

UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

**Détection d'incohérences et d'incomplétudes dans des politiques de
contrôle d'accès utilisant un algorithme de classification de données**

MÉMOIRE PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DU PROGRAMME DE MAÎTRISE EN SCIENCES ET TECHNOLOGIES DE
L'INFORMATION

PAR

OUSSAMA HATTAK

Octobre 2017

Jury d'évaluation

| | |
|--------------------------|------------------------|
| Président du Jury : | Dr. Karim El Guemhioui |
| Membre du Jury : | Dr. Kamel Adi |
| Directeur de recherche : | Dr. Luigi Logrippo |

Remerciement

Au premier lieu, je tiens à remercier mon directeur de recherche Monsieur Luigi Logrippo, je voudrais lui exprimer toute ma reconnaissance pour son encadrement, ses conseils, l'aide et le temps qu'il a bien voulu me consacrer.

Mes vifs remerciements s'adressent également aux membres de jury; Professeur Kamel Adi et Professeur Karim El Guemhioui.

Enfin je remercie toute personne qui a, à des degrés divers, contribuer sur le plan intellectuel, technique, moral ou encore matériel à l'achèvement de ce travail.

Table des matières

| | |
|---|-----------|
| Liste des figures | 8 |
| Liste des tableaux | 10 |
| CHAPITRE I : INTRODUCTION..... | 12 |
| 1. Introduction..... | 12 |
| 2. Motivation..... | 13 |
| 3. Contribution | 14 |
| 4. Plan de mémoire..... | 17 |
| CHAPITRE II : CONTEXTE INDUSTRIEL..... | 20 |
| 1. Introduction..... | 20 |
| 2. EEM (Embedded Entitlement Manager) | 20 |
| 2.1. Gestion d'identités | 21 |
| 2.1. Structure de politiques | 21 |
| 2.1.1. Identités..... | 22 |
| 2.1.2. Ressource | 25 |
| 2.1.3. Calendrier..... | 26 |
| 2.1.4. Politique | 27 |
| 2.1.5. Safex | 27 |
| 3. SITEMINDER..... | 29 |
| 3.1. Introduction..... | 29 |
| 3.2. Architecture de Siteminder..... | 30 |
| 3.3. Structure de politiques et règles | 33 |
| 3.3.1. Structure de politiques | 33 |
| 3.3.2. Règle..... | 34 |
| 3.3.3. Répertoire d'utilisateurs | 35 |
| 3.3.4. Structure de Règles (rules)..... | 36 |
| 3.3.5. Realm | 38 |

| | | |
|---|--|-----------|
| 3.3.6. | Ressource | 39 |
| 3.3.7. | Action | 40 |
| 3.3.8. | Calendrier (Time) | 41 |
| 3.3.9. | Active Rule (expression logique)..... | 42 |
| 3.4. | Conclusion | 46 |
| CHAPITRE III: ÉTAT DE L'ART..... | | 47 |
| 1. | Modèles de contrôle d'accès..... | 47 |
| 2. | Modèle à base de rôle RBAC (Role Based Access Control)..... | 47 |
| 2.1. | Implantation de RBAC dans le Web..... | 51 |
| 2.1.1. | Architecture user-pull | 52 |
| 2.1.2. | Architecture server-pull | 53 |
| 2.1.3. | Conclusions sur le modèle RBAC | 54 |
| 3. | Langage XACML..... | 55 |
| 3.1. | Introduction..... | 55 |
| 3.2. | Description du langage XACML | 55 |
| 3.3. | Algorithmes de combinaison..... | 59 |
| 3.4. | Architecture à la base de XACML | 62 |
| 4. | Aperçu des techniques de vérification de politiques de sécurité | 64 |
| 4.1. | Vérification des politiques via une description logique | 65 |
| 4.1.1. | Introduction..... | 65 |
| 4.1.2. | Spécification du langage | 66 |
| 4.1.3. | Détection de conflits entre les politiques de sécurité..... | 67 |
| 4.2. | Vérification des politiques de sécurités par l'outil SEMPO..... | 69 |
| 4.2.1. | Introduction..... | 69 |
| 4.2.2. | Architecture de SEMPO..... | 70 |
| 4.2.3. | Formalisation de politiques..... | 71 |
| 4.2.4. | Détection de conflits..... | 71 |
| 4.2.5. | Conclusion | 73 |
| 4.3. | Analyse des politiques de sécurité par l'outil EXAM..... | 74 |
| 4.3.1. | Introduction..... | 74 |

| | | |
|---|--|-----------|
| 4.3.2. | Architecture de l'outil EXAM | 75 |
| 4.3.3. | Analyseur de similarité de politiques (ASP)..... | 76 |
| 4.3.4. | Architecture de l'analyseur de similarité de politiques (ASP)..... | 77 |
| 4.3.5. | Conclusion | 79 |
| 5. | Algorithme de classification | 80 |
| 5.1. | Algorithme ID3 | 80 |
| 5.1.1. | Introduction..... | 80 |
| 5.1.2. | Arbre de décision | 81 |
| 5.1.3. | Entropie | 81 |
| 5.1.4. | Information de gain | 81 |
| 5.2. | Algorithme CHAID..... | 82 |
| 5.2.1. | Introduction..... | 82 |
| 5.2.2. | Tableau de calcul | 82 |
| 5.2.3. | T de Tschuprow | 83 |
| 5.3. | Algorithme C4.5 | 83 |
| 5.3.1. | Introduction..... | 83 |
| 5.3.2. | Arbre de décision | 84 |
| 5.3.3. | Algorithme de construction d'arbre..... | 85 |
| 5.4. | Étude comparative..... | 86 |
| CHAPITRE IV : FONDEMENT THEORIQUE DE NOTRE APPROCHE..... | | 91 |
| 1. | Introduction | 91 |
| 2. | Algorithme d'analyse de politiques C4.5 modifié | 91 |
| 2.1. | Incohérence ou contradiction..... | 93 |
| 2.1.1. | Incohérence directe..... | 93 |
| 2.2. | Incomplétude..... | 94 |
| 3. | Processus de vérification | 96 |
| 3.1. | Conversion de fichier de politiques | 96 |
| 3.2. | Stratégie d'analyse d'incohérence et d'incomplétude | 98 |
| 3.2.1. | Analyse d'incohérence..... | 99 |

| | |
|---|------------|
| 3.2.2. Analyse d'incomplétude | 104 |
| 4. Conclusion..... | 106 |
| CHAPITRE V : OUTIL DE DETECTION D'ANOMALIES..... | 107 |
| 1. Introduction..... | 107 |
| 2. Architecture..... | 107 |
| 3. Description de l'outil..... | 108 |
| 4. Vérification d'incohérences..... | 112 |
| 5. Vérification d'incomplétudes..... | 116 |
| CHAPITRE VI : CONCLUSION..... | 117 |
| 1. Travail accompli et contribution visée..... | 117 |
| 2. Travaux futurs..... | 119 |
| Bibliographie..... | 121 |

Liste des figures

| | |
|---|----|
| Figure 1.1 : Exemple d'un arbre de décision complet | 17 |
| Figure 2.1: Page d'authentification d'EEM..... | 22 |
| Figure 2.2 : Interface principale d'EEM..... | 23 |
| Figure 2.3 : Page de création d'identités..... | 24 |
| Figure 2.4 : Informations des identités dans EEM..... | 24 |
| Figure 2.5 : Interface pour créer une ressource dans EEM..... | 25 |
| Figure 2.6 : Interface pour définir un calendrier dans EEM | 26 |
| Figure 2.7 : Règles de contrôle d'accès dans EEM..... | 27 |
| Figure 2.8 : Exemple de fichier de politiques | 28 |
| Figure 2.9 : Architecture de Siteminder (CA Technologies, 2009)..... | 30 |
| Figure 2.10 : Scénario d'authentification et d'autorisation (Swafford, 2009)..... | 32 |
| Figure 2.11 : Structure d'une politique (Netegrity-Inc, 1997)..... | 33 |
| Figure 2.12 : Interface pour créer une politique | 34 |
| Figure 2.13 : Exemple d'un répertoire d'utilisateurs | 36 |
| Figure 2.14 : Structure d'une règle (Netegrity-Inc, 1997)..... | 37 |
| Figure 2.15 : Interface pour créer une règle..... | 37 |
| Figure 2.16 : Exemple de realm (Netegrity-Inc, 1997)..... | 38 |
| Figure 2.17 : Interface pour créer un realm | 39 |
| Figure 2.18 : Interface pour définir les actions | 41 |
| Figure 2.19 : Interface pour définir les intervalles de temps..... | 42 |
| Figure 2.20 : Interface pour définir une expression logique..... | 43 |
| Figure 2.21 : Exemple de fichier de politiques | 45 |
| Figure 3.1 : Affectation des permissions dans RBAC (Sandhu R. S., Coyne, Feinstein, & Youman, 1996) | 48 |
| Figure 3.2 : Famille des modèles RBAC (Sandhu R. S., Coyne, Feinstein, & Youman, 1996).... | 50 |
| Figure 3.3 : Présentation de RBAC3 (Sandhu R. S., Coyne, Feinstein, & Youman, 1996)..... | 51 |
| Figure 3.4 : Architecture User-pull (Park & Sandhu, 2001)..... | 53 |
| Figure 3.5 : Architecture server-pull (Park & Sandhu, 2001)..... | 54 |
| Figure 3.6 : Structure d'une politique XACML..... | 56 |
| Figure 3.7 : Structure d'une cible | 56 |
| Figure 3.8 : Exemple d'une politique XACML..... | 62 |
| Figure 3.9 : Architecture XACML (Keleta, Eloff, & Venter, 2005) | 63 |
| Figure 3.10 : Autorisation et hiérarchie des attributs (Huang, Huang, & Liu, 2009) | 68 |
| Figure 3.11 : Interface de SEMPO (Layouni, 2010)..... | 69 |

| | |
|---|-----|
| Figure 3.12 : Architecture de l'outil SEMPO (Layouni, 2010) | 70 |
| Figure 3.13: Architecture de l'outil EXAM (Lin, Rao, Bertino, Li, & Lobo, 2010) | 75 |
| Figure 3.14 : Architecture du module ASP (Lin, Rao, Bertino, Li, & Lobo, 2010) | 77 |
| Figure 3.15 : Exemple d'un CMTBDD (Lin, Rao, Bertino, Li, & Lobo, 2010)..... | 78 |
| Figure 3.16 : arbre de décision généré par C4.5 | 89 |
| Figure 3.17 : arbre de décision généré par CHAID | 89 |
| Figure 3.18 : arbre de décision généré par ID3 | 90 |
| | |
| Figure 4.1 : Représentation de politique par EEM | 97 |
| Figure 4.2 : Arbre de décision final | 104 |
| Figure 4.3 : Arbre de décision incomplet..... | 106 |
| | |
| Figure 5. 1 : Architecture principale de l'outil de vérification | 108 |
| Figure 5. 2: Interface principale de l'outil de vérification | 109 |
| Figure 5.3 : Fichier XML de politiques | 111 |
| Figure 5. 4 : Ensemble de politiques en format CSV | 112 |
| Figure 5. 5 : Résultat d'analyse d'incohérences | 113 |
| Figure 5. 6 : Graphe global de politiques de contrôle d'accès..... | 114 |
| Figure 5. 7 : Graphe local de politiques de contrôle d'accès | 115 |
| Figure 5. 8 : Résultats de vérification d'incomplétudes..... | 117 |

Liste des tableaux

| | |
|---|-----|
| Tableau 1.1 : Exemple de politiques en format CSV..... | 16 |
| Tableau 2.1: Exemple de ressources dans Siteminder (Netegrity-Inc, 1997)..... | 40 |
| Tableau 2.2 : Exemple d'actions dans Siteminder (Netegrity-Inc, 1997)..... | 40 |
| Tableau 3.1 : Exemple d'un fichier de données..... | 87 |
| Tableau 4.1 : Exemple de conversion de politique | 98 |
| Tableau 4.2 : Exemple de règles en format CSV | 99 |
| Tableau 4.3 : Ensemble de règles en format CSV | 105 |

Résumé

Dans les entreprises, l'utilisation des ressources est régie par des systèmes de contrôle d'accès. Ces systèmes contiennent des ensembles de politiques définissant les droits que les utilisateurs peuvent avoir sur les ressources. Un système de contrôle d'accès peut contenir un grand nombre de politiques, sujettes à des modifications fréquentes. L'introduction de politiques qui causent des anomalies, telles que des incohérences et des incomplétudes, peut rendre un système vulnérable ou non fiable. Ces anomalies peuvent être exploitées par des utilisateurs pour compromettre la sécurité ou peuvent conduire à un comportement indésirable du système. La vérification de ces politiques est donc une étape nécessaire pour assurer la sécurité d'une entreprise. Dans ce mémoire, nous proposons un mécanisme de vérification de politiques basé sur une modification de l'algorithme d'exploration de données C4.5. En utilisant cet algorithme, nous allons implémenter un outil qui permettra la détection des anomalies dans les politiques d'un système de contrôle d'accès industriel, IAM de la compagnie CA. Notre travail comporte deux étapes principales :

1. Conversion des politiques IAM dans un format approprié pour l'algorithme C4.5.
2. Modification de l'algorithme C4.5 pour notre but, qui est la détection des anomalies.

CHAPITRE I : INTRODUCTION

1. Introduction

La sécurité d'information a trois objectifs principaux (Sandhu, 1993) :

- Confidentialité : protéger les informations sensibles contre les accès non autorisés
- Intégrité : empêcher toute sorte de modification (ajout, suppression, etc) d'une ressource par un utilisateur non habilité
- Disponibilité : garantir l'accessibilité des ressources lorsqu'un utilisateur légitime en a besoin

La préservation de la confidentialité des ressources au sein des applications, qu'il s'agisse des données à caractère personnel ou des systèmes d'informations électroniques, est devenue une fonctionnalité indispensable pour la majorité des entreprises qui contiennent multiples environnements et plusieurs utilisateurs. Dans ce contexte, la mise en place des mécanismes de contrôle d'accès est devenue une priorité majeure dans les entreprises.

Les mécanismes de contrôle d'accès aident à limiter l'exécution de codes malveillants ou d'actions non autorisées. Un système de contrôle d'accès doit protéger les données contre toute sorte d'affichage non autorisé, modification ou copie des informations. Il prend ses décisions conformément à un ensemble de politiques qui définissent les droits de chaque utilisateur par rapport aux ressources du système. Un droit représente une action qu'un utilisateur peut appliquer sur des ressources.

Au fil des ans, une variété de modèles de contrôle d'accès ont été développés afin d'assurer la bonne utilisation des ressources et des services dans un système, chacun

d'eux étant conçu pour aborder différents aspects de sécurité (intégrité, disponibilité et confidentialité dans différents contextes), selon des principes différents.

L'autorisation et le contrôle d'accès sont des termes souvent interchangeables. L'autorisation est l'opération de vérification pour vérifier si un utilisateur a le droit d'accéder à une ressource particulière ou pour effectuer une action spécifique, tout en supposant que l'utilisateur ait réussi l'étape de l'authentification. L'autorisation est dépendante des politiques et des listes de contrôle qui sont prédéfinies par l'administrateur de l'application ou par le propriétaire des données.

Dans la suite de ce travail, nous allons voir que les ensembles de politiques de contrôle d'accès peuvent être incomplets ou incohérents. Ces incomplétudes ou incohérences peuvent causer des failles dans les systèmes de contrôle d'accès ; leur identification est le sujet de ce mémoire.

2. Motivation

Notre projet de recherche a été effectué en collaboration avec «CA Technologies» et «CA Labs». La compagnie CA Technologies commercialise des outils qui permettent la définition et la gestion des politiques de contrôle d'accès, ainsi que la protection des ressources et la prise des décisions pour les requêtes envoyées par un utilisateur pour utiliser une ressource. L'outil que nous avons utilisé pour effectuer notre recherche est IAM, voir chapitre II pour plus de détails.

IAM est un logiciel qui permet à un administrateur de définir des politiques de contrôle d'accès. Cependant, il ne traite pas le problème mentionné dans la section précédente, le problème d'incohérence et d'incomplétude. Afin de corriger ce problème, nous avons pensé à concevoir un outil complémentaire qui permet, d'un côté, la vérification de l'ensemble des politiques de contrôle d'accès défini par l'administrateur et, de l'autre côté, la détection des incohérences et des incomplétudes, ainsi que d'avertir l'administrateur de l'existence de telles anomalies.

Notre projet de recherche est basé sur un algorithme d'exploration de données (C4.5), voir chapitre IV pour plus d'explications.

3. Contribution

Un système de contrôle d'accès peut permettre à un administrateur de définir des politiques positives, qui permettent l'accès aux ressources (données, fichiers, programmes, etc.), et des politiques négatives, qui interdisent l'accès aux ressources. Toutefois, plus les organisations grandissent, plus le nombre de politiques de sécurité augmente dans les systèmes de contrôle d'accès. Par conséquent, des problèmes d'incohérences et d'incomplétudes peuvent se présenter, ce qui met la sécurité de ces systèmes en danger.

Une incohérence est une situation dans laquelle deux politiques différentes donnent des résultats contradictoires dans le même ensemble de politiques de contrôle d'accès.

Prenons comme exemple les deux politiques suivantes :

- L'utilisateur Alice a le droit d'écrire dans le fichier des notes <F_note>
- L'utilisateur Alice n'a pas le droit d'écrire dans le fichier des notes <F_note>

Dans cette situation, on parle d'une incohérence directe puisque l'utilisateur Alice, dans le même contexte, a une autorisation positive et négative pour écrire dans le fichier des notes <F_note>.

Supposons qu'un système de contrôle d'accès, dans un hôpital, contient les deux politiques suivantes :

- Une infirmière qui travaille durant le jour peut accéder aux dossiers des patients de 8H01 à 17H59

- Une infirmière qui travaille durant la nuit peut accéder aux dossiers des patients de 18H01 à 7H59

Dans cet exemple, il n'existe pas de règles qui contrôlent l'accès aux dossiers des patients à 8h00 et 18h00 exactes, ce qui génère une incomplétude dans le système.

Les incohérences et incomplétudes que nous venons de donner comme exemples sont faciles à voir, car nous avons présenté les politiques impliquées côte à côte. Or, en pratique, il est possible d'avoir des milliers de politiques de contrôle d'accès exprimées dans un ordre quelconque.

Dans ce mémoire, nous développons un outil, complémentaire à IAM, qui est capable de détecter certains types d'incohérences et incomplétudes et qui peut donc améliorer la fiabilité aux systèmes de contrôle d'accès. Notre outil permet à un administrateur d'entreprise, responsable de la sécurité et la gestion de la base de données des politiques, de détecter et de corriger les incohérences et les incomplétudes dans des ensembles de politiques de contrôle d'accès.

La procédure de détection des anomalies dans notre outil est basée sur deux étapes essentielles ; la conversion et l'analyse des politiques comme présenté sur la figure 1.2.

Conversion : Afin de détecter les anomalies dans un ensemble de politiques de contrôle d'accès, on procède premièrement à la conversion de politiques générées par l'outil IAM vers un format lisible par notre outil. Cette procédure contient deux parties.

La première partie est l'extraction des politiques de contrôle d'accès définies par l'outil IAM, qui peuvent être générées dans un format XML.

La deuxième partie consiste à la conversion des données XML vers le format CSV (*Comma-separated values*). Ce dernier permet de présenter les politiques sous forme d'un tableau qui contient le nom des attributs (sujet, ressource, permission, etc.) et les valeurs de ces attributs.

Analyse : cette étape représente le cœur de notre outil. Elle est basée sur une modification de l'algorithme d'exploration de données C4.5. Après avoir converti l'ensemble des politiques de contrôle d'accès, l'outil procède d'abord à la création d'un arbre de décision complet d'après les données converties. Puis, l'outil procède à l'analyse des politiques qui sont représentées par des branches au niveau de l'arbre de décision, pour pouvoir détecter l'existence de telles anomalies. Une branche est une liaison entre un nœud racine et un nœud feuille.

Prenons comme exemple le tableau suivant.

| Sujet | Action | Ressource | Permission |
|-------|--------|-----------|------------|
| Alice | écrire | F_note1 | permis |
| Bob | écrire | F_note1 | permis |
| Alice | écrire | F_note1 | interdit |
| Mark | lire | F_note2 | permis |

Tableau 1.1 : Exemple de politiques en format CSV

Dans cet exemple, nous avons deux classes de politiques dont trois sont positives, « permis », et une négative, « interdit ».

Une permission positive signifie que l'utilisateur a le droit d'exécuter une action sur une ressource. Une permission négative signifie que l'utilisateur n'a pas le droit d'exécuter une action sur une ressource.

Après avoir appliqué l'algorithme C4.5 modifié, on obtient un arbre de décision complet qui contient tous les attributs ainsi que les valeurs de chaque attribut. Dans cet arbre, les nœuds représentent les attributs et les feuilles représentent les décisions associées aux politiques. Chaque règle est représentée par un ensemble de nœuds et d'arcs, comme le montre le graphe suivant.

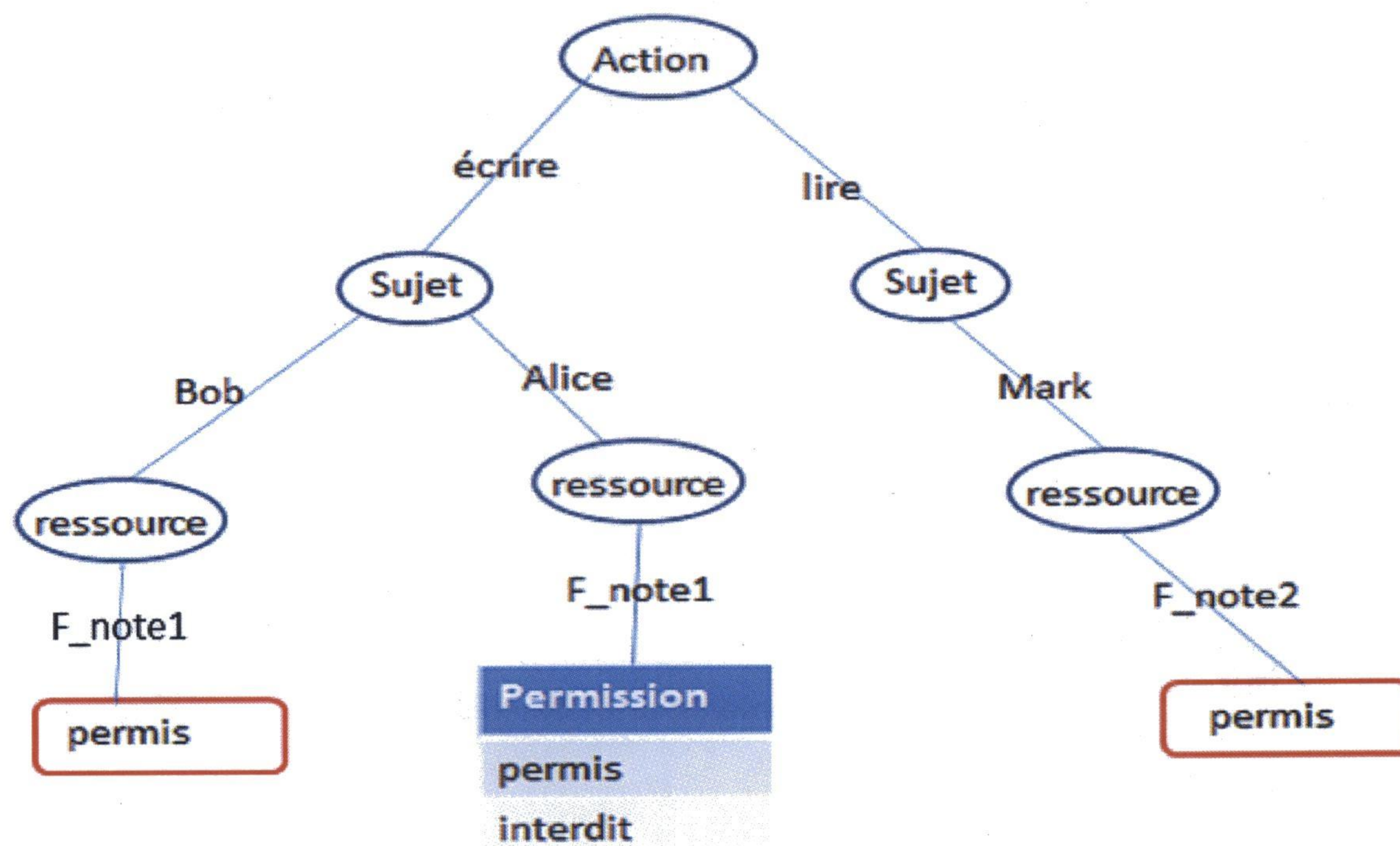


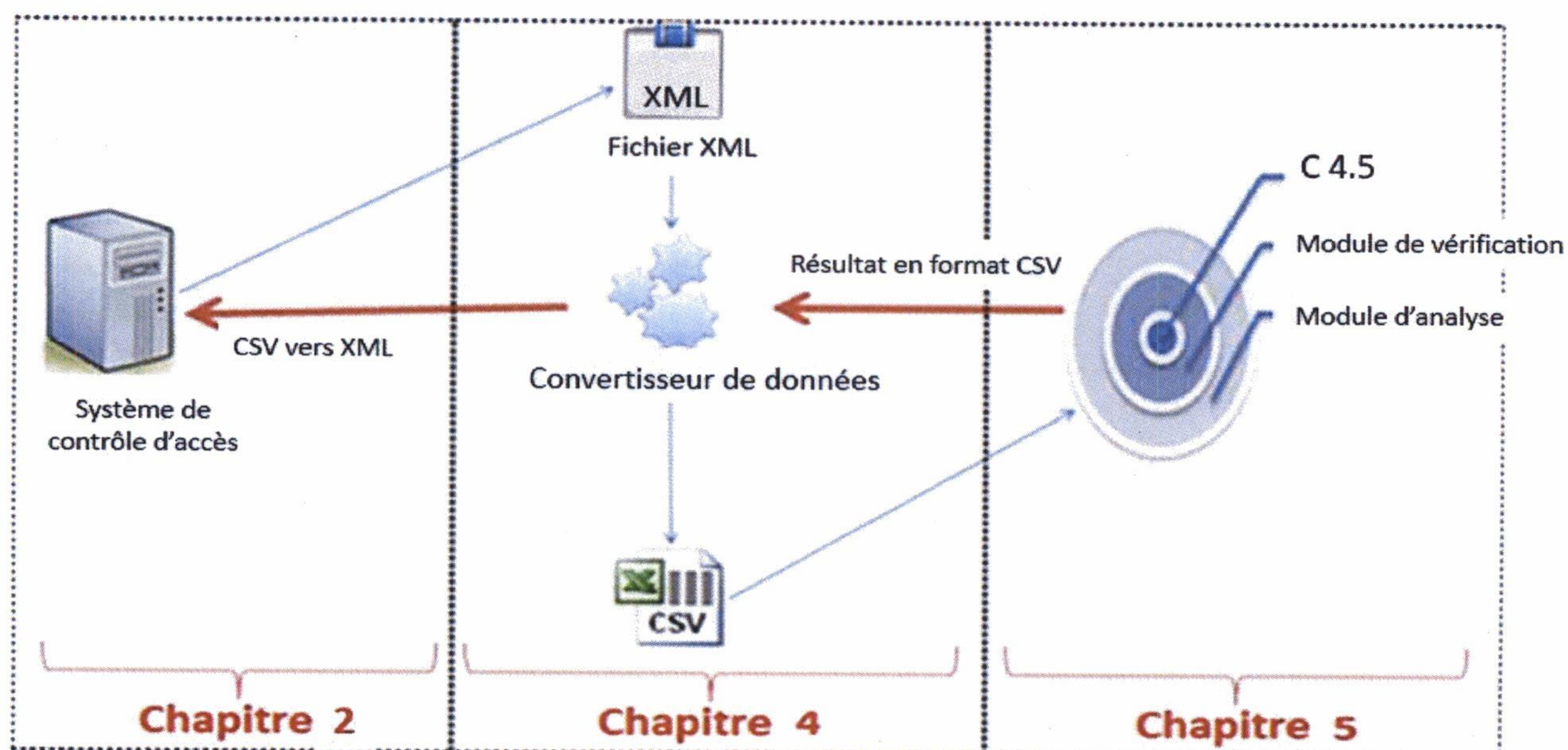
Figure 1.1 : Exemple d'un arbre de décision complet

Après l'analyse de cet arbre, on constate clairement l'existence d'une incohérence au niveau des politiques qui donnent à l'utilisateur Alice, dans le même contexte, une permission positive qui permet l'écriture dans le fichier F_note1 et une permission négative qui interdit l'écriture dans le même fichier.

Note terminologique : dans quelques langages et systèmes de contrôle d'accès, les *règles* sont appelées *politiques*. Cette terminologie est celle utilisée dans le système IAM qui fait l'objet de ce mémoire. Dans d'autres langages, tels que XACML, les *politiques* de contrôle d'accès sont des ensembles de *règles*. Dans ce mémoire, nous chercherons à être cohérents avec la terminologie des langages et des systèmes dont nous parlerons de temps à autre. Ceci devrait être clair dans le contexte de notre travail et ne devrait pas occasionner des confusions.

4. Plan de mémoire

Dans cette section, nous présentons le plan de notre mémoire ainsi que le contenu de chacun de ces chapitres. La figure 1.2 montre les différentes étapes qui constituent le processus de vérification ainsi que les chapitres qui décrivent chacune de ces étapes.



Le chapitre 2 présente les outils IAM et Siteminder de la compagnie CA Technologies, sur lesquels notre recherche a été effectuée. Ce chapitre décrit les différentes composantes de ces outils, ainsi que la procédure pour définir des politiques via une interface graphique.

Le chapitre 3 présente une variété de modèles de contrôle d'accès qui existent dans la littérature, tels que les modèles de contrôle d'accès à base de rôle RBAC (Rôle Based Access Control). Dans ces modèles, le contrôle d'accès est appliqué sur des rôles qui sont affectés à un ou plusieurs utilisateurs. Dans ce même chapitre, on présente aussi quelques méthodes et outils qui ont été développés pour détecter les conflits entre les politiques de sécurité. Nous introduisons également l'algorithme de classification C4.5 qui a été présenté à l'université de Sydney par le professeur Ross Quinlan en 1993 (Quinlan, C4.5: Programs for Machine Learning, 1993). De plus, nous présentons les

étapes qui permettent la construction d'un arbre de décision ainsi qu'une étude comparative avec d'autres algorithmes de classification de données.

Le chapitre 4 présente la proposition de notre recherche qui décrit le processus de détection des anomalies. On commence d'abord par les définitions formelles des anomalies qui peuvent être détectées telles que l'incohérence et l'incomplétude. Ensuite, on présente le processus de vérification d'un ensemble de politiques de contrôle d'accès. Ce processus comporte, en premier lieu, les étapes qui permettent la conversion de l'ensemble des politiques de sécurité vers le format CSV (*Comma-separated values*) qui est supporté par notre outil et, par la suite, on discute notre stratégie d'analyse des anomalies afin de générer un rapport à l'administrateur de politiques de contrôle d'accès.

Le chapitre 5 présente l'outil de détection d'anomalies que nous avons développé durant notre recherche. Cet outil est basé sur une version modifiée de l'algorithme C4.5 et permet à un administrateur de vérifier la cohérence et la complétude d'un ensemble de politiques de contrôles d'accès.

Dans la suite de ce mémoire, les notions suivantes seront utilisées :

- **Sujet** : une entité active, correspondant à un processus qui s'exécute pour le compte d'un utilisateur. Un utilisateur peut être une personne physique ou une personne virtuelle représentant un ensemble de programmes ou de processus.
- **Objet** : une entité considérée comme passive qui contient ou reçoit des informations.
- **Action** : un événement qui permet à une entité active d'exécuter une opération sur une entité passive.

CHAPITRE II : CONTEXTE INDUSTRIEL

1. Introduction

Depuis l'arrivée de la World Wide Web (WWW) dans le monde des entreprises, l'utilisation de l'internet a augmenté de façon spectaculaire. De ce fait, plus les entreprises entrent dans le monde des affaires électroniques, plus elles doivent se préoccuper de défis comme l'attraction et la satisfaction des clients et des fournisseurs. La façon dont ces défis peuvent être surmontés consiste, entre autres, à personnaliser les services et les applications pour chaque utilisateur afin de répondre aux besoins individuels et d'assurer la sécurité des transactions. La compagnie CA Technologies offre ainsi une suite de logiciels (Embedded Entitlement Manager, Siteminder, etc.) qui proposent des solutions permettant de résoudre certains problèmes auxquels sont confrontés les environnements d'affaires sur le Web.

CA Technologies est une compagnie multinationale qui propose un large ensemble de logiciels dans différents domaines. Cet ensemble inclut des outils destinés à la gouvernance informatique, la gestion et l'administration des systèmes d'information, la gestion de l'infrastructure et des opérations, ainsi que la gestion de la sécurité. Cette dernière comporte la gestion des identités et des accès et c'est sur cela que nous avons mettons l'accent dans notre travail.

Les figures présentées dans ce chapitre sont toutes des figures obtenues par l'utilisation des outils de la compagnie CA.

2. EEM (Embedded Entitlement Manager)

eTrust EEM est un outil qui aide les entreprises à gérer en toute sécurité les identités, les ressources et les politiques de contrôle d'accès dans les systèmes. Ce logiciel permet aux administrateurs d'intégrer facilement, dans les applications de leurs entreprises, les aspects d'authentification, d'autorisation et d'audit de sécurité. Il permet de traiter ces aspects à travers une interface graphique simple.

Dans ce qui suit, nous donnons un aperçu des principales caractéristiques de l'outil EEM et de la façon dont il permet de construire des politiques de base. Compte tenu de la quantité considérable d'options et de configurations offertes par EEM, nous ne discuterons que de certaines en particulier.

2.1. Gestion d'identités

EEM permet aux administrateurs de contrôler leur environnement d'application en assignant aux identités (i.e. en général, aux usagers) les accès nécessaires aux ressources du système et de définir les actions qu'elles peuvent exécuter. La gestion des identités est basée sur deux aspects principaux. Le premier est une base d'identités interne qui permet à celles-ci d'utiliser les services d'authentification et d'autorisation à l'aide de l'application SDK eTrust IAM. Seules les identités légitimes peuvent accéder aux applications d'EEM. Le deuxième est une base d'identités externe compatible avec l'outil EEM telle que Microsoft Active Directory, Novell eDirectory, eTrust Admin. Ces systèmes permettent de sauvegarder les ensembles d'identités et leurs informations, alors que les identités des utilisateurs individuels peuvent être gérées directement par l'outil EEM.

2.1. Structure de politiques

Les politiques de sécurité dans l'outil EEM sont structurées sur une approche orientée objet. Une politique de contrôle d'accès est une combinaison de trois objets principaux : soit identité (utilisateur, groupe), ressource et action, auxquels s'ajoutent par la suite certains objets secondaires tels que le calendrier de temps, etc.

2.1.1. Identités

L'outil EEM offre aux administrateurs, après une connexion réussie, une interface qui contient un lien vers la création des identités et des politiques, voir figure 2.1.

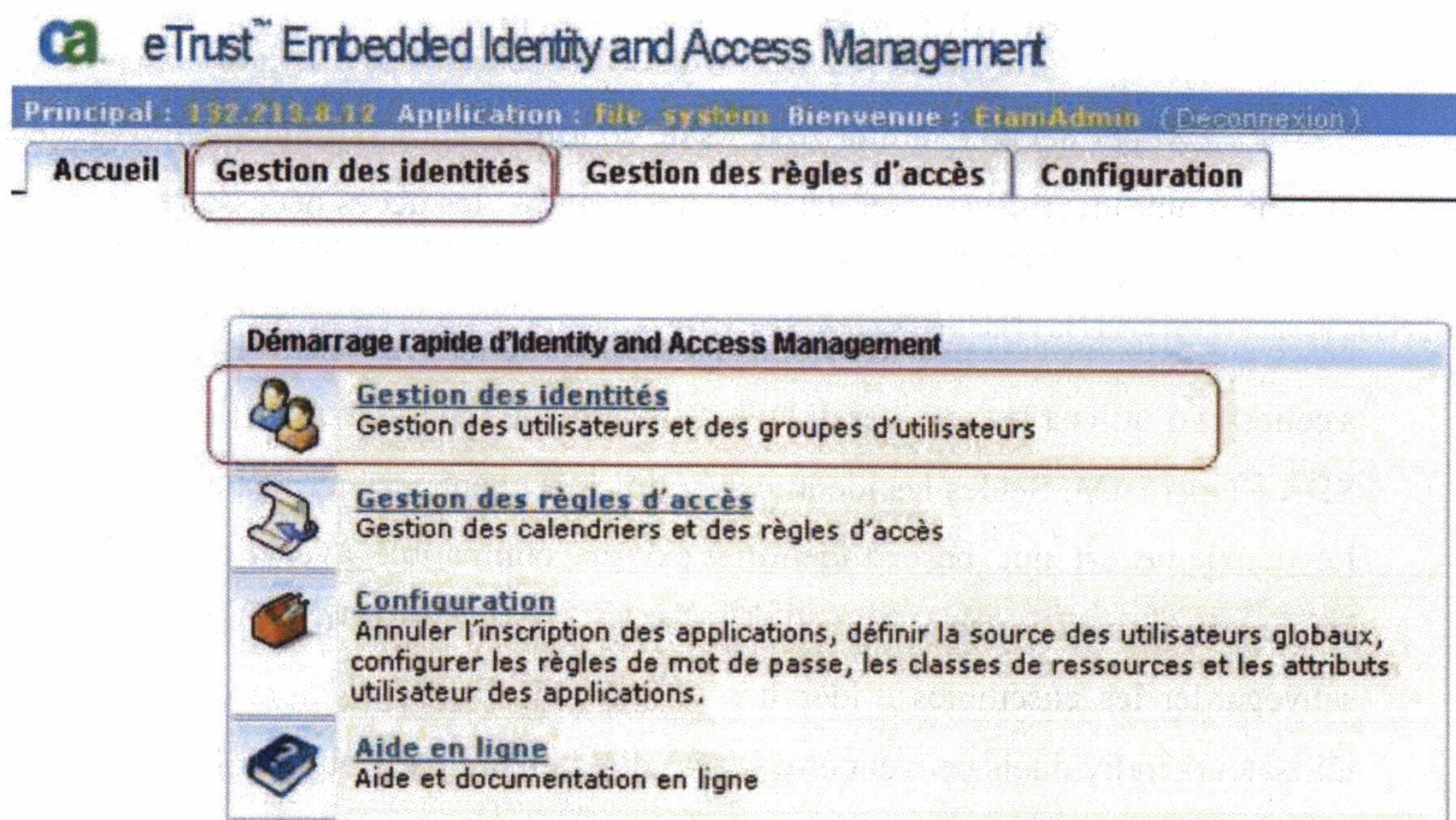


Figure 2.1: Page d'authentification d'EEM

Pour créer une identité au sein de l'outil EEM, nous choisissons le lien «gestion des identités». En cliquant sur ce lien, EEM affichera la page de création d'identités suivante.

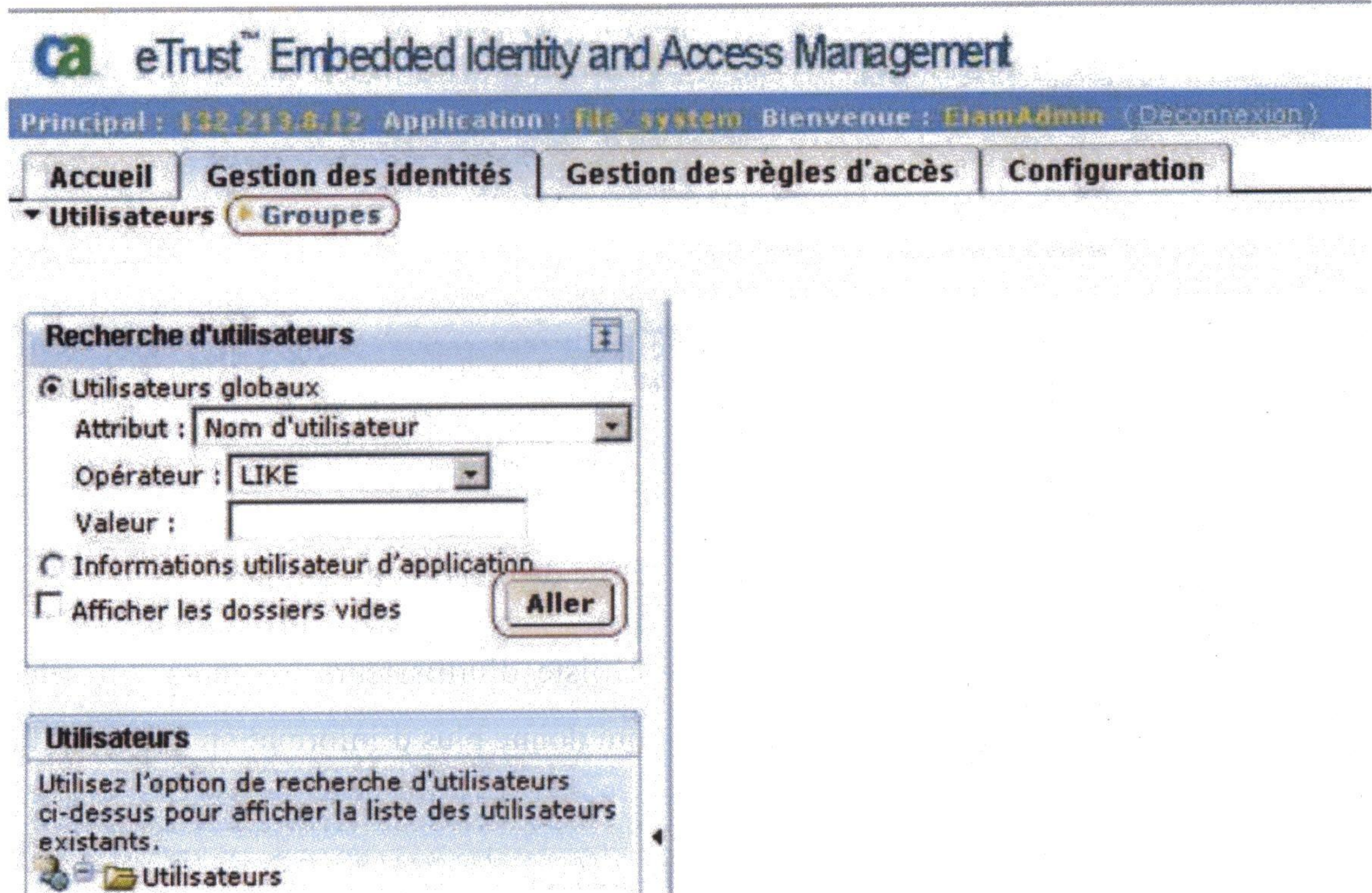


Figure 2.2 : Interface principale d'EEM

À la figure 2.2, nous voyons l'interface qui permet de gérer les groupes d'utilisateurs. Sur cette interface, nous pouvons créer un groupe d'utilisateurs en cliquant sur le bouton «Groupes». L'outil EEM donne aussi la possibilité de vérifier les utilisateurs existants dans le système, par différentes méthodes de recherche, en cliquant sur le bouton «Aller». L'outil IAM utilise des opérateurs logiques pour manipuler les groupes d'utilisateurs, par exemple la figure 2.2 montre l'opérateur «LIKE» qui signifie l'égalité à la valeur spécifiée dans le champ *Valeur*.

Après avoir cherché les utilisateurs qui existent dans le système, on obtient la figure 2.3 qui montre les utilisateurs existants ainsi que l'interface pour créer un nouvel utilisateur.

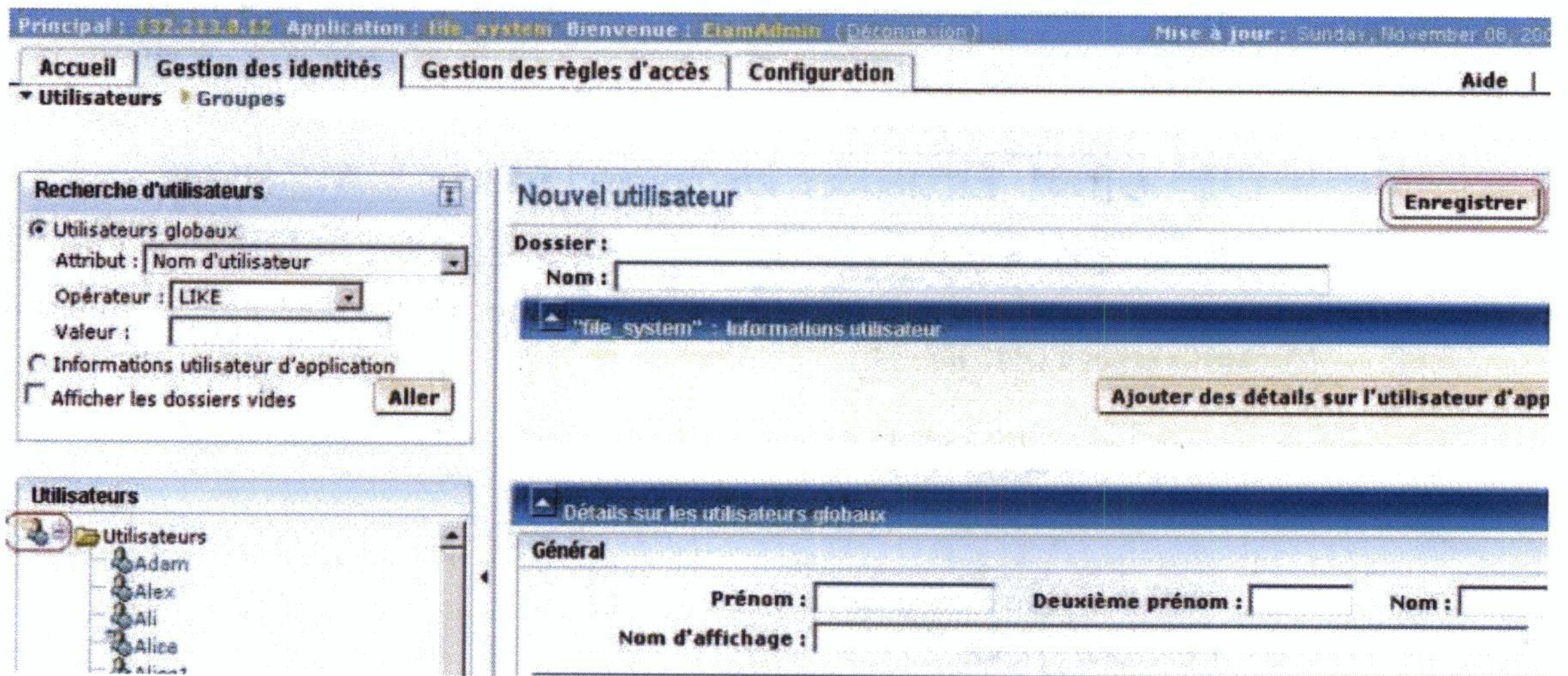


Figure 2.3 : Page de création d'identités

Sur la figure 2.3 nous voyons la liste d'utilisateurs existants. En sélectionnant l'utilisateur Bob, on obtient la figure 2.4 qui donne plus d'information sur cet utilisateur.

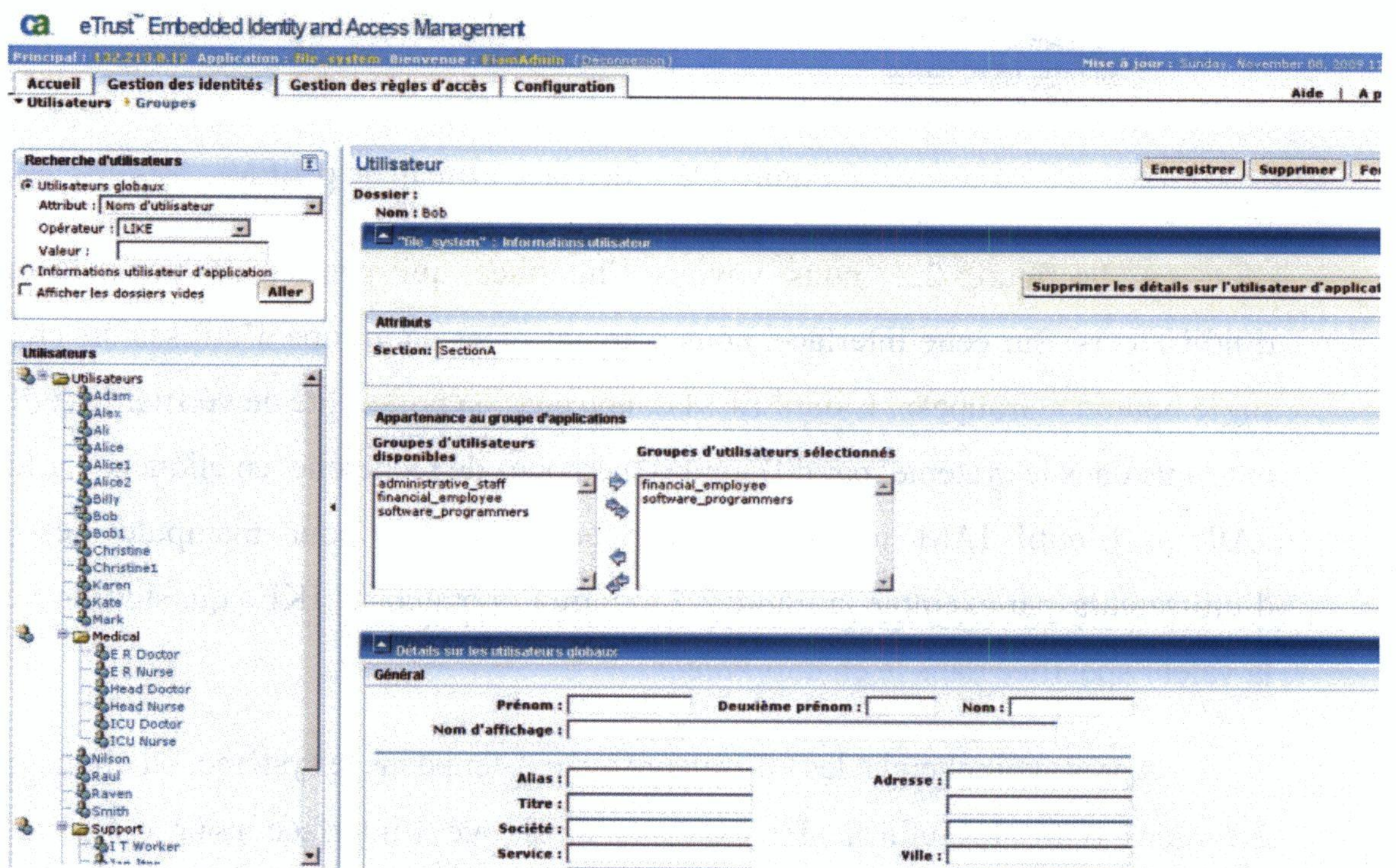


Figure 2.4 : Informations des identités dans EEM

Dans cette boîte de dialogue, on peut voir les informations de l'utilisateur Bob, telles que les groupes auxquels il appartient. Dans cet exemple, l'utilisateur Bob appartient aux groupes «financial_employees» et «software_programmers», mais pas au groupe «administrative staff».

2.1.2. Ressource

Les ressources sont les objets protégés dans un système, par exemple des bases de données ou des fichiers.

Pour créer une nouvelle ressource dans le système, nous devons choisir l'onglet « Configuration ». Cet onglet nous montre les applications existantes dans le système ainsi que les différents attributs qui permettent de créer une nouvelle instance de ressource. Par exemple, dans l'interface de la figure 2.5, nous créons la ressource «Financial_folder» et les actions associées à celle-ci, qui sont «Read» et «Write». Il est aussi possible de définir des attributs pour la ressource en cliquant sur le bouton «Ajouter une classe de ressource».

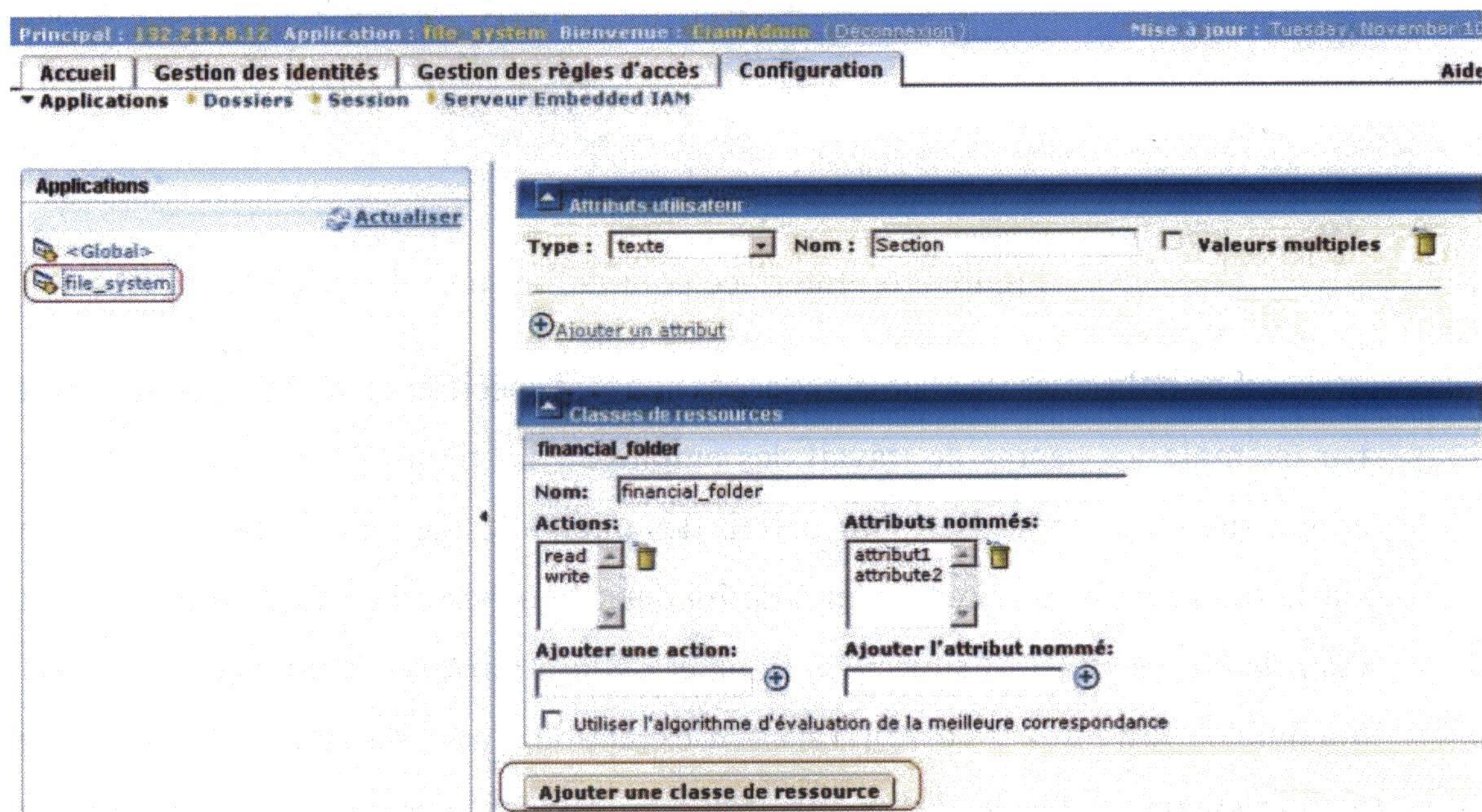


Figure 2.5 : Interface pour créer une ressource dans EEM

2.1.3. Calendrier

Dans cette section, nous montrons comment ajouter la contrainte de temps via l'interface graphique de l'outil EEM. Cet attribut est parmi les éléments nécessaires pour contrôler l'accès aux ressources selon des périodes de temps spécifiques.

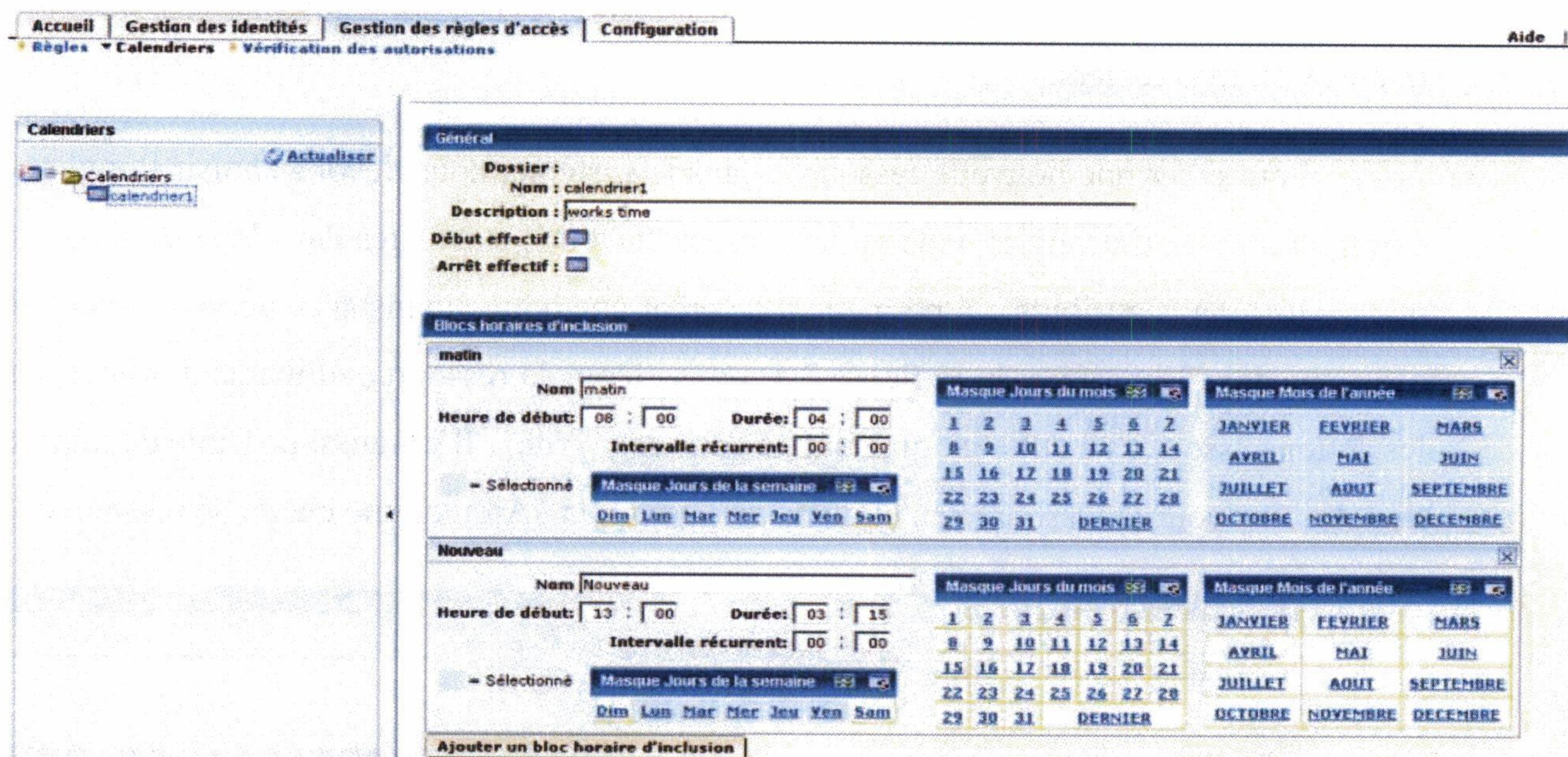


Figure 2.6 : Interface pour définir un calendrier dans EEM

Les politiques de contrôle d'accès peuvent spécifier des critères basés sur le temps, par exemple les heures de travail des employés. La figure 2.6 montre l'interface qu'un administrateur peut utiliser pour définir des intervalles de temps. Dans l'exemple illustré sur la figure 2.6, l'administrateur a défini deux tranches d'heures de travail ; le matin qui dure de 8h jusqu'à 12h situé sous l'onglet <Blocs horaires d'inclusion> et l'après-midi qui dure de 13h jusqu'à 16:15h. En cas d'exception, l'administrateur a toujours l'option de définir un intervalle de temps spécifique en cliquant sur le bouton <Ajouter un bloc horaire d'inclusion>.

2.1.4. Politique

Dans les sections précédentes, nous avons présenté les différents éléments qui constituent les politiques de contrôle d'accès. Une politique doit avoir un nom unique dans le système. Les politiques sont assignées aux utilisateurs et elles définissent leur comportement dans le système et leurs droits d'accès à une application ou à un groupe d'applications particulier.

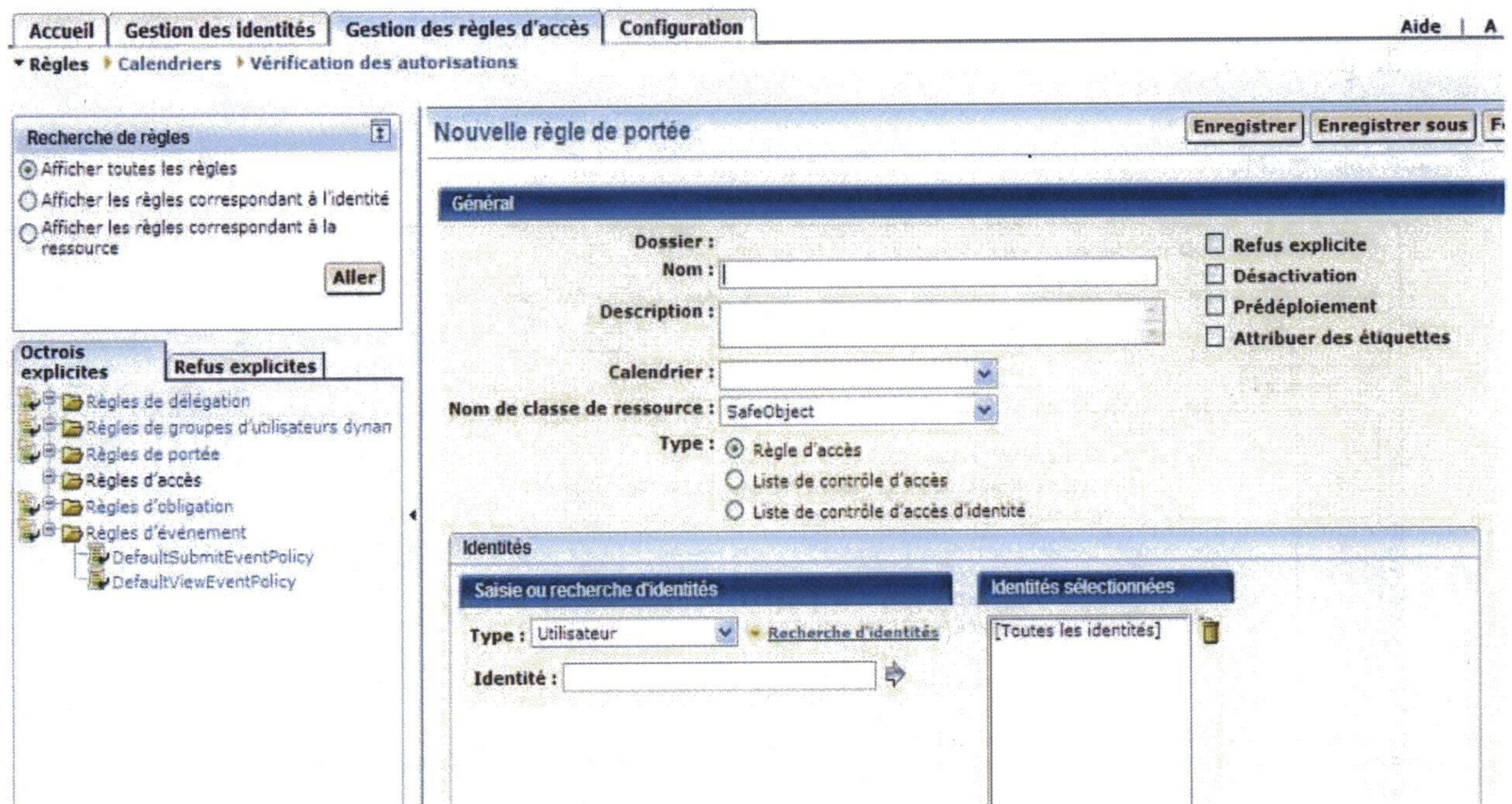


Figure 2.7 : Règles de contrôle d'accès dans EEM

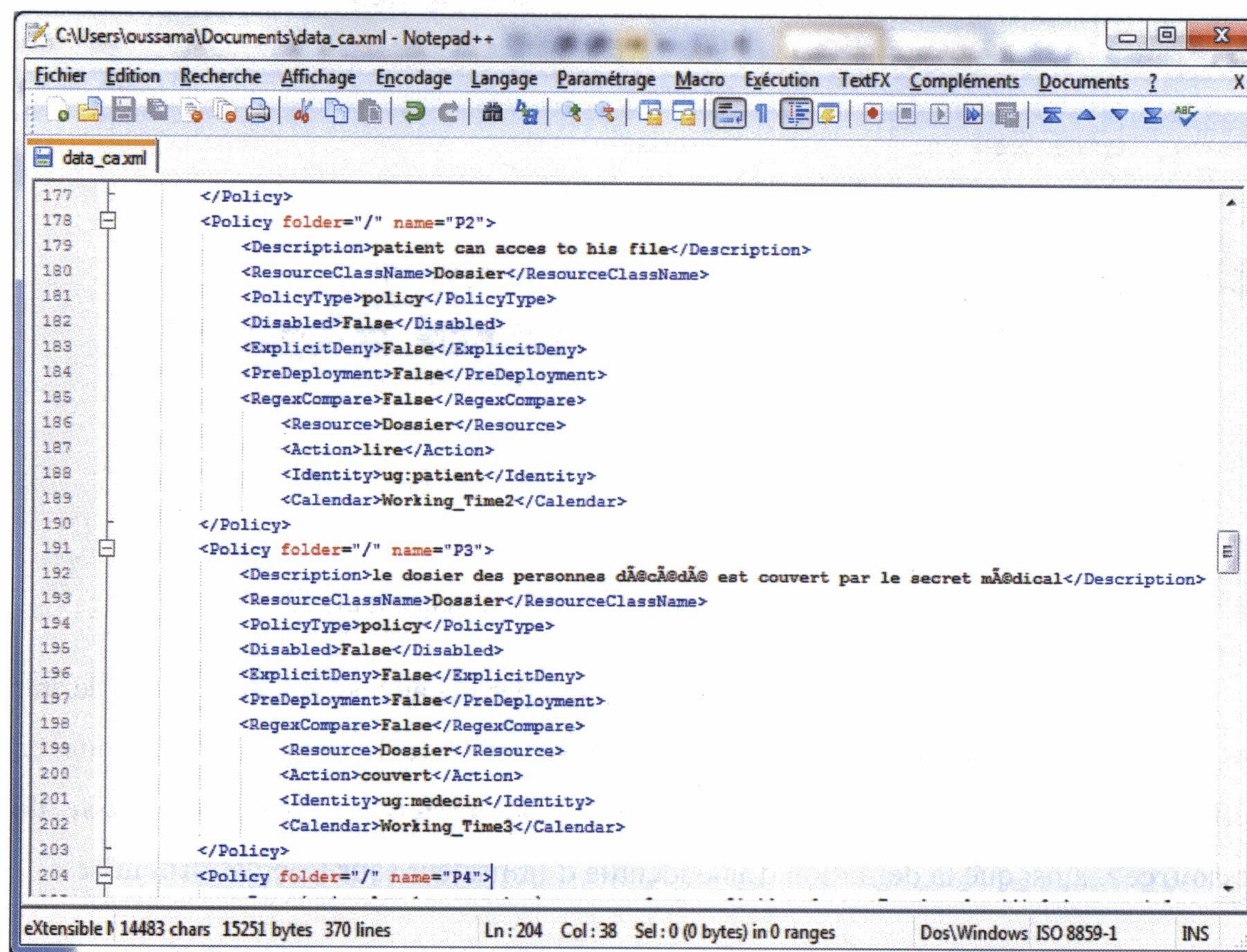
Cette figure montre l'interface graphique qui permet aux administrateurs de définir des règles de contrôle d'accès dans l'outil EEM. Cette règle peut être construite en définissant des valeurs aux attributs <Nom>, <Calendrier> et <Nom de classe de ressource>, ainsi que la définition d'une identité d'utilisateur sous l'onglet <Identité>.

2.1.5. Safex

L'outil EEM offre aux administrateurs la possibilité de générer des ensembles de politiques de contrôle d'accès sous la forme d'un fichier XML en utilisant l'application

Safex. Ce fichier peut être utilisé pour des tests de sécurité dans les systèmes et peut aussi servir à la migration des politiques de contrôle d'accès dans d'autres environnements de travail. Safex permet aussi de fournir les détails de l'outil utilisé, les politiques existantes et les détails des attributs associés tels que les utilisateurs, les ressources, le calendrier, etc.

Le fonctionnement de l'outil Safex ne sera pas expliqué dans ce mémoire, vu qu'il s'agit d'un outil indépendant de EEM. Le seul aspect qui nous intéresse est qu'il permet de générer des fichiers de politiques comme celui de la figure 2.8.



```
177 </Policy>
178 <Policy folder="/" name="P2">
179   <Description>patient can acces to his file</Description>
180   <ResourceClassName>Dossier</ResourceClassName>
181   <PolicyType>policy</PolicyType>
182   <Disabled>False</Disabled>
183   <ExplicitDeny>False</ExplicitDeny>
184   <PreDeployment>False</PreDeployment>
185   <RegexCompare>False</RegexCompare>
186   <Resource>Dossier</Resource>
187   <Action>lire</Action>
188   <Identity>ug:patient</Identity>
189   <Calendar>Working_Time2</Calendar>
190 </Policy>
191 <Policy folder="/" name="P3">
192   <Description>le dossier des personnes dÃ©cÃ©dÃ© est couvert par le secret mÃ©dical</Description>
193   <ResourceClassName>Dossier</ResourceClassName>
194   <PolicyType>policy</PolicyType>
195   <Disabled>False</Disabled>
196   <ExplicitDeny>False</ExplicitDeny>
197   <PreDeployment>False</PreDeployment>
198   <RegexCompare>False</RegexCompare>
199   <Resource>Dossier</Resource>
200   <Action>couvert</Action>
201   <Identity>ug:medecin</Identity>
202   <Calendar>Working_Time3</Calendar>
203 </Policy>
204 <Policy folder="/" name="P4">
```

eXtensible 14483 chars 15251 bytes 370 lines Ln: 204 Col: 38 Sel: 0 (0 bytes) in 0 ranges Dos\Windows ISO 8859-1 INS

Figure 2.8 : Exemple de fichier de politiques

Dans la suite de notre projet, ce fichier XML de politiques sera utilisé comme entrée pour tester notre outil de vérification d'anomalies dans un système de contrôle d'accès. Nous donnerons des explications sur le contenu de ce fichier dans le chapitre V.

3. SITEMINDER

3.1. Introduction

Siteminder est une plate-forme qui permet de sécuriser la gestion des portails, intranets et extranets, ainsi que la gestion et la construction des sites web sécurisés via des services d'authentification, d'autorisation et de personnalisation. Par l'utilisation de Siteminder, un administrateur peut facilement mettre en œuvre des politiques de sécurité qui protègent les ressources et les applications web, ainsi que la gestion de l'authentification et les privilèges d'autorisation en se basant sur le modèle de politiques de sécurité user-centric (Boettner, Gupta, Wu, & Allen, 2009).

Gestion de l'authentification

L'authentification d'un utilisateur est la première étape qui permet la sécurisation des applications Web, l'établissement de l'identité de l'utilisateur et la détermination des tâches que chaque utilisateur peut exécuter. CA Siteminder supporte une variété de méthodes d'authentification, y compris l'authentification par mot de passe, par jetons (tokens) (O'Gorman, 2003), par certificats X.509¹, etc. Les méthodes d'authentification peuvent également être combinées afin de présenter une forte authentification, à titre d'exemple, une authentification par certificat peut être exigée en plus d'une authentification par mot de passe.

¹ une norme de cryptographie de l'Union internationale des télécommunications qui spécifie les formats standard de certificats électroniques et établit un algorithme pour la validation de chemin de certification.

Gestion de l'autorisation

Dans Siteminder, la gestion des droits pour les clients, les partenaires et les employés dans toutes les applications Web est centralisée par le biais d'un service de sécurité partagé entre ces applications. L'utilisation de l'autorisation centralisée permet de réduire le coût de développement en permettant aux développeurs de se concentrer sur l'aspect logique de l'application et non pas sur les politiques de sécurité de chaque application.

3.2. Architecture de Siteminder

L'outil Siteminder permet aux administrateurs d'attribuer des méthodes d'authentification, de gérer les privilèges d'autorisation pour accéder à des ressources spécifiques et de créer des politiques qui permettent la mise en œuvre de ces autorisations. Siteminder est constitué de deux composantes d'exécution principales et une composante d'administration, voir la figure 2.9 qui montre l'architecture de Siteminder.

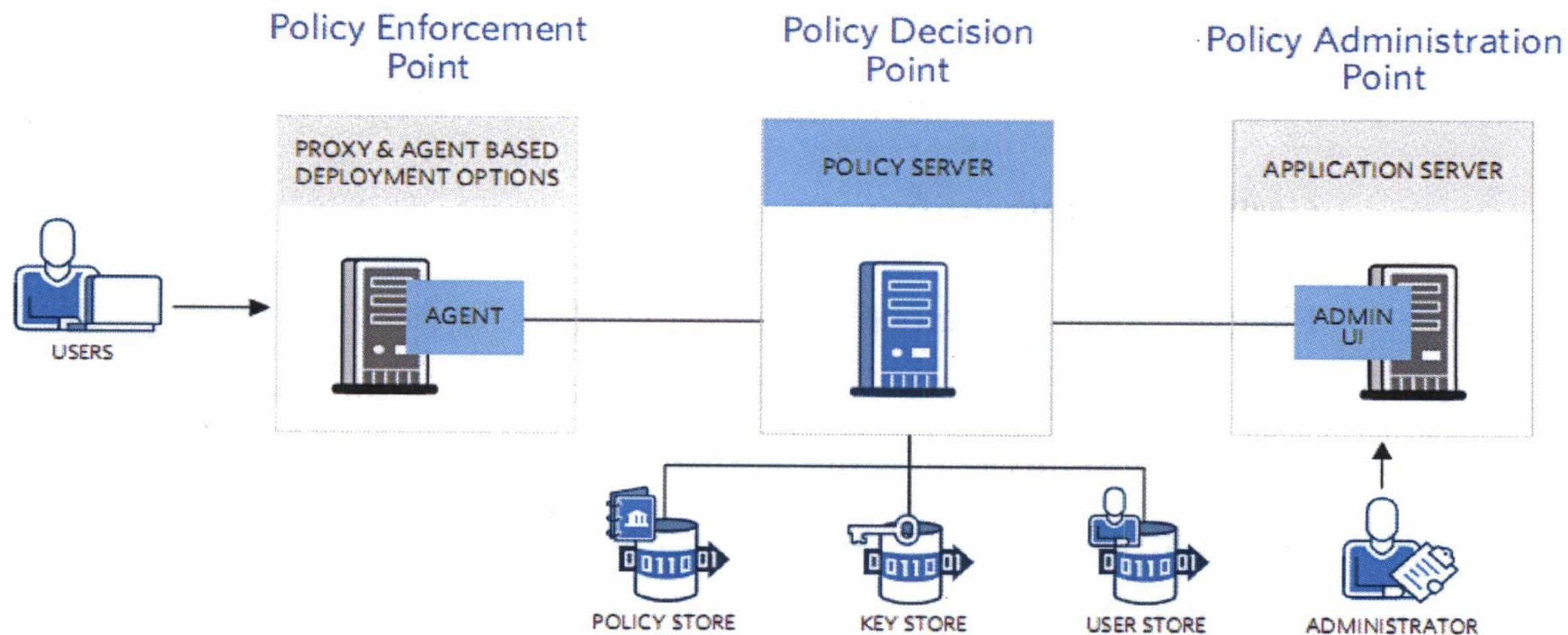


Figure 2.9 : Architecture de Siteminder (CA Technologies, 2009)

Premièrement, l'agent web de Siteminder représente le point d'application de politiques PEP (Policy Enforcement Point). L'agent web fonctionne en combinaison avec le serveur web afin d'intercepter toutes les requêtes envoyées par un utilisateur pour accéder aux ressources et de déterminer si la ressource est protégée par l'outil Siteminder.

Si la ressource n'est pas protégée par Siteminder, la requête passe par le processus normal sans intervention de l'agent web. Par contre, si la ressource est protégée, l'agent web communique avec le serveur de politiques afin de déterminer si l'accès à une ressource spécifique est permis ou non. Deuxièmement, le serveur de politiques représente aussi le PDP (point de décision de politiques). Le PDP évalue les politiques de contrôle d'accès établies par un administrateur afin de prendre une décision d'autorisation qui sera communiquée à l'agent PEP. Ces politiques définissent les ressources qui sont protégées ainsi que les utilisateurs qui ont le droit d'accéder aux ressources. Un PDP peut communiquer avec un ou plusieurs PEP. Troisièmement, le PAP représente le point d'administration de politique. Ce dernier permet aux administrateurs de définir des politiques de contrôle d'accès par le biais d'une interface graphique. Un PAP peut connecter et gérer un ou plusieurs serveurs de politiques.

Une politique de Siteminder protège les ressources en permettant ou en interdisant explicitement l'accès des utilisateurs. Ce processus de protection contient quatre étapes :

- La détermination des ressources à protéger
- La définition des utilisateurs, groupes ou rôles qui ont accès aux ressources
- La livraison des ressources aux utilisateurs autorisés
- En cas d'interdiction d'accès, la politique définit aussi la façon dont cet utilisateur est traité

La figure suivante présente un scénario qui montre le rôle de chacune des entités mentionnées précédemment.

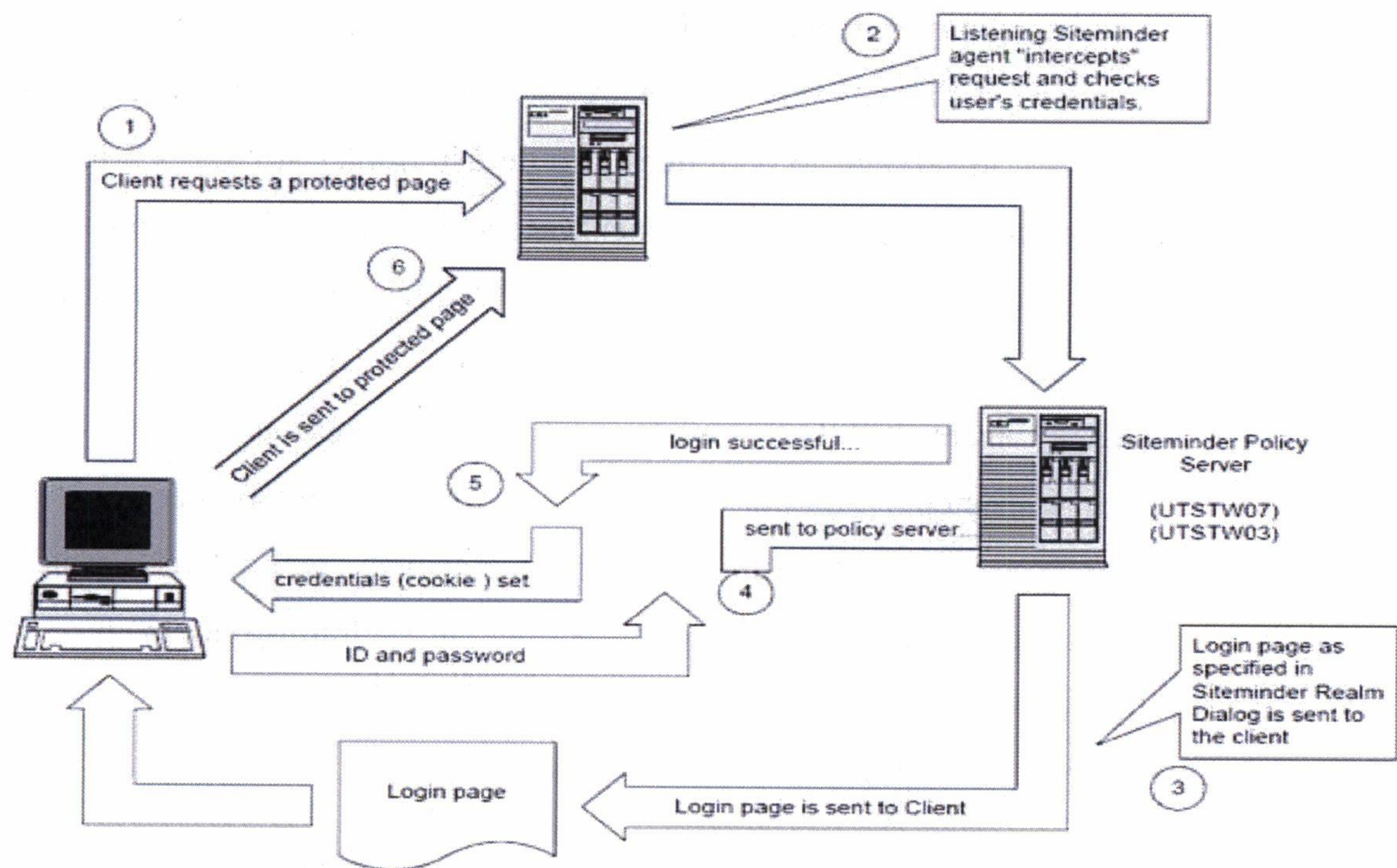


Figure 2.10 : Scénario d'authentification et d'autorisation (Swafford, 2009)

1. L'utilisateur tente d'accéder à une application protégée par Siteminder. Une application est un ensemble de ressources, par exemple *Protected Page* dans la figure ci-dessus.
2. L'agent web (PEP) de Siteminder intercepte la requête envoyée par l'utilisateur afin de vérifier si la ressource concernée est protégée. Dans le cas échéant, le PEP transmet la requête au serveur de politique (PDP).
3. Le serveur de politique indique à l'agent web que la ressource est protégée, et présente à l'utilisateur la page d'authentification (Login page).
4. L'utilisateur fournit les informations requises pour s'authentifier (exemple : nom d'utilisateur/mot de passe).
5. Le serveur de politique authentifie l'utilisateur. Après avoir vérifié l'identité de l'utilisateur, le serveur cherche les politiques associées au profil de cet utilisateur pour prendre une décision concernant l'accès aux ressources du système.

- Après avoir validé l'identité de l'utilisateur par le PDP, l'agent web PEP redirige l'utilisateur vers l'application protégée par Siteminder

3.3. Structure de politiques et règles

3.3.1. Structure de politiques

La sécurité et la gestion d'accès au sein de Siteminder sont basées sur les politiques. Les politiques de Siteminder ne définissent pas seulement les droits de l'utilisateur et les actions à appliquer sur la ressource, mais aussi la relation entre l'utilisateur et la ressource, ce qui donne plus de flexibilité à la gestion d'accès et de la sécurité. Voir la figure 2.12 qui montre l'interface de la structure de base pour créer une politique dans Siteminder.

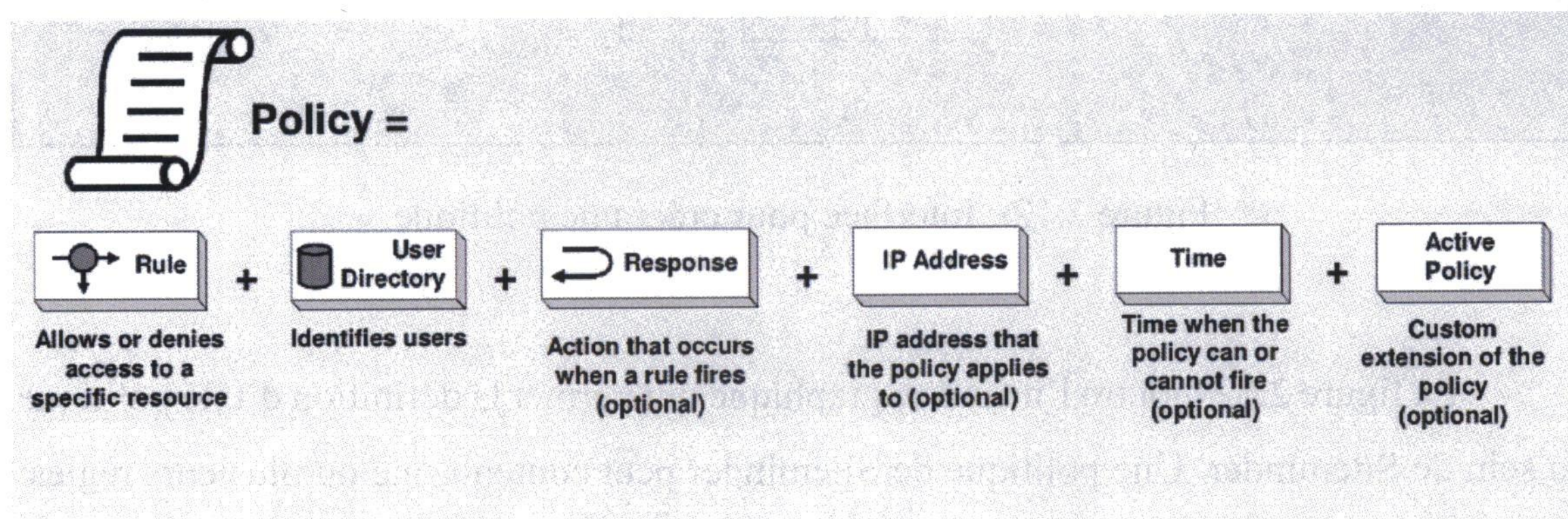


Figure 2.11 : Structure d'une politique (Netegrity-Inc, 1997)

La figure 2.11 montre la structure d'une politique de contrôle d'accès dans Siteminder, ainsi que les différentes composantes de cette politique. Dans la suite de ce chapitre, nous expliquons plus en détails chaque composante et le processus de définition de chacune d'elles à travers une interface graphique. Nous montrerons aussi la différence entre une politique de contrôle d'accès et une règle de contrôle d'accès.

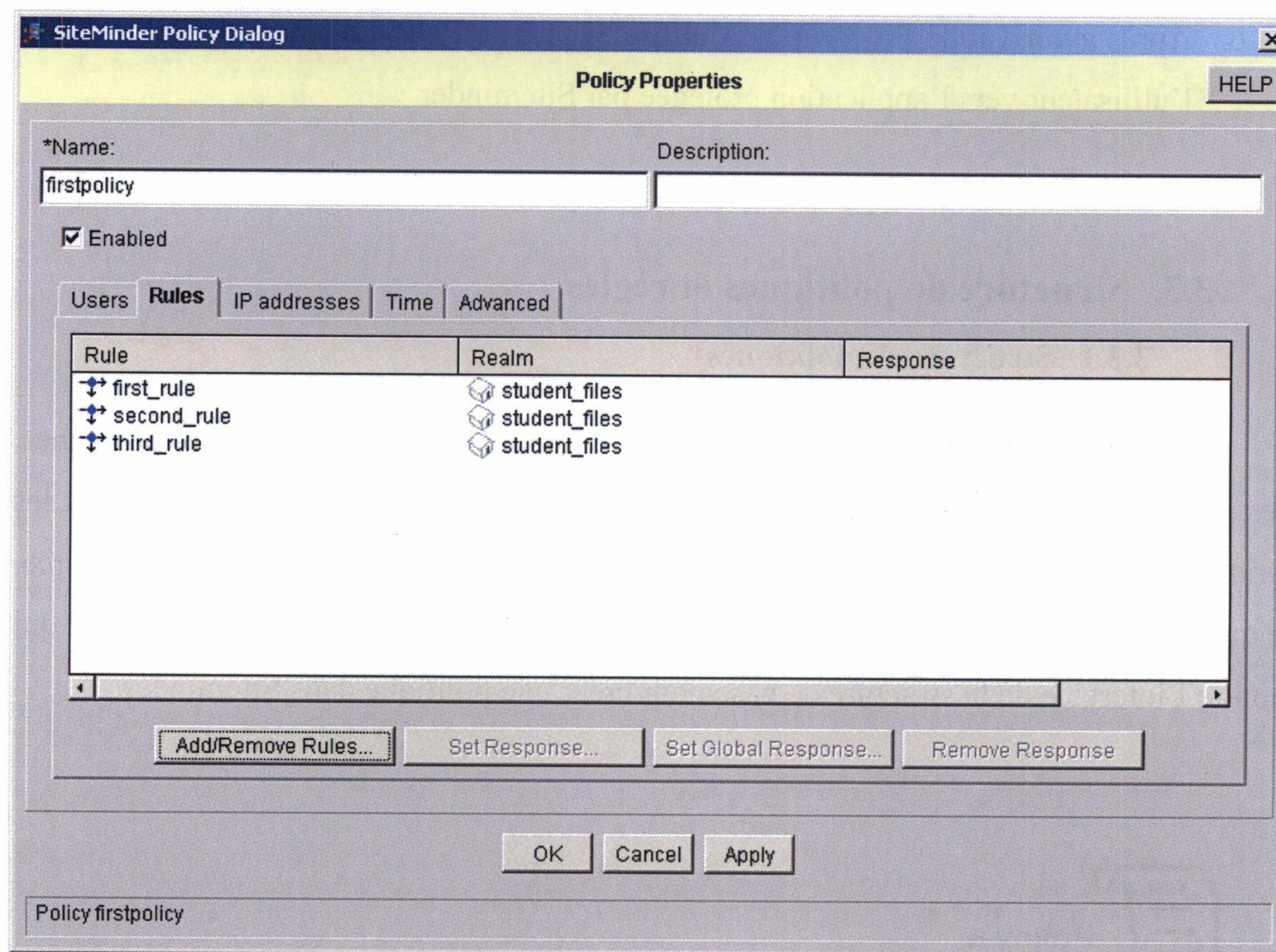


Figure 2.12 : Interface pour créer une politique

La figure 2.12 montre l'interface graphique qui permet la définition d'une politique au sein de Siteminder. Une politique de Siteminder peut contenir une ou plusieurs règles. On constate, sur cette figure, que la politique « firstpolicy » contient trois règles qui sont «first_rule», «second_rule» et «third_rule».

Dans la suite de cette section, nous expliquons les éléments essentiels d'une politique de sécurité.

3.3.2. Règle

Dans Siteminder, les règles représentent une partie des politiques qui déterminent précisément les ressources protégées ainsi que les types d'action qui peuvent activer ces règles. Cet élément est expliqué plus en détail dans la section 3.3.4.

3.3.3. Répertoire d'utilisateurs

Siteminder permet aux administrateurs de vérifier les utilisateurs qui peuvent accéder aux applications et de contrôler leurs environnements d'applications.

Le contrôle d'accès se fait via une connexion à un annuaire d'utilisateur ou une base de données existante dans le réseau de l'entreprise. Le but de cette connexion est de mettre les données de chaque utilisateur à la disposition du serveur de politique qui décide l'autorisation d'accès aux ressources. Les données enregistrées dans le répertoire d'utilisateur sont composées des éléments suivants :

- L'information organisationnelle
- Les attributs d'utilisateur et groupes
- L'information d'identification de l'utilisateur, par exemple le mot de passe
- Les attributs d'utilisateurs, par exemple le nom et le prénom

La figure suivante représente un exemple d'un répertoire d'utilisateurs qui contient deux utilisateurs (Alice et Bob) et un groupe d'administrateurs.

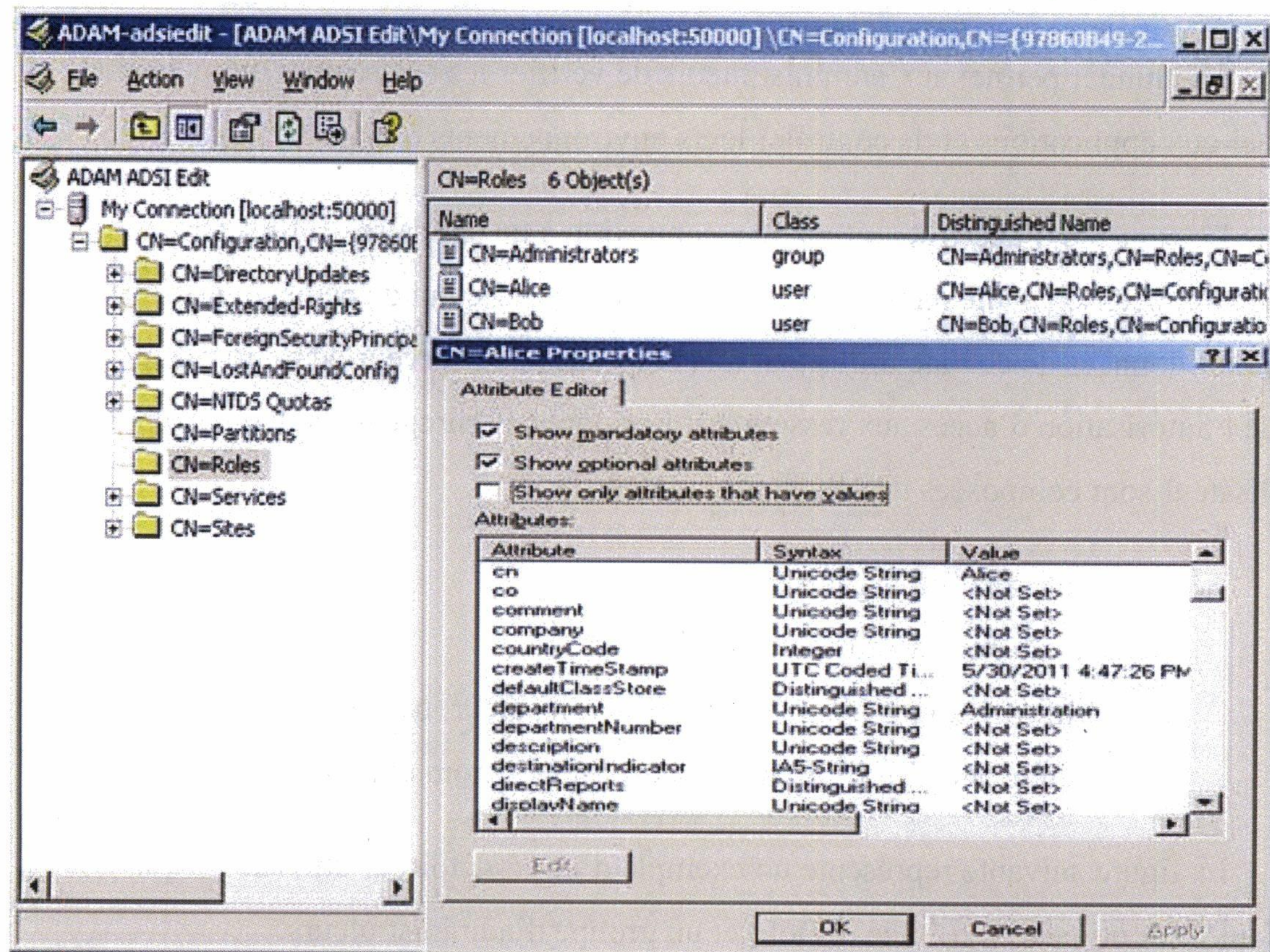


Figure 2.13 : Exemple d'un répertoire d'utilisateurs

Dans la structure de politique, nous nous sommes restreints à expliquer les règles et le répertoire d'utilisateurs qui sont des éléments essentiels dans une politique de contrôle d'accès. Le reste des éléments sont des éléments optionnels et ne sont pas sujet de notre mémoire.

3.3.4. Structure de Règles (rules)

Une règle définit un ensemble d'actions à exécuter par un utilisateur autorisé sur une ressource protégée. La figure suivante montre la structure générale des règles dans Siteminder.

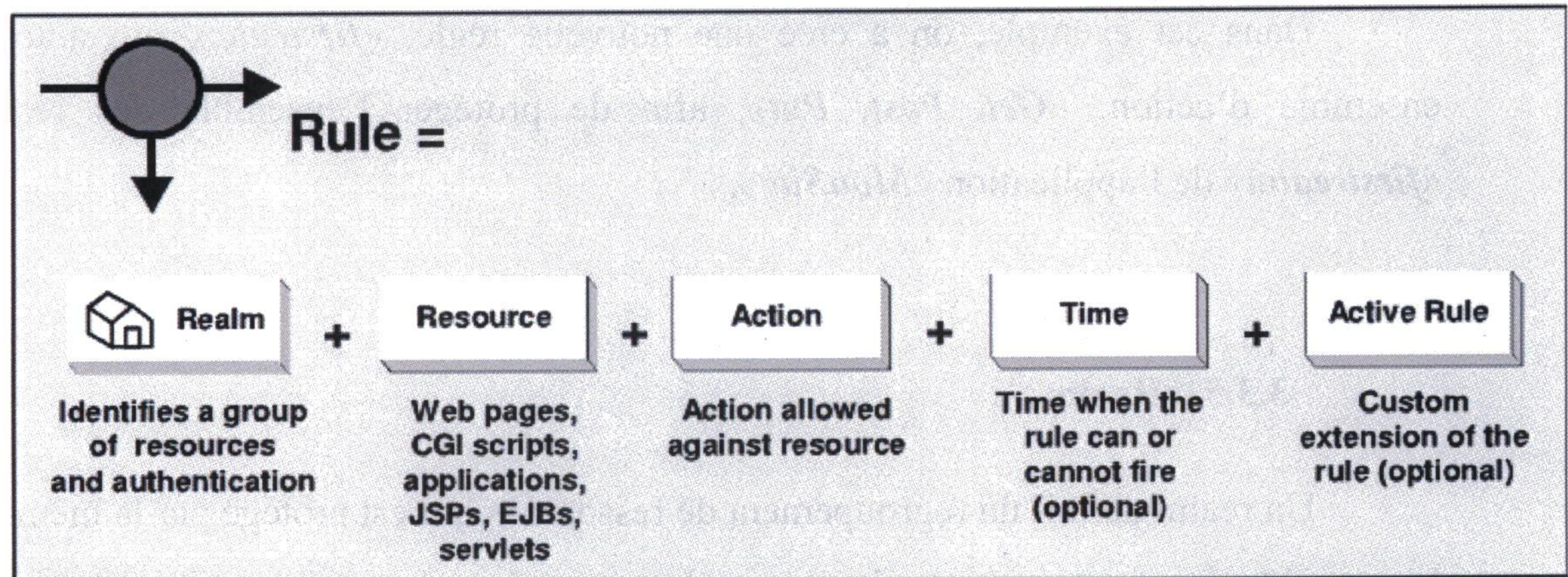


Figure 2.14 : Structure d'une règle (Netegrity-Inc, 1997)

L'interface suivante permet à un administrateur de créer une nouvelle règle dans Siteminder.

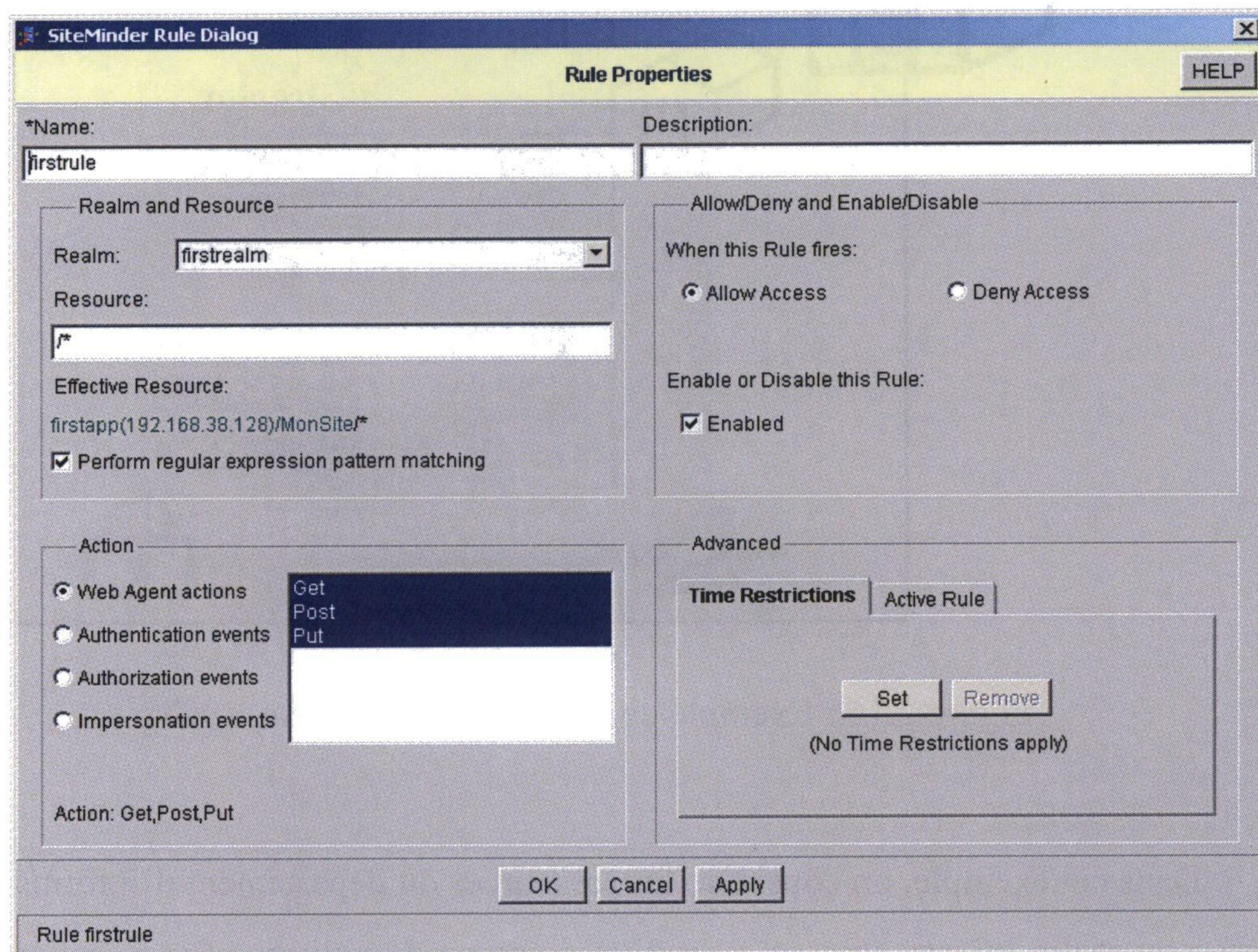


Figure 2.15 : Interface pour créer une règle

Dans cet exemple, on a créé une nouvelle règle, «*firstrule*», qui autorise un ensemble d'action, «*Get, Post, Put*», afin de protéger l'ensemble des ressources «*firstrealm*» de l'application «*MonSite*».

3.3.5. Realm

Un realm définit un regroupement de ressources qui est protégé par la même règle. L'ensemble des ressources au sein d'un realm est géré par le même agent web.

La figure 2.16 montre un exemple de realm.

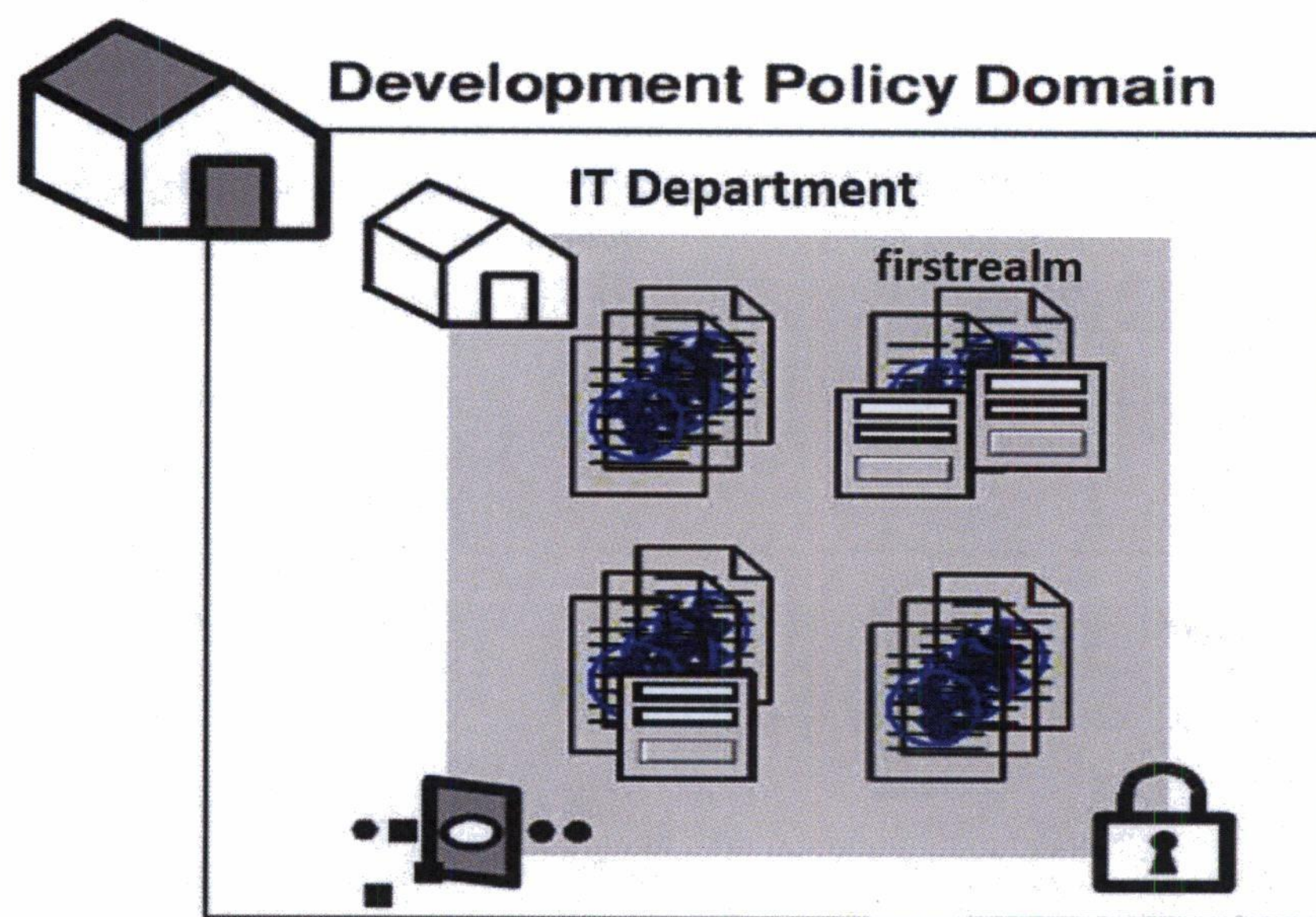


Figure 2.16 : Exemple de realm (Netegrity-Inc, 1997)

Dans cet exemple, on considère les ressources du département d'informatique afin de créer un realm «*firstrealm*» au sein d'un domaine de politique (voir section suivante pour plus de détails).

Sur la figure 2.17, on trouve l'interface graphique de Siteminder qui permet la définition d'un realm.

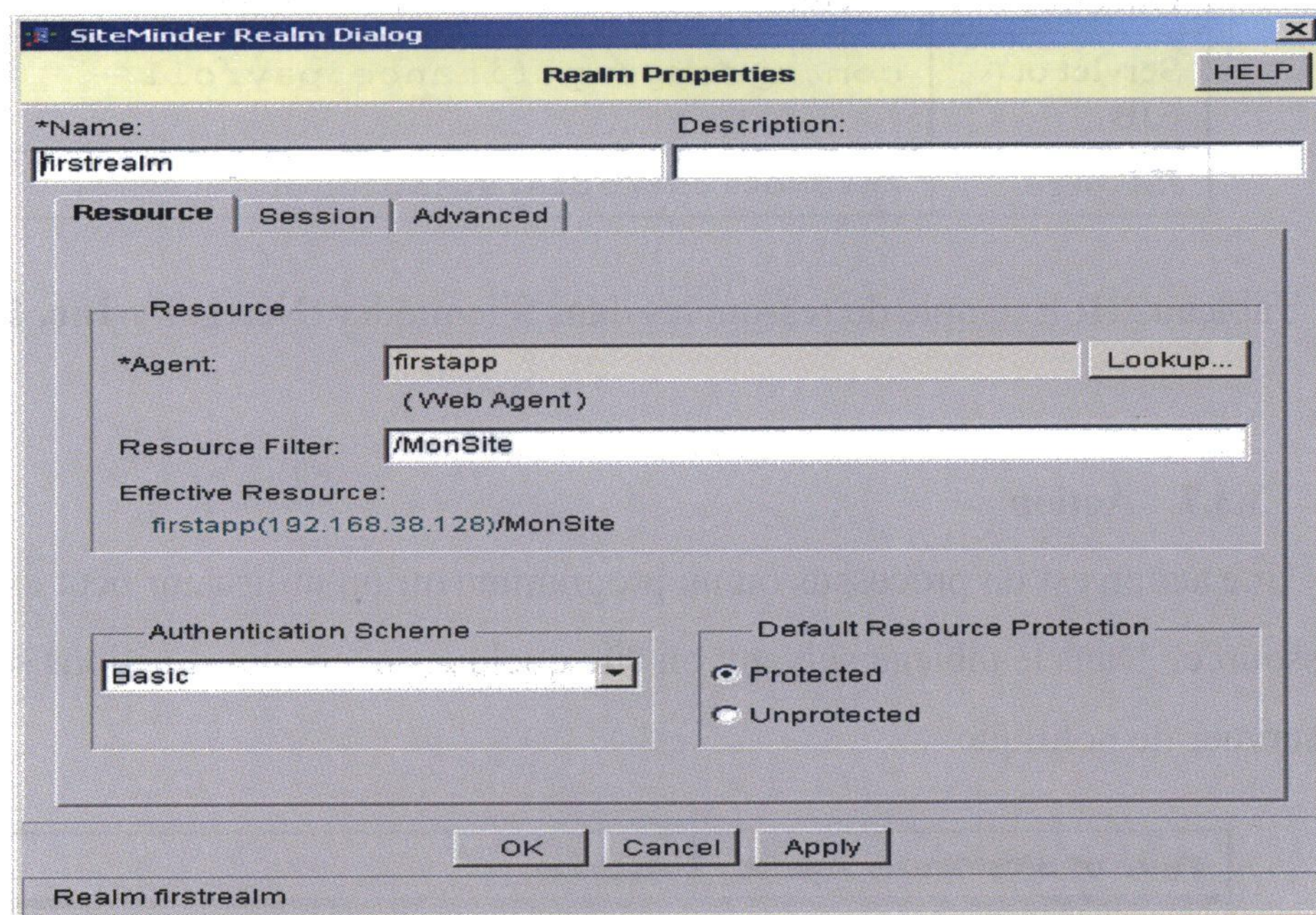


Figure 2.17 : Interface pour créer un realm

Dans cet exemple, on définit un realm, «*firstrealm*», de l'application «*Monsite*» qui est protégé par l'agent web «*firstapp*».

3.3.6. Ressource

Dans Siteminder, une ressource qui est protégée par un agent web signifie tout objet qu'un utilisateur peut tenter d'utiliser ou tout privilège qu'un utilisateur peut tenter de posséder.

Le tableau suivant montre quelques exemples des ressources qui peuvent exister dans un système, par exemple des fichiers, des pages web ou des bases de données.

| Resource | Example |
|----------------|--|
| Web page | /applications/myapp.exe |
| CGI script | /www.acme.com/price/1,2,0-a-0-0,000.html?st.dl.search.qs.results |
| Directory | /mydirectory |
| Servlet or EJB | com.mycompany.finance.payroll |
| JSP page | /promotions/offers.jsp |

Tableau 2.1: Exemple de ressources dans Siteminder (Netegrity-Inc, 1997)

3.3.7. Action

Une action est un processus ou un programme qu'un utilisateur peut exécuter sur une ressource. Dans le tableau suivant, on cite quelques exemples d'actions supportées par le serveur de politique.

| Type of SiteMinder Agent | Actions |
|--------------------------|--|
| Web Agent | HTTP Get, HTTP Post, HTTP Put |
| Affiliate Agent | Visit - this action lets the SiteMinder portal and the affiliate interact. |
| Application Server Agent | For servlets: doGet, doPost For EJBs: invoke, lookup, load, close |
| RADIUS | Authenticate - a RADIUS server only authenticates users. |

Tableau 2.2 : Exemple d'actions dans Siteminder (Netegrity-Inc, 1997)

Dans notre projet, nous avons utilisé l'agent web comme agent de Siteminder.

Dans la figure suivante, la partie encadrée en rouge montre l'interface graphique qui permet la spécification des actions.

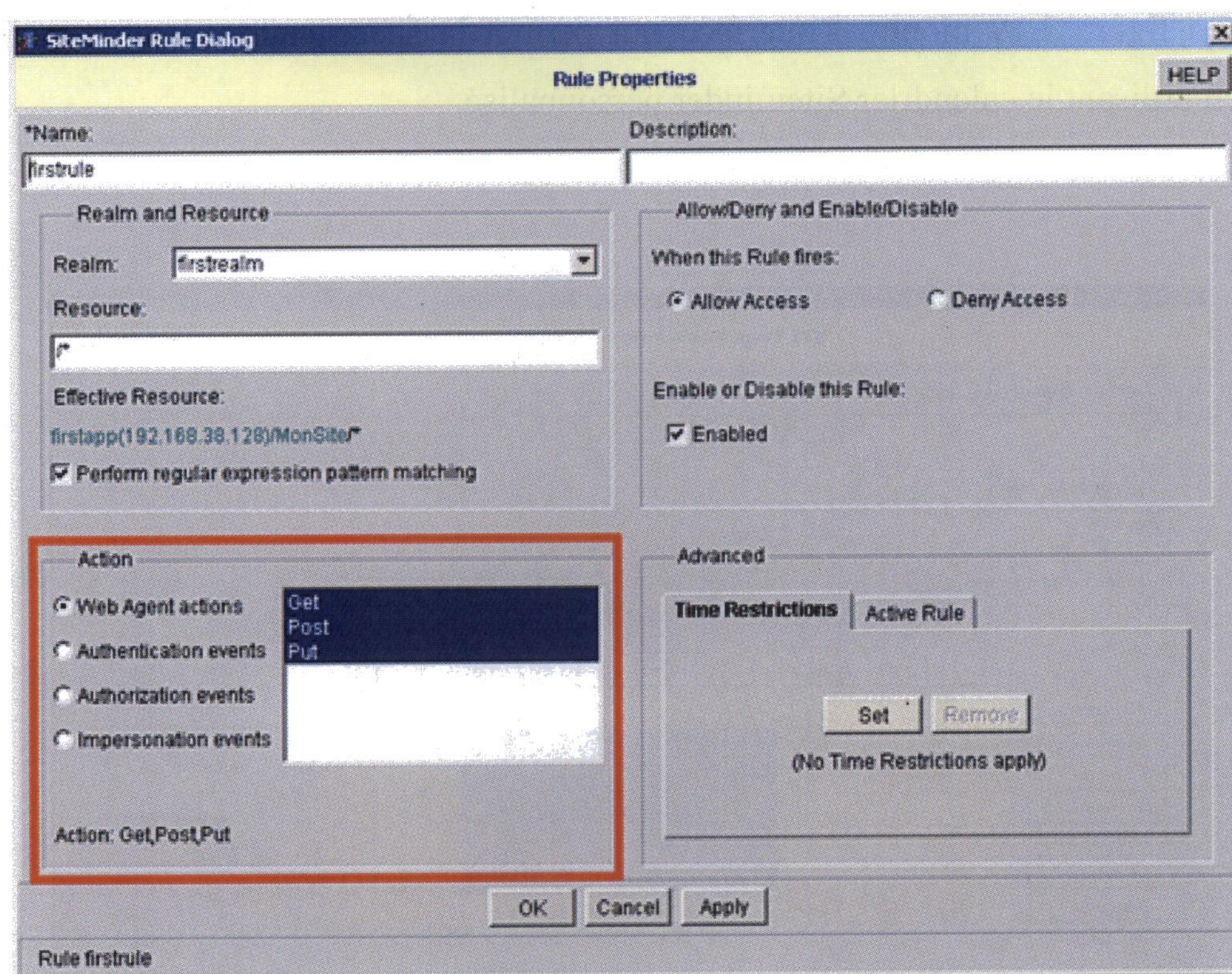


Figure 2.18 : Interface pour définir les actions

Dans cet exemple, on définit la règle «*firstrule*» qui contient trois types d'action, «*Get, Post, Put*» correspondant à lire, écrire et mettre à jour.

3.3.8. Calendrier (Time)

SiteMinder permet la spécification des conditions et des filtres qui restreignent l'accès aux ressources du système selon les règles définies. Une règle n'est exécutée que si la condition spécifiée par le filtre est vraie. Dans notre projet, nous avons étudié le calendrier qui représente un élément important de filtrage.

Dans cette section, on montre comment ajouter la contrainte de temps en utilisant l'interface du calendrier SiteMinder personnalisé. Les règles de contrôle d'accès peuvent spécifier des critères basés sur des intervalles de temps comme les heures de travail par le biais du calendrier.

La figure 2.19 montre l'interface qui permet la spécification des intervalles de temps en utilisant le calendrier Siteminder personnalisé.

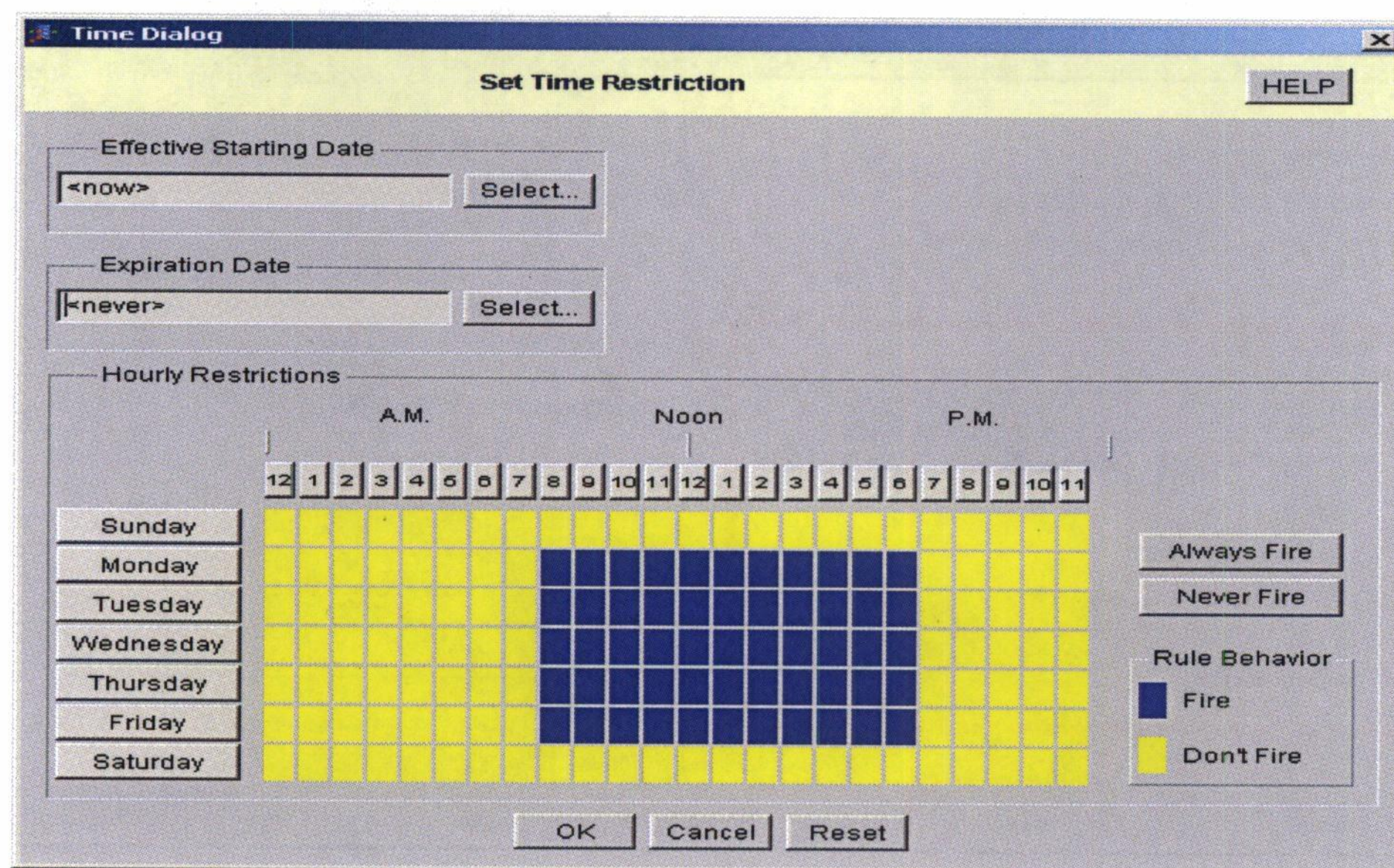


Figure 2.19 : Interface pour définir les intervalles de temps

Dans cet exemple, nous avons défini des intervalles de temps qui sont colorés en bleu. Ce calendrier permet l'activation des règles selon les intervalles définis.

3.3.9. Active Rule (expression logique)

Active rule est une expression logique qui permet le contrôle d'accès aux données selon des conditions logiques. Une expression logique est incluse dans la décision d'autorisation dynamique. Cette décision est basée sur les fonctions logiques qui se trouvent dans une bibliothèque de données. Ces fonctions sont invoquées par le serveur de politique.

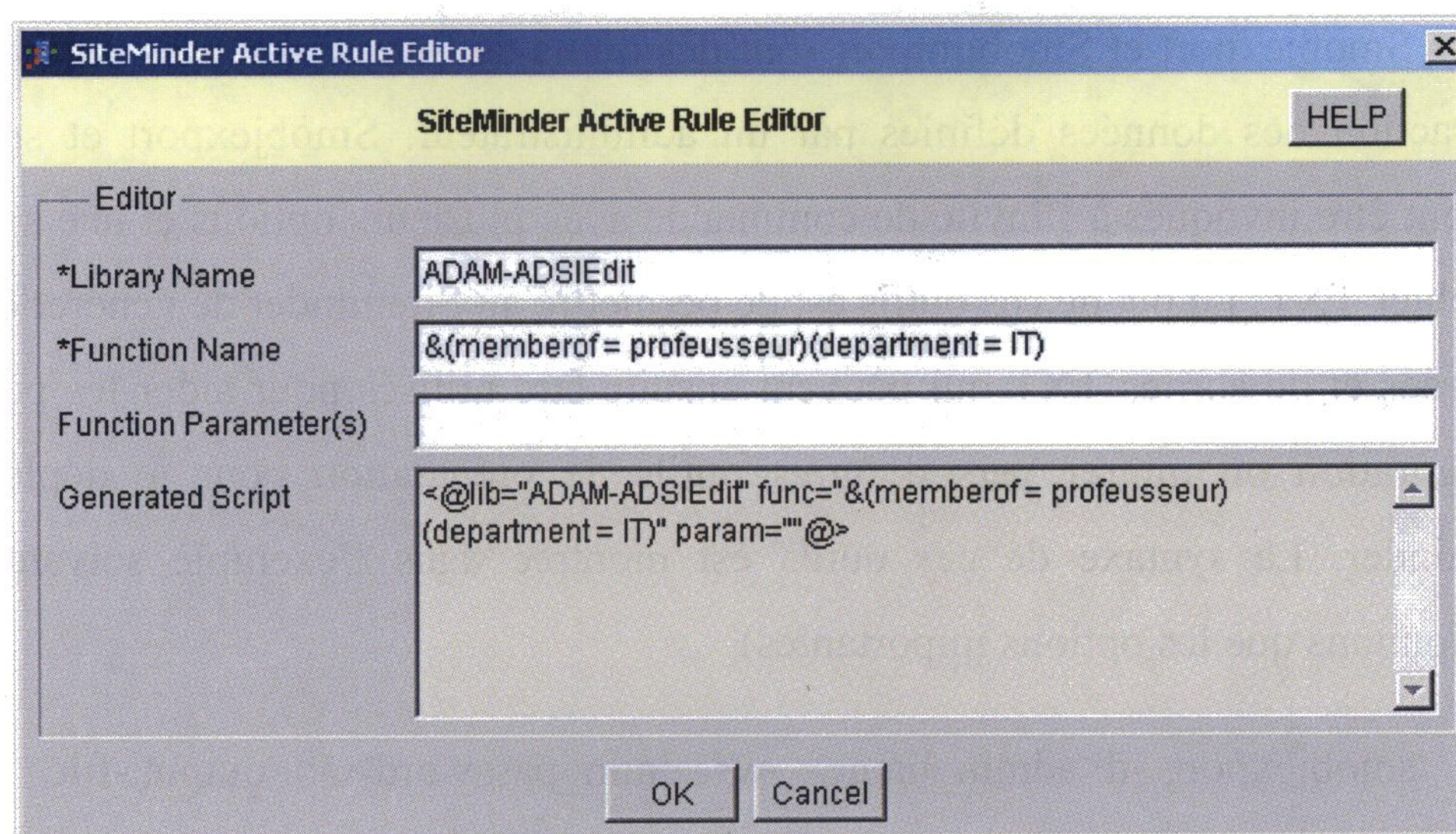


Figure 2.20 : Interface pour définir une expression logique

Une expression logique peut être définie via une interface graphique de Siteminder. Cette interface contient deux champs obligatoires comme le montre la figure 2.20.

- Library name représente le nom de la bibliothèque partagée qui prend en charge la règle active.
- Function name représente la fonction logique qui permet l'implémentation de la règle active.

L'expression logique mentionnée dans la figure 2.20 définit qu'un utilisateur doit être un membre du groupe professeur et appartenir au département IT.

3.3.10. Smobjexport et Smobjimport

Smobjexport et Smobjimport sont des outils de Siteminder. Ces outils permettent l'extraction des données définies par un administrateur. Smobjexport et smobjimport peuvent être invoqués à l'invite de commande avec plusieurs options et une spécification de fichier TXT. Le but de ces outils est de permettre au Siteminder de générer des fichiers d'entrées et de sorties TXT qui peuvent ensuite être utilisés pour aider les politiques de configuration ou migrer vers d'autres outils de préparation pour le déploiement de Siteminder. La syntaxe de ces outils est montrée dans l'exemple suivant (nous ne mentionnons que les options importantes).

```
smobjexport -d<admin_name> -w<admin_password> -o<output_file>
```

```
smobjimport -i<input_filename> -d<admin_name> -w<admin_password>
```

- admin_name : un administrateur qui a le droit d'exécuter cette commande
- admin_password : le mot de passe qui doit être fourni par un administrateur
- output_file : l'emplacement où un administrateur peut enregistrer les résultats
- input_filename : le fichier de données qu'un administrateur peut importer dans un système

La figure suivante montre un exemple de fichier de politiques que nous avons défini dans ce chapitre. Ce fichier a été exporté en utilisant la commande expliquée précédemment.

3.4. Conclusion

Dans ce chapitre, nous avons montré comment les outils de CA Technologies (EEM, Siteminder) permettent de formuler des politiques de contrôle d'accès par le biais d'une interface graphique. Nous avons également discuté des principales caractéristiques de ces outils et de la façon dont nous pouvons utiliser d'autres outils pour importer ou exporter les politiques définies par un administrateur. Toutefois, si un administrateur ajoute des politiques contradictoires, EEM et Siteminder ne sont pas en mesure de détecter des anomalies (incohérences ou incomplétudes). La détection de ces anomalies est l'objectif de notre travail.

Nous avons seulement mentionné les principaux services offerts par les outils EEM et Siteminder, soit ceux qui sont nécessaires pour comprendre notre travail.

CHAPITRE III: ÉTAT DE L'ART

1. Modèles de contrôle d'accès

Un grand nombre de modèles ont été proposés afin d'exprimer les politiques de contrôle d'accès dans le domaine de la sécurité informatique. Ces modèles contiennent une variété d'aspects qui visent à résoudre les soucis de la sécurité de l'information, par exemple, la détection des conflits entre les politiques de sécurité (l'incohérence, l'incomplétude et la redondance). Dans la littérature, il existe une vaste gamme de modèles de contrôle d'accès qui ont été proposés pour mieux s'adapter aux besoins des organisations. Nous trouvons, par exemple, les modèles basés sur le rôle RBAC (Rôle Based Access Control) (Sandhu R. S., Coyne, Feinstein, & Youman, 1996) qui affectent des permissions à des rôles. Dans RBAC, un utilisateur peut avoir différents rôles dans une même session ou durant des sessions différentes. Voir section 2 pour plus de détails. Nous présentons aussi dans la section 3, le langage XACML qui a été adopté par plusieurs organisations et qui a eu un grand succès dans le domaine de contrôle d'accès.

2. Modèle à base de rôle RBAC (Role Based Access Control)

Afin de faciliter l'administration des politiques de contrôle d'accès, les chercheurs ont conçu un modèle de contrôle d'accès qui permet de refléter la structure organisationnelle de l'entreprise (Ferraiolo, Sandhu, & Gavrila, 2001) (Ferraiolo, Cugini, & Kuhn, 1995). Ce modèle permet de structurer l'expression de politiques d'autorisation à base du **rôle**. Le rôle est un concept organisationnel qui permet à un utilisateur d'obtenir certaines permissions, c'est-à-dire d'utiliser les ressources en fonction de sa responsabilité dans l'organisation (par exemple, professeur, étudiant, directeur, etc). Il peut être vu

comme un ensemble de tâches associées à une activité de travail particulière. Au lieu de spécifier individuellement les accès autorisés, l'autorisation d'accès à un objet est accordée selon le rôle. Contrairement au contrôle d'accès discrétionnaire, les permissions d'accès ne sont pas affectées directement au sujet, et ce, comme illustré dans la figure 3.1 qui montre l'attribution des permissions aux utilisateurs via les rôles. RBAC (Rôle Based Access Control) fut le premier modèle de contrôle d'accès basé sur ces concepts. Le modèle RBAC (Sandhu R. S., Coyne, Feinstein, & Youman, 1996) a aussi introduit un nouveau concept appelé **session**. Afin d'exécuter une action sur un objet, le sujet doit d'abord créer une session et le rôle qui a reçu l'autorisation de réaliser cette action sera activé durant cette session. Durant une session, plusieurs rôles peuvent être activés et chaque session est associée à un seul utilisateur.

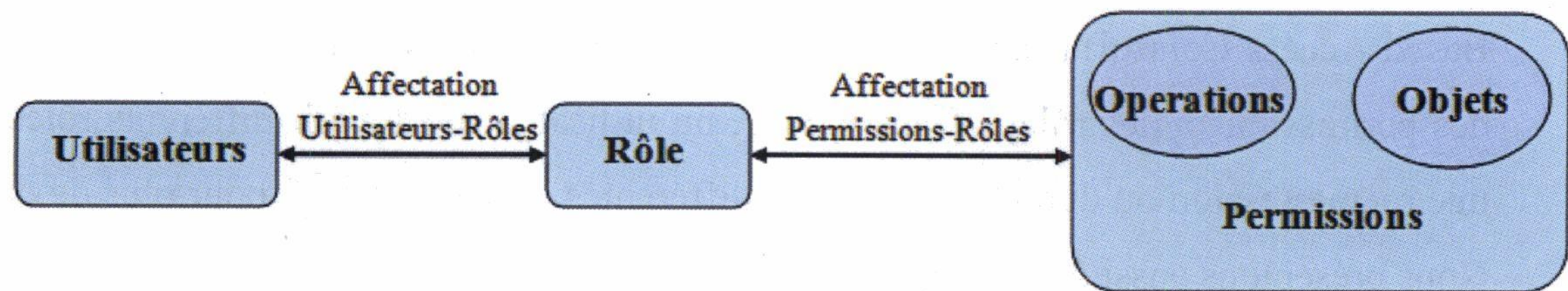


Figure 3.1 : Affectation des permissions dans RBAC (Sandhu R. S., Coyne, Feinstein, & Youman, 1996)

Un rôle peut être affecté à plusieurs utilisateurs et un utilisateur peut jouer plusieurs rôles. Il en est de même pour les permissions; un rôle peut détenir plusieurs permissions et une permission peut être associée à plusieurs rôles. Cette tâche d'affectation est faite séparément par des administrateurs.

Prenons comme exemple le domaine d'enseignement, un professeur Bob peut être à la fois professeur et directeur de département au sein de l'université. Dans ce cas, il pourra accéder aux fichiers des notes des étudiants en jouant le rôle professeur et aux dossiers administratifs de l'université en jouant le rôle directeur de département. Donc le

concept de rôle simplifie la tâche des administrateurs, car au lieu de devoir gérer séparément les permissions aux usagers individuellement, il permet de créer les rôles dans une organisation (e.g rôle ingénieur), et puis affecter les usagers aux rôles. En outre, la structure des rôles dans une organisation avec les permissions associées est beaucoup plus stable que l'affectation des usagers directement aux rôles.

Une famille de modèle de RBAC a été proposée dans la littérature (Sandhu R. S., Coyne, Feinstein, & Youman, 1996). La figure 3.2 montre quatre modèles généraux de cette famille.

$RBAC_0$ représente le modèle de base qui spécifie les attributs minimaux (Sujet, Rôle, Permissions, Session) pour un système qui supporte un modèle de contrôle d'accès basé sur le rôle.

Les extensions $RBAC_1$ et $RBAC_2$ incluent les principes de $RBAC_0$, mais chacun a aussi des caractéristiques indépendantes. $RBAC_1$ ajoute le concept d'hierarchie de rôle, qui implique des situations dans lesquelles les rôles peuvent hériter des autorisations des autres rôles. À titre d'exemple, un « professeur titulaire » peut hériter de toutes les permissions qui sont attribuées à un rôle « professeur associé ». Des extensions de $RBAC_0$ ont aussi été définies par l'ajout d'un ensemble de contraintes. Ces extensions ont donné naissance à un nouveau type de modèle sous le nom de $RBAC_2$. Ces contraintes peuvent être des contraintes de temps et de lieu. Par exemple, l'accès à une base de données au sein de l'université UQO après 18h n'est pas autorisé.

Le modèle $RBAC_3$ est l'association des deux modèles précédents $RBAC_1$ et $RBAC_2$ et, par transitivité, le modèle $RBAC_0$ est aussi inclus dans $RBAC_3$. Ce dernier supporte les deux concepts ; la hiérarchie de rôle et les contraintes (voir figure 3.2).

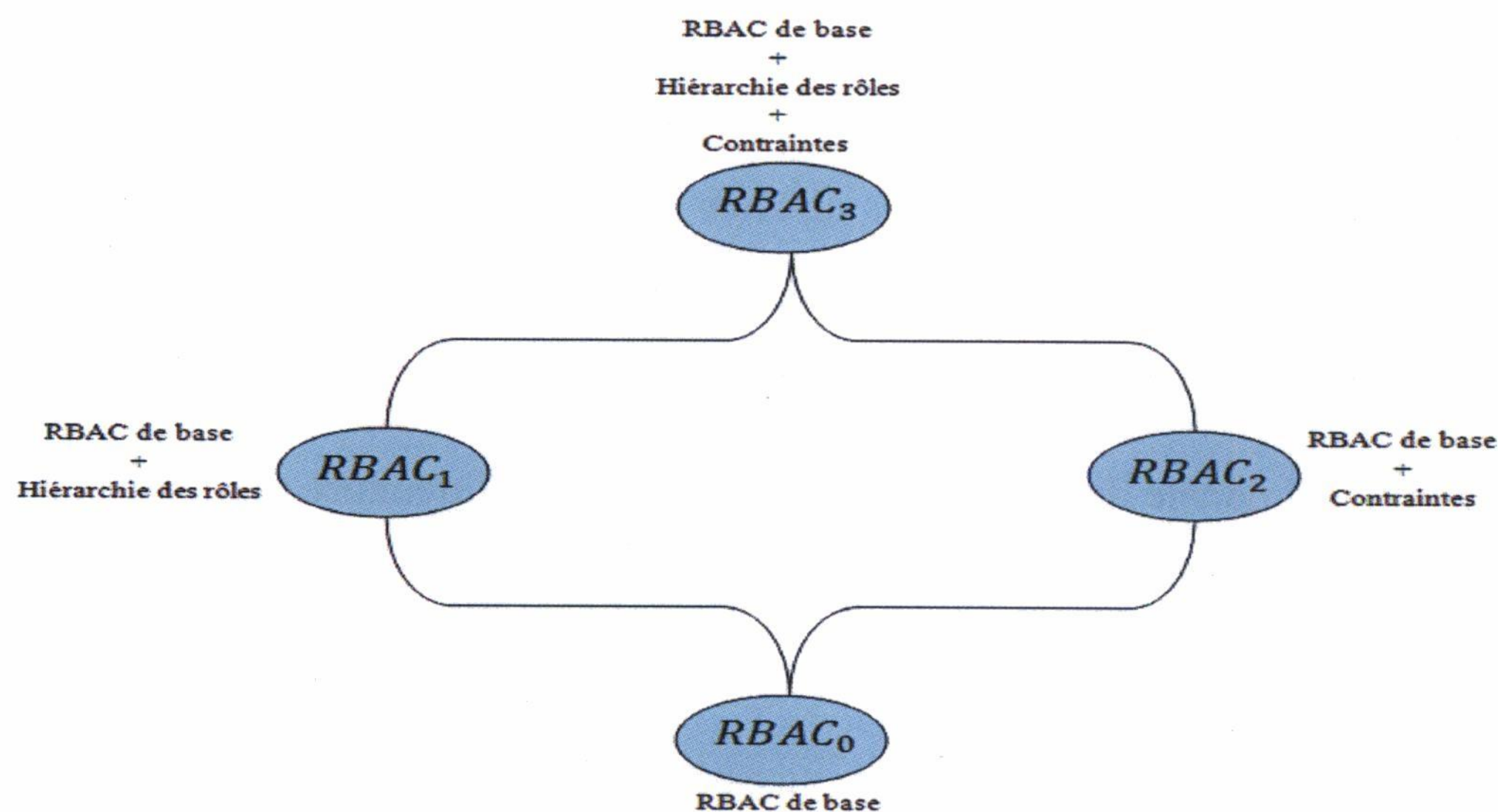


Figure 3.2 : Famille des modèles RBAC (Sandhu R. S., Coyne, Feinstein, & Youman, 1996)

Dans (Sandhu R. S., Coyne, Feinstein, & Youman, 1996), un modèle de contrôle d'accès basé sur la notion de rôle est décrit formellement de la manière suivante. Plus de détails sur la figure 3.3 qui propose la structure du modèle *RBAC₃* :

- Soit U un ensemble des utilisateurs, R un ensemble de rôles, P un ensemble de permissions et S un ensemble de sessions
- $AP \subseteq R \times P$, définit la relation qui affecte une permission à un rôle
- $AU \subseteq U \times R$, définit la relation qui affecte un ou plusieurs rôles à un utilisateur
- $HR \subseteq R \times R$, définit une hiérarchie des rôles partiellement ordonnés; $r \geq r'$ signifie que r' est un sous-rôle de r
- $User : S \rightarrow U$, une fonction qui associe chaque session s_i à un seul utilisateur $user(s_i)$. Cette fonction reste constante pour la durée de vie de la session
- $Rôle : S \rightarrow 2^R$, une fonction associant chaque session s_i à un ensemble de rôles $Rôle(s_i)$, avec : $Rôle(s_i) \subseteq \{ r | (\exists r' \leq r) \wedge (User(s_i), r') \in AU \}$
- Un ensemble de contraintes qui définit si certains éléments du modèle RBAC sont acceptables

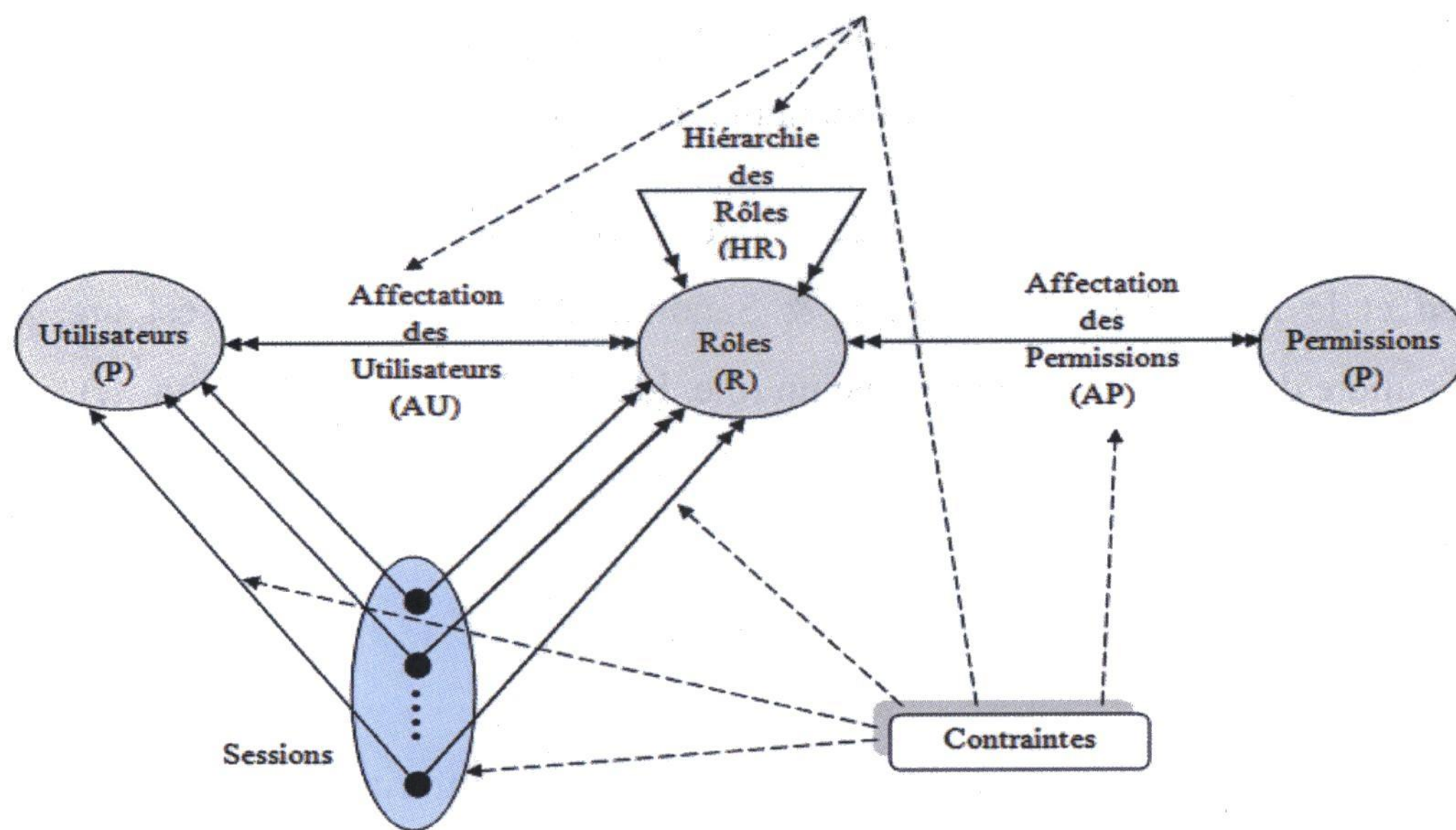


Figure 3.3 : Présentation de *RBAC₃* (Sandhu R. S., Coyne, Feinstein, & Youman, 1996)

2.1. Implantation de RBAC dans le Web

Dans cette section, nous présentons deux différentes approches, proposées par Park et Sandhu (Park & Sandhu, 2001), qui décrivent des stratégies afin d'obtenir les attributs des utilisateurs sur le web. Ces deux approches peuvent être appliquées en respectant deux différentes architectures, user-pull et server-pull. Ces dernières sont composées de trois éléments connus dans le monde du Web : client, serveur web et serveur de rôle. Un client se connecte à un serveur web via un navigateur web en utilisant le HTTP² (HyperText

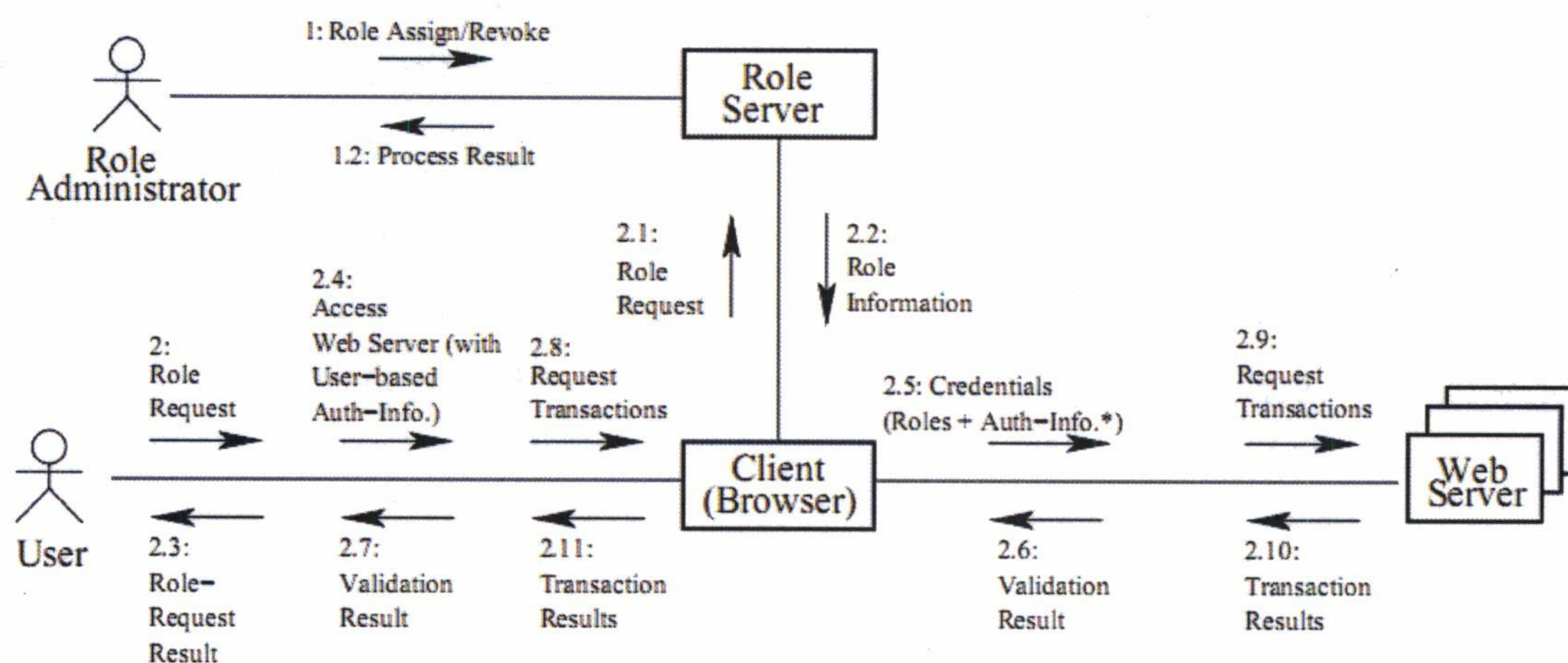
² Un protocole de communication client-serveur développé pour le World Wide Web (WWW), ce protocole peut fonctionner sur n'importe quelle connexion fiable.

Transfer Protocol). Le serveur de rôles est maintenu par un administrateur qui affecte des utilisateurs à des rôles au sein d'un domaine³ (Park & Sandhu, 2001).

Chaque architecture est basée sur deux différents modes qui sont le mode *user-based* et le mode *host-based* (Park & Sandhu, 1999). Un attribut est une propriété particulière d'une entité (par exemple un rôle, identité d'accès, habilitation ...).

2.1.1. Architecture user-pull

Dans cette architecture, l'utilisateur récupère les informations concernant son rôle depuis le serveur de rôle, puis il présente ces informations au serveur web comme expliqué sur la figure 3.4. Les rôles sont affectés aux utilisateurs au sein d'un domaine. À titre d'exemple, l'utilisateur Bob a le rôle d'un professeur au sein du domaine «<http://www4.uqo.ca/corps-professoral/prof>» et le rôle d'un directeur au sein du domaine «<http://www4.uqo.ca/direction-services/secretariat-général>». Le HTTP est utilisé afin d'assurer l'interaction du serveur des utilisateurs avec le navigateur web et le serveur web.



³ Un regroupement logique de ressources associé à un ou plusieurs annuaires (un type de base de données spécialisée permettant de stocker les informations d'utilisateurs de manière hiérarchique et offrant des mécanismes simples pour rechercher l'information).

Figure 3.4 : Architecture User-pull (Park & Sandhu, 2001)

Dans cette architecture, un utilisateur est censé récupérer les informations concernant son rôle depuis le serveur de rôles et les enregistrer sur sa machine physique qui contient les informations d'authentification (e.g numéro IP). Ce mode de connexion est appelé mode host-based. Par d'exemple, quand l'utilisateur Bob tente d'accéder à un serveur web, sa machine physique présente les informations nécessaires d'authentification et de rôle au serveur web qui effectue une vérification de ces informations, et par la suite, le serveur web utilise ce rôle pour le contrôle d'accès basé sur RBAC. Contrairement au mode user-based qui permet une haute mobilité de l'utilisateur. Ce dernier récupère les informations de son rôle depuis le serveur de rôle, et par la suite, il présente son rôle au serveur web via les informations d'authentification (e.g mot de passe). Après la vérification de ces informations, le serveur web utilise ce rôle pour le contrôle d'accès basé sur RBAC. Au niveau de ce mode de connexion, l'utilisateur peut utiliser n'importe quelle machine qui supporte le HTTP.

2.1.2. Architecture server-pull

Dans cette architecture, c'est le serveur web qui s'occupe de récupérer, en cas de besoin, le rôle de l'utilisateur depuis le serveur de rôles, et utilise par la suite ce rôle pour accorder l'accès. La figure 3.5 présente un scénario de cette architecture.

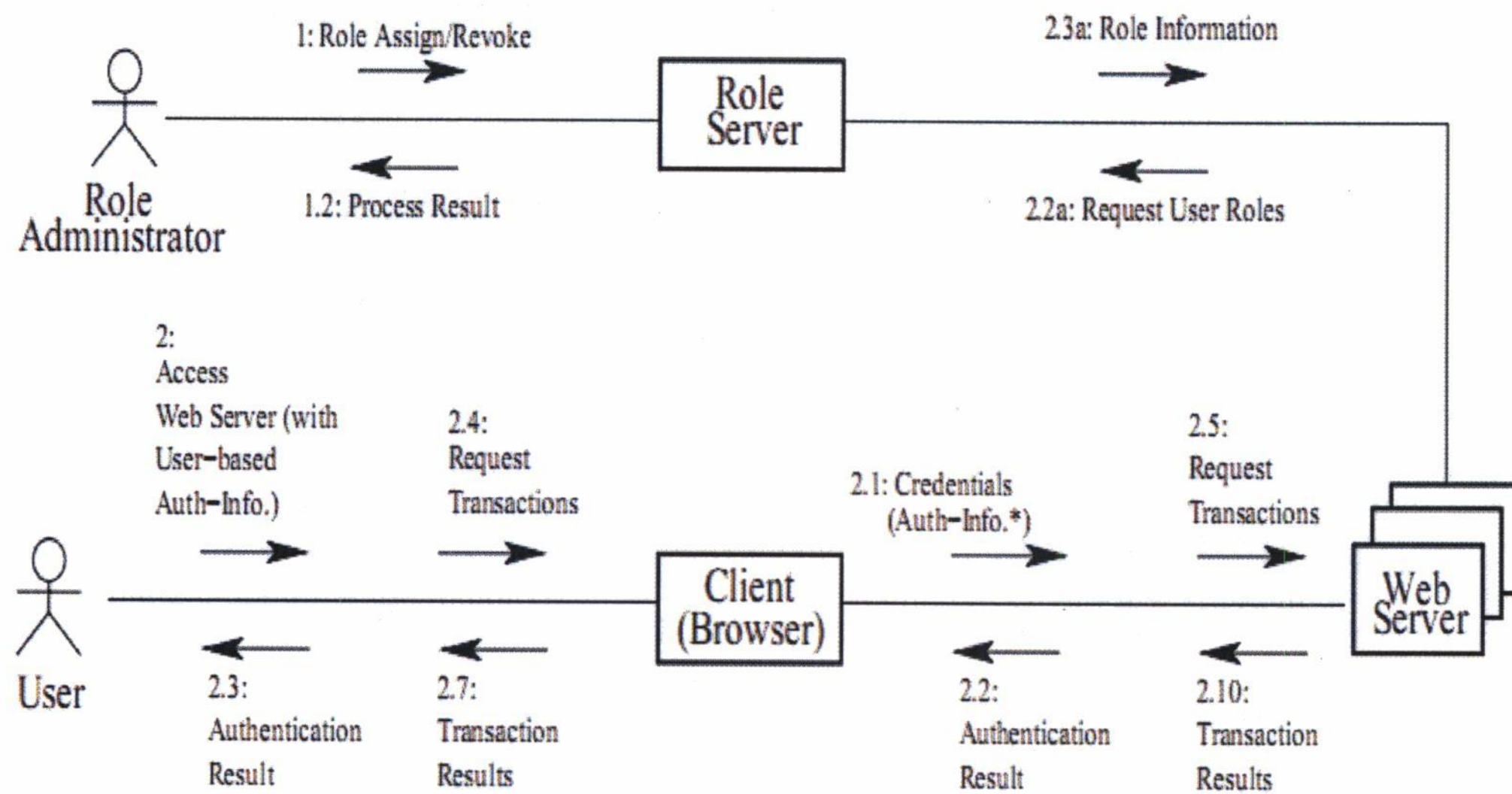


Figure 3.5 : Architecture server-pull (Park & Sandhu, 2001)

Dans l'architecture server-pull, l'utilisateur a seulement besoin des informations d'authentification, le mécanisme d'obtention de rôle est transparent pour l'utilisateur. Ce dernier se limite à présenter seulement les informations d'authentification de sa machine physique, telles que l'adresse IP. Ce mode de connexion doit être host-based. Cependant, le mode host-based n'est pas capable d'assurer la portabilité de l'utilisateur, ce qui a donné avantage au mode user-based. Au niveau de ce dernier, l'utilisateur présente ses informations d'authentification, tel que le mot de passe, au web serveur. Après l'authentification de l'utilisateur, le web serveur récupère le rôle de l'utilisateur depuis le serveur de rôle afin de l'utiliser pour accorder l'accès à l'utilisateur selon son rôle (RBAC).

2.1.3. Conclusions sur le modèle RBAC

Le modèle RBAC, avec ses nombreuses variations et adaptations, est approprié pour les organisations dans lesquelles le contrôle d'accès est effectivement basé sur les rôles des usagers. Un grand nombre d'organisations sont dans cette catégorie et, pour cette raison, RBAC est très utilisé en pratique. Toutefois, les besoins des organisations

nécessitent des modèles plus flexibles et, par conséquent, les modèles basés sur le langage XACML ont été inventés.

3. Langage XACML

3.1. Introduction

XACML (eXtensible Access Control Markup Language) est un langage de contrôle d'accès basé sur la spécification XML, qui a été standardisé par l'organisme OASIS (Moses, 2005). XACML définit à la fois ; 1) un langage de politiques de contrôle d'accès qui a pour but d'exprimer les politiques de contrôle d'accès de la forme :(qui peut faire quoi et quand), et 2) un protocole de type requête / réponse qui a comme objectif d'exprimer des requêtes pour savoir si un accès demandé par un sujet est autorisé.

XACML est un langage conçu pour supporter le modèle ABAC (Attribut Based Access Control).

Dans cette section, nous décrivons le langage XACML, les étapes pour évaluer une requête par rapport aux politiques XACML, ainsi que l'architecture et la spécification de XACML.

3.2. Description du langage XACML

La structure d'une politique dans XACML (voir figure 3.6) est basée sur trois éléments principaux : une cible, une ou plusieurs règles et l'identificateur de l'algorithme de combinaison de règles.


```

<Policy PolicyId="deny-test"
RulecombiningAlgId="rule-combining-algorithm:first-applicable"
<Description>structure de la politique</Description>
<Target>
<Subjects>.....</Subjects>
<Resources>.....</Resources>
<Actions>.....</Actions>
</Target>
<Rule RuleId="R1" Effect="Permit">.....</Rule>
<Rule RuleId="R2" Effect="Deny">.....</Rule>
</Policy>

```

Figure 3.6 : Structure d'une politique XACML

La décision d'une politique est définie par la combinaison de différentes règles en utilisant l'algorithme de combinaison de règles.

Cible (Target) définit l'ensemble des attributs auxquels une politique XACML s'applique, tel que sujets, ressources, actions et environnement, voir figure 3.7.



Figure 3.7 : Structure d'une cible

- Le sujet décrit les informations de l'utilisateur qui demande l'accès à une ressource. Un sujet peut être représenté par un identificateur, un groupe auquel il appartient ou un rôle

- La ressource décrit les attributs de l'objet auquel l'accès est demandé. Une ressource est caractérisée par un identificateur, des propriétés, le type de ressource et le niveau de sécurité
- L'action définit les attributs qui décrivent les fonctions qu'un utilisateur veut appliquer sur une ressource demandée. Une action est aussi définie par un identificateur et le nom de l'action à effectuer
- L'environnement concerne les attributs qui décrivent le contexte de l'application

Exemple d'une requête XACML :

L'étudiant Bob, qui appartient au groupe des étudiants de maîtrise, veut accéder aux notes de cours en mode lecture.

Pour cette requête, on peut distinguer les différents attributs qui possèdent différentes valeurs :

- Sujet : Bob
- Ressource : notes de cours
- Action : lecture

Règle

Dans le langage XACML, une règle est définie comme un ensemble de prédicats qui peut être évalué sur la base de son contenu. Une politique est composée d'une ou plusieurs règles. Une règle est composée principalement de trois composantes : une cible, un effet et une condition.

- Cible (Target) : permet de déterminer l'ensemble des attributs qui correspond au(x) requête(s), et pour lesquelles une règle doit être exécutée
- Effet : détermine la décision d'une règle. Cette décision peut être «*Permit*» ou «*Deny*»

- Condition : expression booléenne qui doit être évaluée pour qu'une règle soit attribuée son effet

Prenons l'exemple de la section précédente :

- Cible :
 - L'identificateur du sujet est « étudiant »
 - La ressource est les « notes de cours »
 - L'action est « lecture »
- Condition :
 - L'étudiant doit appartenir au groupe des étudiants de maîtrise
- Effet :
 - Autorisation de lecture

D'après cet exemple, on peut évaluer trois scénarios possibles :

1. Si un étudiant de maîtrise tente d'accéder aux notes de cours en mode écriture, cette règle sera évaluée et retournera la décision «*Deny*», car il n'y a pas d'autorisation explicite pour l'écriture.
2. Si un étudiant n'appartient pas au groupe étudiant de maîtrise, cette règle ne sera pas appliquée puisque la condition n'est pas satisfaite.
3. Si un étudiant de maîtrise demande d'accéder en mode écriture au « fichier de résultats », cette règle ne sera pas évaluée car elle traite seulement les « notes de cours » (cible).

Pendant l'évaluation d'une requête, l'identification des politiques appropriées se déroule en deux étapes :

- La comparaison de la cible de la requête avec la cible de la politique.
- La vérification des conditions qui sont définies par la règle de politique afin de déterminer la décision «*Permit*» ou «*Deny*».

Étant donné qu'une politique est composée de plusieurs règles, il est nécessaire de prendre en considération le cas où il y a différentes décisions menées par différentes règles dans une politique. Ceci explique l'existence des algorithmes de combinaison.

L'évaluation d'une requête par rapport à une règle peut donner quatre résultats différents, comme suit :

- *Permit* : autorise l'accès à une ressource donnée
- *Deny* : interdit l'accès à une ressource
- *Indeterminate* : quand la construction d'une *politique* ne permet pas d'aboutir à une décision finale, comme, quand un élément des attributs (sujet, ressource, action, etc.) est inconnu
- *Notapplicable* : la politique ne contient pas de règles appropriées pour traiter la requête

Ensemble de politique (policysset) : un ensemble de politiques est une association de plusieurs politiques, exprimé par trois composantes principales : cible, une ou plusieurs politiques et un algorithme de combinaison de politiques.

3.3. Algorithmes de combinaison

Dans cette section, nous décrivons quatre algorithmes de combinaison qui sont définis par le standard XACML 2.0 (Moses, 2005). Nous rappelons qu'un algorithme de combinaison est une procédure pour combiner la décision de plusieurs politiques ou règles.

- ***Permit-overrides*** : dans l'ensemble des politiques, s'il y a au moins une politique qui est évaluée à «*Permit*», alors la décision fournie par la combinaison sera un «*Permit*». Par contre s'il y a au moins une politique évaluée à «*Deny*», et toutes les autres politiques sont évaluées à «*NotApplicable*», alors la décision de combinaison sera «*Deny*». Autrement, si toutes les politiques sont jugées

«*NotApplicable*», alors la décision de combinaison sera évaluée à «*NotApplicable*».

- ***Deny-overrides*** : dans l'ensemble des politiques, s'il y a au moins une politique qui retourne l'effet «*Deny*», alors la décision fournie par la combinaison sera un «*Deny*». Si ce n'est pas le cas, et qu'il existe une politique qui est évaluée à «*Permit*», pendant que les autres politiques sont évaluées à «*NotApplicable*», alors la décision de combinaison sera «*Permit*». Autrement, si toutes les politiques sont jugées «*NotApplicable*», alors le résultat de la combinaison sera évalué à «*NotApplicable*».
- ***First-applicable*** : dans cet algorithme, l'évaluation des politiques est importante selon l'ordre des règles répertoriées dans l'ensemble des politiques. Pour une politique en particulier, si la condition est évaluée à «*Vrai*», alors la décision de la politique sera l'effet de la première règle («*Permit*» ou «*Deny*»). Par contre, si la condition n'est pas satisfaite par la requête, alors la règle suivante dans l'ordre doit être évaluée. Si aucune règle supplémentaire n'existe dans l'ordre, alors la décision de la combinaison est évaluée à «*NotApplicable*».
- ***Only-one-applicable*** : dans l'ensemble des politiques, si plusieurs politiques sont considérées comme applicable en vertu de leurs conditions, alors la décision de la combinaison est évaluée à «*Indeterminate*». Si ce n'est pas le cas, alors la décision de la politique sera celle de la règle applicable.

Exemple de politique XACML :

Soit «*Accès-Imprimante*», une politique qui contrôle l'accès aux imprimantes sur le réseau de l'UQO. Cette politique est décrite de la façon suivante :

- Tous les professeurs de l'UQO ont le droit d'utiliser les imprimantes publiques et privées
- Tous les étudiants qui étudient à l'université UQO ont le droit d'utiliser les imprimantes publiques

D'après cette politique, l'accès à une imprimante dépend du rôle de l'utilisateur, du type de l'imprimante (publique ou privée) et de l'action demandée, voir tableau 3.2.

| | Attributs | Valeurs |
|------------|----------------|------------|
| Sujets | Rôle | Professeur |
| | | Étudiant |
| | Institution | UQO |
| Ressources | Imprimante | Publique |
| | | Privé |
| Actions | Identificateur | Utiliser |

Tableau 3.2 : Exemple des valeurs d'une requête en format tabulaire

Afin de mieux comprendre la politique «Accès-Imprimante», nous décrivons les règles dégagées par cette politique. En langage naturel, le contenu du tableau précédant peut être exprimé de la manière suivante :

- Règle1 (R1) : un professeur qui appartient à l'institution UQO a le droit d'utiliser toutes les imprimantes
- Règle2 (R2) : un étudiant qui est inscrit à l'institution UQO a le droit d'utiliser les imprimantes publiques

La figure 3.8 présente les mêmes règles montrées dans le tableau 3.2 en forme d'arbre de décision.

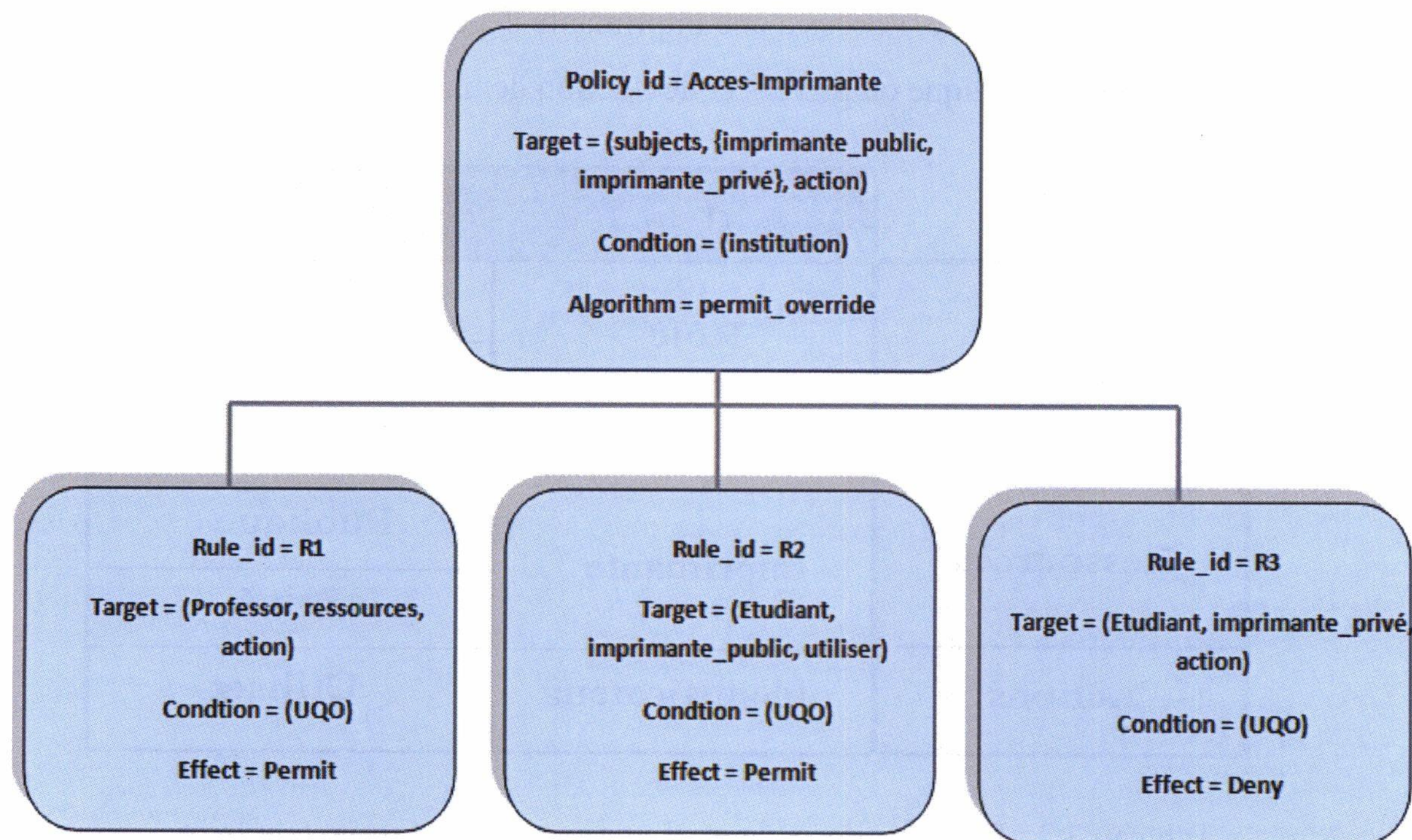


Figure 3.8 : Exemple d'une politique XACML

3.4. Architecture à la base de XACML

L'architecture globale à la base de XACML est composée de plusieurs entités collaborant entre elles (Keleta, Eloff, & Venter, 2005), comme illustré dans la figure 3.9. En plus de la syntaxe qu'elle apporte le langage XACML, ce dernier a permis, d'un côté, de concevoir un système indépendant de la plate-forme utilisée et, de l'autre côté, d'intégrer le système de contrôle d'accès dans les applications existantes. Nous présentons dans la suite de cette section les différentes entités de ce langage.

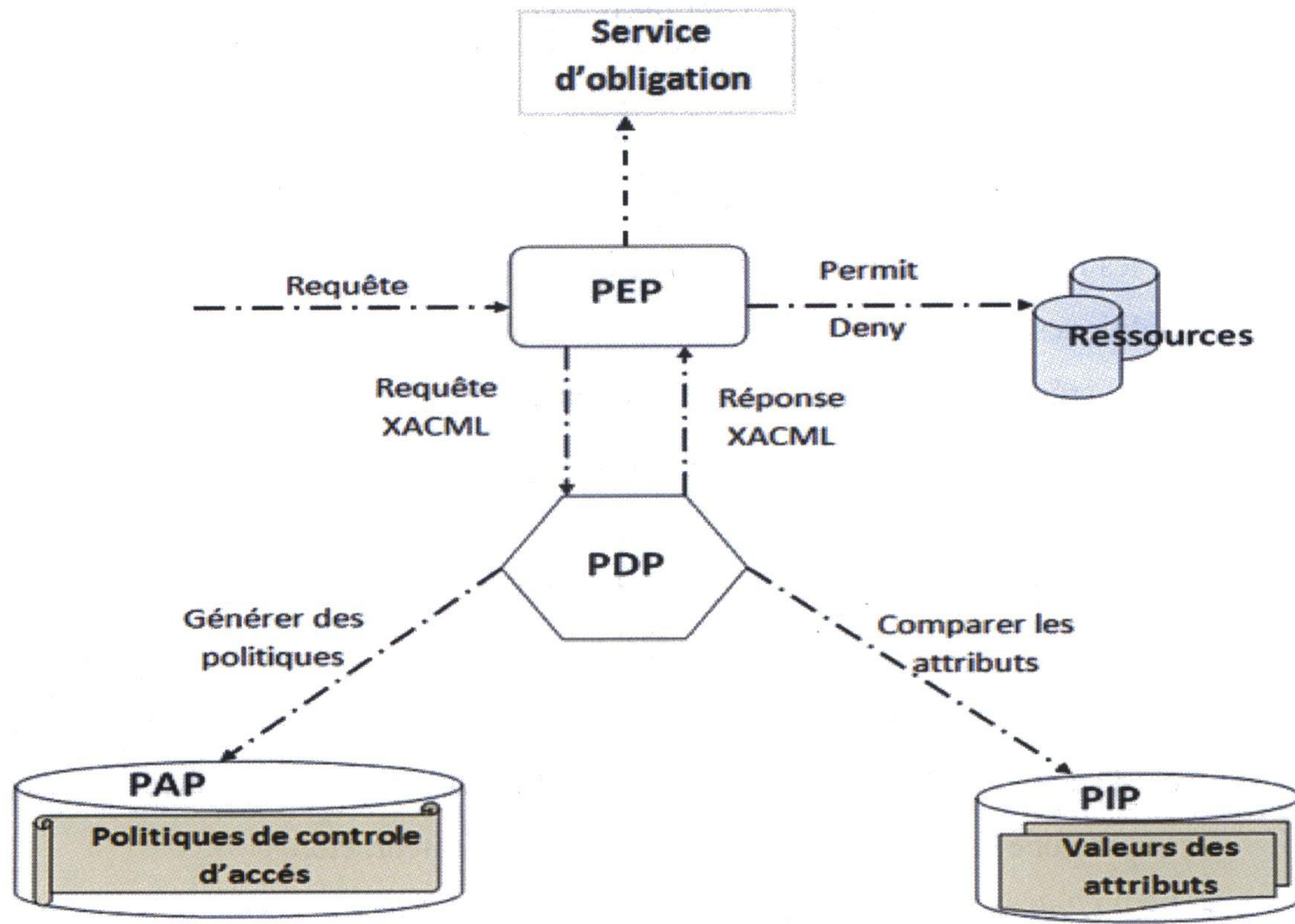


Figure 3.9 : Architecture XACML (Keleta, Eloff, & Venter, 2005)

Comme le montre la figure ci-dessus, l'architecture d'un système XACML est basée sur quatre composantes importantes :

- PEP (*Policy Enforcement Point*) : cette entité est le point d'application des politiques qui protègent les ressources. Cette entité reçoit une demande d'accès via des entités externes. Le PEP se charge d'extraire les attributs de la requête puis il génère une requête XACML à l'aide du gestionnaire de contexte (Context Handler) ; ce dernier a le rôle de transformer les requêtes dans un format compréhensible par les autres entités appelé contexte XACML. Le PEP s'occupe d'envoyer la requête générée aux autres entités afin d'avoir une évaluation de cette requête. Il s'assure également que toutes les obligations avec une décision d'autorisation sont exécutées. Une obligation est une action qui est

indiquée dans la politique de contrôle d'accès afin d'être exécutée conjointement avec une décision d'autorisation.

- PDP (*Policy Decision Point*) : cette entité est le point de décision de politique qui reçoit une demande XACML. Le PDP se charge de récupérer les politiques et les règles applicables à cette demande, à partir du point d'administration de politiques. Après avoir évalué la demande via les valeurs des attributs récupérées à partir du point d'exécution de politique (PEP), par rapport aux politiques applicables de contrôle d'accès, le PEP retourne une décision finale à l'entité PEP.
- PIP (*Policy Information Point*) : cette entité est la source d'information de politiques qui agit comme un serveur d'annuaire qui enregistre les informations des attributs afin de rendre ses valeurs accessibles par le PDP. Cependant, cette entité peut elle-même aller chercher des informations dans des sources externes comme une base de données, un annuaire d'utilisateur, etc.
- PAP (*Policy Administration Point*) : cette entité est le point d'administration de politique. Le PAP définit les politiques et les règles XACML de contrôle d'accès, puis il les stocke dans un serveur de base de données de politiques. En outre, le PAP effectue un contrôle régulier afin d'assurer l'unicité des identifiants des politiques.

4. Aperçu des techniques de vérification de politiques de sécurité

Dans cette section, nous présentons quelques techniques qui ont été développées pour la vérification et la détection des conflits ou des incomplétudes dans des ensembles de politiques de contrôle d'accès. Les approches utilisées par ces méthodes sont différentes, mais la finalité du travail reste la même.

Nous avons vu que, dans les environnements d'entreprises, il peut y avoir différentes méthodes de contrôle d'accès aux ressources. Selon le système de contrôle d'accès, l'administrateur peut définir différentes politiques de contrôles d'accès. La complexité d'un système de contrôle d'accès dépend de plusieurs facteurs, tels que le nombre de sujets, le nombre de ressources, les actions à exécuter, etc. En raison de ces facteurs, il est possible qu'un système de contrôle d'accès contienne des anomalies, telles que les incohérences ou incomplétudes, comme discuté dans l'introduction.

Un ensemble de politiques de contrôle d'accès est *incohérent* lorsqu'il existe des politiques incompatibles, c'est à dire qu'il y a des requêtes pour lesquelles le mécanisme de contrôle d'accès, après considération des politiques dans le cas spécifique, peut tout autant arriver à une décision de permission que d'interdiction. Un ensemble de politiques de contrôle d'accès est *incomplet* quand l'ensemble de politiques ne permet pas d'arriver à une décision concernant certaines requêtes. La littérature existante s'est surtout penchée sur la détection d'incohérences.

4.1. Vérification des politiques via une description logique

4.1.1. Introduction

Afin d'examiner les politiques de contrôle d'accès dans un système d'information utilisant XACML, Huang et F. Liu (Huang, Huang, & Liu, 2009) ont proposé une méthode basée sur un langage de Description Logique (DL). Leur méthode permet de transformer les politiques XACML vers une description logique et, par la suite, de détecter les conflits entre les politiques au sein d'une base de données ABox⁴.

Description Logique est une classe de langages logiques conçus pour la description des ontologies et bases de données. Ces langages permettent la description des concepts,

⁴ Un composant d'assertion qui décrit l'état associé à des instances de classe d'attributs, par exemple Bob est un professeur. Monsieur Bob est une instance de classe professeur.

ainsi que des relations entre les concepts et les individus au sein d'un domaine spécifique. Ils sont essentiellement des versions limitées de la logique du premier ordre (Baader, McGuinness, Nardi, & Patel-Schneider, 2003). Les langages de DL sont décidables en temps polynomial.

4.1.2. Spécification du langage

Dans ce langage, les auteurs ont introduit, en plus du concept d'hierarchie de rôle, un nouveau concept d'hierarchie de ressources. Dans un système de contrôle d'accès, l'autorisation positive d'une classe de ressources ne peut pas être propagée vers les ressources de bas niveau. Par contre, l'autorisation négative d'une classe de ressources peut être transmise vers les niveaux inférieurs. À titre d'exemple, le rôle post doc a une autorisation négative pour accéder à la ressource «fichier de notes», alors ce rôle aura une autorisation négative pour toutes les ressources situées dans un niveau inférieur par rapport à la ressource «fichier de notes».

Les règles d'autorisation dans une politique de contrôle d'accès sont représentées par le tuple $\langle EF, \langle S, R, A, EA \rangle \rangle$:

- EF : est l'effet de la règle, qui peut prendre deux valeurs $\{Permit, Deny\}$
- S : l'ensemble des rôles
- R : l'ensemble des ressources
- A : l'ensemble des actions
- EA : l'environnement de la règle

Prenons comme exemple le tuple $\langle Permit, \langle professeur, fichier de notes, lire, EA \rangle \rangle$. Cette règle spécifie une autorisation positive pour le sujet «professeur» qui peut exécuter l'action «lire» sur la ressource «fichier de notes» dans un environnement spécifique EA.

La hiérarchie des ressources est représentée par le tuple $\langle R, \leq \rangle$. La relation \leq est une relation d'ordre partiel sur l'ensemble des ressources R. Par exemple $\leq(R1, R2)$, signifie que la classe de ressources R1 est une sous classe de la classes de ressources R2.

De la même façon, la hiérarchie de rôles est représentée par le tuple $\langle S, \leq \rangle$. Le tuple $\leq(S1, S2)$ signifie que le rôle $S1$ est un sous rôle de $S2$.

Afin de contrôler la propagation des autorisations, les auteurs ont proposé trois règles qui doivent être respectées (Huang, Huang, & Liu, 2009).

- **Règle 1** : (la propagation d'une autorisation positive au niveau de la hiérarchie de rôles)

Si $\leq(S1, S2)$ et $\langle Permit, \langle S2, R1, A, EA \rangle \rangle$ alors $\langle Permit, \langle S1, R1, A, EA \rangle \rangle$;

- **Règle 2**: (la propagation d'une autorisation négative au niveau de la hiérarchie de rôles)

Si $\leq(S1, S2)$ et $\langle Deny, \langle S1, R1, A, EA \rangle \rangle$ alors $\langle Deny, \langle S2, R1, A, EA \rangle \rangle$;

- **Règle 3** : (la propagation d'une autorisation négative au niveau de la hiérarchie de ressources)

Si $\leq(R1, R2)$ et $\langle Deny, \langle S1, R1, A, EA \rangle \rangle$ alors $\langle Deny, \langle S1, R2, A, EA \rangle \rangle$;

4.1.3. Détection de conflits entre les politiques de sécurité

Dans cette section, nous discutons de deux scénarios de propagation d'autorisation qui peuvent causer des conflits par rapport aux politiques définies explicitement par le langage XACML, voir figure 2.11. Dans les deux scénarios, nous considérons que les politiques ont les mêmes valeurs de l'attribut action, ainsi que les politiques sont appliquées dans le même environnement.

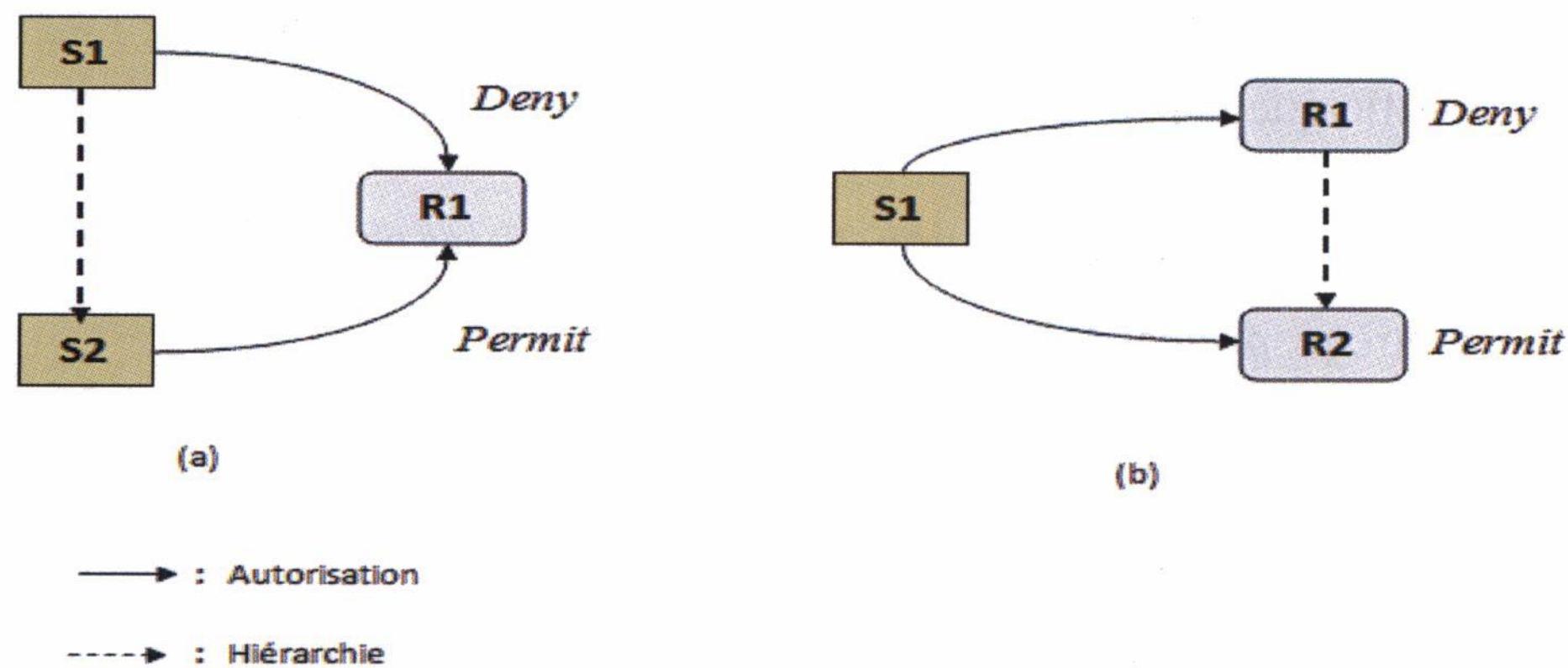


Figure 3.10 : Autorisation et hiérarchie des attributs (Huang, Huang, & Liu, 2009)

Scenario (a) :

La formalisation de ce scénario est décrite comme suit :

$\langle Deny, \langle S1, R1, A, EA \rangle \rangle$ et $\leq(S1, S2)$ et $\langle Permit, \langle S2, R1, A, EA \rangle \rangle$

Par l'application de la **règle 2** mentionnée plus haut, sur les deux premières règles de la formalisation, on peut avoir le résultat suivant $\langle Deny, \langle S2, R1, A, EA \rangle \rangle$, ce qui entre en conflit avec la troisième règle de la formalisation.

Scénario (b) :

La formalisation de ce scénario est décrite comme suit :

$\langle Deny, \langle S1, R1, A, EA \rangle \rangle$ et $\leq(R1, R2)$ et $\langle Permit, \langle S1, R2, A, EA \rangle \rangle$

Par l'application de la **règle 3** mentionnée plus haut, sur les deux premières règles de la formalisation, on peut avoir le résultat suivant $\langle Deny, \langle S1, R2, A, EA \rangle \rangle$, ce qui entre en conflit avec la troisième règle de la formalisation.

Cette méthode a un intérêt indéniable, mais dans notre travail nous avons choisi d'utiliser une méthode différente, pour détecter des incohérences directes dans des

ensembles de règles plutôt que des inférences entre règles d'autorisation avec hiérarchies de rôles et de données. Nos algorithmes ont les avantages de la simplicité et de l'efficacité

4.2. Vérification des politiques de sécurités par l'outil SEMPO

4.2.1. Introduction

Afin de bien gérer les politiques de sécurité et d'éviter les incohérences, au sein de l'outil EEM⁵ (Embedded Entitlement Manager), Y. Layouni (Layouni, 2010) a proposé un outil, sous le nom SEMPO (voir figure 3.11), qui permet d'analyser un ensemble de politiques de sécurité et de produire, pour l'administrateur de politiques, les informations dont il a besoin pour éviter ces incohérences. L'outil SEMPO extrait les politiques depuis l'outil EEM pour les vérifier en utilisant le moteur de vérification Alloy.

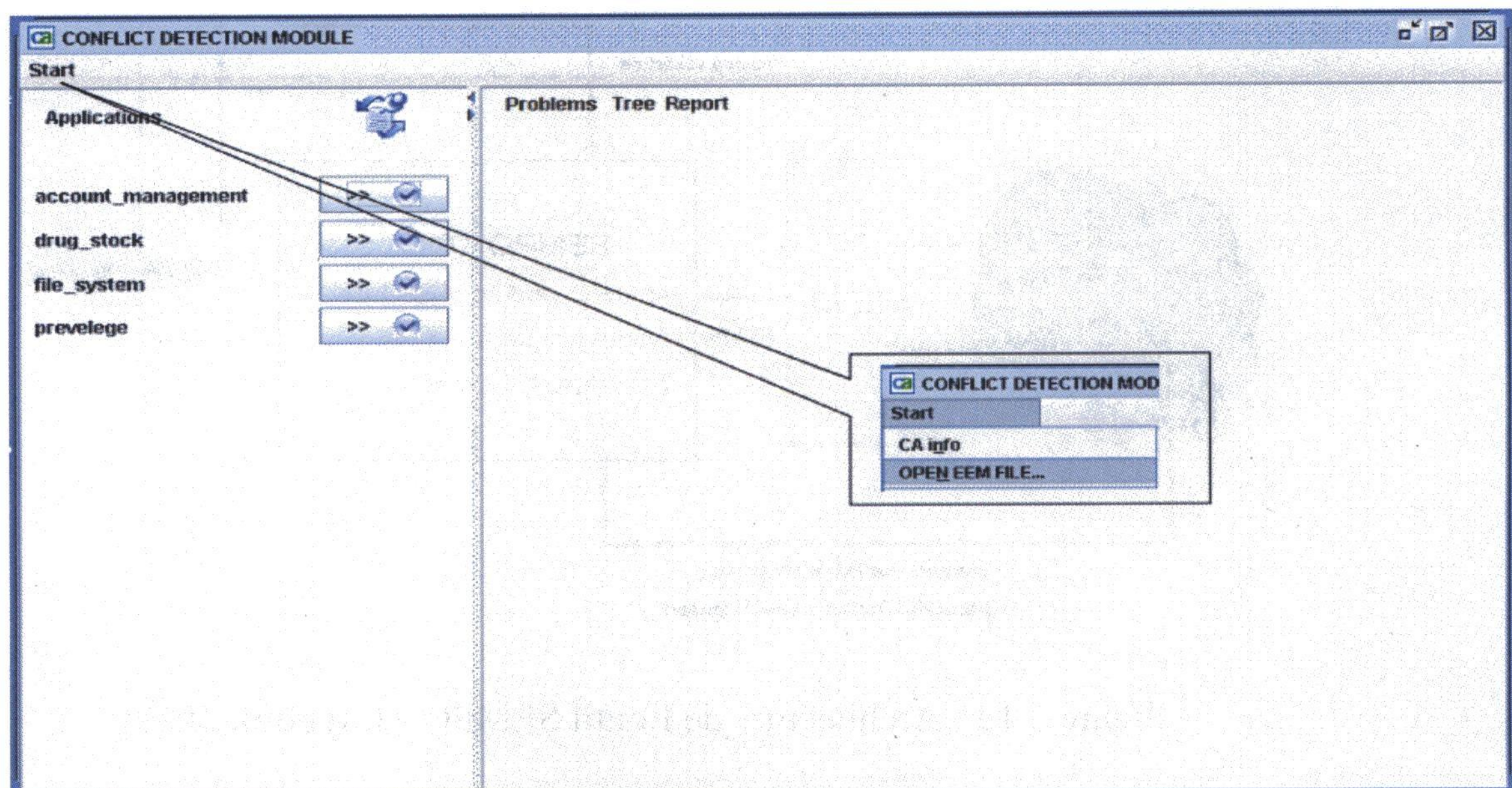


Figure 3.11 : Interface de SEMPO (Layouni, 2010)

⁵ Un outil de la compagnie CA World (Computer Associates World) qui permet de définir les politiques de sécurité ainsi que la gestion d'accès des identités.

4.2.2. Architecture de SEMPO

L'outil SEMPO travaille en connexion avec l'outil EEM. Les politiques sont extraites dans un format interne de EEM (fichier XML) et, par la suite, elles sont traduites vers un langage, basé sur la logique de premier ordre, qui va être exploité par le moteur de vérification Alloy afin d'identifier les anomalies. Le moteur de vérification Alloy est basé sur deux composantes essentielles : l'analyseur logique de Alloy et l'algorithme de satisfaction de SAT Solver. La figure 3.12 montre plus de détails sur l'architecture de l'outil SEMPO.

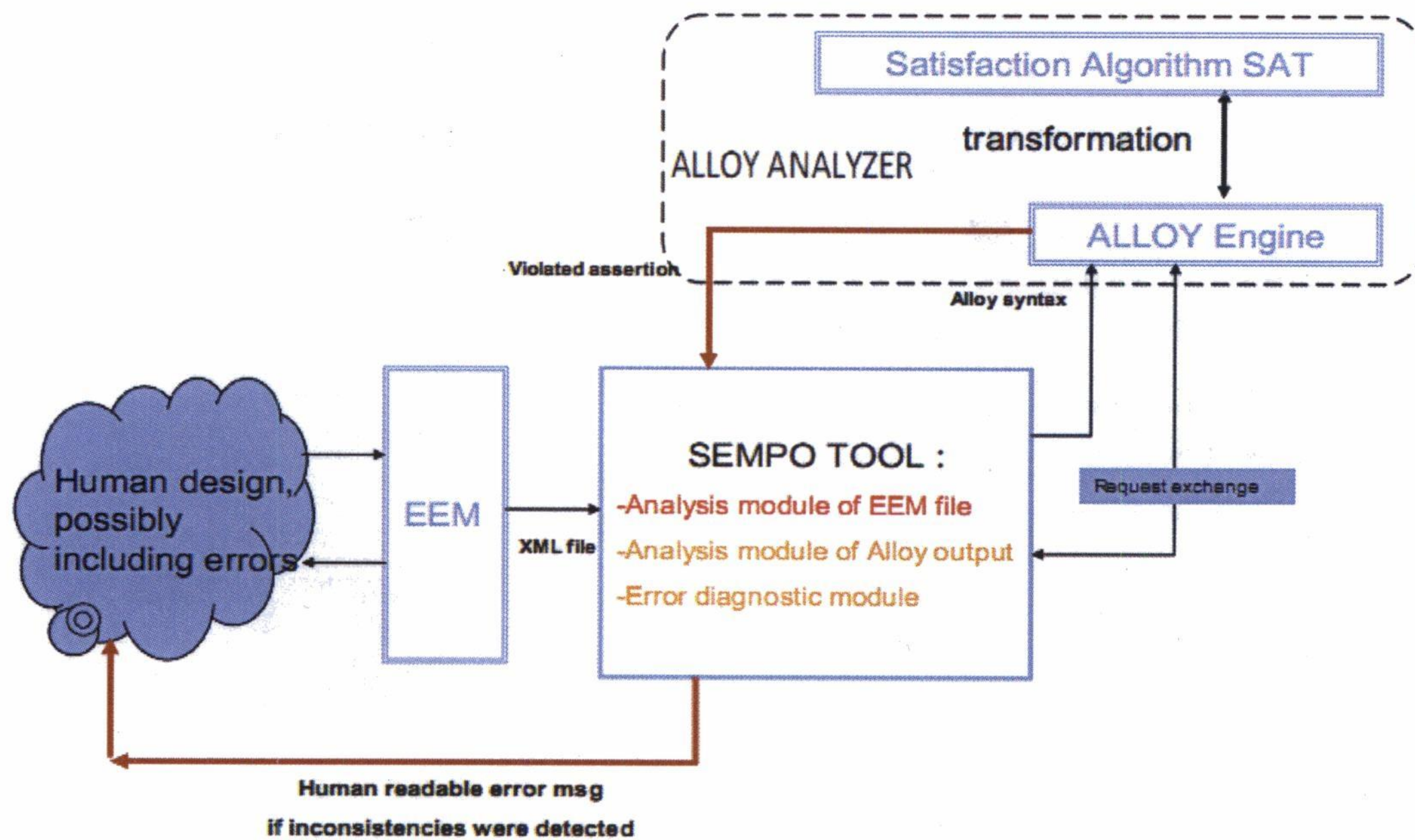


Figure 3.12 : Architecture de l'outil SEMPO (Layouni, 2010)

Alloy est un langage qui permet la vérification et la validation des modèles formels par le biais des formalismes graphiques et textuels. Alloy permet de modéliser les systèmes de contrôle d'accès en mettant l'accent sur les contraintes et les propriétés. Dans le laboratoire de recherche en sécurité informatique de l'Université du Québec en Outaouais, deux étudiants ont utilisé l'analyseur des contraintes, basé sur le langage Alloy, afin de

détecter d'éventuels conflits entre les politiques de sécurité existantes dans des ensembles de politiques de contrôle d'accès.

4.2.3. Formalisation de politiques

La spécification formelle des politiques représente la clé de l'outil pour la détection d'éventuels conflits dans un système de contrôle d'accès. En général, l'outil SEMPO prend en considération les politiques de la forme suivante :

$C \rightarrow$ permission pour effectuer une action sur un objet

Ou

$C \rightarrow$ interdiction pour effectuer une action sur un objet

Où C représente une condition.

Prenons comme exemple la politique suivante :

Tous les professeurs du département d'informatique peuvent lire les notes de cours du programme de génie informatique.

D'après cette politique, on peut dégager la condition C, qui doit être satisfaite afin de pouvoir lire les notes de cours.

C : un utilisateur doit être un professeur et appartenir au département d'informatique.

$C \rightarrow$ permission pour lire les notes de cours.

La notation formelle des politiques de contrôle d'accès est exprimée par la logique du premier ordre. La permission peut être précédée par le symbole \neg pour exprimer sa négation.

4.2.4. Détection de conflits

La stratégie de détection de conflits utilisée par SEMPO consiste à traduire les politiques d'EEM dans le langage de logique de premier ordre Alloy, puis exécuter l'outil

Alloy ANALYSER qui est capable de détecter les incohérences dans les politiques, et finalement, afficher des diagnostics montrant les incohérences.

Prenons l'exemple suivant qui montre la stratégie de détection de conflits au sein du moteur de vérification Alloy.

- P1 : un post_doc n'a pas le droit de lire ni d'écrire dans les fichiers de notes (F_notes) des étudiants en informatique
- P2 : les professeurs (prof) peuvent écrire dans les fichiers de notes des étudiants en informatique
- P3 : tous les employés du département informatique (dep_info) peuvent lire les fichiers de notes des étudiants en informatique

Chaque politique contient une condition qui doit être satisfaite. Avant de procéder à l'analyse des politiques, ces dernières sont traduites dans le langage de logique du premier ordre comme suit :

- P1 : $\forall a, b (\text{Utilisateur}(a) \wedge \text{Groupe}(a, \text{post_doc}) \wedge \text{F_notes}(b) \rightarrow \neg \text{Permis}(a, \text{écrire ou lire}, b))$
- P2 : $\forall a, b (\text{Utilisateur}(a) \wedge \text{Groupe}(a, \text{prof}) \wedge \text{F_notes}(b) \rightarrow \text{Permis}(a, \text{écrire}, b))$
- P3 : $\forall a, b (\text{Utilisateur}(a) \wedge \text{Attribut}(\text{dép_info}, a, \text{Zone}) \wedge \text{F_notes}(b) \rightarrow \text{Permis}(a, \text{lire}, b))$

Pour chaque politique, l'autorisation aura effet si la conjonction des prédicats est satisfaite.

- Utilisateur (a) : ce prédicat est vrai, si a est un utilisateur
- Groupe (a, post_doc) : ce prédicat signifie que l'utilisateur a appartient au groupe post_doc
- F_notes(b) : le type de ressource à exploiter

- Attribut (dép_info, a.Zone) signifie que l'attribut Zone de l'utilisateur <a> doit être égal à dep_info

Dans cet exemple, il n'existe pas de contradiction car les politiques définies ont des conditions indépendantes. Or, supposons que le système a défini les propriétés suivantes pour l'utilisateur Bob :

Utilisateur (Bob) \wedge Groupe (Bob, post_doc) \wedge Attribute (dep_info, Bob.Zone)

On constate que l'utilisateur Bob satisfait les conditions P1 et P3, ce qui génère en occurrence un conflit ; selon la politique P1, Bob ne peut pas lire le fichier F_notes, tandis que selon la politique P3, l'utilisateur Bob peut lire le fichier F_notes.

La règle P1 donne comme autorisation \neg Permis (a, écrire ou lire, b) et la règle P3 donne comme autorisation Permis (a, lire, b). Par l'application de la conjonction des deux politiques, on a :

$$\neg \text{Permis (a, écrire ou lire, b)} \wedge \text{Permis (a, lire, b)}$$

$$(\neg \text{Permis (a, écrire, b)} \vee \neg \text{Permis (a, lire, b)}) \wedge \text{Permis (a, lire, b)}$$

Afin de détecter le conflit et de déterminer les attributs qui entrent en jeu, l'auteur a défini l'axiome suivant :

$$\neg \text{Permis (a, c, b)} \wedge \text{Permis (a, c, b)} \rightarrow \text{Conflit}$$

Si un utilisateur (a) détient, en même temps, une permission et une interdiction d'effectuer une action (c) sur un objet (b), alors il existe un conflit.

4.2.5. Conclusion

La technique adoptée par cet outil semble intéressante pour de petits ensembles de politiques de contrôle d'accès. L'inconvénient majeur de cette technique réside dans le fait que l'algorithme de satisfaction booléenne utilisé par Alloy est de complexité

exponentielle, et par conséquent l'utilisation de cet algorithme ne sera pas performante pour de grands ensembles de politiques.

4.3. Analyse des politiques de sécurité par l'outil EXAM

4.3.1. Introduction

L'intégrité et la compatibilité des politiques sont souvent des exigences primordiales. Ces exigences sont plus difficiles à satisfaire lorsque les parties d'une organisation, avec différentes politiques de sécurité, doivent participer à des applications collaboratives qui impliquent la croissance du nombre de politiques. Une étape importante dans l'intégration des politiques est d'analyser la similarité des politiques. L'article (Lin, Rao, Bertino, Li, & Lobo, 2010) a proposé un outil sous le nom d'EXAM, ce dernier permet l'analyse des politiques exprimées par le langage XACML.

EXAM est un environnement complet d'analyse et de gestion des politiques de contrôle d'accès. Il prend en charge l'acquisition, l'édition et la récupération de politiques. En outre, il permet l'analyse des propriétés, l'analyse de la similarité et l'intégration des politiques (Lin, Rao, Bertino, Li, & Lobo, 2010).

4.3.2. Architecture de l'outil EXAM

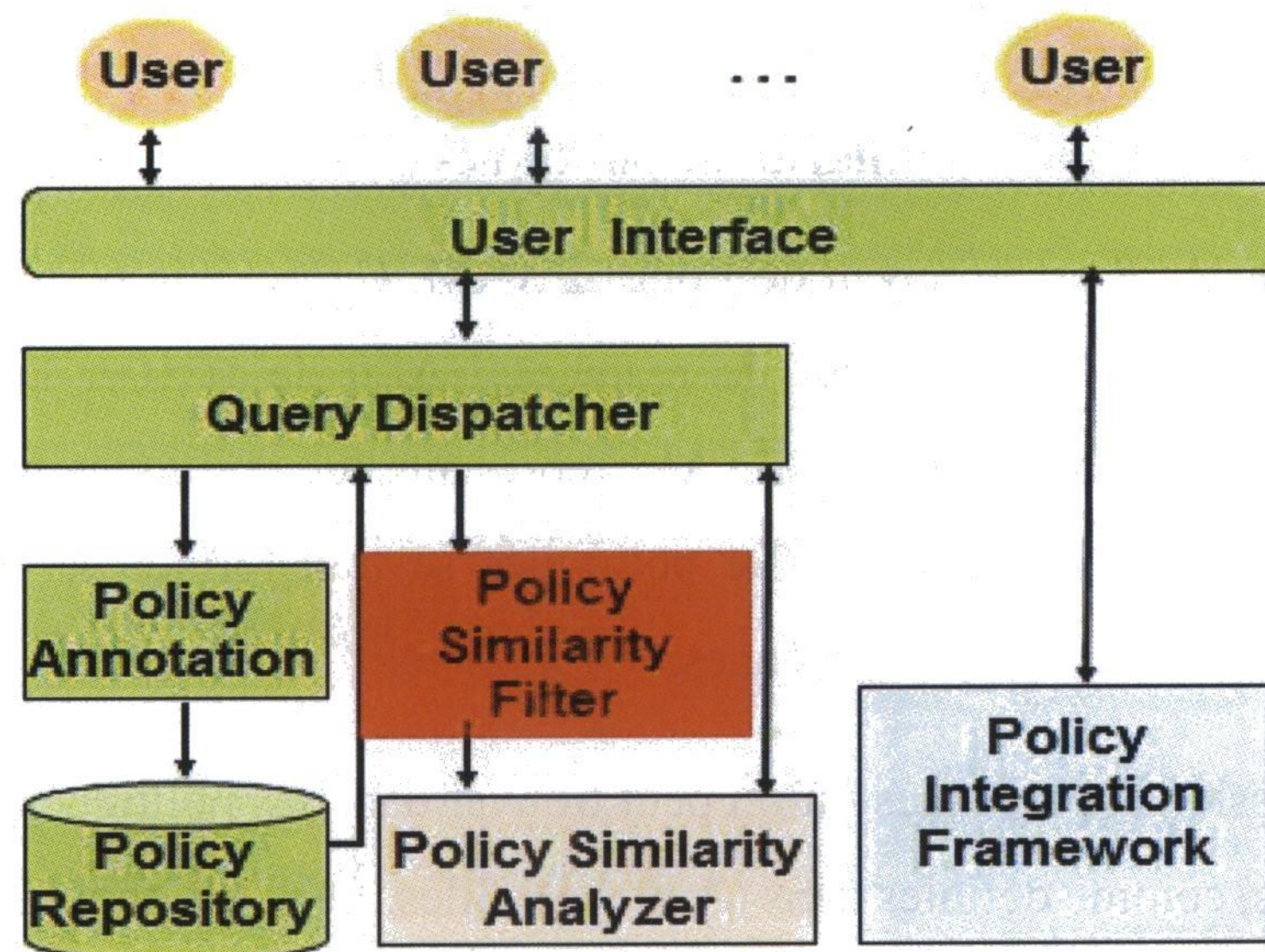


Figure 3.13: Architecture de l'outil EXAM (Lin, Rao, Bertino, Li, & Lobo, 2010)

L'architecture de l'outil EXAM est composée de trois niveaux :

- Premier niveau : représente l'interface d'utilisateur. Ce dernier permet la réception des politiques et des requêtes envoyées par des utilisateurs et retourne, par la suite, les résultats adéquats pour ces requêtes
- Deuxième niveau : joue le rôle d'un répartiteur de requêtes. Cette composante gère les requêtes reçues par l'interface d'utilisateur afin de les transmettre vers le module d'analyse approprié
- Dernier niveau : est la composante essentielle de l'outil EXAM. Ce niveau est basé sur quatre modules (annoteur de politiques, filtre de politiques, analyseur de similarité de politiques et l'intégrateur de politiques) qui ont différentes fonctionnalités.

Dans la suite de cette section, nous discutons du module qui permet l'analyse de similarité de politiques. L'approche utilisée dans ce module pour détecter l'existence des

conflits est différente de notre approche de vérification de politiques, mais la finalité reste la même.

4.3.3. Analyseur de similarité de politiques (ASP)

Le module ASP est le noyau de l'outil EXAM. Ce module permet l'analyse des requêtes envoyées par le distributeur de requêtes. Afin de vérifier la cohérence des politiques de contrôle d'accès, l'outil EXAM représente les politiques par des formules booléennes et le problème d'analyse de politiques se transforme, par la suite, en un problème d'analyse de formules booléennes.

Les expressions booléennes qu'on peut trouver dans les politiques sont classifiées en cinq catégories, comme définies dans (Agrawal, Giles, Lee, & Lobo, 2005) :

- Catégorie 1 : contrainte d'égalité d'une variable.
 $x = c$, tel que x est une variable et c est une constante.
- Catégorie 2 : contrainte d'inégalité d'une variable.
 $x \triangleright c$, tel que x est une variable, c est une constante et $\triangleright \in \{<, \leq, >, \geq\}$.
- Catégorie 3 : contrainte linéaire d'une valeur réelle.
 $\sum_{i=1}^n a_i x_i \triangleright c$, tel que x_i est une variable, c_i est une constante et $\triangleright \in \{=, <, \leq, >, \geq\}$.
- Catégorie 4 : contrainte d'expression régulière.
 $s \in L(r)$ ou $s \notin L(r)$, tel que s est une variable de type chaîne de caractères, et $L(r)$ est le langage généré par l'expression régulière r .
- Catégorie 5 : contrainte composée d'expressions booléennes.
Cette catégorie contient les contraintes obtenues par la combinaison des contraintes booléennes qui appartiennent aux catégories citées précédemment. Les opérateurs de cette combinaison sont \vee, \wedge, \neg . En utilisant l'opérateur \neg , on peut représenter la contrainte d'inégalité $x \neq c$ par l'expression $\neg(x=c)$.

Le module ASP détermine toutes les affectations de variables qui peuvent satisfaire les formules booléennes correspondant à une ou plusieurs politiques, ainsi que les affectations des variables qui mènent à différentes décisions pour différentes politiques. L'idée principale du module ASP est de combiner les fonctionnalités de la technique de ratification de politiques (Agrawal, Giles, Lee, & Lobo, 2005) et la technique de MTBDD (Multi Terminal Binary Décision Diagram) (Fisler, Krishnamurthi, Meyerovich, & Tschantz, 2005).

4.3.4. Architecture de l'analyseur de similarité de politiques (ASP)

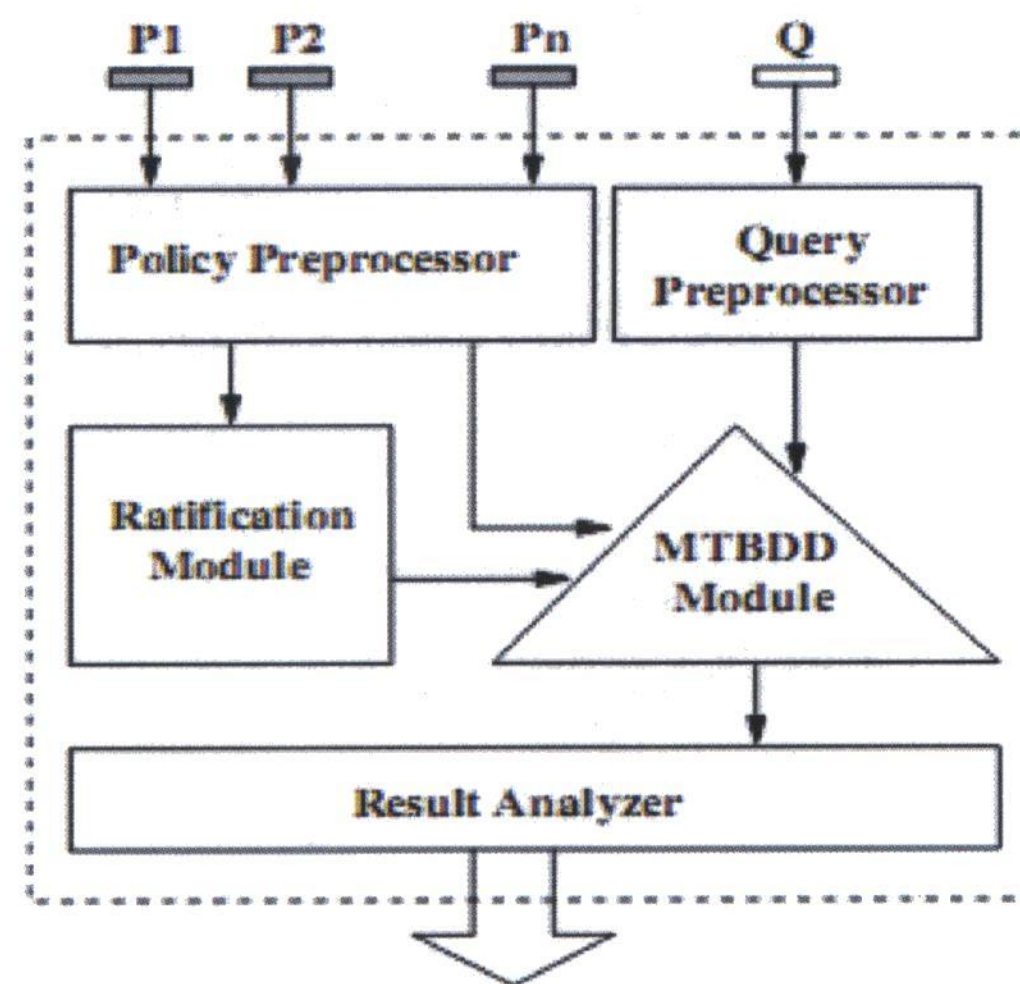


Figure 3.14 : Architecture du module ASP (Lin, Rao, Bertino, Li, & Lobo, 2010)

Afin de procéder à la vérification d'un ensemble de contrôle d'accès, le processus d'analyse comporte trois phases qui permettent la détection d'une telle anomalie ; la représentation de politiques, la comparaison de politiques et, finalement, une stratégie de traitement de requêtes.

Représentation de politiques

Le préprocesseur transforme une politique d'entrée au maximum en deux expressions booléennes de la catégorie 5 qui correspondent aux effets permis ou interdit. Ces expressions booléennes, représentées par un MTBDD, sont composées d'autres

expressions booléennes atomiques qui appartiennent généralement aux quatre premières catégories dont on a discuté dans le paragraphe précédent.

La structure d'un MTBDD est représentée par un graphe dirigé acyclique. Les nœuds internes représentent les expressions booléennes atomiques et les terminaux représentent les effets, Permit(P), Deny(D) et NotApplicable(N).

Comparaison de politiques

Afin de comparer deux politiques, les MTBDD générés par ces deux politiques sont combinés pour former un CMTBDD (Combined Multi Terminal Binary Décision Diagram) par le biais d'une opération binaire appelée Apply (Clarke, Fujita, McGeer, & Yang, 1997). Premièrement, on considère le CMTBDD construit par deux différentes politiques, voir figure 3.15. L'opération Apply est une opération récursive qui parcourt simultanément deux MTBDD en commençant par le nœud racine.

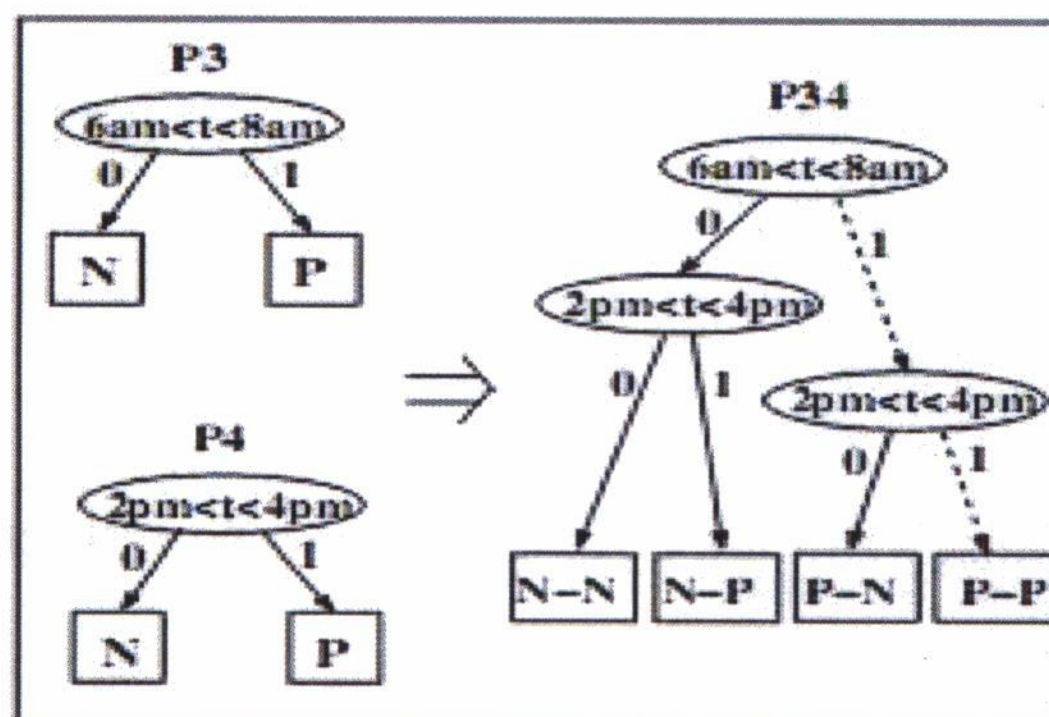


Figure 3.15 : Exemple d'un CMTBDD (Lin, Rao, Bertino, Li, & Lobo, 2010)

Dans la figure 3.15, nous voyons les MTBDD des politiques P3 et P4, et leur CMTBDD représenté par la politique P34, ce dernier est construit par l'opération Apply. La politique P3 permet d'accéder pendant l'intervalle de temps 6am à 8am, alors que la politique P4 permet l'accès durant la période 2pm à 4pm. Étant donné que ces deux intervalles de temps sont disjoints, le chemin indiqué par une ligne en pointillés dans leur

CMTBDD ne devrait pas exister, c'est à dire, qu'aucune requête ne peut satisfaire à ces deux conditions.

Traitement de requêtes

L'analyse des requêtes de politiques est effectuée sur la base des MTBDD et CMTBDD. Pour le même ensemble de politiques, le système a seulement besoin de construire le MTBDD et le CMTBDD de cet ensemble afin de le stocker dans la base de stratégies pour le traitement des requêtes.

Chaque requête q dans ce module est composée de trois types de contraintes (Lin, Rao, Bertino, Li, & Lobo, 2010) B_q , e_q et f_q .

- B_q : une expression booléenne sur un attribut de la requête $Attr_q$
- e_q : l'effet désiré sur la requête
- f_q : une contrainte sur un ensemble de requêtes

Le processus d'analyse de requête comprend trois étapes. Pour une requête donnée, il faut tout d'abord normaliser l'expression booléenne B_q sur les attributs $Attr_q$, en mettant les attributs en correspondance avec les nœuds unifiés déjà existants et, par la suite, on construit le MTBDD de la requête. Après avoir défini le MTBDD de requête, ce dernier est combiné avec le MTBDD ou le CMTBDD des politiques interrogées, ce qui génère une structure temporaire de CMTBDD de requête. À ce niveau, l'outil est en mesure de trouver les requêtes qui satisfont les contraintes B_q et e_q , en utilisant le modèle de vérification.

4.3.5. Conclusion

Les méthodes basées sur les arbres binaires MTBDD (Multi Terminal Binary Decision Diagram) peuvent être utiles pour la détection des conflits grâce à la rapidité de leurs parcours. Cependant, la technique adoptée dans l'outil EXAM n'est pas très expressive et l'interprétation des arbres binaire reste difficile pour l'identification des anomalies et la détection des incohérences.

Le but de l'outil EXAM est de répondre à des questions concernant des combinaisons de politiques, tandis que le but de notre outil est de déterminer si des ensembles de politiques contiennent des incohérences. Il est aussi important d'observer que l'outil EXAM ne détecte pas les incomplétudes, ce qui est une des fonctionnalités de base de notre outil.

5. Algorithme de classification

Dans cette section, nous présentons une étude comparative entre trois algorithmes connus dans le domaine de l'exploration des données ainsi que les étapes principales pour construire un arbre de décision pour chaque méthode. D'abord, nous présentons l'algorithme ID3 qui a été proposé en 1986 par J. Ross Quinlan (Quinlan, 1986). Ensuite, nous discutons l'algorithme CHAID (Chi-square Automatic Interaction Detection) qui a été inventé par Kass en 1980 (Rakotomalala R. , 2005), pour construire les arbres de décision à partir d'un ensemble de données. Finalement, nous parlons de l'algorithme C4.5 qui a vu le jour en 1993 par J. Ross Quinlan de l'Université de Sydney (Quinlan, 1993), et qui traite aussi du même type de données comme les algorithmes mentionnés précédemment. Dans la suite, nous allons utiliser un échantillon de données pour appliquer ces trois algorithmes et discuter les résultats.

5.1. Algorithme ID3

5.1.1. Introduction

ID3 est un algorithme d'apprentissage qui permet la génération d'un arbre de décision à partir d'un ensemble de données. La construction d'un arbre de décision, en utilisant l'algorithme ID3, consiste à effectuer une recherche descendante à travers tout l'ensemble de données pour sélectionner l'attribut adéquat à chaque nœud de l'arbre. La sélection d'un attribut est basée sur les valeurs de l'entropie et l'information de gain qui se

calculent en utilisant des formules mathématiques. Dans la suite de cette section, nous discuterons les formules utilisées par cet algorithme.

5.1.2. Arbre de décision

Le processus de construction d'un arbre de décision en utilisant ID3 est composé de trois étapes essentielles.

- À partir d'un ensemble de données, calculer l'entropie pour chaque attribut
- Calculer l'information de gain pour chaque attribut
- Sélectionner l'attribut qui a une valeur de gain maximale

5.1.3. Entropie

Cette valeur représente la mesure de l'information qui caractérise l'impureté d'un ensemble de données arbitraire. Si un attribut contient un nombre d'instance C , alors l'entropie S relative à ce nombre d'instance est calculée de cette façon :

$$Entropy(S) = \sum_{i=1}^C -P_i \log_2 P_i$$

P_i Représente la probabilité que S appartienne à la classe d'instance i

5.1.4. Information de gain

Ce paramètre mesure la réduction d'entropie pour un attribut A en divisant l'ensemble d'instance relative à cet attribut. L'information de gain, $Gain(S, A)$ d'un attribut A , par rapport à cette collection d'instance S , est définie de la manière suivante :

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{valeurs}(A)} \frac{S_v}{S} Entropy(S_v)$$

Valeurs(A) représente l'ensemble de toutes les valeurs possibles pour l'attribut A , et S_v est le sous-ensemble de S pour lequel l'attribut A possède des valeurs v . L'algorithme utilise cette mesure pour classifier les attributs et construire l'arbre de décision où chaque nœud contient l'attribut possédant la plus grande valeur de l'information de gain parmi l'ensemble d'attributs non considérés dans l'arborescence à partir de la racine.

5.2. Algorithme CHAID

5.2.1. Introduction

L'algorithme CHAID est l'une des méthodes les plus connues pour la classification des données et la construction des arbres de décision. Cet algorithme, particulièrement adapté à l'analyse de grands ensembles de données, permet la construction des arbres non binaires dont chaque racine ou nœud peut avoir plus que deux branches.

Dans la suite de cette section, nous adoptons la notation suivante :

X représente l'attribut classe pouvant avoir K modalités.

Y représente un descripteur pouvant avoir L modalités.

5.2.2. Tableau de calcul

Cette étape consiste à créer, à partir de l'ensemble de données, des prédicteurs catégoriels en divisant les distributions en un nombre de catégories avec un nombre approximativement égal d'observations. La connaissance de X améliore la connaissance des valeurs de Y .

| Y/X | X_1 | X_l | X_L | Σ |
|-------|-------|-------|-------|----------|
| Y_1 | | : | | |
| | | : | | |

| | | |
|----------|----------------------|----------|
| Y_k | n_{kl} | $n_{k.}$ |
| Y_K | \vdots | |
| Σ | $n_{.l}$ | n |

Mesure d'association

Pour déterminer la pertinence de la variable dans la segmentation, CHAID utilise le paramètre Khi-2 d'écart à l'indépendance. Ce paramètre permet de comparer les valeurs observées avec les valeurs théoriques lorsque X et Y sont indépendants. La valeur de ce paramètre varie entre 0 et $+\infty$

$$\chi^2 = \sum_{k=1}^K \sum_{l=1}^L \frac{\left(n_{kl} - \frac{n_{k.} \times n_{.l}}{n} \right)^2}{\frac{n_{k.} \times n_{.l}}{n}}$$

5.2.3. T de Tschuprow

Afin de normaliser la valeur de Khi-2, qui avantage souvent les descripteurs ayant un nombre élevé de modalités, l'algorithme CHAID applique le t de Tschuprow qui représente le nombre de degrés de liberté. La valeur de t peut varier dans l'intervalle [0, 1]

$$t^2 = \frac{\chi^2}{n \times \sqrt{(K - 1) \times (L - 1)}}$$

5.3. Algorithme C4.5

5.3.1. Introduction

Comme mentionné dans l'introduction, notre méthode se base sur l'utilisation de l'algorithme de classification C4.5 pour détecter les incohérences et incomplétudes dans

les politiques de contrôle d'accès. Cette section est dédiée à une explication de cet algorithme.

Les algorithmes de classification ont suscité un intérêt considérable à la fois dans le domaine des algorithmes d'apprentissage et dans celui de l'exploration des données (data mining). Plusieurs algorithmes ont été proposés dans ces domaines et, parmi ces algorithmes, il existe l'algorithme de classification C4.5 que nous discutons en détail dans ce chapitre.

C4.5 est un algorithme qui a été proposé en 1993 par Ross Quinlan de l'Université de Sydney (Quinlan, C4.5: Programs for Machine Learning, 1993). Cet algorithme représente une extension de l'algorithme ID3 qui fut également proposé par Ross Quinlan en 1986 (Quinlan, Induction of Decision Trees, Machine Learning, 1986). L'algorithme C4.5 est basé sur le principe de construction d'arbres de décision afin de résoudre certains types de problèmes qui se présentent dans l'exploration des données.

Nous verrons, dans le chapitre V, comment cet algorithme peut être utilisé pour détecter l'incohérence et l'incomplétude dans un ensemble de politiques définies par l'outil IAM.

5.3.2. Arbre de décision

Le processus de construction d'un arbre de décision par l'algorithme de classification C4.5 est basé sur deux étapes essentielles. Premièrement, l'assemblage de toutes les valeurs des attributs qui permettent la construction de l'arbre. Deuxièmement, la finalisation de l'arbre de décision en adoptant une approche heuristique⁶ pour l'élagage des branches (Mazid, Shawkat, & Tickle, 2010). Dans un arbre de décision, on trouve :

⁶ Une approche heuristique est une approche basée sur un algorithme qui fournit rapidement une solution valable, mais pas nécessairement optimale.

- Le nœud racine de l'arborescence, qui considère toutes les valeurs et sélectionne les attributs qui ont la signification la plus importante
- L'information d'un attribut qui est passée au nœud suivant, ce dernier s'appelle aussi « nœud de branche », jusqu'à l'atteinte d'un nœud feuille qui permet de prendre une décision.
- Les règles qui sont définies par des chemins qui relient un nœud racine avec les nœuds feuilles.

5.3.3. Algorithme de construction d'arbre

Dans l'algorithme C4.5, le processus de construction de l'arbre de décision commence par un nœud unique qui représente l'ensemble de toutes les données (Zaiane, 1999). Si toutes les instances dans un ensemble de données appartiennent à la même classe, alors le nœud devient une feuille étiquetée par cette classe. Sinon, l'algorithme C4.5 sélectionne l'attribut adéquat en fonction des critères suivants (Kotsiantis, 2007) :

1. Pour chaque attribut A, trouver l'information de gain normalisée qui permet la répartition sur A
 - 1.1. Soit A_{best} l'attribut qui a la valeur la plus élevée de l'information de gain
2. Créer un nœud de décision qui permet de répartir l'attribut A_{best} .
3. Répéter les étapes précédentes pour le prochain attribut, puis définir les nœuds obtenus comme des nœuds fils du nœud courant.

Soit S un ensemble de données qui contient deux classes P et N. L'information de gain pour un attribut A est calculée de la façon suivante (Zaiane, 1999).

$$\text{gain}(A) = I(S_P, S_N) - E(A)$$

$I(S_P, S_N)$ représente la quantité d'information nécessaire pour décider si un exemple arbitraire dans S appartient à P ou N.

$E(A)$ représente l'information nécessaire pour classifier les instances dans les sous-arbres. La quantité d'information $I(S_P, S_N)$ est calculée de la manière suivante.

Soit x le nombre d'instances dans la classe P et y le nombre d'instances dans la classe N.

$$I(S_P, S_N) = - \frac{x}{x+y} \log_2 \frac{x}{x+y} - \frac{y}{x+y} \log_2 \frac{y}{x+y}$$

Supposons que l'attribut A, qui représente la racine de l'arbre de décision, répartit l'ensemble des données S en des sous-ensembles $\{S_1, S_2, \dots, S_k\}$. Si le sous ensemble S_i contient x_i exemples de la classe P et y_i exemples de la classe N, alors l'information de classification $E(A)$ est calculée par la formule suivante (Zaiane, 1999).

$$E(A) = \sum_{i=1}^k \frac{x_i + y_i}{x + y} I(S_P, S_N)$$

Afin d'améliorer cet algorithme pour répondre au besoin de notre projet, nous avons ajouté la condition suivante qui permet d'assurer que tous les attributs soient présents dans un arbre de décision.

$$T_\alpha \geq A_{r_i} + 1$$

Où T_α est le niveau du nœud feuille en commençant par le nœud racine, A_{r_i} représente le nombre total des attributs non catégoriques définis pour une ressource i et le chiffre 1 représente l'attribut de catégorie du nœud racine.

5.4. Étude comparative

Dans cette section, nous utilisons un échantillon de données afin d'appliquer les différents algorithmes dont on a discuté dans les paragraphes précédents. Pour simuler ces algorithmes et générer les arbres de décision, nous utilisons l'outil SIPINA qui a été développé par Ricco Rakotomalala dans le laboratoire de recherche (Rakotomalala R. , 2009).

L'outil SIPINA est principalement un logiciel spécialisé d'exploration de données qui permet la simulation d'une vaste famille d'algorithmes de classification de données. Ce logiciel permet de générer un arbre de décision selon l'algorithme utilisé pour classifier l'ensemble de données. Dans notre travail, nous allons utiliser ce logiciel pour simuler les algorithmes ID3, CHAID et C4.5

Soit l'ensemble de données représenté par le tableau suivant.

| Attributs | Sujet | Objet | Action | permission |
|-----------|-------|----------|-----------|------------|
| Instances | Bob | fichier3 | supprimer | interdit |
| | Alice | fichier2 | ecrire | permis |
| | Alice | fichier3 | lire | interdit |
| | Alice | fichier4 | ecrire | interdit |
| | Alice | fichier2 | lire | permis |

Tableau 3.1 : Exemple d'un fichier de données

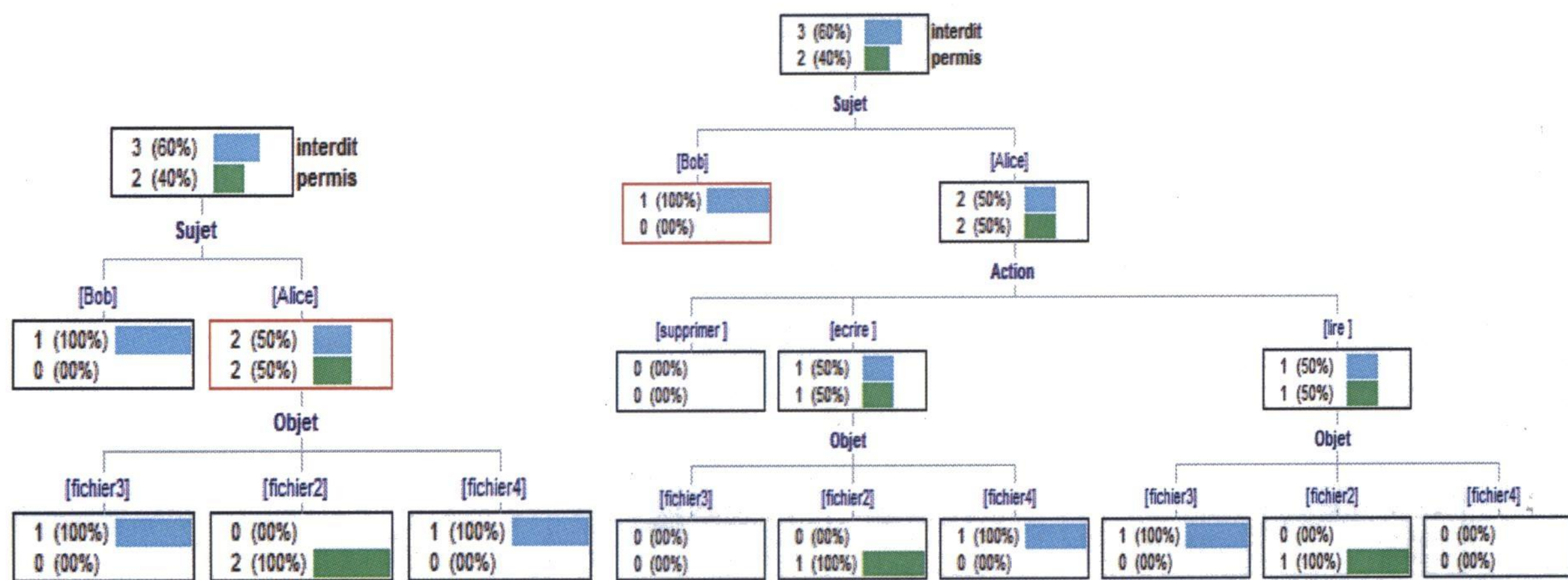
Le tableau 3.1 montre cinq règles de contrôle d'accès que nous désirons analyser en utilisant les différentes méthodes.

Dans cet exemple, nous avons deux classes «permis» et «interdit» et cinq politiques dont deux représentent une permission pour exercer des actions et trois représentent une interdiction pour exercer des actions.

Afin de générer l'arbre de décision pour cet ensemble de politiques, nous avons appliqué trois différents algorithmes d'exploration de données : ID3, CHAID et C4.5.

La figure 3.16 (a) montre l'arbre de décision généré par l'algorithme C4.5. Dans cette arborescence, nous constatons que la profondeur de l'arbre est égale à trois, et que les attributs, définis dans l'ensemble de politiques, ne sont pas tous présents dans l'arbre. Par exemple, l'attribut «Action» n'est pas inclus dans l'arborescence. Par contre, après avoir appliqué à nouveau l'algorithme C4.5 en ajoutant la condition $T_{\alpha} \geq A_{r_i} + 1$, nous

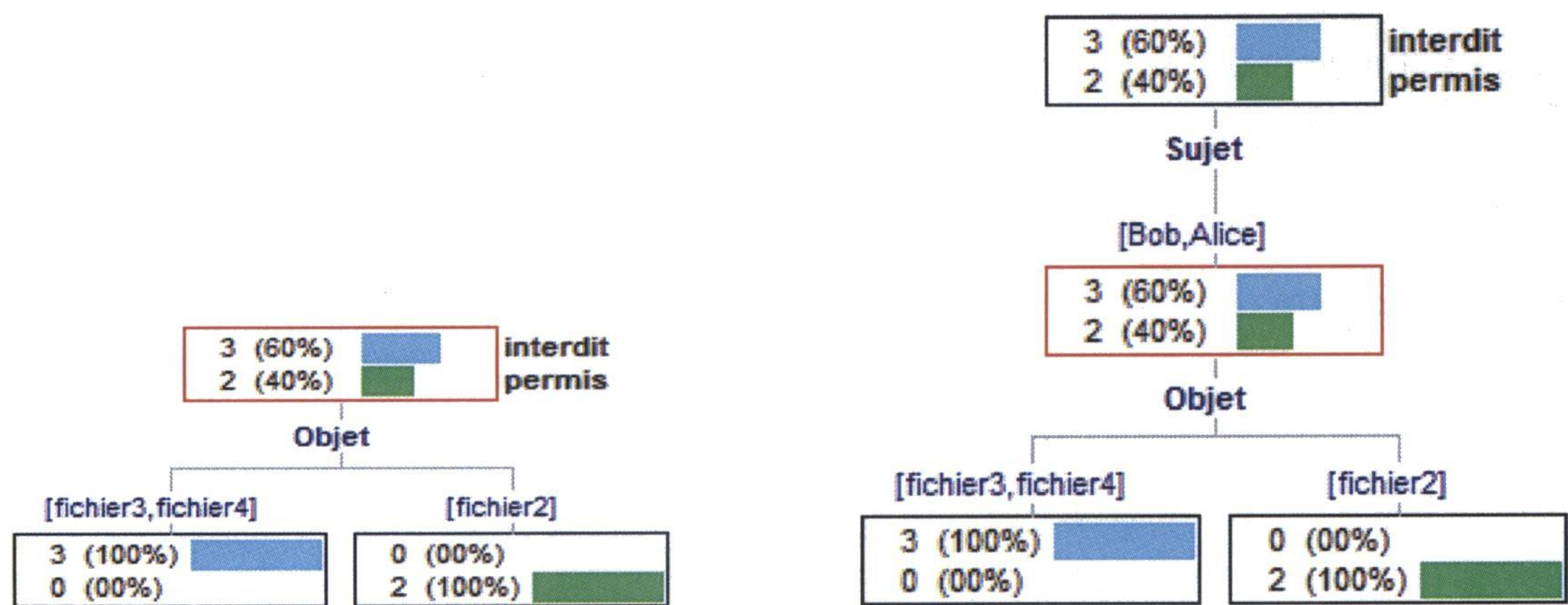
obtenons un arbre complet qui contient tous les attributs comme illustré dans la figure 3.16 (b). Toutefois, lorsque nous avons appliqué l'algorithme CHAID sur le même ensemble de données pour générer un arbre de décision, nous avons obtenu une arborescence plus compacte, qui ne contient pas tous les attributs comme montré dans la figure 3.17 (a). Semblablement à l'algorithme C4.5, cet algorithme ne fournit pas un arbre de décision complet. Par contre, lorsque nous avons ajouté la condition $T_\alpha \geq A_{r_i} + 1$, nous avons obtenu un arbre plus détaillé comme montré dans la figure 3.17 (b). Cependant, cet algorithme reste limité pour la détection des incomplétudes. Enfin, nous avons appliqué l'algorithme ID3 sur le même ensemble de données afin de générer un arbre de décision. Comme constatable dans la figure 3.18 (a), nous avons obtenu un arbre compact semblablement à C4.5. Après avoir ajouté la condition $T_\alpha \geq A_{r_i} + 1$, nous avons obtenu un arbre plus profond comme montré sur la figure 3.18 (b), par contre cet algorithme, d'une part n'est pas capable de satisfaire la condition de montrer tous les attributs des politiques, et d'autre part plus d'itérations peuvent être nécessaires pour pouvoir générer cet arbre de décision.



(a) C4.5

(b) C4.5 avec la condition $T_\alpha \geq A_{r_i} + 1$

Figure 3.16 : arbre de décision généré par C4.5



(a) CHAID

(b) CHAID avec la condition $T_\alpha \geq A_{r_i} + 1$

Figure 3.17 : arbre de décision généré par CHAID

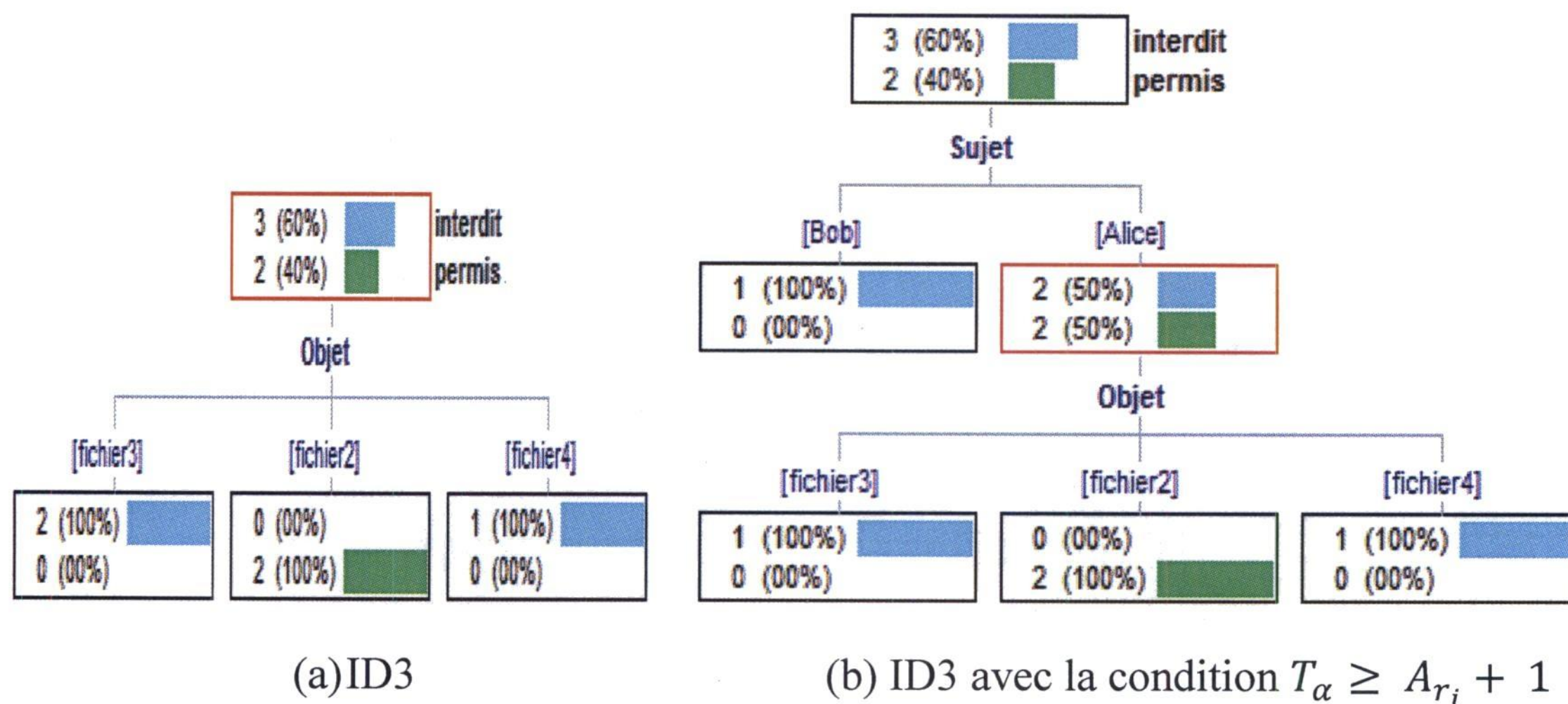


Figure 3.18 : arbre de décision généré par ID3

Dans cette section, nous avons étudié trois différentes méthodes de classification de données afin de pouvoir détecter les incohérences et les incomplétudes dans un ensemble de politiques de contrôle d'accès. Cette étude a montré que l'algorithme C4.5, en ajoutant certaines modifications, est une méthode efficace, pratique et simple pour atteindre l'objectif de notre recherche. À l'inverse, les autres méthodes ont montré des limitations et n'ont pas pu inclure tous les attributs dans l'arbre de décision.

CHAPITRE IV : FONDEMENT THEORIQUE DE NOTRE APPROCHE

1. Introduction

Un processus de vérification et de test est un élément essentiel pour la construction d'un système de contrôle d'accès cohérent et complet. Cependant, ce processus ne fait pas partie de l'outil EEM. L'organisation manuelle d'un grand nombre de politiques de contrôle d'accès est une tâche complexe pour un administrateur étant donné que tout changement (addition, suppression, modification) de politiques peut causer des effets négatifs sur la sécurité du système, tels que des conflits entre les politiques et la violation des exigences de sécurité.

Dans ce chapitre, nous présentons une solution, automatique et efficace, pour la validation d'un ensemble de politiques de contrôle d'accès. Notre idée est basée sur une modification de l'algorithme d'exploration de données C4.5.

2. Algorithme d'analyse de politiques C4.5 modifié

Dans notre projet, nous avons adopté une modification de la technique de classification C4.5 (Shaikh, Adi, & Logrippo, 2010) afin de détecter les incohérences et les incomplétudes dans les ensembles de politiques de sécurité.

Dans la suite de ce chapitre, nous utilisons la terminologie suivante :

Une politique de contrôle d'accès peut contenir une ou plusieurs *règles*. Nous utilisons le terme *règle* pour signifier une partie d'une politique.

Nous allons faire l'hypothèse que l'ensemble des règles sera normalisé de la façon suivante :

1. Chaque règle aura une valeur pour chacun des attributs possibles dans l'ensemble des règles en considération. Si aucune valeur n'a été donnée pour un de ces attributs dans la règle originale, on donnera à cet attribut une valeur défaut dans la normalisation. Cette valeur sera toujours considérée égale à toute autre valeur possible pour le même attribut.
2. Les attributs sont dans le même ordre dans toutes les règles.

Les règles de contrôle d'accès seront représentées comme des collections ordonnées d'attributs. Nous utilisons les termes de l'exploration de données qui sont : attributs de catégorie et attributs de non-catégorie.

Un attribut de *catégorie* (A) représente la classe de la règle et la racine de l'arbre de décision. Dans notre projet, nous considérons l'attribut «permission» comme attribut de catégorie qui peut prendre deux valeurs : {permis, interdit}.

Un attribut de *non-catégorie* (C) représente les attributs de décision (e.g sujet, action, localisation). Chaque attribut définit des caractéristiques importantes pour une règle spécifique.

Prenons comme exemple la règle suivante :

$R : \text{rôle(Professeur)} \wedge \text{ressource(F_notes)} \wedge \text{action(\acute{e}crire)} \rightarrow \text{permis}$

Dans cet exemple, on considère que les éléments «Professeur», «F_notes» et «écrire» sont des attributs de non-catégorie et «permis» est l'attribut de catégorie.

Soit R un ensemble de politiques ($R = \{R_1, R_2, \dots, R_n\}$), tel que $R \neq \phi$.

Soit $A = A_1 \wedge A_2 \wedge \dots \wedge A_n$ une conjonction d'attributs non catégorie et C un attribut de catégorie.

Une règle R_i est de la forme $\langle A, C \rangle$ où

$$R_i : A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow C$$

On définit la valeur attribuée à un attribut A_j dans une règle R_i par la formule $v(R_i, A_j)$. Nous écrivons aussi $R_i.C$ pour dénoter l'attribut de catégorie de la règle R_i .

2.1. Incohérence ou contradiction

Dans un ensemble de politiques de contrôle d'accès, une incohérence ou une contradiction définit une situation où deux règles incompatibles existent.

2.1.1. Incohérence directe

Une incohérence *directe* est une situation dans laquelle deux règles différentes donnent des résultats contradictoires dans le même ensemble de politiques. Par exemple, soit R_1 une règle qui permet au professeur Bob d'accéder à la ressource X et R_2 une règle qui interdit au professeur Bob d'accéder à la même ressource dans le même contexte. Dans cet exemple, le professeur Bob a une permission et une interdiction pour accéder à la même ressource.

La définition suivante signifie que deux règles sont incohérentes si leurs prémisses sont identiques, mais leurs conséquences sont différentes.

Définition 1

Soit $R_i, R_j \in R$, les règles R_i, R_j sont incohérentes si et seulement si :

- 1) $\forall A_k \in A, v(R_i, A_k) = v(R_j, A_k)$
- 2) $v(R_i, C) \neq v(R_j, C)$

Algorithme de détection d'incohérence (Shaikh, Adi, & Logrippo, 2010)

Dans cette section, nous proposons l'algorithme qui permet la détection d'incohérences dans un ensemble de politiques de contrôle d'accès.

b_i : une branche qui connecte un nœud racine à un nœud feuille dans l'arbre de décision.

$A(b_i)$: l'ensemble des attributs qui appartient à une branche b_i

$v(A(b_i))$: les valeurs des attributs qui appartiennent à la même branche

Input: Decision tree

Output: Context of inconsistency

1: Let $A(b_i)$ be the set of all attributes present in one branch.

2: Bool *consistent* = *true*;

3: **for** each branch b_i in Decision tree **do**

4: **if** more than 1 category attribute is assigned to terminal node $b_i.tnode$ **then**

5: $A(b_i)$ = fetch all attributes of branch(b_i);

6: **for** each actual rule R_a in the policy set **do**

7: **if** $v(A(R_a)) = v(A(b_i))$ **then**

8: Highlight: $R_a : A_1 \wedge \dots \wedge A_n \rightarrow C$;

9: **end if**

10: **end for**

11: *consistent* = *false*;

12: **end if**

13: **end for**

14: **if** *consistent* = *true* **then**

15: No inconsistency found;

16: **end if**

2.2. Incomplétude

Dans un système de contrôle d'accès, une incomplétude peut avoir lieu si une situation spécifique n'est pas prévue dans un ensemble de règles. Une telle situation pourrait permettre à un utilisateur non habilité d'accéder à des ressources critiques. Prenons comme exemple les deux règles suivantes :

- Une infirmière qui travaille durant le jour peut accéder aux dossiers des patients de 8H01 à 17H59.
- Une infirmière qui travaille durant la nuit peut accéder aux dossiers des patients de 18H01 à 7H59.

Dans cet exemple, il n'existe pas de règle qui contrôle l'accès aux dossiers des patients à 8h00 et 18h00 exactement, générant ainsi une incomplétude dans le système.

Définition 2

Soit $\mathcal{E}(A_j)$ l'ensemble des valeurs possibles pour un attribut A_j .

Soit $A_j \in A$, si $\bigcup_{i=1..m} v(R_i.A_j) \subset \mathcal{E}(A_j)$

Alors, l'ensemble des règles R est incomplet pour l'attribut A_j . À noter que nous utilisons le symbole \subset pour dénoter l'*inclusion propre*. Il y a donc des valeurs dans $\mathcal{E}(A_j)$ pour lesquelles il n'y a aucune décision.

Algorithme de détection d'incomplétude (Shaikh, Adi, & Logrippo, 2010)

Dans cette section, nous proposons un algorithme qui permet la détection d'incomplétudes dans un ensemble de politiques de contrôle d'accès.

b_i : une branche qui connecte un nœud racine à un nœud feuille dans l'arbre de décision.

$A(b_i)$: l'ensemble des attributs qui appartient à une branche b_i

Input: Decision tree

Output: Context of incompleteness

1: Let $A(b_i)$ be the set of all attributes present in one branch.

2: Bool *complete* = true;

3: for each branch b_i in Decision tree do

4: if no category attribute is assigned to terminal node

$b_i.tnode$ then

5: A = fetch all attributes of branch(b_i);


```
6: Policy set is incomplete w.r.to label(bi.tnode);  
7: Complete context: A(b);  
8: complete = false;  
9: end if  
10: end for  
11: if complete = true then  
12: No incompleteness found;  
13: end if
```

3. Processus de vérification

Dans cette section, nous présentons les démarches qui permettent la vérification des ensembles de politiques de contrôle d'accès de l'outil Siteminder. Nous allons aussi montrer le résultat de ces démarches dans un cas pratique.

3.1. Conversion de fichier de politiques

Le format *Comma-separated values*, connu sous le sigle CSV, est un format informatique ouvert représentant des données tabulaires sous forme de valeurs séparées par des virgules. Il est le format d'entrée lisible par notre outil.

Dans cette section, nous étudions un exemple de politiques qui nous permet d'illustrer la phase de conversion des politiques EEM en format XML vers le format CSV. Ce dernier sera utilisé par notre outil afin de détecter les anomalies.

Soit la politique suivante : «Un patient a le droit de lire son dossier durant la période *Working_Time2*».

La figure suivante montre la représentation de cette politique dans les structures de données internes de l'outil EEM.


```
</Policy>
<Policy folder="/" name="P2">
  <Description>patient can acces to his file</Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>False</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
  <Resource>Dossier</Resource>
  <Action>lire</Action>
  <Identity>ug:patient</Identity>
  <Calendar>Working_Time2</Calendar>
</Policy>
```

Figure 4.1 : Représentation de politique par EEM

Nous observons que l’outil EEM exprime les permissions disant qu’un explicit deny est faux comme \montré à la ligne 7.

Dans cette politique, on constate que les différents attributs (Identity, Resource et Action) sont représentés par des balises. Dans la figure 4.1, nous sommes seulement intéressés par les balises <Ressource>, <ExplicitDeny>, <Action>, <Identity> et <Calendar>.

Afin de garder une terminologie simple dans la suite de ce travail, nous utilisons les attributs sujets, objet, action et permission qui correspondent respectivement aux attributs Identity, Ressource, Action et ExplicitDeny.

Afin de convertir cette règle vers un format CSV, lisible par l’algorithme C4.5 modifié, nous avons développé un programme en utilisant le langage JAVA. Ce programme permet d’extraire les informations nécessaires à partir d’un fichier de politiques EEM et par la suite, de représenter ces informations en format CSV. Le tableau suivant montre un exemple de résultat de ce processus de conversion dans le cas de l’exemple précédent. Dans la représentation interne, les valeurs dans ce tableau sont séparées par des virgules.

| Policy | Identity | Resource | Action | Calendar | ExplicitDeny |
|--------|------------|----------|--------|---------------|--------------|
| P2 | ug:patient | Dossier | lire | Working_Time2 | False |

Tableau 4.1 : Exemple de conversion de politique

Le programme de transformation de politiques sera décrit en détail dans le chapitre suivant.

3.2. Stratégie d'analyse d'incohérence et d'incomplétude

La stratégie basée sur l'algorithme MC4.5 peut aider à détecter l'incomplétude en même temps que l'incohérence. Le processus d'analyse de l'ensemble de politiques de contrôle d'accès est composé de deux étapes.

Étape 1 : la création d'un arbre de décision complet. Afin d'atteindre ce but, on sélectionne l'attribut qui a une valeur *minimale* de l'information de gain, contrairement à l'algorithme standard C4.5 qui sélectionne la valeur maximale. Ce changement de critère nous permet de présenter tous les attributs dans l'arbre, ce qui est nécessaire pour notre application comme discuté dans la Section 2.

Étape 2 : regroupement des politiques de contrôle d'accès en fonction des ressources, considérant seulement les ensembles de ressources non vides. Par la suite, pour chaque ensemble de ressources, on définit les attributs non catégoriques qui permettent la prise de décision (e.g sujet, localisation). Après avoir regroupé les politiques et défini les attributs non catégoriques, on doit récupérer, pour chaque politique, les valeurs des attributs qui sont reliées à un ensemble de ressources. À cette étape, les attributs doivent être représentés en respectant un ordre spécifique, par exemple (sujet, action, localisation, temps, etc.), et, dans le cas où un attribut est absent dans une règle particulière, on lui attribue une valeur par défaut.

Afin d'avoir un arbre de décision complet, nous avons ajouté une condition qui nous permet d'assurer la présence de tous les attributs dans l'arbre de décision. Cette condition est décrite par la formule suivante :

$$T_{\alpha} \geq A_{r_i} + 1$$

Où T_{α} est le niveau du nœud feuille en commençant par le nœud racine. A_{r_i} représente le nombre total des attributs non catégoriques définis pour une ressource i et le chiffre 1 représente l'attribut de catégorie du nœud racine.

3.2.1. Analyse d'incohérence

Dans l'arbre de décision résultant, chaque politique est représentée par une branche qui relie le nœud racine avec un nœud feuille. On commence par vérifier le nœud feuille de chaque branche. Si le nœud contient plus qu'une valeur d'un attribut de catégorie, cela signifie qu'il existe des politiques qui sont incohérentes. Dans le cas où chaque feuille contient une seule valeur d'un attribut de catégorie, on peut conclure que l'ensemble de politiques de contrôle d'accès est cohérent.

Exemple

Prenons comme exemple le tableau suivant.

| Sujet | Action | Ressource | Permission |
|-------|--------|-----------|------------|
| Alice | écrire | F_note1 | permis |
| Bob | écrire | F_note1 | permis |
| Alice | écrire | F_note1 | interdit |
| Mark | lire | F_note2 | permis |

Tableau 4.2 : Exemple de règles en format CSV

Dans cet exemple, on a deux classes qui sont représentées par une permission positive «permis» ou une permission négative «interdit» et quatre règles dont trois positives et une négative.

En premier lieu, on calcule la quantité d'information des deux classes $I(S_{permis}, S_{interdit})$.

$$\triangleright I(S_{permis}, S_{interdit}) = I\left(\frac{3}{4}, \frac{1}{4}\right) = -\frac{3}{3+1} \log_2 \frac{3}{3+1} - \frac{1}{3+1} \log_2 \frac{1}{3+1}$$

$$I(S_{permis}, S_{interdit}) = -\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4}$$

$$I(S_{permis}, S_{interdit}) = 0.31 + 0.5 = 0.81$$

Ensuite, on calcule l'information de classification qui correspond à chaque attribut.

$$\triangleright E(\text{sujet}) = \frac{2}{4} * I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{1}{4} * I\left(\frac{1}{1}, \frac{0}{1}\right) + \frac{1}{4} * I\left(\frac{1}{1}, \frac{0}{1}\right)$$

$$E(\text{sujet}) = \frac{2}{4} * \left(-\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2}\right) + \frac{1}{4} * \left(-\frac{1}{1} \log_2 \frac{1}{1} - \frac{0}{1} \log_2 \frac{0}{1}\right) + \frac{1}{4} * \left(-\frac{1}{1} \log_2 \frac{1}{1} - \frac{0}{1} \log_2 \frac{0}{1}\right)$$

$$E(\text{sujet}) = 0.6 + 0 + 0 = 0.6$$

$$\triangleright E(\text{action}) = \frac{3}{4} * I\left(\frac{2}{3}, \frac{1}{3}\right) + \frac{1}{4} * I\left(\frac{1}{1}, \frac{0}{1}\right)$$

$$E(\text{action}) = \frac{3}{4} * \left(-\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3}\right) + \frac{1}{4} * \left(-\frac{1}{1} \log_2 \frac{1}{1} - \frac{0}{1} \log_2 \frac{0}{1}\right)$$

$$E(\text{action}) = \frac{3}{4} * (0.38 + 0.52) + 0$$

$$E(\text{action}) = 0.67$$

$$\begin{aligned} \text{➤ } E(\text{ressource}) &= \frac{2}{4} * I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{4} * I\left(\frac{2}{2}, \frac{0}{2}\right) \\ E(\text{ressource}) &= \frac{2}{4} * \left(-\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2}\right) + \frac{2}{4} * \left(-\frac{2}{2} \log_2 \frac{2}{2} - \frac{0}{2} \log_2 \frac{0}{2}\right) \\ E(\text{ressource}) &= 0.6 \end{aligned}$$

Après avoir obtenu les valeurs de quantité d'information ainsi que l'information de classification, on peut calculer l'information de gain qui permet la sélection du prochain attribut. À ce niveau, l'attribut adéquat est celui qui a une valeur de gain minimale.

$$\text{➤ } \textit{gain}(\textit{sujet}) = I(S_{\textit{Permis}}, S_{\textit{interdit}}) - E(\textit{sujet})$$

$$\textit{gain}(\textit{sujet}) = 0.81 - 0.6 = 0.21$$

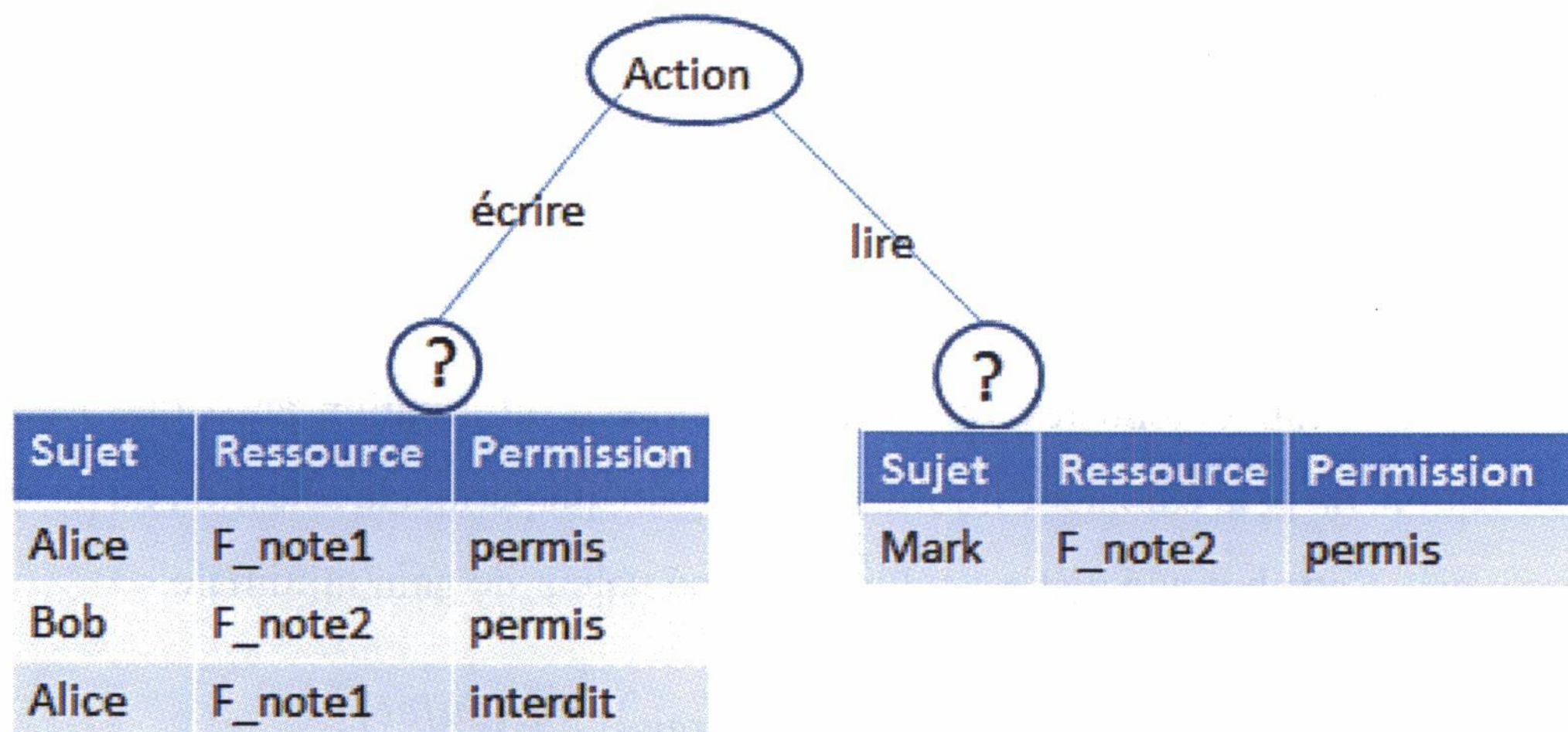
$$\text{➤ } \textit{gain}(\textit{action}) = I(S_{\textit{Permis}}, S_{\textit{interdit}}) - E(\textit{action})$$

$$\textit{gain}(\textit{action}) = 0.81 - 0.67 = 0.14$$

$$\text{➤ } \textit{gain}(\textit{ressource}) = I(S_{\textit{Permis}}, S_{\textit{interdit}}) - E(\textit{ressource})$$

$$\textit{gain}(\textit{ressource}) = 0.81 - 0.6 = 0.21$$

Après avoir calculé l'information de gain de chaque attribut, le premier niveau de l'arbre de décision est défini comme montré dans le graphe suivant :



De la même façon, on calcule les valeurs de quantité d'information et l'information de classification afin de sélectionner le prochain attribut pour les sous-tableaux.

$$\text{➤ } I(S_{\text{permis}}, S_{\text{interdit}}) = I\left(\frac{2}{3}, \frac{1}{3}\right) = -\frac{2}{2+1} \log_2 \frac{2}{2+1} - \frac{1}{2+1} \log_2 \frac{1}{2+1}$$

$$I(S_{\text{permis}}, S_{\text{interdit}}) = -\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3}$$

$$I(S_{\text{permis}}, S_{\text{interdit}}) = 0.38 + 0.52 = 0.9$$

$$\text{➤ } E(\text{sujet}) = \frac{2}{3} * I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{1}{3} * I\left(\frac{1}{1}, \frac{0}{1}\right)$$

$$E(\text{sujet}) = \frac{2}{3} * \left(-\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2}\right) + \frac{1}{3} * \left(-\frac{1}{1} \log_2 \frac{1}{1} - \frac{0}{1} \log_2 \frac{0}{1}\right)$$

$$E(\text{sujet}) = 0.66 + 0 = 0.66$$

$$\text{➤ } E(\text{ressource}) = \frac{2}{3} * I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{1}{3} * I\left(\frac{1}{1}, \frac{0}{1}\right)$$

$$E(\text{ressource}) = \frac{2}{3} * \left(-\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2}\right) + \frac{1}{3} * \left(-\frac{1}{1} \log_2 \frac{1}{1} - \frac{0}{1} \log_2 \frac{0}{1}\right)$$

$$E(\text{ressource}) = 0.66 + 0 = 0.66$$

Après avoir obtenu les valeurs de quantité d'information ainsi que l'information de classification, on peut calculer l'information de gain qui permet la sélection du prochain attribut. L'attribut choisi est celui qui a une valeur de gain minimale.

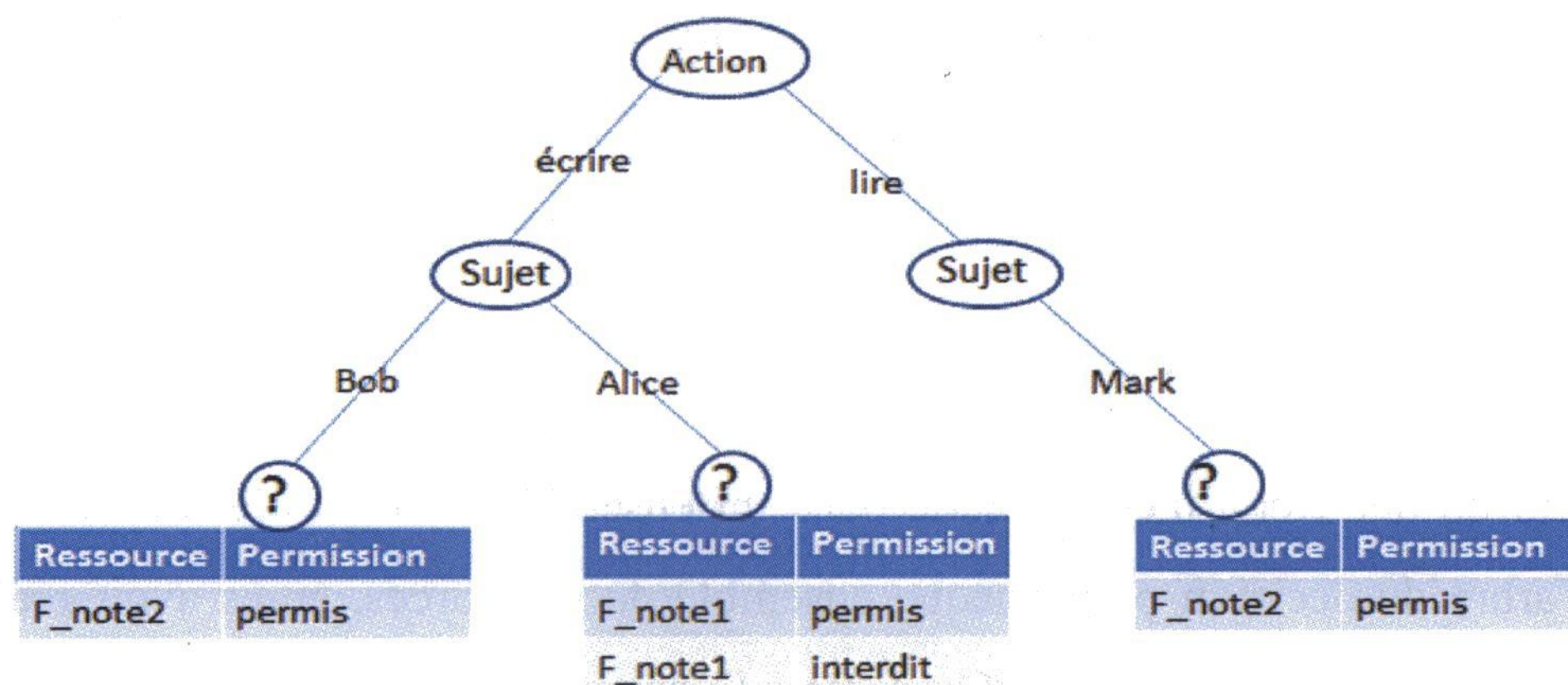
$$\text{gain}(\text{sujet}) = I(S_{\text{permis}}, S_{\text{interdit}}) - E(\text{sujet})$$

$$\text{gain}(\text{sujet}) = 0.9 - 0.66 = 0.24$$

$$\text{gain}(\text{ressource}) = I(S_{\text{permis}}, S_{\text{interdit}}) - E(\text{ressource})$$

$$\text{gain}(\text{ressource}) = 0.9 - 0.66 = 0.24$$

Après avoir calculé l'information de gain de chaque attribut, on peut définir le deuxième niveau de l'arbre de décision comme suit :



Récursivement, on répète le même calcul jusqu'à l'obtention de l'arbre de décision complet.

$$\text{I}(S_{\text{permis}}, S_{\text{interdit}}) = I\left(\frac{1}{2}, \frac{1}{2}\right) = -\frac{1}{1+1} \log_2 \frac{1}{1+1} - \frac{1}{1+1} \log_2 \frac{1}{1+1}$$

$$I(S_{\text{permis}}, S_{\text{interdit}}) = 1$$

$$E(\text{ressource}) = \frac{1}{2} * I\left(\frac{1}{1}, \frac{0}{1}\right) + \frac{1}{2} * I\left(\frac{0}{1}, \frac{1}{1}\right)$$

$$E(\text{ressource}) = \frac{1}{2} * \left(-\frac{1}{1} \log_2 \frac{1}{1} - \frac{0}{1} \log_2 \frac{0}{1} \right) + \frac{1}{2} * \left(-\frac{0}{1} \log_2 \frac{0}{1} - \frac{1}{1} \log_2 \frac{1}{1} \right)$$

$$E(\text{ressource}) = 0$$

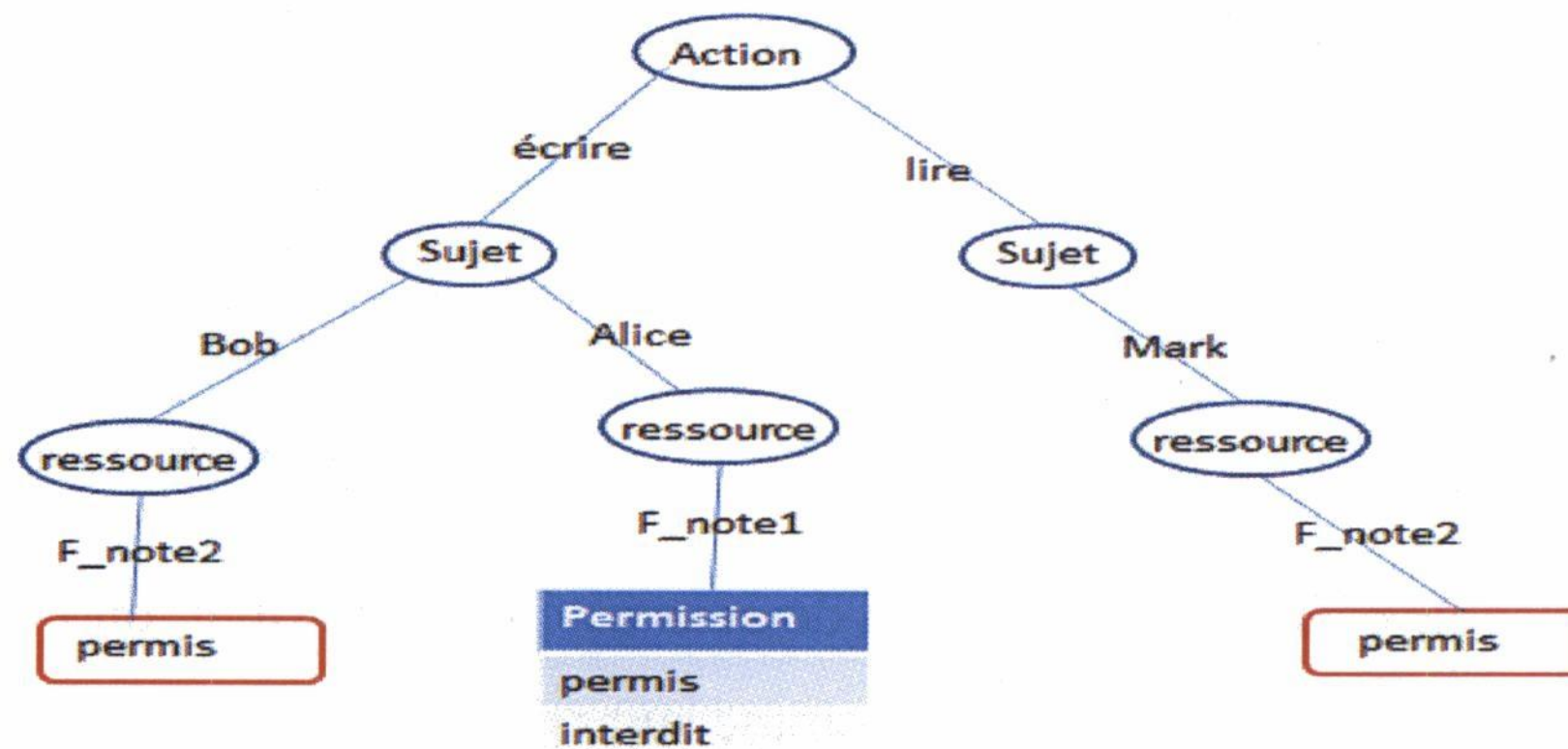


Figure 4.2 : Arbre de décision final

Dans la figure 4.3, on constate clairement l'existence d'une incohérence dans les règles qui donnent à l'utilisateur Alice une permission positive et négative en même temps, et dans le même contexte, pour écrire dans le fichier «F_note1».

3.2.2. Analyse d'incomplétude

Afin de détecter l'incomplétude dans un ensemble de contrôle d'accès, on doit générer l'arbre de décision pour chaque ensemble de ressources, après avoir réalisé les étapes mentionnées antérieurement. On passe à l'analyse de l'arbre de décision où chaque politique est représentée par une branche qui relie un nœud racine à un nœud feuille. On commence par la vérification des nœuds feuilles pour chaque branche. Dans le cas où un nœud feuille ne contient aucune valeur de l'attribut catégorie, on peut conclure qu'il n'existe pas de règle explicite qui contrôle l'accès à cette ressource dans un contexte spécifique décrit par cette branche. Si tous les nœuds feuilles contiennent des valeurs de l'attribut catégorie, cela signifie que l'ensemble de politiques de contrôle d'accès est complet.

Exemple

Prenons comme exemple l'ensemble des règles représentées par le tableau suivant.

| Sujet | Jour | Ressource | Permission |
|-------|----------|-----------|------------|
| Alice | lundi | F_note1 | permis |
| Bob | lundi | F_note1 | permis |
| Alice | mercredi | F_note1 | interdit |
| Alice | mardi | F_note1 | interdit |
| Alice | vendredi | F_note1 | permis |
| Bob | mercredi | F_note1 | permis |
| Alice | jeudi | F_note1 | permis |
| Bob | vendredi | F_note1 | interdit |
| Bob | mardi | F_note1 | interdit |

Tableau 4.3 : Ensemble de règles en format CSV

Dans cet exemple, on a deux classes qui sont représentées par une permission positive «permis» ou une permission négative «interdit» et neuf règles dont cinq sont positives et quatre sont négatives.

Après avoir effectué le calcul comme expliqué dans la section précédente, on obtient un arbre de décision contenant l'ensemble de règles, dont chaque branche représente une règle (voir figure 4.4).

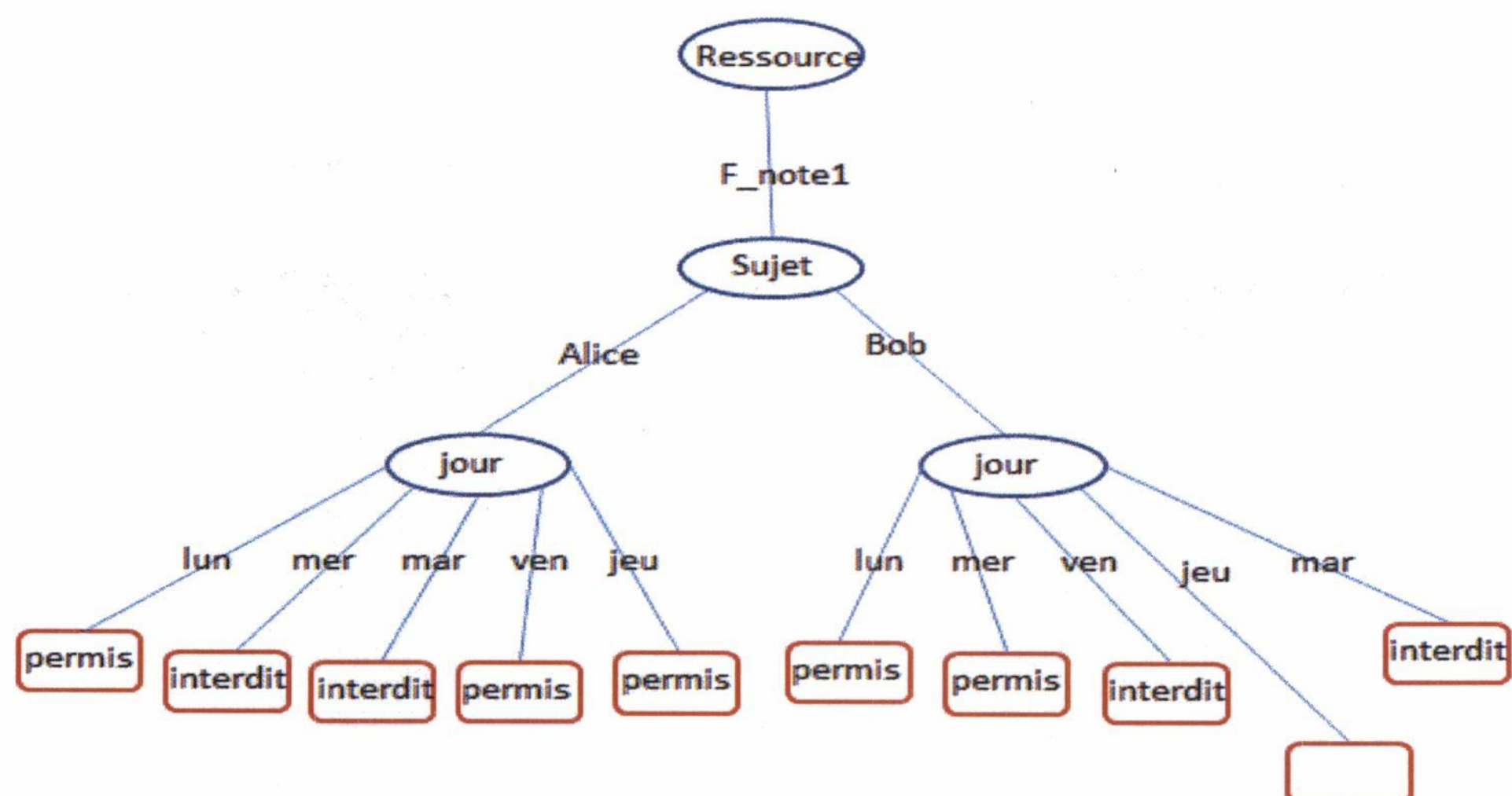


Figure 4.3 : Arbre de décision incomplet

À la figure 4.4, on constate que l'accès à la ressource «F_note1» par l'utilisateur «Bob» durant le jour «jeudi» n'est pas contrôlé par une règle explicite, ce qui veut dire qu'il y a une incomplétude dans l'ensemble des règles de contrôle d'accès.

La capacité de détecter l'incomplétude en même temps que l'incohérence est un atout de notre technique. À notre connaissance, aucune autre technique proposée dans la littérature ne possède cette caractéristique.

4. Conclusion

Dans ce chapitre, nous avons présenté la base théorique de notre méthode pour détecter les incohérences et les incomplétudes dans un ensemble de politiques de contrôles d'accès. De plus, nous avons démontré que cette analyse automatisée est faisable en principe et peut détecter des incohérences. Nous avons aussi analysé des propriétés intéressantes telles que les incohérences directes et les incomplétudes. Dans le chapitre suivant, nous décrivons un outil qui implémente ces principes.

CHAPITRE V : OUTIL DE DETECTION D'ANOMALIES

1. Introduction

Comme mentionné précédemment, l'organisation d'un grand nombre de politiques de contrôle d'accès est une tâche complexe pour les administrateurs. Tout ajout, suppression ou modification peut entraîner des effets secondaires inconnus, que nous appelons collectivement *anomalies*, tels que des incohérences et des incomplétudes.

Nous essayons de résoudre ce problème en proposant un outil qui analyse l'ensemble des politiques et qui fournit aux administrateurs les informations dont ils ont besoin pour détecter les anomalies et en comprendre les causes. Cet outil aide les administrateurs à vérifier les ensembles de politiques générés par des outils de contrôle d'accès tels que IAM, Siteminder, etc. Il permet d'extraire les politiques définies par ces logiciels et de les analyser à travers une méthode d'exploration de données. Dans ce chapitre, nous présentons l'architecture et l'interface de notre outil. Nous montrons aussi comment notre outil peut détecter les anomalies et produire des diagnostics graphiques. Notre approche sera illustrée en introduisant quelques exemples de politiques.

2. Architecture

Comme mentionné antérieurement, l'outil fonctionne en connexion avec des logiciels de contrôle d'accès. Ces derniers permettent de spécifier les politiques de contrôle d'accès, et de les ranger dans des fichiers XML, sans prendre en considération la possibilité d'existence d'anomalies. Notre outil extrait les ensembles de politiques à partir de ces fichiers XML et les transforme dans un format CSV (Comma-Separated Values) lisible par notre programme d'analyse.

L'outil fournit des diagnostics clairs et compréhensibles aux administrateurs, dans le cas où des anomalies existent dans l'ensemble de politiques en considération. L'administrateur peut ensuite corriger les anomalies et exécuter de nouveau le processus de vérification afin d'assurer que les corrections ont réellement éliminé toutes les anomalies sans en introduire des nouvelles.

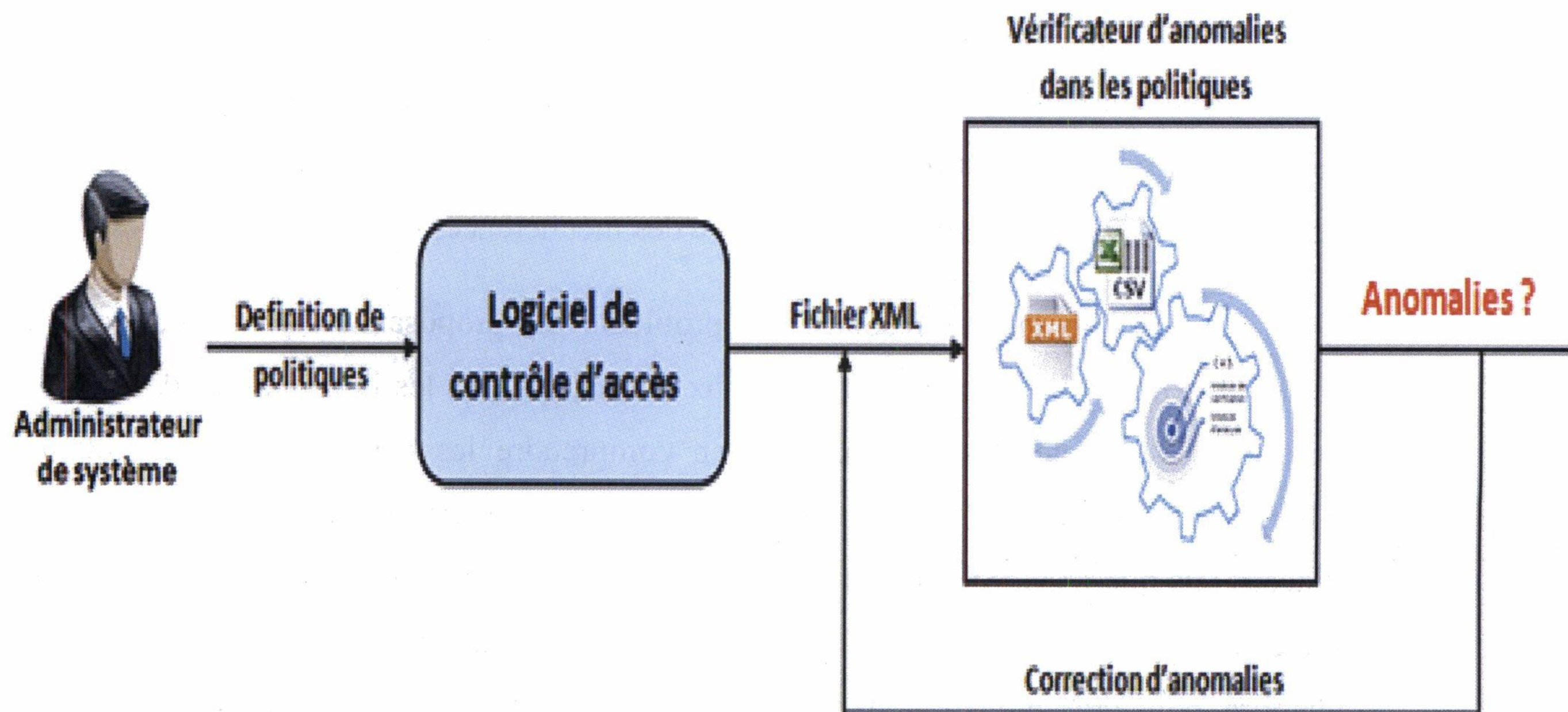


Figure 5. 1 : Architecture principale de l'outil de vérification

3. Description de l'outil

Nous avons conçu l'interface utilisateur de cet outil afin qu'elle soit aussi conviviale que possible, permettant ainsi aux administrateurs de gérer facilement les anomalies. La figure 5.2 montre l'interface principale de notre outil.



Figure 5. 2: Interface principale de l'outil de vérification

L'outil permet de vérifier l'ensemble des politiques décrites en XML et d'identifier les politiques qui causent des anomalies en les indiquant dans une fenêtre séparée. Il donne aussi la possibilité de corriger les conflits à travers une interface.

L'interface de l'outil contient six boutons qui permettent de réaliser les tâches suivantes :

1. Define XML file structure : définir la structure interne du fichier XML de politiques
2. Select XML file : sélectionner le fichier à vérifier sous forme XML
3. Check inconsistencies : choisir le type de vérification qu'un administrateur peut effectuer
4. Export : enregistrer les résultats de la vérification dans un fichier Excel
5. Graph : fournir un diagnostic graphique des politiques de contrôle d'accès
6. XML correction : corriger les conflits entre les politiques dans le fichier XML afin de procéder à une nouvelle vérification

Exemple

Afin de tester l'outil développé, nous avons pris comme exemple un ensemble de politiques de contrôle d'accès défini par le logiciel IAM. Ces politiques sont représentées dans un fichier XML qui sera le fichier d'entrée pour notre application, voir figure 5.3.

Dans la suite de ce chapitre, nous définissons les périodes de travail que nous allons utiliser dans notre exemple :

Working_Time1 réfère à l'intervalle de temps 8h à 16h

Working_Time2 réfère à l'intervalle de temps 16h à 24h

Working_Time3 réfère à l'intervalle de temps 24h à 8h


```
177 </Policy>
178 <Policy folder="/" name="P2">
179   <Description>patient can access to his file</Description>
180   <ResourceClassName>Dossier</ResourceClassName>
181   <PolicyType>policy</PolicyType>
182   <Disabled>False</Disabled>
183   <ExplicitDeny>False</ExplicitDeny>
184   <PreDeployment>False</PreDeployment>
185   <RegexCompare>False</RegexCompare>
186   <Resource>Dossier</Resource>
187   <Action>lire</Action>
188   <Identity>ug:patient</Identity>
189   <Calendar>Working_Time2</Calendar>
190 </Policy>
191 <Policy folder="/" name="P3">
192   <Description>le dossier des personnes dÃ©cÃ©dÃ© est couvert par le secret mÃ©dical</Description>
193   <ResourceClassName>Dossier</ResourceClassName>
194   <PolicyType>policy</PolicyType>
195   <Disabled>False</Disabled>
196   <ExplicitDeny>False</ExplicitDeny>
197   <PreDeployment>False</PreDeployment>
198   <RegexCompare>False</RegexCompare>
199   <Resource>Dossier</Resource>
200   <Action>couvert</Action>
201   <Identity>ug:medecin</Identity>
202   <Calendar>Working_Time3</Calendar>
203 </Policy>
204 <Policy folder="/" name="P4">
```

Figure 5.3 : Fichier XML de politiques

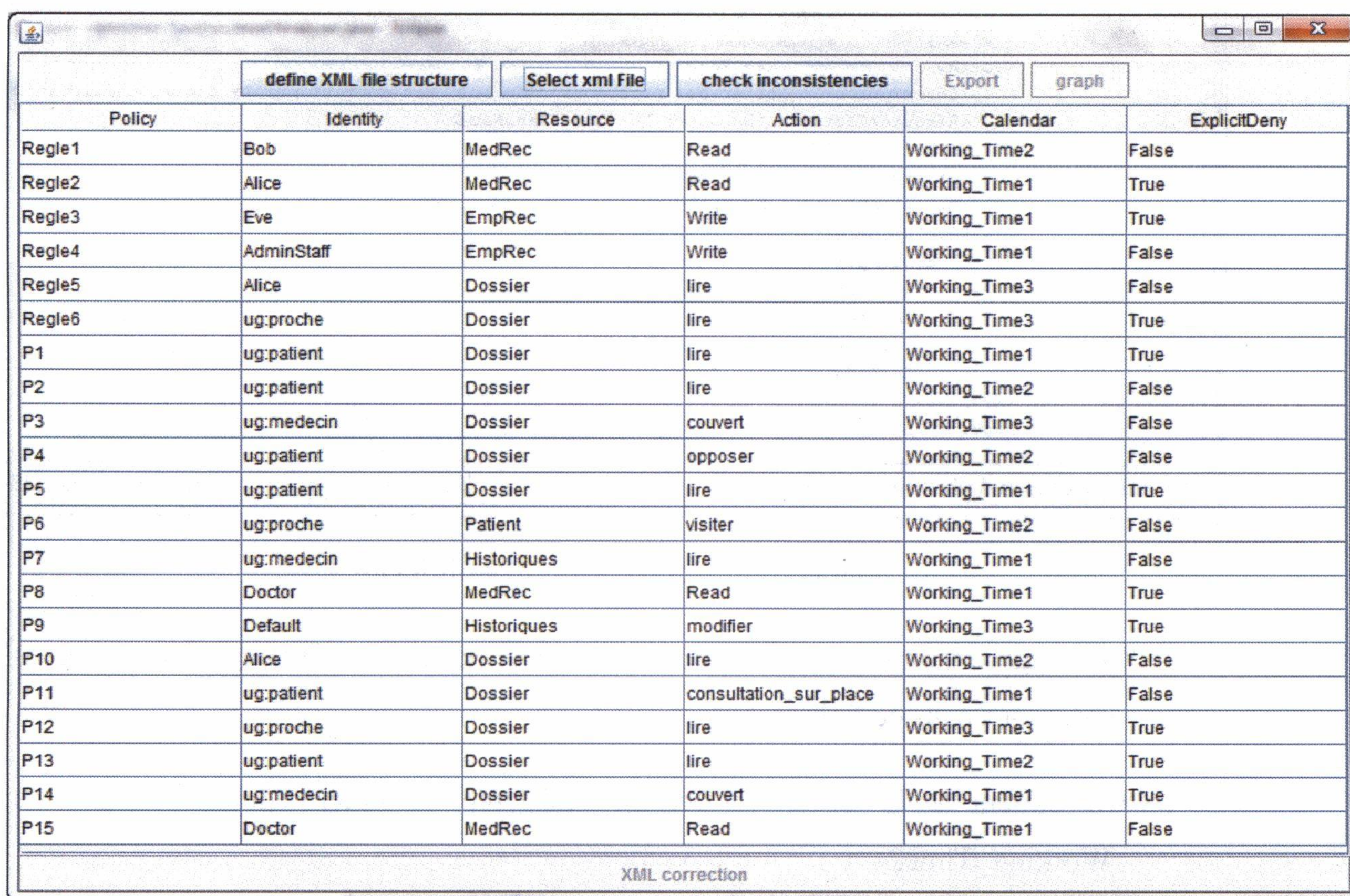
Cette figure montre le fichier utilisé pour tester notre outil. En langue naturelle, les politiques définies dans ce fichier sont :

- 1- Un patient a le droit de lire la ressource «Dossier» durant la période «Working_Time2»
- 2- Un médecin a le droit de lire la ressource «Dossier» durant la période «Working_Time3»

Dans cet exemple, nous avons montré seulement une partie du fichier XML. Le fichier entier est donné dans l'Appendice A.

4. Vérification d'incohérences

Dans l'interface principale de notre outil, les ensembles de politiques sont représentés dans un format CSV (Comma-Separated Values) qui sera présenté, dans ce qui suit, sous forme de tableaux Excel. Ces politiques seront analysées pour vérifier l'existence d'anomalies. La figure 5.4 présente le résultat de la conversion du fichier XML de politiques, exposé dans la section précédente, sous forme d'un fichier CSV. La première ligne de ce fichier CSV représente les noms des différents attributs qui existent dans le fichier XML et les autres lignes représentent les instances de chaque attribut.



| Policy | Identity | Resource | Action | Calendar | ExplicitDeny |
|--------|------------|-------------|------------------------|---------------|--------------|
| Regle1 | Bob | MedRec | Read | Working_Time2 | False |
| Regle2 | Alice | MedRec | Read | Working_Time1 | True |
| Regle3 | Eve | EmpRec | Write | Working_Time1 | True |
| Regle4 | AdminStaff | EmpRec | Write | Working_Time1 | False |
| Regle5 | Alice | Dossier | lire | Working_Time3 | False |
| Regle6 | ug:proche | Dossier | lire | Working_Time3 | True |
| P1 | ug:patient | Dossier | lire | Working_Time1 | True |
| P2 | ug:patient | Dossier | lire | Working_Time2 | False |
| P3 | ug:medecin | Dossier | couvert | Working_Time3 | False |
| P4 | ug:patient | Dossier | opposer | Working_Time2 | False |
| P5 | ug:patient | Dossier | lire | Working_Time1 | True |
| P6 | ug:proche | Patient | visiter | Working_Time2 | False |
| P7 | ug:medecin | Historiques | lire | Working_Time1 | False |
| P8 | Doctor | MedRec | Read | Working_Time1 | True |
| P9 | Default | Historiques | modifier | Working_Time3 | True |
| P10 | Alice | Dossier | lire | Working_Time2 | False |
| P11 | ug:patient | Dossier | consultation_sur_place | Working_Time1 | False |
| P12 | ug:proche | Dossier | lire | Working_Time3 | True |
| P13 | ug:patient | Dossier | lire | Working_Time2 | True |
| P14 | ug:medecin | Dossier | couvert | Working_Time1 | True |
| P15 | Doctor | MedRec | Read | Working_Time1 | False |

Figure 5. 4 : Ensemble de politiques en format CSV

Cette figure montre le résultat de la conversion du fichier XML montrée dans l'appendice et partiellement observé dans la figure 5.3. Les seules politiques qui correspondent à la figure 5.3 sont les lignes relatives aux politiques P2 et P3.

Après avoir analysé l'ensemble des politiques, l'outil montre celles qui sont en conflit dans une nouvelle fenêtre. La partie supérieure affiche un tableau qui contient les politiques en question et la partie inférieure explique le résultat de l'analyse (voir figure 5.5).

| Policy | Identity | Resource | Action | Calendar | ExplicitDeny |
|--------|------------|----------|--------|---------------|--------------|
| P2 | ug:patient | Dossier | lire | Working_Time2 | False |
| P13 | ug:patient | Dossier | lire | Working_Time2 | True |
| P8 | Doctor | MedRec | Read | Working_Time1 | True |
| P15 | Doctor | MedRec | Read | Working_Time1 | False |

[P2] est en conflit avec [P13]

[P8] est en conflit avec [P15]

Figure 5. 5 : Résultat d'analyse d'incohérences

Le diagramme suivant, aussi produit par notre outil, montre l'ensemble des politiques de la figure 5.4 sous forme d'un graphe. Afin d'en faciliter aux administrateurs la gestion de ce graphe, nous avons éliminé les instances redondantes pour en avoir une vision globale. Les conflits sont indiqués sur le graphe global par une couleur différente (voir figure 5.6).

Noter que la première ligne nomme les attributs. L'incohérence est identifiée par le fait que, en correspondance de l'attribut *Explicit Deny*, on a simultanément VRAI et FAUX.

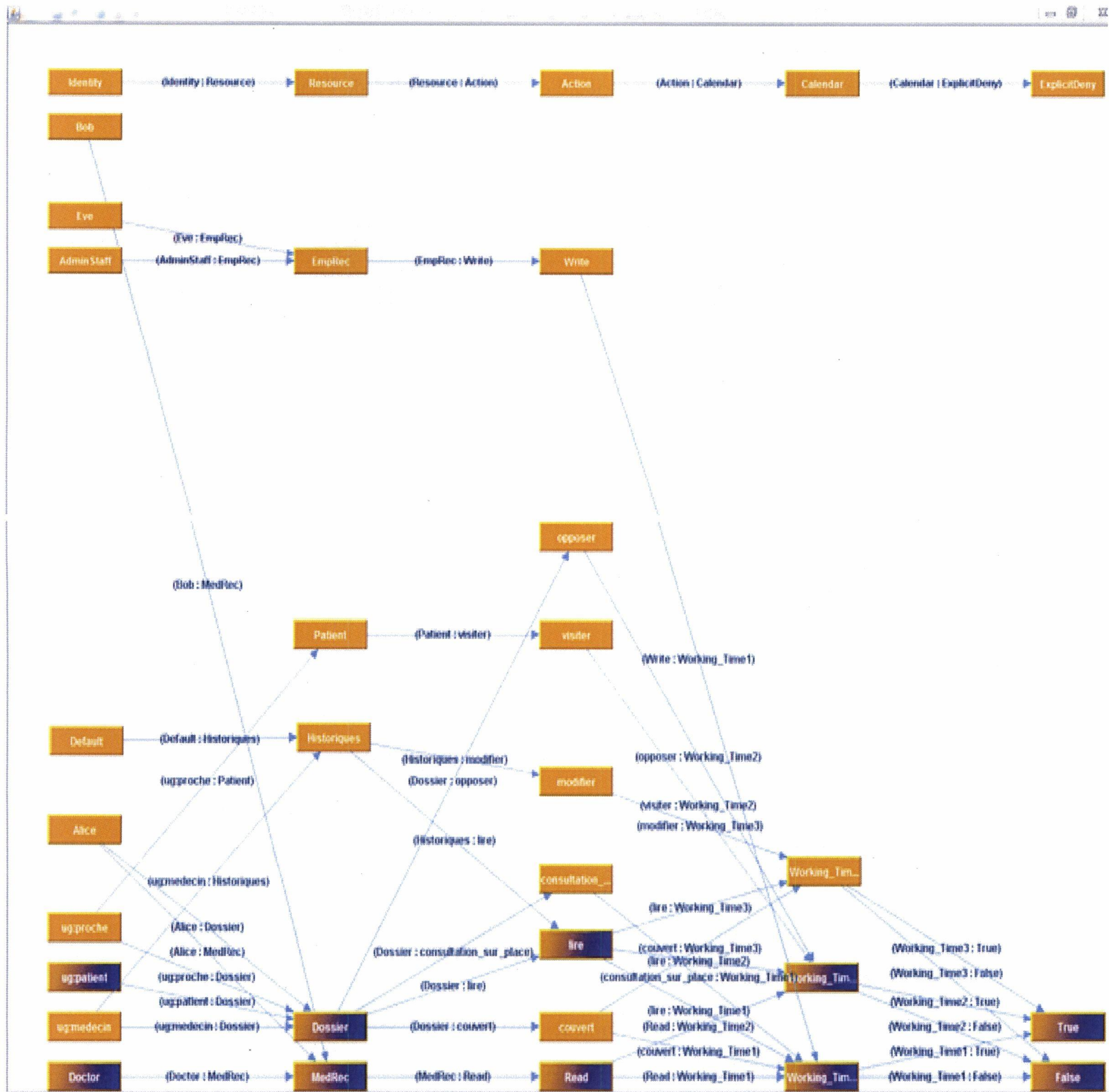


Figure 5.6 : Graphe global de politiques de contrôle d'accès

Afin de faciliter le diagnostic et l'analyse du graphe global par un administrateur, dans le cas où le graphe généré par notre outil est de grande taille, nous avons aussi fourni à l'administrateur la possibilité d'afficher un graphe local pour vérifier des instances spécifiques en cliquant deux fois de suite sur le nœud en question (voir figure 5.7).

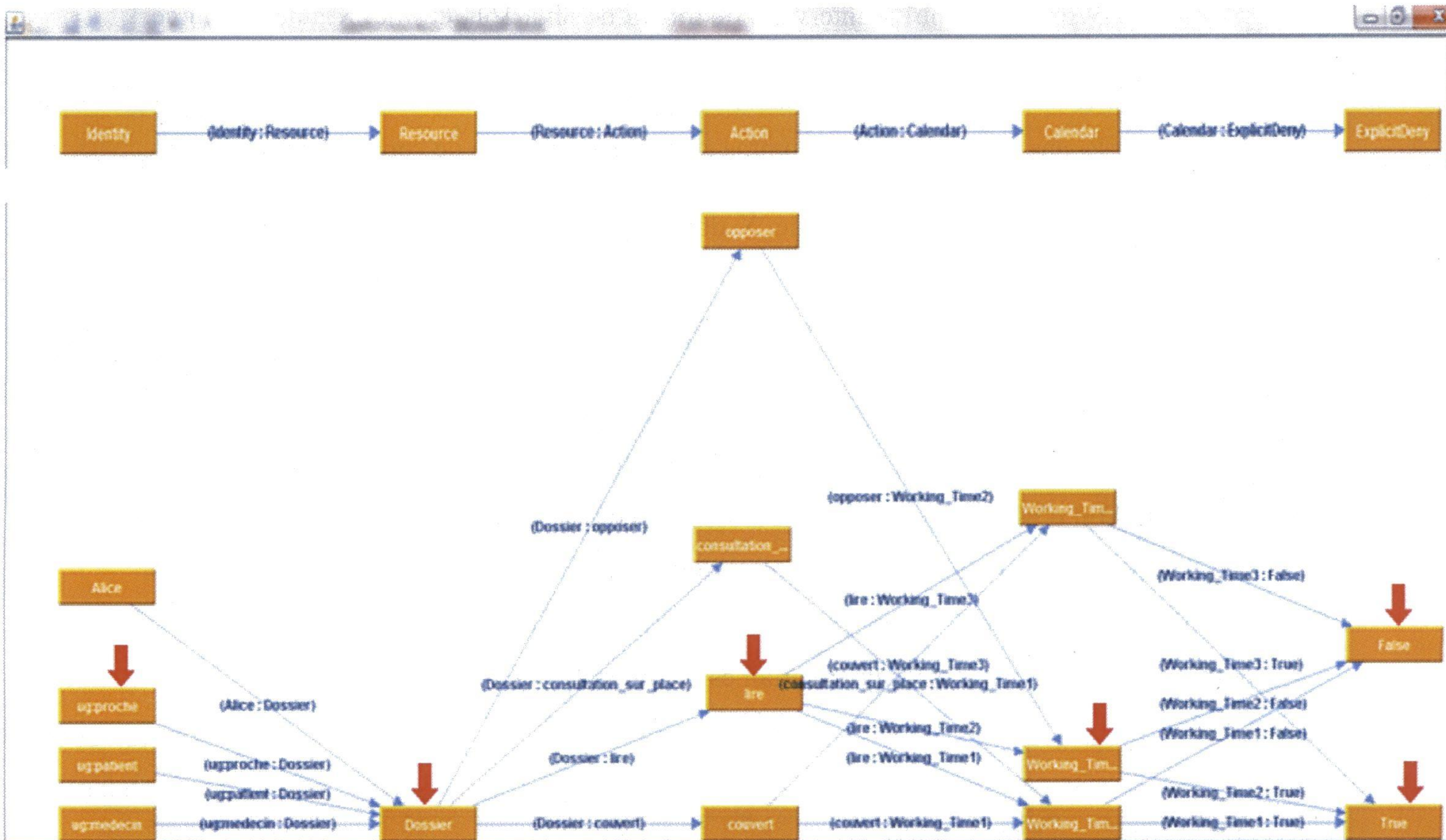


Figure 5.7 : Graphe local de politiques de contrôle d'accès

Le graphe de la figure 5.7 est obtenu en cliquant deux fois de suite sur le nœud « Dossier ». Ce graphe montre toutes les règles liées à cette ressource. L'incohérence montrée dans cette figure est la première montrée à la figure 5.5, entre les règles P2 et P3.

5. Vérification d'incomplétudes

Une incomplétude est une situation où il n'existe pas de politique qui contrôle l'accès à une ressource pendant une période spécifique.

Soit `Working_Time1`, l'intervalle de travail de 8h à 16h, et `File_X`, une ressource qui contient des informations secrètes.

Soit P_x une politique décrite comme suit dans le système de contrôle d'accès.

P_x : l'accès à la ressource `File_X` durant la période `Working_Time1` est permis lundi, mardi et jeudi. L'accès à la ressource `File_X` durant la période `Working_Time1` est interdit le vendredi.

Dans cette situation, le système de contrôle d'accès ne sait pas comment réagir pour la permission d'accès à la ressource `File_X` durant la période `Working_Time1` pour le mercredi. Afin d'aider l'administrateur à détecter ce type d'anomalie, nous avons produit une composante, complémentaire à notre outil, qui permet d'afficher les incomplétudes existantes dans l'ensemble des politiques, en indiquant les ressources manquantes des politiques de contrôle d'accès pour certaines périodes de travail (voir figure 5.8).

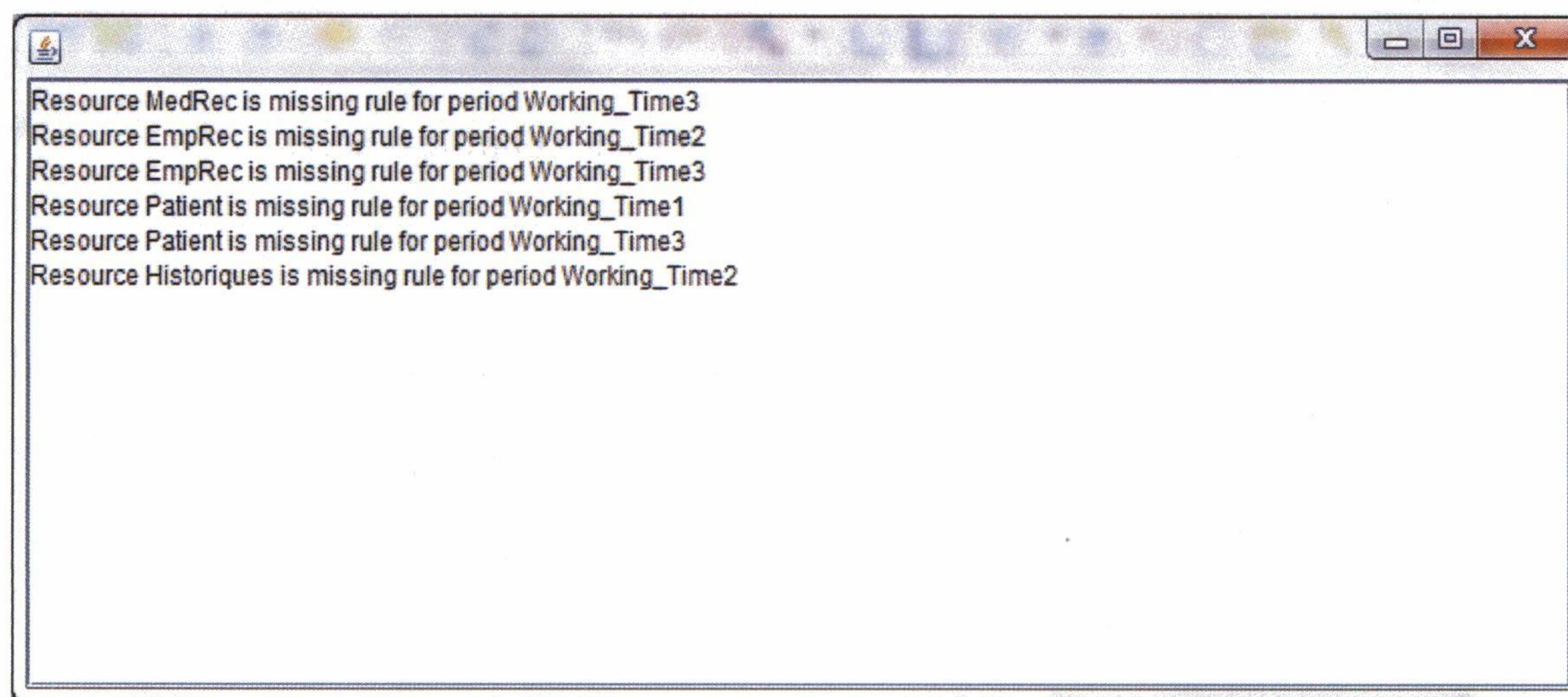


Figure 5. 8 : Résultats de vérification d'incomplétudes

Sur cette figure, nous montrons les incomplétudes trouvées par notre outil dans l'ensemble de politiques définies dans la figure 5.4. Par exemple, la ressource « MedRec » n'a pas de règle qui contrôle l'accès durant « Working_Time3 ». La même situation s'applique pour la ressource « EmpRec » qui n'a pas de règle pour contrôler l'accès durant les périodes « Working_Time2 » et « Working_Time3 ».

CHAPITRE VI : CONCLUSION

1. Travail accompli et contribution visée

Dans ce travail, nous nous sommes consacrés au problème de la détection des incohérences et d'incomplétudes dans les ensembles de politiques de contrôle d'accès. Les incohérences et les incomplétudes que nous avons collectivement appelées anomalies dans les politiques de contrôle d'accès sont des erreurs qui doivent être détectées, car elles peuvent conduire à des failles de sécurité. Suivant les idées de travaux précédents dans

notre groupe de recherche (Shaikh, Adi, & Logrippo, 2010), nous avons adopté une méthode de détection d'anomalies basée sur la construction d'arbres de décision, et ce, en utilisant une variation de l'algorithme de classification C4.5 (voir chapitre III, section 5.3), algorithme bien connu dans le domaine de l'exploration des données. Nous avons appliqué notre idée à l'outil IAM de la compagnie CA Technologies, qui a été présenté dans le chapitre II. La contribution finale de ce travail est un logiciel qui est capable de lire les fichiers de politiques IAM et de produire des arbres de décision qui permettent l'identification des anomalies dans les politiques. Ce logiciel pourra être utilisé par les administrateurs de systèmes de sécurité basés sur IAM, pour les aider à produire des ensembles de politiques cohérents et complets. À noter que notre solution n'est pas, en principe, limitée à IAM. En effectuant des ajustements sur cet outil, ce dernier peut être utilisé avec d'autres systèmes de contrôle d'accès (voir chapitre V pour plus de détails).

Comme expliqué dans le chapitre IV, notre processus s'effectue en plusieurs étapes. D'abord, nous extrayons les valeurs des attributs (sujet, ressource, action, etc.) à partir des politiques de contrôle d'accès définies par l'outil IAM. Pour atteindre ce but, nous avons développé un module basé sur le langage JAVA permettant la conversion des fichiers de politiques vers le format CSV (*Comma-separated values*). Par la suite, nous avons procédé à l'analyse de fichiers de politiques afin de détecter l'existence des incohérences et des incomplétudes en adoptant deux différentes stratégies. La première stratégie, qui consiste à la détection des incohérences, est constituée de deux étapes : d'abord, on construit un arbre de décision complet à partir des données du fichier de politiques, puis on procède à l'analyse de l'arbre dont chaque politique est représentée par une branche qui relie le nœud racine à un nœud feuille. La seconde stratégie permet la détection des incomplétudes. Après avoir construit l'arbre de décision, où chaque politique est représentée par une branche qui relie un nœud racine à un nœud feuille, on passe à la vérification des nœuds feuille pour chaque branche. Dans le cas où un nœud feuille ne contient aucune valeur de l'attribut catégorie, on peut conclure qu'il n'existe pas de règle explicite qui contrôle l'accès à cette ressource.

Nous avons aussi présenté, dans le chapitre III, un état de l'art portant sur les modèles de contrôle d'accès. Par la suite, nous avons discuté de quelques méthodes qui permettent la vérification de la cohérence dans des ensembles de politiques. Cette étude de l'état de l'art nous a permis de mieux comprendre notre méthode par rapport aux modèles et méthodes existants.

Plusieurs chercheurs ont tenté de résoudre le problème de conflit dans des ensembles de politiques de contrôle d'accès en utilisant différentes méthodes, notamment des méthodes basées sur la logique formelle. Notre méthode se démarque par le fait qu'elle effectue tant la détection d'incohérences que la détection d'incomplétudes avec le même algorithme. Elle est simple, efficace et pratique. Nous utilisons une modification de l'algorithme C4.5 et, d'après nos connaissances, notre groupe est le premier à utiliser un algorithme de classification de données pour détecter les incohérences dans un ensemble de contrôle d'accès (Shaikh, Adi, & Logrippo, 2010).

En conclusion, les contributions principales visées par ce mémoire sont :

- Conception et implémentation de la méthode de conversion de fichier de politiques IAM vers une notation CSV (*Comma-separated values*) pour la préparation des données d'entrée à la phase suivante
- Implémentation de l'algorithme C4.5 modifié pour la détection d'incohérences et d'incomplétudes dans des ensembles de politiques de contrôle d'accès
- Développement d'un outil avec une interface graphique pour l'utilisation de l'algorithme mentionné

2. Travaux futurs

Dans ce travail, les politiques de délégation et d'obligation n'ont pas été considérées. Ainsi, nous avons étudié deux types de conflits : conflit direct et incomplétude. Il serait intéressant d'étendre notre méthode pour pouvoir détecter d'autres types de conflits.

Exemple d'incohérence causée par d'une délégation

Soit les trois politiques suivantes :

P1 :

- Un administrateur de compte a le droit de lire, écrire et supprimer les informations d'un compte bancaire
- Un préposé au service a le droit de lire et écrire les informations d'un compte bancaire

P2 :

- Un administrateur a le droit de supprimer un compte bancaire
- Un préposé au service n'a pas le droit de supprimer un compte bancaire

P3 :

- Un administrateur peut déléguer ses droits à un préposé au service.

Supposons que dans une telle situation, l'administrateur de compte délègue ses droits à un préposé au service. Dans ce cas, on constate qu'un préposé au service va hériter, depuis l'administrateur, du droit de suppression. Toutefois, la politique P2 mentionne qu'un préposé au service n'a pas le droit de supprimer un compte bancaire. On parle alors d'une incohérence indirecte.

Bibliographie

Agrawal, D., Giles, J., Lee, K.-W., & Lobo, J. (2005). Policy Ratification. *In: Proceedings of the IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pp. 223–232.

Baader, F., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F. (2003). THE DESCRIPTION LOGIC HANDBOOK: Theory, implementation, and applications. *Cambridge University Press*.

Boettner, P., Gupta, M., Wu, Y., & Allen, A. A. (2009). Towards Policy Driven Self-Configuration of User-Centric Communication. *ACMSE, Clemson, SC, USA. ACM*, pp. 19-21.

CA Technologies. (2009). CA SiteMinder Prepares You for What's Ahead. *white paper*.

Clarke, E., Fujita, M., McGeer, P., & Yang, J. (1997). Multi-Terminal Binary Decision Diagrams: An Efficient Data Structure for Matrix Representation. *Form. Methods Syst. Des.* 10(2–3), pp. 149–169.

- Ferraiolo, D. F., Kuhn, D. R., & Chandramouli, R. (1992). Role-Based Access Control. *15th National Computer Security Conference, Baltimore MD*, pp. 554 - 563.
- Ferraiolo, D., Cugini, J., & Kuhn, R. (1995). Role-Based Access Control (RBAC) : Features and Motivations. *Annual Computer Security Applications Proceeding*, pp. 241-248.
- Ferraiolo, D., Sandhu, R., & Gavrila, S. (2001). Proposed NIST Standard for role-Based Access Control. *ACM Transactions on Information and System Security*, pp. 224-247, Vol. 4, No. 3.
- Fisler, K., Krishnamurthi, S., Meyerovich, L. A., & Tschantz, M. C. (2005). Verification and Change-Impact Analysis of Access-Control Policies. *In: Proceedings of International Conference on Software Engineering (ICSE)*, pp. 196–205.
- Harrison, A. M., Ruzzo, L. W., & Ullman, D. (1976). Protection in Operating Systems. *Communication of the ACM*, pp. 461-471.
- Huang, F., Huang, Z., & Liu, L. (2009). A DL-based Method for Access Control Policy Conflict Detecting. *ACM Internetware, Beijing, China*, pp. 54-69, 978-1-60558-872-8/09/10.
- Keleta, Y., Eloff, J., & Venter, H. (2005). Proposing a Secure XACML architecture ensuring privacy and trust. *Research in Progress Paper, University of Pretoria*, p. http://icsa.cs.up.ac.za/issa/2005/Proceedings/Research/093_Article.pdf.
- Kotsiantis, S. (2007). Supervised machine learning: A review of classification techniques. *Informatica*, pp. 249–268, vol. 31.
- Landwehr, C. E. (1981). Formal Models for Computer Security. *ACM Computing Surveys (CSUR)* , pp. 247 - 278.
- Layouni, S. Y. (2010). Détection des conflits dans les politiques de controle d'accés. *Mémoire de Maitrise en Informatique. Université du Quebec en Outaouais*.
- Lin, D., Rao, P., Bertino, E., Li, N., & Lobo, J. (2010). EXAM: a comprehensive environment for the analysis of access control policies. *International Journal of Information Security*, pp. 253-273, Volume 9.
- Mazid, M. M., Shawkat, A., & Tickle, K. S. (2010). Improved C4.5 Algorithm for Rule Based Classification. *school of Computing Science Central Queensland University AUSTRALIA*.
- Moses, T. (2005). Extensible access control markup language (xacml) version 2.0. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

- Netegrity-Inc. (1997). Siteminder Concept Guide.
<http://fr.scribd.com/doc/52046770/siteminder-concepts-guide>.
- O’Gorman, L. (2003). Comparing Passwords, Tokens, and Biometrics for User Authentication. *Proceedings of the IEEE*, pp. 2019-2040, Vol. 91.
- Park, J., & Sandhu, R. (1999). Smart certificates: Extending X.509 for secure attribute services on the Web. In *Proceedings of 22nd National Conference on Information Systems Security*, p. 12.
- Park, S. J., & Sandhu, R. (2001). Role-Based Access Control on the Web. *ACM Transactions on Information and System Security*, pp. 37-71, Vol. 4.
- Quinlan, J. (1986). Induction of Decision Trees, *Machine Learning*. pp. 81-106.
- Quinlan, J. (1993). C4.5: Programs for Machine Learning. *Morgan Kaufmann Publishers, San Mateo, CA*.
- Rakotomalala, R. (2005). Arbres de Décision. *Laboratoire ERIC, Revue MODULAD*.
- Rakotomalala, R. (2009). *Sipina data mining software*. Récupéré sur <http://eric.univ-lyon2.fr/~ricco/sipina.html>
- Sandhu, R. (1993, Nov). Lattice Based Access Controls models. *IEEE Computer Society*, pp. 9-19.
- Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-Based Access Control Models. *IEEE*, pp. 38 - 47.
- Sandhu, R. S., Coyne, J. E., Feinstein, L. H., & Youman, E. C. (1996). Role-Based Access Control Models. *IEEE*, pp. 38-47, 0018-9162.
- Shaikh, R. A., Adi, K., & Logrippo, L. (2010). Detecting Incompleteness in Access Control Policies Using Data Classification Schemes. in *proc of the 5th International Conference on Digital Information Management (ICDIM), Thunder Bay, Canada, IEEE Press*, pp. 417-422.
- Shaikh, R. A., Adi, K., & Logrippo, L. (2010). Inconsistency Detection Method for Access Control Policies. *Information Assurance and Security (IAS), Sixth International Conference*.
- Swafford, S. (2009). SiteMinder Versus Custom Code. <http://radicaldevelopment.net/siteminder-custom-code/>.
- Zaiane, O. (1999). Chapter7: Data classification.
<http://webdocs.cs.ualberta.ca/~zaiane/courses/cmput690/slides/Chapter7/index.htm>.

Appendice A

Dans cet appendice, nous montrons au complet le fichier de politiques de contrôle d'accès que nous avons utilisé pour tester notre outil de détection d'anomalies. Ce fichier a été généré à partir de l'outil IAM de la compagnie CA Technologies et il contient vingt politiques. Nous avons expliqué ce fichier plus en détail dans le chapitre V.


```

<Policy folder="/" name="Regle1">
  <ResourceClassName>MedRec</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>False</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
  <Resource>MedRec</Resource>
  <Action>Read</Action>
  <Identity>Bob</Identity>
  <Calendar>Working_Time2</Calendar>
</Policy>
<Policy folder="/" name="Regle2">
  <ResourceClassName>MedRec</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>True</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
  <Resource>MedRec</Resource>
  <Action>Read</Action>
  <Identity>Alice</Identity>
  <Calendar>Working_Time1</Calendar>
</Policy>
<Policy folder="/" name="Regle3">
  <ResourceClassName>EmpRec</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>True</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
  <Resource>EmpRec</Resource>
  <Action>Write</Action>
  <Identity>Eve</Identity>
  <Calendar>Working_Time1</Calendar>
</Policy>
<Policy folder="/" name="Regle4">
  <ResourceClassName>EmpRec</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>False</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
  <Resource>EmpRec</Resource>
  <Action>Write</Action>
  <Identity>AdminStaff</Identity>
  <Calendar>Working_Time1</Calendar>
</Policy>

```



```

<Policy folder="/" name="Regle5">
  <Description>en cas d urgence le representant légal peut lire
  le dossier du patient</Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>False</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
    <Resource>Dossier</Resource>
    <Action>lire</Action>
    <Identity>Alice</Identity>
  <Calendar>Working_Time3</Calendar>
</Policy>
<Policy folder="/" name="Regle6">
  <Description>les proches ne peuvent accéder au dossier du
  </Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>True</Disabled>
  <ExplicitDeny>True</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
    <Resource>Dossier</Resource>
    <Action>lire</Action>
    <Identity>ug:proche</Identity>
  <Calendar>Working_Time3</Calendar>
</Policy>
<Policy folder="/" name="P1">
  <Description>un patient mineur ne peut pas accéder a son Dossier
  ssi par l intermédiaire de sont representant légal</Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>True</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
    <Resource>Dossier</Resource>
    <Action>lire</Action>
    <Identity>ug:patient</Identity>
  <Calendar>Working_Time1</Calendar>
</Policy>

```



```

<Policy folder="/" name="P2">
  <Description>patient can acces to his file</Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>False</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
    <Resource>Dossier</Resource>
    <Action>lire</Action>
    <Identity>ug:patient</Identity>
    <Calendar>Working_Time2</Calendar>
</Policy>
<Policy folder="/" name="P3">
  <Description>le dossier des personnes d'acÃ©dÃ© est couvert par
  le secret mÃ©dical</Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>False</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
    <Resource>Dossier</Resource>
    <Action>couvert</Action>
    <Identity>ug:medecin</Identity>
    <Calendar>Working_Time3</Calendar>
</Policy>
<Policy folder="/" name="P4">
  <Description>le mineur peut s'opposer a la consultation de son
  Dossier des titulaire paental</Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>False</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
    <Resource>Dossier</Resource>
    <Action>opposer</Action>
    <Identity>ug:patient</Identity>
    <Calendar>Working_Time2</Calendar>
</Policy>

```

I


```

<Policy folder="/" name="P5">
  <Description>un majeur sous tutelle ne peut accéder a son
  Dossier</Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>True</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
    <Resource>Dossier</Resource>
    <Action>lire</Action>
    <Identity>ug:patient</Identity>
    <Calendar>Working_Time1</Calendar>
  </Policy>
<Policy folder="/" name="P6">
  <Description>les proches peuvent visiter le patient durant
  les heures de visites</Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>False</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
    <Resource>Patient</Resource>
    <Action>visiter</Action>
    <Identity>ug:proche</Identity>
    <Calendar>Working_Time2</Calendar>
  </Policy>
<Policy folder="/" name="P7">
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>False</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
    <Resource>Historiques</Resource>
    <Action>lire</Action>
    <Identity>ug:medecin</Identity>
    <Calendar>Working_Time1</Calendar>
  </Policy>

```



```

<Policy folder="/" name="P8">
  <Description> </Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>True</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
  <Resource>MedRec</Resource>
  <Action>Read</Action>
  <Identity>Doctor</Identity>
  <Calendar>Working_Time1</Calendar>
</Policy>
<Policy folder="/" name="P9">
  <Description>personne ne peut modifier l' historique du patient
  </Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>True</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
  <Resource>Historiques</Resource>
  <Action>modifier</Action>
  <Calendar>Working_Time3</Calendar>
</Policy>
<Policy folder="/" name="P10">
  <Description>en cas d urgence le representant l gal peut lire
  le dossier du patient</Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>False</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
  <Resource>Dossier</Resource>
  <Action>lire</Action>
  <Identity>Alice</Identity>
  <Calendar>Working_Time2</Calendar>
</Policy>

```



```

<Policy folder="/" name="P11">
  <Description>le Dossier peut etre consulter sur place ou envoyer
  une copie a domicile.</Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>False</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
    <Resource>Dossier</Resource>
    <Action>consultation_sur_place</Action>
    <Identity>ug:patient</Identity>
    <Calendar>Working_Time1</Calendar>
</Policy>
<Policy folder="/" name="P12">
  <Description>les proches ne peuvent accéder au dossier du patient
  </Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>True</Disabled>
  <ExplicitDeny>True</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
    <Resource>Dossier</Resource>
    <Action>lire</Action>
    <Identity>ug:proche</Identity>
    <Calendar>Working_Time3</Calendar>
</Policy>
<Policy folder="/" name="P13">
  <Description>patient can acces to his file</Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>True</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
    <Resource>Dossier</Resource>
    <Action>lire</Action>
    <Identity>ug:patient</Identity>
    <Calendar>Working_Time2</Calendar>
</Policy>

```



```

<Policy folder="/" name="P14">
  <Description>le dossier des personnes dÃ©cÃ©dÃ© est
  couvert par le secret mÃ©dical</Description>
  <ResourceClassName>Dossier</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>True</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
  <Resource>Dossier</Resource>
  <Action>couvert</Action>
  <Identity>ug:medecin</Identity>
  <Calendar>Working_Time1</Calendar>
</Policy>
<Policy folder="/" name="P15">
  <ResourceClassName>MedRec</ResourceClassName>
  <PolicyType>policy</PolicyType>
  <Disabled>False</Disabled>
  <ExplicitDeny>False</ExplicitDeny>
  <PreDeployment>False</PreDeployment>
  <RegexCompare>False</RegexCompare>
  <Resource>MedRec</Resource>
  <Action>Read</Action>
  <Identity>Doctor</Identity>
  <Calendar>Working_Time1</Calendar>
</Policy>

```