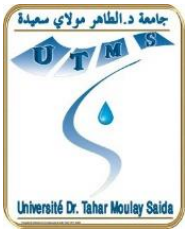
الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Dr. Tahar Moulay Saida جامعة د. الطاهر مولاي سعيدة
Faculté de la Technologie كلية التكنولوجيا
Département d'Informatique قسم الإعلام الآلي



Mémoire de Master

Option : sécurité informatique et cryptographie

Thème :

Modélisation et génération des attaques dans les politiques de sécurité

Présenté par :
KOUBI Mohamed
SAYAH Mohamed Kamel Eddine

Encadré par :
Mr ADJIR Noureddine

Septembre 2016

Remerciement

Nous remercions en premier notre grand Dieu pour nous avoir donné le courage et la volonté durant les moments difficiles.

Au terme de ce travail, je tiens à exprimer ma profonde gratitude et mes sincères remerciements aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire et durant toutes les années de notre étude.

Mes profonds remerciements vont à mon encadrant à ***Noureddine Adjir*** *qui a accepté d'encadrer notre travail, Nous exprimons nos gratitudes à tous les consultants et internautes rencontrés lors des efforts effectués et qui ont accepté de répondre à nos questions avec gentillesse.*

Nous tiens également à remercier les membres du Jury.

Mes plus vifs remerciements s'adressent aussi à tout le cadre professoral et administratif de Faculté de technologie d'université de

DR. MoulayTaher, Saida.

Et bien sûr nous gardons une place toute particulière à nos parents, nos frères, nos sœurs et mes oncles, nos amis qui sont toujours à nos côtés. Mes remerciements vont enfin à toute personne qui a contribué de près ou de loin à l'élaboration de ce travail.

Dédicace

Merci Allah (mon dieu) de m'avoir donné la capacité d'écrire et de réfléchir, la force d'y croire, la patience d'aller jusqu'au bout du rêve et le bonheur de lever mes mains vers le ciel et de dire « Ya Kayoum ».

Je dédie ce modeste travail à celle qui m'a donné la vie, le symbole de tendresse qui s'est sacrifiée pour mon bonheur et ma réussite, à ma mère, à mon père, à mes frères et Sœurs, à La Famille Koubi,SAYAH,ZERROUKI , à mes amis, à tous l'ensemble des étudiants du département d'informatique et à tous ceux qui m'aiment

ACRONYMES ET ABRÉVIATIONS

AC	***Access Control***
MAC	***Mandatary Access Control***
DAC	***Discretionary Access Control***
RBAC	***Role-Based Access Control***
ORBAC	**Organization-Based Access Control**
XACML	**Extensible Access Control Markup Language**
HRU	**Harrisson,ruzzo,ullman**
SOD	**Ségrégation of Duties (répartition des taches)**
ITSEC	**Information Technology Security Evaluation Criteria**
OASIS	**Organization for the Advancement of Structured Information Standards**
LMS	**Système de gestion de bibliothèque**

Table des matières

Table des figures

Liste des tableaux

Introduction générale

La sécurité est de plus en plus en importance et devient actuellement une question clé pour toute application ou système informatique pour les organisations, les entreprises et les gouvernements. Le marché se déplace dans l'Internet et le montant des actifs en cause est énorme.

Pour les entreprises à vendre leurs produits et services sur Internet, la sécurité est un problème majeur à résoudre. Pour les banques en ligne, garantir la sécurité est cruciale et une condition préalable avant même l'exécution du service.

Les entreprises ont consenti des efforts significatifs dans la mise en place de politiques et de procédures de sécurité. Et pourtant, force est de constater que des lacunes importantes laissent à désirer.

Les systèmes de stockage de données et les réseaux d'informations des entreprises doivent répondre à des exigences opposées, à savoir la protection de la vie privée et la divulgation des informations.

En d'autres termes, ce qui doit être protégé dans certaines situations et par rapport à certaines entités doit probablement être divulgué dans d'autres situations et par rapport à d'autres entités. Par exemple dans les applications médicales, les informations échangées contiennent généralement des données très sensibles. En effet, la fuite d'information ne devrait jamais être permise.

Par exemple, lors d'une urgence médicale, l'interdiction d'accès à l'information pourrait mettre des vies en danger.

Les lois exigent que les entreprises disposent de mécanismes fiables pour mettre en œuvre ces exigences dans leurs systèmes informatiques, dans leurs services Web, etc. Ainsi, les organismes gouvernementaux et les compagnies qui ne se conforment pas à ces lois risquent d'encourir de véritables conséquences ; par exemple entacher leur réputation et/ou devoir se soumettre à des obligations légales. Dans ces systèmes, l'accès aux données est gouverné par des politiques ou des règles de contrôle d'accès. Ou, la définition des politiques ne garantit pas à elle seule le fonctionnement correct du système.

Des politiques mises en place peuvent engendrer des décisions d'accès ne traduisant pas les exigences de sécurité souhaitées.

Ces politiques doivent respecter certaines propriétés essentielles pour pouvoir admettre que le système est sécurisé. Parmi ces propriétés, nous avons la validité, la cohérence, la complétude, le déterminisme. Ainsi, les systèmes de contrôle d'accès doivent être vérifiés de telle manière à ne pas empêcher les utilisateurs d'avoir les usages qui leur sont nécessaires, à ne pas permettre des fuites d'information, et ainsi, de faire en sorte que les systèmes d'informations puissent être utilisés en toute confiance. Une politique de sécurité se développe selon trois axes : physique, administratif et logique.

Le premier précise l'environnement physique du système à protéger. Le deuxième décrit les procédures organisationnelles (répartition des tâches, séparation des pouvoirs). Le troisième a trait aux contrôles d'accès logiques (qui, quoi, quand, pourquoi, comment) et s'intéresse aux fonctions d'identification, d'authentification et d'autorisation mises en œuvre par le système informatique.

Dans ce mémoire, nous nous intéressons particulièrement aux politiques d'autorisation (dites aussi politiques de contrôle d'accès).

Le contrôle d'accès est un mécanisme par lequel un système autorise ou interdit le droit à des entités actives (sujets : personnes, processus, machines, etc.) d'accéder et d'effectuer des opérations sur des entités passives (objets : fichier, dossier, etc).

La validation du mécanisme de contrôle d'accès est très importante pour garantir la fiabilité de la sécurité de l'application et l'absence de failles.

Notre travail s'intéresse donc à la validation. Notre objectif est de valider la conformité du mécanisme de sécurité vis-à-vis de sa politique de CA et faire la génération des attaques dans les politiques de sécurité. Ceci en utilisant la technique de mutation par l'exécution de certains scénarii qui déclenchent les mécanismes de CA. La mutation est une technique de test fonctionnel.

Les tests de sécurité sont générés à la partir de la politique de CA. Pour cela, nous proposons une technique d'injection de fautes en se basant sur un modèle de fautes pour politiques de contrôle d'accès. Nous pouvons donc détecter des erreurs par les tests fonctionnels et par les tests de sécurité.

Ce mémoire est structuré, en plus de l'introduction et de la conclusion générale, en trois chapitres. Le premier chapitre rappelle les concepts de base des

politiques et des modèles de sécurité existants. Le deuxième chapitre montre la nécessité de valider la politique de sécurité implémentée et les langages utilisés pour exprimer les règles de politique de sécurité pour un système ou une organisation et par conséquent comment bâtir des cas de test basés sur la mutation pour tester une politique de sécurité. Le troisième chapitre présente notre application. Il s'agira de décrire une concrétisation de nos idées à travers une implémentation d'un logiciel pour tester les règles de contrôle d'accès.

1.1 Introduction

Dans le présent chapitre nous présentons les principaux modèles de sécurité publiés dans la littérature, en l'occurrence les politiques discrétionnaires, Les politiques obligatoires et les politiques à base de rôles. Nous évaluons les avantages et les limites de ces modèles et politiques de sécurité.

Une politique de sécurité se développe selon trois axes : physique, administratif et logique. Le premier précise l'environnement physique du système à protéger (les éléments critiques, les mesures prises vis-à-vis du vol et des catastrophes).

Le deuxième décrit les procédures organisationnelles (répartition des tâches, séparation des pouvoirs).

Le troisième a trait aux contrôles d'accès logiques (qui, quoi, quand, pourquoi, comment) et s'intéresse aux fonctions d'identification, d'authentification et d'autorisation mises en œuvre par le système informatique. Dans ce mémoire, nous nous intéressons particulièrement aux politiques d'autorisation (dites aussi politiques de contrôle d'accès).

1.2 Les propriétés de sécurité

L'objectif de la sécurité informatique est la mise en œuvre de protection permettant d'assurer les propriétés suivantes : [SG06]

- La confidentialité : Assurer que l'information ne soit divulguée ou révélée qu'aux personnes autorisées.
- L'intégrité : Assurer que l'information contenue dans les objets ne soit ni créée, ni altérée, ni détruite de manière non autorisée.
- La disponibilité : L'accès par un sujet autorisé aux ressources et informations du système doit être toujours possible.

1.3 Les menaces

Ce sont les sources de violations potentielles de la sécurité. C'est l'ensemble des personnes et des événements qui présentent un danger pour un patrimoine

en termes de confidentialité, d'intégrité, et de disponibilité. On cite à titre d'exemple :

1. Usurpation de l'identité : un utilisateur anonyme utilise l'identité d'un utilisateur valide d'une application.
2. Accès aux données (tampering with data) : un utilisateur détruit ou modifie le contenu d'un message sans autorisation.
3. Divulgation d'informations (information disclosure) : des données confidentielles sont rendues visibles à des utilisateurs non autorisés.
4. Répudiation (Repudiability) : la possibilité qu'un utilisateur nie d'avoir effectué telle ou telle opération.

1.4 Les objectifs de la sécurité informatique

La sécurité informatique a pour objectif de répondre aux différentes menaces et tenant compte des types de ressources à protéger et leur niveau des sensibilités, Néanmoins, les principaux points sont les suivants : [SG06]

1. Empêcher la divulgation non autorisée de données.
2. Empêcher la modification non autorisée de données.
3. Empêcher l'utilisation non autorisée des ressources réseaux ou informatiques de façon générale.

1.5 Le contrôle d'accès

Le contrôle d'accès est un mécanisme par lequel un système autorise ou interdit le droit à des entités actives (sujets : personnes, processus, machines, etc.) d'accéder et d'effectuer des opérations sur des entités passives (objets : fichier, dossier, etc). Le contrôle d'accès fonctionne à plusieurs niveaux :

- Les mécanismes de contrôle d'accès au niveau des applications expriment des politiques de sécurité. Par exemple, dans une entreprise le personnel pourrait être affecté à plusieurs rôles différents. Chaque rôle peut initier plusieurs opérations possibles dans le système. Chaque opération nécessite des autorisations préalables pour pouvoir être exécutée.
- Les applications peuvent être au-dessus d'un « middleware », comme un système de gestion de base de données, qui met en application un certain

nombre de propriétés de protection. Le middleware utilisera les moyens fournis par les systèmes d'exploitation sous-jacent. Comme ce dernier construit les ressources de bas niveau telles que les fichiers et les ports de communication, il a donc la responsabilité de fournir les moyens de contrôler l'accès à ceux-ci.

- Finalement, les commandes d'accès de système d'exploitation se basent sur des fonctionnalités du matériel fourni par le processeur ou celle de la gestion de mémoire associée. Ces derniers contrôlent les accès d'un processus donné aux adresses de la mémoire.

Les modèles de CA (comme OrBAC ou RBAC par exemple) permettent de définir un ensemble de règles qui constitueront la politique de CA. Dans cette étude, nous considérons un modèle inspire par RBAC et étendu pour intégrer les notions de contexte et les règles d'interdiction. Nous avons essayé d'avoir une approche plus générale qui ne soit pas limitée à un modèle donnè. Plus formellement, une règle de CA consiste en cinq paramètres [TEJ09] :

- Statut S : permission ou interdiction
- Rôle R : dans un domaine de noms défini RN
- Permission P : dans PN
- Contexte C : dans CN

Une règle de CA est de définie ainsi : S (R, P, C)

Pour illustrer les règles d'une politique de CA, prenons l'exemple d'un système de librairie LMS (qui sera aussi utilisé après pour illustrer d'autres aspects) qui offre des fonctionnalités de gestion des livres, des comptes des utilisateurs et des prêts. Dans ce système, les utilisateurs peuvent réaliser trois opérations emprunter les livres, les réserver et les rendre. Les ressources dont on veut contrôler l'accès sont les livres et les comptes. Les entités (RN, PN) peuvent être ordonnées de manière hiérarchique. Par exemple, dans le cas de LMS, il y a deux types de rôles qui sont les emprunteurs et le personnel.

Les rôles Etudiant et Enseignant héritent donc du rôle Emprunteur. Cette hiérarchisation permet de définir des règles au niveau du rôle Emprunteur pour

qu'elles soient ensuite appliquées pour les rôles Etudiant et Enseignant. Enfin, on distingue trois contextes temporels qui sont les jours travailles, les jours fériés et les jours de maintenance. Les différentes entités sont présentées dans la figure 1.1.

Une fois les entités définies, il faut écrire la politique de contrôle d'accès. Voici quelques exemples de règles :

- R1: Permission (Administrator, Create Account, default)
- R2: Permission (Borrower, Borrower Activities, Working Days)
- R3: Prohibition (Borrower, Modify Account,default)

On distingue 2 types de règles : les règles primaires et les règles concrètes. Les règles primaires représentent l'ensemble des règles de nies. Quant aux règles concrètes, ce sont l'ensemble des règles obtenues après avoir dérivé les règles primaires en se basant sur la hiérarchie. C'est le cas de R2, qui s'applique a 'BorrowerActivity' et qui désigne l'ensemble des actions qu'un emprunteur peut effectuer (qui sont d'après la figure1.1) : BorrowerBook, ReserveBook et ReturnBook). Les règles primaires n'incluent pas ces règles dérivées.

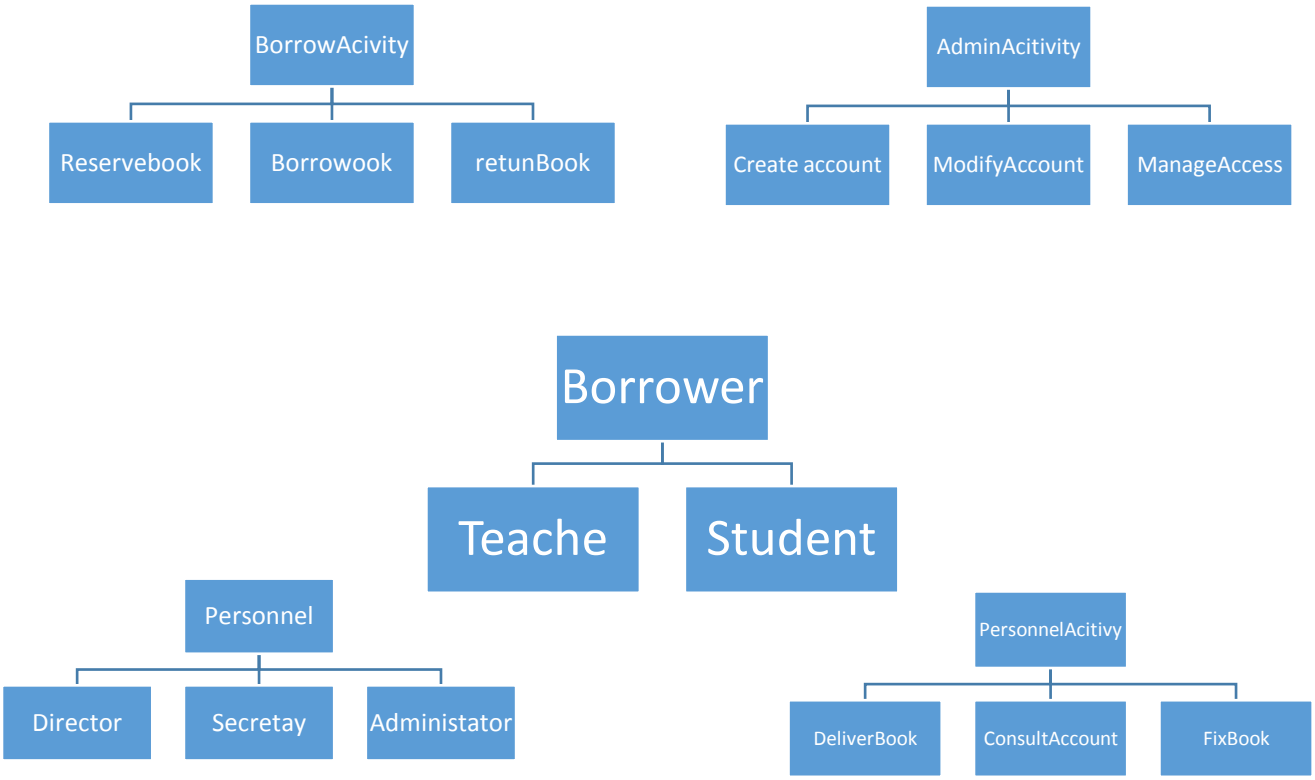


Figure 1.1 : Les entités de LMS

La distinction entre règles primaires et concrètes est importante pour la suite de l'article, quand on définira les critères de génération de tests. Après n'avoir défini la politique de CA.

1.5.1 Le but et les principes de base du contrôle d'accès

Les concepts de rôles, de vues et d'activités sont des concepts organisationnels. Chaque organisation définit ainsi les rôles, les activités et les vues dont elle souhaite réglementer l'accès en appliquant une politique d'autorisation.

Les modèles de contrôle d'accès (DAC, MAC, RBAC, ORBAC) reposent habituellement sur les trois entités : ***sujet, action, objet***. Donc, pour contrôler l'accès, on spécifie si un sujet a la permission de réaliser une action sur un objet.

- Les sujets sont abstraits en rôle. Un rôle est un ensemble de sujets sur lequel les mêmes règles de sécurité sont appliquées.
- Les actions sont abstraites en activité. Une activité est un ensemble d'actions sur lequel les mêmes règles de sécurité sont appliquées.
- Les objets sont abstraits en vue. Une vue est un ensemble d'objets sur lequel les mêmes règles de sécurité sont appliquées.

Chaque politique de sécurité est définie pour et par une organisation. Donc, la spécification de la politique de sécurité est complètement paramétrée par l'organisation. Donc il est possible de spécifier simultanément plusieurs politiques de sécurité associées à différentes organisations.

Le contrôle d'accès est le moyen le plus utilisé pour sécuriser les systèmes et les réseaux informatiques. L'utilité du contrôle d'accès est d'assurer les propriétés de sécurité tel que :

- **La confidentialité** : assurer que l'information ne soit accessible qu'à ceux qui ont l'autorisation.
- **L'intégrité** : assurer que l'information ne puisse être modifiée par des personnes non autorisées.

- **La disponibilité** : empêcher les données d'être supprimées ou de devenir inaccessibles. Cela s'applique non seulement aux informations mais aussi aux machines en réseau ou à d'autres aspects de l'infrastructure technologique.

L'impossibilité d'accéder à des ressources requises est appelée un refus de service (Denial of Service). [NET01]

1.5.2 Les phases d'élaboration d'un système de contrôle d'accès

L'élaboration d'un système de contrôle d'accès s'effectue par une approche multi phases basée sur les concepts suivants :

- **Les politiques de sécurité** : Le standard européen des ITSEC définit une politique de sécurité comme étant " l'ensemble des lois, règles et pratiques qui régissent la façon dont l'information sensible et les autres ressources sont gérées, protégées et distribuées à l'intérieur d'un système spécifique ".Un mécanisme de contrôle d'accès est l'un des outils permettant d'implémenter une politique d'accès aux données pour préserver la vie privée (Privacy) des utilisateurs.
- **Les modèles de sécurité** : ces modèles décrivent une représentation abstraite (souvent formelle) des politiques de sécurité et de leur fonctionnement. Ils permettent de faciliter la construction de preuves sur la sécurité d'un système, c'est la raison pour laquelle les efforts se sont focalisés autour de la construction des modèles pour la sécurité.
- **Les mécanismes de sécurité** : ceux-ci définissent les fonctions de bas niveau (logiciels et matériels) permettant de mettre en application les contrôles imposés par la politique de sécurité [MIC76].

1.6. Les modèles de contrôle d'accès

1.6.1 Les contrôles d'accès discrétionnaires (DAC)

Proposées en 1971 par Lampson, les politiques discrétionnaires sont basées sur l'identité des utilisateurs et les règles explicites d'accès qui stipulent que les utilisateurs (sujets) sont autorisés à définir leur propre règlement de sécurité sur les informations (objets) dont ils sont propriétaires. En d'autres termes, chaque objet a un propriétaire qui décide des sujets qui y ont accès.

Ces permissions d'accès sont représentées par une matrice dans laquelle chaque ligne correspond à un utilisateur, chaque colonne à une ressource et le contenu de cette matrice définit le droit d'accès (lecture, écriture, exécution, ...) de l'utilisateur sur la ressource.

Cependant, leur mise en œuvre est coûteuse en mémoire lorsque le nombre d'utilisateurs est important. Leur mise à jour est difficile et ne permet pas de contrôler une information ou ce qui en est fait une fois qu'elle a été accédée par un utilisateur légitime, ce qui rend le système vulnérable à des chevaux de Troie et l'expose à des fuites d'informations. Son avantage est l'utilisation d'une politique de gestion décentralisée. DAC est apte à être appliqué aux systèmes de fichiers.

Par exemple, appliqué aux systèmes de fichiers, les types d'accès de DAC :

- r : l'autorisation de lire l'objet
  w: la permission d'écrire
- x: l'autorisation d'exécuter
- c: l'autorisation de contrôle l'autorisation, la possibilité de modifier 'r w x' pour cet objet.
- Cp: le contrôle et la capacité passant de contrôle.

Voici un exemple simple de la politique du DAC.

La politique suivante définit deux sujets (kamel et Admin).

kamel peut lire ou exécuter fichier1, tandis que d'administration a le droit de lire, écrire et exécuter le fichier en plus du contrôle et capacité de passage.

1. POLICY systemDAC (DAC)
    - R1 -> DAC Rule (kamel r fichier1)
    - R2 -> DAC Rule (kamel x fichier1)
    - R3 -> DAC Rule (Admin cp fichier1)
    - R4 -> DAC Rule(Admin r fichier1)
    - R5 -> DAC Rule(Admin w fichier1)

Le contrôle d'accès discrétionnaire est un contrôle d'accès qui est à la discrétion du propriétaire de l'objet ou de toute personne qui est autorisée à

contrôler l'accès à l'objet. Des droits peuvent être passés d'un sujet (aussi appelé utilisateur) à un autre. [MEM13]

1.6.1.1 Modèle de Lampson

Dans ce modèle, le contrôle d'accès a trois composantes principales : un ensemble d'objets, un ensemble de domaines et une matrice d'accès. Les objets sont les entités dans le système qui doivent être protégées. Les domaines sont les entités qui ont accès à des objets. Chaque objet a un identificateur unique.

Chaque entrée dans la matrice est composée d'une liste d'opérations d'accès (attributs d'accès) permises pour être effectuées par un domaine sur un objet :

	O1	O2=D2	O3	O4
D1	Lire		*Propriétaire	
D2	Lire ; écrire	Control	*Lire	
D3	Propriétaire			

Tab 1 Modèle de Lampson

* drapeau pour indiquer le droit de délégation de l'attribut d'accès Le modèle présente un ensemble de règles qui déterminent comment les entrées de la matrice peuvent être modifiés :

1. R1 : un domaine peut supprimer des attributs d'accès pour n'importe quel domaine qui le contrôle. Exemple : domaine 2 peut supprimer des attributs de la ligne 1 car domaine 2 contrôle domaine 1.
2. R2 : un domaine ayant un privilège sur un objet o avec le droit de délégation, sans en être le propriétaire, peut transmettre ce privilège à d'autres domaines.
3. R3 : un domaine propriétaire d'un objet peut ajouter des attributs d'accès sur cet objet pour d'autres domaines.
4. R4 : un domaine propriétaire d'un objet peut supprimer des attributs d'accès sur cet objet

La mise à jour d'une politique de sécurité exprimée par ce modèle est coûteuse car si de nouveaux objets, de nouveaux sujets ou de nouvelles actions sont ajoutés dans le système, il devient nécessaire d'enregistrer toutes les permissions accordées pour ces nouvelles entités. Enfin, ce modèle ne permet pas de contrôler les interdictions ou d'exprimer des obligations. [MAH06]

1.6.1.2 Modèle HRU

Ce modèle est similaire à celui de BUTLER Lampson. Le système de protection est un ensemble fini R de droits et un ensemble fini C de commandes. Le système est un triplé (S, O, M), où S est l'ensemble de sujets courants, O est l'ensemble des objets courants, $S \subseteq O$, et M est une matrice d'accès, avec une ligne pour chaque sujet dans S et une colonne pour chaque objet dans O. M[s,o] est un sous ensemble de R. M[s,o] donne les droits que possède un sujet sur un objet.

L'arrangement d'autorisations est déni par un ensemble de commandes. Chaque commande a une partie conditionnelle et un corps.

La partie conditionnelle spécifie les droits qui doivent exister dans la matrice avant qu'un corps ne soit exécuté pour les arguments actuels. Le corps consiste en une séquence d'opérations primitives.

Ces dernières ajoutent ou suppriment des droits dans les cellules de la matrice, créent une ligne ou une colonne ou détruisent une ligne ou une colonne déjà existante dans la matrice. Il existe six primitives pour manipuler S, O et M :

1. ajouter r dans Mso∗.
2. retirer r de Mso.
3. créer sujet s.
4. détruire sujet s.
5. créer objet o.
6. détruire objet o.

∗ Mso désigne l'entrée de la matrice M contenant les droits d'accès du sujet s sur l'objet o.

Exemple :

{

command c(x1...xk)

if a1 in M [s1,o1] a2 in M [s2,o2] ...an in M [sm,om]

then op1 op2 ...opn

}

end

La flexibilité dans la définition des commandes fait apparaître un problème lié à la protection dans le système de contrôle d'accès ou « safety problem «. Etant donné un système avec une configuration initiale Q, le problème de protection détermine si un sujet « s « peut obtenir une autorisation « a « sur un objet « o « suite à l'exécution d'une séquence de commandes. Les auteurs montrent que le problème de protection est indécidable.

Cependant, il devient décidable dans le cas des systèmes mono-opérationnels (c.à.d chaque opération consiste en une seule action : lire, écrire, etc).Les mêmes auteurs ont proposé un algorithme de vérification de sûreté pour le système mono opérationnel. L'algorithme consiste à tester un ensemble fini d'exécutions.

Cependant, l'algorithme est difficilement utilisable du fait de sa complexité (NP-Complete) et l'hypothèse d'un système mono-opérationnel ne correspond pas aux systèmes réels[MIC76].

Enfin, ce modèle ne permet pas d'exprimer des interdictions, des obligations ou des recommandations.

1.6.1.3 Modèle TAM

Ce modèle est une extension du modèle HRU mais qui introduit la notion de type. La matrice d'accès typé (noté TAM pour Typed Accès Matrix) développée par Sandhu est composée de trois éléments :

R : définit un ensemble fini des droits d'accès.

T : définit un ensemble fini de types d'objets (types).

Ts : définit un ensemble fini de sujets avec Ts inclus dans T.

Les types et les droits sont définis quand le système est initialisé et demeurent constants. Exemple : T = {utilisateur, os, fichier}, Ts = {utilisateur, os}, R = {lire, écrire, exécuter, propriétaire}.

Ici le type os désigne un officier de sécurité. Le modèle présente la distribution des droits dans le système avec une matrice d'accès ; une ligne pour chaque sujet et une colonne pour chaque objet. L'état de protection est un quadruplet (S, O, t, M) où :

1. S : l'ensemble des sujets.

2. O : ensembles des objets.

3. t : la fonction qui accorde un type à chaque objet.

4. M : la matrice d'accès.

Un point important, dans ce modèle, est que la création de nouveaux types n'est pas possible. La gestion des types est prise en compte dans les opérations élémentaires décrites plus haut pour HRU.

Sandhu s'est intéressé à la version monotone de TAM, soit MTAM (Monotonic Typed Access Matrix), obtenue en ôtant les opérations de suppression (droits, sujets ou objets).

Il démontre que le problème de la sûreté est décidable dans le cas d'un modèle de protection MTAM où le graphe de création des sujets et objets est acyclique. Toutefois la complexité de ce problème reste NP.

C'est pourquoi Sandhu définit le modèle MTAM ternaire, dans lequel toutes les commandes ont au maximum trois arguments [MEM 10].

Au prix d'une perte d'expressivité, le problème de sûreté voit sa complexité ramenée à un degré polynomial.

1.6.1.4 Les limites des politiques discrétionnaires

L'avantage évident du DAC est qu'il est extrêmement flexible. Cependant, DAC ne fournit pas une vraie assurance sur la protection du flux d'information dans un système. Il est possible de dévier les restrictions d'accès indiquées par les autorisations.

Par exemple, un utilisateur qui est autorisé à lire des données peut les passer à d'autres qui ne sont pas autorisés à les lire sans que le propriétaire des données le sache. De plus, une analyse plus précise du problème de contrôle d'accès fait surgir l'utilité d'établir la distinction entre sujets et utilisateurs.

Le terme utilisateur désigne des entités passives possédant des autorisations et qui se connectent au système. Un utilisateur connecté au système génère un sujet ou un processus qui effectue à son compte les demandes au système.

Le fait que ces modèles ne font pas cette distinction les rend vulnérables aux attaques malveillantes telles que des chevaux de Troie [GIA95]

Ainsi, DAC semble être approprié aux environnements dont le partage d'informations est plus important que sa protection.

1.6.2 Les contrôles d'accès obligatoires (MAC)

Introduites en 1976 par Bell et LaPadula afin d'apporter une solution aux problèmes de fuites d'informations des modèles DAC, les politiques mandataires sont utilisées spécialement dans les environnements militaires à cause de leurs contrôles centralisés. Elles permettent à l'administrateur du système de définir des privilèges pour protéger la confidentialité et l'intégrité des ressources dans le système et affectent aux sujets et objets d'une organisation, des niveaux de sécurité qui sont non modifiables par les utilisateurs et qui régissent la manière dont l'information est transmise dans les systèmes. [NET02].

Figure1.2 : Les niveaux de sensibilité dans le modèle MAC**.**

En effet chaque sujet reçoit un niveau d'habilitation et chaque objet un niveau de classification. Si le niveau d'habilitation d'un sujet est supérieur ou égal au niveau de classification d'un objet, alors ce sujet a la permission d'accéder à cet objet.

Les règles de cette politique diffèrent selon qu'il s'agisse de maintenir des propriétés de confidentialité (on a le modèle Bell-Lapadula utilisé pendant longtemps dans les systèmes militaires pour assurer la confidentialité) ou d'intégrité (on a les modèles Biba, DTE, Clark et Wilson, la Muraille de Chine).

Ce modèle est très rigide car, il ne permet pas de gérer les exceptions entre les différents niveaux de sécurité. Il est adapté pour les administrations centralisées, les systèmes clos et contrôlés à haute confidentialité ou intégrité. Les systèmes ici peuvent être détournés et exploités pour transférer des informations via des canaux cachés non désignés pour la communication.

Dans ce modèle, les politiques de sécurité imposent que les décisions de protection ne doivent pas être prises par le propriétaire des objets concernés, et doivent lui être imposées par le système.

1.6.2.1 Modèle de Bell-LaPadula

Élaboré en 1975 pour le département de la défense américaine, ce modèle propose des règles pour prévenir la divulgation de l'information dans un système informatique. Les entités de base dans ce modèle sont [NET02]:

1. Entité active : sujet (processus, programmes en exécution)
2. Entité passive protégée : objets (données, fichiers, programmes, sujets)

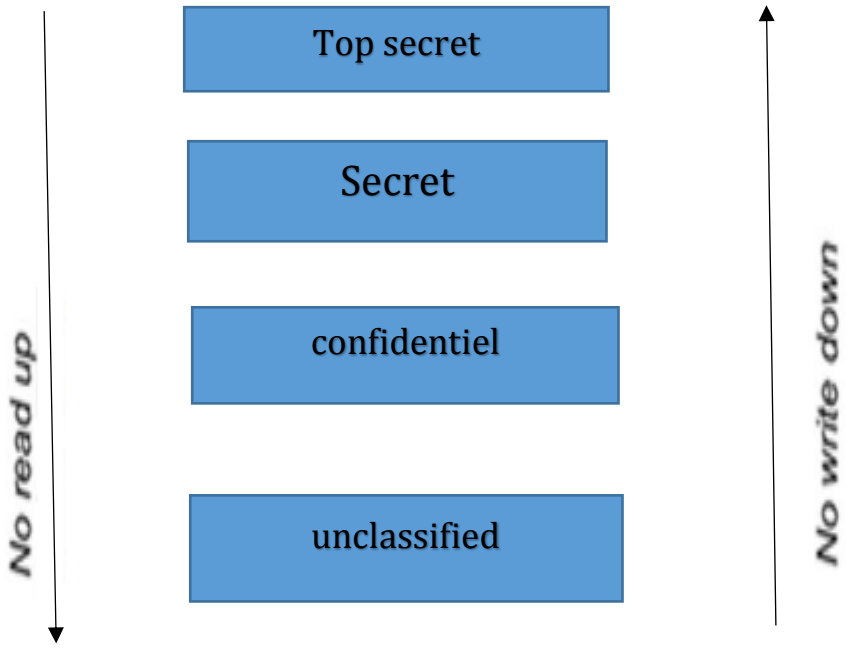


Figure1.3 Modèle de Bell-LaPadula

* Confidential cannot read secret

* Confidential cannot write unclassified

Simple proprieté de securité: no read up

*Propriété : no write down

La classification des sujets et des objets accorde un niveau de sécurité à chaque sujet et chaque objet. Formellement, chaque objet est associé à un niveau de sécurité de la forme (niveau de classification, un ensemble de catégories). Chaque sujet est également associé à un niveau de sécurité maximum et un niveau de sécurité courant, qui peut être changé dynamiquement. L'ensemble des niveaux de classification est ordonné par la relation « < «.

Le niveau de sécurité A dominé B si et seulement si le niveau de classification de A est supérieur au niveau de classification de B, et l'ensemble de catégories de B est inclus dans l'ensemble de catégories de A. Par exemple, si on a l'ensemble (top secret, secret, confidentiel, non classifié) et un ensemble de catégories (nucléaire, défense, etc) où : Non classifié < confidentiel < secret < top secret

Le niveau de sécurité (top secret, {nucléaire, défense}) domine le niveau de sécurité (secret, {nucléaire}) car top secret > secret et l'ensemble de catégories {nucléaire} est inclus dans l'ensemble {nucléaire, défense}. Le modèle est une machine à état (B, M, F, H) où :

B : représente l'accès courant d'un sujet à un objet. C'est un triplet (sujet, objet, attribut d'accès). Les attributs d'accès permis dans le modèle sont :

E : non observation et non altération

R : observation et non altération

A : non observation et altération

W : observation et altération M : la matrice(Mso), $s \in S$, $o \in O$, enregistre les modes d'accès permis pour un sujet sur un objet.

H : la hiérarchie, structures orientées imposées aux objets du système (arbres orientés, points isolés).

F : la fonction de niveau, les sujets et les objets reçoivent deux types de désignations formelles de sécurité : niveau de classification (non classifié, confidentiel, secret, top secret, etc) et une catégorie (nucléaire, défense, etc.). Donc, la désignation de sécurité est une paire (classification, catégorie) cette paire est appelée niveau de sécurité.

Pour éviter la divulgation de l'information, deux caractéristiques doivent être maintenues : La propriété simple (no Read up) : si on a un accès courant (sujet, objet, attribut d'accès) alors le niveau de sécurité du sujet domine celui de l'objet. Propriété étoile (no write down) : si un sujet a accès à un objet o1 et peut

altérer un objet o2 alors le niveau de l'objet o1 doit être dominé par le niveau de l'objet o2.

Exemple : admettons que le niveau de sécurité de l'objet o1 domine celui de l'objet o2. Si Sujet1 lit l'objet o1 et Sujet1 écrit les informations de l'objet o1 dans l'objet o2, Alors il y aura une fuite de données. [UNI76]

1.6.2.2 Le modèle de Brewer- Nash

Ce modèle pour régler le problème de conflit d'intérêt relié aux activités de consultation à l'intérieur des banques et autres institutions financières. L'objectif de ce modèle est de prévenir le flux d'information illicites. Exemple : un consultant ne doit pas avoir le droit d'accéder aux données confidentielles de deux compagnies concurrentes (conflit d'intérêt).

Muraille de chine (The Chinese Wall) Toute l'information de corporation est stoquée dans un système de fichiers disposés hiérarchiquement. Il y a trois niveaux d'importance :

1. Bas niveau : ce niveau comprend les items individuels d'information, chacun concernant une simple société. Les fichiers dans lesquels une telle information est stoquée sont dits des objets.

2. Niveau intermédiaire : tous les objets concernant la même société sont groupés dans ce qui est appelé un ensemble de données de compagnie (company dataset).

3. Haut niveau : tous les ensembles de données de compagnie dont les sociétés sont en concurrence sont groupés ensemble. Chaque groupe est appelé une classe de conflit d'intérêt (conflit of interest class). Ainsi, si dans une application, nous avons des informations sur la banque A, la compagnie pétrolière B et la compagnie pétrolière C. Deux classes de conflit d'intérêt sont créés, une pour les banques (contenant l'ensemble des données de la banque A) et une autre pour les compagnies pétrolières (contenant les ensembles de données pour la compagnie B et la compagnie C. La permission d'accès à un objet (o) est accordée pour un sujet (s) seulement si :

(a) L'objet o appartient au même ensemble de données déjà accédées par le sujet s (à l'intérieur de la muraille) ou

(b) L'objet o appartient à une autre classe de conflits d'intérêt

Une des distinctions par rapport aux politiques de Bell-LaPadula est le fait que les permissions d'accès d'un sujet à un objet dépendent des anciens accès de ce sujet.

L'information aseptisée (Sanitized information) : Une information est dite aseptisée si elle a été purgée de détails sensibles et elle n'est pas sujette à des restrictions d'accès. Considérons maintenant le cas suivant :

1. Sujet1 accède aux données de la compagnie C1 et les données de la banque A.
2. Sujet2 accède aux données de la compagnie C2 et les données de la banque A.
3. La compagnie 1 et la compagnie C2 appartiennent à la même classe d'intérêt.
4. Si Sujet1 lit les données de la compagnie C1 et les écrit dans des fichiers de la banque A, cela ne doit pas être permis car le Sujet2 peut accéder aux données de la compagnie C1.

Donc le modèle propose deux règles pour gérer la permission d'écriture (propriété étoile). L'accès en écriture à un objet O est permis seulement si les deux conditions suivantes sont vraies [TCW89]:

1. Accès en lecture est permis par les deux règles précédentes.

2. S ne peut lire aucun objet appartenant à un ensemble de données de compagnies différent de celui de O, et contenant des données non aseptisées.

1.6.3 Les modèles de contrôle d'accès basées sur les rôles RBAC

Proposées en 1992 par David Ferrailo et Richard Kuhn, les politiques basées sur les rôles sont adaptées pour les organisations comportant un nombre

important d'utilisateurs et d'objets. Leur principe général est d'introduire un niveau d'indirection entre utilisateurs et permissions. C'est le concept de rôle qui a été intercalé entre eux afin de mieux structurer les droits d'accès.

Il représente une fonction dans le cadre d'une organisation, c'est-à-dire qu'il décrit facilement les activités qu'un sujet a le droit d'accomplir au sein de cette organisation. Les sujets ayant reçus l'autorisation de jouer un rôle, héritent alors des permissions associées à ce rôle. Ceci dit, d'un côté nous avons des permissions qui sont accordées aux rôles, et de l'autre côté, des utilisateurs se voient affecter un ou plusieurs rôles.

De ce fait, si un nouvel employé est recruté dans une organisation, il suffit de lui assigner un (des) rôle(s) pour que le système l'intègre automatiquement dans les limites autorisées par la politique de cette organisation. Ce modèle simplifie l'administration des systèmes, facilite la compréhension de la structure de l'organisation et même de la politique de sécurité, réduit la complexité de gestion des droits d'accès et permet à ce que les rôles soient organisés de manière à former une hiérarchie permettant de raffiner les différentes permissions attribuées à chaque rôle. L'un de ses problèmes majeurs est le fait que, les utilisateurs ayant le même rôle obtiennent les mêmes privilèges.

Ce qui réduit la flexibilité des politiques de sécurité. On souhaiterait par exemple que dans certaines organisations telles qu'un hôpital, un médecin n'ait accès qu'aux dossiers de ses patients et ce n'est pas exprimable dans RBAC.

RBAC est le modèle le plus populaire et largement utilisé le contrôle d'accès. Plusieurs produits de l'industrie (bases de données, OS, applications d'entreprise, etc.) dépendent.

RBAC définit trois entités, à savoir :

1. Les utilisateurs,
2. Les rôles,
3. Les permissions.

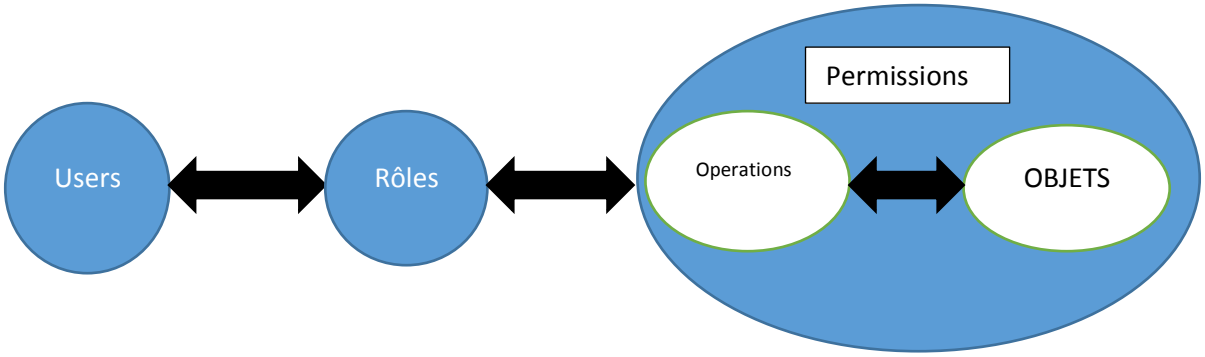


Figure 1.4 model RBAC

RBAC associe les utilisateurs aux rôles d'une part et des rôles avec des autorisations d'autre part. Deux types de règles doivent être définies :

- Rôle de l'utilisateur : les règles qui ont deux paramètres : un utilisateur et un rôle.
- Règles Rôle d'autorisation qui ont deux paramètres : rôles et autorisations

Pour illustrer comment construire une politique RBAC, nous utilisons l'exemple d'un système de gestion de bibliothèque. Ce système définit trois types d'utilisateurs : les étudiants, les secrétaires et un directeur.

Les étudiants peuvent emprunter des livres à la bibliothèque, le secrétaire gère les comptes des étudiants, mais seul le directeur peut créer des comptes.

Le modèle ORBAC (organisation Based Access Control) a été proposé pour faire face à cette limite.

POLICY LibraryRBAC (RBAC)

- R1 -> User Role(kamel Student)
- R2 -> UserRole(mohamed Director)
- R3 -> User Role (reda Secretary)
- R4 -> Role Permission (Student Borrow Book)

- R5 -> Role Permission (Personnel ModifyUserAccount)
- R6 -> Role Permission (Director CreateUserAccount)

Il étend le modèle RBAC et attribut des

Permissions/obligations/recommandations/interdictions pour la réalisation d'activités dans une organisation par un rôle dans un contexte donné. [LUC04]

- **Le cœur de RBAC (core RBAC)** : Il couvre les critères de base inclus dans tous les systèmes RBAC. Il reconnaît cinq éléments administratifs

1. Utilisateur : se réfère à la personne qui interface avec le système informatique. Dans plusieurs conceptions, un utilisateur peut avoir plusieurs login IDs et ces derniers sont actifs simultanément. Le terme sujet se réfère au processus jouant au nom d'un utilisateur.

Deux relations sont définies : la première relie un utilisateur à un sujet et une autre relie un sujet à l'ensemble de ces rôles actifs.

2. Rôle : les fonctions ou les responsabilités des employés dans une organisation.

3. Opération : c'est un processus actif invoqué par le sujet (exemple : lire, écrire, etc).

4. Objet : toute ressource accessible dans un système informatique (fichiers, périphériques comme l'imprimante, etc).

5. Permission : (privilège) une autorisation d'exécuter une opération sur un objet.

- **Propriétés :**

1. Autorisation d'un rôle : un sujet ne peut jamais avoir un rôle actif qui ne lui est pas permis.

2. Autorisation d'accès à un objet : un sujet S peut exécuter une opération Op sur un objet O seulement s'il existe un rôle R qui appartient à l'ensemble des rôles actifs du sujet S et il existe une permission accordée au rôle R qui autorise ce rôle à exécuter l'opération Op sur l'objet O.

- **Hiérarchie RBAC** : La motivation d'introduire cet aspect dans le RBAC c'est qu'à l'intérieur d'une organisation plusieurs rôles peuvent avoir des permissions communes. Par exemple : des permissions générales comme l'accès à un site Web interne, la possibilité de télécharger des documents, etc. Ces permissions peuvent être accordées à tous les employés ou à la plupart d'entre eux. Outre l'assignation de rôles et de permissions qui caractérisent la structure plate de rôle, la relation d'héritage de rôle crée une troisième forme d'autorisation. Si le rôle

  A hérite du rôle B, cela veut dire que toutes les permissions de B sont permises via le rôle A. En d'autres mots, les permissions de B sont un sous-ensemble de l'ensemble de permissions de A et tous les utilisateurs pouvant jouer le rôle A peuvent aussi jouer le rôle B. Deux types de hiérarchie entre les rôles ont été définis :

  1. Général : permet l'héritage multiple des permissions. Cela signifie qu'un rôle peut en même temps avoir un ou plusieurs descendants (potentiellement hérite des permissions à partir de plusieurs sources). De manière formelle, nous définissons un ordre partiel noté « ≥ « sur RH ⊆ ROLES ×ROLES tel que : étant donné deux rôles r1 et r2 :

     $r1 \geq r2 \Rightarrow \text{authorized}-\text{permissions}(r2) \subseteq \text{authorized}-\text{permissions}(r1) \wedge \text{authrized}-\text{users}(r1) \subseteq \text{authorized}-\text{users}(r2)$ avec $\text{authorized}-\text{users}(r) = u \in \text{USERS}, | r1 \geq r2 \wedge (u,r1) \in \text{UA}$ et $\text{authorized}-\text{permissions}(r) = p \in \text{PRMS}, | r1 \geq r2, (p,r1) \in \text{PA}$. où UA est l'ensemble des paires (utilisateur, rôle) et PA est l'ensemble des paires (permission, rôle).

  2. Limité : est défini comme la hiérarchie générale mais il ne permet pas l'héritage multiple.

3. Séparation de tâches (SOD) : La notion de séparation de tâches a été ajoutée ultérieurement dans le modèle RBAC.

Ce concept permet d'assurer qu'aucune personne n'a la capacité de commander toutes les étapes impliquées dans une opération à haut risque. Aucun utilisateur n'a assez de droits pour abuser seul du système. Cela permet d'éviter des fraudes et des erreurs majeures. Dans la littérature, plusieurs variétés de SOD ont été proposées où deux larges catégories :

1. Séparation statique de tâches (static separation of duty) : aucun usager ne peut avoir deux rôles qu'on peut désigner comme mutuellement exclusifs (conflictuels).

2. Séparation Dynamique de tâches (dynamic separation of duty) : aucun usager ne peut avoir, pendant une même session, deux rôles dits mutuellement exclusifs.

Critique de RBAC :

1. L'un des problèmes majeurs de ce modèle est le fait que tous les utilisateurs associés au même rôle possèdent forcément les mêmes privilèges. Ceci réduit la flexibilité des politiques de sécurité ainsi modélisées. En effet, dans une organisation tel qu'un hôpital, on peut souhaiter qu'un médecin n'ait accès qu'aux dossiers des patients dont il a le consentement.
2. Le concept de permission est primitif. Il n'y a pas de structure générique de permissions. Celles-ci sont considérées comme dépendantes de l'application concrète du modèle.
3. Le concept de permission est général, il est incorrect de considérer que la hiérarchie des rôles correspond à la hiérarchie organisationnelle. Par exemple, le directeur de l'hôpital a un rôle administratif supérieur au rôle de médecin. Pour autant, un directeur de l'hôpital n'est pas nécessairement un médecin, ainsi il n'est pas souhaitable d'accorder au directeur les permissions du médecin.
4. Il n'est pas possible de définir des obligations et des recommandations.

1.6.4 Les modèles de contrôle d'accès basées sur l'organisation OrBAC

Proposé en 2003 par Abou El Kalam , le contrôle d'accès basé sur l'organisation reprend les principes de rôles du modèle RBAC en offrant en plus, la possibilité de modifier la politique de sécurité en fonction d'une circonstance concrète, c'est-à-dire qu'il exprime facilement les permissions qui dépendent d'un contexte. En dehors des permissions, il offre la possibilité d'exprimer des obligations, des interdictions et même des recommandations dépendant bien évidemment des contextes. Il est centré sur le concept d'organisation (groupe structuré d'entités actives), et tous ses autres concepts sont définis par rapport à l'organisation. A partir des relations ternaires (habilite, utilise et considère), il définit les relations qui existent entre les entités du niveau concret (sujets, objets, et actions), du niveau abstrait (rôles, vues et activités) et l'entité contexte ainsi qui suit :

1. Entité Sujet : qui réfère aux utilisateurs par exemple, "Jean", "Marie ", " Pierre ", etc.
2. Entité Organisation : comme " département comptable ".
3. Entité Rôle : est utilisée pour structurer le lien entre les sujets et les organisations. Dans le domaine médical, le rôle " cardiologue " est joué par des utilisateurs alors que le rôle "service des urgences " est joué par des organisations.
4. Entité Objet : représente principalement les entités non actives comme les fichiers, les courriers électroniques, les formulaires imprimés, etc.
5. Entité Action : en globe principalement les actions informatiques comme " lire ", "écrire ", "envoyer ", etc
6. Entité Vue : un ensemble d'objets qui satisfont une propriété commune, ex : dans un hôpital, la vue "dossiers administratifs" peut désigner l'ensemble des dossiers administratifs du patient.

La même vue peut être définie différemment suivant l'organisation. Exemple : la vue "dossier médical » peut être définie dans un hôpital comme un ensemble de documents Word, et comme un ensemble de documents Latex dans un autre hôpital.

7. Entité Activité : correspond à des actions qui ont un objectif commun. Exemple : " consulter", "modifier ", " transmettre ", etc. L'activité " consulter " peut correspondre, dans l'organisation hôpital, à l'action " lire " un fichier, mais peut tout aussi bien correspondre à l'action " select " sur une base de données dans une autre organisation.
8. Entité Contexte : permet d'exprimer des circonstances telles que "urgence", " médecin traitant ", etc.

- Douze relations ont été introduites : La relation **Habilite** entre l'organisation, le sujet et le rôle.
- La relation **Utilise** lie une organisation, un objet et une vue.
- La relation **Considère** lie une organisation, une action et une activité.
- La relation **Définit** lie une organisation, un sujet, un objet, une action et un contexte.
- La relation **Permission** (org, r, a, v) signifie que l'organisation org accorde au rôle r la permission de réaliser l'activité a sur la vue v.
- Les relations Obligation et recommandation peuvent être définies de manière analogue à celle de la relation Permission.
- La relation Est_permis entre les sujets, les objets et les actions. Les relations Est_interdit, Est_obligatoire et Est_recommandé sont dénies de façon analogue à la relation Est_permis.
- La relation Sous-rôle (ST1, r1, r2) : est une relation introduite pour pouvoir modéliser la notion de hiérarchie entre les rôles.

Les contraintes sont exprimées dans le modèle par des règles s'appliquant à diverses relations.

OrBAC introduit la notion d'organisation, qui définit un champ d'application des règles de contrôle d'accès, et il est très utile pour traiter l'interopérabilité entre les différentes organisations. Il ajoute également la notion de contexte.Ce qui permet de définir des politiques dynamiques.

OrBAC définit les entités suivantes; organisations, rôles, activités, points de vue, le contexte. Il permet de définir trois types de règles

- Permission (Organisation, Rôle, Activité, vue, contexte).
- Interdiction (Organisation, Rôle, Activité, vue, contexte).
- Obligation (Organisation, Rôle, Activité, vue, contexte).

En outre, OrBAC soutient les hiérarchies plus des organisations, des rôles, des vues et même délégation et l'administration de la politique. Pour illustrer les politiques OrBAC, nous utilisons l'exemple du système de gestion de bibliothèque.

POLICY LibraryOrBAC (OrBAC)

- R1 -> Permission (Library Student Borrow Book Working Days)
- R2 -> Prohibition (Library Student Borrow Book Holidays)
- R3 -> Prohibition (Library Secretary Borrow Book Default)
- R4->Permission (Library Personnel ModifyAccount UserAccount WorkingDays)
- R5->Permission (Library Director CreateAccount UserAccount WorkingDays)

L'avantage de ce modèle est qu'il permet d'exprimer des règles contextuelles qui peuvent être spécifiques à une organisation. Il permet de spécifier au sein d'une même organisation structurée en plusieurs sous organisations plusieurs politiques de sécurité. Contrairement aux autres modèles qui ne modélisent que des politiques de sécurité se restreignant à des permissions statiques.

OrBAC offre la possibilité d'exprimer des règles relatives aux permissions, interdictions, obligations et recommandations. Malheureusement, il ne permet pas d'assurer qu'il n'y aura pas de conflits dans la politique de sécurité (par exemple, pour un sujet donné, une action donnée, et un objet donné, il nous faut détecter et résoudre une situation dans laquelle il serait possible de dériver à la fois une permission et une interdiction). Il ne montre pas comment détecter une violation de la politique de sécurité et pour finir, ne dit en rien si une politique de sécurité est cohérente ou pas.

En vue de se rassurer de sa sûreté, il faut vérifier que la politique mise en place est cohérente.

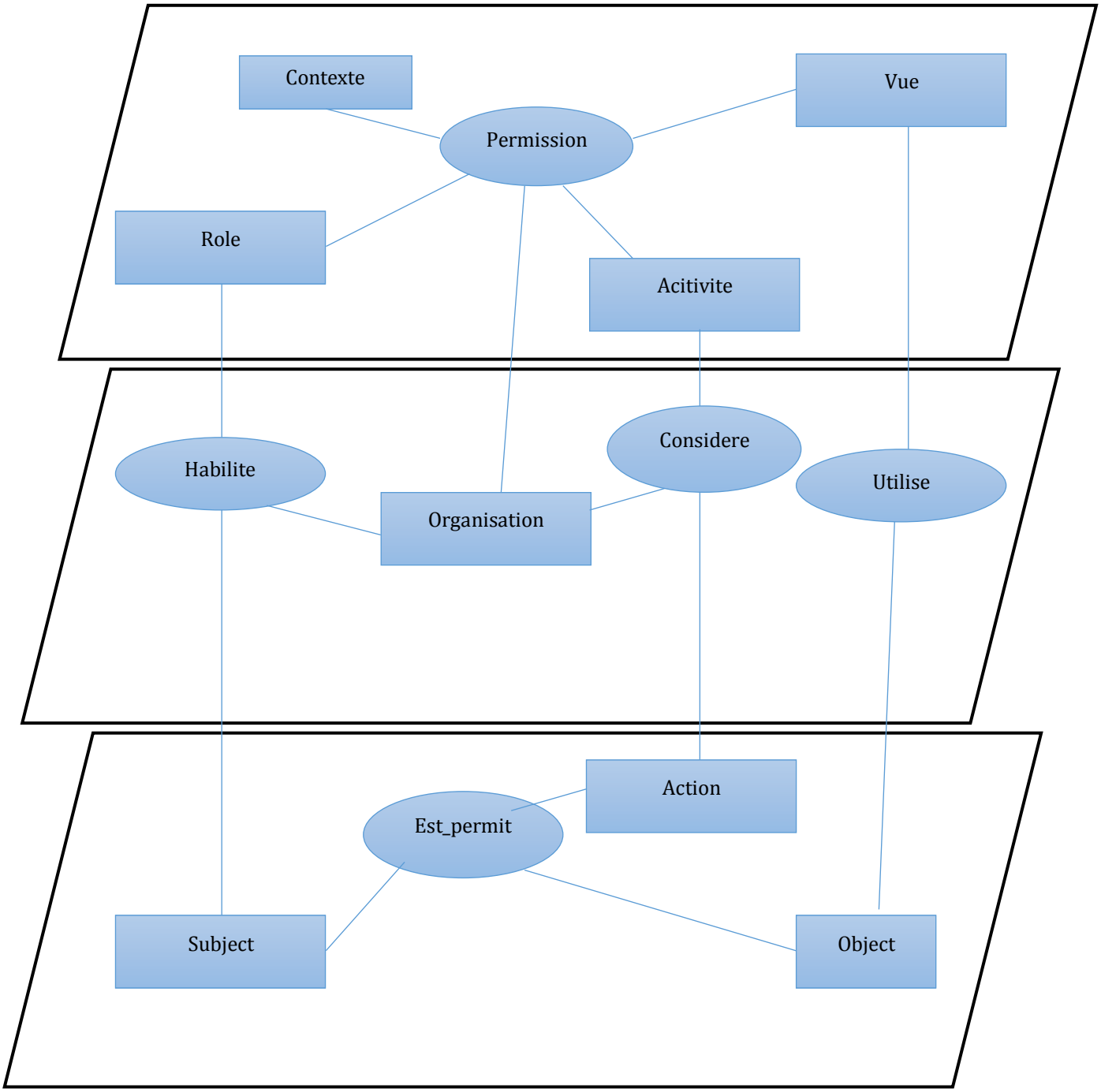


FIGURE 1.5 Modèle OrBAC

2.1 Introduction

Les politiques permettent de définir les règles de la sécurité et de la gestion des différents composants du système. Cela implique l'emploi d'un langage pour exprimer les règles d'affaires et les règles non fonctionnelles, et de donner aux utilisateurs la possibilité de tester et de corriger les politiques. Plusieurs langages tels que **XACML**, **Rei** ou **PONDER,** sont utilisés pour exprimer les politiques par rapport aux objectifs du système d'information. Ces langages peuvent définir plusieurs règles et politiques.

2.2 Tests de sécurité

Les tests de sécurité sont générés à la partir de la politique de CA. Ils ont pour objectifs de valider la conformité du mécanisme de sécurité vis-à-vis de sa politique de CA et faire la génération des attaques dans les politique de sécurité.

Les tests sécurité est composé de trois parties : l'intention du test (ce qu'on veut tester), la séquence du test (la suite d'opérations effectue) et l'oracle (qui vérifie et décide si le test réussi ou échoue)

2.2.1 Exemple Test de sécurité

Test qu'un emprunteur peut emprunter et rendre un livre les jours travaillés (comme spécifié par la politique de CA) :

- **Intention** : tester qu'un emprunteur a le droit d'emprunter et rendre un livre les jours travaillés
- **Séquence** : emprunter un livre et ensuite le rendre pendant les jours travaillés
- **Oracle** : Interroger le PDP pour vérifier que les bonnes règles ont été activées.

Les tests de sécurité peuvent être génères la partir de différents critères. Nous proposons, les critères suivants :

- Toutes les règles primaires : couvrir les règles primaires, si une règle est dérivable, tester une de ses règles dérivées.

- Toutes les règles concrètes : couvrir l'ensemble des règles concrètes.

Avant de faire le test de sécurité, Nous créons des mutants, Un mutant est une Utilisations de la mutation pour les tests de contrôle d'accès. Les tests sont lances sur les mutants pour détecter les erreurs.

2.3 la mutation

2.3.1 La création des mutants

La création des mutants est effectuée de manière systématique via des opérateurs de Mutation, Chaque opérateur injecte un type particulier d'erreur. Une seule erreur est injectée à la fois, pour créer chaque mutant. On propose les opérateurs de mutations Suivants :

1. **Operateurs basiques modifiant le type :**
   - ✓ PPR (permission to prohibition) : remplace une règle de permission par une interdiction.
   - ✓ PRP (prohibition to permission) : remplace une règle d'interdiction par une permission.

Exemple 1:

- Règle initial:

Permission (Secretary, Consult BorrowerAccount ,BorrowerAccount,default)

- Règle utiliser à la place **(PPR)** :

Prohibition (Secretary, ConsultBorrowerAccount, BorrowerAccount,default)

Exemple 2:

- Règle initial :

Prohibition (Student,GiveBackBook, Book, Holidays)

- Règle utiliser à la place **(PRP)** :

  Permission (Student, GiveBackBook, Book, Holidays)

2.Operateurs basiques modifiant les paramètres :

- ✓ RRD (role replaced with different one) : remplace dans une règle le rôle d'une règle par un autre rôle pris au hasard.

- ✓ CRD (context replaced with different one) : remplace dans une règle le contexte par un autre contexte pris au hasard.

Exemple 1 :

- Règle initial :

  Permission (Administrator,ModifyAccount, BorrowerAccount, default)
- Règle utiliser à la place **(RRD)** :

  Permission (Secretary,ModifyAccount, BorrowerAccount, default)

Exemple 2:

- Règle initial :

  Permission (Student, BorrowBook, BorrowerAccount, WorkingDays)
- Règle utiliser à la place **(CRD)**

  Permission (Student, BorrowBook, BorrowerAccount, Holidays)

3.**Operateurs basiques modifiant la hiérarchie :**

- ✓ RPD (parent role replaced with a descendant) : remplace dans une règle un rôle par un de ses descendants (modifiant ainsi les règles dérivées)

- ✓ APD (parent action replaced with a descendant) : remplace dans une règle une permission par un de ses descendants (modifiant les règles dérivées)

Exemple 1 :

- Changer les hiérarchies de rôle, Règle initial :

  Permission (Borrower,reserveBook, Book,WorkingDays)
- Règle utiliser à la place**(RPD)** :

Permission (Teacher,reserveBook, Book,WorkingDays)

Exemple 2 :

- Changer activité hiérarchies, règle initiale :
  Permission (Student,BorrowerActivity, Book,WorkingDays)
- Règle utiliser à la place **(APD)** :
  Permission (Student,BorrowBook, Book,WorkingDays)

2.Operateur avance :

- ANR (Add New Rule) : ajoute une nouvelle règle ne faisant pas partie des règles définies.

Examples de ANR

Permission(Teacher,consultPersonnelAccount,PersonnelAccount, WorkingDays)
Permission (Secretary,ManageAccess , PersonnelAccount,MaintenanceDay)

Les mutants crées contiennent par construction des règles différentes de la politique initiale parce qu'ils ajoutent une nouvelle règle ou modifient une règle existante.

On distingue deux types d'opérateurs de mutation, les opérateurs basiques (tous les opérateurs sauf ANR) et l'opérateur avancé de mutation qui est ANR. Ce dernier est particulier parce qu'il vise la tester le comportement par défaut du mécanisme de CA. En effet, toute politique de CA contient X règles et 1 règle par défaut (permission ou interdiction) qui s'appliquent quand les autres règles ne correspondent pas aux entités d'une requête. L'opérateur ANR permet de créer des règles qui viennent se substituer la règle par défaut et viennent compléter les X règles définies. Il s'agit donc d'un operateur avancé puisqu'il permet de tester plutôt la robustesse du mécanisme de CA.

2.4 langage de politiques de sécurité XACML

2.4.1 Définition

XACML (*eXtensible Access Control Markup Language)* est un langage de politique permettant d'exprimer les règles utilisées pour le contrôle d'accès. La spécification du langage XACML est définie par un ensemble de schémas XML, qui décrit la syntaxe des règles et des politiques d'accès. Le langage XACML vise à atteindre plusieurs objectifs comme :

- Assurer une protection efficace pour les ressources du système.
- Permettre de concevoir un système indépendant de la plate-forme utilisée.
- Permettre d'intégrer les politiques XACML dans des applications déjà existantes.

2.4.2 La norme XACML

XACML est une norme de l'OASIS. Elle consiste essentiellement en un langage XML qui permet d'encoder, d'une part, des politiques de contrôle d'accès de type RBAC étendu et, d'autre part, des requêtes/réponses aux décisions d'autorisation. La norme décrit aussi un modèle de flux de données pour une application et un gestionnaire de mise en œuvre d'une politique XACML [MEG12].

2.4.3 Description du langage XACML

Une politique en XACML (figure 2.1) est composée d'une cible, d'une ou plusieurs règles et d'un algorithme de combinaison des règles.

```
<policy PolicyId="deny-test"
RuleCombiningAlgId= " rule-combining-algorithm:fist-applicable"
< Description > structure de la politique </Description>
<Target>
<Subjccts> ... </Subjects>
<Resources>... </ Resources >
<Actions>.. </Actions>
<target>
<Rule/>    </Rule>    #
régIe      1      <Rule/>
</Rule> # régIe 1

</Policy>
```

Figure 2.1: Exemple de la structure d'une politique XACML

La décision de la politique est calculée en utilisant des algorithmes de combinaison sur les décisions des règles.

La figure (2.2) suivante présente le diagramme de classes d'un modèle XACML qui illustre les concepts du langage utiles pour la description d'un document XACML. Un document XACML définit un ensemble de politiques (PolicySet).

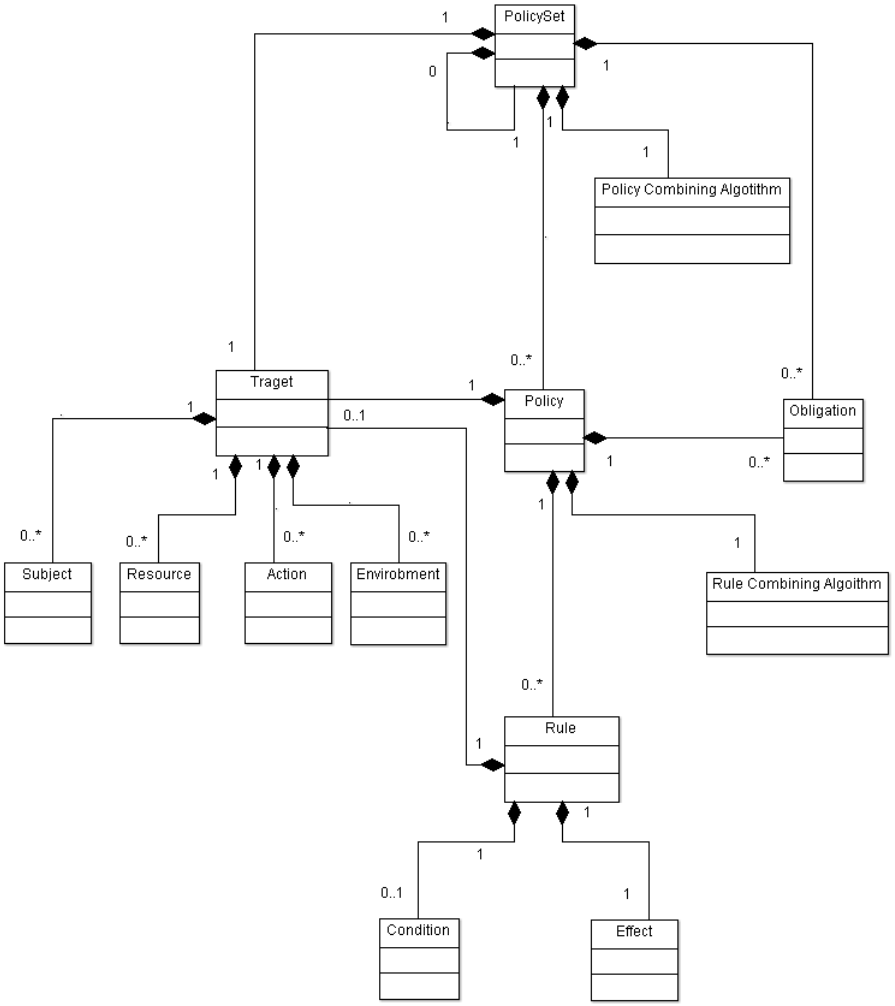


Figure 2.2: Modèle du langage XACML [NET03]

Chaque politique (Policy) vise une cible (Target) et comporte des règles (Rule) qui sont évaluées dans le contexte courant de la requête d'autorisation reçue de l'environnement. Les règles d'une politique sont évaluées seulement lorsque la cible de celle-ci correspond à la cible de la requête d'autorisation et les résultats de cette évaluation sont combinés pour fournir un résultat unique pour la politique. La combinaison de ces résultats est faite suivant un algorithme de combinaison (Rule Combining Algorithm) précisé par la politique. La politique comporte également des obligations (Obligation).

XACML définit les obligations comme des actions à exécuter par le gestionnaire de mise en œuvre lorsque la requête d'autorisation traitée est autorisée. Cette fonctionnalité est utilisée par les règles, les politiques et les ensembles de politiques. [MEG12]

2.4.3.1 La cible

XACML a introduit la notion cible « *Target* » afin d'identifier les règles et les politiques qui concernent une requête. La cible est composée d'attributs qui décrivent le sujet, les ressources, les actions et l'environnement :

- Le sujet décrit les attributs de l'utilisateur qui a fait une demande d'accès.
- La ressource décrit les attributs de l'objet auquel l'accès est demandée.
- L'action représente les attributs qui décrivent les mesures que le sujet veut prendre sur la ressource demandée.
- L'environnement concerne les attributs détenant des informations sur le contexte.

Chacun de ces composants est déterminé par des propriétés. Un sujet peut être défini par un identificateur, un groupe auquel il appartient, un rôle etc. Une ressource peut être caractérisée par un identificateur, des propriétés et un type. La même chose pour l'action qui peut être définie par un identificateur, le nom d'action à effectuer. Par exemple, nous considérons la requête suivante :

Un étudiant identifie par « *user_id* » qui appartient au groupe « A » veut accéder à un document public en mode écriture. Pour cette requête nous pouvons distinguer :

- Le sujet : user_id .
- La ressource : document public.
- L'action : écriture.

Les propriétés des sujets, ressources et actions sont appelées attributs. Chaque attribut possède une valeur.

2.4.3.2 La règle

Une règle de contrôle d'accès est définie avec le langage XACML comme étant un ensemble de prédicats qui répondent aux questions suivantes :

- Quels sont les sujets concernés ?
- Quelles sont les ressources demandées ?
- Quelles sont les actions demandées ?
- Quelle est la décision à renvoyer ?

Une règle XACML est composée d'un effet, d'une cible et d'une condition :

- **Effet :** détermine la décision de la règle. C'est soit « *Permit »* soit « *Deny » .*
- **Cible :** permet de déterminer si la règle correspond à la requête ou non.
- **Condition :** décrite par un prédicat sur les attributs de la règle.

Par exemple, soit *« R »* une requête que nous désirons évaluer par rapport à une règle. La première étape de validation consiste à évaluer la cible de cette requête afin de savoir si la règle « R » peut être appliquée ou non :

- Si la cible de la règle satisfait celle de la requête, les conditions de la règle seront évaluées ;
- Dans le cas où ces conditions seraient satisfaites, la réponse de la règle sera l'effet spécifié.

Dans le cas contraire ou si la cible de la règle ne correspond pas à la requête, la règle « R » ne sera pas considérée. Prenons l'exemple suivant :

- **Cible :**

- L'identificateur du sujet est l'étudiant.
- La ressource est un document public.
- L'action est écriture.

- **Condition :**

- L'étudiant doit appartenir au groupe des étudiants de maitrise (groupe A).

- **Effet :**

- Autorisation d'écriture.

Si un étudiant de groupe « A » demande de modifier un document public, cette règle sera appliquée et retournera la décision *« permit* ». Si un étudiant de groupe « A » demande de modifier un document privé, cette règle ne sera pas appliquée car elle traite seulement les documents publics. Si l'étudiant n'appartient pas au groupe « A », cette règle ne sera pas appliquée car sa condition n'est pas satisfaite.

2.4.3.3 La politique

Une politique est exprimée par une cible, un ensemble de règles et un algorithme de combinaison.

Pour identifier les politiques appropriées à l'évaluation de la requête, il faut d'abord comparer la cible de la requête avec la cible de la politique, et par la suite, vérifier les conditions des règles de la politique afin de déterminer la décision « permit» ou « deny ».

Il est possible que les règles d'une politique retournent des décisions différentes par rapport à une requête donnée. L'algorithme de combinaison des règles permet de spécifier comment déterminer la décision de la politique.

Les décisions possibles sont :

- ***Permit*** *:* l'accès est autorisé.
- ***Deny*** *:* l'accès à la ressource est refusé.
- ***Indeterminate*** *:* il n'est pas possible d'appliquer la politique à la requête parce qu'un élément (sujet, ressource, etc) est inconnu ou parce que la construction de la politique ne permet pas d'aboutir à une décision (erreur)
- ***NotApplicable :*** il n'est pas possible d'appliquer la politique à la requête car elle ne contient aucune règle qui s'applique à la requête.

Un ensemble de Politiques *(Policyset)* est une agrégation de plusieurs politiques ou des ensembles de politiques. *Policyset* contient aussi un algorithme de combinaison pour combiner la décision de politiques.

2.4.3.4 Algorithme de combinaison

Nous rappelons qu'un algorithme de combinaison permet de calculer la décision d'une politique et d'un ensemble de politiques à partir des décisions de leurs agrégats. XACML offre quatre algorithmes de combinaison prédéfinis :

- ***Permit-overrides*** *:* s'il y a une règle évaluée avec effet « *Permit »* alors la décision de combinaison donne également un effet «*permit» .*
- ***Deny-overrides*** : s'il y a une règle évaluée avec effet « *Deny »* alors la décision de combinaison donne un effet « *Deny* ».
- ***First-applicable*** : avec l'algorithme First-applicable, l'ordre d'évaluation des règles est important. La politique prend l'effet de la première règle qui s'applique (on ignore la règle non applicable).
- ***Only-one-applicable*** : si plusieurs règles sont applicables, la décision «*Indeterminate »* est retournée, sinon la décision de la politique est celle de la règle applicable.

2.4.4 Evaluation de la politique XACML

En plus des informations fournies dans la requête, une politique XACML pourrait exiger des informations supplémentaires pour prendre une décision. Ces informations sont récupérées à partir d'une base de données externe.
Lors de ('évaluation d'une requête par rapport à une règle ou une politique, plusieurs types d'erreurs peuvent engendrer la décision « lndeterminat » :

•Des erreurs de réseau : une politique XACML peut contenir une règle qui réside sur une machine distante temporairement inaccessible.

•Des erreurs de syntaxe : la requête et la politique XACML peuvent comporter des erreurs de syntaxe .

•Les requêtes incomplètes : si la requête ne contient pas les valeurs de certains attributs utilisés dans la cible d'une politique.

Si aucune erreur ne se produit au moment de l'évaluation de la cible d'un contenant, sa décision dépend de l'évaluation des décisions des contenus en utilisant l'algorithme de combinaison. [GH97]

2.5 Exemple de la politique XACML :

Soit « Pol_document » une politique qui permet le contrôle d'accès aux documents. L'accès à un document dépend du rôle de l'utilisateur, du type de document et de l'action demandée, voir le tableau 3.1 :

- La politique est décrite de la façon suivante : Tous les utilisateurs ont le droit de consulter les documents « public » ;
- Les gestionnaires ont le droit d'accès aux documents « privé » en lecture seulement ;
- Par contre l'administrateur a le droit de consulter et de modifier les documents « privé ».

	Attribut	**Valeur**
Sujets	Rôle	Administrateur
		Manager
		Employé
Ressources	Document	Privé
		Public
Actions	Action	Consulter
		Modifier

Tab 3 Exemple des valeurs de la cible XACML

Pour mettre en place la politique « Pol_document », nous décrivons d'abord les règles suivantes :

- R1 : L'administrateur a le droit de consulter et de modifier tous les documents.
- R2 : Les gestionnaires ont le droit de consulter les documents privés.
- R3 : Tous les utilisateurs ont le droit de consulter les documents publics.

La politique « Pol_document » est représentée schématiquement par la figure 2.3

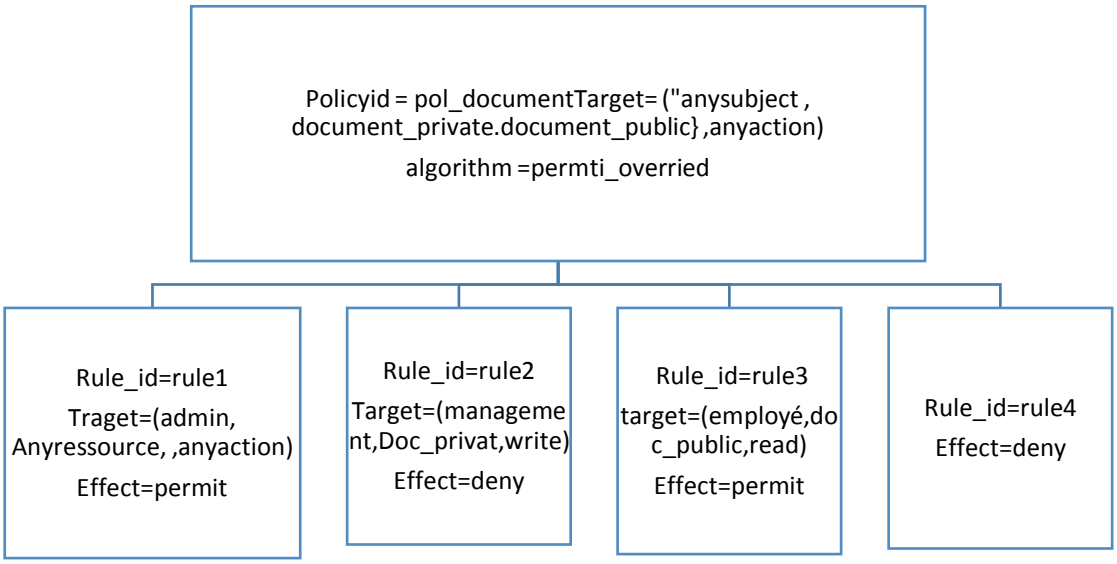


Figure 2.3 Un exemple de la politique XACML

2.6 L'architecture de XACML

L'architecture du système utilisant XACML se compose d'entités. La figure (2.4) synthétise l'ensemble des entités de l'architecture de XACML :

- **Seuil d'application de politique** *(Policy Enforcement Point (PEP))* est l'entité du système qui contrôle la protection des ressources. Le *PEP* fonctionne en collaboration avec le gestionnaire de contexte qui permet d'obtenir *les valeurs des* attributs des entités du système.
- **Centre de décision de politique** *(PDP* : *Policy Decision Point)* est l'entité qui prend en charge de déterminer les règles et les politiques applicables à une demande. Après l'évaluation des cibles et des conditions de l'ensemble des règles, le PDP renvoie la réponse au PEP.

- **La source d'information de politique** *(PlP : Policy Information Point)* a le rôle d'extraire des informations supplémentaires non présentes dans la demande d'accès. Le PIP permet de chercher les informations au sein des sources externes dans différentes plates-formes.

Le centre de décision de politique « PDP : *Policy Decision Point* » est considéré comme une boîte noire, la requête XACML serait à l'entrée de PDP, puis la sortie serait la réponse du XACML. Sur la base des informations fournies par la

requête, une politique XACML est vérifiée pour déterminer si la demande d'accès à une ressource est autorisée ou non. **[ME11]**

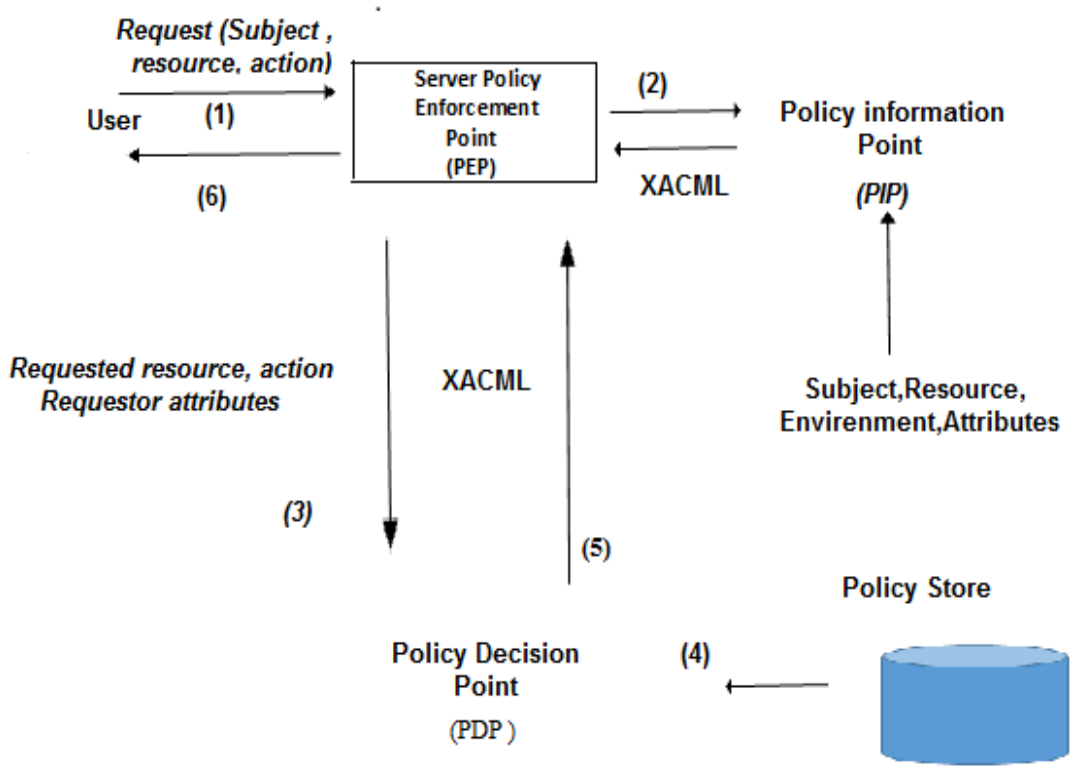


Figure 2.4: Schéma d'une requête de permission avec le langage XACML **[ME11]**

2.7 Autre langages de politiques de sécurité

2.7.1 Langage de politiques Rei

Description du langage *Rei*

Le langage de politiques Rei' permet aux utilisateurs de spécifier des politiques basées sur la logique déontique qui est une formalisation mathématique qui permet de modéliser les obligations, les interdictions et les permissions dans une organisation.

Chaque politique décrit avec le langage Rei est associée à une entité. Elle est définie par des règles qui gèrent l'ensemble des droits de ['entité, ses interdictions et ses obligations.

La spécification du langage Rei offre des moyens d'analyse des politiques et une résolution des conflits avec l'utilisation de la classe « RulePrioriy » qui spécifie la priorité des règles.

Par exemple, pour fixer les priorités entre deux règles potentiellement contradictoires RuleA (a la permission d'imprimer) et RuieB (une interdiction de l'impression), la classe rulePriority peut être utilisée pour indiquer que l'interdiction détient la priorité sur l'autorisation.

```
<policy:rulePriority rdf:resource="#PriorityBA"1>
   <metapolicy:RulePriority rdfJD="PriorityBA"> <metapolicy:ruleOfGreaterPriority
         rdfresource="#RuleB"I> <metapol icy: ru leOtLesserPriority
         rdfresource="#RuleA "/>
                                   <mctapolicy:RulcPriority>
```

Figure 2.5 : Exemple de la politique « rulePriority »

- **La spécification du langage *Rei***

La spécification du langage ***Rei*** est composée de plusieurs ontologies :

- *ReiPolicy .*
- *ReiMetaPolicy*
- *ReiEntity .*
- *ReiDeontic.*
- *ReiConstraint.*
- *ReiAnalysis.*
- *ReiAction.*

Chacune de ces ontologies décrit des classes et des propriétés. L'ontologie de base « ReiAction » comprend la description des actions.

Chaque action est décrite par :

- Son identificateur unique.
- Les objets de la cible, sur laquelle l'action peut être effectuée ;
- Un ensemble de conditions préalables qui doivent être remplies avant que l'action puisse être effectuée.
- Les effets.

- **Analyse des politiques *Rei***

La classe « *What-if »* permet d'analyser les politiques. Plus précisément, elle permet de spécifier des modifications temporaires portées à la politique ou à des entités afin de tester leurs effets.

Il existe deux sous-classes d'analyse de la classe *« what-if» :*

- ***WhatifProperty*** cette classe permet d'ajouter ou de retirer temporairement une propriété d'une certaine valeur à une entité pour vérifier des modifications portées aux entités.
- ***WhatifPolicyRule*** cette classe permet d'ajouter ou de retirer une règle pour vérifier les changements portés aux politiques. ***[NET04]***

2.7.2 Langage de politiques *Ponder*

Description du langage *Ponder*

Ponder est un langage orienté objet permettant de spécifier les politiques de sécurité et de gestion des systèmes d'objets distribués. Il fournit des techniques de structuration des politiques visant à répondre à la complexité de gestion des politiques dans les grands systèmes d'information de l'entreprise.

Le langage ***Ponder*** permet de décrire quatre types de politiques : les autorisations, les retenues, les délégations, et les obligations :

- **Politique d'autorisation :** Ce type de politique définit, pour un ensemble de domaines, les actions qui peuvent être effectuées ou non. Une politique d'autorisation positive définit les actions autorisées pour les sujets. Une politique d'autorisation négative précise les actions interdites pour les sujets.

- **Politique de retenue** *(Restrain Policies)* : Consiste à définir les actions que les sujets ne doivent pas effectuer sur les objets. Ce type de politiques agit comme des restrictions sur les actions mises en œuvre par les sujets.
- **Politiques de délégation** : ce type de politique décrit les autorisations et les droits d'accès à transférer d'un groupe d'utilisateur à un autre.
- **Politiques d'obligation** : ce type de politique permet de préciser les actions qui doivent être effectuées au sein du système quand certains événements se produisent, fournissant ainsi la capacité de réagir aux circonstances changeantes.

Les deux exemples de la figure 2.6 décrivent l'autorisation d'accès au document privé :

1. La première règle décrit une autorisation positive : l'utilisateur avec le profil administrateur « *admin* » a le droit d'accès au document de type « *documentprivat »* en mode lecture, écriture, et modification.

2. La deuxième règle décrit une autorisation négative : les sujets ayant le profil gestionnaire « manager » n'ont pas le droit d'écrire ou de modifier les documents de type « *documentprivat* ».

```
inst auth+ documentPolicyOps {
subject /Admin;
target <document_privat> /document;
action read(), write(), update;
 }
 inst auth-documentPolicyOps {
 subject /manager;
 target <document_privat> /document;
 action write(), update;
}
```

Figure 2.6 Exemple de politique d'autorisation

Les composantes clés du langage *Ponder* (groupes, rôles, relations et le rôle hiérarchie) permettent de structurer les règles, et de réutiliser les spécifications :

- **Les groupes** : Il s'agit d'un regroupement des politiques. C'est un concept commun à la plupart des langages de politique (comme Policy Sel pour langage XACML).
- **Le rôle** : cette composante fournit un regroupement sémantique des politiques avec un thème commun. Le rôle est représenté par un ensemble de propriétés profile, classe, type.
- **Les relations** : une relation est une composante qui inclut des politiques ayant des rôles communs.
- **Les rôles hiérarchie** : Ponder permet la spécialisation des types de politiques à travers le mécanisme d'héritage semblable aux langages orienté objets. [NNEM01]

2.8 Comparaison des langages de politiques

[PLJS] décrit un ensemble des critères qui permettent de juger la qualité d'un langage de politique :

- **Formalisme** : la syntaxe et la structure du langage de la politique doit être claire et sans ambiguïté. Le sens d'une politique écrite avec le langage devrait être indépendant de sa mise en œuvre particulière.
- **La flexibilité** est la définition des abstractions pour gérer une grande variété de types. L'architecture du système devrait être suffisamment souple pour autoriser l'ajout de nouveaux types.
- **L'interopérabilité** : le langage doit disposer une architecture lui permettant d'inter opérer avec d'autres architectures qui peuvent exister dans d'autres plates-formes.
- **Détection des conflits** : le langage doit être en mesure de vérifier qu'une politique n'entre pas en conflit avec une autre.
- **Évolutivité** : il convient de maintenir des performances de qualité sous une charge accrue du système.

Le tableau suivant présente une comparaison des langages XACML, Ponder et Rei selon un ensemble des critères préétablis considérés comme importantes :

Langage	**XACML**	**Ponder**	**Rei**
Flexibilité	Oui	Oui	Non
Formalisme	Oui	Oui	Non
L'interopérabilité	Oui	Non	Oui
Détection des conflits	Oui	Oui	Oui
Evolutivité	Moyen	Moyen	Elevé

Tab 4 : Comparaison des langages de politiques **[PLJS]**

3.1 Introduction

L'Implémentation est la dernière étape dans ce travail, elle est l'étape la plus importante pour atteindre l'objectif voulu qui est la réalisation des fonctionnalités présentés dans notre application. Pour obtenir le succès il faudra regrouper plusieurs facteurs majeurs comme les logiciels utilisés dans travail est le langage de programmations ….

Une politique de sécurité est une déclaration d'intention concernant les méthodes envisagées pour protéger vos actifs numériques et surveiller l'organisation. Elle représente un référentiel d'informations central pour la gestion, le personnel et les tiers, et regroupe tout, des processus et procédures à une description des mesures techniques en place et des méthodes de reprise sur sinistre, en passant par les fonctions et responsabilités des employés.

La politique de sécurité doit s'appuyer sur une connaissance des menaces les plus sérieuses auxquelles vous êtes confronté.

Considérez la politique de sécurité comme un plan d'action qui décrit dans les grandes lignes les informations critiques de la société et les méthodes de protection de celles-ci.

3.2 Environnement de programmation

3.2.1 Langage JAVA :

Notre application a été réalisée avec Le langage java Eclipse neon,Le langage Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld.

Figure 3.1 logo java

La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java.

La particularité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications. Pour cela, divers plateformes et frameworks associés visent à guider, sinon garantir, cette portabilité des applications développées en Java. .[NET05]

ALPFA plugin Conçu pour intégrer à votre environnement de développement, la langue Axiomatics d'autorisation (ALFA) est similaire à la hausse des langages de haut niveau comme Java et C #, ce qui permet aux développeurs d'écrire rapidement et facilement des politiques XACML.

Axiomatics a proposé ALFA en tant que profil de XACML intitulé «Abbreviated Langue d'autorisation" .[NET06]

Figure 3.2 logo axiomatics

Le plug-in pour Eclipse ALFA est un outil qui convertit votre IDE de programmation Eclipse à un éditeur de stratégies d'autorisation en utilisant la syntaxe ALFA.

ALFA politiques peuvent alors facilement être convertis, au sein de l'IDE, en XACML 3.0 politiques réelles et ensuite chargés dans votre outil de gestion de la politique XACML.

3.2.2 L'environnement matériel :

Pour développer cette application nous avons utilisé une machine configurée comme suit :

- Pc portable HP
- Mémoire Vive : 4 Go.
- Disque Dur : 500 Go.
- Processeur : AMD E1_2100 APU with Radeon™ HD graphique 1.00 GHZ.
- Type de système : Windows 8.1

3.4 Application :

3.4.1 L'objectif

L'objectif de notre application est l'utilisation du langage XACML pour exprimer les règles des contrôle d'accès en politique XACML , après cette création du politique XACMl (fichier xml) on fait une génération des mutants avec les différents opérateurs de mutation. Ces mutants sont des attaques générées dans les politique de sécurité initial. Enfin on fait une comparaison entre le fichier xml initial et le fichier xml mutant.

3.4.2 Schéma général de l'application

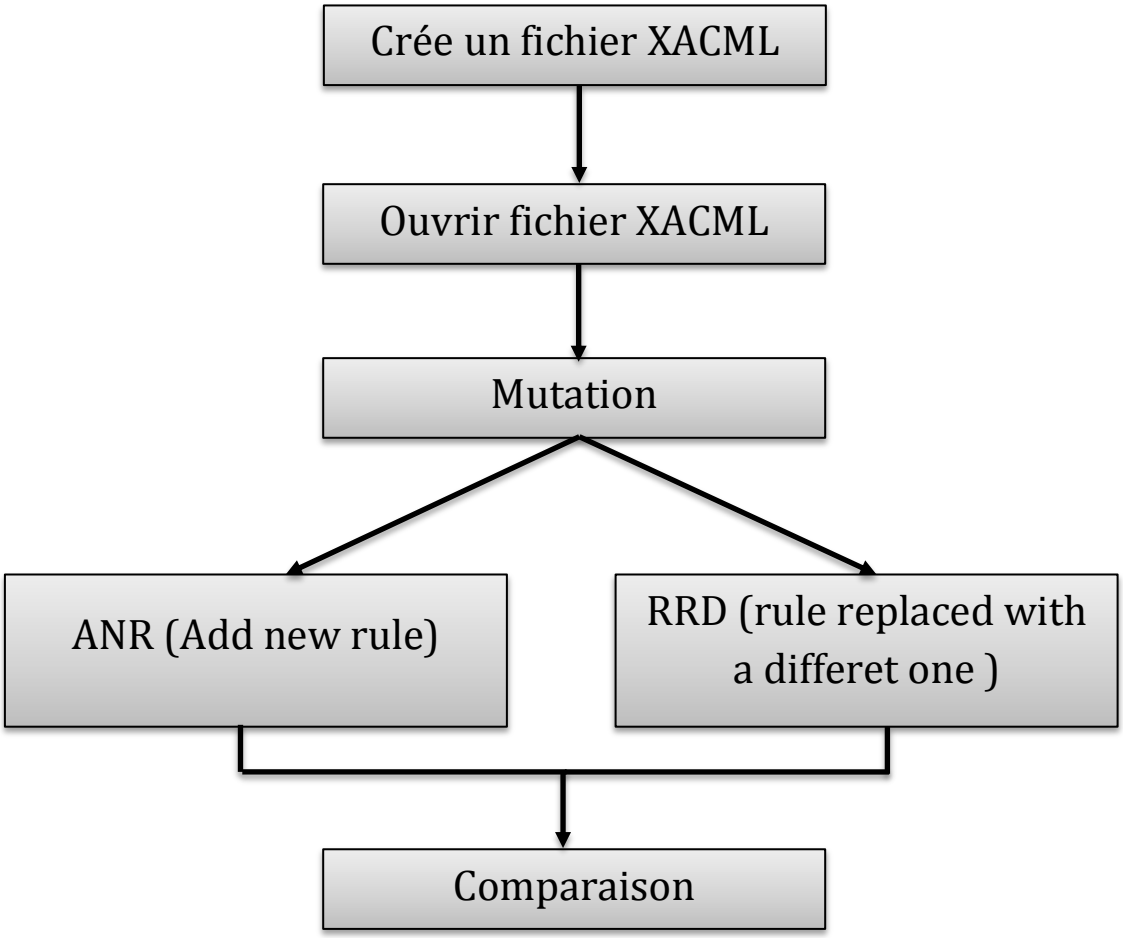


Figure 3.3 Schéma général de l'application

3.4.3 Présentation d'Application

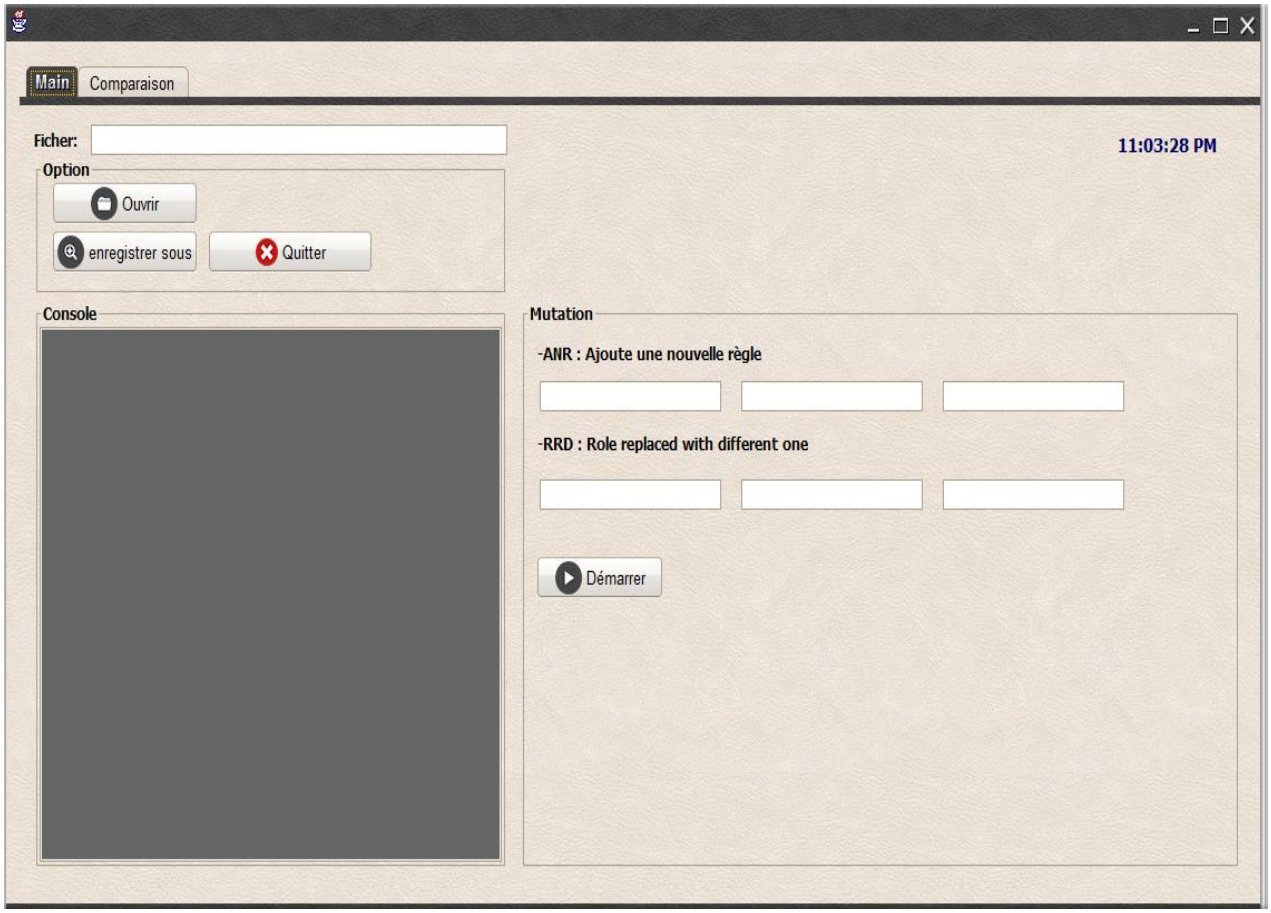


Figure 3.4 L'interface principale

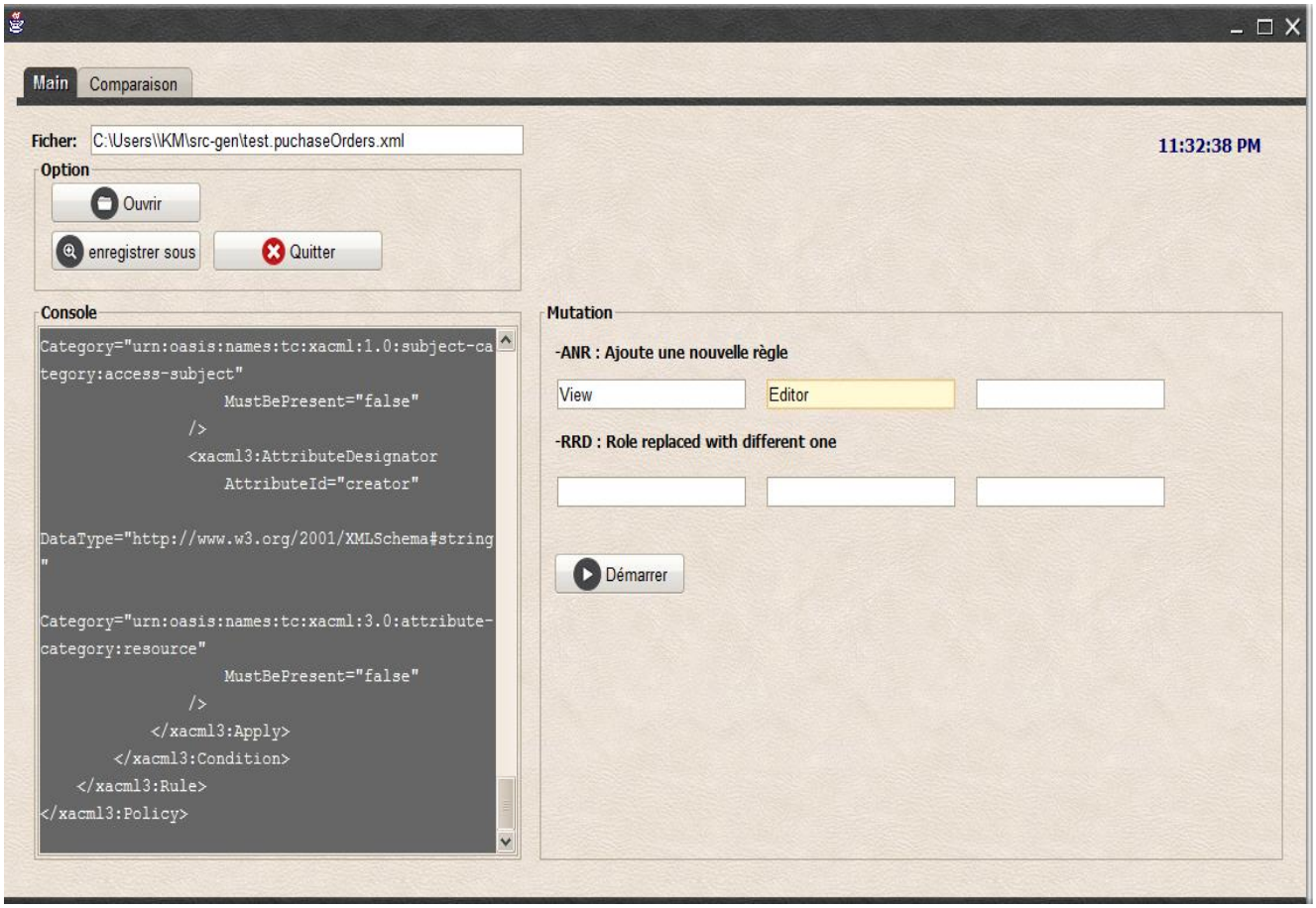


Figure 3.5 Ouvrir ficher XACML

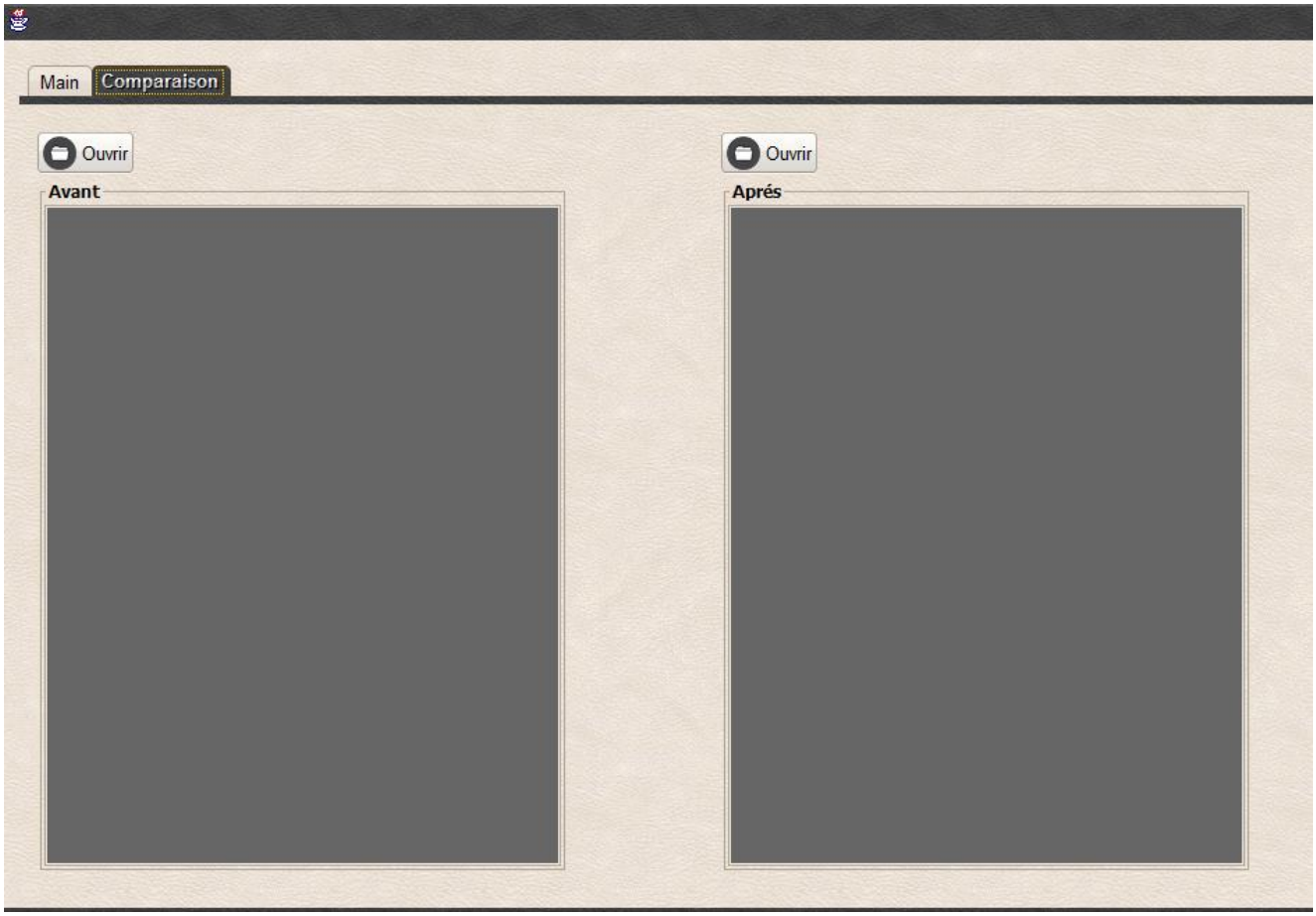


Figure 3.6 interface de Comparaison entre les fichiers XACML

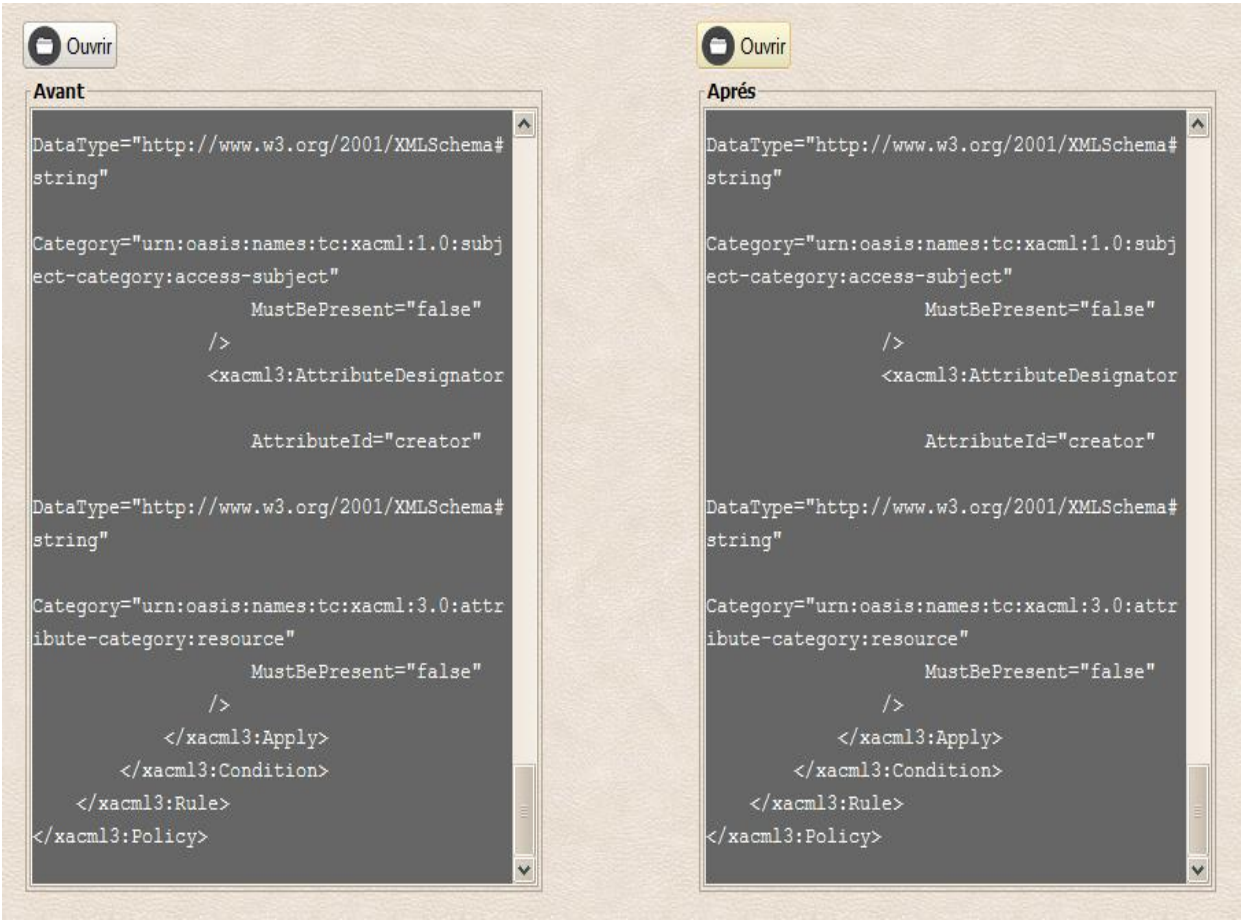


Figure 3.7 ouverture des fichiers XACML initial et mutant

```
<?xml version="1.0" encoding="UTF-8"?>
 <!--This file was generated by the ALFA Plugin for Eclipse from
Axiomatics AB (http://www.axiomatics.com).
 Any modification to this file will be lost upon recompilation of
the source ALFA file-->
<xacml3:Policy
xmlns:xacml3="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"

PolicyId="http://axiomatics.com/alfa/identifier/test.puchaseOrders"
    RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-
combining-algorithm:first-applicable"
    Version="1.0">
    <xacml3:Description />
    <xacml3:PolicyDefaults>
        <xacml3:XPathVersion>http://www.w3.org/TR/1999/REC-xpath-
19991116</xacml3:XPathVersion>
    </xacml3:PolicyDefaults>
    <xacml3:Target>
        <xacml3:AnyOf>
            <xacml3:AllOf>
                <xacml3:Match
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <xacml3:AttributeValue

DataType="http://www.w3.org/2001/XMLSchema#string">purchase
order</xacml3:AttributeValue>
                    <xacml3:AttributeDesignator
                        AttributeId="resource-type"

DataType="http://www.w3.org/2001/XMLSchema#string"

Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                        MustBePresent="false"
                    />
                </xacml3:Match>
                <xacml3:Match
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <xacml3:AttributeValue

DataType="http://www.w3.org/2001/XMLSchema#string">purchasing</xacm
l3:AttributeValue>
                    <xacml3:AttributeDesignator
                        AttributeId="department"

DataType="http://www.w3.org/2001/XMLSchema#string"

Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
subject"
                        MustBePresent="false"
                    />
                </xacml3:Match>
            </xacml3:AllOf>
        </xacml3:AnyOf>
    </xacml3:Target>
```

Figure 3.8 Exemple ficher XACML

Conclusion générale

Dans ce travail, nous nous sommes intéressés aux mécanismes de sécurité, et plus précisément à la génération des attaques pour tester la sécurité. Après une recherche dans le domaine, notre étude s'est penchée précisément sur les mécanismes de contrôle d'accès. Premièrement, nous avons illustré ces mécanismes avec ces principes de base et leurs buts d'assurer les propriétés de sécurité. Deuxièmement, nous avons donné une revue de la littérature portant sur les modèles de politiques de contrôle d'accès, pour modéliser des politiques tels que RBAC, OrBAC, DAC et MAC. Après une comparaison des langages de politiques, on a choisi le langage XACML pour exprimer ces politiques de sécurité et faciliter l'utilisation de la mutation. Troisièmement, on a utilisé les opérateurs de la mutation dans les politiques XACML pour générer des mutants. Finalement, nous avons implémenté ceci dans une application.

Bibliographie

[IKH10] IKHLASS HATTAK , ANALYSE FORMELLE DES
POLITIQUES DE SÉCURITÉ pour l'obtention du grade de maîtreès science (M.Sc.)

[SG06] Solange Ghernaouti-Hélie, Sécurité informatique et réseaux, Livre, Dunod, Février2006

[MEM10] mémoire intitulé : ANALYSE FORMELLE DES POLITIQUES DE SECURITE UNIVERSITE DU QUEBEC EN OUTAOUAIS 2010

[TAJ09] Tejeddine Mouelhi , Utilisations de la mutation pour les tests de controle d'acces dans les applications 2009

[MIC76] MichaelA.Harrison,WalterL.Ruzzo,andJereyD.Ullman. Protection in operating systems.1976.

[MAH06] MahdiMankai. Véricationetanalysedespolitiquesdecontrôled'accès: Application au langage xacml. Master's thesis, Université du Québec en Outaouais, 2006

[GIA95] Giancarlo Martella Pierangela Samarati Silvana Castano, Maria Grazia Fugini.

Database Security. Addison-Wesley & ACM Press, 1995

[UNI76] D. Elliott Bell; Leonard J. La Padula. Secure computer system : Unied exposition and multics interpretation.Proc. IEEE Computer Society Symposium on Research in Security and Privacy, pp. 215-228.

[TCW89] D. Brewer and M. Nash. The chinese wall security policy. Proc. IEEComputer Society Symposium on Research in Security and Privacy,

[LUC04]
LucaCardelli.Typesystems.http://www.eecs.umich.edu/~bchandra/courses/papers/Cardelli_Types.pdf, February 2004 consulté le 26 AVRIL 2016.

[MEG12] Michel Embe Jiague : MISE EN OEUVRE DE POLITIQUES DE CONTRÔLE D'ACCÈS FORMELLES POUR DES APPLICATIONS BASÉES SUR UNE ARCHITECTURE ORIENTÉE SERVICES '2012

[ME11] MOHAMMED ERRACHID VÉRIFICATION DES POLITIQUES XACML AVEC LE LANGAGE EVENT-B MÉMOIRE PRÉSENTÉ COMME EXIGENCE PARTIELLE DE LA MAÎTRISE EN INFORMATIQUE ' 2011

[NNEM01] Nicodemos Damianou, Naranker Dulay, Emil Lupu, Morris Sloman : The Ponder Policy Specification Language ' Department of Computing, Imperial College, '2001

[PLJS] Ponnurangam Kumaraguru,Lorrie Faith Cranor,Jorge Lobo,Seraphin B. Calo : A Survey of Privacy Policy Languages from Carnegie Mellon University and IBM Research, USA

[GH97] Gerard J. Holzmann,IEEE Transactions on Software Engineering - Special issue on formal methods in software practice 05/May/1997 p:279-295

Webographie

[NET01]
https://fr.wikipedia.org/wiki/Contr%C3%B4le_d%27acc%C3%A8s_bas%C3%A9_sur_l%27organisation 20/06/2016-22:20

[NET02]

http://www.memoireonline.com/11/15/9285/m_Conception-et-verification-de-la-coherence-dune-politique-de-securite-dans-un-reseau-local0.html 07/07/2016-00:20

[NET03]
https://yuwang.gitbooks.io/dataprotection/content/extensible_access_control_markup_language_xacml_2.html 28/07/2016-20:20

[NET 04] http://www.csee.umbc.edu/~lkagal1/rei/ 29/07/2016-21:00

[NET05] https://fr.wikipedia.org/wiki/Java_(langage) 16/08/2016-17:00

[NET06]

https://www.axiomatics.com/solutions/products/authorization-for-applications/developer-tools-and-apis/192-axiomatics-language-for-authorization-alfa.html 16/08/2016-18:00