**République Algérienne Démocratique et Populaire**

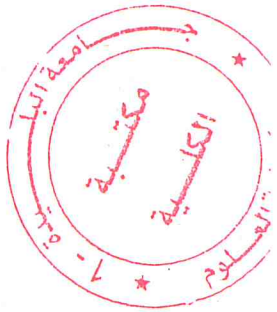
**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Université Saad Dahleb Blida 1**

**Faculté des Sciences et de la Technologie**

**Département Informatique**

**Projet de fin d'étude pour l'obtention du diplôme Master**



**Option**

## Sécurité des Systèmes d'Information

**Thème**

# Conception et réalisation d'un Framework WBEM dédié à l'assistance dans la gestion de politiques de sécurité SELinux

***Sujet proposé et encadré par :***

Dr. BOUABID Mohamed Amine

Maître de Recherche Classe B, Division R&D Réseaux, CERIST.

***Promotrice :***

Mme. REZOUG Nachida

***Réalisé par :***

M. BELGROUN Islam

Mlle. BOUNAB Bouchra

Président de jury : Mme Bounab

**Soutenue le 09/2017**

# *Remerciement*

*Tout d'abord, nous remercions Allah pour la patience, la volonté, la force et la santé qu'il nous a donné afin de réaliser ce travail.*

*Nos chaleureux remerciements s'adressent à notre encadreur Monsieur BOUABID Amine ; nous voudrons lui exprimer toute nos reconnaissances pour son encadrement, ses conseils, l'aide et le temps qu'il a bien voulu nous consacrer et sans qui ce mémoire n'aurait jamais vu le jour.*

*Nos vifs remerciements s'adressent également à notre promotrice Madame REZOUG Nachida, pour sa précieuse contribution, ses conseils et son aide.*

*Enfin, nous tenons à remercier toute personne qui a, à des degrés divers, contribué sur le plan intellectuel, technique, moral ou encore matériel à l'achèvement de ce travail.*

# Dédicace

*Nous dédions ce mémoire*

*À nos parents pour leur soutien moral et leurs sacrifices le long de notre formation.*

*À nos chères frères et sœurs que Allah les protège.*

*À nos chers amis, cousins et tout autre membre de nos familles.*

*Nous espérons qu'ils trouveront dans ce travail toutes nos reconnaissances et tout notre amour.*

## ملخص

ان عدد التهديدات الامنية التي تتعرض لها الحواسيب المرتبطة بالشبكات تتزايد باستمرار . لذلك تم التركيزبكثرة على الوقاية من مخاطر استغلال امن الشبكات.

فالعديد من المشاكل الأمنية في أنظمة يونكس هي ناتجة من طريقة الادارة الامنية المنتهجة ، اي انها تعتمد على تفويض جميع القرارات الأمنية لممتلكي الأشياء . الا انه يوجد العديد من الأطر الأمنية التي تهدف إلى علاج هذا في نظام لينكس عن طريق تقييد الوصول إلى أشياء النواة كالملفات.
في اطار مشروعنا هذا اننا مهتمون بادارة السياسات الامنية لالية مراقبة الوصول الالزامي التابعة لعائلة لينكس المسماة سولينكس.

باستخدام المعيارسيم/واب ام لقد اقترحنا نظام يهدف الى التعبير عن القواعد والممتلكات التي تحدد سياسات مراقبة الوصول سولينكس بلغة سيم ، فالامر يتعلق بمزود قادر على ترجمة قواعد سياسة سيلينوكس الى كائنات سيم والعكس صحيح.

فالعمل الذي تطرقنا اليه يعتبر بمثابة مساهمة في مجال إدارة السياسة الأمنية ماك المتعلقة بانظمة يونكس ، المنهج المتبع سيم/واب ام يهدف الى ادخال فكرة مجردة عن الية مراقبة الدخول سولينكس المنفذة في مختلف انظمة لينكس كما يهدف الى تسيير موزع ، متجانس و مستقل.



**الكلمات الرئيسية**

سيم/واب ام ، سولينكس ، ادارة ، السياسات الامنية ، مراقبة الوصول الالزامي.

**Résumé**

Le nombre de menaces pour les ordinateurs rattachés aux réseaux augmente continuellement. L'accent a été mis sur la prévention des exploits de sécurité sur le réseau, alors que les exploits locaux ont été surtout négligés. De nombreux problèmes de sécurité dans les systèmes Unix découlent de la gestion de la sécurité ; En déléguant toutes les décisions de sécurité aux propriétaires d'objets. Il existe un certain nombre de Framework de sécurité qui visent à remédier à cela dans Linux en restreignant l'accès aux objets du noyau, tels que les fichiers.

Dans le cadre de notre projet, nous nous intéressons à la gestion des politiques de sécurité d'un mécanisme de contrôle d'accès obligatoire déployé dans les systèmes d'exploitation de la famille Linux nommé SELinux.

En utilisant le standard CIM/WBEM, nous proposons un système permettant d'exprimer les règles et propriétés qui définissent les politiques du contrôle d'accès SELinux en langage CIM, il s'agit d'un Provider qui permet de traduire les règles de politiques SELinux en objets CIM et vice-versa, que nous allons intégrer dans une solution qui implémente tous les composants de l'architecture WBEM.

Notre travail représente une contribution dans le domaine de la gestion des politiques de sécurité MAC des systèmes Unix, l'approche adoptée (CIM/WBEM) permet d'introduire une abstraction sur le mécanisme de contrôle d'accès SELinux implémenté dans différents systèmes Linux et d'assurer une gestion distribuée homogène et indépendante des plateformes.

**Mots clés**

CIM/WBEM, SELinux, gestion, politiques de sécurité, contrôle d'accès mandataire

**Abstract**

The number of threats to computers attached to networks is continually increasing. Emphasis was placed on preventing security exploits on the network, while local exploits were mostly neglected. Many security issues in Unix systems arise from security management; By delegating all security decisions to object owners. There are a number of security frameworks that aim to remedy this in Linux by restricting access to kernel objects, such as files.

As part of our project, we are interested in standardizing the security policy management of a mandatory access control mechanism deployed in the operating systems of the Linux family named SELinux.

Using the CIM / WBEM standard, we propose a system allowing to express the rules and properties which define the policies of the access control SELinux in CIM, it is a provider that allows to translate the rules of policies SELinux into CIM objects and vice versa, which we will integrate into a solution that implements all the components of the WBEM architecture.

Our work represents a contribution in the management of MAC security policies of Unix systems, the adopted approach (CIM / WBEM) allows to introduce an abstraction on the SELinux access control mechanism implemented in different Linux systems and ensure consistent, platform-independent distributed management.

**Keywords**

CIM/WBEM, SELinux, management, security policy, mandatory access control.

# TABLE DES MATIÈRES

# TABLE DES FIGURES

# TABLE DES TABLEAUX

**Liste des abréviations**

| | |
|---|---|
| **ACSE** | Association Control Service Entity |
| **ASEs** | Application Service Entities |
| **AVC** | Access Vector Cac |
| **CIM** | Common Information Model |
| **CIMOM** | Cim Object Manager |
| **CIMOR** | Cim Object Repository |
| **CMISE** | Common Management Information Service Entity |
| **CIM SPL** | CIM SIMPLIFIED POLICY LANGUAGE |
| **CMIS** | Common Management Information Service |
| **CMIP** | Common Management Information Protocol |
| **CMPI** | Common ManagementProgramming Interface |
| **DAC** | Discretionary Access Control |
| **DTD** | Document Type Definition |
| **DTD** | Definition Type Document |
| **DMTF** | Distributed Management Task Force |
| **FCAPS** | Fault Configuration Accounting Performance Security |
| **GDMO** | Guideline for the Definition of Managed Objects |
| **HTTP** | HyperText Transfer Protocol |
| **IETF** | Internet Engineering Task Force |
| **IDL** | Interface Definition Language |
| **IP** | Internet Protocol |
| **ISO** | International Organization For Standardization |
| **LSM** | Linux Seurity Modules |
| **MAC** | Mandatory Access Control |
| **MCS** | Multi Category Security |
| **MIB** | Management Information Base |
| **MLS** | Multi Level Security |
| **MOF** | Managed Object Format |
| **NSA** | National Security Agency |

| | |
|---|---|
| **OID** | Object Identifier |
| **OS** | Operating System |
| **OSI** | Open Systems Interconnection |
| **PAP** | Policy Administration Point |
| **PBM** | Policy Based Management |
| **PDP** | Policy Decision Point |
| **PEP** | Policy Enforcement Point |
| **PRP** | Policy Repository point |
| **RBAC** | Role Based Access Control |
| **RHEL** | Red Hat Enterprise Linux |
| **ROSE** | Remote Operation Service Entity |
| **SCC** | Secure Computing Corporation |
| **SDN** | Software Defined Networking |
| **SELinux** | Security Enhanced Linux |
| **SMAEs** | Systems Management Application Entities |
| **SMASE** | Service Management Application Entity |
| **SMF** | System Management Fonctions |
| **SMI** | Structure of Management Information |
| **SMO** | Systems Management Overview |
| **SNMP** | *Simple Network Management Protocol* |
| **TE** | Type Enforement |
| **UDP** | User Datagram Protocol |
| **UML** | Infied Modeling Language |
| **WBEM** | Web Based Enterprise Management |
| **XML** | Extensible Markup Language |
| **XML** | Extensible Markup Language |

# INTRODUCTION

# Contexte et problématique

Ce projet vise l'amélioration de la gestion de la sécurité des systèmes distribués modernes qui sont de plus en plus larges, complexes et hétérogènes. De plus, et grâce aux nouvelles technologies telles que le Cloud Computing, la virtualisation et le SDN, les systèmes d'information sont devenus **élastiques**.

Les avantages et bénéfices gagnés par ces (r)évolutions ne sont pas sans poser des défis, en particulier ceux liés à la gestion de ces systèmes et en particulier leur sécurité.

La gestion dirigée par les politiques (*Policy Based Management* ou *PBM*) est une approche développée depuis les années 1990 qui permettrait de gérer des systèmes informatiques qui peuvent être larges et complexes et hétérogènes via des politiques de haut niveau exprimant des objectifs métiers. Le système sous-jacent s'occupera de traduire ces politiques en actions de bas niveau qu'il va appliquer d'une manière automatique dès que les conditions nécessaires sont réunies et veillera à les renforcer tant que ces conditions sont constatées.

Dans ce projet nous voulons appliquer les principes du PBM aux systèmes de contrôle d'accès obligatoire (*Mandatory Access Control* ou *MAC*) ces mécanismes sont présents dans les systèmes d'exploitation et permettent de contrôler les accès des processus aux objets d'un OS d'une manière fine via des règles d'accès de bas niveau.

Ce qui motive ce projet, est que dans un système informatique, plusieurs systèmes d'exploitation hétérogènes peuvent coexister, ayant chacun son propre système MAC, ce qui rend la tâche de renforcement des politiques de sécurité lourde et onéreuse en plus des erreurs et omissions qui caractérisent le travail manuel.

Nous voudrions donc d'un côté simplifier, unifier et homogénéiser la gestion des politiques d'accès applicables à un parc informatique via un standard de gestion largement adopté dans l'industrie, en l'occurrence CIM/WBEM et d'un autre côté, capitaliser l'expertise cumulée de la communauté qui a permis le développement d'une bibliothèque de politiques pour les systèmes MAC permettant de sécuriser une large gamme d'application et de services.

# Objectifs

L'objectif de ce projet est la réalisation d'un Framework, dédié à assister les administrateurs dans leurs tâches de modélisation, d'application et de suivi de politiques de sécurité de type MAC sur des systèmes distribués appartenant au même domaine d'administration.

La réalisation de ce système a comme objectifs :

1. Contribuer à la normalisation des politiques de sécurité sur deux plans :
    - Traduction de politiques exprimées en **langage SELinux** en politiques sous format **CIM**, ce qui va permettre, entre autres, de faciliter le partage et l'échange entre plusieurs administrateurs.
    - Intégrer le système **SELinux**, dans un dispositif global normalisé d'application de politiques de sécurité.
2. Centraliser la gestion de tous les systèmes qui mettent en œuvre des politiques SELinux dans le parc informatique.
3. Permettre la rédaction des politiques de sécurité abstraite de type MAC et les appliquer sur un système distribué.
4. Permettre la réutilisation des politiques développées par des tierces parties reconnues de confiance et pertinentes par rapport à une application ou un service donnée. Cette dernière fonctionnalité va permettre aux administrateurs de composer et combiner entre plusieurs politiques disponibles puis les appliquer sur des systèmes cibles.

## Organisation du mémoire

Dans cette partie introductive, nous avons expliqué le contexte de notre travail avant d'aborder les objectifs de notre travail. Notre travail est subdivisé en cinq chapitres et d'une conclusion générale. Les trois premiers chapitres traitent les aspects théoriques et les deux derniers traitent la conception et la réalisation de notre projet.

Nous présentons dans le premier chapitre les différentes approches de gestion existantes ainsi que les mécanismes de sécurité afin de justifier le choix des méthodes adoptées dans notre travail.

Dans le deuxième chapitre, nous étudions le standard de gestion adopté dans notre projet, en l'occurrence CIM/WBEM en se focalisant sur les profiles *CIM Policy Model* et *CIM Simplified Policy Language,* ses derniers sont dédiés à la modélisation des politiques génériques en plus de l'étude de profiles CIM spécialisés en contrôle d'accès.

Dans le troisième chapitre, nous détaillons le principe de fonctionnement de SELinux, qui représente le système de contrôle d'accès que nous utilisons pour concrétiser nos politiques de sécurité.

Le quatrième chapitre présente la conception de notre système de gestion. Nous présentons l'architecture générale de notre dispositif de gestion des politiques ainsi qu'une amélioration des éléments de modélisation de ces politiques fondés sur des profiles spécialisés du standard CIM développé antérieurement (dérivé des modèles *CIM Policy Model*) afin de représenter pertinemment des politiques de sécurité de type MAC et permettant en plus l'intégration des politiques existantes ou proposés par des parties tierces.

Dans le dernier chapitre, nous présentons les détails de la réalisation de notre système ainsi que le développement des composants logiciels qui assurent, d'un côté, la traduction des règles de sécurité exprimées en langage CIM en règles de sécurité applicables par SELinux et d'un autre retrouver les politiques SELinux déjà appliquées et les traduire en format CIM.

Nous terminons ce document par une conclusion générale ainsi que quelques perspectives.

# CHAPITRE 1 : GESTION DES SYSTEMES ET RESEAUX – CONTROLES D'ACCES

# 1.1 Introduction

Si l'objectif final de toute plate-forme de gestion de réseaux et de services est toujours de faciliter l'administrateur dans sa tâche quotidienne, les modèles et les approches employés ont évolué dans le temps au gré des nouvelles problématiques posées.

Pour cela, nous allons étudier et analyser quelques architectures de gestion existantes. Nous commençons par l'architecture de gestion ISO, qui est définie dans les normes « Cadre de gestion » et « Système de gestion ». Par rapport aux autres architectures de gestion de réseau, l'architecture ISO a reçu la plus grande attention au sein de la communauté de recherche. Ensuite, nous abordons l'approche de gestion définie par l'IETF, cette dernière repose sur le protocole « Simple Network Management Protocol » (SNMP) dont le but est de répondre aux besoins de gestion immédiats d'Internet. Enfin, nous introduisons l'initiative « Web Based Entreprise Management » (WBEM) du DMTF qui appréhende la difficulté de l'hétérogénéité posée par la multiplication des solutions de gestion propres à des domaines spécifiques, standardisées ou propriétaires en unifiant ces architectures pour un contrôle global des réseaux et des systèmes.

La sécurité des systèmes et des réseaux est une responsabilité fondamentale en matière de gestion. Presque toutes les applications qui traitent la sécurité, incluent une certaine forme de contrôle d'accès. Pour cela, nous présentons d'abord les différents concepts de contrôles d'accès. Ensuite, nous discutons les politiques de sécurité, les modèles de contrôles d'accès ainsi que quelques mécanismes de sécurité.

# 1.2 La gestion des systèmes et réseaux

La gestion des systèmes et réseaux constitue un problème dont l'enjeu est de garantir au meilleur coût non seulement la qualité du service global rendu aux utilisateurs mais aussi la réactivité face aux besoins de changement et d'évolution.

## 1.2.1 Concepts et principe de la gestion

Nous commençons par définir la gestion de réseaux. Ensuite, nous abordons les différentes activités ainsi que l'architecture général de la gestion de réseau.

### 1.2.1.1 La gestion de réseaux

La gestion du réseau est définie comme l'**exécution** de l'**ensemble des fonctions** requises pour **contrôler**, **planifier**, **affecter**, **déployer**, **coordonner** et **surveiller** les **ressources** d'un réseau informatique ou d'un réseau de télécommunications [29].

### 1.2.1.2 Les activités de la gestion des réseaux et systèmes

Le coût et la qualité de service qui se décline sur plusieurs critères, du point de vue de l'utilisateur final, notamment la disponibilité, la performance (temps de réponse), la fiabilité, la sécurité sont des contraintes qui doivent être respectées par des activités de gestion. Ces activités sont communément classées en activités de [30] :

- **Supervision :** l'activité de supervision consiste à surveiller les systèmes et à récupérer des informations sur leur état et leur comportement afin d'assurer leur bon fonctionnement.
- **Administration :** l'administration réseau désigne l'ensemble des opérations de contrôle du réseau ainsi que la gestion des configurations et de la sécurité.
- **Exploitation :** l'ensemble des activités permettant d'exécuter des applications qui font appel à des protocoles implémentés par les systèmes d'exploitation, afin d'exploiter les réseaux et leurs services et de traiter les problèmes opérationnels sur le réseau : maintenance, assistance technique…

### 1.2.1.3 Architecture de gestion de réseau

Bien que chaque protocole de gestion de réseau ait sa propre architecture, un modèle général ou basique d'architecture de réseau est conçu avec les éléments clés suivants [31] :

- **Station de gestion :** est utilisé pour gérer et superviser l'ensemble du réseau. Elle reçoit toutes les informations et l'affiche.
- **Agent de gestion** : L'agent est un programme logiciel dans le dispositif réseau qui répond aux demandes d'informations ou d'actions émises par la station de gestion.
- **Base d'information de gestion** : La troisième partie de l'architecture est l'information échangée entre le gestionnaire et l'agent. C'est ce qu'on appelle la base d'information de gestion ou la MIB. Cette information est une collection d'objets ou de valeurs de données. Chacun représente un aspect du dispositif géré.

- **Protocole de gestion de réseau** : Un protocole de gestion de réseau est utilisé pour transmettre des informations de gestion entre les agents et les stations de gestion en spécifiant les règles de communication. Le protocole fonctionne également entre l'entité gestionnaire et les entités gérées.

## 1.2.2 Les standards de gestion

Nous présentons dans cette section, les approches dites classiques c'est-à-dire celles de l'ISO et de l'IETF. Ensuite, nous introduisons l'initiative WBEM du groupe DMTF qui tente d'unifier les différentes technologies de gestion.

### 1.2.2.1 L'approche de gestion de l'ISO : OSI Management

L'ISO (**International Organisation for Standardisation**) a développé le cadre de gestion OSI (**OSI Management Framework**) [ISO-7498-4, Genève, 1989] qui a été complété par la vue d'ensemble de la gestion des systèmes (OSI Systems Management Overview) [ISO 10040, Genève 1992]. Ensemble, ces normes servent de base à la gestion OSI.

#### 1. Le cadre de gestion OSI (OSI Management Framework)

Le cadre de gestion est la première norme dans laquelle la gestion OSI est définie [1]. Nous discutons d'abord dans cette partie les domaines fonctionnels pour lesquels la gestion OSI a été développée. Ensuite, nous présentons les méthodes possibles d'échanger les informations de gestion. Enfin, nous traitons les objets gérés, les informations de gestion et la base d'informations de gestion (MIB).

**A. Domaines fonctionnels de la gestion**

Le cadre de gestion OSI propose la répartition des activités de gestion en fonction des domaines fonctionnels. Ces fonctions de gestion ont progressivement évolué vers ce que l'on appelle actuellement les cinq domaines fonctionnels de l'OSI. Pour désigner ces domaines, le terme « **FCAPS** » est communément utilisé pour **Fault Configuration Accounting Performance Security**. Ces fonctions se résument en [5] :

- **La gestion des pannes (Fault Management)** : Les installations qui permettent la détection, l'isolement et la correction d'un fonctionnement anormal de l'environnement OSI, a pour but de déceler et de résoudre les pannes qui se produisent dans un réseau.
- **La gestion de la configuration (Configuration Management)** : Les installations qui exercent le contrôle, identifient, collectent des données et fournissent des données à des

objets gérés afin d'aider à assurer le fonctionnement continu des services d'interconnexion. Elle a pour objectif de maintenir les composantes du réseau, de les surveiller et de les faire évoluer.

- **La gestion de la comptabilité (Accounting Management)** : Les installations qui permettent d'établir des charges pour l'utilisation d'objets gérés et les coûts à identifier pour l'utilisation de ces objets gérés, elle administre les coûts et les usages des services (par exemple, métriques de consommation d'un service, facturation d'un service, etc.).
- **La gestion de la performance (Performance Management)** : Les installations nécessaires pour évaluer le comportement des objets gérés et l'efficacité des activités de communication. Elle prend en charge la qualité de fonctionnement des services.
- **La gestion de la sécurité (Security Management)** : Les aspects de la sécurité OSI sont essentiels pour gérer correctement la gestion du réseau OSI et pour protéger les objets gérés. Elle tente de garantir la protection et le bon usage des services.

**B. Échange d'informations de gestion**

Trois façons différentes d'échanger des informations de gestion sont identifiées dans le cadre de la gestion OSI comme suit [1]:

- **Gestion du système (System Management)** : contrôle les actions de gestion de chaque système en fournissant des informations de gestion, en tant qu'agent ou en demandant des informations de gestion en tant que gestionnaire. La communication est effectuée au niveau de la couche application.
- **Gestion des couches ((N) - Layer Management)** : contrôle les paramètres de communication spécifiques sur la couche N, vérifie la fonctionnalité de la couche (N-1) et collecte les erreurs et les anomalies d'information de la couche N.
- **Gestion de l'opération de couche ((N) - Layer Operation Management)** : contrôle des instances particulières d'une connexion. Il est de la responsabilité du protocole de distinguer les informations de gestion des autres types de gestion.

**C. Objets gérés, informations de gestion et MIB**

Les concepts de gestion d'information ont été traités dans le cadre de gestion OSI, ces concepts sont définis comme suit [1] :

- **Objet géré (Managed Object)** : « *Toute ressource de traitement et de communication de données (qu'elles soient ou non OSI) qui peut être gérée par l'utilisation d'un protocole de gestion OSI* ».
- **Informations de gestion (Management Information)** : « *Informations associées à un objet géré exploité par le protocole OSI Management pour contrôler et surveiller cet objet* ».
- **Base d'information de gestion (Management Information Base)** : La MIB peut être considérée comme une sorte de base de données dont le contenu n'est pas l'ensemble des objets gérés eux-mêmes, mais les informations associées aux objets gérés.

## 2. OSI Systems Management Overview

La définition de OSI Systems Management Overview (SMO) vient compléter le cadre de gestion OSI. Par rapport à ce dernier, le SMO contient beaucoup plus d'informations et est bien mieux accepté. Le SMO distingue les aspects suivants [6] :

### A. Le modèle d'information de gestion

Le modèle d'information décrit les objets gérés, représentant la perception qu'a le système de gestion des ressources, mais aussi les objets de support de gestion qui sont des objets participant aux activités de gestion. Cette description est complétée par une série de standards complémentaires dont le tout constitue le **SMI (Structure of Management Information)** [2]. La structure de l'information de gestion du modèle OSI utilise une approche orientée objet où chaque équipement est modélisé par des classes d'objets organisés par des liens d'héritage et de dépendance. L'ensemble des objets gérés forme la **MIB (Management Information Base)**. Le langage **GDMO (Guideline for the Definition of Managed Objects)** permet la définition non ambiguë de toutes les caractéristiques des classes d'objets [5].

### B. Le modèle organisation

L'aspect organisation décrit le concept manager-agent correspondant à l'approche centralisée adoptée par l'OSI. Selon cette approche, un seul manager peut contrôler plusieurs agents. Le manager effectue des opérations sur les agents, les agents transmettent des notifications à leurs managers. Le concept manager-agent est illustré dans la figure 1.1. La gestion OSI offre aussi la possibilité de découper l'environnement de gestion en domaines de gestion. L'idée étant de faciliter la gestion en offrant des vues fonctionnelles, géographiques, ou technologiques des réseaux [1,2].

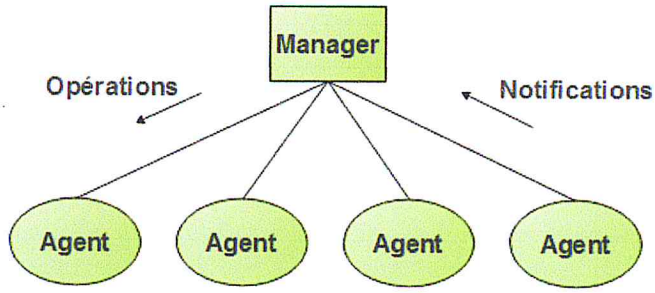



Figure 1.1. Concept Agent-Manager [1].

### C. Le modèle fonctionnel

En plus des cinq aires fonctionnelles du cadre de gestion OSI, l'ISO a commencé à définir des normes de protocole pour chacun d'eux. Après un certain temps, une observation intéressante a été faite : la plupart des protocoles de zone fonctionnelle ont utilisé un ensemble similaire de fonctions élémentaires de gestion [1]. Ces fonctions élémentaires qui sont définies à un niveau d'abstraction beaucoup plus faible que les zones fonctionnelles d'origine, sont appelées : « **Systems Management Fonctions – SMF** ». Les SMFs fournissent ensemble une plate-forme générique de capacités de gestion de systèmes pour diverses applications de gestion [7]. La figure 1.2 montre la hiérarchie fonctionnelle des SMF et des fonctions du modèle FCAPS.

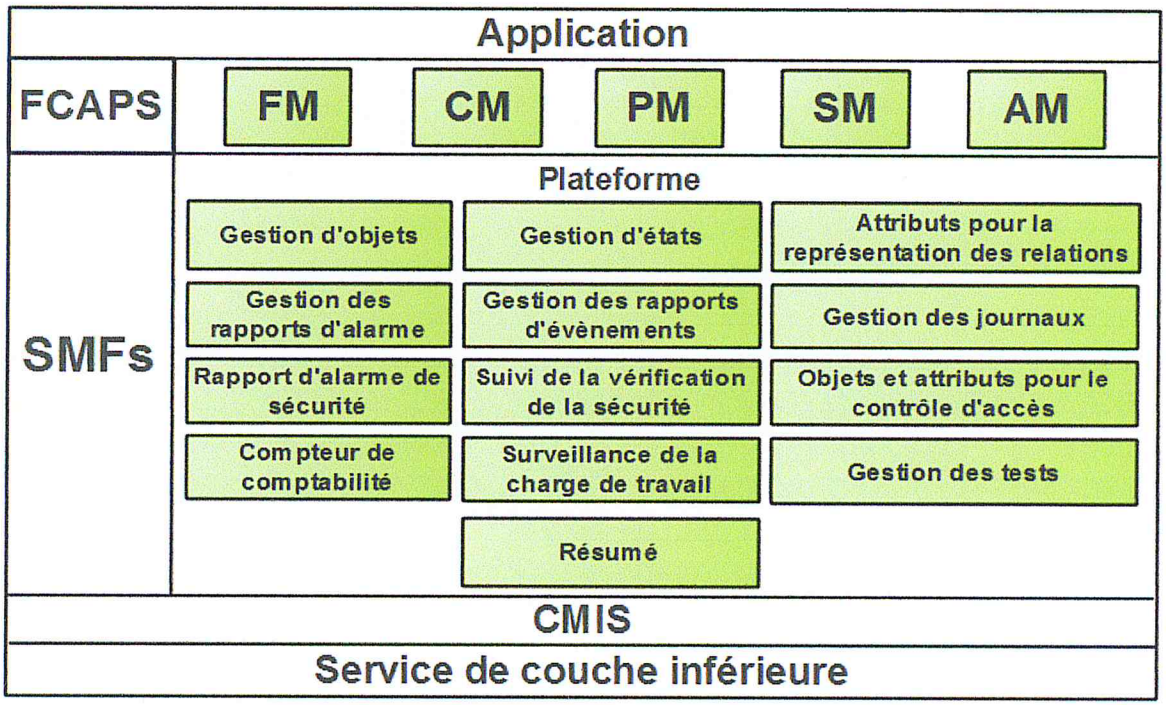



Figure 1.2. Hiérarchie fonctionnelle des SMFs et des fonctions du modèle FCAPS [7].

### D. Le modèle de communication

Ce modèle décrit les services d'échange d'informations, l'OSI a défini le **CMIS (Common Management Information Service)** comme le service d'échange d'informations de gestion. **CMIS** peut être décomposé dans le cas où deux ou plusieurs entités **SMAEs (Systems Management Application Entities)** apparaissent. Ces entités contiennent un nombre de : **ASEs (Application Service Entities)** dont le **SMASE (Service Management Application Entity)** fait le lien avec les processus applicatifs de gestion, le **ACSE (Association Control Service Entity)** gère l'association à travers laquelle se déroule l'échange de données applicatives, le **ROSE (Remote Operation Service Entity)** est l'élément de service d'opération distante et le **CMISE (Common Management Information Service Entity)** gère les échanges entre les entités de gestion système. Ce dernier élément comprend le protocole de gestion **CMIP (Common Management Information Protocol)** qui définit les procédures de transmission de l'information de gestion [1,2].

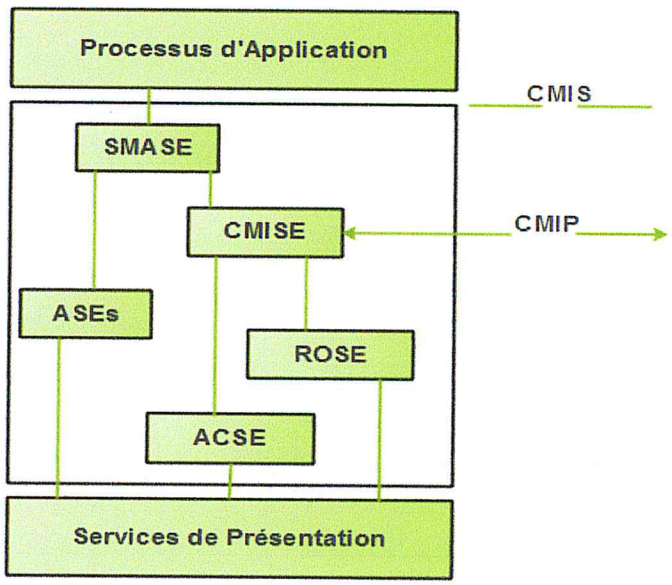



Figure 1.3. Modèle de communication au niveau de la couche d'application [5].

## Évaluation du modèle de gestion OSI

Le modèle de gestion OSI présente certaines caractéristiques positives comprennant :

- Un puissant groupe de services (CMIS / CMIP).
- Des installations de filtrage qui réduisent le trafic de gestion.
- Une architecture modulaire qui, en principe, facilitera la mise en œuvre.

Les facteurs positifs sont largement chevauchés par les caractéristiques négatives qui ont été le motif principal de la dégradation successive de cet effort de normalisation. Parmi ces caractéristiques négatives :

- Difficulté d'adaptation à la gestion de système et de périphérique.
- Complexité de la solution de gestion, ceci est visible au nombre et au volume des standards.
- Coût élevé dû au processus de normalisation sur lequel il repose qui est trop long.
- La non correspondance de certaines nouvelles technologies au modèle OSI.

### 1.2.2.2 L'approche de gestion de l'IETF : SNMP

Le protocole de gestion de réseau simple (*Simple Network Management Protocol - SNMP*) a été introduit en 1988 pour répondre au besoin croissant d'un standard pour la gestion des équipements réseau du protocole Internet (IP). Le protocole SNMP couvre l'ensemble des standards de gestion préconisés par la communauté Internet (IETF) [14]. SNMP fournit un standard pour surveiller et contrôler un réseau, de manière indépendante du fournisseur. D'autre part, il permet la récupération et la modification des informations de réseau entretenues par les hôtes et les routeurs connectés à un réseau. Un administrateur réseau peut utiliser SNMP pour diagnostiquer et corriger les problèmes de réseau à partir d'hôtes distants [31].

#### 1. Architecture

L'architecture SNMP est basée sur le paradigme manager-agent et comprend cinq composants de base [4] :

**A. Le gestionnaire (Manager)**

Le gestionnaire est la base de la gestion du réseau, il gère et mène des tâches d'exploitation et de gestion pour les périphériques gérés.

**B. L'agent (Agent)**

L'agent est un logiciel qui réside dans les dispositifs gérés et facilite les tâches de communication et de gestion entre le gestionnaire et ces dispositifs gérés. Un agent doit répondre à la demande de lecture et d'écriture d'informations, provenant de la station de gestion, ou il peut fournir des notifications asynchrones sur des événements ou des conditions internes.

**C. Le dispositif géré (Managed Device)**

Est un élément actif de réseau qui rassemble les informations et les rend disponibles pour le gestionnaire utilisant l'agent SNMP.

**D. Les informations de gestion (Management Information)**

Tout comme les MIBs de OSI, la structuration de l'information de MIBs de l'IETF est aussi orientée objet. Néanmoins, la modélisation de l'information de gestion ne se fait ici qu'au moyen de simples variables. Dans SNMP, les ressources sont représentées comme des objets gérés en utilisant le mécanisme défini dans la structure de l'information de gestion (SMI). La collection de tous ces objets gérés est appelée base d'informations de gestion. L'objet géré contient normalement trois attributs, identificateur d'objet (OID), type et syntaxe et encodage. Les identificateurs d'objets sont utilisés pour identifier des objets gérés où tous les objets gérés sont organisés en arborescence. Les OID de niveau supérieur dans l'arbre représentent différentes organisations standard, tandis que les fournisseurs définissent des branches privées, y compris des objets gérés pour leurs propres produits.

**E. Le protocole de gestion de réseau (Management Protocol)**

Le transfert d'informations de gestion se fait à travers SNMP qui est un protocole de couche d'application utilisé pour accéder à des objets gérés. Ce protocole se décline en trois versions représentant son évolution face aux nouveaux besoins comme la facilité de gestion ou la sécurité.

## 2. Versions du protocole SNMP

Il existe trois versions de SNMP : **SNMPv1**, **SNMPv2** et **SNMPv3**. Les deux versions 1 et 2 comportent un certain nombre de fonctionnalités en commun, mais SNMPv2 offre des améliorations, telles que des opérations de protocole supplémentaires. SNMP version 3 (SNMPv3) ajoute des fonctionnalités de sécurité et de configuration à distance aux versions précédentes. Une brève description des versions SNMP est donnée ci-dessous [31,4]:

- **SNMPv1** : sa mise en œuvre est simple et contient un petit ensemble de commandes opérationnelles et comporte de mauvaises procédures de sécurité.
- **SNMPv2** : comprend plus de commandes opérationnelles, définit plus de types de données SMI et tente de fournir de meilleures procédures de sécurité, mais

malheureusement, il ne pouvait pas résoudre complètement les problèmes de sécurité et n'était pas compatible avec la version précédente.

- **SNMPv3** : la dernière version et la plus avancée, offre une sécurité améliorée, un contrôle d'accès, une configuration à distance de paramètres SNMP et est également compatible avec les versions précédentes.

## 3. Evaluation du modèle de gestion SNMP

La mise en place d'un réseau compatible SNMP offre des avantages importants. Parmi ces avantages [31] :

- Il inclue les capacités pour prévenir, détecter et corriger les problèmes liés au réseau.
- Il est facile à utiliser et permet aux administrateurs de contrôler le besoin de maintenir un réseau sain.
- Il fournit aux administrateurs un mécanisme de gestion de réseau qui surveille efficacement les performances du réseau.
- Sa popularité : il est pratiquement supporté par tous les fabricants d'équipement de réseau dans le monde. Son système de gestion centralisé est une solution extrêmement efficace et répandue à la gestion du réseau.
- Son efficacité : il utilise également le protocole de transport UDP. UDP est un protocole de transport plus rapide et présente donc de faibles coûts généraux. Mais il présente l'inconvénient d'être non fiable, on compte généralement sur la fiabilité du réseau sous-jacent.

La simplicité et la facilité de mise en œuvre de SNMP sont les principales raisons de son implantation rapide et de sa généralisation dans les éléments et dispositifs des réseaux. Cette simplicité a aussi créé des problèmes :

- La pile de protocoles SNMP plus simple avec moins de commandes opérationnelles est insuffisante et ne peut fournir une évolutivité.
- SNMP est basé sur le protocole sans connexion UDP qui le rend peu fiable, il n'y a aucun moyen d'être assuré que les commandes émises ont fonctionné selon les exigences.
- Les objets gérés définis dans SNMP sont basés sur des variables orientées et n'ont pas de propriétés héritées. SNMP gaspille la bande passante avec des informations inutiles réalisées dans chaque message.

- SNMP n'est pas encore préparé à supporter la distribution notamment en raison de la structure du système d'information utilisée.
- Les quelques 3000 MIBs propriétaires qui sont utilisées, représentent une entrave indéniable à l'interopérabilité des systèmes de gestion.
- SNMP n'a été qu'une solution provisoire développée par l'IETF en attendant le développement de standards complets.
- La version de SNMP la plus utilisée par les agents est la version 2 avec ses lacunes en sécurité.

### 1.2.2.3 L'approche du DMTF : WBEM

L'initiative Web Based Entreprise Management (WBEM) élaborée par le **Distributed Management Task Force (DMTF)** est une norme de gestion unifiée dont l'objectif premier était d'unifier les normes de gestion de l'informatique et de la télécommunication avec un modèle d'information commun.

WBEM définit un groupe de technologies et d'outils qui seront détaillés dans le chapitre suivant et sont abordés brièvement ci-dessous [4,28]:

- **Le modèle d'information commun (Common Information Model - CIM)**

Il est basé sur une représentation orientée objet dont l'objectif principal est de fournir un modèle unifié pour représenter tout type de données, plate-forme, application, périphérique, réseau, etc. CIM se compose d'un schéma et d'un ensemble de spécifications.

- **Le schéma CIM**

Il décrit la modélisation exacte. Le langage choisi pour exprimer le schéma CIM sous forme textuelle s'appelle **MOF (Managed Object Format)** tandis que le langage choisi pour la modélisation visuel du schéma CIM est **UML (Unified Modeling Language)**.

- **Les spécifications CIM**

Ils concernent les détails d'intégration avec d'autres modèles de gestion.

- **Le schéma d'accès WBEM**

Il est basé sur **CIM via HTTP**, où HTTP est utilisé pour le transport de données, tandis que le codage **xmlCIM** est utilisé pour exprimer les informations de gestion. Les hôtes WBEM communiquent entre eux de manière ouverte et standardisée en échangeant des documents XML via HTTP. Le protocole HTTP est défini par DMTF comme protocole de communication, mais il existe d'autres produits disponibles qui sont basés sur WBEM.

- **Le codage de transport WBEM**

Il est désigné par xmlCIM et il utilise **XML** (**eXtensible Markup Language**) pour coder les informations CIM selon un **DTD** standardisé (**Definition Type Document**).

- **Le gestionnaire d'objets CIM (CIM Object Manager - CIMOM)**

Il est l'entité centrale avec le référentiel CIM (CIM Repository) avec lequel on interagit pour collecter des informations sur les ressources gérées tandis qu'une API XML est utilisée pour accéder au référentiel principal.

**Evaluation du modèle de gestion WBEM**

WBEM apporte de nombreux avantages, parmi lesquels :

- La résolution des problèmes persistants d'interopérabilité.
- L'amélioration des capacités de gestion par abstraction et décomposition des processus et des services d'affaires.

WBEM n'est pas non plus sans échappatoires, car :

- WBEM est basé sur XML, qui manque de la représentation adéquate de la base de données relationnelle et sa description n'est pas compréhensible pour la machine, il est donc incapable de fournir une définition de métadonnées sur les ressources Web.
- La dépendance de l'architecture du protocole d'accès sur HTTP apporte des problèmes inhérents tels que l'absence de procédure pour fournir une notification, de sorte que XML sur HTTP manque d'une meilleure communication bidirectionnelle.

### 1.2.3 Comparaison des approches de gestion

Une comparaison entre le standard le plus utilisé pour la gestion des réseaux informatiques, à l'instar SNMP et le standard de gestion WBEM est présentée dans le tableau 1.1 ci-dessous :

| Environnements | SNMP | WBEM |
|---|---|---|
| **Interopérabilité des données** | Moyen, données comme décrit dans la MIB, peut être spécifique à la plateforme | Haute, comportement des méta-données sur toutes les plates-formes/ fournisseurs |
| **Sécurité** | Moyenne, DES | Haute, les implémentations utilisent le protocole HTTPS. |
| **Évolutivité du modèle** | Moyen, limité à une vue des ressources et une petite syntaxe du langage. | Élevée, beaucoup de modèles (classes) et des constructions de langage. |
| **Extensibilité de mise en œuvre** | Moyen, beaucoup de mises en œuvre, mais pas d'interface API commune. | Élevé, beaucoup d'implémentation et API Interface plateforme de gestion commune (CMPI) à travers des implémentations Open Source WBEM. Il existe d'autres APIs : PEGASUS, OpenWBEM, WMI (Microsoft) |
| **La vitesse de récupération/ Interrogation des données** | Moyen. Réservé aux deux opérations (get/ set). | Haute, opérations existent pour fournir un accès à grains fins données. |
| **La mise en œuvre Open Source** | Oui. Une mise en œuvre Open Source : NetSNMP. Beaucoup d'implémentations propriétaires. | Oui. Trois implémentations Open Source : OpenPegasus (Principale), SBLIM Project (supporte l'interface CMPI), OpenWBEM, WBEMSource. Aussi existe des implémentations propriétaires (WMI de Microsoft, IBM, ORACLE, etc). |

Tableau 1.1. Comparaison de l'approche de gestion SNMP et WBEM [15].

Par rapport à la solution de gestion SNMP, WBEM a tous les avantages que présentent les solutions existantes. En outre, bien que les solutions existantes offrent une ressource ou une vision orientée problème, WBEM offre des vues et plus encore. Il offre également une grande vue d'image, permettant à l'utilisateur de se concentrer sur la grande image de l'écosystème des systèmes, des réseaux, des utilisateurs, des politiques, des applications, etc. et se concentrer uniquement lorsque cela est nécessaire sur la vue détaillée [33]. De plus, les données sont

interopérables ce qui signifie qu'elles portent le même sens sur différentes plates-formes. Pour cela, l'approche de gestion WBEM a été choisie dans le cadre de ce projet.

## 1.3 Les contrôles d'accès

Le contrôle d'accès est le processus qui détermine si un utilisateur a la permission de mener une action spécifique sur les ressources du système [18]. C'est un mécanisme qui consiste à accorder ou interdire le droit d'accès à une entité active (sujets : utilisateurs, processus, etc.) pour effectuer des opérations sur des entités passives (objets : fichier, dossier, etc.) [13].

### 1.3.1 Concepts de base des contrôles d'accès

Dans cette section, nous présentons les objectifs des contrôles d'accès ainsi que les phases d'élaboration d'un système de contrôle d'accès.

#### 1.3.1.1 Les objectifs du contrôle d'accès

Le contrôle d'accès vise à préserver la **confidentialité**, la **disponibilité** et l'**intégrité** des données [16].

- **La confidentialité :** assurer que l'information ne soit accessible qu'à ceux qui en ont l'autorisation.
- **La disponibilité :** garantit que les éléments considérés sont accessibles au moment voulu par les personnes autorisées.
- **L'intégrité :** assurer que l'information ne puisse être modifiée par des personnes non autorisées.

#### 1.3.1.2 Les phases d'élaboration d'un système de contrôle d'accès

Afin d'atteindre les objectifs cités précédemment, différents concepts ont été définis. Ces concepts décrivent comment le contrôle d'accès devrait être défini d'un niveau abstrait à un niveau concret. Ici, nous présentons ceux décrits dans [17] :

- **Politique de sécurité :** Elle décrit la vue la plus abstraite du système. À ce niveau, les règles de contrôle d'accès sont définies. Les exigences du système sont décrites afin de se conformer à certaines spécifications. Cette description ne fournit aucune méthode sur la façon dont elle devrait être appliquée. Ces politiques ne sont qu'un ensemble de règles définissant qui est autorisé à accéder à quoi et à quelles conditions et aux critères sous lesquels cette autorisation est donnée ou annulée.

- **Modèle de sécurité** : Il formalise les règles définies dans la politique de sécurité et décrit la manière dont elles devraient fonctionner.
- **Mécanisme de sécurité** : Il décrit les méthodes de bas niveau qui sont utilisées pour appliquer les règles formalisées au niveau du modèle de sécurité.

Les modèles de contrôle d'accès sont importants en raison de leur capacité à combler l'écart d'abstraction entre les politiques et les mécanismes d'accès.

Plutôt que de tenter d'évaluer et d'analyser les systèmes de contrôle d'accès exclusivement au niveau du mécanisme, les modèles de contrôle d'accès sont généralement écrits pour décrire les propriétés de sécurité d'un système de contrôle d'accès.

## 1.3.2 Les politiques de sécurité

Le principal problème du concept de politique de gestion est le nombre de définitions énoncées. La définition de politique dans le cadre de la gestion unanimement admise est celle donnée dans [34] :

**Définition** :

Les politiques sont des règles qui gouvernent les choix de comportement d'un système.

Différents niveaux d'abstractions sont utilisés pour exprimer les règles allant des règles exprimant des objectifs jusqu'au règles exprimant des paramètres de configurations spécifiques à chaque système. Le processus de raffinement des règles abstraites en règles plus concrètes est appelé processus de dérivation de la politique. Il existe deux types de politique et donc de moyens pour les exprimer :

- Les ***politiques d'obligation*** sont de la forme un *événement* déclenché dans une *condition* particulière *entraîne* une *action* particulière ;
- Les ***politiques d'autorisation*** qui reflètent l'approche utilisée par les modèles de contrôle d'accès spécifient les services et ressources auxquelles un sujet peut accéder.

## 1.3.3 Les modèles basiques de contrôle d'accès

Plusieurs modèles de contrôle d'accès ont été proposés afin de garantir une accessibilité sécurisée aux ressources d'un système d'information. Parmi ces modèles :

### Contrôle d'accès discrétionnaire (DAC – Discretionnary Access Control)

Dans DAC, l'utilisateur possède l'autorité complète sur toutes les ressources dont il possède et détermine également les autorisations pour les autres utilisateurs disposant de ces ressources et programmes.

### Contrôle d'accès obligatoire (MAC – Mandatory Access Control)

Dans cette catégorie de politiques de sécurité, les utilisateurs n'ont pas le pouvoir d'annuler les règles et ils sont entièrement contrôlés centralement par l'administrateur de la politique de sécurité. L'administrateur de la politique de sécurité définit l'utilisation des ressources et leur politique d'accès, qui ne peut pas être annulée par les utilisateurs finaux, et la politique, décidera qui a le pouvoir d'accéder aux programmes et aux fichiers particuliers.

### Contrôle d'accès basé sur les rôles (RBAC – Role Based Access Control)

Cette politique est très simple à utiliser. Dans le RBAC, les rôles sont attribués par l'administrateur système de manière statique. Dans lequel l'accès est contrôlé en fonction des rôles que les utilisateurs ont dans un système. (RBAC) est principalement utilisé pour contrôler l'accès aux ressources informatiques ou réseau en fonction des rôles des utilisateurs individuels au sein d'une organisation.

Le système d'exploitation Linux repose sur le contrôle d'accès DAC qui présente des limitations, notamment l'utilisation de seulement deux grandes catégories d'utilisateurs qui sont les administrateurs et les autres et le principe très dangereux du super utilisateur root qui peut tout faire.

Pour pallier la faiblesse du DAC, nous allons utiliser l'un des mécanismes fondés sur les principes du MAC qui ont été implémenté sous la forme de modules au sein du noyau Linux.

## 1.3.4 Les mécanismes de sécurité

Plusieurs projets ont développé des modules de sécurité pour le système d'exploitation Linux. Les deux plus répandues étant : SELinux et AppArmor. Mais d'autres existent également comme GrSecurity et TOMOYO.

Dans cette partie nous présentons et nous analysons les trois modules : SELinux, AppArmor et GrSecurity afin de choisir le module qu'on va utiliser dans ce projet.

### 1.3.4.1 SELinux

Security-Enhanced Linux (SELinux) est une implémentation d'un mécanisme MAC dans le noyau Linux, initialement développé par la NSA et la Secure Computing Corporation (SCC) [13]

Security-Enhanced Linux (SELinux) est une fonctionnalité Linux qui fournit une variété de politiques de sécurité pour le noyau Linux. Il est inclus avec CentOS / RHEL / Fedora Linux, Debian / Ubuntu, Suse, Slackware et bien d'autres distributions.

SELinux a trois modèles de politique obligatoire, qui travaillent ensemble pour formuler une politique MAC flexible et complète. Il existe aussi deux modèles facultatifs : Multi Level Security (MLS) et Multi Category Security (MCS).

Le contrôle d'accès basé sur l'identité (IBAC) sert à identifier le sujet, avec le droit d'utiliser une certaine identité du système, lors de la connexion. Les identités SELinux sont orthogonales aux UID Linux. Chaque fois qu'un UID d'un processus est modifié, son composant d'identité SELinux sera conservé.

Le contrôle d'accès basé sur les rôles (RBAC) restreint les actions d'un utilisateur en fonction d'un ensemble de rôles attribués à l'utilisateur. La transition entre les rôles est contrôlée par la politique.

Le contrôle d'accès basé sur le type, *Type Enforcement*, associe un type à chaque sujet et objet. Il est utile en termes d'intégrité, de séparation et de confinement.

Rien ne peut échapper à cette granularité si la politique est correctement construite. Cela constitue la plupart de la politique de SELinux.

### 1.3.4.2 AppArmor

AppArmor (Application Armor) précédemment connu sous le nom de SubDomain est un autre logiciel de sécurité pour Linux qui fonctionne avec les chemins de fichiers [9,10]. Il a d'abord été développé par la société Immunix, qui l'a utilisé dans son offre GNU / Linux entre 1998 et 2003. Immunix a été acquise par Novell en 2005 et a maintenu AppArmor sous GPL jusqu'en 2007, lorsque l'équipe d'AppArmor a été mise à pied [8].

Le module de sécurité AppArmor [13] fournit une méthode de contrôle d'accès centrée sur les applications. Il applique des politiques qui définissent les accès autorisés pour chaque application individuellement en se basant sur les chemins d'accès. Conçu pour être plus simple à utiliser que SELinux, il propose un mode apprentissage [23].

### 1.3.4.3 GrSecurity

Grsecurity est un ensemble de correctifs pour le noyau Linux avec un accent sur l'amélioration de la sécurité [10]. Il est associé au correctif PaX qui ajoute des protections au noyau Linux et aux programmes en espace utilisateur [13]. Il protège contre une large gamme de menaces de

sécurité grâce au contrôle d'accès intelligent, à la prévention de l'exploitation de la corruption de la mémoire et à une série d'endommagement du système qui ne nécessitent généralement aucune configuration [12]. Il met l'accent sur le principe de sécurité et applique MAC via RBAC. Il a été lancé par Brad Spengler en 2001 [8], il s'agissait à l'origine d'un port des fonctionnalités de sécurité d'Openwall et est depuis devenu un projet propre [9]. GrSecurity fournit un mécanisme MAC souple et flexible car il utilise les ACLs et RBAC.

### 1.3.5 Comparaison des mécanismes de sécurité

Le tableau 1.2 illustre une comparaison détaillée des trois mécanismes de sécurité présentés auparavant :

| | **SELinux** | **AppArmor** | **GrSecurity** |
|---|---|---|---|
| **Support du noyau** | Noyau principal (Mainline Kernel) | Patch | Patch |
| **Type de crochet** | LSM | LSM | GrSecurity |
| **Stockage d'étiquettes** | EA / métadonnées (dépendant du système de fichiers) | Interne uniquement (système de fichiers indépendant) | Extension des structures de données |
| **Apprentissage des politiques** | Audit2allow | Génération initiale de la politique et mises à jour incrémentales | Extensif, intégré |
| **Protection contre le débordement du tampon** | Exec-Shield | N/A | PaX |
| **Modèles de politique** | IBAC, TE, RBAC, MLS, MCS | Profils | ACL, RBAC |
| **Portabilité** | Oui | N/A | N/A |
| **Intégration par défaut et recommandée** | CentOS / RedHat / Debian | Suse / OpenSuse /UBUNTU | Toute distribution Linux |
| **Formation et support des fournisseurs** | Oui (RedHat / TRESYS + Communauté) | Oui (Novell / UBUNTU + communauté) | Non (forum communautaire et listes) |
| **Fonctionnalité** | Joindre des étiquettes à tous les fichiers, processus et objets | Le système basé sur Pathname ne nécessite pas l'étiquetage ou le système de ré-encre | ACLs |

Tableau 1.2. Comparaison détaillée de SELinux, AppArmor et GrSecurity [9].

Nous avons comparé ici trois mécanismes de sécurité MAC, et pour notre projet nous avons choisi d'implémenter le mécanisme SELinux qu'on considère prioritaire car il couvre la quasi-totalité des objets d'un système, et aujourd'hui il dépasse ce cadre aux bases de données (PGSQL) et la virtualisation (XEN et KVM) entre autres. Cependant l'objectif dans le futur est de couvrir le maximum de mécanismes MAC (SELinux, AppArmor et les autres). En effet nous voulons être capable un jour de configurer d'une manière centralisée, des systèmes utilisant des mécanismes MAC différents, grâce au langage CIM et son profile Policy.

## 1.4 Conclusion

Ce chapitre est découpé en deux grandes parties, la première partie a été consacré pour la présentation des concepts généraux de la gestion des réseaux, l'étude et l'analyse des principaux standards de gestion existants. Cette partie a été conclue par une comparaison entre l'approche de gestion populaire SNMP et la nouvelle approche de gestion WBEM qui présente un cadre unificateur dans la gestion des réseaux et systèmes et qui a été choisi dans la réalisation de notre Framework dédié à l'assistance dans la gestion des politiques de sécurité. La deuxième partie du chapitre a introduit les concepts de base des contrôles d'accès, politiques de sécurité qui les représentent, modèles qui les formalisent tels que le DAC, le MAC et le RBAC et mécanismes de sécurité qui l'implémentent. Les mécanismes de sécurité SELinux, AppArmor et GrSecurity ont été étudiés, analysés et au final comparés. Ces trois mécanismes offrent une très bonne protection et on peut les sélectionner en fonction de critères simples, d'après l'analyse faite on constate que GrSecurity est mieux adapté pour les nouveaux utilisateurs pour sa facilité d'utilisation, AppArmor fournit une politique et des outils facile à comprendre, alors que le mécanisme de contrôle d'accès le plus puissant est SELinux. La puissance de ce mécanisme ainsi que plusieurs autres critères nous ont poussé à l'adopter pour définir et appliquer les politiques de sécurité.

# CHAPITRE 2 : L'APPROCHE DE GESTION WBEM

## 2.1 Introduction

Web Based Entreprise Management (WBEM), conçue par la Distributed Management Task Force (DMTF) en 1996, vise à résoudre le problème des systèmes hétérogène en décrivant les ressources gérées selon un schéma commun. Les outils de gestion de système existants deviennent de plus en plus complexes, des solutions telles que SNMP n'ont pas assez de fonctionnalités. La grande diversité du matériel, chacun avec sa propre interface de gestion personnalisée, utilise plus de ressources et le temps de gérer. Le WBEM, avec ses racines dans des technologies éprouvées, offre un cadre commun adaptable par les fournisseurs, basé sur des standards, extensible, réutilisable et hautement adaptable au nouveau matériel. Tout cela se traduit par un chemin d'amélioration d'économie de coûts.

Pour comprendre l'approche WBEM, il est important de comprendre ses concepts et ses objectifs, son modèle d'architecture et les quatre normes principales qui la régissent. Ils sont la spécification CIM (Common Information Model), la spécification de codage CIM / XML, la norme de transport CIM sur HTTP et le schéma DMTF. Ces concepts seront présentés dans la suite de ce chapitre.

## 2.2 Présentation de WBEM

**WBEM (Web-Based Enterprise Management)** est un ensemble de **standards techniques d'Internet** pour le **management des réseaux et des systèmes** mis au point pour unifier la gestion des environnements informatiques distribués. Elle fournit à tous les acteurs, constructeurs et éditeur informatique, la capacité à livrer un **ensemble cohérent d'outils d'administration et de supervision** axée sur des normes, facilitant l'échange de données entre des technologies et des plateformes. Les principaux objectifs de cette approche sont :

- Amélioration du délai de mise sur le marché - en utilisant les normes.
- Temps de développement réduit - les modèles d'information existants peuvent être utilisés ou développés.
- Prise en charge des autres solutions de gestion - support de migration pour SNMP, DMI, etc.
- La création d'une vue homogène de l'ensemble des ressources gérées et ce quelles que soient leur nature, localisation et/ou méthode d'accès ;

En outre, il réutilise également de nombreux bons concepts éprouvés trouvés dans les solutions de gestion de système existantes, telles que les notifications basées sur évènements (notifications en SNMP), l'extensibilité (paradigme orienté objet), la sécurité (HTTPS) et plus encore.

## 2.3 L'architecture fonctionnelle

WBEM propose une architecture fonctionnelle dont le but est de classifier les acteurs dans le processus de gestion. L'architecture spécifie les **entités** d'une part et leur associe des **rôles**.

### 2.3.1 Les entités

Les entités que décrivent l'architecture WBEM sont au nombre de quatre, elles sont illustrées dans la figure 2.1 [34] :

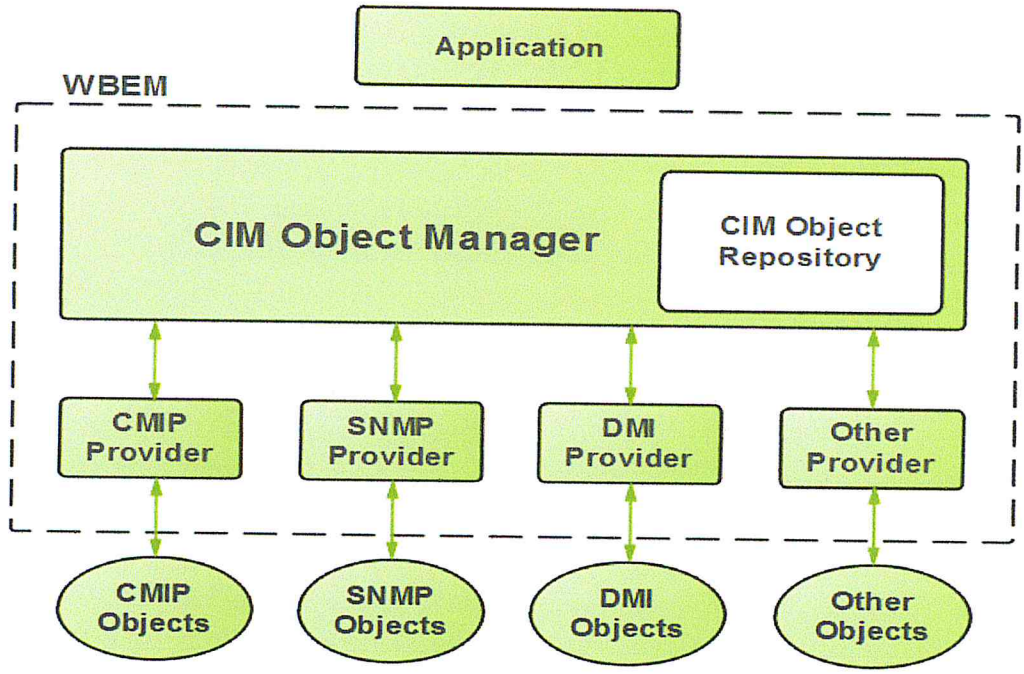



Figure 2.1. Les entités dans l'architecture WBEM [34].

1. **L'application de gestion (Client CIM)** : il s'agit d'une interface à travers laquelle on interagit avec le gestionnaire d'objets CIMOM (lecture/écriture d'information sur les objets gérés, invocation d'opérations sur les objets gérés).
2. **Gestionnaire d'objets CIMOM (Serveur CIM)** : il représente l'élément principal de l'architecture WBEM. Il permet de :
   - Maintenir la base d'informations de gestion (MIB) nommée ici **CIM Object Repository** qui représente une base de données objets contenant les informations de gestion existantes dans un système.

- Responsable de la communication entre les composants WBEM Server.
- Il offre aux applications une interface d'accès aux informations de gestion.
- Il donne une vue globale et homogène des ressources gérés.

3. **Provider CIM** : est une passerelle entre le gestionnaire d'objets CIMOM et les éléments gérés. Il a pour rôle de traduire en opération CIM les informations en provenance de sources hétérogènes externes. Un Provider présente deux interfaces : une interface CIM pour communiquer avec le CIMOM et une interface native pour communiquer avec l'élément géré.
4. **L'agent** : représente les ressources gérées (Hardware, Application, Service), il permet d'exécuter les opérations de gestion requises par le gestionnaire d'objets.

Le serveur CIM regroupe le CIMOM, le CIMOR et l'ensemble des providers. Il reçoit et traite les demandes et les réponses CIM.

L'architecture définie offre une couche d'abstraction permettant d'assurer une gestion homogène centralisée d'un ensemble de ressources hétérogènes distribuées, tout en conservant les technologies existantes.

L'architecture WBEM présentée est purement théorique. Le choix de la distribution de cette architecture sur le réseau a été laissé à l'implémentation. Généralement, dans une implémentation réelle, les entités WBEM sont réparties comme le montre la figure 2.2 :

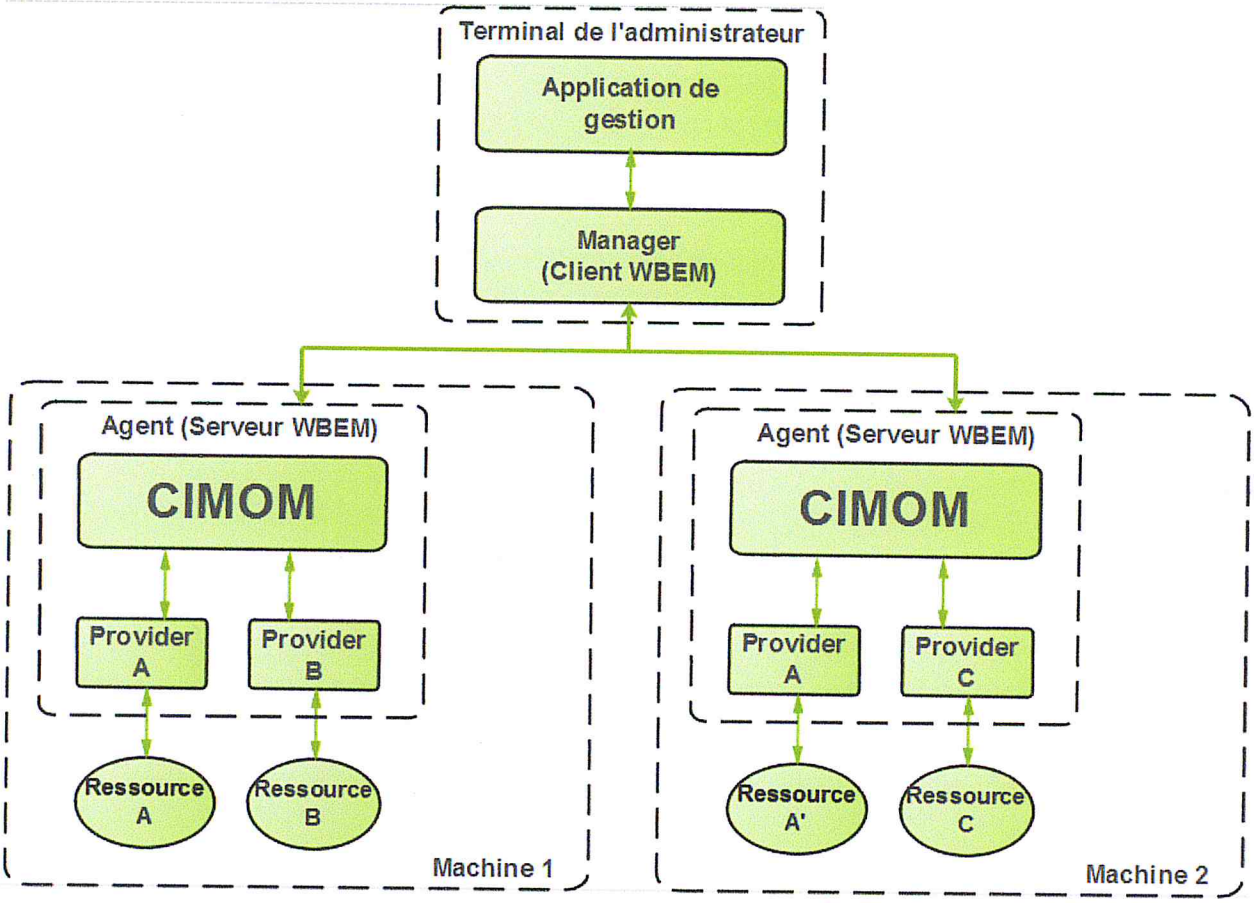



Figure 2.2. Distribution pratique de l'architecture WBEM.

## 2.3.2 Les rôles

Un rôle définit la capacité d'une entité dans une interaction de gestion, un ou plusieurs rôles peuvent être associés à une entité. Les rôles sont illustrés dans la figure 2.3 [34] :

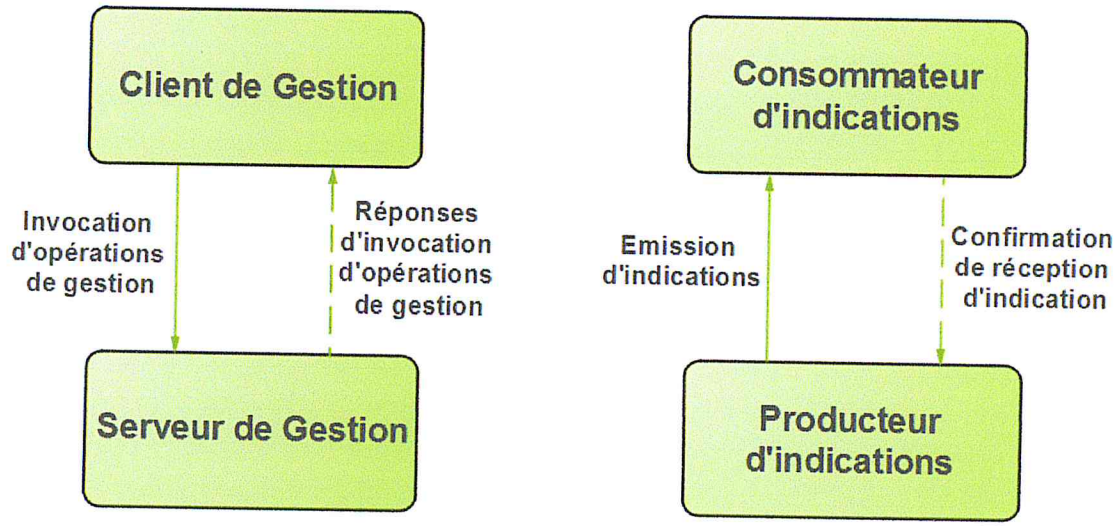



Figure 2.3. Les rôles des intervenants dans l'architecture WBEM [28].

1. **Client** : demande des informations ou exécute des opérations de gestion à une entité dans le rôle serveur.
2. **Serveur** : maintient les informations de gestion dans une base d'informations et exécute les opérations émises par les clients.
3. **Producteur** : implémente un service d'émission de notifications, émet des indications vers les consommateurs.
4. **Consommateur** : reçoit et traite les indications émises par les producteurs.

## 2.4 Le modèle d'information commun CIM

Le modèle d'information commun (**Common Information Model - CIM**) est un modèle d'information conceptuelle pour décrire les entités informatiques et métiers dans les environnements Internet, d'entreprise et de fournisseur de services [43]. Le modèle CIM comprend [25] :

- **La spécification CIM** : définit les détails pour l'intégration avec d'autres modèles de gestion.
- **Le schéma CIM** : fournit les descriptions réelles du modèle. Il capture des notions qui sont applicables à tous les domaines communs de gestion, indépendamment des implémentations.

## 2.4.1 La spécification CIM

La spécification CIM décrit le **méta-modèle** orienté objet basé sur UML (Unified Modeling Language) qui spécifie les composants du modèle, chacun des **éléments du méta-modèle** et les **règles** de chaque élément. La spécification définit également un langage de syntaxe CIM basé sur le langage de définition d'interface (IDL) appelé **MOF (Managed Object Format)** pour la traduction textuelle du modèle ainsi qu'un **mécanisme de nommage CIM** qui garantit l'unicité des noms des composants.

### 2.4.1.1 Le méta-modèle

Le méta-modèle est une définition formelle du modèle. Il définit les termes utilisés pour exprimer le modèle et son utilisation et sa sémantique [28]. Il définit les briques de base pour la construction des spécifications [28]. Les éléments du méta-modèle sont décrits ci-dessous, et la figure 2.4 illustre la relation entre eux [30] :

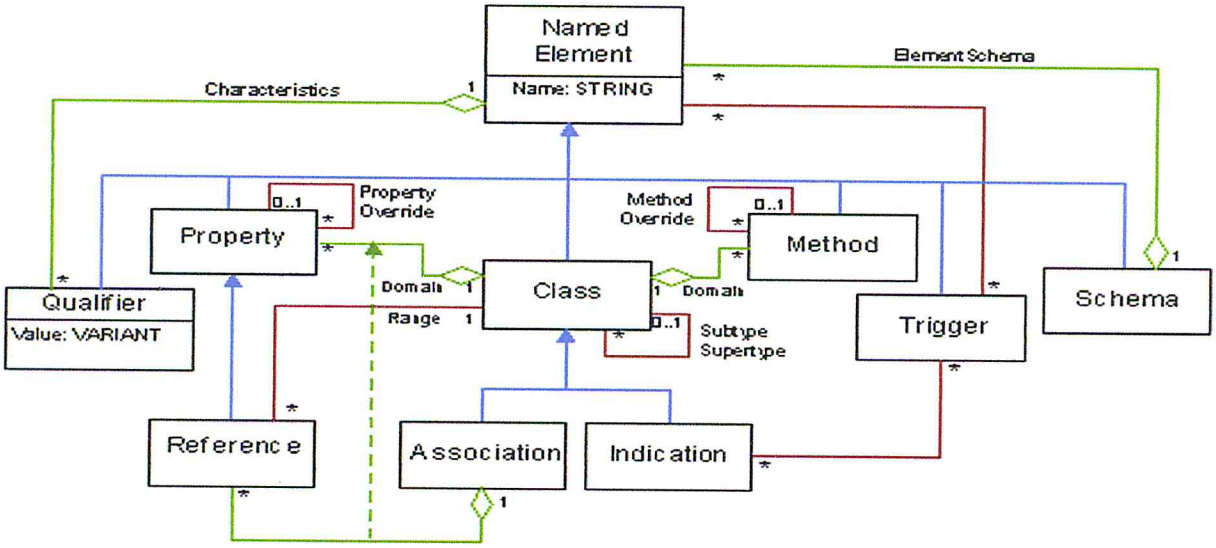



Figure 2.4. Méta-modèle CIM [25].

- **Le schéma** : Un schéma est un groupe de classes, d'instances et de qualificateurs appartenant à un seul propriétaire. Les schémas sont utilisés pour l'administration et la dénomination des classes.
- **La classe** : Une classe est un modèle, ou prototype, qui définit les propriétés et les méthodes communes à un type particulier d'objet.

- **La propriété** : Une propriété est une valeur utilisée pour désigner une caractéristique d'une classe. Elle possède un nom, un type de données, une valeur et une valeur par défaut facultative.
- **La méthode** : C'est une opération qui peut être invoquée. Elle décrit le comportement de la classe.
- **Le qualificateur** : C'est une métadonnée qui fournit des informations supplémentaires sur les autres éléments du méta-modèle.
- **La référence :** C'est un type de données pour des propriétés spéciales indiquant qu'il s'agit d'un pointeur vers l'instance d'une autre classe. Une référence définit le rôle que joue chaque objet dans une association.
- **L'association** : C'est un type de classe qui contient deux références ou plus. Les associations représentent des relations et des dépendances entre deux ou plusieurs classes.
- **L'indication** : Il s'agit d'une représentation active de l'occurrence d'un événement.
- **Le déclencheur** : Une opération invoquée lorsqu'un changement d'état survient au niveau de la MIB. Sa définition est donnée par un qualificateur pour chacun des composants.

### 2.4.1.2 Le schéma de nommage et l'architecture de la MIB

La base d'informations de gestion (CIMOR) utilise le concept d'espace de nommage pour organiser les informations de gestion. L'espace de nommage regroupe des objets (classes, instances, relations) où chaque composant est identifié par un nom unique qui le distingue des autres composants. Un espace de nommage peut contenir des composants de spécification des modèles d'informations ainsi que d'autres espaces de nommage en plus des objets gérés. L'arbre résultant de cette contenance définit la hiérarchie de la base d'informations de gestion [34].

La figure 2.5 ci-dessous illustre l'architecture de la MIB.

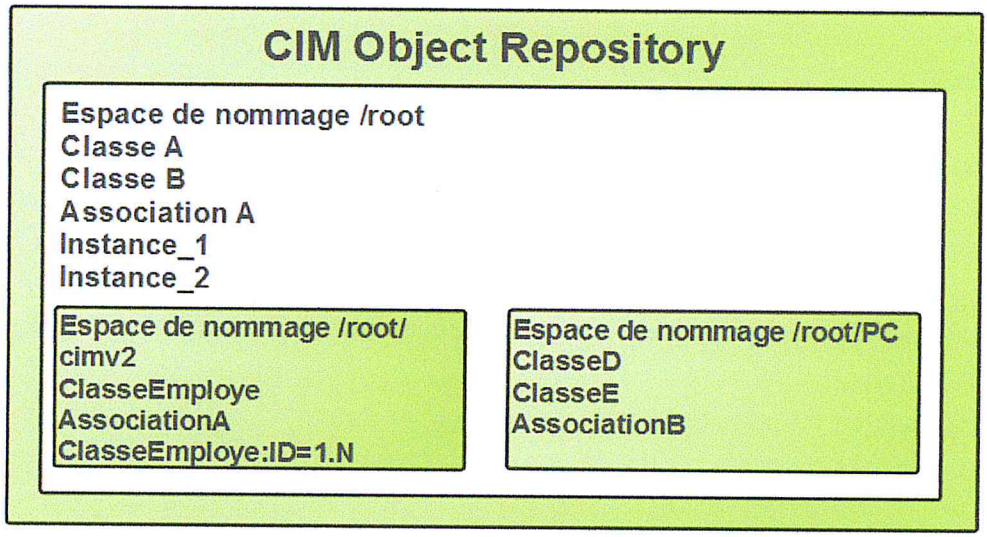



Figure 2.5. Architecture de la MIB dans un serveur WBEM [28].

Pour garantir l'identification unique des objets gérés contenus dans la MIB, un mécanisme de nommage a été défini. Ce dernier consiste en :

- Une **classe** ou un **composant de spécification** est caractérisé par un label dans la spécification qui est précédé par un chemin dans l'espace de nommage.
- Une **instance** est définie par le chemin dans l'espace de nommage suivi du nom de la classe de cette instance plus les valeurs des attributs formant la clé de l'instance.

### 2.4.1.3 Le langage de spécification

Le langage utilisé pour spécifier les objets gérés s'appelle **Managed Object Format (MOF)**. MOF est un langage de spécification orienté objet dérivé du langage IDL (Interface Définition Language). Il est intégré au modèle CIM afin de donner une ***représentation textuelle*** des différents composants d'un modèle de l'information ou schéma. Il fournit une syntaxe et une sémantique pour chacun des composants d'un modèle CIM. Le MOF fournit les moyens d'écrire des définitions d'interface des types de ressources gérés, y compris leurs propriétés, leurs comportements et leurs relations avec d'autres objets [36].

## 2.4.2 Le schéma CIM

Le schéma CIM définit également un ensemble de schémas de référence dont on distingue le schéma de base (ou Core Schema) et le schéma commun (ou Common Schema). Ces schémas offrent une structure de référence qui peut être directement réutilisée ou bien étendus pour la spécification de nouveaux systèmes.

### 2.4.2.1 Schéma de base

Le schéma de base définit un ensemble générique de classes et d'associations commun aux différents domaines de gestion. Il représente un point de départ pour déterminer comment étendre le schéma commun. Les classes abstraites du schéma de base représentent à proprement parler les racines du modèle dont hérite toute autre classe.

### 2.4.2.2 Schéma commun

Le schéma commun définit un ensemble de classes et d'associations spécifiques a des domaines de gestion. Il hérite du schéma de base et se spécialise en fonction du domaine.

Ce schéma se décompose en 12 sous-schéma associés aux domaines de gestion, ces modèles sont :

Le modèle **Application**, le modèle **Physique**, le modèle **Système**, le modèle **Évènement**, le modèle **Interopérabilité**, le modèle **Sécurité de l'utilisateur**, le modèle **Politique**, le modèle **Métrique**, le modèle **Réseau**, le modèle **Base de données**, le modèle **Périphérique** et le modèle **Soutien**. Le modèle qui nous intéresse dans le cadre de notre projet est le **modèle de politique**.

***Le modèle « Politique »*** **:** ce modèle fournit un cadre commun pour décrire toute la configuration système qui contrôle le comportement d'un système spécifié. Il permet aux développeurs d'applications, aux administrateurs de la politique, et aux administrateurs de réseau de représenter et de gérer les politiques à travers une variété de domaines techniques qui incluent la sécurité, le réseautage et l'administration du système.

### 2.4.2.3 Schéma d'extension

Ce type de schéma étend le modèle commun pour des technologies spécifiques. Ces schémas sont propres à des environnements tels que des systèmes d'exploitation. On s'attend à ce que les modèles communs évoluent en raison de la promotion des objets et des propriétés définis dans les schémas d'extension.

En utilisant les extensions, les développeurs peuvent développer à partir des classes de modèles de base et des associations, leurs propres modèles spécifiques pour couvrir des zones de gestion qui leurs sont propres. L'un des nombreux objectifs de CIM est qu'il s'agit d'un langage de modélisation beaucoup plus large et plus propre pour saisir les caractéristiques de gestion. Il permet aussi aux développeurs un contrôle beaucoup plus important entre le fournisseur et le consommateur d'informations (clients / agents) et entre les fournisseurs (systèmes).

### 2.5.1.1 La représentation XML

Le langage de balisage extensible (XML) est un sous-ensemble simplifié de SGML qui offre des fonctionnalités de modélisation de données puissantes et extensibles. Un document XML est une collection de données représentée en XML. Un schéma XML est une grammaire qui décrit la structure d'un document XML [45]. Le but de xmlCIM est de créer une grammaire XML qui peut être définie par un DTD (Document Type Definition) et peut être utilisée à la fois pour représenter les déclarations CIM (Classes, Instances et Qualificateurs) et les messages à utiliser par les protocoles CIM [25].

### 2.5.1.2 L'encapsulation HTTP

Le standard HTTP permet l'encapsulation des données CIM encodées en XML dans une unité de données HTTP. Ceci est schématisé dans la figure 2.7 :

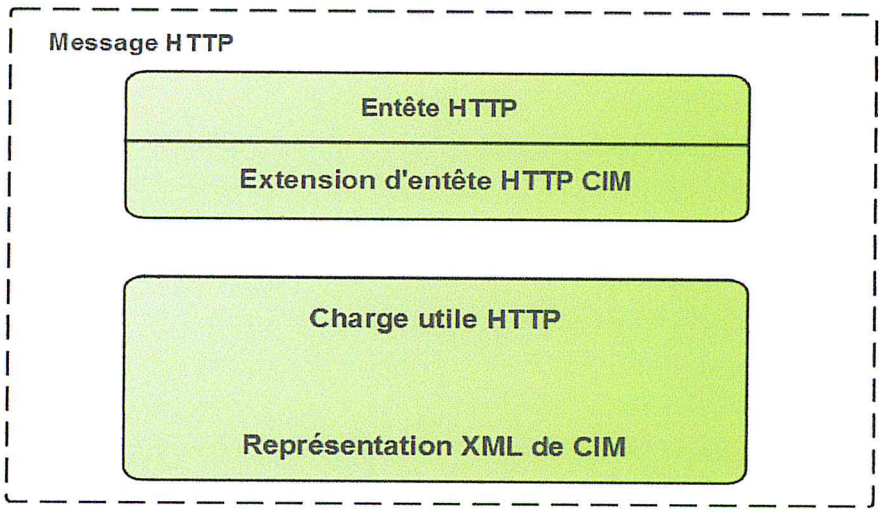



Figure 2.7. Format des messages HTTP dans WBEM [28].

## 2.5.2 Les opérations CIM

Les opérations CIM définissent un ensemble d'opérations qu'un client WBEM implémente pour fonctionner de manière ouverte et normalisée. Ces opérations peuvent être des opérations individuelles ou multiples [25].

Il existe deux types d'opérations CIM qui sont [28] :

1. **Les opérations Intrinsèques** : Les méthodes intrinsèques sont caractérisées par le fait qu'elles sont faites liées à un espace de noms CIM ; Elles sont classifiées en sept catégories :

Les opérations de **lecture** de base (Basic Read), Les opérations d'**écriture** de base (Basic

L'extension d'un schéma peut signifier plusieurs choses. Cela peut signifier :

- Ajout d'une propriété à une classe ou sous-classe existante.
- Ajout d'une nouvelle classe ou d'un ensemble de classes.
- Création de son propre espace de noms et de son schéma.

## 2.5 Le modèle de communication

Le DMTF a défini un protocole de communication pour que les différentes entités de gestion puissent s'échanger des informations. Il s'agit de deux standards du web et d'un ensemble d'opérations qui sont :

- **XML (eXtensible Markup Langage)** : pour l'encodage de l'information.
- **HTTP (Hyper-Text Transport Protocol)** : pour le transport de l'information.
- **Opérations CIM** : pour permettre l'accès et la manipulation des données CIM.

### 2.5.1 Le protocole de communication XML/ HTTP

La figure 2.6 illustre la communication de l'information entre deux entités WBEM, en utilisant le couplage XML/ HTTP qui permettent l'encodage et le transport de l'information respectivement.

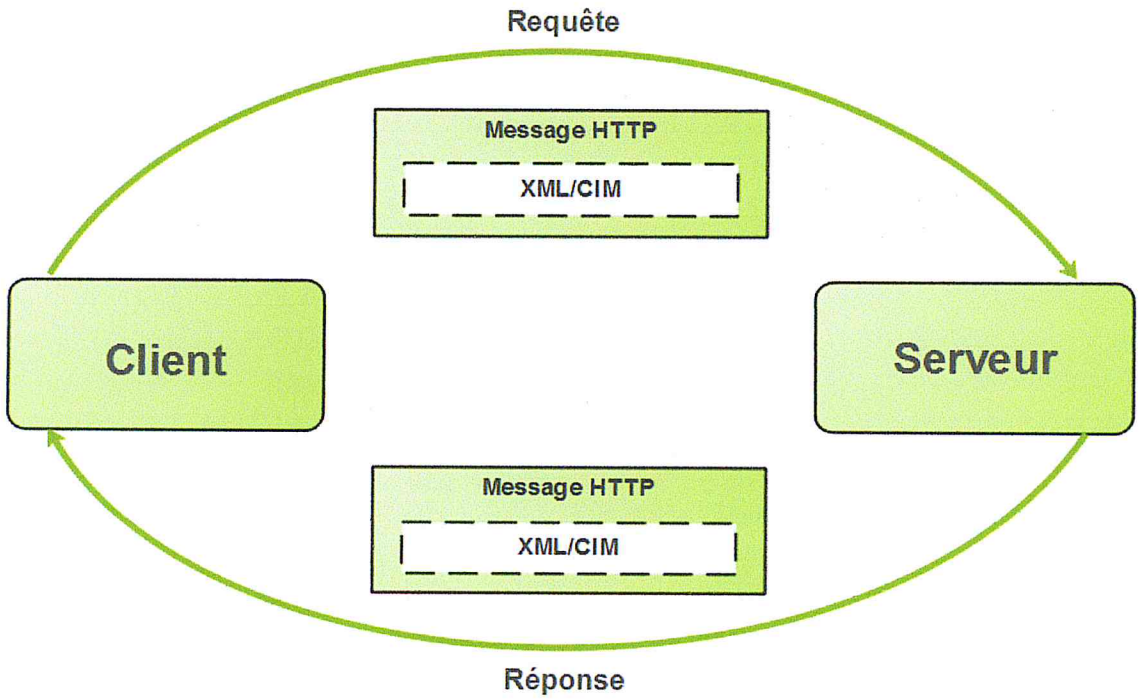



Figure 2.6. La communication de l'information dans WBEM [28].

Write) Les opérations de **manipulation de schémas** (Schema Manipulation), Les opérations de **manipulation d'instances** (Instance Manipulation), Les opérations de **parcours d'associations** (Association Traversa), Les opérations d'**exécution de requêtes** (Query Execution), et les opérations de **manipulation de qualificateurs** (Qualifiers Declaration).

**2. Les opérations Extrinsèques** : définies comme des méthodes sur une classe CIM dans certains modèles. Les méthodes extrinsèques sont invoquées sur une classe CIM (si statique) ou sur une instance.

## 2.6 Mécanismes de sécurité WBEM

WBEM emploie plusieurs mécanismes pour assurer un accès sécurisé à ses données, notamment [32] :

- **Authentification** : La procédure consistant à spécifier l'identité utilisateur d'un client sur le serveur WBEM, puis à démontrer que ce client est vraiment cet utilisateur en spécifiant les informations d'identification de l'utilisateur.
- **Hypothèse de rôle** : Le processus consistant à supposer qu'une identité de rôle de contrôle d'accès basé sur le rôle (RBAC) doit être utilisée par le serveur WBEM lorsqu'il vérifie l'autorisation.
- **Messages sécurisés** : La procédure d'ajout d'une signature numérique à chaque message de demande client. Cette signature permet au serveur WBEM de vérifier l'origine du message et de déterminer si ce message a été modifié lors de sa livraison au serveur WBEM.
- **Autorisation** : La procédure consistant à vérifier qu'un utilisateur authentifié ou une identité de rôle possède l'autorisation d'accès aux données WBEM gérées par chaque appel de méthode WBEM.
- **Audit** : La procédure d'écriture d'un enregistrement d'audit d'une opération spécifique qui a été effectuée par le serveur WBEM. Ces enregistrements suivent les modifications apportées par un utilisateur authentifié aux données de gestion du système serveur WBEM.
- **Enregistrement** : Écriture d'événements liés à la sécurité dans le journal WBEM.

## 2.7 Gestion des politiques dans WBEM

Le DMTF propose plusieurs modèles et profils qui traitent le domaine de la gestion des politiques ainsi qu'un langage de spécification de politiques qui s'appelle ***CIM Simplified Policy Language (CIM_SPL)*** [33]. Nous allons décrire dans un premier temps le langage de spécification de politiques ***CIM_SPL***. Ensuite, nous présentons le ***CIM Policy Model*** et le ***CIM Policy Profile*** qui sont dédiés à la gestion des politiques ainsi que le profil spécifique ***CIM Integrated Access Control Policy Management model*** qui est dédié aux politiques de contrôle d'accès intégré.

### 2.7.1 Le langage de spécification de politiques CIM-SPL

***CIM-Simplified Policy Language*** est un langage défini par le DMTF qui sert à spécifier des règles de politiques du type ***« si condition alors actions »*** pour gérer les ressources informatiques à l'aide de constructions définies par le modèle sous-jacent du modèle d'information CIM. La motivation derrière la conception d'un tel langage est la fourniture d'un langage compatible aux modèles de politiques CIM. Ce langage est basé sur le méta-modèle CIM déjà abordé et le CIM Policy Model qui sera abordé dans la section suivante.

### 2.7.2 Le modèle de politique CIM (CIM_Policy_Model)

Le modèle de politique comprend une hiérarchie de classes extensibles pour définir des objets de politique qui permettent aux développeurs d'applications, aux administrateurs de réseau et aux administrateurs de règles de représenter des politiques de différents types [39]. Son abstraction nous permet de construire des règles de politique de la forme :

**Si** <*condition (s)*> **Alors** <*action (s)*>.

Le terme <condition (s)> est une expression booléenne utilisée pour spécifier les critères de sélection de règle. Ces critères peuvent inclure des conditions temporelles (quand appliquer la règle), les conditions de détermination de la portée (ce qui s'applique à la règle) et les conditions liées à l'état (dans quelles circonstances l'action (s) de la règle est tenté).

### 2.7.3 Le profil de politique CIM (CIM_Policy_Profile)

Le profil de politique est un profil abstrait qui étend la capacité de gestion des profils de référencement en ajoutant la capacité de représenter les politiques qui s'appliquent aux éléments gérés et de définir des instructions CIM-SPL spécifiques représentant les politiques [38].

### 2.7.4 Le profil de gestion des politiques de contrôle d'accès intégré ( CIM_Integrated_Access_Control_Policy_Profile)

Le profil de gestion des politiques de contrôle d'accès intégré permet de composer et de distribuer des politiques communes de contrôle d'accès pour différents composants de contrôle d'accès dans les systèmes gérés [40].

## 2.8 Conclusion

Dans ce chapitre, nous avons présenté les différents concepts du standard de gestion homogène CIM/WBEM dédié à la gestion d'entités hétérogènes complexes (composants physiques, services, utilisateurs, etc.). D'abord, nous avons présenté l'architecture fonctionnelle de WBEM. Ensuite, nous avons abordé les composants de base du modèle de l'information commun (CIM). Après, nous avons étudié le modèle de communication utilisé dans ce standard. L'aspect sécurité ainsi que la gestion des politiques ont été discuté.

L'étude faite dans ce chapitre nous a permis de comprendre la puissance d'expression du standard WBEM/CIM qui est considéré comme l'un des standards les plus prometteurs du domaine. Nous avons choisi WBEM pour la réalisation de notre projet, à cause de sa capacité de gérer des entités complexes, vu que la gestion des politiques n'est pas un aspect supporté par n'importe quelle approche. De plus, l'interopérabilité offerte par WBEM répond à notre besoin de réalisation d'un système de gestion pouvant être intégré à d'autres systèmes.

# CHAPITRE 3 : LES POLITIQUES DE SÉCURITÉ SELINUX

## 3.1 Introduction

L'accès aux applications et aux ressources doit être contrôlé de manière granulaire et flexible afin de sécuriser les informations sensibles.

La majorité des systèmes d'information, sont construits sur la base du système d'exploitation qui met en œuvre les mécanismes DAC (pour *Discretionary Access Control* ou contrôle d'accès discrétionnaire). Cependant, il existe des problèmes de sécurité en ce qui concerne l'utilisation du DAC dans la couche OS, et qui sont notamment **l'utilisation de seulement deux grandes catégories d'utilisateurs qui sont les administrateurs et les autres** et **le principe très dangereux du super utilisateur root qui peut tout faire**. Les mécanismes MAC (pour Mandatory Access Control ou contrôle d'accès obligatoire) sont une solution privilégiée dans laquelle l'accès aux ressources système ne repose pas sur le propriétaire des ressources, mais dans les politiques de sécurité mises en œuvre dans l'OS.

Dans ce chapitre, SELinux est présenté comme une alternative aux systèmes d'exploitation standards qui mettent en œuvre le DAC. SELinux implémente un système MAC souple et fin dans le noyau Linux. Cela se fait par l'utilisation d'une architecture flexible qui est le Framework LSM. SELinux utilise un type de MAC appelé TE (pour Type Enforcement ou renforcement de type) et un mécanisme RBAC construit sur TE.

## 3.2 Linux Security Module

Le module de sécurité Linux (Linux Security Module - LSM) est le Framework de sécurité offert par le noyau Linux afin de relier des mécanismes de contrôle d'accès tiers au noyau du système en fournissant des crochets "***hooks***" à l'intérieur de ce dernier. Il permet une implémentation de ces mécanismes de sécurité afin de fournir des fonctions à appeler lorsqu'un crochet est déclenché. Ces fonctions vérifient la politique et d'autres informations avant de renvoyer un aller / non-retour. Comme le montre la figure 3.1, une demande d'utilisateur pour lire, écrire ou traiter n'importe quelle ressource doit passer par un ensemble d'étapes avant que l'autorisation ne soit accordée.

Dans un système Linux standard, la requête doit passer par la logique du noyau Linux traditionnelle et la gestion des erreurs suivie par les contrôles standards DAC. Avec les crochets LSM, l'accès aux ressources est accordé après que le crochet LSM soit appelé. Ces crochets sont définis juste après les contrôles DAC Linux et avant d'accéder à l'objet demandé.

LSM ne fournit pas de services de sécurité en lui-même, seuls des crochets et des structures pour supporter des modules tiers. Si aucun module tiers n'est chargé, le mécanisme de contrôle d'accès discrétionnaire est appliqué.

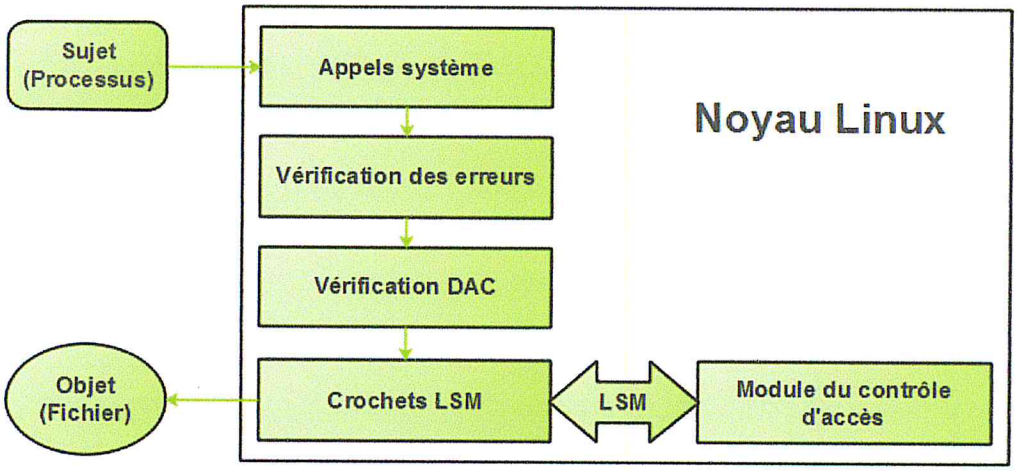



Figure 3.1. LSM Framework [9].

## 3.3 Architecture et fonctionnement de SELinux

Pour commencer avec SELinux, nous devons distinguer deux points clés : les **sujets** et les **objets**. Le sujet est une entité, initiant une action. En tant qu'action, nous considérons une sorte d'opération effectuée. Typiquement, les sujets effectuent certaines opérations sur des objets, ou même sur d'autres sujets. Les sujets sont les acteurs des systèmes informatiques. La première pensée est que les utilisateurs seraient les sujets, cependant, les processus sont les vrais acteurs [39].

La figure 3.2 montre un diagramme de haut niveau des composants principaux de SELinux qui gèrent l'application de la politique et comprend les éléments suivants :

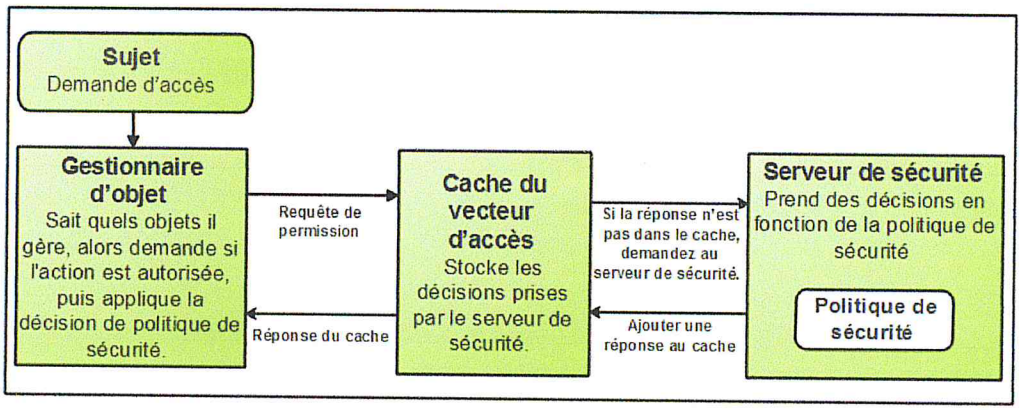



Figure 3.2. Composants principaux de base de SELinux [19].

1. **Un sujet** qui doit être présent pour provoquer une action sur un objet (par exemple, lire un fichier).
2. **Un gestionnaire d'objets** qui connaît les actions requises de la ressource particulière (tel qu'un fichier) et peut appliquer ces actions (c'est-à-dire lui permettre d'écrire sur un fichier si la politique l'autorise).
3. **Un serveur de sécurité** qui prend des décisions concernant les droits des sujets pour effectuer l'action demandée sur l'objet, en fonction des règles de politique de sécurité.
4. **Une politique de sécurité** qui décrit les règles à l'aide du langage de politique SELinux.
5. **Un cache de vecteur d'accès (AVC)** qui améliore les performances du système en mettant en cache les décisions du serveur de sécurité.

En bref, chaque action dans SELinux est identifiée par le sujet, l'objet et l'opération effectuée. Le serveur de sécurité vérifie si une requête / réponse donnée est dans le cache vectoriel d'accès, et si ce n'est pas le cas, il consulte les règles d'accès définies dans la logique de politique de sécurité. Enfin, il prend des décisions et enregistre une nouvelle entrée à AVC.

## 3.4 Modes de fonctionnement SELinux

SELinux a trois principaux modes de fonctionnement [19] :

1. **Exécution (Enforcing)** - SELinux applique la politique chargée.
2. **Permissive (Permissive)** - SELinux a chargé la politique, mais elle ne fait pas respecter les règles de la politique. Ceci est généralement utilisé pour le test car le journal d'audit contiendra les messages refusés par le contrôleur AVC.

3. **Désactivé (Disabled)** - L'infrastructure SELinux n'est pas activée, donc aucune politique SELinux ne peut être chargée.

## 3.5 SELinux, un contrôle d'accès obligatoire (MAC)

Le contrôle d'accès obligatoire (MAC) est un type de contrôle d'accès dans lequel le système d'exploitation est utilisé pour contraindre un utilisateur ou un processus (le sujet) à accéder ou à exécuter une opération sur un objet (tel qu'un fichier, un disque, une mémoire, etc.).

Chacun des sujets et objets possède un ensemble d'attributs de sécurité qui peuvent être interrogés par le système d'exploitation pour vérifier si l'opération demandée peut être exécutée ou non.

Pour SELinux :

- Les sujets sont des processus.
- Les objets sont des ressources système telles que des fichiers, des sockets, etc.
- L'ensemble des attributs de sécurité est appelé contexte de sécurité.
- Le serveur de sécurité dans le noyau Linux autorise l'accès (ou non) en utilisant la politique de sécurité qui décrit les règles qui doivent être appliquées. Notons que le sujet (et par conséquent l'utilisateur) ne peut pas décider de contourner les règles de politique appliquées par le système MAC lorsque SELinux est activé.

Comparons cela au système DAC standard de Linux, qui régit également la capacité des sujets à accéder aux objets, mais il permet aux utilisateurs de prendre des décisions stratégiques (comme donner des privilèges d'accès à d'autres utilisateurs).

## 3.6 Contexte de sécurité

SELinux utilise des contextes de sécurité pour identifier les sujets et les objets dans le système [21]. Un contexte de sécurité est représenté sous forme de chaîne de longueur variable, définissant l'**utilisateur** SELinux, le **rôle** SELinux, le **type** et un **niveau de sécurité MLS/MCS** facultatif comme suit : ***Utilisateur : rôle : type [: level]***

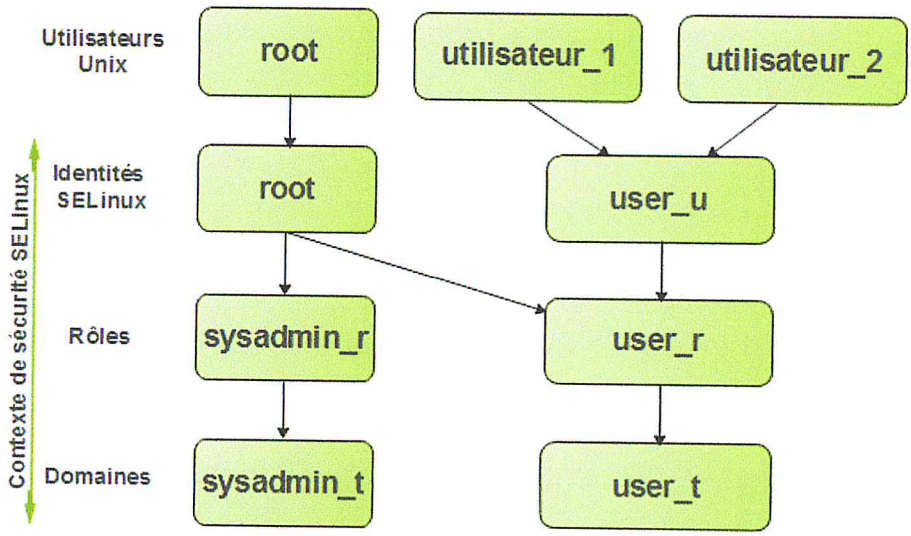



Figure 3.3. Contexte de sécurité dans SELinux [20].

Les **rôles** sont utilisés pour avoir accès à un ou plusieurs types. L'**utilisateur** représente l'identité de l'utilisateur SELinux et peut être associé à un ou plusieurs rôles. Un champ facultatif définissant un **niveau de sensibilité** et une catégorie ne peut être présent que si la politique prend en charge Multi-Level Security (MLS) ou Multi-Category Security (MCS).

## 3.7 Mécanismes de contrôles d'accès SELinux

SELinux implémente un système MAC appelé **renforcement de type** *(Type Enforcement - TE)* avec une forme de **contrôle d'accès basé sur les rôles** (*Role Based Access Control* - RBAC) et le support de la **sécurité multi-niveaux** *(Multi-Level-Security - MLS)* ainsi que la **sécurité multi-catégorie** *(Multi-Category Security - MCS)*. Le Type Enforcement implémenté ne permet aucun accès par défaut. Pour cette raison, chaque accès doit être spécifié avec une règle TE, fournissant ainsi un contrôle d'accès à grain fin pour protéger les processus et les objets. SELinux met en œuvre une forme de RBAC construit sur le TE qui aide à simplifier la complexité de la gestion des utilisateurs dans les systèmes à grande échelle. Le MLS et le MCS sont fournis aussi, mais sont optionnels.

### 3.7.1 Mécanisme de renforcement de type (Type Enfoncement)

Le renforcement de type est un mécanisme de base de SELinux, basé sur les types et les domaines. A chaque objet est associé un type et à chaque processus un domaine correspondant (qui est juste un type particulier). L'accès à un objet ayant un type particulier serait autorisé si le sujet est étiqueté avec un domaine particulier.

Les types et les domaines sont des éléments de base des règles de langage de politique SELinux. Après avoir défini ces règles, chaque sujet ne peut effectuer que des opérations, explicitement spécifiées dans la portée du domaine associé. Dans le renforcement de type, comme dans tous les mécanismes MAC, le concept clé est le principe des moindres privilèges. Seules les actions autorisées peuvent être exécutées, toutes les autres sont refusées.

## Les règles de type AV (Access Vector)

Les règles de type vecteur d'accès abrégées AV (pour Access Vector) déterminent pour les processus ce qui est autorisé à exécuter. Le langage de politique SELinux supporte actuellement quatre types de règles AV [20] :

**Allow** : est l'instruction la plus courante dans le langage de politique, qui vérifie si les opérations entre le type source et le type cible sont autorisées. La règle **allow** est toujours requise.

**Dontaudit** : arrête l'audit des refus. Cette règle est utile à appliquer quand on sait que l'évènement se passe toujours, mais sans vrais problèmes.

**Auditallow** : vérifie uniquement l'événement.

**Neverallow** : est utilisée lorsqu'une règle **allow** ne doit pas être générée pour l'opération, même si elle a déjà été autorisée. Ce type de règles n'est utile que pour l'audit.

La règle « allow » de la famille « Access Vector Rule » est la seule règle dans SELinux qui donne

l'autorisation d'accès à une ressource donnée. Voici la syntaxe employée :

***allow source_type target_type : object_class perm ;***

Le « source_type » est le domaine du processus. Le « target_type » est le type de l'objet accédé par le processus. L'« object_class » est la classe de l'objet auquel nous voulons accéder. Le champ « perm » est la liste de permission(s) que le domaine est autorisé à effectuer sur la classe typée avec le champ « target_type ». La figure 3.4 illustre une règle allow de type AV.

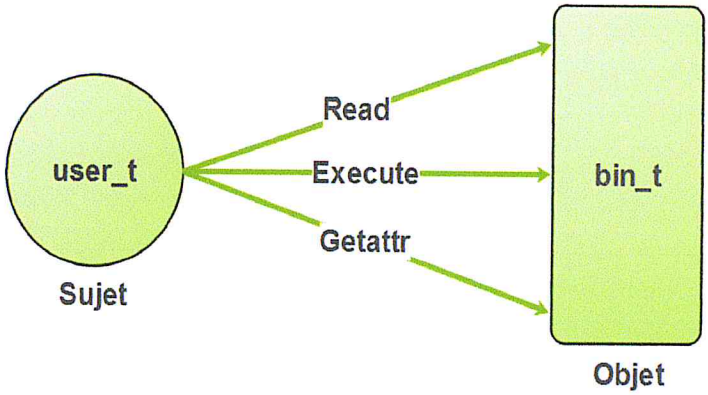



Figure 3.4. Représentation d'une règle allow [20].

## Les règles Type

Les règles de renforcement de type (TE en abrégé pour Type Enfoncement) spécifient les types par défaut pour les objets créés ou renommés au moment de l'exécution. Il existe deux règles de type définies dans le langage de politique SELinux [20] :

- **Type_transition**

Spécifie le comportement d'étiquetage du type par défaut pour la transition de domaine et la création d'objets. Actuellement, il existe deux formes de la règle type_transition :

1. **Transitions de domaine par défaut** : Les transitions de domaine modifient le type de processus lors de l'exécution d'un fichier.

La transition de domaine est fondamentale pour appliquer le principe du moindre privilège et restreindre les possibilités d'accès d'une application. Voici comment la transition est orchestrée :

***type_transition source_type target_type : process default_type ;***

Le « source_type » est le domaine d'un utilisateur ou d'un processus. Le « target_type » est le type d'un fichier exécutable. « Process » est la classe objets processus. « Default_type » est le domaine auquel le nouveau processus lancé transit lorsque le domaine « source_type » exécute un fichier binaire avec un type « target_type ». Pour qu'une transition de domaine ait lieu, il faut trois règles « allow » pour l'autoriser et une règle « type transition ».

2. **Transitions d'objets par défaut** : Les règles de transition d'objet spécifient un type par défaut pour les objets nouvellement créés.

Les deux formes de la règle type_transition permettent de rendre la sécurité améliorée de SELinux transparente pour l'utilisateur Linux. Les règles type_transition n'autorisent pas l'accès, elles ne fournissent qu'un nouveau type d'étiquetage par défaut.

- **Type_change** : spécifie les types par défaut pour la ré-écriture effectuée par les applications SELinux-aware (c'est à dire les applications conscientes pour lesquelles SELinux n'est pas transparent).

Nous appelons ces règles "règles de type" car elles sont similaires aux règles AV, sauf que le dernier champ de la règle est un nom de type plutôt qu'une liste de permissions.

### Transition de domaine

Une transition de domaine est l'endroit où un processus dans un domaine démarre un nouveau processus dans un autre domaine sous un contexte de sécurité différent [19]. La transition de domaine est fondamentale pour appliquer le principe du moindre privilège et restreindre les possibilités d'accès d'une application [3].

Bien évidemment la transition d'un domaine à un autre est strictement réglementée pour éviter qu'un processus change de domaine selon son bon vouloir. Pour celà, avant toute transition de domaine peut avoir lieu, la politique doit spécifier que [19] :

1. Le domaine source a l'autorisation de se transmettre au domaine cible.
2. Le fichier binaire de l'application doit être exécutable dans le domaine source.
3. Le fichier binaire de l'application nécessite un point d'entrée dans le domaine cible.

Ce qui suit est un exemple d'une transition de domaine :

***type_transition unconfined_t secure_services_exec_t : process ext_gateway_t;***

Cette transition de domaine indique que lorsqu'un processus exécuté dans le domaine ***unconfined_t*** (le domaine source) exécute un fichier étiqueté ***secure_services_exec_t***, le processus doit être changé en ***ext_gateway_t*** (le domaine cible) s'il est autorisé par la politique.

Cependant, comme indiqué ci-dessus, pour pouvoir passer au domaine ext_gateway_t, les autorisations minimales suivantes doivent être accordées dans la politique à l'aide de règles allow, où (notez que les numéros des points ci-dessous correspondent aux chiffres affichés à la figure 3.5) :

1. Le domaine a besoin d'autorisation pour passer au domaine ext_gateway_t (cible) :

***allow unconfined_t ext_gateway_t : process transition;***

2. Le fichier exécutable doit être exécutable dans le domaine unconfined_t (source) et nécessite donc que le fichier soit lisible :

***allow unconfined_t secure_services_exec_t: file {execute read getattr};***

3. Le fichier exécutable nécessite un point d'entrée dans le domaine ext_gateway_t (cible) :

***allow ext_gateway_t secure_services_exec_t : file entrypoint;***

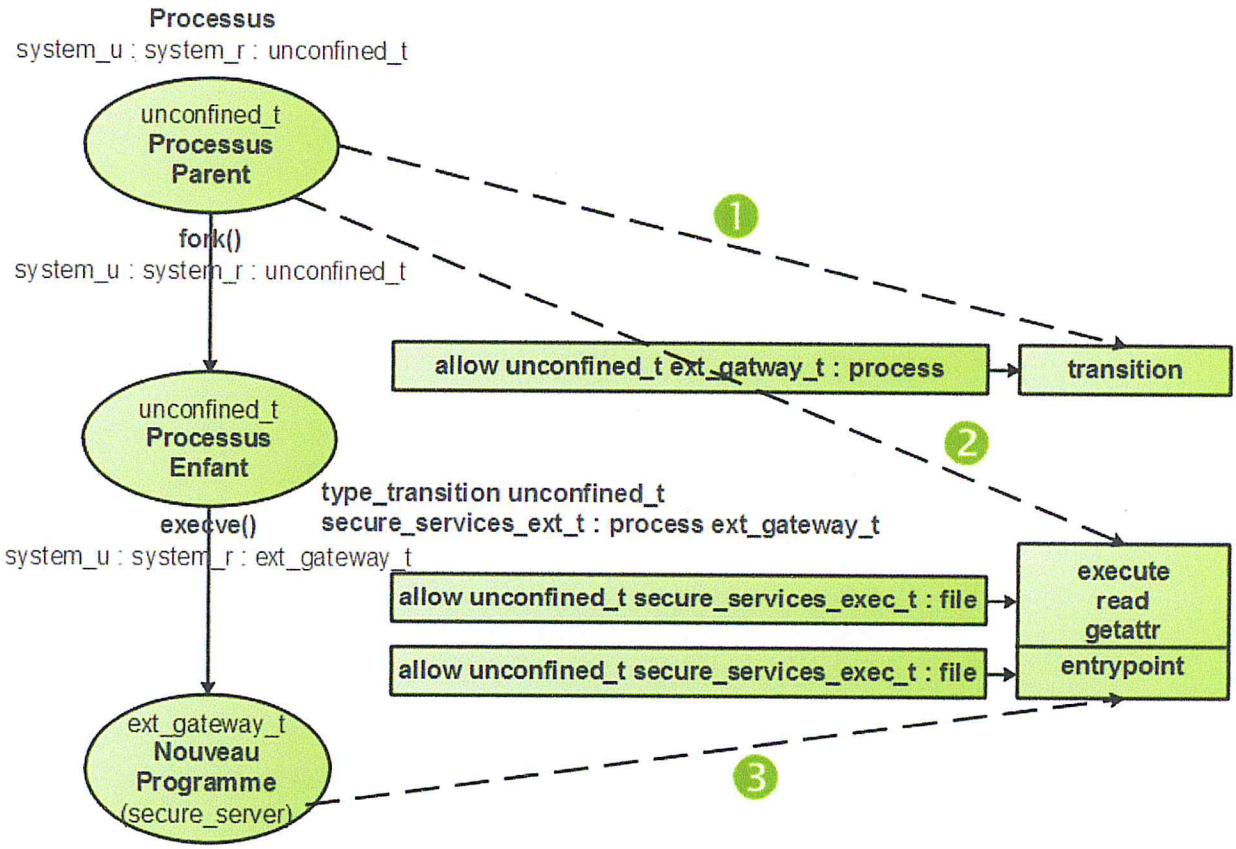



Figure 3.5. **Transition du domaine** - Lorsque le secure-server est exécuté dans le domaine unconfined_t et ensuite transité au domaine ext_gateway_t [19].

## 3.7.2 Mécanisme du contrôle d'accès basé sur les rôles

Dans SELinux, les fonctionnalités RBAC s'appuient sur les fonctionnalités TE et les accompagnent. Les privilèges sont accordés à un utilisateur indirectement en associant des types de domaine à un ou plusieurs rôles. Les énoncés de politique de RBAC n'accordent pas l'accès. Au lieu de cela, RBAC contraint la stratégie TE en contrôlant les associations de types de domaine, de rôles et d'utilisateurs dans un contexte de sécurité. De cette façon, les transitions de

domaine disponibles pour le type de domaine d'un utilisateur sont restreintes en fonction du rôle de l'utilisateur, restreignant finalement les privilèges totaux de l'utilisateur.

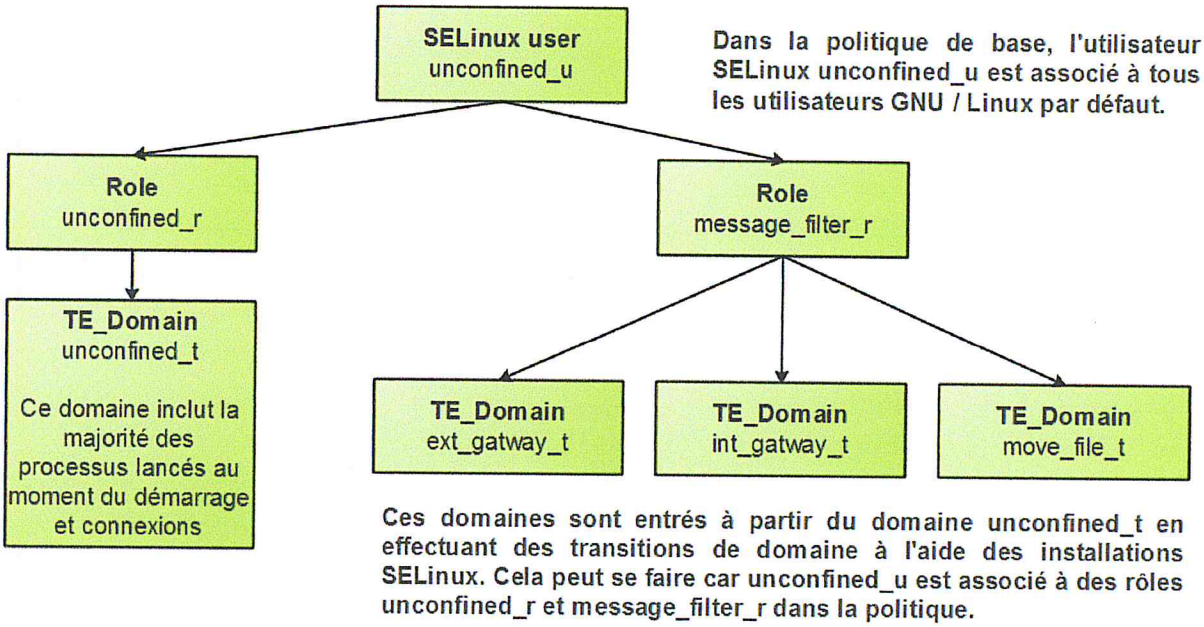



Figure 3.6. **Contrôle d'accès basé sur les rôles** - Affichage de la façon dont SELinux contrôle l'accès par l'intermédiaire de l'utilisateur, du rôle et du type de domaine [19].

### 3.7.3 Mécanisme du Multi Level Security (MLS)

Le renforcement de type est de loin le mécanisme de contrôle d'accès obligatoire (MAC) le plus important qu'introduit SELinux. Cependant, dans certaines situations, principalement pour un sous-ensemble d'applications gouvernementales classifiées, le MAC de sécurité multiniveaux traditionnel (MLS) couplé au renforcement de type est précieux. En reconnaissance de ces situations, SELinux a toujours eu une certaine forme de capacité MLS. Les fonctionnalités MLS sont facultatives et généralement moins importantes des deux mécanismes MAC dans SELinux. Le niveau de sécurité utilisé par les systèmes MLS est une combinaison d'une sensibilité hiérarchique et d'un ensemble de catégories non hiérarchiques. Ces sensibilités et ces catégories sont utilisées pour refléter la confidentialité réelle de l'information ou les autorisations de l'utilisateur.

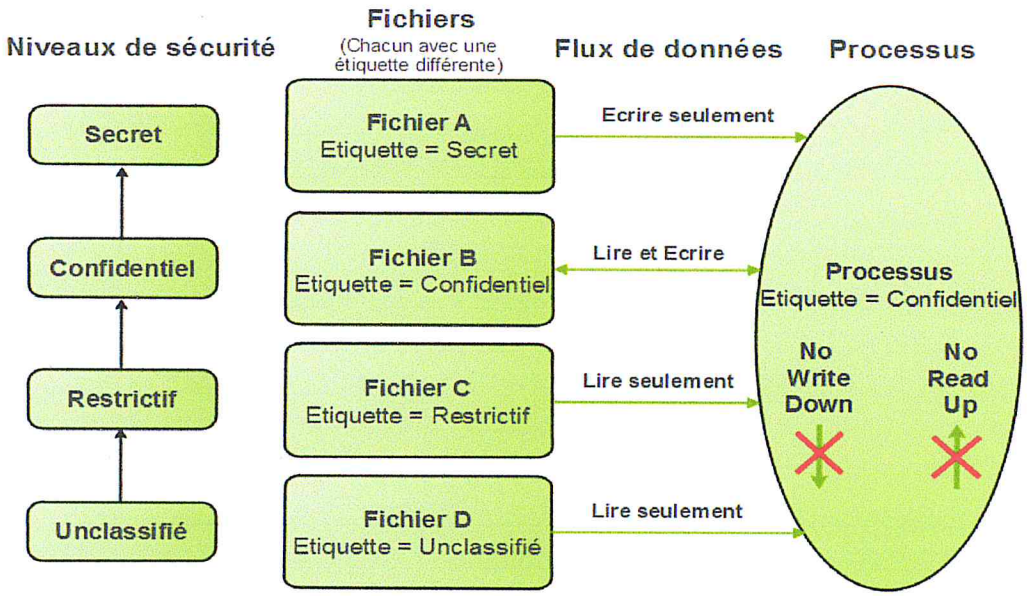



Figure 3.7. **Niveaux de sécurité et flux de données** - Cela montre comment le processus ne peut que « Lire en bas » et « Écrire en haut » dans un système MLS activé.

Les services MLS / MCS sont maintenant plus généralement utilisés pour maintenir la séparation des applications, par exemple :

• Les machines virtuelles utilisent des catégories MCS pour permettre à chaque machine virtuelle de s'exécuter dans son propre domaine afin d'isoler les machines virtuelles les unes des autres.

• Les appareils Android utilisent des catégories MCS générées dynamiquement afin qu'une application exécutée au nom d'un utilisateur ne puisse pas lire ou écrire des fichiers créés par la même application exécutée pour le compte d'un autre utilisateur.

## Extensions MLS aux contextes de sécurité

Pour les systèmes MLS SELinux, le contexte de sécurité est étendu pour inclure deux niveaux de sécurité : un niveau de sécurité faible et un niveau de sécurité élevé. En général, le niveau bas reflète le niveau de sécurité actuel d'un processus ou la sensibilité des données contenues dans un objet. Le niveau élevé reflète le niveau de validation de l'identifiant de l'utilisateur dans le contexte ou la plage maximale de données autorisée pour certains objets dits à plusieurs niveaux. Lorsque MLS est activé, le contexte de sécurité étendu a le format suivant :

***User : role : type : sensitivity [:category,...] [-sensitivity [:category,...]]***

## 3.8 Politiques SELinux

SELinux est configuré via des politiques. La politique SELinux est l'ensemble des règles qui guident le mécanisme de sécurité SELinux. Elle définit les types d'objets de fichiers et de domaines pour les processus. Elle utilise des rôles pour limiter les domaines qui peuvent être entrés et a des identités d'utilisateurs pour spécifier les rôles qui peuvent être atteints. Essentiellement, les types et les domaines sont équivalents, la différence étant que les types s'appliquent aux objets alors que les domaines s'appliquent aux processus.

## 3.9 Journaux d'audit

L'écriture et la configuration de la politique SELinux seraient très difficiles sans l'aide de la journalisation. Ils aident l'administrateur système à résoudre différents problèmes et à analyser les causes de refus d'accès. Deux types principaux d'événements d'audit peuvent être classés [19] :

- **SELinux-aware application events** - Erreurs système, changement d'états booléens, paramétrage du mode d'application / permissivité, ré-étiquetage, etc. Tous ces événements sont enregistrés par les services du noyau SELinux et les applications SELinux-aware.
- **AVC (Access Vector Cache) audit events** - refus d'accès, généré par le système AVC.

Les deux types de messages d'événement peuvent être stockés en trois endroits (pour le système RedHat et ses dérivés comme Fedora, Centos etc.) [22] :

Le journal ***/ var / log / dmesg log*** où sont enregistrés les événements de démarrage du noyau SELinux. Le journal du système ***/ var / log / messages*** qui contient des messages générés par le service syslog et des messages AVC, générés avant le chargement du démon d'audit (auditd). Une fois le démon d'audit chargé, les événements d'audit sont stockés dans le journal ***/var/log/audit/audit.log***.

## 3.10 Les politiques conditionnelles (Booléens SELinux)

L'une des fonctionnalités les plus importantes de SELinux est le support des politiques conditionnelles [11]. Dans certains cas, SELinux peut être configuré sans connaissance de l'écriture de politique. Les fonctionnalités importantes de la politique SELinux peuvent être modifiées sans recharger la politique. Les booléens sont des composants SELinux qui permettent de le faire. Ils prennent de la valeur sur ou hors tension.

Les booléens définissent des politiques conditionnelles, par des instructions conditionnelles spéciales. Ces instructions supportent les constructions ***if / else*** qui spécifient quelles règles sont valides sous la condition.

Le but des booléens est de simplifier le travail des utilisateurs et des administrateurs système lors de la configuration des politiques.

## 3.11 Les modules SELinux

Les modules de politique SELinux sont actuellement des composantes de base de la politique de référence. La majorité des règles de renforcement de type sont mises en œuvre dans les modules. Ils contrôlent le comportement de différentes applications, services, périphériques et de nombreux autres éléments du système [22]. L'objectif principal des modules est d'aider les administrateurs système à déployer, modifier et mettre à jour les stratégies SELinux de manière sécurisée et convenable sans être obligé de recompiler toute la politique et la recharger dans le noyau [11].

## 3.12 Conclusion

Ce chapitre a présenté SELinux comme une approche réalisable pour protéger les ressources. La flexibilité des mécanismes dans SELinux permet de mettre en œuvre SELinux dans différentes organisations avec des exigences de sécurité différentes. SELinux met en œuvre un système MAC flexible et fin appelé TE et un type de RBAC construit sur TE. Ces deux mécanismes répondent à des exigences telles que la séparation du domaine et l'application du principe du moindre privilège. En outre, l'utilisation de RBAC contribue à simplifier les tâches de gestion des utilisateurs. Des mécanismes supplémentaires tels que des modules de politiques chargeables et des politiques conditionnelles sont également des caractéristiques souhaitables tout en gérant les politiques SELinux.

Les stratégies conditionnelles permettent d'apporter des modifications à la stratégie sur place et les modules de stratégie chargeables simplifient la création et la mise en œuvre des stratégies SELinux. En outre, SELinux fait partie des solutions ouvertes, qui sont nécessaires pour fournir une meilleure sécurité des systèmes d'information à l'avenir. Par conséquent, SELinux est une approche viable recommandée pour aider à la protection des ressources.

# CHAPITRE 4 : CONCEPTION DU SYSTÈME RÉALISÉ

## 4.1 Introduction

Afin de modéliser notre Framework, l'approche WBEM a été choisie pour satisfaire nos besoins en termes de standards de gestion (modélisation des règles, leur distribution dans le réseau, etc.). Pour le renforcement de ces politiques, le module de sécurité SELinux a été choisi comme premier système de contrôle d'accès obligatoire. C'est un système qui offre des mécanismes de sécurité très robustes et configurables (contrôle d'accès obligatoire basé sur le renforcement de type, utilisation d'un mécanisme de RBAC, support des politiques de type MLS et MCS, en plus de la possibilité de configurer de nouveaux types, permissions, etc.).

Dans ce chapitre, nous présentons notre démarche pour la conception et d'un gestionnaire de politique de sécurité obligatoire (MAC). Dans cette partie, nous abordons les points suivants :

- L'architecture de notre Framework dédié à l'assistance dans la gestion des politiques de sécurité obligatoire fondée sur le standard WBEM.
- La modélisation des politiques MAC génériques.
- La modélisation des politiques SELinux, ainsi que leur application dans un environnement Linux.

Notre modélisation repose sur le standard CIM/WBEM permettant la création de systèmes de gestion agnostiques. Bien que CIM/WBEM est aujourd'hui l'un des standards de gestion les plus prometteurs, l'implémentation de *Providers* permettant de gérer n'importe quel composant d'un système reste son plus grand défi. Notre travail va participer à l'enrichissement de WBEM, en implémentant un *Provider* dédié à SELinux.

## 4.2 L'architecture de notre système de gestion

L'architecture que nous avons adoptée pour le déploiement de notre système est basée sur l'architecture générique spécifiée par la standard PCIM [3] élaboré conjointement par l'IETF et le DMTF qui définit quatre composantes [24,26]:

- **Le point d'administration des politiques** : appelé **PAP (Policy Administration Point)** est l'interface utilisateur permettant la construction, le déploiement des politiques et la surveillance de l'état de l'environnement géré par les politiques. Dans notre système ; il correspond au client CIM/WBEM, il fait appel aux services du PDP afin de définir, distribuer et appliquer des politiques.

- **Le point de décision des politiques :** appelé **PDP** ***(Policy Decision Point)***, correspond au serveur CIMOM qui maintient l'application des politiques de contrôle d'accès applicables aux systèmes gérés. Concrètement, les règles sont définies par l'administrateur via un client CIM, et peuvent être activées et désactivées par la suite.
- **Le point de référentiel des politiques** : appelé **PRP (Policy Repository Point)** représente le référentiel contenant les règles des politiques, leurs conditions et actions et les données de politique connexes et permet de les lire. Il peut également être défini comme une abstraction de modèle représentant un conteneur logique défini par l'administration pour les éléments de politiques réutilisables. Dans notre cas c'est le serveur CIMOM qui constitue le point d'entrée au référentiel CIMOR contenant les politiques sous la forme d'instances CIM.
- **Le point de renforcement des politiques :** appelé **PEP (Policy Enforcement Point)** est une entité logique qui exécute des décisions de politique et / ou effectue un changement de configuration. Il applique les règles de contrôle d'accès définies par le PDP, dans notre système c'est SELinux augmenté par le Provider CIM/WBEM associé.

L'architecture de notre système de gestion est présentée dans la figure 4.1.

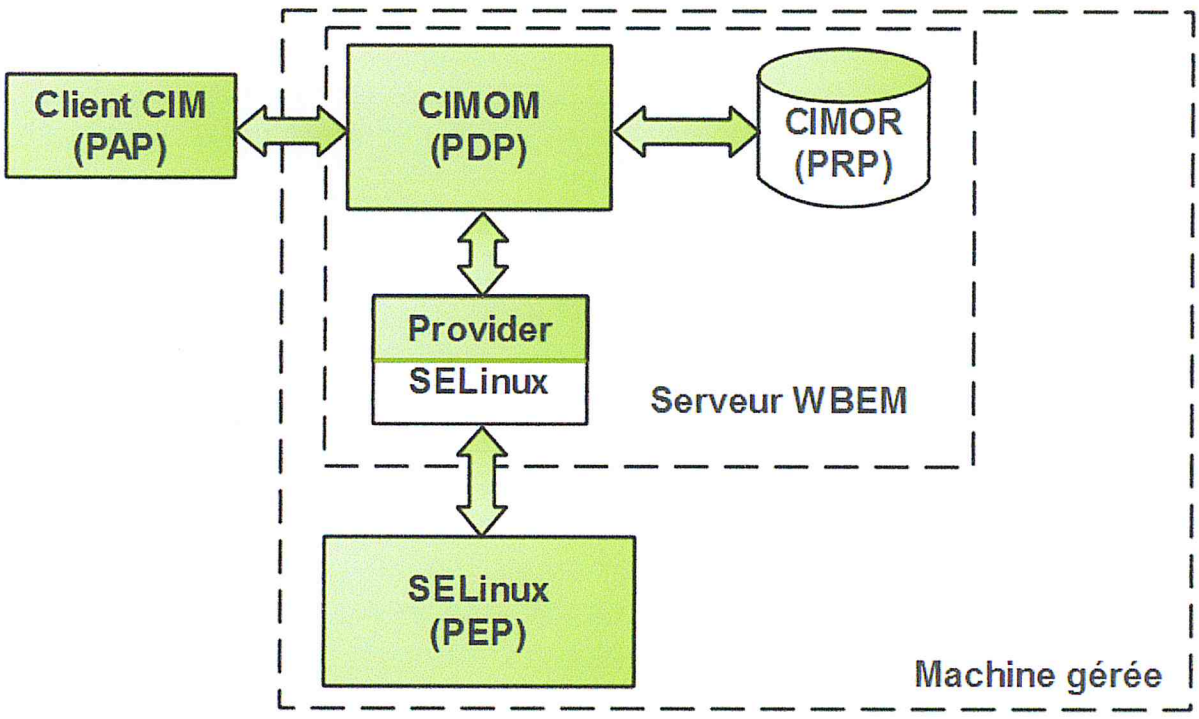



Figure 4.1. Architecture du système de gestion.

En résumé, les composants de l'architecture de notre système sont :

- **Le client WBEM** (le PAP) : Interface utilisateur permettant la définition et le suivie des politiques de contrôle d'accès obligatoire pouvant un ensemble de règles ou groupe de règles.
- **Le serveur WBEM** (le PDP) : contrôle les opérations de création, modification, suppression des politiques introduites par les utilisateurs (en format CIM) ou découvertes sur les systèmes gérés (en format natif) en utilisant le format CIM dans la base CIMOR.
- **Le Provider SELinux** (PEP CIM) : c'est le composant d'intégration de SELinux au serveur WBEM. Il s'occupe d'un côté, de la traduction CIM/SELinux des politiques en format CIM vers le format SELinux natif. Et d'un autre côté de la traduction inverse SELinux vers CIM permettant ainsi de superviser l'état de SELinux (le statut et le mode de SELinux, les règles appliquées, l'étiquetage des fichiers, etc.).
- **SELinux** (PEP Natif) : c'est l'un des systèmes de contrôle d'accès obligatoire sous Linux, choisi dans le cadre de ce projet comme cible des politiques CIM après qu'elles soient traduite en règles SELinux concrètes par le Provider.

Le Provider que nous développons possède deux interfaces :

- **Une interface CIM :** permet l'interaction avec le CIMOM, la lecture des politiques définies et stockées dans la base CIMOR et la présentation de l'état du système sous format CIM.
- **Une interface SELinux :** permet l'interaction avec SELinux, la transmission des règles concrètes à appliquer et la récupération de l'état de SELinux ainsi que celui des ressources contrôlées.

Le *Provider* SELinux est en réalité un **ensemble de *modules*** dont chacun maintient une classe de notre modèle CIM. C'est à dire, chaque module assure l'exécution des méthodes d'une classe donnée ainsi que la gestion de ses instances (modifier et retourner les valeurs des attributs).

## 4.3 Modélisation fonctionnelle

La modélisation fonctionnelle nous permettra de définir une abstraction du système d'un point de vue fonctionnel en identifiant les acteurs, leurs différentes façons d'utiliser le système, d'établir précisément les frontières du système et de déterminer les cas d'utilisation et leurs descriptions.

Dans cette section, nous allons présenter les différents scénarios d'utilisation de notre Provider en utilisant des diagrammes de cas d'utilisations.

### 4.3.1 Digramme de cas d'utilisation global

Les cas d'utilisations de notre Provider sont :

| Cas d'utilisation | Description textuelle |
|---|---|
| **Gérer la configuration du service SELinux** | Le gestionnaire a la possibilité de consulter, et modifier l'état de configuration du service SELinux (Statut, mode courant, mode par défaut, type de politique, …) |
| **Gérer les règles de politiques SELinux** | Le gestionnaire a la possibilité d'effectuer toutes les opérations de gestion sur une règle SELinux (rédaction, ajout, activation / désactivation). |
| **Gérer les composants de politiques SELinux** | Le gestionnaire a la possibilité d'effectuer toutes les opérations de gestion d'un composant spécifique de la politique SELinux (ajout de composants, récupération de composant). |
| **Gérer les groupes de règles de politiques SELinux** | Le gestionnaire a la possibilité de consulter la liste des groupes de règles SELinux (Booléens et modules), d'activer/désactiver un groupe de règles quelconque. |

Tableau 4.1 : Identification des cas d'utilisation du provider SELinux.

La figure 4.2 présente le diagramme de cas d'utilisation global de notre Provider.

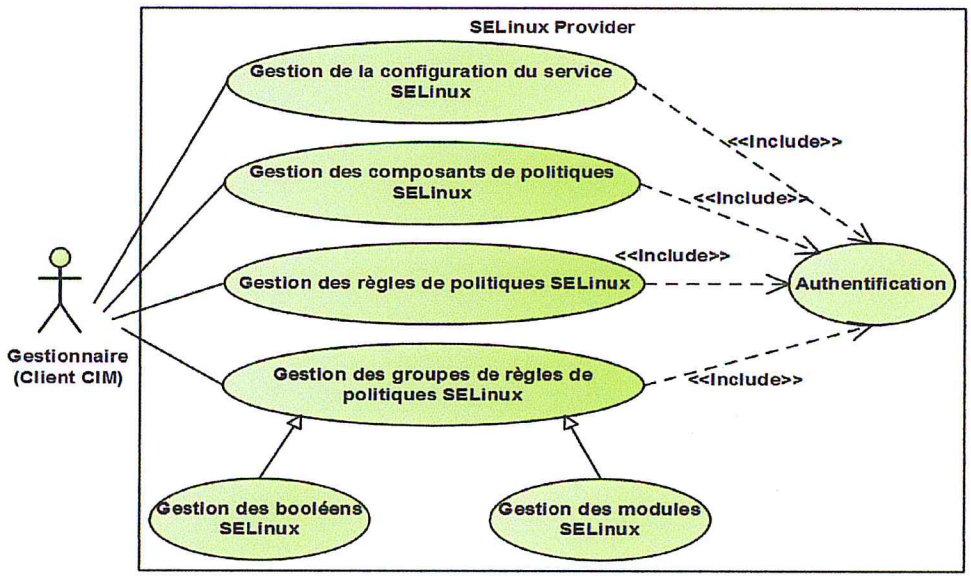



Figure 4.2. Diagramme de cas d'utilisation global du Provider SELinux.

Afin de garantir une sécurité maximale des données (confidentialité et intégrité), aucune action n'est permise sans authentification. Car le rôle principal de notre provider et d'assurer l'efficacité et la pertinence de la gestion des politiques de sécurité.

Dans la figure ci-dessus nous avons proposé une vue globale du système représentée par des cas d'utilisation généraux. Dans la section qui suit nous allons analyser en détail chacun de ces cas.

## 4.3.2 Raffinement des cas d'utilisation

Nous allons raffiner le diagramme de cas d'utilisation global de notre Provider. Ces diagrammes sont plus détaillés et correspondent à une vision informatique du système.

### 4.3.2.1 Diagramme de cas « Authentification »

La figure 4.3 illustre le diagramme du raffinement du cas d'utilisation « Authentification ».

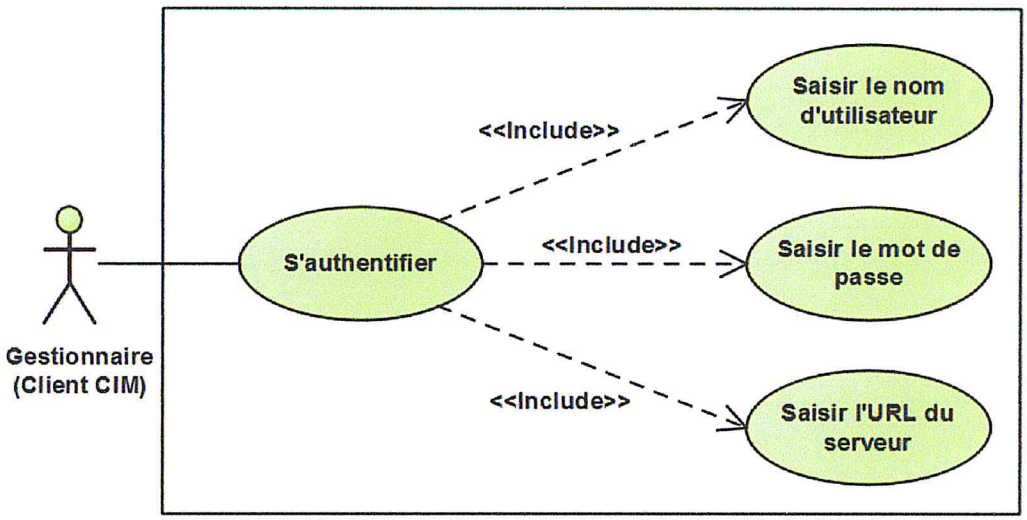



Figure 4.3. Raffinement du CU « Authentification ».

Afin de garantir la confidentialité et l'intégrité des données échangées entre le gestionnaire et la machine gérée, nous utilisons l'authentification à base de clés publique/privée fournie par OpenPegasus. La configuration du CIMOM OpenPegasus à utiliser le protocole qui se base sur le principe des certificats numériques, le chiffrement asymétrique et le hachage est nécessaire afin de contrer les attaques de l'**homme du milieu (Man in the middle)** par exemple et d'établir un canal chiffré entre le client et le serveur.

### 4.3.2.2 Diagramme de cas « Gestion de la configuration du service SELinux »

La figure 4.4 illustre le raffinement du cas d'utilisation « Gestion de la configuration du service SELinux ».

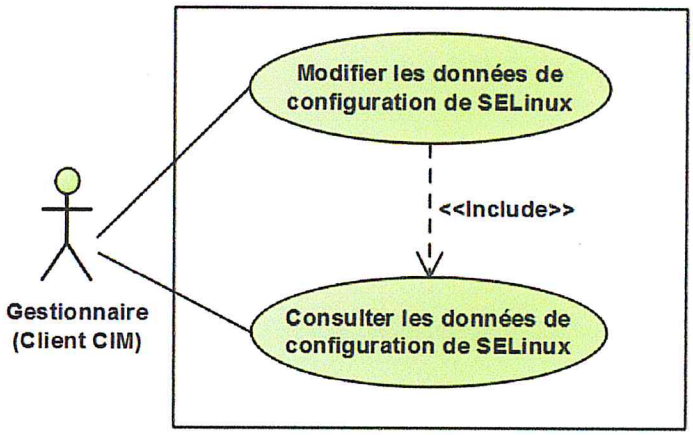



Figure 4.4. Raffinement du CU « Gestion de la configuration du service SELinux ».

Ce diagramme montre les fonctionnalités de la gestion de la configuration du service SELinux. La modification d'un paramètre de configuration nécessite la consultation de l'état actuel de ce dernier.

### 4.3.2.3 Diagramme de cas « Gestion des règles de politiques SELinux »

La figure 4.5 illustre le raffinement du cas d'utilisation « Gestion des règles de politiques SELinux ».

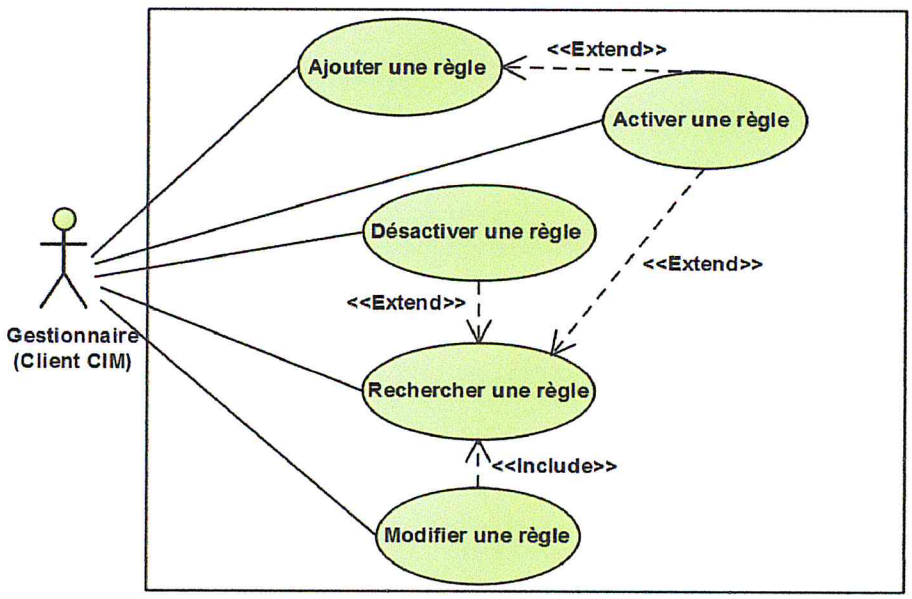



Figure 4.5. Raffinement du CU « Gestion des règles de politiques SELinux ».

Ce diagramme détaille les fonctionnalités de la gestion des règles de politiques SELinux. Un gestionnaire authentifié peut ajouter une règle, l'activer ou la désactiver. Il peut aussi modifier une règle mais après avoir cherché cette dernière.

### 4.3.2.4 Diagramme de cas « Gestion des composants de politiques SELinux »

La figure 4.6 illustre le raffinement du cas d'utilisation « Gestion des composants de politiques SELinux ».

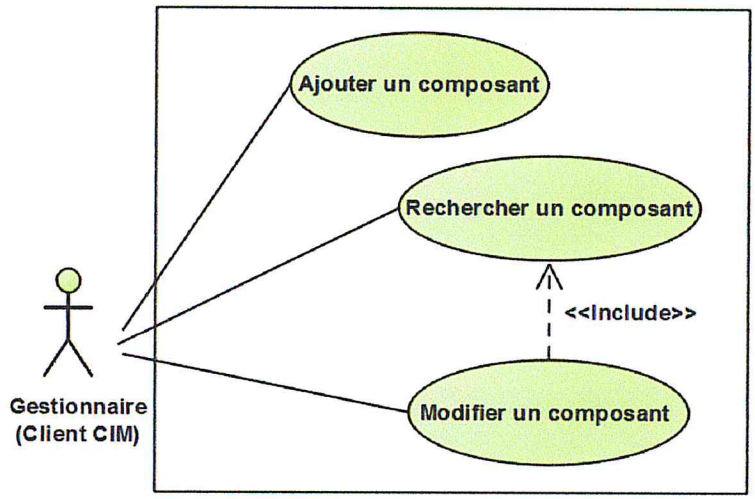



Figure 4.6. Raffinement du CU « Gestion des composants de politiques SELinux ».

Ce diagramme spécifie qu'un gestionnaire authentifié peut ajouter un composant SELinux (utilisateur, rôle, type, ...). Il peut aussi modifier un composant déjà existant après l'avoir cherché.

### 4.3.2.5 Diagramme de cas « Gestion des groupes des booléens SELinux »

La figure 4.7 illustre le raffinement du cas d'utilisation « Gestion des booléens SELinux ».

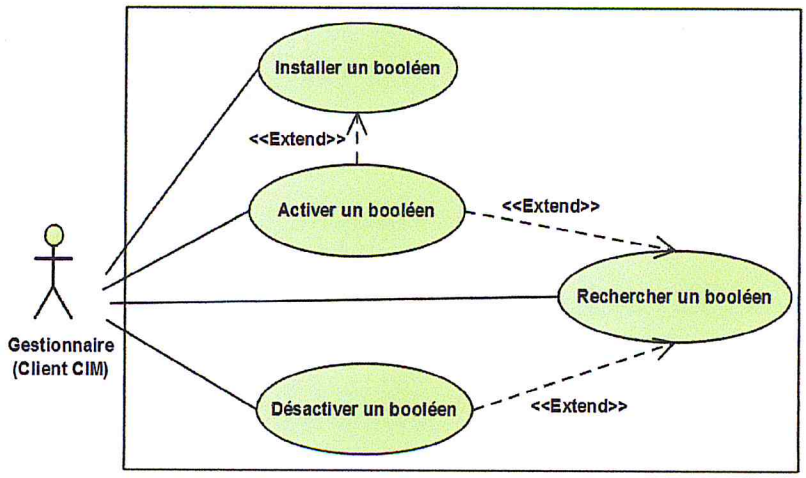


Figure 4.7. Raffinement du CU « Gestion des booléens SELinux ».

Ce diagramme montre les fonctionnalités de la gestion des booléens qui sont des goupes de règles SELinux. Un gestionnaire authentifié peut installer, chercher et activer ou désactiver un booléen s'il existe.

### 4.3.2.6 Diagramme de cas « Gestion des groupes des modules SELinux »

La figure 4.8 illustre le raffinement du cas d'utilisation « Gestion des modules SELinux ».

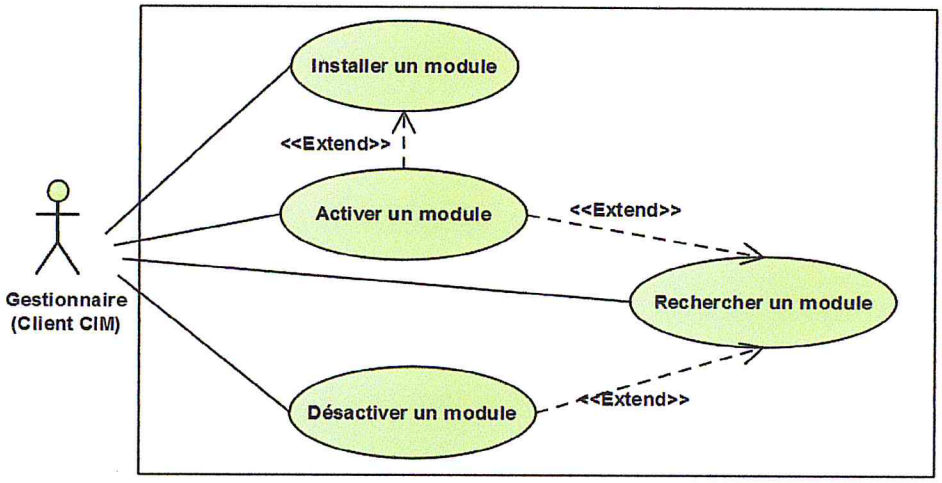



Figure 4.8. Raffinement du CU « Gestion des modules SELinux ».

Ce diagramme montre les fonctionnalités de la gestion des modules qui sont des groupes de règles SELinux. Un gestionnaire authentifié peut installer, chercher et activer ou désactiver un module s'il existe.

## 4.4 Modélisation CIM du domaine

Comme notre objectif principal est de gérer les politiques dans les environnements distribués et indépendamment des mécanismes sous-jacents, nous introduisons un schéma CIM générique appelé "**MAC_**" définissant un profil spécifique aux mécanismes MAC. Ce profil est ensuite spécialisé pour prendre en charge le système visé dans notre projet : SELinux. Par convention, les classes spécifiques à SELinux commencent par le préfixe **MAC_SEL**.

Bien entendu, grâce aux propriétés de l'héritage et du polymorphisme, toutes les relations dans le modèle de gestion des politiques CIM natif se propagent à nos classes dérivées.

### 4.4.1 Modèle générique du contrôle d'accès obligatoire

Afin d'atteindre notre premier objectif, nous avons développé un métamodèle abstrait et générique basé sur la norme CIM de DMTF pour décrire, appliquer et analyser les politiques abstraites de contrôle d'accès obligatoire au niveau intermédiaire.

Dans ce travail, nous avons décortiqué les règles et les groupes de règles de contrôle d'accès obligatoire en leurs composants élémentaires :

- Un sujet : processus source de l'action (déclenché par un utilisateur, ayant un certain rôle)
- Un objet : cible de l'action (fichier, répertoire, port réseau, etc.)
- Un ensemble de **conditions** (Si le **sujet** est autorisé à accéder à l'objet),
- Une **action** : autorisation, journalisation, transition, etc., sinon refus).
- Un ensemble de **permissions** (lecture, écriture, exécution, .....) sur des **objets gérés**.

Notre modèle est fondé sur la règle générale suivante : ***Si un sujet est autorisé à accéder à l'objet et que la permission demandée est accordée : permettre l'opération.***

À partir de ce principe et de la perspective du modèle CIM existant, nous avons développé un modèle de règle de politique de contrôle d'accès obligatoire élémentaire comme le montre la figure 4.9 :

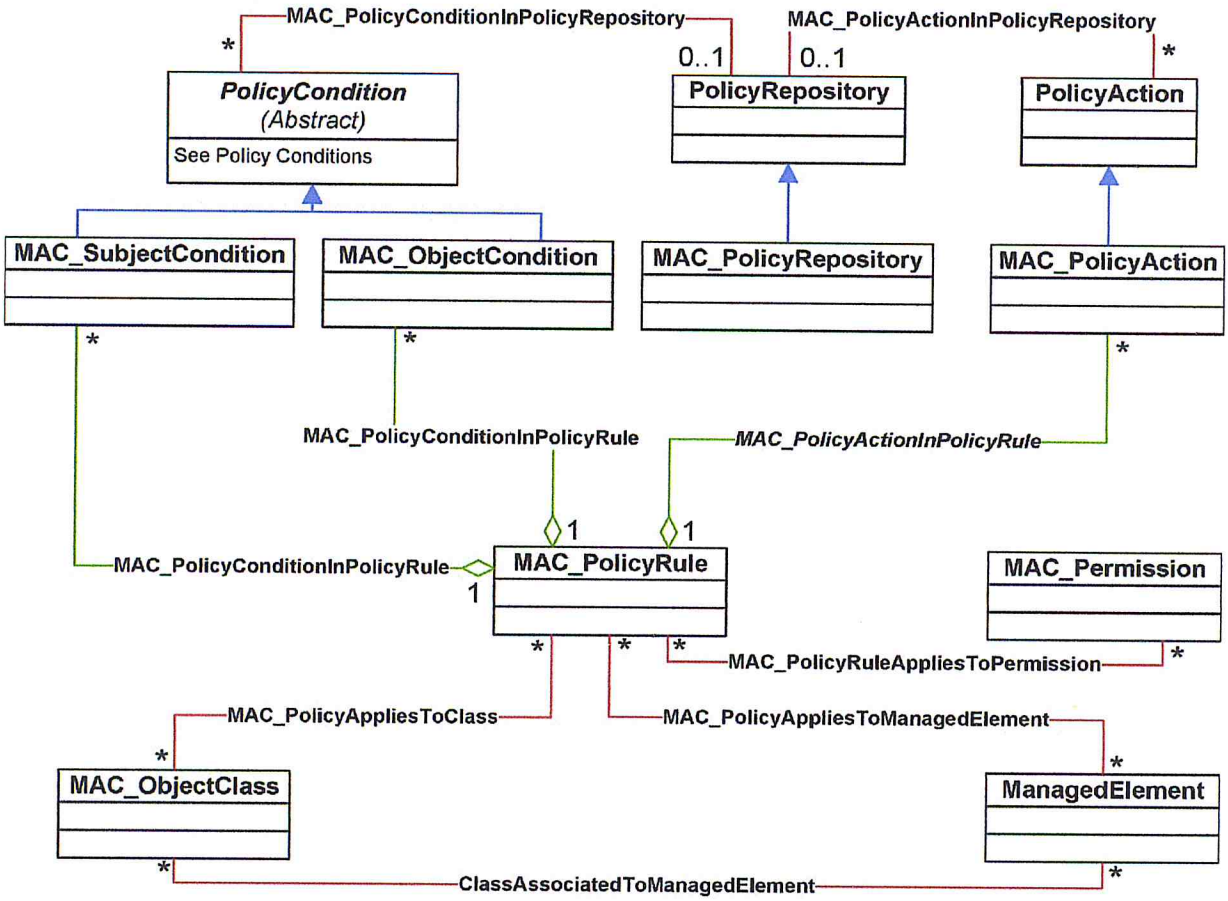



Figure 4.9. Structure de la règle de contrôle d'accès obligatoire

- **Classe MAC_PolicyRule :** cette classe représente une règle MAC de base.
- **Classe MAC_PolicyAction :** cette classe représente l'action d'autorisation, c'est-à-dire l'action d'une règle MAC à exécuter si les conditions pour cette règle s'évaluent à VRAI.
- **Classe MAC_SubjectCondition :** cette classe représente la requête sujet, c'est à dire la condition sur le sujet d'une règle à évaluer.
- **Classe MAC_ObjectCondition :** cette classe représente la requête objet, c'est à dire la condition sur l'objet d'une règle à évaluer.
- **Classe MAC_Permission :** cette classe représente le type d'accès demandé (lire, écrire, exécuter, etc.).

- **Classe ManagedElement :** cette classe représente tout objet système (fichier, répertoire, processus, interface réseau, port réseau, etc.) sur lequel la politique de contrôle d'accès obligatoire s'applique.
- **Classe MAC_ObjectClass :** Pour certains mécanismes MAC (comme SELinux), la classe ManagedElement est remplacée par une instance de MAC_ObjectClass représentant une catégorie d'objets gérés.

Le modèle qui vient d'être décrit est abstrait, il sera étendu par héritage afin d'implémenter des mécanismes MAC spécifiques. Dans notre projet, c'est le mécanisme SELinux qui sera modélisé et par la suite implémenté.

## 4.4.2 Extension du modèle de contrôle d'accès MAC

Dans cette section, nous présentons trois modèles CIM permettant de définir les politiques SELinux. Ces modèles sont les suivants :

- Le modèle de construction de règles SELinux.
- Le modèle spécifique aux contextes SELinux.
- Le modèle des booléens et des modules SELinux.

### 4.4.2.1 Modèle de construction de règle de politique SELinux

Ce modèle est dérivé du méta-modèle générique des règles de contrôle d'accès obligatoire décrit plus haut, il permet de représenter la structure des règles SELinux. La figure 4.10 représente ce modèle.

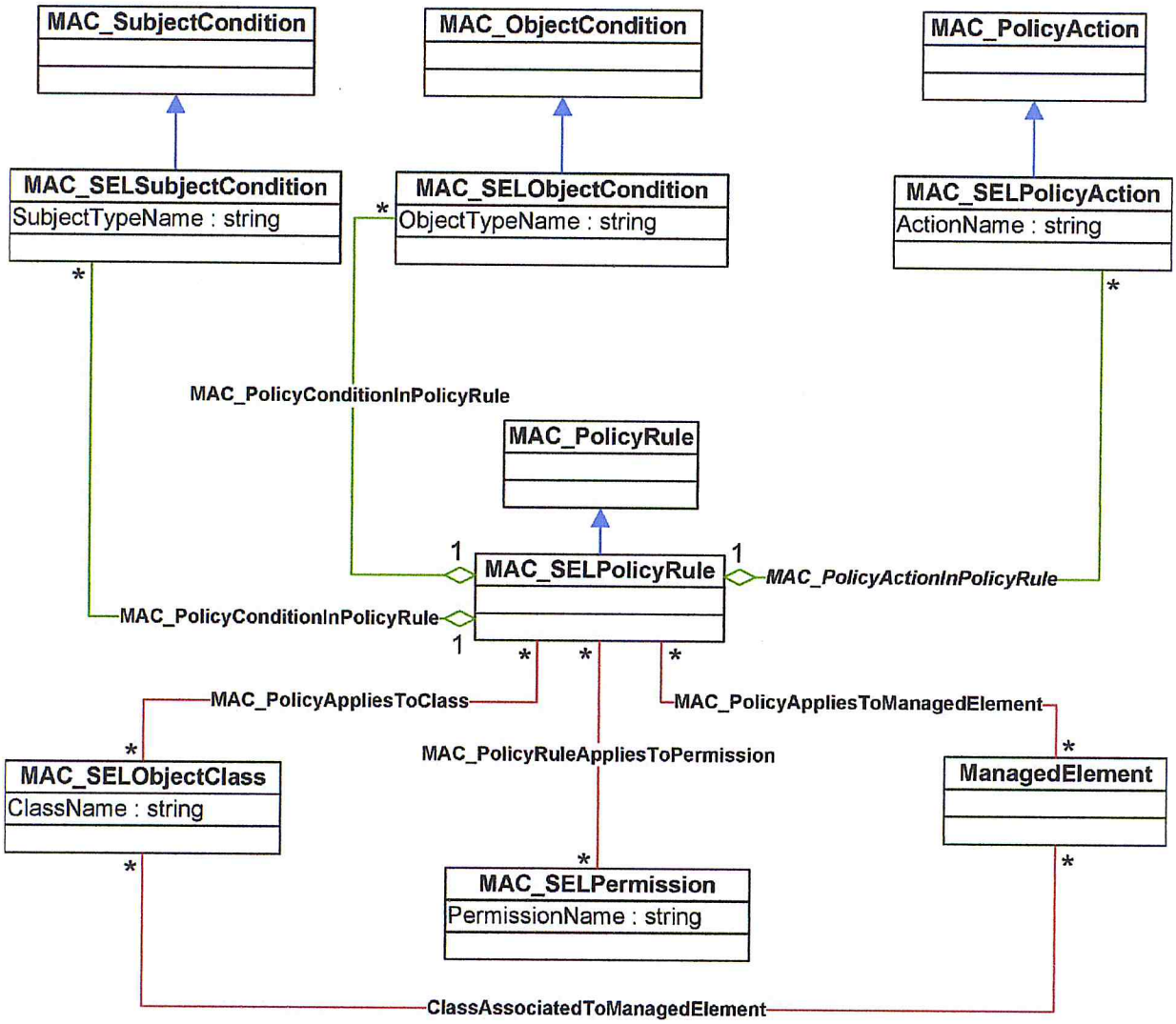



Figure 4.10. Modèle de construction des règles de politiques SELinux.

- **Classe MAC_SELPolicyRule :** cette classe représente une règle de politique de contrôle d'accès obligatoire SELinux.
- **Classe MAC_SELPolicyAction :** cette classe contient les actions d'autorisation SELinux. Parmi ces actions, on peut citer : **Allow, Dontaudit, Auditallow, Neverallow Type_transition, Type_Change** ...
- **Classe MAC_SELSubjectCondition :** cette classe contient les types attribués aux sujets (domaines).
- **Classe MAC_SELObjectCondition :** cette classe contient les types attribués aux objets (Labels ou étiquettes).

- **Classe MAC_SELPermission :** cette classe contient les types d'accès SELinux demandés tels que : **Read**, **Write**, **Execute**, **Getattr**, **Rename**, **Create** …
- **Classe MAC_SELObjectClass :** cette classe contient les classes d'objets SELinux tels que : **Dir**, **Blk_file**, **Chr_File**, **File**, **Fd**, **Sock_file** …

### 4.4.2.2 Modèle spécifique aux contextes SELinux

La figure 4.11 représente la partie du modèle qui permet de modéliser les contextes de sécurité SELinux (qui ne sont en fait que des politiques particulières) :

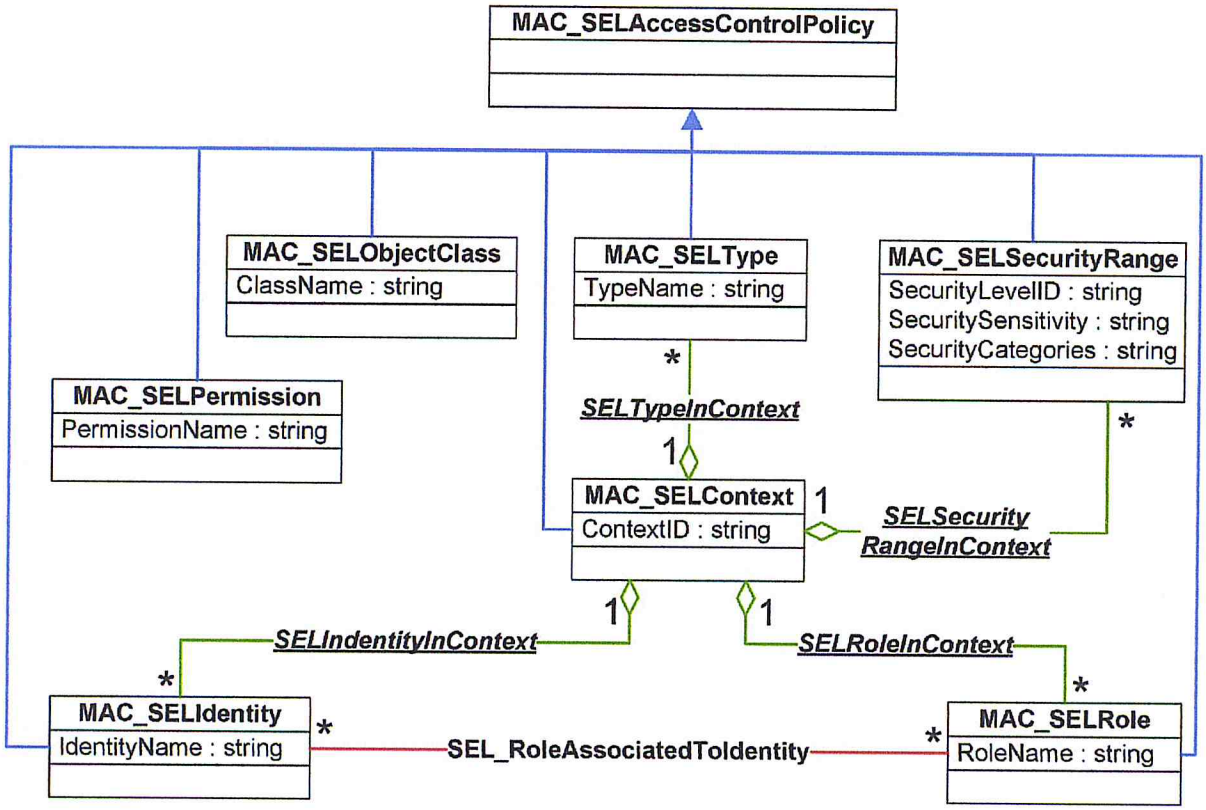



Figure 4.11. Modèle spécifique pour les politiques SELinux.

a) **Classe MAC_SELContext :** cette classe représente le contexte de sécurité (associé à tout objet du système d'exploitation par SELinux) qui agrège les classes représentant les attributs associés (Type, Identité d'utilisateur, Rôle et le MLS / MCS).

b) **Classe MAC_SELType :** cette classe représente le type associé aux éléments gérés. Généralement, les types associés aux objets (fichiers) sont appelés étiquettes ou labels, alors que les types associés aux sujets (processus) sont appelés domaines.

c) **Classe MAC_SELIdentity :** cette classe représente l'identité de l'utilisateur SELinux.

d) **Classe MAC_SELRole :** cette classe représente le rôle SELinux.

e) **Classe MAC_SELSecurityRange :** cette classe représente l'attribut MLS / MCS qui étend optionnellement le contexte de sécurité.

f) **Classe MAC_SELObjectClass :** cette classe représente la classe des objets concernés par les règles SELinux.

g) **Classe MAC_SELPermission :** cette classe représente l'ensemble des autorisations possibles sous SELinux.

### 4.4.2.3 Modèle des Booléens et des Modules SELinux

Le modèle global de politique MAC que nous avons proposé dans la figure 4.10 permet d'intégrer, théoriquement, les conditions par héritage. La figure 4.12 représente le modèle des groupes de règles SELinux (booléens et modules).

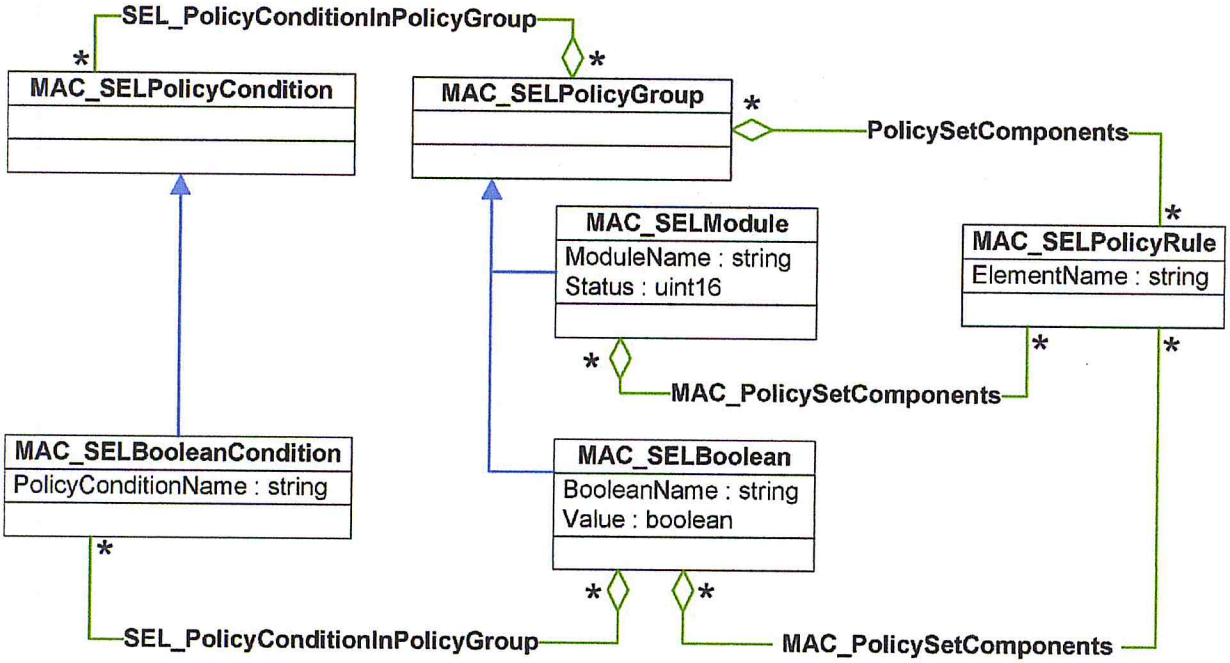



Figure 4.12. Modèle des Booléens et Modules SELinux.

a) **Classe MAC_SELPolicyGroup :** cette classe représente une agrégation d'instances de la classe MAC_SELPolicyRule qui ont la même stratégie de décision et les mêmes fonctions.

b) **Classe MAC_SELModule :** cette classe représente un module SELinux. Un module peut être considéré comme une méta-règle qui permet d'activer ou désactiver un ensemble de

règles SELinux élémentaires permettant d'atteindre un objectif de sécurité global (confiner un package logiciel, un service réseau, etc.).

**Propriétés :** Les propriétés de cette classe sont les propriétés **ModuleName** et **Status.**

- **ModuleName :** C'est le nom du module. Il est de type **chaine de caractères**.
- **Status :** Il représente le statut du module. Il est de type **entier**.

**Méthodes :** Les méthodes implémentées pour cette classe sont seulement les deux méthodes intrinsèques **Load** et **EnumerateInstances.**

c) **Classe MAC_SELBoolean :** cette classe représente un booléen SELinux qui est considéré comme une méta-règle regroupant deux ensembles de règles élémentaires permettant d'atteindre un certain objectif. Un booléen SELinux est caractérisé par une variable booléenne (d'où son nom) qui, lorsqu'elle est positionnée à TRUE, un ensemble de règles est activé, et lorsqu'elle est positionnée à FALSE, le premier ensemble de règle est désactivé et optionnellement l'autre ensemble de règles qui est activé.

**Propriétés :** Les propriétés de cette classe sont les propriétés **BooleanName** et **Value**.

- **BooleanName :** C'est le nom du booléen. Il est de type **chaine de caractères.**
- **Value :** Représente la valeur du booléen. Il est de type **booléen**.

**Méthodes :** Les méthodes implémentées pour cette classe sont seulement les deux méthodes intrinsèques **EnumerateInstances** et **Load.**

d) **Classe MAC_SELBooleanCondition :** cette classe représente la condition associée à un booléen SELinux.

### 4.4.3 Modèle de gestion des politiques SELinux

La classe MAC_SELPolicyGroup permet de lier le modèle spécifique des politiques SELinux avec la partie du modèle permettant l'application de ces politiques (représentée par la figure 4.13 dans la page suivante).

a) **Classe MAC_SELPolicyActivationService :** cette classe représente le seul et unique service SELinux présent dans le système permettant l'activation et la désactivation des règles (ou groupe de règles) définies par l'administrateur, par l'exécution de méthodes invoquant à leur tour les méthodes appropriées définies dans la classe MAC_SELAccessControlService.

**Propriétés :** Aucune propriété héritée n'est utilisée et aucune propriété n'est ajoutée à cette classe.

**Méthodes :** Les méthodes implémentées pour cette classe sont au nombre de huit, et sont :

- **ActivatePolicyRule / DeactivatePolicyRule :** Permet l'ajout / suppression d'une règle SELinux par appel à la méthode CreateRule / DeleteRule de la classe MAC_SELAccessControlService. Elle prend en argument une instance de la classe MAC_SELPolicyRule présente dans la base CIMOR. Cette instance sera ajoutée / supprimée si la méthode se termine avec succès.
- **ActivateBoolean / DeactivateBoolean :** Permet l'activation / désactivation temporaire d'un booléen SELinux par appel à la méthode SetBooleanCurrentValue permettant l'application du changement dans SELinux en mettant la valeur courante du booléen à "on" / "off". Elle prend en argument une instance de la classe MAC_SELBoolean dont l'attribut Value sera mis à "on" / "off" en cas de succès.
- **ActivateBooleanPermanetly / DeactivateBooleanPermanetly :** Similaires aux méthodes ActivateBoolean et DeactivateBoolean mais permettant un changement permanant de l'état des booléens (même après le redémarrage du système) par appel à la méthode SetBooleanPermanentValue de la classe MAC_SELAccessControlService.
- **ActivateModule / DeactivateModule :** Permettant l'activation / désactivation d'un module SELinux par appel à la méthode SetModuleStatus de la classe MAC_SELAccessControlService. Elle prend en argument une instance de la classe MAC_SELModule.

b) **Classe MAC_SELAccessControlService :** c'est la classe principale du profile DMTF de référence, permettant la modification de l'état de SELinux dans le système ainsi que la présentation des informations de base de ce système. Cette classe contient les méthodes permettant la modification de l'état SELinux (activation/désactivation SELinux, changement de la valeur d'un booléen, changement du statut d'un module, etc.). Une seule instance de cette classe peut être créée pour être associée au seul et unique service SELinux présent dans le système.

**Propriétés :** cette classe comprend les propriétés suivantes :

- **Name :** C'est le seul attribut clé de cette classe et il représente le nom du service. Il est de type **chaîne de caractères.**
- **CurrentStatus :** Il représente le statut de SELinux qui peut être ***Disable*** ou ***Enable***. Il est de type **entier.**

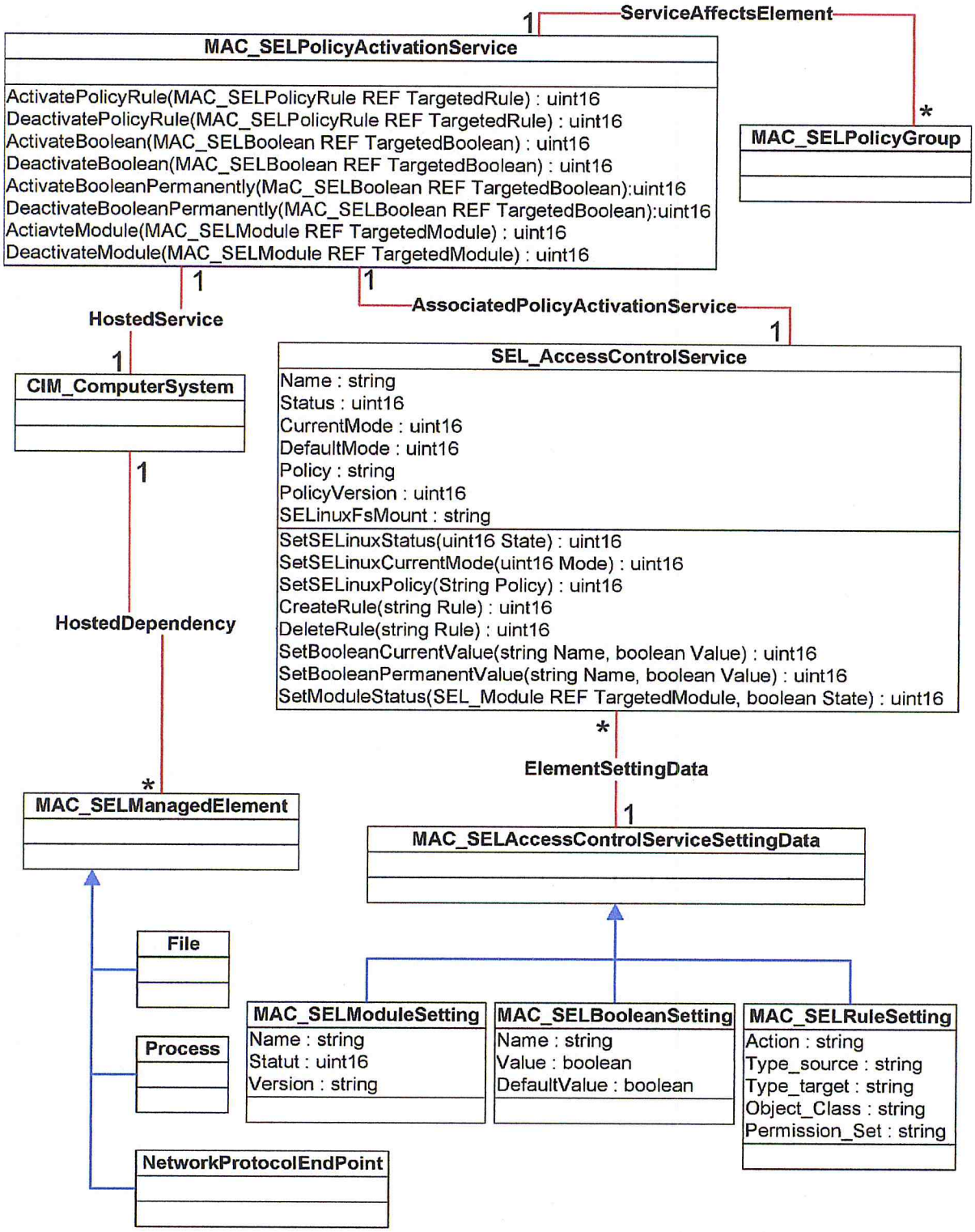



Figure 4.13. Diagramme de classes UML représentant l'application des politiques.

- **CurrentMode :** Il représente le mode courant de SELinux qui peut être ***Enforcing*** ou ***Permessive.*** Il est de type **entier**.
- **DefaultMode :** Il représente le mode par défaut écrit dans le fichier de configuration de SELinux. Ses valeurs possibles sont : ***Enforcing*** et ***Permissive.*** Il est de type **entier**.
- **PolicyType :** Il représente le nom de la politique appliquée par SELinux (le fichier **Policy**). Il est de type **chaîne de caractères**.
- **PolicyVersion :** Il représente la version du fichier de la politique appliquée par SELinux. Il est de type **chaîne de caractères**.
- **SELinuxFsMount :** Il représente le fichier racine de SELinux. Il est de type **chaîne de caractères**.

**Méthodes :** Cette classe comprend toutes les méthodes extrinsèques permettant d'interagir avec le système et une seule méthode intrinsèque EnumerateInstances :

- **EnumerateInstances :** retourne l'état de SELinux (les valeurs de tous les attributs).
- **SetSELinuxDefaultMode :** Permet le changement du mode par défaut de SELinux écrit dans le fichier de configuration. Cette méthode prend en argument le mode par défaut souhaité qui peut être : Permissive, Enforcing, ou Disable.
- **SetSELinuxCurrentMode :** Permet le changement du mode courant de SELinux. Elle prend en argument le mode courant souhaité qui peut être : Permissive ou Enforcing.
- **SetSELinuxPolicy :** Permet le changement de la politique SELinux utilisée. Le nom du fichier de la politique (situé dans le répertoire /etc/selinux) est donné en argument à cette méthode.
- **CreateRule / DeleteRule :** Permet la création / suppression d'une règle SELinux. La règle sous forme textuelle est donnée en argument.
- **SetBooleanCurrentValue / SetBooleanPermanentValue :** Permet le changement de la valeur courante / permanente d'un booléen SELinux. Cette classe prend en argument le nom du booléen et sa nouvelle valeur.
- **SetModuleStatus :** Permettant le changement du statut d'un module SELinux. Le nom du module ainsi que son statut (activé/désactivé) sont donnés en argument à cette méthode.
- **InstallModule :** permettant l'installation d'un nouveau module SELinux en donnant l'emplacement de ce dernier (le fichier .pp).

- **RemoveModule :** permettant la suppression d'un module SELinux existant. Le seul argument qu'elle prend est le nom du module concerné.

c) **Classe SEL_AccessControlServiceSettingData :** Elle présente les informations de configuration utilisées par SELinux. En utilisant ces informations, un administrateur système compose des politiques de contrôle d'accès pour le service à partir d'un hôte client distant. Il doit y avoir une cohérence entre les politiques contenues dans la classe MAC_SELPolicy et celles contenues dans la classe MAC_SELAccessControlServiceSettingData.

- **MAC_SELModuleSetting :** représente les informations de configuration des modules. Une instance de cette classe représente la configuration d'un module donné. Trois attributs sont définis dans cette classe : **Name, Version** et **Status**, qui représentent le nom, la version et le statut du module respectivement.
- **MAC_SELBooleanSetting :** représente les informations de configuration des booléens. Une instance de cette classe représente la configuration d'un booléen donné (activé/désactivé). Elle définit trois attributs qui sont : **Name, CurrentValue** et **PermanentValue**, qui représentent respectivement le nom, la valeur courante et la valeur par défaut du booléen.
- **MAC_SELRuleSetting :** représente la configuration d'une règle définie. Chaque instance de cette classe représente une règle déjà représentée par une instance de MAC_SELPolicyRule. Elle définit cinq attributs qui sont : **Action, Source, Target, Class** et **Permission** qui représentent respectivement l'action appliquée par la règle, le domaine (le sujet) auquel l'action est appliquée, l'étiquette (l'objet) sur laquelle l'action est appliquée, la classe (file, dir, socket, etc.) de la ressource sur laquelle l'action est appliquée et la permission ou le type d'accès défini dans la règle.

d) **Classe MAC_SELManagedElement :** le contrôle d'accès opéré par SELinux doit s'appliquer sur les ressources concrètes du système d'exploitation qui sont : les fichiers au sens du système Linux (fichiers ordinaires, fichiers exécutables, répertoires, périphériques, etc.), les processus et les ports réseaux (UDP ou TCP).

## 4.5 Conclusion

Dans ce chapitre, nous avons présenté l'architecture de notre Framework de gestion inspirée par le standard PCIM de l'IETF / DMTF, dédié à la gestion de politiques. Ensuite, nous avons introduit un modèle permettant la traduction des politiques de niveau brut de / vers des politiques de niveau intermédiaire fondées sur le méta-modèle CIM et relatives aux systèmes de contrôle d'accès obligatoire des systèmes d'exploitation.

Notre objectif est de gérer d'une manière centralisée et standardisée la sécurité au niveau des systèmes d'exploitation distribués. Afin d'atteindre cet objectif, nous avons identifié les objectifs suivants :

- Définir un méta-modèle générique, abstrait et standard pour représenter le contrôle d'accès obligatoire au niveau brut.
- Améliorer les politiques MAC existantes en ajoutant des conditions permettant l'activation / désactivation des politiques.
- Dériver le méta-modèle générique pour représenter des mécanismes MAC spécifiques tels que SELinux.

Pour atteindre ces objectifs, nous avons utilisé les standards du DMTF qui sont largement adoptées par les fournisseurs et la communauté pour la gestion des réseaux et des systèmes. A travers nos modèles, nos outils seront interopérables avec les implémentations CIM existantes.

La dernière étape de ce projet consiste à implémenter les modèles conçus dans ce chapitre sur un système Linux. Cette implémentation se traduit par l'installation d'une plateforme de gestion WBEM (client/serveur) ainsi que des outils nécessaires pour le développement des providers chargés de gérer les classes des modèles proposés. Ceci sera détaillé dans le chapitre qui suit.

# CHAPITRE 5 : IMPLÉMENTATION DU SYSTEME

## 5.1 Introduction

Afin de répondre aux objectifs de notre projet, nous avons développé un Provider CIM/WBEM permettant la gestion des politiques du contrôle d'accès SELinux. Il s'agit d'un pilote intégré dans une solution WBEM. Ce chapitre présente les différentes étapes d'implémentation de notre Framework. Cette implémentation permet tout d'abord, de prouver la possibilité de gérer des politiques de sécurité abstraites en format CIM en les traduisant en politiques concrètes SELinux, mais aussi d'offrir une plateforme pouvant être exploitée par un administrateur afin de composer, diffuser et appliquer, via un client WBEM, des politiques de sécurité sur des systèmes en réseau. Dans ce chapitre, nous présentons dans un premier temps les outils utilisés pour l'implémentation de notre solution, ensuite nous détaillerons les fonctionnalités que nous avons développées en montrant essentiellement comment notre Provider, que nous avons appelé SELinuxProvider, permettrait de gérer les politiques du contrôle d'accès SELinux. Au final, nous présenterons quelques illustrations des fonctionnalités les plus importantes de notre Framework.

## 5.2 Les choix techniques

Pour implémenter notre Framework, plusieurs technologies ont été utilisées :

### 5.2.1 Linux CentOS 7

Nous avons choisi l'environnement Linux CentOs 7 puisque sur cette version, SELinux est installé par défaut. CENTOS est un clone au système Linux commercial RedHat et en même temps très similaire au système communautaire Linux Fedora, ce qui élargi la portée de notre contribution à l'ensemble des installations basées sur cette famille d'OSs. Mais la raison principale de ce choix réside dans le fait que SELinux est le mécanisme MAC par défaut de cette famille de systèmes d'exploitation. En plus, il n'est pas nécessaire d'être expert SELinux pour pouvoir l'utiliser, en effet un grand nombre de modules et de règles dédiés à un bon nombre de packages logiciels et de services réseaux ont déjà été développés et sont maintenus par la communauté et sont prêt à l'emploi.

Toutefois notre implémentation est exploitable sur n'importe quel système de la famille Linux pourvu qu'il supporte le module SELinux.

### 5.2.2 OpenPegasus 12.2.0

Le projet OpenPegasus fourni par l'OpenGroup, est une implémentation open-source des standards CIM et WBEM du DMTF. Il est conçu pour être portable (multi-OS) et est modulaire. Il est codé en C++ afin qu'il traduise efficacement les concepts d'objet des objets CIM dans un modèle de programmation, mais tout en conservant la performance et l'efficacité d'un langage compilé. OpenPegasus implémente l'ensemble des composants de l'architecture WBEM ainsi que les différents points de gestion de politiques de notre architecture (PAP, PDP, PRP) à l'exception des Providers CIM qui représentent les PEP. Nous avons choisi ce projet pour sa stabilité, sa maturité et pour l'exhaustivité de ses fonctionnalités.

### 5.2.3 CIMPLE 2.0.24

Nous avons choisi le Framework de développement CIMPLE dédié à la programmation de Providers et de clients CIM. Cet outil facilite énormément l'implémentation par la génération de squelettes de programmes en C++ à partir de la définition en MOF des classes CIM relatives aux ressources cibles. La puissance de ce Framework réside en son approche qui consiste à générer des classes C++ qui correspondent (et qui sont identiques) aux classes MOF avec une puissante gestion des types de données et la génération des squelettes des méthodes intrinsèques et extrinsèques. Le travail du développeur consiste à coder les méthodes en fonction de la (des) ressource(s) visée(s) par le Provider ou le client.

### 5.2.4 Libselinux et libsemanage

Nous avons utilisé les fonctions offertes par les bibliothèques de programmation natives à SELinux, notamment libselinux et libsemanage pour l'instrumentation du module SELinux et ses politiques à travers les méthodes de notre Provider. Ces APIs étant écrites en langage C, leur intégration dans les squelettes de programmes C++ générés par le Framework CIMPLE est tout simplement naturelle.

### 5.2.5 Wbemcli 1.6.1

Nous avons choisi ce client WBEM pour tester notre Provider (gestion des politiques SELinux par l'invocation des méthodes des classes CIM correspondante) comme point d'administration des politiques PAP.

## 5.3 Implémentation des providers

Nous rappelons que la gestion de SELinux avec ses politiques de sécurité en format CIM est assurée par le Provider que nous allons élaborer. Les composants de notre Provider seront plusieurs modules dont chacun représente l'implémentation de chaque classe des modèles que nous avons conçus (présentés dans le chapitre 4). Cependant nous n'implémentons pas des modules pour les classes CIM_LogicalFile et CIM_SoftwareIdentity car des implémentations existent à travers d'autres projets (SBLIM Project, OpenLMI et autres), Ceci démontre l'interopérabilité et l'intégration de Providers issus de différentes implémentations dans le même système WBEM, car en effet, le client CIM ne manipule que des instances de classes « abstraites » qui sont appariées par le serveur CIMOM avec l'implémentation correspondante. Cette caractéristique représente l'un des points forts du standard CIM.

Dans la suite de ce rapport, un module qui permet de maintenir une classe, dont le nom est **"MAC_SELClass"**, est nommé **"MAC_SELClass_Provider".**

### 5.3.1 Le module MAC_SELPolicyRule_Provider

Ce composant du Provider nous permet de manipuler les instances de MAC_SELPolicyRule ajoutées par l'administrateur. La récupération des règles présentes dans le système n'est pas implémentée, à cause de leur nombre trop important dans le système (500 mille règles de type AV présentes lors de l'installation de SELinux). Un administrateur ne peut donc récupérer, supprimer ou modifier que des instances individuelles identifiées par leurs clés. Les méthodes implémentées dans ce module sont **CreateInstance** qui est invoquée lorsque l'administrateur veut créer une nouvelle règle, **GetInstance**, **DeleteInstance** et **ModifyInstance**.

### 5.3.2 Le module MAC_SELModule_Provider

Ce composant du Provider permet de récupérer l'état des politiques SELinux de type module présents dans le système et modélisées par la classe MAC_SELModule. Il implémente les méthodes intrinsèques suivantes :

- **Load** qui s'exécute automatiquement lors du chargement du Provider (en général lors du (re)démarrage du serveur WBEM). Cette méthode permet de récupérer dynamiquement l'état des modules SELinux en utilisant les fonctions des bibliothèques de ce dernier, pour ensuite créer et enregistrer les instances correspondantes dans le CIMOR. Lors de la création d'une instance, les attributs **ModuleName, Version** et **Status** sont fixés respectivement avec le

**nom,** la **version** et le **statut** du module correspondant. La méthode Load permet en fait d'assurer, au démarrage du système, la synchronisation et la cohérence de l'état actuel des modules SELinux avec les politiques CIM correspondantes abstraites présentes dans le CIMOR.

- **EnumerateInstances** Permet de retourner toutes les instances de la classe MAC_SELModule présentes dans le CIMOR (créées par la méthode Load). Elle est invoquée par un client CIM pour connaître l'état des modules. Cette méthode ne fait appel à aucune fonction de la bibliothèque SELinux, car elle interagit seulement avec le CIMOM.
- **GetInstance** Permet de retourner une instance dans le CIMOR. Elle reçoit en paramètre les valeurs des attributs clés.
- **CreateInstance** Permet de créer une instance de MAC_SELModule dans le CIMOR en recevant en argument les valeurs de ses attributs. Cette méthode doit être invoquée seulement lors de l'installation du module par le MAC_SELAccessControlService_Provider si non, une invocation manuelle par l'administrateur risque de créer des problèmes de cohérence (instance crée sans module installé).
- **DeleteInstance** qui est similaire à CreateInstance mais qui permet de supprimer une instance du CIMOR lorsqu'un module est désinstallé.
- **ModifyInstance** qui permet de modifier une instance en donnant en argument les nouvelles valeurs de ModuleName et Status. Elle est invoquée lors de l'activation/désactivation d'un module.

### 5.3.3 Le module MAC_SELBoolean_Provider

Similaire à MAC_SEL_Module_Provider, il permet de récupérer l'état des booléens SELinux. Il implémente les méthodes suivantes :

- **Load** qui est similaire à celle du module précédent. Elle récupère l'état des booléens et les enregistre dans le CIMOR sous formes d'instances de MAC_SELBoolean.
- **EnumerateInstances** retourne les instances de type MAC_SELBoolean présentes dans le CIMOR. Invoquée par un client WBEM, elle récupère l'état des booléens (préalablement chargé par la méthode Load).
- **GetInstance** permet de récupérer l'état d'un booléen dont la clé est donnée en argument.

- **InstallBoolean** et **RemoveBoolean** qui permettent respectivement d'installer et désinstaller un

  booléen SELinux. La première reçoit en argument le chemin vers le fichier (.pp) représentant le booléen à installer, et la seconde reçoit le nom du booléen à supprimer.

### 5.3.4 Le module MAC_SELPolicyActivationService_Provider

Ce module contient les méthodes qui permettent d'activer ou désactiver tous les types de politiques présents dans le système. Il ne fait qu'invoquer des méthodes de la classe MAC_SELAccessControlService afin d'appliquer les changements imposés par la politique donnée en argument.

Les règles activées/désactivées par ce module sont les booléens et les modules SELinux existants dans le système, ainsi que les règles ajoutées par l'administrateur à travers notre système de gestion.

### 5.3.5 Le module MAC_SELAccessControlService_Provider

Ce module regroupe toutes les méthodes qui permettent de modifier l'état de SELinux. Rappelons que l'état de SELinux est défini par les paramètres suivants : son mode, la politique qu'il applique, l'état actuel des modules, etc. Les méthodes implémentées par ce module sont :

- **SetSELinuxStatus** qui permet de changer le mode par défaut de SELinux ou de le désactiver. Elle reçoit en argument un entier dont les valeurs possibles sont : -1 pour *Disable*, 0 pour *Permissive* et 1 pour *Enforcing*. Cette méthode modifie le fichier /etc/selinux/config, les changements ne sont pris en compte qu'après le redémarrage du système.
- **SetSELinuxCurrentMode** qui permet de changer le mode courant de SELinux (*Enforcing* ou *Permissive*). Elle reçoit en argument un entier qui peut être : 0 pour *Permissive* ou 1 pour *Enforcing*.
- **SetSELinuxPolicy** qui permet de changer la politique utilisée par SELinux. Elle reçoit en argument une chaîne de caractères contenant le nom de la politique souhaitée.
- **CreateRule** qui permet d'ajouter une règle à la politique SELinux appliquée. Sous SELinux, une règle doit être contenue dans un module, c'est pour ça que cette méthode place la règle créée dans un module dédié. L'ajout d'une règle se fait en trois étapes comme le montre la figure 5.1 suivante :

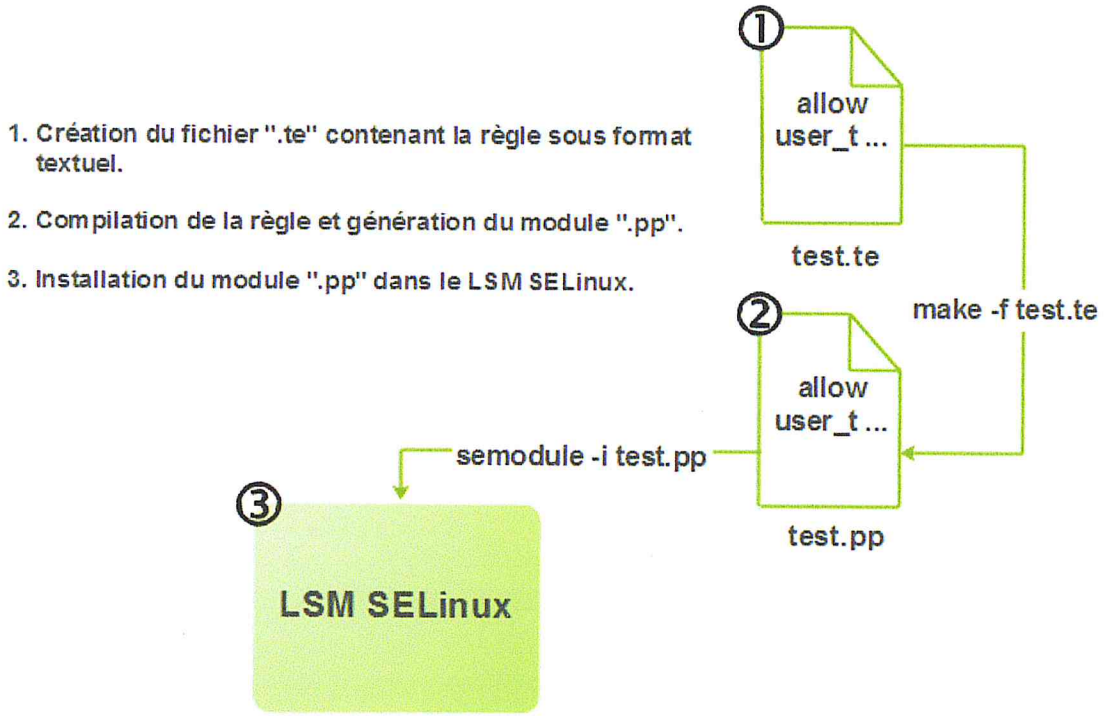



Figure 5.1. Étapes de l'ajout d'une règle SELinux.

Si les étapes s'exécutent correctement, la méthode **CreateInstance** de la classe **MAC_SELRuleSetting** est invoquée pour créer une instance représentant une information de configuration correspondante à la règle ajoutée et permettant à l'administrateur de vérifier le succès de son opération. Cette méthode ne doit pas être invoquée directement par le client, mais via le module **MAC_SELPolicyActivationService_Provider.**

- **DeleteRule** qui permet de supprimer une règle dont le nom est donné en argument. Si la suppression réussit, elle supprime l'instance de **MAC_SELRuleSetting** correspondante. Comme la méthode CreateRule, elle ne doit pas être invoquée directement par un client.
- **SetBooleanCurrentValue** et **SetBooleanPermanentValue** qui permettent de changer l'état d'un *boolean* SELinux, dont le nom est donné en argument. La première le fait de façon temporaire (jusqu'au prochain redémarrage) et la seconde de façon permanente. Le changement apporté par l'une de ces méthodes est appliqué sur l'instance de MAC_SELBooleanSetting Correspondante au booléen concerné.
- **SetFileLabel** et **SetProcessDomain** qui permettent de changer respectivement l'étiquette d'un fichier et le domaine d'un processus (lancés à partir du nom d'un fichier exécutable).

Chaque méthode reçoit en argument le chemin du fichier/exécutable et l'étiquette/domaine concerné.

- **SetModuleStatus** qui reçoit le nom d'un module ainsi que son statut souhaité (activé/désactivé). Elle est invoquée par les méthodes **ActivateModule** et **DeactivateModule** de la classe **MAC_SELPolicyActivationService** pour changer le statut du module concerné. Si ce changement réussit, l'attribut **Status** de l'instance de MAC_SELModuleSetting est mis à jour.
- **InstallModule** et **RemoveModule** qui permettent respectivement d'installer et désinstaller un module SELinux. La première reçoit en argument le chemin vers le fichier (.pp) représentant le module à installer, et la seconde reçoit le nom du module à supprimer.

### 5.3.6 Le module MAC_SELAccessControlPolicy_Provider

On parle ici de sept modules qui maintiennent les classes : **MAC_SELContext**, **MAC_SELType** **MAC_SELIdentity**, **MAC_SELRole**, **MAC_SELSecurityRange**, **MAC_SELObjectClass** et **MAC_SELPermission**. On rappelle que ces classes représentent des états administratifs des politiques SELinux. Les méthodes implémentées pour chacune de ces classes sont :

- **Load** qui permet de charger ces états administratifs SELinux dans leurs classes appropriées dans le CIMOR.
- **EnumerateInstances** et **GetInstance** qui retournent les instances à partir du CIMOR.
- **CreateInstance**, **DeleteInstance** et **ModifyInstance** qui permettent de gérer les instances de la classe concernée.

## 5.4 Phase de développement : détails de la programmation

Dans ce qui suit, nous présentons la démarche de l'implémentation d'un module quelconque de notre Framework qui est schématisée dans la figure 5.2. Cette démarche est la même suivie pour n'importe quel module.

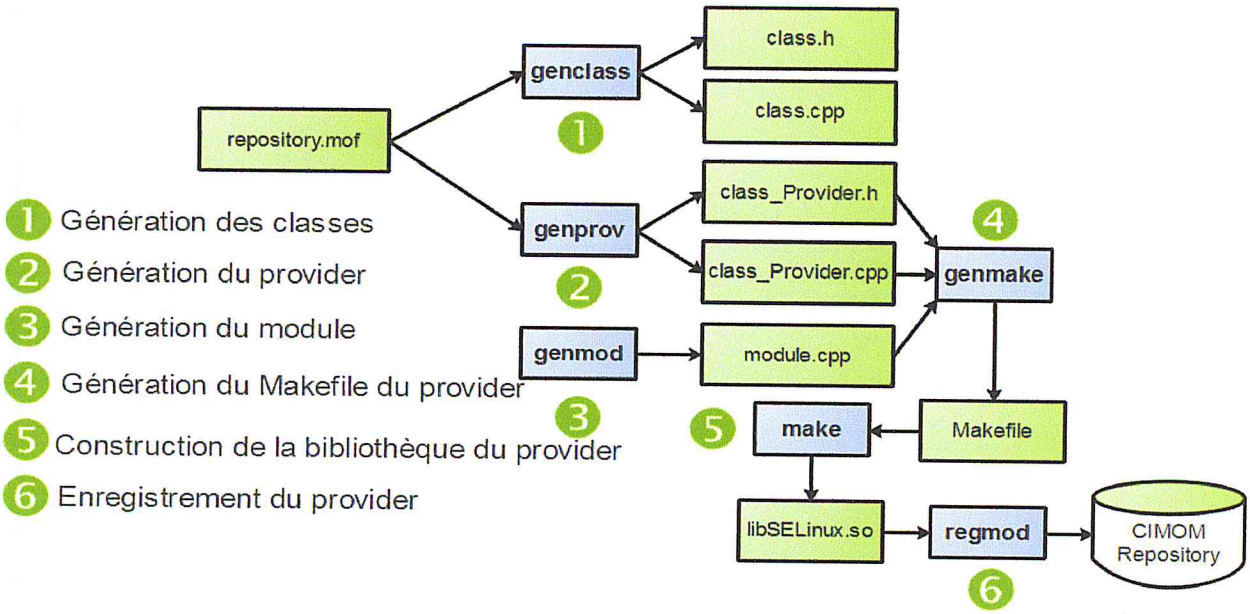



Figure 5.2. Etapes de programmation du Provider.

## a. Définition de la classe en format MOF :

Cette étape consiste en la traduction des classes des modèles conçues dans le chapitre précédent (sous la forme de diagrammes de classes UML) vers le format textuel MOF. Chacune des classes de notre modèle est définie dans un fichier nommé Nom_classe.mof ensuite elle et incluse dans un fichier nommé repository.mof en utilisant la directive **#pragma include ("Nom_classe.mof")** afin de bien structurer ce fichier.

## b. Génération du projet C++ :

Dans cette étape, nous utilisons le Framework CIMPLE qui nous permet de traduire nos classes sous format MOF en classes C++. Et cela se fait en utilisant la commande **genclass** fournie par ce framework. Cette phase va générer automatiquement des définitions de classes C++ utilisant les types et structures de données correspondant aux types définis par le standard CIM. Ainsi le programmeur va pouvoir manipuler aisément des classes et objets C++ mappés aux classes et objets CIM.

## c. Génération des squelettes des providers :

Cette étape permet la génération des squelettes de programmes des providers sous forme de fichiers d'entête (.h) et de code C++ (.cpp). Ces fichiers qui contienderont les définitions ainsi que les corps des méthodes CIM aussi bien intrinsèques qu'extrinsèques (définis au sein des classes CIM).

La séparation des fichiers de définition des classes CIM de la première phase et les fichiers des providers de cette seconde phase va permettre de rendre indépendant la définition des classes CIM des opérations sur celles-ci. Ainsi le programmeur pourra opérer des changements dans son modèle MOF et régénérer les classes C++ correspondants avec la commande genclass, comme il pourrait opérer des modifications sur le code des opérations CIM sans toucher au modèle associé. Pour générer les squelettes de nos providers, nous exécutons la commande **genprov** du Framework CIMPLE.

### d. L'ajout des codes des méthodes :

Après la génération des squelettes des providers, nous pouvons maintenant procéder à l'implémentation des méthodes définies de chaque classe.

### e. Génération du module :

A cette étape, la commande **genmod** est utilisée afin de permettre la génération du fichier module.cpp contenant le code et les informations permettant l'enregistrement du provider auprès du serveur CIMOM.

### f. Compilation du Provider

Avant de procéder à la compilation de nos programmes, nous devons générer le fichier Makefile avec la commande **genmak** du Framework CIMPLE. Ce fichier décrit l'ordre de la compilation des différents fichiers sources ainsi que d'autres informations nécessaires à cette opération comme les chemins des bibliothèques d'OpenPegasus de CIMPLE (libcimple, libcommon et libpegadaptater). Bien entendu nous devons y rajouter manuellement les chemins des bibliothèques de SELinux (libselinux et libsemanage).

Nous procédons ensuite à la compilation de notre projet en lançant la commande **make**. Cette opération va générer des fichiers objet intermédiaires (ayant l'extension « **.o**») qui seront compilés à leur tour pour générer notre bibliothèque partagée **SELinuxProvider.so,** le fruit de notre travail.

### g. Enregistrement du Provider

L'enregistrement du provider constitue la dernière étape. Cette étape permet d'instruire le serveur CIMOM d'associer toutes les requêtes sur les classes que nous avons défini dans le chapitre conception aux programmes implémentés par notre module construit grâce au Framework CIMPLE.

## 5.5 Plan de test

Dans cette section, nous présentons un plan de tests qui couvre les fonctionnalités clés de notre provider dont :

- La modification et vérification des paramètres de configuration du service SELinux.
- La création d'une règle, l'activation /désactivation de cette règle SELinux.
- La consultation des composants SELinux (contexte par exemple) et étiquetage d'un fichier.
- L'activation et la désactivation des groupes de règles SELinux (un booléen par exemple).

### 5.5.1 Modification de la configuration SELinux

Ce test consiste en :

1. Récupération des paramètres de configuration du service SELinux.
2. Modification de l'un des paramètres de configuration (CurrentMode par exemple).
3. Vérification du succès de l'opération de modification.

```
[root@localhost providers]# wbemcli ei https://root:SELinux28@localhost:5989/root/cimv2:MAC
_SELAccessControlService
localhost:5989/root/cimv2:MAC_SELAccessControlService.SystemCreationClassName="",SystemName
="",CreationClassName="",Name="SELinux" InstanceID=,Caption=,Description=,ElementName=,Gene
ration=,InstallDate=,OperationalStatus=,StatusDescriptions=,Status=,HealthState=,Communicat
ionStatus=,DetailedStatus=,OperatingStatus=,PrimaryStatus=,EnabledState=5,OtherEnabledState
=,RequestedState=12,EnabledDefault=2,TimeOfLastStateChange=,AvailableRequestedStates=,Trans
itioningToState=12,SystemCreationClassName=,SystemName=,CreationClassName=,PrimaryOwnerName
=,PrimaryOwnerContact=,StartMode=,Started=,LoSID=,LoSOrgID=,ImplementationType=,Name="SELin
ux",CurrentStatus=1,CurrentMode=0,DefaultMode=0,Policy="targeted",PolicyVersion=28,SElinuxF
sMount="/sys/fs/selinux"
[root@localhost providers]# ciminvoke MAC_SELAccessControlService SetCurrentMode NewMode=1
return=1

[root@localhost providers]# wbemcli ei https://root:SELinux28@localhost:5989/root/cimv2:MAC
_SELAccessControlService
localhost:5989/root/cimv2:MAC_SELAccessControlService.SystemCreationClassName="",SystemName
="",CreationClassName="",Name="SELinux" InstanceID=,Caption=,Description=,ElementName=,Gene
ration=,InstallDate=,OperationalStatus=,StatusDescriptions=,Status=,HealthState=,Communicat
ionStatus=,DetailedStatus=,OperatingStatus=,PrimaryStatus=,EnabledState=5,OtherEnabledState
=,RequestedState=12,EnabledDefault=2,TimeOfLastStateChange=,AvailableRequestedStates=,Trans
itioningToState=12,SystemCreationClassName=,SystemName=,CreationClassName=,PrimaryOwnerName
=,PrimaryOwnerContact=,StartMode=,Started=,LoSID=,LoSOrgID=,ImplementationType=,Name="SELin
ux",CurrentStatus=1,CurrentMode=1,DefaultMode=0,Policy="targeted",PolicyVersion=28,SElinuxF
sMount="/sys/fs/selinux"
```

Figure 5.3. Consultation et modification d'un paramètre de configuration du service SELinux.

## 5.5.2 Création et activation de règles MAC_SELPolicyRule

Ce test consiste en :

1. Utilisation de l'éditeur de texte **gedit**, dont le domaine est **unconfined_t**, pour lui permettre l'accès en écriture à un fichier, étiqueté **fichier_t**, dont il a seulement la permission de lecture. Au début, un accès en écriture est tenté à partir d'une session super utilisateur **root** via l'éditeur gedit. La figure 5.4 montre que cet accès est refusé.

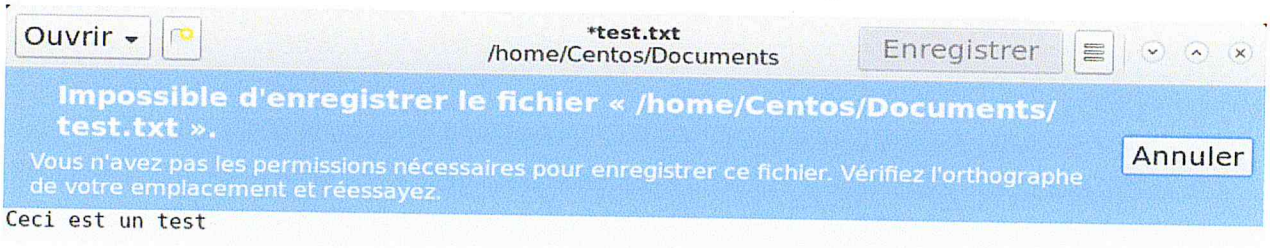



Figure 5.4. Modification du fichier **test.txt** refusée par SELinux.

2. Consultation du fichier de journalisation **/var/log/audit/audit.log** pour prouver que l'accès a été bloqué par SELinux. La figure 5.5 le montre :

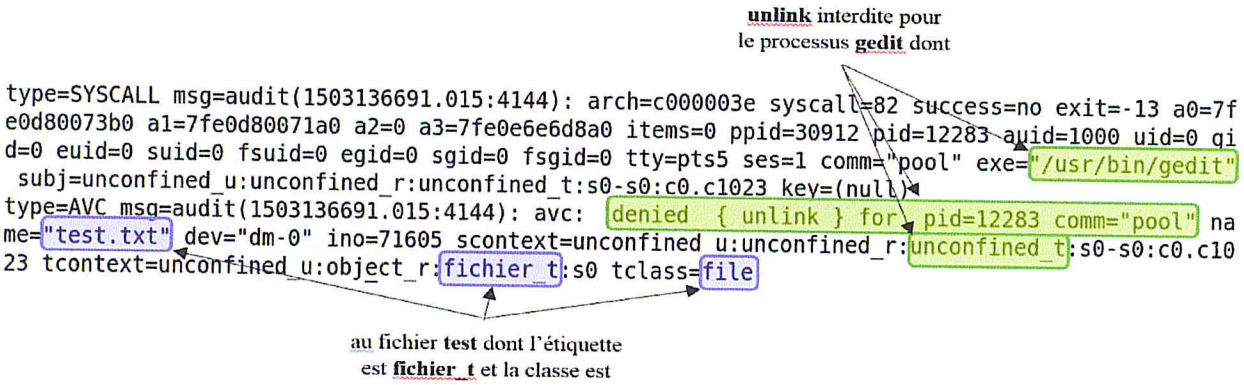



Figure 5.5. Trace du blocage de l'exécutable gedit dans les journaux.

3. Création et activation de deux politiques CIM **MAC_SELPolicyRule** permettant d'autoriser la modification du fichier **test** par **gedit.**

```
[root@localhost providers]# wbemcli ci 'https://root:SELinux28@localhost:5989/root/cimv2:MAC_SELPolicyRule.SystemCreationClassName="MAC_SELPolicyRule",SystemName="localhost",CreationClassName="MAC_SELPolicyRule",PolicyRuleName="allow:unconfined_t:fichier_t:file:unlink"' SystemCreationClassName="MAC_SELPolicyRule",SystemName="localhost",CreationClassName="MAC_SELPolicyRule",PolicyRuleName="allow:unconfined_t:fichier_t:file:unlink"
localhost:5989/root/cimv2:MAC_SELPolicyRule.CreationClassName="MAC_SELPolicyRule",PolicyRuleName="allow:unconfined_t:fichier_t:file:unlink",SystemCreationClassName="MAC_SELPolicyRule",SystemName="localhost"
[root@localhost providers]# ciminvoke MAC_SELPolicyActivationService ActivatePolicyRule targetPolicyRule="allow:unconfined_t:fichier_t:file:unlink"
return=0
```

Figure 5.6. Ajout d'une règle MAC_SELPolicyRule.

### 5.5.3 Test de consultation des contextes et étiquetage de fichier

Ce test consiste en :

1. Affichage des contextes SELinux représentés par la classe MAC_SELContext.

```
[root@localhost providers]# wbemcli ei https://root:SELinux28@localhost:5989/root/cimv2:MAC_SELContext
localhost:5989/root/cimv2:MAC_SELContext.SystemCreationClassName="MAC_SELContext",SystemName="localhos
t",CreationClassName="MAC_SELContext",PolicyID="Context" InstanceID="",Caption="",Description="",Eleme
ntName="",Generation=,CommonName="",PolicyKeywords=,PolicyDecisionStrategy=0,PolicyRoles=,SystemCreati
onClassName="MAC_SELContext",SystemName="localhost",CreationClassName="MAC_SELContext",Enabled=0,Polic
yID="Context",ContextId="system u:object r:default t:s0"
localhost:5989/root/cimv2:MAC_SELContext.SystemCreationClassName="MAC_SELContext",SystemName="localhos
t",CreationClassName="MAC_SELContext",PolicyID="Context" InstanceID="",Caption="",Description="",Eleme
ntName="",Generation=,CommonName="",PolicyKeywords=,PolicyDecisionStrategy=0,PolicyRoles=,SystemCreati
onClassName="MAC_SELContext",SystemName="localhost",CreationClassName="MAC_SELContext",Enabled=0,Polic
yID="Context",ContextId="system u:object r:etc runtime t:s0"
localhost:5989/root/cimv2:MAC_SELContext.SystemCreationClassName="MAC_SELContext",SystemName="localhos
t",CreationClassName="MAC_SELContext",PolicyID="Context" InstanceID="",Caption="",Description="",Eleme
ntName="",Generation=,CommonName="",PolicyKeywords=,PolicyDecisionStrategy=0,PolicyRoles=,SystemCreati
onClassName="MAC_SELContext",SystemName="localhost",CreationClassName="MAC_SELContext",Enabled=0,Polic
yID="Context",ContextId="system u:object r:quota db t:s0"
```

Figure 5.7. Consultation des contextes SELinux.

2. Affectation d'un nouveau contexte de sécurité SELinux au fichier test.txt.

```
[root@localhost Documents]# ls -Z test.txt
-rw-r--r--. root root unconfined_u:object_r:usr_home_t:s0 test.txt
[root@localhost Documents]# ciminvoke MAC_SELAccessControlService SetFileContext Path=
"/home/Centos/Documents/test.txt" NewContext="unconfined_u:object_r:usr_t:s0"
return=0

[root@localhost Documents]# ls -Z test.txt
-rw-r--r--. root root unconfined_u:object_r:usr_t:s0   test.txt
```

Figure 5.8. Modification du contexte de sécurité d'un fichier.

### 5.5.4 Test d'activation de MAC_SELBoolean

Ce test consiste à :

1. Désactiver le booléen **selinuxuser_ping** qui permet d'empêcher le domaine **user_t** de pinger.

```
[root@localhost providers]# wbemcli gi https://root:SELinux28@localhost:5989/root/cimv2:MAC_SELBool
eanSetting.InstanceID="selinuxuser_ping"
localhost:5989/root/cimv2:MAC_SELBooleanSetting.InstanceID="selinuxuser_ping" Caption="",Descriptio
n="",Generation=,InstanceID="selinuxuser_ping",ElementName="",ConfigurationName="",ChangeableType=0
,ComponentSetting=,SoID=,SoOrgID=,PrincipalType=0,OtherResourceType="",ResourceType=,BooleanName="s
elinuxuser_ping",BooleanCurrentValue=TRUE,BooleanPermanentValue=TRUE
[root@localhost providers]# ciminvoke MAC_SELPolicyActivationService DeactivateBoolean TargetBoolea
n="selinuxuser_ping"
return=0

[root@localhost providers]# wbemcli gi https://root:SELinux28@localhost:5989/root/cimv2:MAC_SELBool
eanSetting.InstanceID="selinuxuser_ping"
localhost:5989/root/cimv2:MAC_SELBooleanSetting.InstanceID="selinuxuser_ping" Caption="",Descriptio
n="",Generation=,InstanceID="selinuxuser_ping",ElementName="selinuxuser_ping",ConfigurationName="",
ChangeableType=0,ComponentSetting=,SoID=,SoOrgID=,PrincipalType=0,OtherResourceType="",ResourceType
=,BooleanName="selinuxuser_ping",BooleanCurrentValue=FALSE,BooleanPermanentValue=TRUE
```

Figure 5.9. Désactivation du booléen **selinuxuser_ping**.

2. Vérifier que le domaine user_t ne peut pas effectuer un ping.

```
[Centos@localhost Bureau]$ id -Z
user u:user r:user t:s0
[Centos@localhost Bureau]$ ping localhost
ping: socket: Permission non accordée
```

Figure 5.10. Permission de ping non accordée au domaine **user_t**.

3. Réactiver le booléen **user_ping** afin d'autoriser le domaine **user_t** à pinger.

```
[root@localhost providers]# wbemcli gi https://root:SELinux28@localhost:5989/root/cimv2:MAC_SELBool
eanSetting.InstanceID="selinuxuser_ping"
localhost:5989/root/cimv2:MAC_SELBooleanSetting.InstanceID="selinuxuser_ping" Caption="",Descriptio
n="",Generation=,InstanceID="selinuxuser_ping",ElementName="selinuxuser_ping",ConfigurationName="",
ChangeableType=0,ComponentSetting=,SoID=,SoOrgID=,PrincipalType=0,OtherResourceType="",ResourceType
=,BooleanName="selinuxuser_ping",BooleanCurrentValue=FALSE,BooleanPermanentValue=TRUE
[root@localhost providers]# ciminvoke MAC_SELPolicyActivationService ActivateBoolean TargetBoolean=
"selinuxuser_ping" return=0

[root@localhost providers]# wbemcli gi https://root:SELinux28@localhost:5989/root/cimv2:MAC_SELBool
eanSetting.InstanceID="selinuxuser_ping"
localhost:5989/root/cimv2:MAC_SELBooleanSetting.InstanceID="selinuxuser_ping" Caption="",Descriptio
n="",Generation=,InstanceID="selinuxuser_ping",ElementName="",ConfigurationName="",ChangeableType=0
,ComponentSetting=,SoID=,SoOrgID=,PrincipalType=0,OtherResourceType="",ResourceType=,BooleanName="s
elinuxuser_ping",BooleanCurrentValue=TRUE,BooleanPermanentValue=TRUE
```

Figure 5.11. Réactivation du booléen **selinuxuser_ping**.

4. Vérifier que le booléen a été bien activé et que le domaine **user_t** peut pinger.

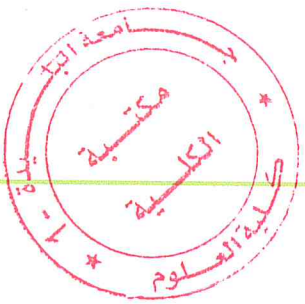
```
[Centos@localhost Bureau]$ id -Z
user u:user r:user t:s0
[Centos@localhost Bureau]$ ping localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.039 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.044 ms
```

Figure 5.12. Permission de ping accordée au domaine **user_t**.

## 5.6 L'IHM de gestion des politiques SELinux

Afin de montrer les fonctionnalités du Provider que nous avons implémenté, nous avons utilisé l'outil **PENCIL** pour créer un Design représentant les interfaces que nous pourrons implémenter plus tard pour faciliter l'exploitation de notre système et en tirer le maximum de profit.

Dans ce qui suit, nous présentons des captures d'écrans de l'IHM que nous comptons réaliser :



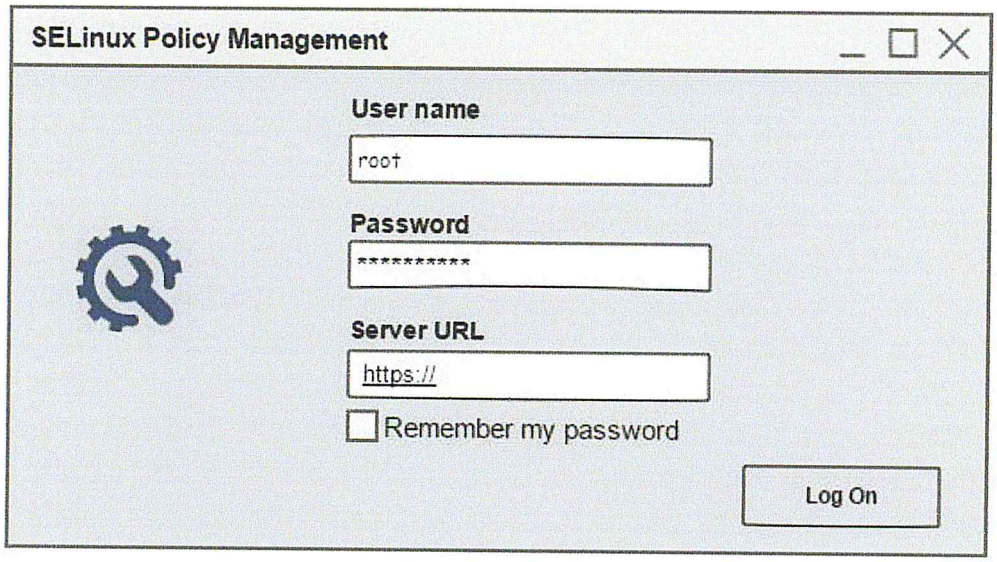



Figure 5.13. Fenêtre de connexion.

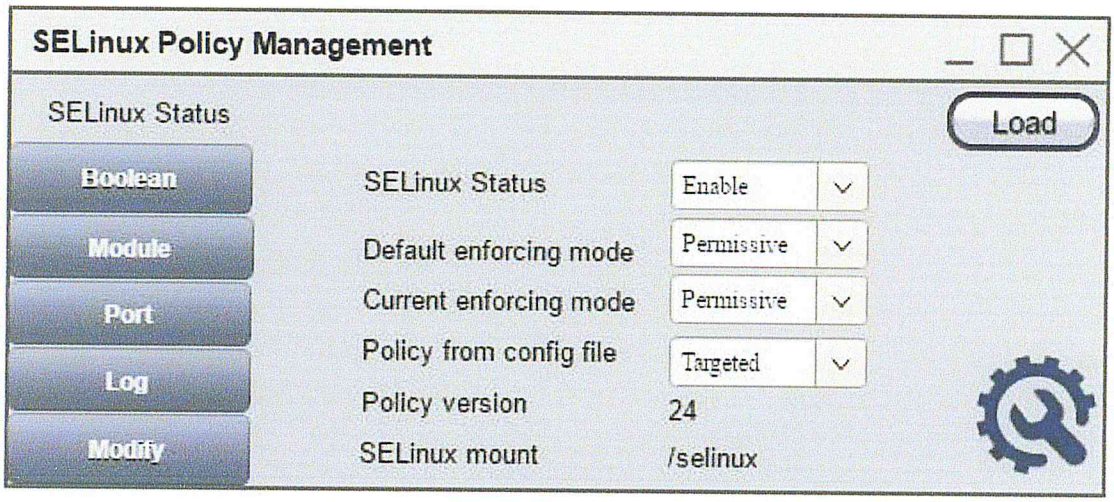



Figure 5.14. Onglet SELinux Statut.

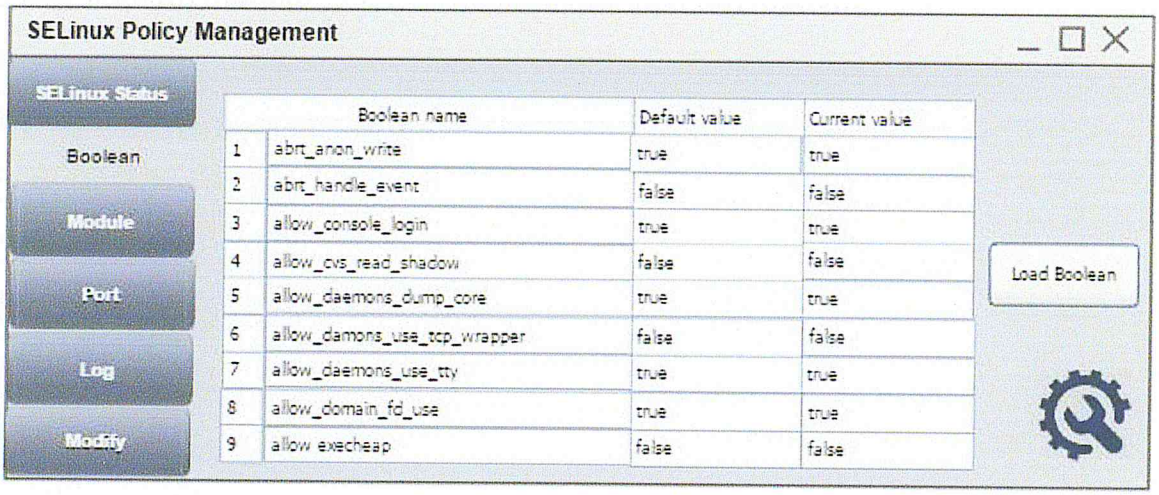



Figure 5.15. Onglet de gestion des Booléens SELinux.

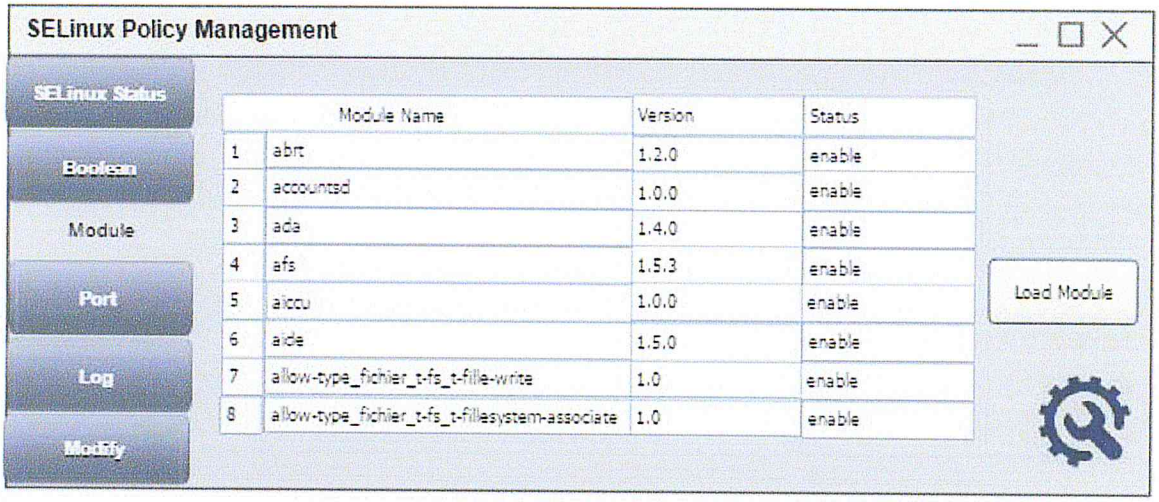



Figure 5.16. Onglet de gestion des Modules SELinux.

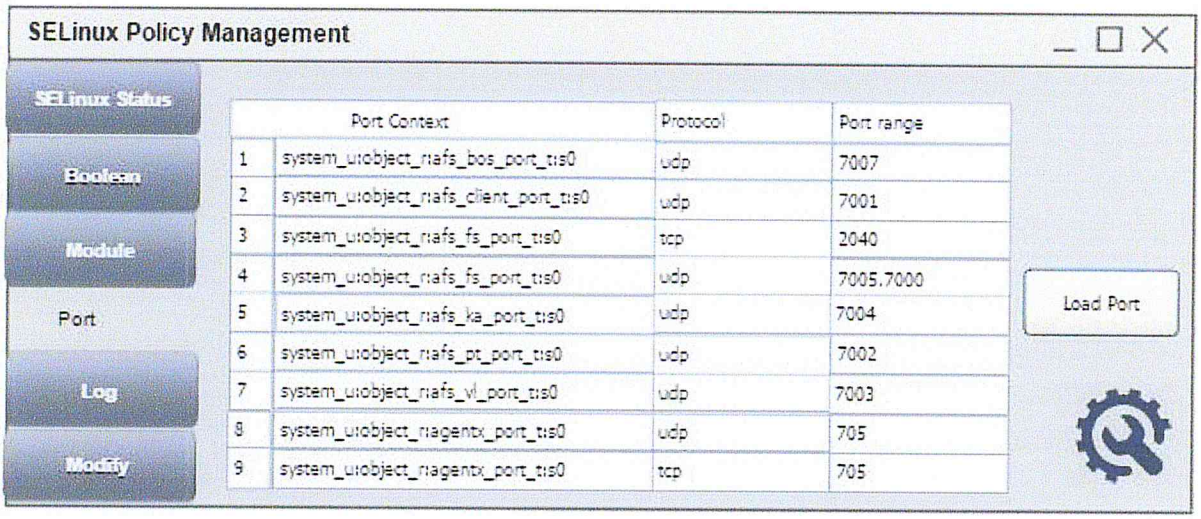



Figure 5.17. Onglet gestion des Ports.

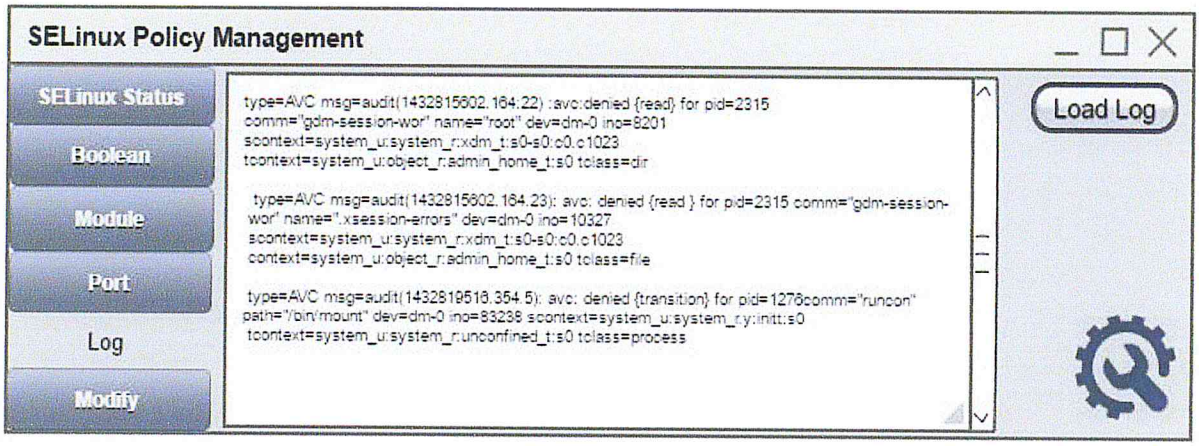



Figure 5.18. Onglet des Logs SELinux.

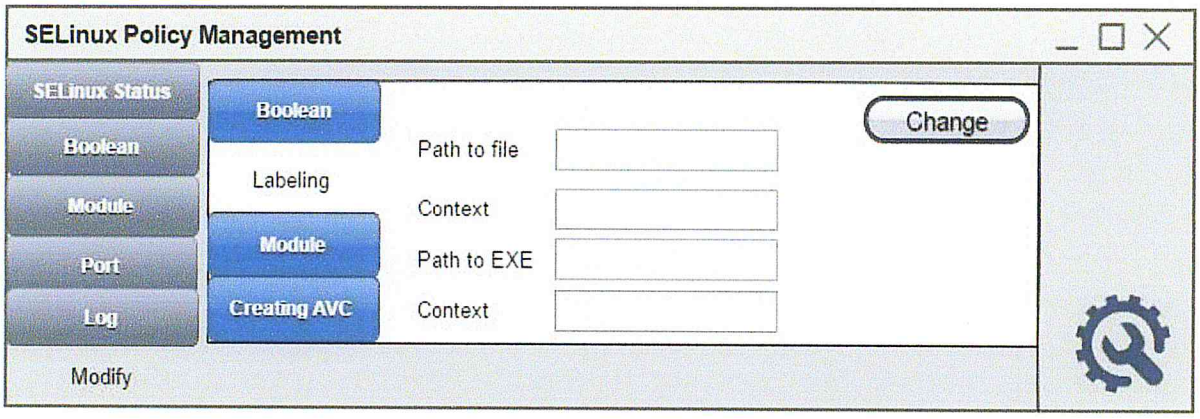



Figure 5.19. Onglet dédié à la manipulation du Labeling.

## 5.7 Conclusion

Dans ce chapitre, nous avons exposé l'implémentation de notre Framework. Nous avons d'abord présenté les choix techniques adoptés dans notre projet, ensuite nous avons expliqué comment nous avons procédé pour réaliser notre Provider qui représente le fruit de notre travail. Il s'agit d'une bibliothèque que nous avons nommé SELinuxProvider.so qui offre une gestion abstraite et homogène des politiques du contrôle d'accès obligatoire SELinux en se basant sur le standard CIM. Ainsi l'intégration de notre travail dans la solution WBEM de l'OpenGroup (ou n'importe quelle autre solution mettant en œuvre le standard CIM/WBEM comme SBLIM-SFCB) va nous permettre d'implémenter une gestion distribuée, homogène et unifiée des politiques de sécurité dans un réseau. En effet, il suffit d'avoir des connaissances de base en CIM pour pouvoir éditer et composer des politiques de sécurité « abstraites » à appliquer sur n'importe quel système doté d'un dispositif de contrôle d'accès obligatoire et d'un serveur CIM/WBEM. Ensuite il suffit d'avoir un client CIM/WBEM pour pouvoir interagir avec plusieurs systèmes distants (donc distribués) en leur transmettant les politiques à appliquer en format CIM (abstrait) que notre Provider (préalablement intégré à ces systèmes) va traduire en politiques SELinux (concrètes et opérationnelles). Enfin ceci va permettre d'avoir une vision unifiée sur les politiques de sécurité appliquées à tout le parc informatique, DataCenter ou tout autre composant de l'infrastructure.

# CONCLUSION GENERALE ET PERSPECTIVES

Le travail présenté dans ce document porte sur la gestion des politiques de sécurité du contrôle d'accès obligatoire (Mandatory Access Control) ciblant le mécanisme MAC dénommé SELinux déployé dans les systèmes d'exploitation Linux.
Ce travail se positionne dans le domaine de la gestion de la sécurité des systèmes d'exploitation, en effet le contrôle d'accès représente l'un des mécanismes importants qui garantissent cette sécurité en délimitant le champ d'action des processus sur les ressources du système par le biais d'utilisation des politiques de sécurité.

Dans ce projet, nous avons commencé d'abord par etudier quelques approches de gestion existantes (OSIManagement de l'ISO,SNMP de l'IETF et WBEM du DMTF) ainsi que quelques mecanismes de contrôle d'acces obligatoire(AppArmor, SELinux et GrSecurity). Vu que notre travail consiste en l'adaptation du standard CIM/WBEM au module LSM SELinux afin de concevoir un système de gestion de politiques générique indépendant des plateformes et repondant aux exigences de DMTF, nous avons détaillé l'approche de gestion CIM/WBEM constituée de deux standards émergeant des activités du DMTF.Ces standards (CIM/WBEM) permettent de développer des composants systèmes d'administration d'infrastructures de telle façon qu'ils soient indépendants de toute plateforme et neutres par rapport aux technologies employées et interopérables entre eux. Ensuite, une bonne etude et maitrise de SELinux était necessaire pour atteindre notre objectif et dont une synthese est presentée au chapitre 3.

Dans notre projet, le langage standard CIM offre la possibilité de spécifier les règles de contrôle d'accès SELinux de manière générique. Quant à l'architecture standard WBEM, elle permet l'adaptabilité dynamique du comportement du système par le biais de politiques de gestion en s'appuyant sur une architecture modulaire qui permet de favoriser l'intégration du système de gestion dans un environnement géré.

Le principal objectif de notre travail consiste en l'adaptation du standard CIM/WBEM au module LSM SELinux afin de concevoir un système de gestion de politiques générique indépendant des plateformes et répondant aux exigences du DMTF. Pour cela en suivant les recommandations du DMTF qui préconisent que toute modélisation du système géré doive être conforme à l'un des profils qu'ils proposent, nous avons conçu un modèle représentant les politiques de sécurité de SELinux en se basant sur les profils CIM dédiés aux politiques et en particulier les politiques de contrôle d'accès intégrées dont les concepts sont présentés dans le chapitre 3.

Une fois l'approche CIM/WBEM et le mécanisme SELinux ont été bien étudies, nous sommes passés à la conception. Dans cette partie, nous avons proposé une architecture de gestion en s'inspirant de l'architecture de gestion PCIM et en se basant sur l'architecture standard WBEM, qui permet l'adaptabilité dynamique du comportement du système par le biais de politiques de gestion en s'appuyant sur une architecture modulaire qui permet de favoriser l'intégration du système de gestion dans un environnement géré. Puis, nous avons présenté une modélisation fonctionnelle du système en utilisant des diagrammes de cas d'utilisation. Par la suite ; et en suivant les recommandations DMTF qui préconisent que toute modélisation du système géré doivent être conforme à l'un des profils qu'ils proposent, nous avons conçu un modèle représentant les politiques de sécurité de SELinux en se basant sur les profils CIM dédiés aux politiques et en particulier les politiques de contrôle d'accès intégrées. Une fois notre modèle validé, nous avons procédé à son implémentation, il s'agit d'un provider WBEM conforme à la spécification CMPI qui implémente les classes et les associations de notre modèle de gestion des politiques SELinux.un autre système proche du notre a été réalisé dans le cadre du projet OpenLMI. Cependant, ce système n'est conforme a aucun profils du DMTF (et donc problème de standardisation).de plus il ne permet de gérer que les ports réseaux et les booléens ainsi que l'état de SELinux.dans notre contribution réside dans la gestion des politiques SELinux par le standard WBEM ce qui :

- Unifie la gestion des politiques SELinux ;
- Assure une gestion centralisée et distribuée des politique SELinux.

En finalité et comme perspective à notre travail, nous envisageons :

- Améliorer les fonctionnalités de notre provider pour traiter la totalité des politiques spécifiques SELinux ;
- Développer une IHM dédiée afin de facilité=er l'exploitation de notre Povider et tirer le maximum de profil des fonctionnalités qu'il présente ;
- Adapter d' autres mécanismes de sécurité tel que APPARMOR, GRESECURITY, TOMOYO et autres au standard CIM/WBEM afin d'élaborer un système global de gestion des politiques de sécurité abstraite fondées sur le standard CIM permettant une gestion unifiée et distribuée.

## Références bibliographiques

[1] Aiko Pras, 1995: Network Management Architecture, pp 23-36.

[2] Pierre Olivier Conti. «Agents Intelligents»: Emergence d'une nouvelle technologie pour la Gestion de réseaux. **[En ligne]** Thèse de doctorat en informatique et réseaux. Paris: Ecole nationale supérieure des télécommunications, 2000, 238p. Format PDF .Disponible sur: <https://www.eurecom.fr/fr/publication/900/download/ce-contpi-000717.pdf>
(Consulté le 18/05/2017).

[3] Anonyme : <https://tools.ietf.org/html/rfc3460>.

[4] Anonyme : <http://airccse.org/journal/jcsit/1109s12.pdf>.

[5] Romain Laborde. Un cadre formel pour le raffinement de l'information de gestion de sécurité réseau : Expression et Analyse. **[En ligne]** Thèse de doctorat en informatique. Toulouse: Université de Toulouse 3, 2005, 272p. Format PDF. Disponible sur: <http://www.irit.fr/~Romain.Laborde/memoires/TheseRomainLaborde.pdf>
(Consulté le 18/05/2017).

[6] Anonyme : <http://faculty.wiu.edu/Y-Kim2/OSISysNetMagt.pdf>.

[7] Anonyme : <http://ycchen.im.ncnu.edu.tw/smf-paper.pdf>.

[8] karlsson Erik. Evaluation of Linux Security Framework. **[En ligne]** Thèse de Master en informatique, 2010, 47p. Format PDF .Disponible sur :
< http://www8.cs.umu.se/education/examina/Rapporter/ErikKarlsson.pdf>
(Consulté le 18/05/2017).

[9] Shahbaz Khan, Muhammad Amin, Muhammad Nauman et Tamleek Ali. A Comprehensive Analysis of MAC Enhancements for Leveraging Distributed MAC, Proceedings of the International MultiConference of Engineers and Computer Scientists, 19-21 mars 2008, Hong Kong. MECS, 2008, 08 P.

[10] anonyme : <http://www.cyberciti.biz/tips/selinux-vs-apparmor-vs-grsecurity.html>.

[11] Louis Franco Marin. SELinux Policy Management Framework for HIS. **[En ligne]** These de Master en technologies informatiques. Australie : Queensland University of Technology, 2008, 20 p.
Format PDF .Disponible sur : < http://eprints.qut.edu.au/26358/1/luis_Franco_Thesis.pdf>
(Consulté le 18/05/2017).

[12] anonyme : <http://grsecurity.net

[13] Ravier Timothée. Optimisation des performances de SELinux. **[En ligne]** Rapport de stage en sécurité et technologies informatiques. France : Ecole Nationale Supérieurs d'ingénieurs des Bourges, 2012, 58p.
Format PDF .Disponible sur : < http://tim.siosm.fr/downloads/Repport_Optim-SELinux_Timoth%C3%A9e_Ravier_08_2012>
(Consulté le 18/05/2017).

[14] Ranc Daniel, 2004 : Gestion de réseaux et services. Vol 2, p24.

[15]Chitrak Grupta. Adoption of WBEM-based Systems Management, juillet2013, U.S.DELL Technical White Paper, 2013, 13p.

[16] Elisa Bertino et Ravi Sandhu. DATA BASE Security. **[En ligne]** Rapport de stage en informatique .West Lafayette : Center for Education and Research in Information Assurance and Security, Perdue University, 2005,19p.

[17] Pierangela Samarati et Sabrina Capitani, 2001 : Access control : Policies, Models, and Mecanisms. p137-138.

[18] Bone Maboudou. Un environnement pour la modélisation des systèmes de contrôle d'accès. **[En ligne]** Mémoire. Québec : Université de Québec en Outaouais, 2015, 210p.

[19] Richard Haines, 2014 : The SELinux Notebook (4 Edition) pp 19, 25, 59,43-45.

[20] Frank Mayer karl, Mac milan et David Caplan, 2006 : SELinux by Exemple (1 Edition). pp 26,30 ,79.

[21]Mirek Jahoda et *al* ., 2017 :Red Hat Entreprise Linux 7 SELinux User's and Administrator's Administrator's Guide Basic and advanced configuration of Security-Enhanced Linux(SELinux), p10.

[22]Bogan Iakym. Multi level classification in SELinux. **[En ligne]** Thèse de Master en informatique. Tchéquie : Masaryk University, 2012,

[23]Erik Karlsson.Evaluation of Linux Security Framework. **[En ligne]** Thèse de Master en science de lordinateur.2010, 47p.
Format PDF .Disponible sur :
<http://www8.cs.umu.se/education/examina/Rapporter/ErikKarlsson.pdf>
(Consulté le 19/05/2017).

[24]Annie Ibrahim et Micbed, 2009.New Role of Policy-based Management in home Area Network-Concepts ; Contraints and Challenges.

[25]Olivier Festor et Nizar Benyoussef.WBEM. 21 Avril 2000. France.INRIA, 2002, 75p.

[26] Distributed Management Task Force, 2012 : Common Information Model(CIM) Metamodel, DSP0004 (3), p31.

[27] Distributed Management Task Force, 2006 : CIM Query Language Specification, DSP0202(1), p04

[28] Olivier Festor et Nizar Benyoucef.wbem.21 avril 2000, France-INRIA, 2000, 75p.

[29] Taylor et Francis group, 2010 : Advances in Network Management, p43.

[30] Nguyen Man Tuang. Protocoles pour la gestion des réseaux Informatiques. **[En ligne]** Rapport de stage en informatique. France : Institut de la francophonie pour l'informatique, 20051, 45p.

Format PDF .Disponible sur :

<http://repo.zenksecurity.com/Protocoles_reseaux_securisation/Les%20Informatiques.pdf>

(Consulté le 19/05/2017).

[31] Anonyme <http://www.sharadavikas.com/CourceMeterials/Electives/network%20 Informatiques.pdf>

[32] Anonyme <http://docs.oracle.com/cd/E19683-01/806-6827/6jfoa8m7p/index.html>

[33] Anonyme <http://darnok.org/presentations/wbem/Introduction%20to%WBEM.html>

[34] Moris Sloman.Policy Driven Management For distributed Systems. Journal of Network and Systems Management,Plenum Press,vol. 2, No. 4, 1994, pp333-360.

[35] Distributed Management Task Force(DMTF).Common Information Model(CIM) ,30 Aout 2000.DMTF White Paper, 18 Décembre 2000, 54 p.

[36] Distributed Management Tak Force(DMTF), 2012 : Managed Object Format (MOF), DSP0221(3), p06.

[37] Distributed Management Tak Force(DMTF), 2014 : Representation of CIM in XML, DSP0201(2), p09.

[38] Distributed Management Tak Force(DMTF), 2007 : Policy Profile, DSP1003(1), p09

[39] Distributed Management Tak Force(DMTF). Common Information Model(CIM), 18 juin 2003. DMTF White Paper, 18 juin 2003, 20p.

[40] Distributed Management Tak Force(DMTF), 2011 : Integrated Access Control Policy Management Profile, DSP1106(1),p09.