# Sécurisation d'une Passerelle XML - SNMP

Laurent Collet, Vincent Cridlig

# RAPPORT DE STAGE
Année scolaire 2004/2005



Laurent COLLET
ESIAL 2A

# RAPPORT DE STAGE
Année scolaire 2004/2005



Laurent COLLET
ESIAL 2A
ESIAL,
*Campus Scientifique*
*B.P. 239*
*54506 Vandoeuvre-lès-Nancy Cedex*

Vincent CRIDLIG
Doctorant 2ème année
LORIA,
*Technopole de Nancy-Brabois – Campus scientifique*
*615, rue de Jardin Botanique - B.P. 101*
*54600 Villers Les Nancy Cedex*

# AVANT PROPOS

Dans le cadre de ma deuxième année à l'ESIAL, j'ai choisi d'effectuer mon stage dans le laboratoire LORIA. Ce stage a pour but d'affiner le projet professionnel, de faire découvrir un nouvel environnement de travail ainsi que de mettre à profit les connaissances acquises au cours des deux premières années d'études dans l'école.

Je tiens tout particulièrement à remercier Vincent Cridlig, mon maître de stage et ancien élève de l'ESIAL de la promotion 2003, ainsi que Olivier Festor, Directeur de recherche INRIA responsable du projet MADYNE, sans oublier Jean-François Leroy pour l'aide qu'il m'a apportée.

# SOMMAIRE

Note : Vous trouverez entre crochets [ ] les références vers la bibliographie pour les chiffres arabes [1] ou vers les annexes pour les chiffres romains [I].

# I. Introduction

## A. Le laboratoire LORIA

Le LORIA, Laboratoire Lorrain de Recherche en Informatique et ses Applications, est une Unité Mixte de Recherche – UMR 7503 – commune aux organismes suivants :

- Centre National de la Recherche Scientifique (CNRS)
- Institut National Polytechnique de Lorraine (INPL)
- Institut National de Recherche en Informatique et en Automatique (INRIA)
- Université Henri Poincaré, Nancy 1 (UHP, Nancy 1)
- Université Nancy 2

Cette unité, dont la création a été officialisée le 19 décembre 1997, succède ainsi au Centre de Recherche en Informatique de Nancy (CRIN), et aux équipes communes entre celui-ci et l'unité de Recherche INRIA-Lorraine.

Dans le secteur des sciences et technologies de l'information et de la communication, le LORIA possède, à travers plus de vingt équipes de recherches, cent cinquante chercheurs et une centaine de doctorants, des compétences reconnues dans des secteurs en pleine évolution et porteurs de développement économique potentiel. Les activités de ces équipes sont centrées autour de cinq thématiques principales "transversales" sur lesquelles elles développent des recherches fondamentales et appliquées. Bien entendu, des équipes ont des activités dans plusieurs de ces thématiques.

## B. L'équipe MADYNE

Le projet MADYNE vise la conception, la validation et la mise en oeuvre de nouveaux paradigmes et architectures de supervision et de contrôle capables de maîtriser la dynamicité croissante des infrastructures et services de télécommunication et de résister aux facteurs d'échelle induits par l'internet ubiquitaire.

L'équipe développe deux axes orthogonaux. L'axe intitulé Gestion autonome, porte sur l'évolution des paradigmes de supervision afin de définir les fondements et l'infrastructure de base de la gestion autonome. Les travaux développés dans cet axe incluent :

- la modélisation et l'évaluation de performances des infrastructures liées aux activités de supervision.
- l'élaboration de méthodes d'auto-organisation des entités de gestion.
- l'évaluation et la mise en œuvre d'architectures de communication distribuées exploitant le modèle pair-à-pair, le routage applicatif, et de nouvelles approches pour la représentation et l'intégration de l'information de gestion.

L'axe Aires fonctionnelles contribue à étendre ces fondements au travers

de trois de ces activités de base :

- la sécurité : nouveaux protocoles de distribution de clefs et infrastructures pour le respect de l'anonymat et la vie privée.
- la configuration et la provision de services : l'automatisation des processus allant de la souscription au déploiement et à l'activation.
- la mesure et l'analyse des performances de bout-en-bout des services supervisés : l'instrumentation automatique, le paramétrage, le monitorage et les modèles de mesure adaptés.

L'Internet nouvelle génération est le domaine d'application principal des résultats des axes précédents. Son architecture ainsi que les services qui s'y déploient offrent toutes les caractéristiques de dynamicité et de besoin des passages à l'échelle que l'équipe aborde dans les autres axes du projet.

# II. Présentation du stage

## A. Conditions de travail

Il y a deux aspects en ce qui concerne les conditions de travail, tout d'abord, aucune contrainte horaire ne m'a été imposée, j'avais la possibilité de choisir afin de travailler dans les meilleures conditions possibles : j'ai donc pris pour habitude de faire 9h – 12h30 le matin et 13h – 17h30 l'après-midi, du lundi au vendredi.

D'autre part, en ce qui concerne le matériel, j'ai eu en ma possession deux ordinateurs fonctionnant sous Linux Debian Testing, avec accès à Internet. L'ensemble du développement à été fait avec GNU Emacs 21.3.1, Tomcat 5 et Net SNMP 5.2. J'ai réalisé la modélisation UML avec Umbrello 1.2.2, les tests avec Mozilla Firefox 0.8, le rapport à été fait grâce au logiciel OpenOffice.org 1.1.2 et enfin les schémas à l'aide de Gimp 2.0.

J'ai également pu profiter de la chance d'être dans un laboratoire de recherche en assistant à diverses conférences:

- « Multi-scale Monitoring of Complex Systems » par J.P. Martin-Flatin, chercheur au CNRS de Genève.
- « A Common Information Model extension for peer-to-peer network and service management » par G. Doyen membre de l'équipe de recherche.
- « RADIUS-BASED SNMP authorization » par J.F Leroy stagiaire

## B. Sujet du stage

Dans de nombreux domaines de l'informatique, XML (eXtensible Markup Language) s'impose comme le format de référence pour construire des applications inter opérables. Ce phénomène est notamment observé dans le cadre de la gestion des réseaux où l'utilisation du langage XML est fortement motivé par les nombreux outils existants liés au transport, au traitement et à la représentation de données XML. La transition du protocole actuellement le plus largement répandu (SNMP : Simple Network Management Protocol) vers XML doit se faire en douceur, notamment par l'utilisation de passerelles capables de traduire des requêtes émises par des managers au format XML vers des requêtes au format SNMP transmises aux agents.

Un problème actuellement soulevé par l'utilisation de ces passerelles est l'absence de considérations liées à la sécurité. En effet, SNMP lui-même a fortement évolué dans sa version 3 vers une architecture prenant en compte les aspects sécurité, aussi bien au niveau de l'authentification , de l'intégrité et du chiffrement des messages qu'au niveau du contrôle d'accès réalisé au sein de l'agent. Il est clairement souhaitable de conserver ce niveau acquis de sécurité

lorsque l'on met en place des passerelles XML/SNMP. Une grande partie des données manipulées dans le cadre de la gestion des réseaux est sensible car ces données incluent par exemple des mots de passe, des tables de routages ou des règles de pare-feu. Leur manipulation à travers le réseau justifie donc la mise en oeuvre de mécanismes de sécurité.

L'équipe MADYNE a défini une nouvelle architecture prenant en compte les problèmes de sécurité introduits par l'utilisation d'une passerelle [1]. Cette architecture est notamment capable de transposer ses politiques de sécurité sur les agents à la demande et d'établir des communications sécurisées que ce soit entre les managers et la passerelle ou entre cette dernière et les agents.

Mon travail fût de mettre en oeuvre concrètement cette architecture sécurisée en me basant sur la passerelle non sécurisée développée par l'équipe de F. Strauss [2].

### C. Organisation du travail

On peut distinguer 2 parties distinctes dans le sujet :

- Le management des informations relatives au bon fonctionnement de la passerelle (les fichiers de configuration) : le manager RBAC
- Les modifications sur le projet existant : la passerelle sécurisée

Cette analyse m'a permis de faire un échéancier afin de rythmer mes 10 semaines de stage :

- 2 semaines de documentation
- 3 semaines d'implantation du manager RBAC
- 3 semaines d'implantation de la passerelle sécurisée
- 2 semaines de rédaction du rapport et de la documentation inhérente à un tel projet (javadoc, manuel d'installation...)

Suite à quelques problèmes d'implantation, voici le calendrier final qui à permis de rendre le projet à la fin du temps imparti :

- 2 semaines de documentation
- 3 semaines d'implantation du manager RBAC
- 5 semaines d'implantation de la passerelle sécurisée

Je vais donc développer maintenant la façon avec laquelle j'ai réalisé les deux parties du projet, avec pour chacune: ses objectifs, une présentation des technologies employées, les méthodes d'implantation...

# III. Implantation : le manager RBAC

## A. Objectifs

Le manager RBAC a pour rôle de gérer le fichier « rbacPolicy.xml ». Il permet de mettre en place l'architecture RBAC au niveau de la passerelle et d'en configurer l'accès.

La connection au manager doit se faire via un servlet JAVA.

## B. Les nouvelles technologies mises en œuvre

### 1. Le modèle RBAC

Le modèle RBAC (Role Based Access Control) à été défini par le NIST (National Institute of Standards and Technology) [3], Il est constitué d'un ensemble de spécifications définissant une architecture permettant la sécurisation d'applications. Cette architecture est constituée d'un ensemble d'objets, et de liens entre ces objets :

- les *Rôles* sont des noms permettant l'association à des contextes sémantiques (ex: IfAdmin, SystemAdmin ...)
- les *Utilisateurs*
- les *Sessions* sont des associations entre un utilisateur et un ensemble de rôles.
- les *Opérations* sont des exécutions de programmes ou de tâches (ex : changer le mot de passe d'un utilisateur)
- les *Objets* sont des ressources (ex : fichier, périphériques)
- Les *Permissions* permettent d'effectuer des opérations sur des objets.

Vous pouvez voir ci-dessous la façon dont sont disposées les différentes entités RBAC.

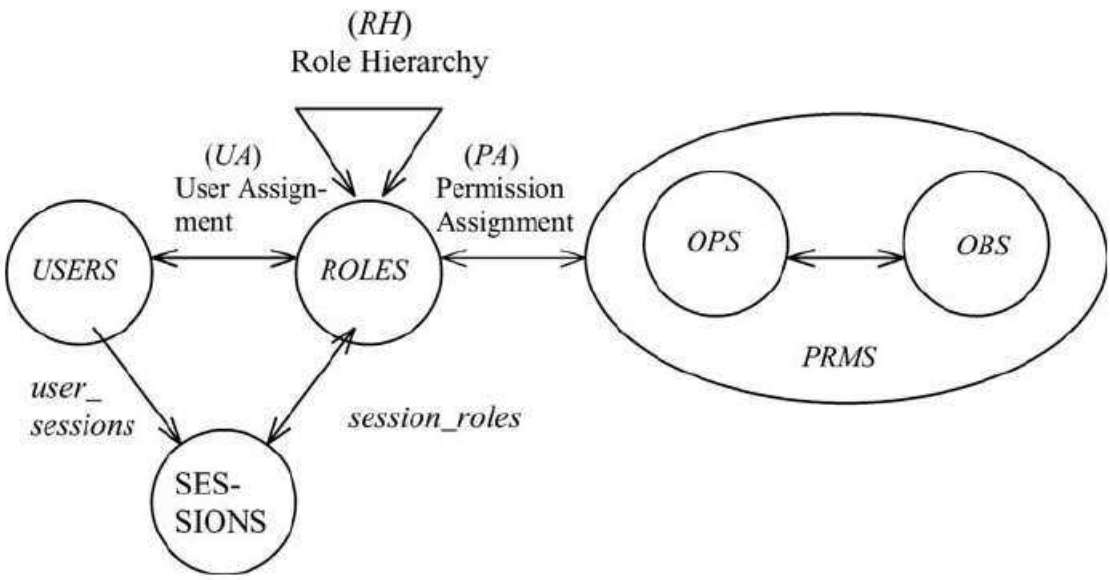


Figure 1 : Schéma du modèle RBAC

Les liaisons entre les différentes entités permettent de définir les domaines d'application de chacune (ex: l'utilisateur Laurent peut endosser le rôle IfAdmin gérant les interfaces, mais pas SystemAdmin qui gère les données système)

## 2. Les servlets JAVA, TOMCAT et ANT

Les servlets sont à la base du projet de passerelle XML – SNMP. D'un point de vue général les servlets sont des applications internet développées « coté serveur », c'est à dire que tous les traitements d'informations sont invisibles pour l'utilisateur qui utilise l'application comme une page HTML ordinaire. Les servlets sont basés sur le langage Java développé par SUN MICROSYSTEM [4].

TOMCAT est un serveur web permettant de faire fonctionner les servlets, il est basé sur le serveur Apache avec une technologie JAVA. Ce serveur est développé au sein du projet JAKARTA de la fondation apache [5].

Quant à ANT, c'est un logiciel permettant le déploiement d'applications. Il est l'équivalent de MAKE (orienté C, C++) pour Java. D'un fonctionnement plus aisé, il utilise un fichier XML [I] permettant de spécifier les différentes cibles. Les arborescences des projets sont standardisées, ce qui permet une plus grande facilité de programmation et de déboguage [6].

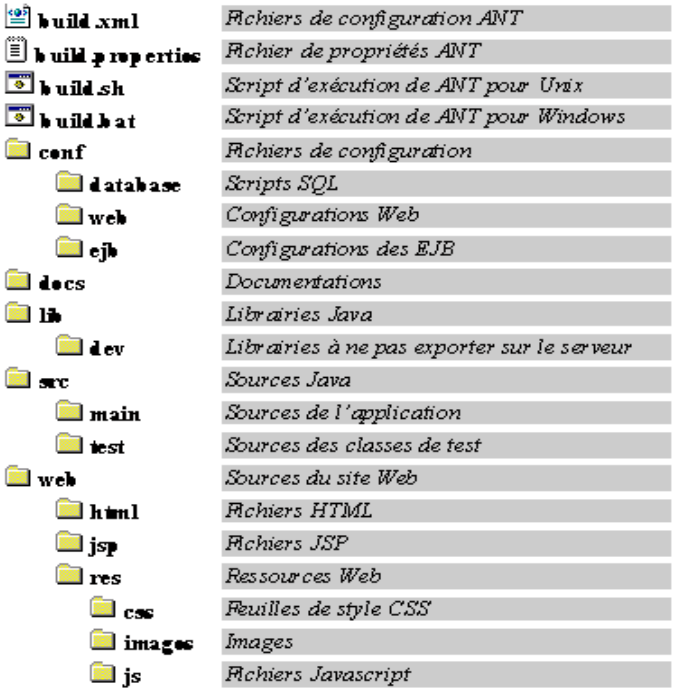


Figure 2 : Exemple d'arborescence avant construction

### 3. XML et schémas XML

XML est un métalangage basé sur le principe d'encapsulation des informations à l'aide de balises (ex : <message>Bonjour</message>). Ses standards sont émis par le World Wide Web Consortium [7].

XML est utilisé dans de plus en plus d'applications, en effet, comme son nom l'indique, ce langage est extensible et il permet donc de faire des normes particulières en fonction des besoins nécessaires pour s'adapter à l'usage voulu.

Les schémas XML permettent de créer ces normes en définissant la syntaxe utilisée pour les fichiers XML. Les fichiers XML Schémas sont eux même des fichiers XML définis par des normes, ce qui permet que l'ensemble (fichier XML + fichier XML Schéma) soit validable par un contrôleur syntaxique.

exemple : ici, exemple.xsd définit donc la syntaxe de exemple.xml

- exemple.xml (fichier XML)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<exemple xmlns:xsd="exemple.xsd>
 <message>Ceci est un exemple</message>
 <affichage>oui</affichage>
</exemple>
```

- exemple.xsd (fichier XML Schéma)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="exemple">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="message" type="xsd:string"/>
<xsd:element name="affichage" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

C. Analyse UML

Afin de mieux appréhender le fonctionnement global de l'application, j'ai commencé par faire une légère analyse UML (Unified Modeling Language), composé de diagrammes d'activités, cas d'utilisation et d'un diagramme de classe [II]. Je vais ici présenter les deux diagrammes d'activités liés au manager RBAC:

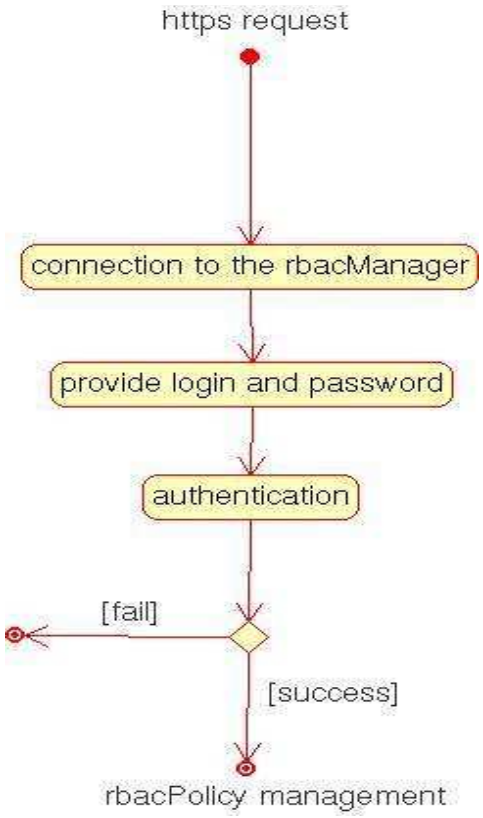


figure 3 : Connection au manager RBAC

L'utilisateur commence à se connecter sur le manager RBAC via un navigateur Internet (ex : Internet Explorer). Le protocole utilisé est HTTPS (Secure Hyper Text Transfer Protocol), il permet de crypter les données transmises grâce à un système de clés de chiffrement.

Ensuite l'utilisateur doit fournir un nom d'utilisateur (« login ») ainsi qu'un mot de passe afin de pouvoir accéder au manager.

Un fois les informations vérifiées, l'utilisateur peut utiliser le manager.

Une fois l'utilisateur connecté, celui-ci peut modifier toutes les informations concernant la configuration de la passerelle, comme le nom des utilisateurs, leurs mots de passe... Il peut même modifier les informations concernant l'accès au manager.

Comme les modifications se font via des formulaires de type HTML (donc statiques), la page doit être rechargée à chaque modification.

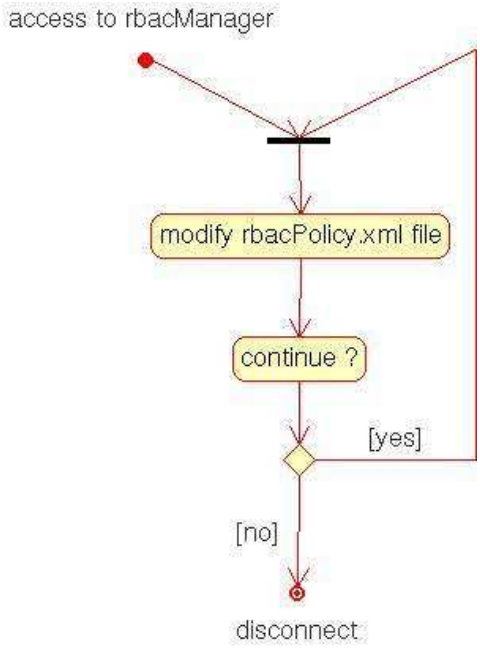


figure 4 : utilisation du manager

D. Étape de développement

1. Création du fichier XML Schéma (rbac-model.xsd)

Afin de créer le schéma XML, je suis parti de l'analyse faite par les membres de l'équipe [1]. On y trouve un exemple de fichier de configuration. J'ai donc modélisé cet exemple grâce à la syntaxe xsd [II].

Après cette première étape, en accord avec mon maître de stage, j'ai rajouté quelques informations supplémentaires :

- un champ permettant de mettre un nom d'utilisateur et un mot de passe afin de s'identifier à la connection sur le manager.

ex :

```
<manager login="laurent" password="jesuisen3a"/>
```

- un ensemble d'informations permettant d'ajouter un peu plus de sécurité en spécifiant des informations propres à chaque agent.

ex :

```
<hosts>
| <host>
| | <hostname>152.81.48.173</hostname>
| | <account-name>laurent</account-name>
| | <password>nopassword</password>
| </host>
| <host>
| | <hostname>chocolat.loria.fr</hostname>
| | <account-name>vincent</account-name>
| | <password>gmlachoucroute</password>
| </host>
</hosts>
```

explication :
Le champ « hostname » permet de spécifier l'adresse internet de l'agent.
Le champ « account-name » permet de spécifier le nom d'un super utilisateur de l'agent SNMP.
Le champ « password » spécifie quant à lui le mot de passe du super utilisateur.

Imaginons un cas où ces informations n'existeraient pas. Le manager a en charge un certain nombre d'agents dans son parc informatique ayant tous le même super-utilisateur avec le même mot de passe. Si un pirate informatique récupère ces informations, tous les ordinateurs du parc sont en danger, alors que si l'on différencie toutes les informations, le pirate ne peut faire que des dégâts réduits à la machine piratée.

2. Récupération des données de rbacPolicy.xml

Le servlet est une interface entre l'utilisateur et le fichier de configuration [IV], pour en modifier les informations, plusieurs technologies sont possibles :

- utiliser un analyseur syntaxique simple qui parcourt l'ensemble

du fichier et s'arrête à chaque mot clef. Cette méthode est simple à mettre en oeuvre, mais elle est n'est pas très puissante : à chaque fois que l'on cherche une information il faut parcourir l'ensemble du fichier.

- utiliser un analyser syntaxique élaboré qui analyse le fichier et le met sous forme d'arbre. Chaque élément devient une feuille de l'arbre. Cette méthode est facile à utiliser, mais les modifications sur l'arbre sont lourdes : c'est un processus très coûteux en temps machine.
- utiliser un des utilitaires de la boîte à outils « Java Web Service Developpement pack ». En effet, le script « xjc.sh » permet de générer à partir d'un fichier xsd une ensemble de classes permettant d'accéder directement aux informations voulues. Cette solution est peux coûteuse, mais elle rend le code difficile à lire : les classes étant générées automatiquement, le code n'est pas optimisé. C'est cette solution qui va être retenue.

ex :
Voici une partie du fichier de configuration :

```
<manager login="laurent" password="jesuisen3a"/>
```

Voici le code retournant la valeur du champ login :

```
public String getManagerLogin(){
      return rbac.getManager().getLogin();
}
```

### 3. Les classes JAVA DataManager et RbacManager

RbacManager est la classe principale de cette première partie, en effet, c'est la classe interface du servlet. Elle va donc servir à gérer les interventions que veut faire l'utilisateur sur le fichier rbacPolicy.xml.

Comme nous l'avons vu précédemment, les informations provenant du fichier rbacPolicy.xml sont transformées en objets. La classe DataManager est donc « chargée » de centraliser toutes ces informations. Cette classe est un singleton, c'est à dire qu'elle n'est créée qu'une seule fois, quelles que soient le nombre d'applications RbacManager qui fonctionnent en même temps sur la machine.

## E. Présentation du résultat

Concernant la présentation du projet, j'ai préféré une interface sobre, clair et précise. Les trois images qui suivent montrent les résultats du manager RBAC. L'image 5 montre la page d'authentification, la 6 et la 7 montrent une partie des formulaires à remplir permettant la configuration de la passerelle.

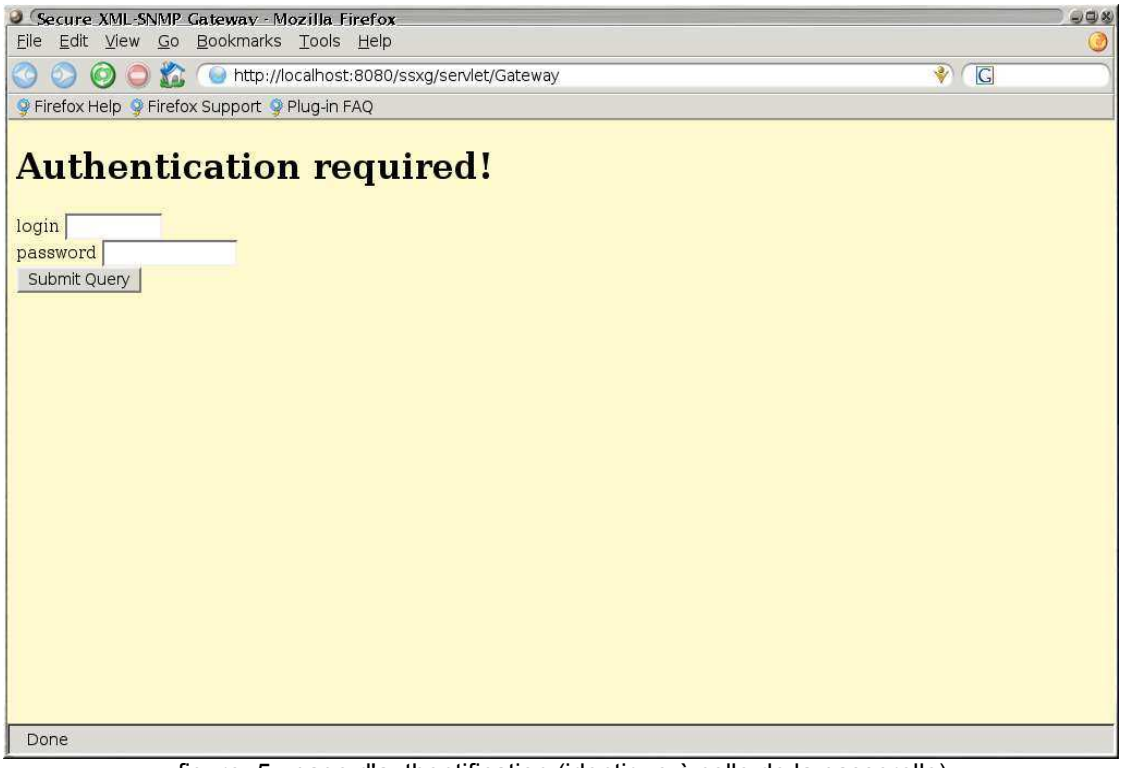


figure 5 : page d'authentification (identique à celle de la passerelle)

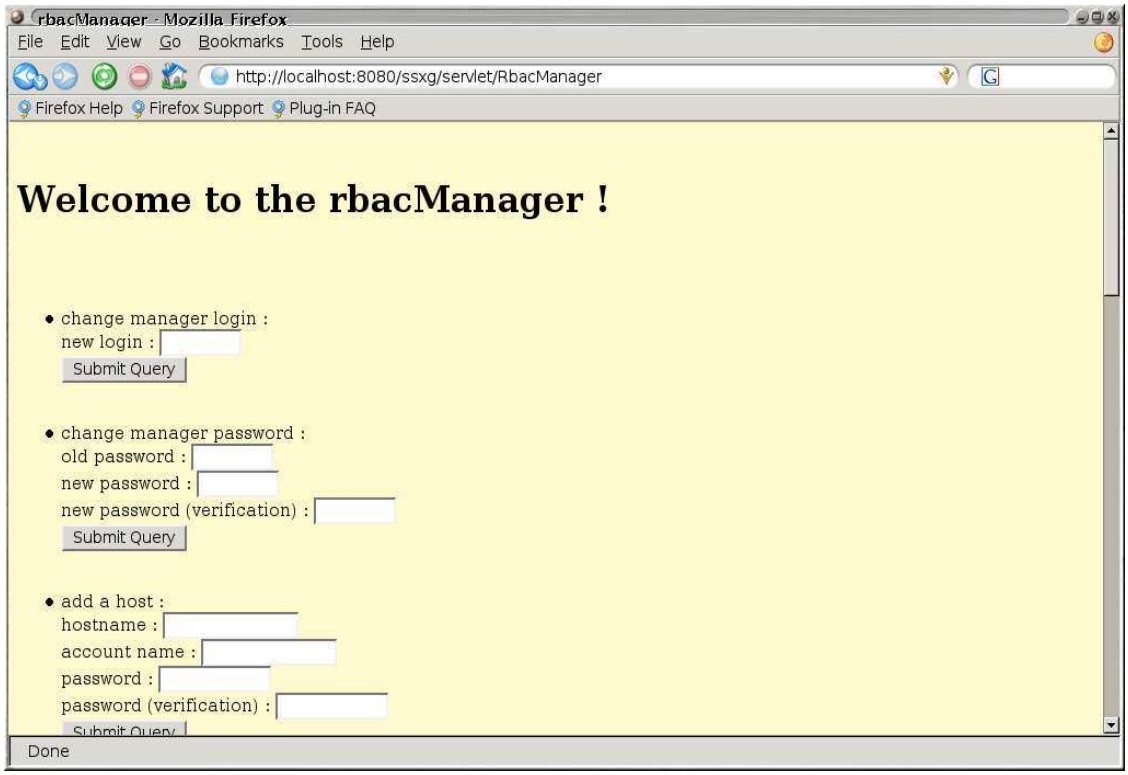


figure 6 : utilisation du manager RBAC

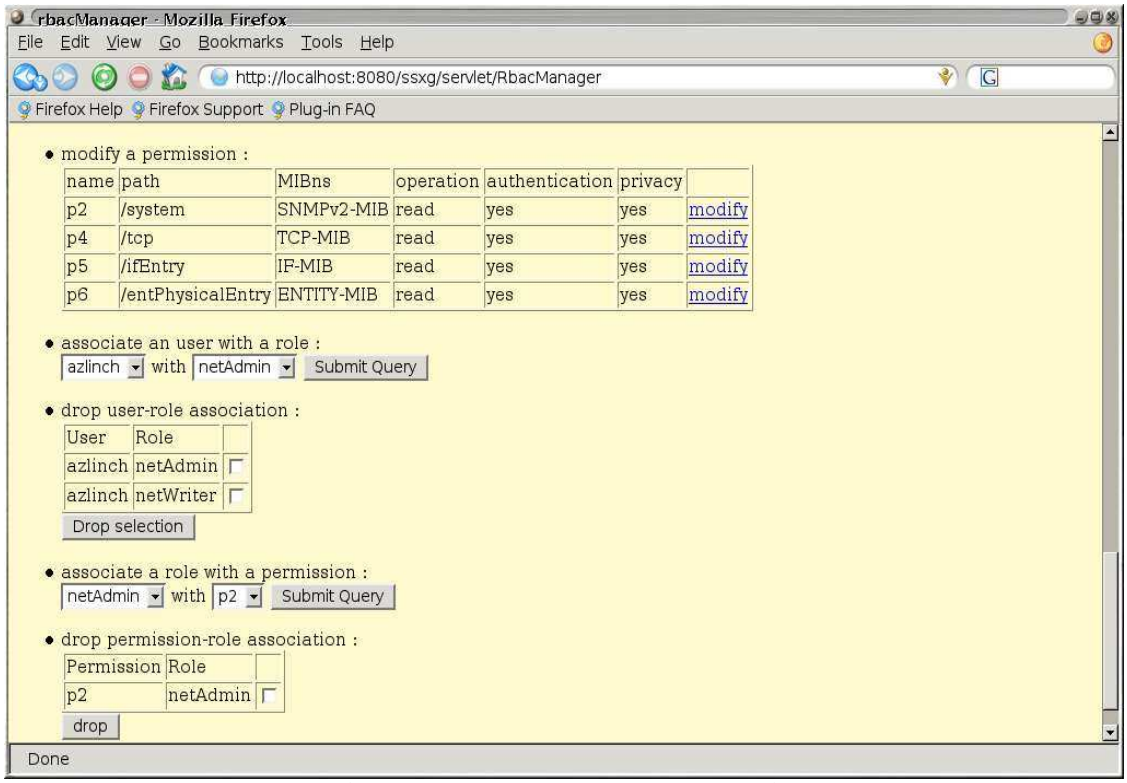

figure 7 : utilisation du manager RBAC

# IV. Implantation : la passerelle sécurisée

## A. Objectifs

La deuxième partie de ce projet est la plus importante, il s'agit de sécuriser la passerelle déjà existante. On peut également diviser cette partie en 2 :

- créer une interface pour l'utilisateur. Précédemment, les requêtes se faisaient via l'URL (Uniform Ressource Locator) du servlet.
- permettre au servlet de faire des requêtes SNMPv3 (authentification requise et cryptage des messages)

## B. Les nouvelles technologies mises en œuvre

### 1. Le protocole SNMP

SNMP (Simple Network Management Protocol) est un protocole permettant le management de ressources informatiques [8]. Il est relativement simple, toutefois l'ensemble de ses fonctionnalités est suffisamment puissant pour permettre la gestion de réseaux hétérogènes complexes (on peut par exemple récupérer le nom d'une machine et le changer, monitorer une machine afin d'anticiper une panne etc. ). Son fonctionnement est celui d'un modèle client/serveur classique.

L'ensemble des informations de l'ordinateur géré par SNMP est contenu dans un arbre appelé la MIB (Management Information Base) : chaque noeud de l'arbre correspond à une information. Ces données sont accessibles grâce à des OID (Object IDentifier) qui représentent le chemin dans l'arborescence de la MIB (exemple d'OID + schéma de la MIB )

Dans sa version 3, SNMP a inclus un certain nombre de nouvelles contraintes liées à des aspects de sécurité :

- le modèle USM (User-Based Security Model) [9] permet entre autre, l'authentification du client via des mots de passe et le cryptage des données échangées.

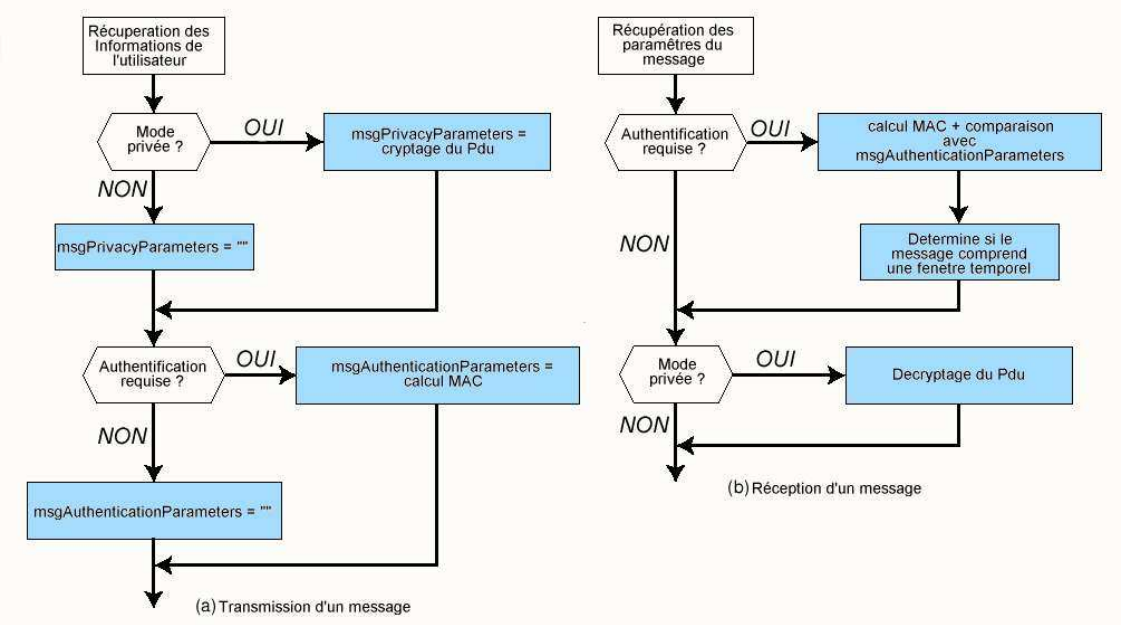


Figure 8 : Traitement des messages USM

- le modèle VACM (View-Based Control Model) [10] permet de limiter le champ des opérations des utilisateurs en vérifiant les droits des utilisateurs sur les objets désignés. Le schéma suivant montre le traitement des informations fait lors des accès à la MIB.

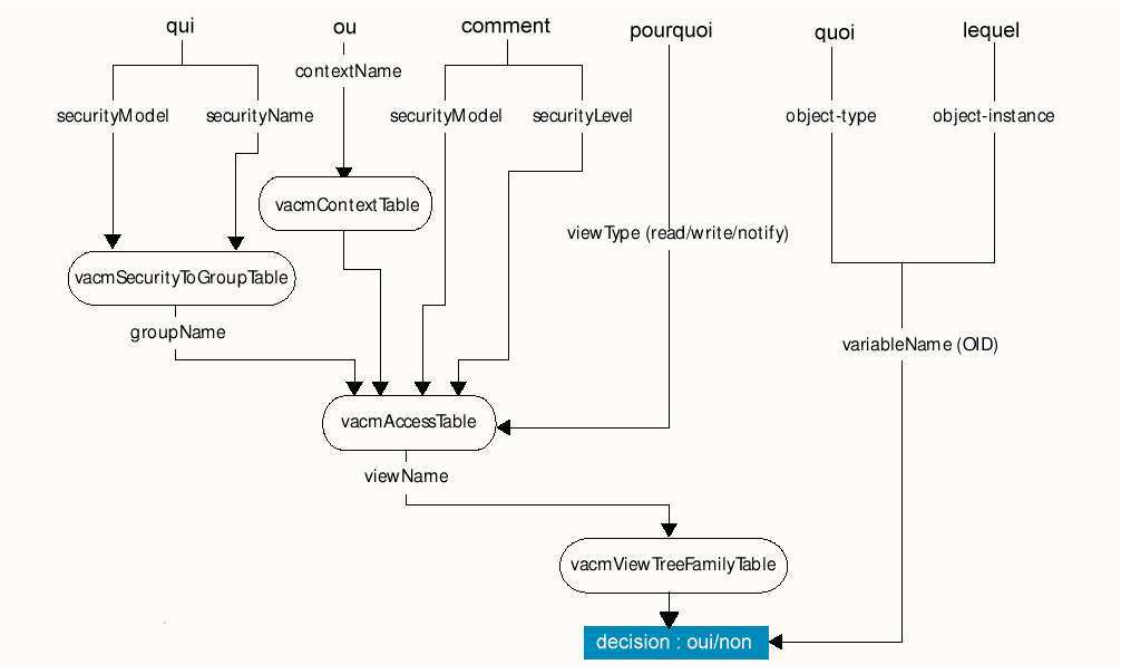


Figure 9 : Enchaînement des informations de la VACM

Par abus de langage, j'associe la VACM et l'USM aux tables qui sont définies dans ces modèles.

Je dois donc intégrer les nouvelles propriétés de SNMPv3 dans la passerelle afin de la rendre plus sûre. Je vais maintenant aborder l'analyse UML afin d'affiner la compréhension du projet.

## C. Analyse UML

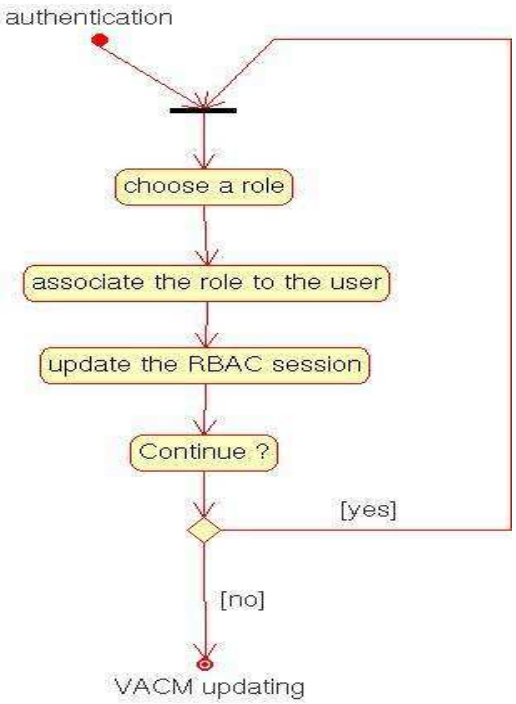


figure 10 : Activation d'un rôle

Le point d'inclusion de ce cas d'utilisation est du même type que celui de la figure 3.

Ici, une fois authentifié, l'utilisateur choisi une série de rôles parmi une liste de rôles possibles (définient par le manager RBAC).

A chaque choix de rôle, la session RBAC est mise à jour. Cette dernière permet à l'utilisateur qui se déconnecte sans réinitialiser ces privilèges de les retrouver lors de la prochaine connection.

Pour éviter un ensemble de traitements qui serait coûteux en terme de temps machine, le plus simple en ce qui concerne la mise a jour de la VACM à été de faire un système qui effaçait toutes les informations concernant l'utilisateur et initialisait ensuite avec les nouvelles valeurs.

Une fois que les agents sont mis à jour l'utilisateur peut utiliser la passerelle.

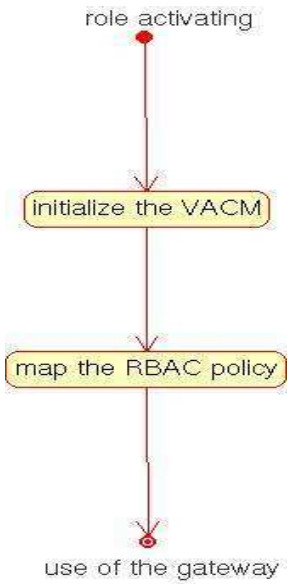

figure 11 : mise à jour de la VACM

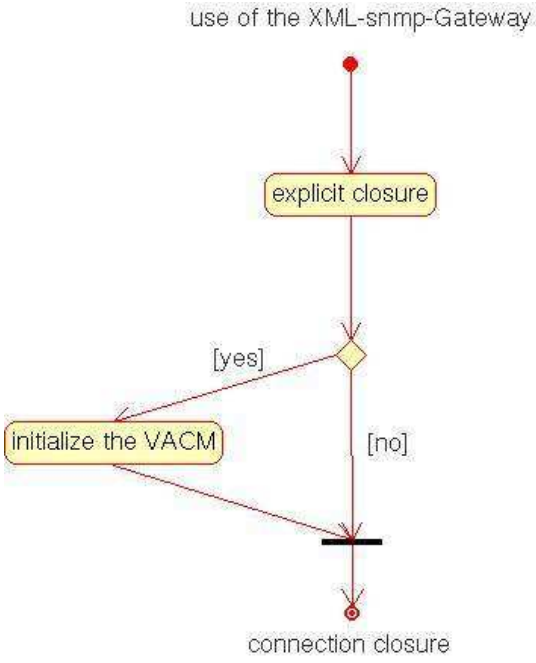

figure 12 : déconnection de la passerelle

Au moment de se déconnecter, l'utilisateur à deux possibilités :
- fermer la fenêtre du navigateur, ce qui implique que les agents ont encore les accès pour l'utilisateur. Ce dernier pourra donc accéder à l'agent via des utilitaire comme snmpwalk ou snmpset ...
- fermer « explicitement » la session, ceci implique que toutes les entrées de l'utilisateur sur les agents sont supprimées. Il n'est donc plus possible pour d'y accéder. La session RBAC est également remise à zéro.

D. Étape de développement

1. Choix d'une librairie JAVA pour le développement SNMP

La première phase du développement fût de choisir une librairie qui puisse convenir à l'ensemble des développements nécessaires. Il en existe actuellement deux :

- SNMP4J [11]
- AdventNet SNMP [12]

Après renseignement sur l'ensemble des deux libraires, il nous a semblé que le développement AdventNet était le plus facile d'utilisation. En effet, AdventNet est une entreprise, elle a donc un site avec un tutoriel, des exemples... alors que SNMP4J dispose juste de la javadoc fournie avec les classes. AdventNet dispose également d'une distribution libre (altéré de quelques possibilité d'utilisation) de sa librairie. Quant à SNMP4J, elle est totalement libre. Mais ce qui à été déterminant, c'est qu'AdventNet à un ensemble de classes utilisant directement une VACM.

2. Passage à SNMP version 3

Afin de ne pas retoucher l'ensemble des classes de la passerelle précédente, j'ai dû développer ma classe SnmpV3Handler.java de la même manière que mon prédécesseur avait fait sa classe SnmpHandler.java, de sorte que pour une utilisation similaire, il suffit juste de donner un nom d'utilisateur et un mot de passe afin d'utiliser les possibilités de SNMP dans sa version 3.

La deuxième étape du passage à SNMPv3 a été de transposer les informations RBAC en entrés dans la VACM et l'USM de l'agent. Pour cette partie qui à été la plus difficile de l'ensemble du projet, j'utilise deux techniques différentes : pour modifier la VACM, j'utilise l'ajout de lignes dans les tables (AccessTable, ViewTreeFamillyTable...), cette technique nécessite les classes « haut niveau » de la librairie. En ce qui concerne la mise à jour de l'USM l'utilisation des classes « haut niveau » était impossible, les tables USM fonctionnant différemment.

E. Problèmes rencontrés

1. Utilisation de la librairie AdventNet

Comme j'ai pu le préciser lors de l'échéancier, j'ai eu de nombreuses difficultés avec l'implantation de la passerelle. En effet, contrairement à ce que je pensais, les documentations sur la librairie AdventNet était relativement peu clair. En effet, si il a été facile de transposer le code SNMPv2 du servlet existant en SNMPv3, toute la partie concernant les modifications à effectuer sur la VACM ou l'USM : ces parties n'étaient pas (ou peu) abordés dans les documentations.

J'ai résolue ces différentes embûches grâce, entre autre, à l'aide de l'équipe technique d'AdventNet, avec qui j'avais pris contact et qui m'ont aiguillé

afin de trouver des solutions.

2. Récupération du code existant

Le second problème que j'ai eu est récurant en informatique, il concerne la réutilisation du code créé par quelqu'un d'autre. La documentation que j'ai récupéré [2] était en allemand, je me suis donc aidé de l'analyse UML afin de comprendre le fonctionnement de la passerelle initiale.

Le code avait été agrémenté de commentaires en anglais ce qui m'a permis de comprendre tout de même les différentes utilisations des classes.

F. Présentation du résultat

Voici 3 images qui montrent le résultat de mon travail sur la passerelle, j'ai laissé de coté la page d'authentification qui est la même que celle du manager RBAC.

La première image est la vue de la passerelle qui permet de faire les manipulations remplaçant les requêtes via URL. On peut y activer (désactiver) des rôles et mettre à jour l'ensemble des agents, se déconnecter, faire des requêtes sur les agents, que ce soient des « get » (récupération de données) ou des « set » (positionnement de données).

Les deuxième et troisième images montrent les résultats d'un « get » sur le chemin « /system ». L'image n°14 ne contient que des valeurs NULL car l'utilisateur n'a pas les droits en lecture sur ce chemin. L'image n°15 montre quant à elle le résultat si les droits sont corrects.

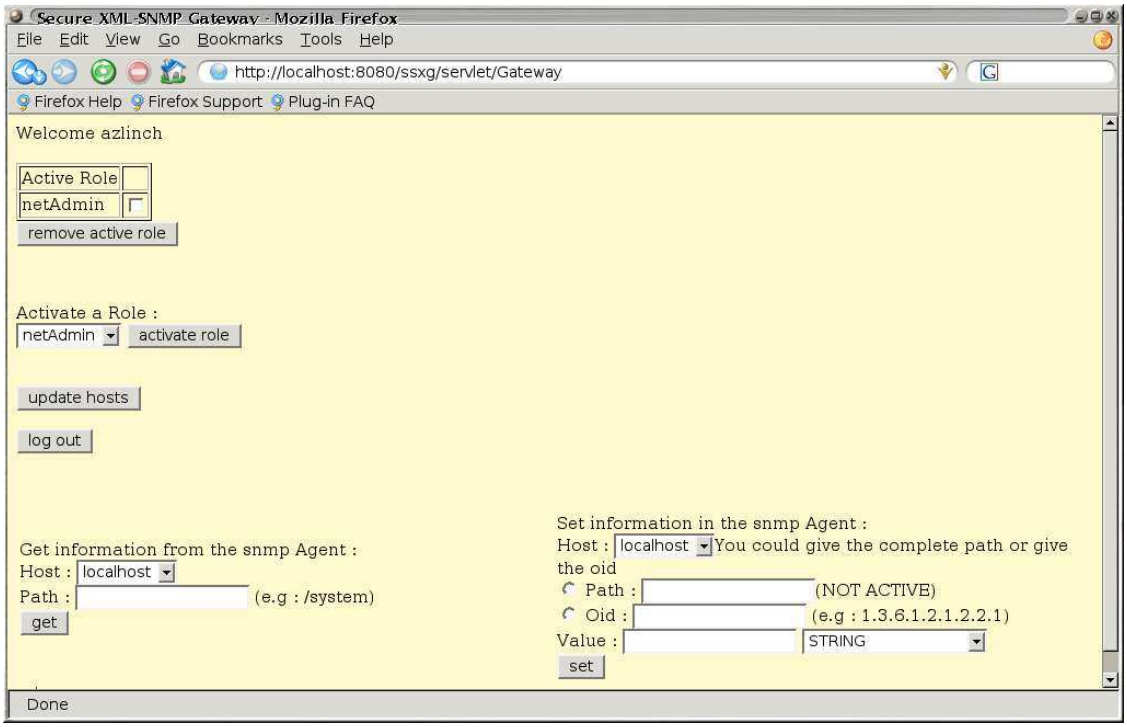


figure 13 : présentation de la passerelle

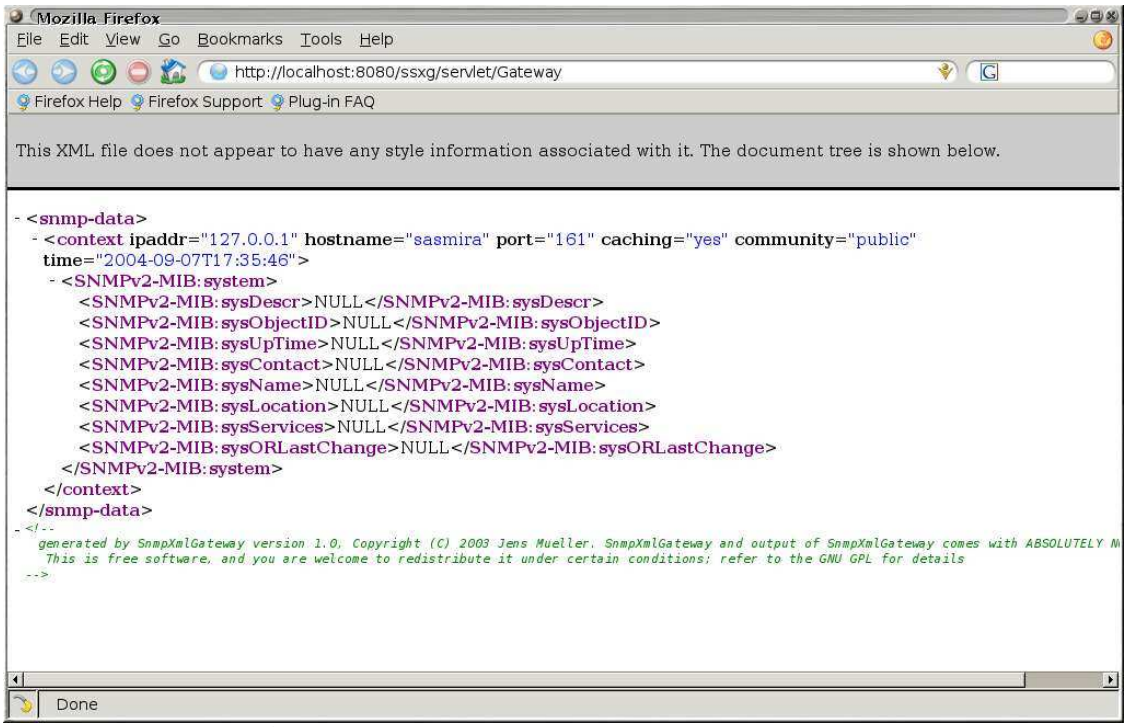


figure 14 : récupération des données sans les droits

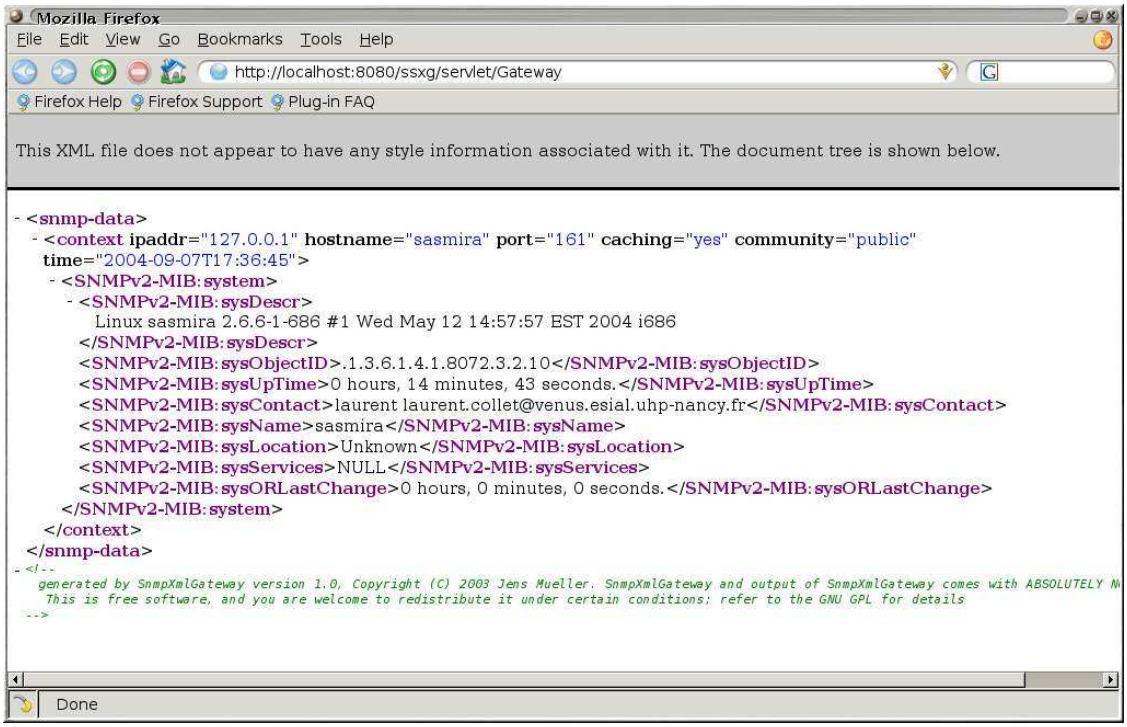

figure 15 : récupération des données avec les droits

# V. Propositions d'extensions du projet

Après avoir exposé mon travail, je vais maintenant parler des extensions possibles afin d'améliorer le projet. Ces évolutions ne sont ni très longues, ni très difficiles à mettre en œuvre, mais je n'ai pu le faire à cause d'un manque de temps.

## A. La gestion des erreurs

Deux types distincts d'erreur ne sont pas gérés :

- Les erreurs de contrôle de mot de passe : je n'ai pas contrôlé les longueurs des mots de passe pour se connecter sur la passerelle. Ces mots de passes vont servir pour la connection avec l'agent SNMP qui réclame un mot de passe de 8 lettres minimum. Un mot de passe trop court est susceptible de poser des problèmes lors de l'ajout d'un utilisateur.
- Les erreurs dues à certaines exceptions : ce problème est plus lié à la présentation et au confort d'utilisation de la passerelle. Lorsqu'une erreur interne au programme survient, une exception JAVA est générée, une page d'erreur avec la pile d'exécution apparaît à l'écran. Un traitement simple avec un message clair serait plus facile à comprendre pour l'utilisateur lambda.

## B. Cryptage des données

Actuellement, les mots de passes utilisateur et administrateur de la passerelle sont dans le fichier de configuration « en clair ». En utilisant les packages de cryptographie livré par SUN, il est possible de crypter ces mots de passe. Ceci permettrait de sécuriser un peu plus le système.

## C. Amélioration du code JAVA - SNMP

Dans la version actuelle de la passerelle, la quasi totalité du code JAVA de la classe SnmpV3Handler.java a été réalisé avec l'API (Application Programming Interface) haut niveau. C'est à dire que j'utilise des classes qui masquent la plupart des étapes dites de bas niveau. L'avantage de cette méthode est la clarté du code, en effet, la programmation est plus facile à faire et à comprendre. Le problème est que cette API nécessite l'utilisation de certaines fonctionnalités de la version non gratuite du package AdventNet. Le passage à l'API bas niveau permettrait d'avoir un code entièrement libre.

# VI. Conclusion

Après 10 semaines de travail, le bilan de mon stage est positif, j'ai terminé une première version de la passerelle XML – SNMP sécurisée. Je pense que l'ensemble de cette période s'est déroulé dans des conditions satisfaisantes et malgré un retard dans l'implantation, j'ai pu terminer à temps mon travail.

Ce stage fut très enrichissant à plusieurs niveaux, il m'a permis d'acquérir de nouvelles connaissances techniques comme les servlets JAVA, XML, et surtout le protocole SNMP. J'ai également appris beaucoup sur un plan personnel en ce qui concerne la démarche de travail et les étapes de développement d'un projet. En effet, dans le cursus scolaire, on ne travail jamais complètement tout seul sur un projet du début à la fin. Cela demande donc beaucoup de méthode et de motivation.

# GLOSSAIRE

API : *Application Programming Interface*

HTML : *Hyper Text Markup Language*

HTTPS *: Secure Hyper Text Transfer Protocol*

NIST : *National Institute of Standards and Technology*

PDU *: Protocol Data Unit*

RBAC : *Role Based Access Control*

SNMP : *Simple Network Management Protocol*

UML : *Unified Modeling Language*

URL *: Uniform Ressource Locator*

USM : *User-Based Security Model*

VACM : *View-Based Access Control Model*

XML : *eXtensible Markup Language*

# BIBLIOGRAPHIE

[1] « Role based access control for XML based management gateways » V. Cridlig, O. Festor, R. State

[2] « Entwurf und Implementierung eines SNMP-XML-Gateways » J. Müller

[3] « Role Based Acces Control »R. Kuhn

[4] http://www.java.sun.com

[5] http://jakarta.apache.org

[6] « Ant : le make de Java » S. François

[7] http://www.w3.org/XML/

[8] « SNMP, SNMPv2, SNMPv3 and RMON 1 and 2 » W. Stallings

[9] RFC 3414 « User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3) »

[10] RFC 3415 « View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP) »

[11] http://www.snmp4j.org/

[12] http://snmp.adventnet.com/

# ANNEXE

## I. Fichier de déploiement (build.xml)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<project name="SecureXmlSnmpGateway" default="deploy" basedir=".">

  <description>
    ant file for version 0.1 of the SecureXmlSnmpGateway
  </description>

  <target name="init">
    <!-- replace the location field to fit with your architecture -->
    <property name="JAVA_HOME" location="/usr/java/j2sdk1.4.2_05"/>
    <property name="CATALINA_HOME" location="/usr/java/jakarta-tomcat-
5.0.25"/>
    <property name="JWSDP_HOME" location="/usr/java/jwsdp-1.4"/>
    <property name="ADVENTNETSNMP_HOME" location="/usr/java/AdventNet"/>
    <property name="DOM4J_HOME" location="/usr/java/dom4j-1.4"/>

    <!-- set global properties for this build -->
    <property name="src" location="src"/>
    <property name="conf" location="conf"/>
    <property name="build" location="build"/>
    <property name="dist" location="dist"/>
    <property name="deploy" location="${CATALINA_HOME}/webapps"/>
    <property name="lib" location="lib"/>
    <property name="doc" location="doc"/>

  </target>

  <path id="classpath">
      <fileset dir="/usr/java/j2sdk1.4.2_05"
includes="jre/lib/endorsed/*.jar" />
      <fileset dir="/usr/java/jwsdp-1.4" includes="jaxb/lib/*.jar" />
      <fileset dir="/usr/java/jwsdp-1.4" includes="jwsdp-
shared/lib/*.jar" />
      <fileset dir="/usr/java/jwsdp-1.4" includes="jaxp/lib/**/*.jar" />
  </path>

  <taskdef name="xjc" classname="com.sun.tools.xjc.XJCTask">
      <classpath refid="classpath" />
  </taskdef>

  <target name="xjc-gen" depends="init">
      <mkdir dir="${build}"/>
      <xjc schema="${conf}/xml/rbac-model.xsd" target="${src}"
package="rbac"/>
      <mkdir dir="${build}/rbac"/>
      <javac srcdir="${src}/rbac" destdir="${build}/rbac">
          <classpath refid="classpath" />
```

```
      </javac>
      <jar destfile="${lib}/rbac.jar" basedir="${build}/rbac"/>
      <delete dir="${src}/rbac"/>
      <delete dir="${build}/rbac"/>
  </target>

  <!-- Compile all java classes -->
  <target name="compile" depends="xjc-gen" description="compile the
source " >
      <javac srcdir="${src}" destdir="${build}">
      <classpath><pathelement location="${JWSDP_HOME}/jaxb/lib/jaxb-
api.jar"/></classpath>
      <classpath><pathelement location="${lib}/rbac.jar"/></classpath>
      <classpath><pathelement location="${CATALINA_HOME}/
common/lib/servlet-api.jar"/></classpath>
      <classpath><pathelement location="lib/jmgmt.jar"/></classpath>
      <classpath><pathelement location="${ADVENTNETSNMP_HOME}/
SNMPAPI/jars/AdventNetSnmp.jar"/></classpath>
      <classpath><pathelement location="${DOM4J_HOME}/
dom4j.jar"/></classpath>
    </javac>
  </target>

  <!-- Create a local copy of the distribution -->
  <target name="dist" depends="compile">
    <mkdir dir="${dist}"/>
    <mkdir dir="${dist}/ssxg"/>
    <mkdir dir="${dist}/ssxg/WEB-INF"/>
    <mkdir dir="${dist}/ssxg/WEB-INF/classes"/>
    <copy todir="${dist}/ssxg/WEB-INF/classes">
      <fileset dir="${build}">
        <include name="**"/>
      </fileset>
    </copy>
    <!-- <copy file="${src}/rbac/jaxb.properties" tofile="${dist}/
ssxg/WEB-INF/classes/rbac/jaxb.properties"/>-->
      <!-- <copy file="${src}/rbac/bgm.ser" tofile="${dist}/ssxg/WEB-
INF/classes/rbac/bgm.ser"/> -->
    <!-- <copy file="conf/web/server.xml" tofile="${catalina_home}/
conf/server.xml"/> -->
    <copy file="conf/xml/web.xml" tofile="${dist}/ssxg/WEB-
INF/web.xml"/>
    <!-- <copy file="src/rbacEditor/rbac.xml" tofile="${dist}/
sxg/rbac.xml"/> -->

    <!-- copy the html files -->
    <mkdir dir="${dist}/ssxg/css"/>
    <copy file="web/html/index.html" todir="${dist}/ssxg"/>
    <copy file="web/res/css/style.css" todir="${dist}/ssxg/css"/>
  </target>

  <!-- Deploy the application in Tomcat -->
  <target name="deploy" depends="dist">
    <mkdir dir="/etc/ssxg"/>
```

```
    <copy file="${conf}/xml/rbacPolicy.xml" todir="/etc/ssxg"/>
    <copy file="${conf}/xml/rbac-model.xsd" todir="/etc/ssxg"/>

    <mkdir dir="/etc/ssxg/mibs"/>
    <copy todir="/etc/ssxg/mibs">
        <fileset dir="${conf}/mibs"/>
    </copy>

    <copy todir="${deploy}">
      <fileset dir="${dist}">
        <include name="**"/>
      </fileset>
    </copy>
  </target>

  <!-- Copy local libraries to catalina lib directory -->
  <target name="library" description="deploy libraries" >
    <copy todir="${CATALINA_HOME}/common/lib">
        <fileset dir="lib"/>
    </copy>
  </target>

  <target name="clean" depends="init" description="clean up" >
    <!-- Delete the ${build} directory trees -->
    <delete dir="${build}"/>
    <delete dir="${dist}"/>
  </target>

  <target name="mrproper" depends="clean" description="clean up" >
      <delete dir="${doc}/api"/>
  </target>



  <!-- Generates javadoc -->
  <target name="javadoc">
    <javadoc destdir="${doc}/api/sxg">
      <fileset dir="${src}/sxg">
        <include name="**"/>
      </fileset>
    </javadoc>
    <javadoc destdir="${doc}/api/rbac">
      <fileset dir="${src}/rbac">
        <include name="*.java"/>
      </fileset>
    </javadoc>
  </target>

</project>
```

# II. Analyse UML du projet

## 1. Diagrammes de cas d'utilisations

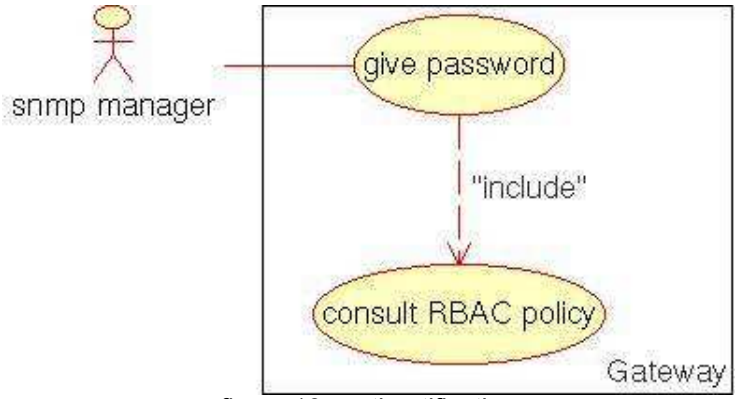


figure 16 : authentification

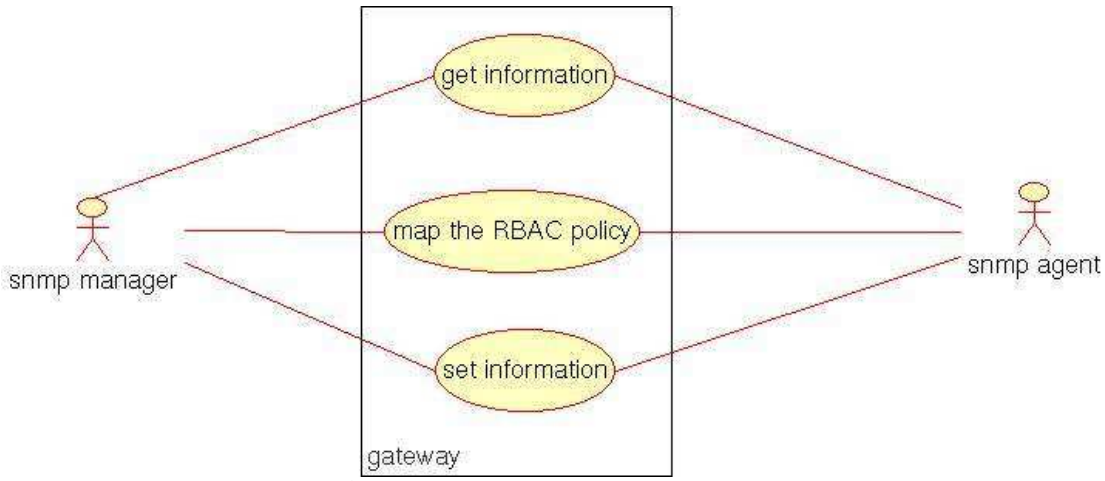


figure 17 : utilisation de la passerelle

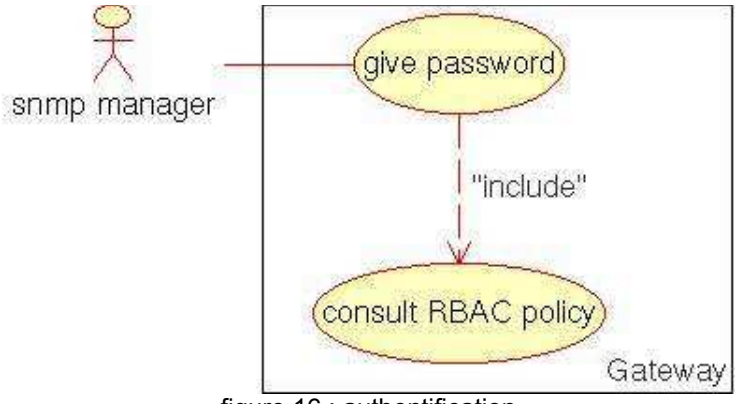

figure 16 : authentification

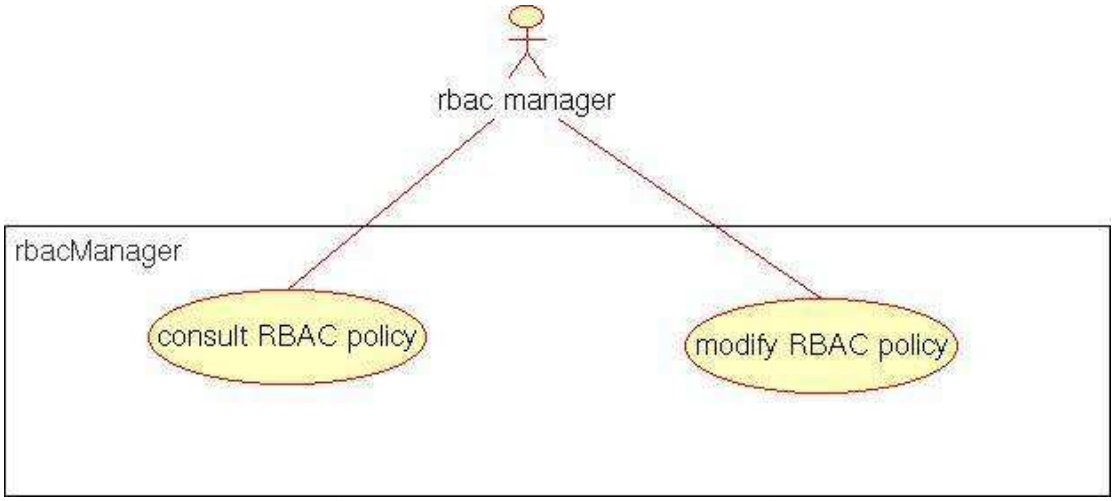

figure 18 : utilisation du manager RBAC

2. Diagrammes d'activités

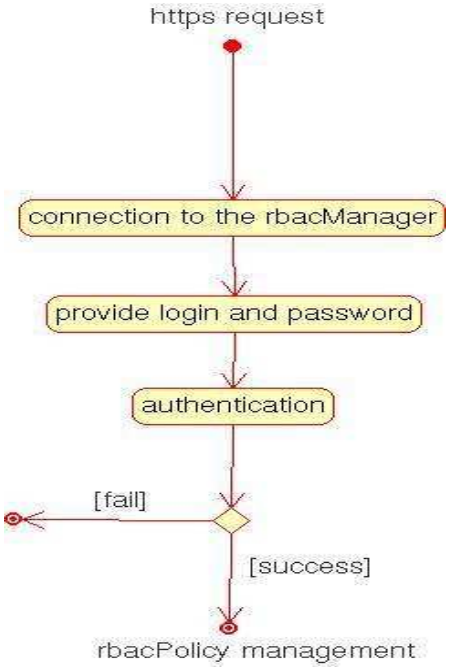


figure 19 : connection au manager RBAC

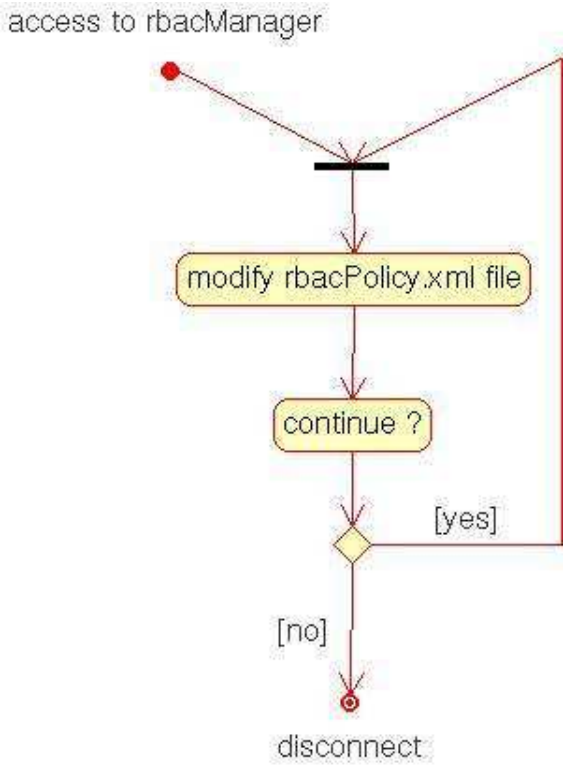


figure 20 : utilisation du manager RBAC

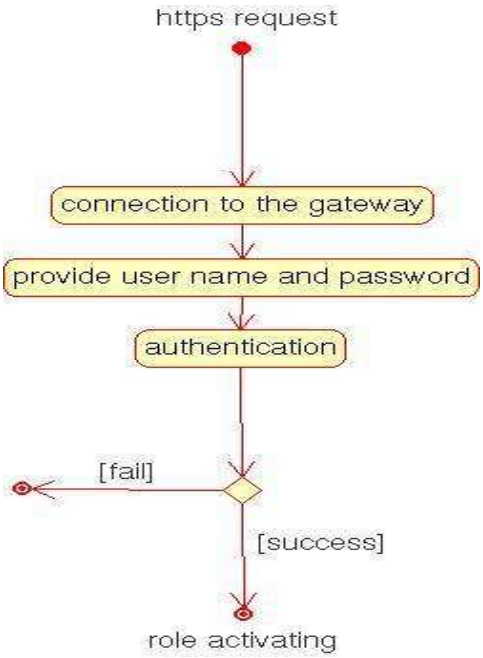

figure 21 : connection a la passerelle

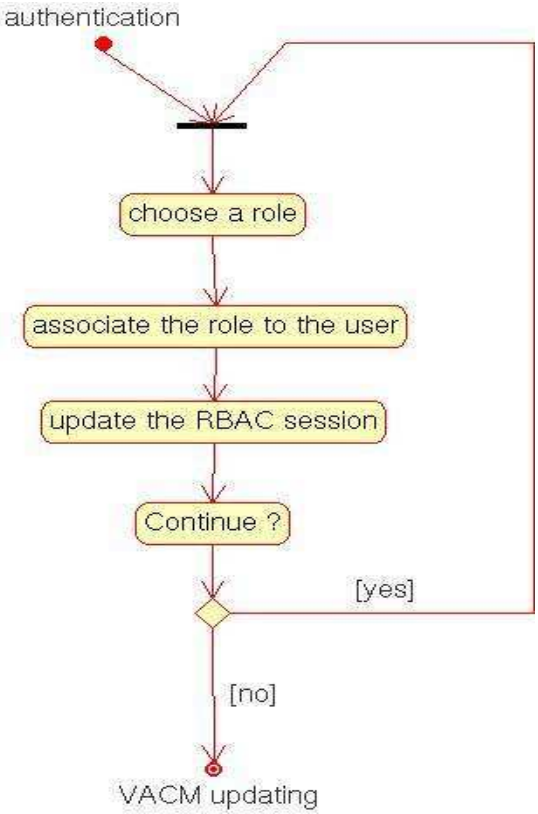

figure 22 : activation d'un rôle

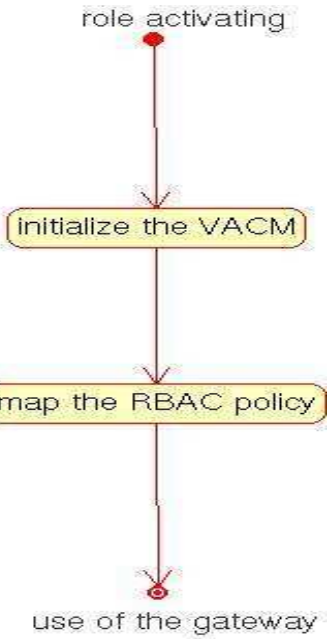

figure 23 : mise à jour de la VACM

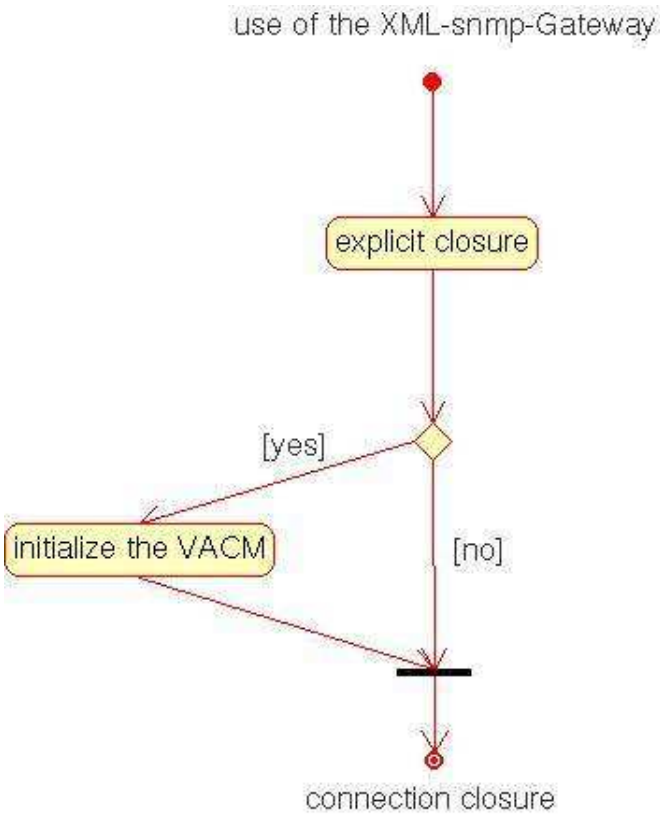

figure 24 : fermeture de la session HTTP

3. Diagramme de classes

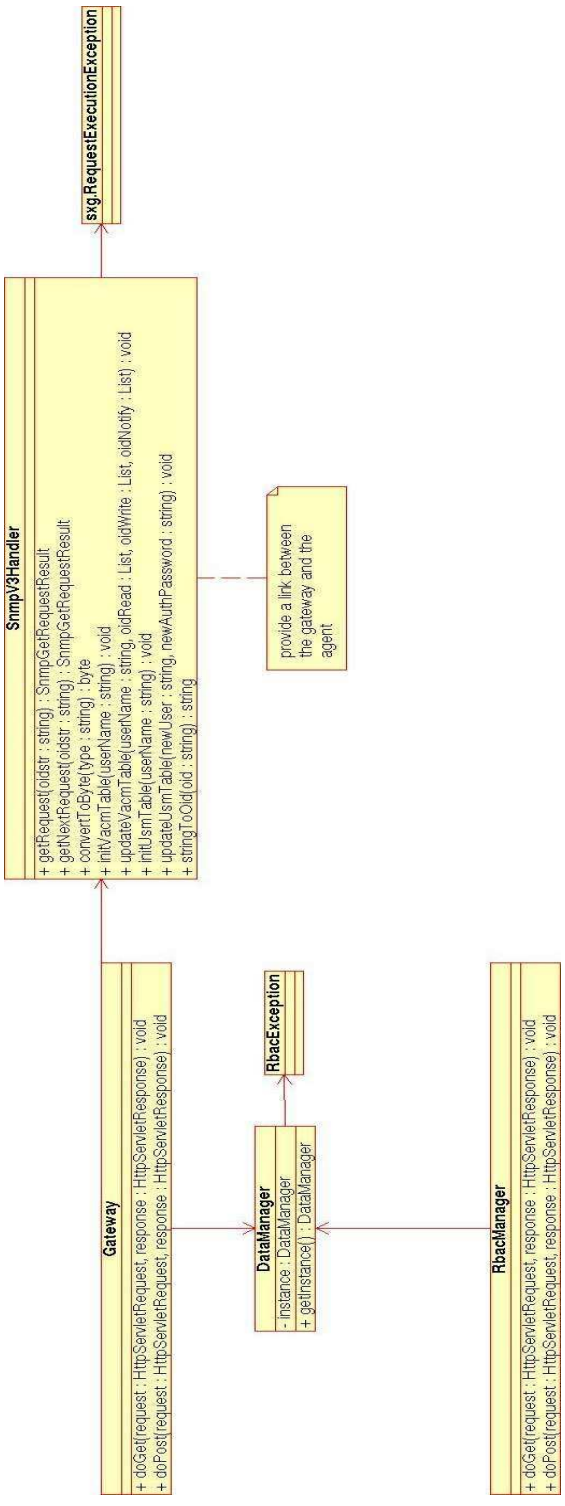


figure 25 : diagramme de classe du projet

# III.Schéma XML du modèle RBAC

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="rbac-model">
    <xsd:complexType>
    <xsd:sequence>
    <xsd:element name="manager" minOccurs="1" maxOccurs="1">
          <xsd:complexType>
          <xsd:attribute name="login" use="required"
type="xsd:string"/>
          <xsd:attribute name="password" use="required"
type="xsd:string"/>
          </xsd:complexType>
    </xsd:element>
    <xsd:element name="hosts" minOccurs="1" maxOccurs="1">
          <xsd:complexType>
          <xsd:sequence>
          <xsd:element name="host" minOccurs="0"
maxOccurs="unbounded">
                <xsd:complexType>
                <xsd:sequence>
                <!-- e.g sasmira.loria.fr or IP address -->
                <xsd:element name="hostname" type="xsd:string"/>
                <xsd:element name="account-name" type="xsd:string"/>
                <xsd:element name="password" type="xsd:string"/>
                </xsd:sequence>
                </xsd:complexType>
          </xsd:element>
          </xsd:sequence>
          </xsd:complexType>
    </xsd:element>
    <xsd:element name="rbac" minOccurs="1" maxOccurs="1">
          <xsd:complexType>
          <xsd:sequence>
          <xsd:element name="users" minOccurs="1" maxOccurs="1">
                <xsd:complexType>
                <xsd:sequence>
                <xsd:element name="user" minOccurs="0"
maxOccurs="unbounded">
                      <xsd:complexType>
                      <xsd:attribute name="id" use="required"
type="xsd:ID"/>
                      <xsd:attribute name="login" use="required"
type="xsd:string"/>
                      <xsd:attribute name="password" use="required"
type="xsd:string"/>
                      </xsd:complexType>
                </xsd:element>
                </xsd:sequence>
                </xsd:complexType>
          </xsd:element>
```

```
          <xsd:element name="roles" minOccurs="1" maxOccurs="1">
                <xsd:complexType>
                <xsd:sequence>
                <xsd:element name="role" minOccurs="0"
maxOccurs="unbounded">
                      <xsd:complexType>
                      <xsd:attribute name="id" use="required"
type="xsd:ID"/>
                      <xsd:attribute name="name" use="required"
type="xsd:string"/>
                      </xsd:complexType>
                </xsd:element>
                </xsd:sequence>
                </xsd:complexType>
          </xsd:element>
          <xsd:element name="scopes" minOccurs="1" maxOccurs="1">
                <xsd:complexType>
                <xsd:sequence>
                <xsd:element name="scope" minOccurs="0"
maxOccurs="unbounded">
                      <xsd:complexType>
                      <xsd:attribute name="id" use="required"
type="xsd:ID"/>
                      <xsd:attribute name="path" use="required"
type="xsd:string"/>
                      <xsd:attribute name="mibns" use="required"
type="xsd:string"/>
                      </xsd:complexType>
                </xsd:element>
                </xsd:sequence>
                </xsd:complexType>
          </xsd:element>
          <xsd:element name="permissions" minOccurs="1" maxOccurs="1">
                <xsd:complexType>
                <xsd:sequence>
                <xsd:element name="permission" minOccurs="0"
maxOccurs="unbounded">
                      <xsd:complexType>
                      <xsd:attribute name="id" use="required"
type="xsd:ID"/>
                      <xsd:attribute name="scopeRef" use="required"
type="xsd:string"/>
                      <xsd:attribute name="operation" use="required"
type="choice"/>
                      <xsd:attribute name="auth" type="yes-no"/>
                      <xsd:attribute name="priv" type="yes-no"/>
                      </xsd:complexType>
                </xsd:element>
                </xsd:sequence>
                </xsd:complexType>
          </xsd:element>
          <xsd:element name="UAs" minOccurs="1" maxOccurs="1">
                <xsd:complexType>
                <xsd:sequence>
```

```
                <xsd:element name="UA" minOccurs="0"
maxOccurs="unbounded">
                      <xsd:complexType>
                      <xsd:attribute name="id" use="required"
type="xsd:ID"/>
                      <!-- type="xsd:IDREF" isn't well generated by
xjc so it became xsd:string -->
                      <xsd:attribute name="userRef" use="required"
type="xsd:string"/>
                      <xsd:attribute name="roleRef" use="required"
type="xsd:string"/>
                      </xsd:complexType>
                </xsd:element>
                </xsd:sequence>
                </xsd:complexType>
          </xsd:element>
          <xsd:element name="PAs" minOccurs="1" maxOccurs="1">
                <xsd:complexType>
                <xsd:sequence>
                <xsd:element name="PA" minOccurs="0"
maxOccurs="unbounded">
                      <xsd:complexType>
                      <xsd:attribute name="id" use="required"
type="xsd:ID"/>
                      <xsd:attribute name="permissionRef"
use="required" type="xsd:string"/>
                      <xsd:attribute name="roleRef" use="required"
type="xsd:string"/>
                      </xsd:complexType>
                </xsd:element>
                </xsd:sequence>
                </xsd:complexType>
          </xsd:element>
          <xsd:element name="sessions" minOccurs="1" maxOccurs="1">
                <xsd:complexType>
                <xsd:sequence>
                <xsd:element name="session" minOccurs="0"
maxOccurs="unbounded" >
                      <xsd:complexType>
                      <xsd:sequence>
                            <xsd:element name="activeRole"
minOccurs="0" maxOccurs="unbounded">
                            <xsd:complexType>
                            <xsd:attribute name="roleRef"
use="required" type="xsd:string"/>
                            </xsd:complexType>
                            </xsd:element>
                      </xsd:sequence>
                      <xsd:attribute name="id" use="required"
type="xsd:string"/>
                      <xsd:attribute name="userId" use="required"
type="xsd:string"/>
                      </xsd:complexType>
                </xsd:element>
```

```
                </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            </xsd:sequence>
            </xsd:complexType>
    </xsd:element>
    </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:simpleType name="choice">
    <xsd:restriction base="xsd:string">
    <xsd:enumeration value="read"/>
    <xsd:enumeration value="write"/>
    <xsd:enumeration value="notify"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="yes-no">
    <xsd:restriction base="xsd:string">
    <xsd:enumeration value="yes"/>
    <xsd:enumeration value="no"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

# IV. Exemple d'un fichier de configuration

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rbac-model xmlns:xsd="rbac-model.xsd">
<manager login="laurent" password="manager"/>
<hosts>
<host>
<hostname>sasmira.loria.fr</hostname>
<account-name>laurent</account-name>
<password>nopassword</password>
</host>
</hosts>
<rbac>
<users><!-- RBAC users -->
<user id="u1" login="laurent" password="none"/>
<user id="u2" login="jeff" password="noneto"/>
</users>
<roles><!-- RBAC roles -->
<role id="r1" name="sysAdmin"/>
<role id="r2" name="netAdmin"/>
</roles>
<scopes><!-- RBAC scopes -->
<scope id="s1" path="/system" mibns="IF-MIB"/>
<scope id="s2" path="/ifEntry" mibns="IF-MIB"/>
</scopes>
<permissions><!-- RBAC permissions -->
<permission id="p1" scopeRef="s1" operation="read" auth="yes" priv="no"/>
<permission id="p2" scopeRef="s2" operation="read" auth="yes" priv="no"/>
<permission id="p3" scopeRef="s2" operation="write" auth="yes" priv="no"/>
</permissions>
<UAs><!-- RBAC user-role association -->
<UA id="ua1" userRef="u1" roleRef="r1"/>
<UA id="ua2" userRef="u1" roleRef="r2"/>
<UA id="ua3" userRef="u2" roleRef="r2"/>
</UAs>
<PAs><!-- RBAC permission-role association -->
<PA id="pa1" permissionRef="p1" roleRef="r1"/>
</PAs>
<sessions>
</sessions>
</rbac>
</rbac-model>
```