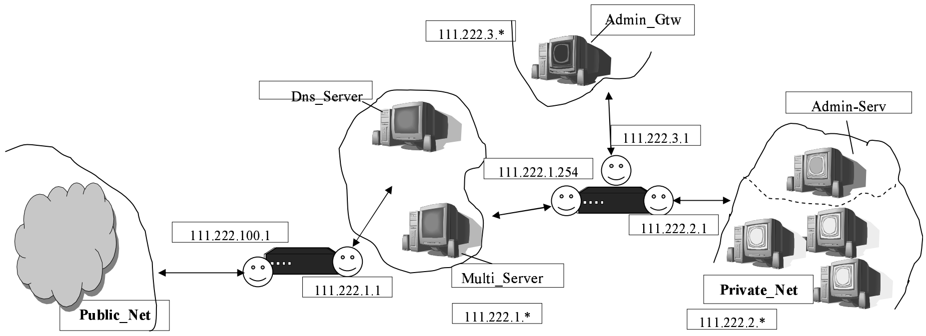

Fig. 4 –. *Exemple d'application*

Nom du rôle	Hôtes affectés au rôle	sous_rôle	Pertinence pour H_fw_1	Pertinence pour H_fw_2
$Public\_host$	Hôtes de la vue *Public_Net*	-	X	
$Private\_host$	Hôtes de la vue *Private_Net*	-		X
$Firewall$	Interfaces des firewalls	*Ext_firewall* *Int_firewall*		X
Ext_firewall	Interfaces du firewall externe	-	X	X
Int_firewall	Interfaces du firewall interne	-		X
DNS_server	Serveur DNS	-	X	X
Ftp_server	Serveur ftp	$Multi\_server$	X	X
$Mail\_server$	Serveur de mails	$Multi\_server$	X	X
Web_server	Serveur web	$Multi\_server$	X	X
$Multi\_server$	Multi-serveur	-	X	X
Adm_fw_host	Hôtes de la vue *Admin_gtw*	-	X	X
Adm_serv_host	Hôtes de la vue *Admin_serv*	-		X

Fig. 5 –. *Description d'un rôle*

6.3 Les rôles

Les rôles décrits dans la figure 5 sont affectés aux hôtes. Tous ces rôles sont pertinents (relevants) pour l'organisation H. La figure 5 spécifie les rôles pertinents respectivement aux organisations H_fw_1 et H_fw_2. Signalons que le rôle $Firewall$ est pertinent pour H_fw_2 (pour des besoins d'administration) mais ne l'est pas pour H_fw_1. La figure 5 décrit également pour chacun des rôles les sous-rôles correspondants. Dans l'exemple, la hiérarchie de sous-rôles correspond à celle d'une spécialisation des rôles.

6.4 Les activités

Les activités correspondent aux divers services offerts par le RLE de H. Nous définissons d'abord l'activité *all_tcp* à laquelle sont associées différentes sous-activités tcp telles que *smtp*, *ssh* et *https*. Nous définissons de façon similaire l'activité *all_icmp* et les différentes sous-activités sous-jacentes telles que *ping*.

Dans le cadre de cet exemple, deux autres activités sont définies. L'activité *admin_to_gtwy* possède deux sous-activités, *ssh* et *ping*. La deuxième activité *gtwy_to_admin* possèdent également deux sous-activités *ssh* et *https*. L'ensemble de toutes ces activités est pertinent pour les organisations H, H_fw_1 et H_fw_2

La principale différence entre l'approche que nous exposons ici et celle proposée dans [2] est l'utilisation des hiérarchies d'activités alors que Firmato a recours aux services élémentaires et aux groupes de services. Elle est également plus générique dans la mesure où les permissions et les interdictions s'appliquent à la seule entité activité.

6.5 Les vues

Les vues sont utilisées pour structurer les objets auxquels s'appliquent les différents services réseau. Ainsi, nous définissons la vue *target* ayant deux attributs : *content* correspondant aux messages transmis lors de l'utilisation du service et *dest* correspondant au hôte destinataire du service. Cet hôte est identifié par son rôle.

Dans l'exemple traité dans cette section, l'attribut *content* n'est pas utilisé parce que, pour les besoins d'illustration, nous ne considérons que des règles de filtrage sur l'hôte destinataire. Cependant, notre modèle peut aussi être facilement étendu pour spécifier des règles de filtrages sur les contenus des messages.

Nous pouvons ensuite introduire des sous-vues de la vue *target* conformément au rôle affecté à l'hôte destinataire. Nous pouvons, par exemple, définir des sous-vues *to_dns* de la façon suivante :

– $\forall o, Use(H,o,to\_dns) \leftarrow Use(H,o,target) \wedge dest(o,dns)$

Il en résulte la nécessité de définir autant de vues que de rôles. Pour assouplir ce processus, nous suggérons de définir une fonction *to_target* ayant pour domaine les rôles et pour co-domaine les vues. Les vues créées par cette fonction sont définies de la façon suivante :

– $\forall o, \forall r, Use(H,o,to\_target(r)) \leftarrow Use(H,o,target) \wedge dest(o,r)$

Nous considérons que la vue $to\_target(r)$ est pertinente pour l'une des deux organisations définies dans notre exemple si r est un rôle pertinent pour cette organisation en question. Nous considérons également que si le rôle r_1 est un sous-rôle du rôle r_2, alors la vue $to\_target(r_1)$ est une sous-vue de la vue $to\_target(r_2)$.

6.6 La politique de sécurité

Les éléments nécessaires à la spécification de la politique de sécurité ayant été introduits, nous pouvons donc définir les différentes permissions qui doivent être mises en application dans l'organisation H. Les objectifs de cette politique de sécurité sont les suivants :

1. Les hôtes de l'organisation interne peuvent accéder à l'Internet.
2. Les hôtes externes ne peuvent accéder qu'aux serveurs de la DMZ.

3. Les hôtes ayant le rôle *Admin_serv_host* peuvent mettre à jour les serveurs de la DMZ. Les autres hôtes de l'organisation ont les mêmes privilèges que les hôtes externes sur les serveurs de la DMZ.
4. Les interfaces des firewalls ne sont accessibles que par les hôtes jouant le rôle *Adm_fw_host*.

Ces permissions correspondent à la politique de sécurité présentée dans [2]. La figure 6 décrit comment ces permissions sont modélisées dans Or-BAC. Dans un souci de simplification, nous ne spécifions pas d'interdictions et nous supposons que toutes les permissions sont valables dans n'importe quel contexte (le contexte considéré est donc $default$ et est toujours évalué à *true*).

Un avantage significatif de notre approche par rapport à d'autres procédés comme celui de Firmato par exemple, est le fait qu'elle permette de dériver automatiquement des permissions qui s'appliquent respectivement à H_fw_1 et H_fw_2. En effet, il suffit d'appliquer la règle OH_1 pour dériver les permissions relatives aux sous-organisations de H. Les résultats que nous avons obtenus pour H_fw_1 sont décrits dans la figure 7. Pour illustrer le processus de dérivation, considérons la permission suivante :

– $Permission(H{,}adm\_fw\_host{,}admin\_to\_gtwy{,}to\_target(firewall){,}default)$

Etant donné que le rôle adm_fw_host, l'activité *admin_to_gtwy* et la vue $to\_target(firewall)$ sont pertinents pour la sous-organisation H_fw_2, nous pouvons appliquer OH_1 pour dériver :

– $Permission(H\_fw_2{,}adm\_fw\_host{,}admin\_to\_gtwy{,}$
$to\_target(firewall){,}default)$

Cependant, puisque la vue $to\_target(firewall)$ n'est pas pertinente pour H_fw_1, nous ne pouvons pas dériver une permission équivalente pour H_fw_1. Mais la vue $to\_target(ext\_firewall)$ est une sous-vue de *to_target(firewall)* et puisque $to\_target(ext\_firewall)$ est une vue pertinente pour H_fw_1, nous pouvons appliquer les règles VH_1 et OH_1 pour dériver :

– $Permission(H\_fw_1{,}adm\_fw\_host{,}admin\_to\_gtwy{,}$
$to\_target(ext\_firewall){,}default)$

Signalons que la permission :

– $Permission(H{,}private\_host{,}all\_tcp{,}to\_target(public\_host){,}default)$

n'est héritée par aucune des deux sous-organisations H_fw_1 et H_fw_2. C'est une conséquence du fait que le rôle *private_host* n'est pertinent que pour H_fw_2 alors que la vue $target(public\_host)$ est pertinente uniquement pour H_fw_1. Par conséquent, un seul des deux firewalls ne suffit pas pour gérer cette permission. Dans ce cas, nous proposons d'utiliser cette permission pour configurer les deux firewalls.

Comparativement à Firmato, l'approche que nous proposons est basée sur un ensemble plus réduit de concepts. En particulier, nous n'avons pas besoin d'introduire des notions comme celle des groupes "closed" (fermés). Un groupe

$Permission(H, adm\_fw\_host, admin\_to\_gtwy, to\_target(firewall), default)$
$Permission(H, firewall, gtwy\_to\_admin, to\_target(adm\_fw\_host), default)$
$Permission(H, private\_host, all\_tcp, to\_target(public\_host), default)$
$Permission(H, adm\_server\_host, all\_tcp, to\_target(dns\_server), default)$
$Permission(H, adm\_server\_host, all\_tcp, to\_target(multi\_server), default)$
$Permission(H, public\_host, smtp, to\_target(mail\_server), default)$
$Permission(H, public\_host, dns, to\_target(dns\_server), default)$
$Permission(H, public\_host, ftp, to\_target(ftp\_server), default)$
$Permission(H, public\_host, https, to\_target(web\_server), default)$
$Permission(H, private\_host, smtp, to\_target(mail\_server), default)$
$Permission(H, private\_host, dns, to\_target(dns\_server), default)$
$Permission(H, private\_host, ftp, to\_target(ftp\_server), default)$
$Permission(H, private\_host, https, to\_target(web\_server), default)$
$Permission(H, dns\_server, dns, to\_target(public\_host), default)$
$Permission(H, ftp\_server, ftp, to\_target(public\_host), default)$
$Permission(H, dns\_server, dns, to\_target(private\_host), default)$
$Permission(H, ftp\_server, ftp, to\_target(private\_host), default)$

Fig. 6 –. *Les permissions dans l'organisation H*

$Permission(H\_fw_1, adm\_fw\_host, admin\_to\_gtwy, to\_target(ext\_firewall), default)$
$Permission(H\_fw_1, ext\_firewall, gtwy\_to\_admin, to\_target(adm\_fw\_host), default)$
$Permission(H\_fw_1, public\_host, smtp, to\_target(mail\_server), default)$
$Permission(H\_fw_1, public\_host, dns, to\_target(dns\_server), default)$
$Permission(H\_fw_1, public\_host, ftp, to\_target(ftp\_server), default)$
$Permission(H\_fw_1, public\_host, https, to\_target(web\_server), default)$
$Permission(H\_fw_1, dns\_server, dns, to\_target(public\_host), default)$
$Permission(H\_fw_1, ftp\_server, ftp, to\_target(public\_host), default)$

Fig. 7 –. *Les permissions dans l'organisation H_fw_1*

fermé n'hérite pas des groupes de niveaux supérieurs de la hiérarchie. L'exemple pris dans Firmato est celui du groupe *firewall* qui ne doit pas hérité de *private_host*. Nous pensons que cette notion de groupe fermé nécessite une gestion quelque peu complexe et en fait inutile. Dans notre approche, il suffit simplement de spécifier que *private-net* ne comprend pas l'interface *firewall-interface* (se référer à la définition de *private-net* présentée dans la section 6.2). De plus, notre approche peut être utilisée pour traiter des applications plus complexes comprenant des interdictions, des exigences sur les contenus des messages ou encore des règles contextuelles.

7 Conclusion

Dans cet article, nous avons montré comment modéliser des hiérarchies d'héritage dans le modèle Or-BAC. Les travaux précédents considéraient seulement des hiérarchies de rôles (comme c'est la cas dans le modèle RBAC). Ici, nous avons défini des hiérarchies de rôles, d'activités, de vues et d'organisations et étudié l'héritage des permissions et des interdictions dans ces hiérarchies.

Concernant la hiérarchie de rôles, nous avons montré qu'il est important d'introduire deux types de hiérarchies : la hiérarchie de type spécialisation/généralisation et la hiérarchie de type junior/senior. Les permissions sont héritées "vers le bas" dans les deux hiérarchies (le rôle le plus spécialisé héritant du rôle le moins spécialisé et le rôle senior héritant du rôle junior). En revanche, nous considérons que les interdictions sont héritées "vers le bas" dans les hiérarchies de type spécialisation/généralisation (comme pour les permissions) alors qu'elles sont héritées "vers le haut" dans les hiérarchies de type junior/sénior (le rôle junior héritant les interdictions du rôle senior). Les propositions précédentes ne font pas ce type de distinction. Par exemple, [3] considère que les interdictions sont toujours héritées "vers le haut" dans la hiérarchie de rôles. Notre proposition permet d'éliminer certaines ambiguïtés présentes dans les approches précédentes.

Concernant la hiérarchie d'activités, nous avons seulement considéré la spécialisation/généralisation. On peut en fait considérer d'autres types de "décomposition" d'activités, par exemple décomposer une activité a en $b; c$ représentant l'activité b suivie par l'activité c. Il est clair que si un certain rôle r a la permission de réaliser l'activité a, alors r doit également avoir la permission de réaliser l'activité b. Cependant, il est plus complexe de définir les permissions concernant l'activité c ; le rôle r ne devrait avoir la permission de réaliser l'activité c qu'après avoir réalisé l'activité b. Dans Or-BAC, il semble qu'il soit possible de représenter cette contrainte en utilisant la notion de contexte. Modéliser ce type de décomposition dans Or-BAC est un problème intéressant qui a plusieurs applications, en particulier la spécification de politiques de sécurité pour les systèmes de "workflow" [4]. Nous comptons étudier ce problème dans le futur.

Nous avons également proposé une hiérarchie de vues de type spécialisation/généralisation. Nous envisageons d'étudier d'autres associations entre les vues, en particulier l'association d'agrégation (également appelée association "$Part\_of$"). On pourrait ainsi définir comment les permissions et les interdictions se propagent depuis une vue vers ses différentes sous parties. Il y a plusieurs applications à un tel modèle, par exemple la modélisation UML ou la protection de documents XML. Cependant, plusieurs problèmes surgissent, en particulier certaines activités peuvent être pertinentes pour une certaine vue mais ne pas s'appliquer à certaines de ses sous parties. Nous comptons également approfondir ce problème dans le futur.

Enfin, nous avons défini une hiérarchie d'organisations et modéliser l'héritage des permissions et des interdictions dans cette hiérarchie. Nous avons montré que cette hiérarchie est utile pour dériver, de la spécification de politiques de sécurité organisationnelle, les politiques de sécurité technique de composants particuliers tels qu'un firewall. Nous avons implanté un module de traduction qui produit les règles de sécurité pour configurer le firewall NetFilter. Nous envisageons d'appliquer une approche similaire pour produire les règles de configuration d'autres composants tels que des systèmes d'exploitation ou des systèmes de gestion de bases de données. Nous comptons également appliquer Or-BAC pour modéliser les exigences d'interopérabilité entre ces différentes politiques de sécurité.

Remerciements : Les travaux présentés dans cet article sont réalisés dans le cadre du projet RNRT exploratoire MP6 et de l'ACI DESIRS tout deux financés par le Ministère de la Recherche. La participation d'Alexandre Miège à ces travaux se fait dans le cadre d'une thèse financée par France Télécom R&D.

Références

1. G.-J. Ahn and R. Sandhu. Role-Based Authorization Constraints Specification. *ACM Transactions on Information and System Security*, 3(4), November 2000.
2. Y. Bartal, A. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit. In *20th IEEE Symposium on Security and Privacy*, pages 17–31, Oakland, California, May 1999.
3. E. Bertino, B. Catania, E. Ferrari, and P. Perlasca. A Logical Framework for Reasoning about Access Control Models. *ACM Transactions on Information and System Security*, 6(1), February 2003.
4. Elisa Bertino, Elena Ferrari, and Vijay Atluri. Specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security*, 2(1), February 1999.
5. J. Crampton. On permissions, inheritance and role hierarchies. In *10th ACM Conference on Computer and Communication Security*, Washington, November 2003.
6. F. Cuppens and A. Miège. Conflict management in the or-bac model. Technical report, ENST Bretagne, December 2003.
7. F. Cuppens and A. Miège. Modelling contexts in the Or-BAC model. In *19th Annual Computer Security Applications Conference*, Las Vegas, December 2003.
8. D. F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, and R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security*, 4(3):222–274, August 2001.
9. S. Jajodia, S. Samarati, and V. S. Subrahmanian. A logical Language for Expressing Authorizations. In *IEEE Symposium on Security and Privacy*, Oakland, CA, May 1997.
10. A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin. Organization Based Access Control. In *Proceedings of IEEE 4th International Workshop on Policies for Distributed Systems and Networks (POLICY 2003)*, Lake Come, Italy, June 2003.
11. J. D. Moffett. Control Principles and Role Hierarchies. In *3rd ACM Workshop on Role-Based Access Control*, October 1998.
12. J. D. Moffett and E. C. Lupu. The use of role hierarchies in access control. In *4th ACM Workshop on Role-Based Access Control*, October 1999.
13. Sejong Oh and Ravi Sandhu. A Model for Role Administration Using Organization Structure. In *Seventh ACM Symposium on Access Control Models and Technologies, (SACMAT'02)*, pages 155–162, Monterey, California, USA, june 3–4 2002.
14. R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.