IsIMeD

Rapport de synthèse sur la politique de sécurité

Livrable IsIMeD 4.9

Projet IsIMeD ANR-10-AIRT-07

b<>com
ZAC des Champs Blancs
13 rue Claude Chappe
35510 Cesson-Sévigné

Auteur

Guillaume Gauvrit	Guillaume.Gauvrit@b-com.com

Historique

Version	*Date*	*Description des modifications*
0.1	04 août 2014	Début de rédaction
0.2	19 août 2014	Avant relecture, sans introduction ni conclusion
0.3	21 août 2014	Version complète
1.0	22 août 2014	Version finale
1.0.1	01 septembre 2014	Corrections minimes
1.1	18 septembre 2014	Mise en forme pour livrable

Propriétés du document

Nom du projet	IsIMeD
Titre du document	Rapport de synthèse sur la politique de sécurité
Sous-titre du document	Livrable IsIMeD 4.9
Identifiant	2014001STAGECHC
Auteur	Guillaume Gauvrit
Résumé	Étude et mise en œuvre de politiques de contrôle d'accès
Date de publication	2014-09-18
Version	1.1
Diffusion	Confidentiel
Mots clés	Contrôle d'accès, Politiques de sécurité, Clinique, Distribué, Cloud

Données du document

Editor name	Guillaume Gauvrit
Editor email	guillaume.gauvrit@b-com.com
Editor address data	IRT b<>com, 13 rue Claude Chappe, 35510 Cesson-Sévigné
Delivery date	2014-09-18

Liste de distribution

Version	*Date*	*Destinataires*
1.0	2014-08-21	Yannick Morvan, Sandrine Blazy, Pierre-Alain Fouque, Emmanuel Cordonnier
1.1	2014-09-18	Yannick Morvan

Résumé

Ce rapport présente le travail que j'ai réalisé lors de mon stage à **b com**. Au cours de celui-ci, j'ai été amené à étudier les problématiques légales et techniques de contrôle d'accès au sein d'une application distribuée qui manipule des données médicales. J'ai dû examiner comment intégrer un mécanisme de contrôle d'accès dans un logiciel existant de façon à ce qu'il soit aisément maintenable. Mon travail était composé en deux phases principales. J'ai tout d'abord passé en revue les méthodes et technologies existantes potentiellement utiles pour remplir ma tâche. Ensuite, j'ai conçu une solution et commencé son implémentation en vue de son intégration future dans un logiciel.

Ce stage a constitué une expérience pratique complémentaire de ma formation en sécurité informatique reçue en cours d'année. Il m'a permis de m'intégrer professionnellement dans une équipe et d'associer mes compétences en sécurité des systèmes d'information et en développement logiciel.

Abstract

This report presents the work I have done during my internship at **b com**. During its course I was led to study the legal and technical issues related to access control inside a distributed application handling medical data. I had to look up how to integrate an access control mechanism in an existing software in order for it to be easier to maintain. My work was mainly cut into two phases. Firstly I had to revue existing methods and technologies that could be useful to complete my task. Next, I conceived a solution and started to implement it in the objective to later integrate it in an application.

This internship was an applied experience complementary to the formation in information security I received previously during the year. It enabled me to integrate myself professionally in a team while complementing my information security skills with my software development skills.

TABLE DES MATIÈRES

1 INTRODUCTION

Ce rapport présente le travail que j'ai effectué dans le cadre de mon stage de Master 2 « Sécurité des Systèmes d'Information » de l'ISTIC, Université de Rennes 1. Je l'ai effectué dans l'entreprise **b com** où j'ai étudié le contrôle d'accès dans une application.

La sécurité des systèmes d'information recouvre beaucoup d'aspects ; l'un d'entre eux est le contrôle d'accès. Il s'agit de s'assurer que seuls ceux qui en ont le droit puissent accéder, modifier, supprimer... certaines informations. À cette fonction peut être ajoutée le besoin de surveiller ces accès pour, par exemple, limiter les abus ou à des fins d'audit ultérieur.

Le contrôle d'accès peut être envisagé à différentes échelles, d'un système d'information complet à un simple fichier. Le contrôle d'accès étudié pendant ce stage est circoncis à une application. C'est pourquoi je ne considère pas ici l'accès aux données de l'application par des voies extérieures à celle-ci, comme par exemple directement par le système de fichiers.

Dans ce stage, je considère plusieurs cas d'utilisation qui ont en commun de nécessiter de contrôler l'accès à des données médicales au travers d'une application. La nature de ces informations les rend confidentielles et sensibles et rend indispensable de limiter leur accès aux seules requêtes légitimes. Cependant, l'accès rapide à ces informations peut aussi être d'importance vitale et primer sur les restrictions existantes. En effet, il est préférable de briser la confidentialité de données médicales plutôt que de risquer la vie ou l'intégrité physique d'une personne. C'est pourquoi les systèmes d'information de santé doivent prévoir un système dit de « bris de glace » qui permet de contourner des restrictions d'accès.

Ce conflit entre la protection de la vie privée d'une personne et son besoin de pouvoir bénéficier de soins dans l'urgence complexifie la conception du contrôle d'accès. Mais je trouve que c'est aussi ce qui rend intéressant cette tâche.

Vous trouverez dans ce rapport nombre de termes du jargon en anglais. Puisqu'il est destiné à un lectorat anglophone, je ne les traduirai pas lorsque leur signification est transparente en français ou lorsqu'ils ne présentent pas de difficultés. Ceci afin d'alléger le style et de faciliter la lecture.

Je commencerai ce rapport en présentant le cadre de mon stage, avant de présenter mes travaux et résultats au chapitre suivant. Je terminerai par une synthèse et les perspectives envisagées.

2 LE CADRE

2.1 b com

Mon stage se déroule à **b com**[1], un institut de recherche technologique (IRT) situé à Cesson-Sévigné dans la technopole Rennes Atalante, à Brest et à Lannion. Cet institut fort d'environ 180 personnes impliquées actuellement a été créé en 2012 par la volonté politique de développer l'économie locale dans le domaine des nouvelles technologies.

Il est financé pour l'instant par des fonds publics à hauteur de 200 M€ sur 10 ans issus du grand emprunt[2] et de financements issus de l'ANR pour les projets. À terme, il bénéficiera des royalties sur les brevets. L'investissement privé se fait principalement par la mise à disposition de **b com** d'employés par les membres de l'IRT. Il se positionne comme partenaire en R&D dans ses domaines d'expertise qui sont le multimédia, les réseaux et la e-santé.

L'institut est composée de 9 laboratoires, chacun constitué pour l'instant d'un projet :
— Advanced Media Coding, avec le projet Media Immersif ;
— Immersive Interactions, ImData ;
— Digital Trust and Identity, CIN Contenu ;
— Usage et Acceptabilité, UXPPE ;
— Cloud Computing, INDEED ;
— Network Interfaces, RS-COM ;
— Future Network Architecture, CUBIQ ;
— Augmented HealthCare, GestChir ;
— Connected HealthCare, IsIMeD.

Pour l'instant dans des locaux temporaires, les employés déménageront en décembre 2014 à deux pas de l'emplacement actuel dans des bureaux qui sont aujourd'hui en construction. Le travail s'effectue en *open space*, sur trois étages.

2.2 IsIMeD

Je travaille dans l'équipe du projet IsIMeD[3] dans le labo Connected HealthCare. L'objectif du projet est de « développer une infrastructure, des composants, des services et de tester des usages du "cloud based medical image sharing" ». En pratique, le but est de produire des briques logicielles, innovantes dans la mesure du possible, dans le domaine de l'imagerie médicale connectée et distribuée.

Le projet a été constitué avant que l'équipe du labo ne soit embauchée, dans le souci de pouvoir recruter les personnes les plus à même de pouvoir contribuer au projet. C'est pourquoi elle est principalement formée de personnes spécialisées dans l'imagerie médicale.

1. http://b-com.com
2. Financement de l'institut : http://www.lesechos.fr/13/03/2013/LesEchos/21395-153-ECH_vincent-marcatte-pousse-b-com-a-inventer-l-internet-du-futur.htm
3. IsIMeD : Info-structure d'Imagerie Médicale Distribuée

Bien qu'initialement il était prévu qu'un collaborateur travaille sur des problématiques de sécurité, celui-ci s'est retiré du projet. Ainsi je suis le seul qui ai des compétences en sécurité informatique dans l'équipe. L'équipe spécialisée dans la sécurité étant CIN Contenu.

Elle est composée d'un responsable de laboratoire qui fait aussi office de chef de projet, de deux ingénieurs, deux doctorants et un stagiaire (dont vous lisez le rapport). À cela s'ajoutent les membres mis à disposition chargés de faire de l'intégration ou du transfert technologique, soit un ingénieur à 50%, et trois autres à 10 ou 20% de leur temps.

De multiples tâches doivent être accomplies pour répondre aux objectifs du projet ANR IsIMeD. Parmi celles-ci, deux concernent la sécurité d'informations médicales. J'ai été recruté pour répondre à l'un de ces objectifs, qui sera détaillé au chapitre suivant.

L'ambiance de travail est plutôt agréable au sein de l'équipe. Nous partageons *l'open space* avec les membres de l'équipe chargée de l'infrastructure et du support technique qui nous font profiter de leur bonne humeur. Je pense m'être plutôt bien intégré dans l'équipe et plus généralement dans l'entreprise.

2.3 ETIAM

ETIAM[4] est une entreprise membre du Groupement d'Intérêt Économique (GIE) PME fondateur de **b com** et qui est un des partenaires les plus impliqués dans IsIMeD. ETIAM met à disposition du projet son CTO à hauteur de 10% (voire plus) et un ingénieur à 50%. Elle commercialise des serveurs qui permettent aux centres hospitaliers de pouvoir s'échanger des données médicales de manière sécurisée, tout en respectant les contraintes de sécurité par l'isolation des services hospitaliers.

Mon stage est en partie motivé par le besoin de cette société de faire évoluer leur serveur pour offrir plus de services. Il s'avère que la gestion actuelle du contrôle d'accès dans ce serveur bénéficierait d'être repensée afin d'être plus facilement maintenable et de pouvoir être enrichie lors de l'évolution des services du serveur. Un des objectifs de mon stage est de proposer une solution technique à ce besoin. J'y reviendrai plus en détails au chapitre suivant.

4. http://www.etiam.com

3 LE STAGE

Ce chapitre constitue le cœur de mon rapport. J'y présente les objectifs de mon stage (section 3.1), le travail que j'ai effectué (section 3.2), ce qu'il reste à faire (section 3.3) et enfin l'expérience que j'ai acquise lors de ce stage (section 3.4).

3.1 Description du sujet

Initialement, le sujet de mon stage a été décrit de la manière suivante :

> L'objectif du stage est de développer une application clinique web et des solutions de sécurité spécifiques aux contraintes du domaine clinique. Le travail proposé est le suivant :
>
> 1. Revue de l'état de l'art de la sécurité des systèmes d'informations médicales. Nous évaluerons par exemple le contrôle d'accès comme le *role-based access control* (RBAC), l'*organisation-based access control* (ORBAC) ou le PrivOrBAC. La pertinence de l'*attribute-based encryption* sera également analysée.
> 2. Implémentation initiale de briques de contrôle d'accès.
> 3. Développement d'une application web intégrant les briques logicielles de contrôle d'accès.

Ce sujet a le mérite d'être assez clair et détaillé. De fait, il n'a que peu évolué et je me suis tenu à cette feuille de route. Il a bien fallu à un moment préciser ce que serait une « brique de contrôle d'accès », quelles sont les contraintes cliniques en question et quelle application web est visée ; mais cela s'est fait sans entraves. Seul le chiffrement basé sur les attributs a été mis de côté car il ne répond pas aux problématiques de sécurité envisagées.

Finalement, le titre du sujet de stage a évolué pour donner le titre de ce rapport : « Livrable IsIMeD 4.9 ».

Ce qui est implicite dans cette description, c'est qu'il est nécessaire de définir une architecture logicielle pour pouvoir y insérer des « briques » de contrôle d'accès. Cette architecture devra être compatible avec une architecture logicielle existante.

Besoins de sécurité dans IsIMeD En pratique, mon stage répond à deux objectifs. D'une part, la société ETIAM souhaiterait faire évoluer sa gestion du contrôle d'accès dans son serveur et désire participer au projet IsIMeD sur ce point. D'autre part, les objectifs du projet ANR doivent être remplis. Sur ce deuxième point, l'équipe aimerait quelque peu dépasser ces objectifs en développant un démonstrateur technologique dans lequel ma solution serait intégrée, afin de pouvoir valoriser ses travaux. En effet il n'était prévu qu'un « rapport de synthèse sur la politique de sécurité ».

Contraintes spécifiques du domaine médical Les propriétés de sécurité des données médicales peuvent être classées par ordre d'importance :

1. *La disponibilité*, car dans certains cas la vie d'une personne peut dépendre d'un accès rapide à son dossier médical.
2. *L'intégrité*, car il faut pouvoir avoir confiance dans les données pour les utiliser dans un diagnostic.
3. *La confidentialité*, pour protéger la vie privée des patients.
4. *L'authenticité*, car il faut pouvoir s'assurer que personne n'a falsifié des informations. Bien que ce point peut paraître en théorie fondamental pour répondre au besoin d'intégrité, il est souvent négligé en pratique. Les risques réels de falsification sont en effet considérés trop faibles pour justifier le coût d'une protection contre ces risques.

On note que le besoin de confidentialité est très secondaire derrière le besoin de disponibilité, la vie d'un patient passant avant sa vie privée. On comprend vite le problème que cela pose aux mécanismes de contrôle d'accès, dont le principe est d'assurer la confidentialité en restreignant la disponibilité.

Les mécanismes de contrôle d'accès doivent donc prévoir un mécanisme rapide de contournement de la procédure habituelle dans les cas d'urgence. Ceci afin d'autoriser des actions qui devraient être interdites selon la politique générale de sécurité. Par exemple, un médecin doit pouvoir accéder au dossier médical d'un patient dont il n'est pas le médecin traitant si cette personne est en danger.

Cas d'utilisation Afin de spécifier les différents objectifs de mon stage, nous avons rédigé plusieurs cas d'utilisation :

1. Lister les télédossiers [1] auxquels l'utilisateur (médecin) a accès.
   — En fonction d'un identifiant utilisateur, d'un numéro de télédossier et d'une action (lister).
   — Si un utilisateur est membre de plusieurs communautés (ex. : neuro-radiologie et oncologie), il est possible de lui demander dans le cadre de quelle communauté il fait sa requête.
2. Le propriétaire de données accède à ses données à distance, potentiellement depuis n'importe où.
   — Le modèle de contrôle d'accès est partagé par les différents acteurs (hôpitaux).
   — Les politiques sont à priori différentes entre les différents acteurs.
3. Quelqu'un qui a un accès à des données veut partager son accès à certaines données avec une autre personne.
   — Ce partage d'accès doit être en accord avec la politique d'accès à laquelle les données sont soumises.
   — Il faut pouvoir traiter les cas d'urgence.

La conception du système de contrôle d'accès doit permettre de répondre à ces cas d'utilisation et être intégrable dans le serveur d'ETIAM et le démonstrateur technologique de l'équipe. Elle est présentée en section 3.2.3.

Maintenant que le sujet et les objectifs du stage sont bien définis, je vais pouvoir vous présenter l'ensemble du travail que j'ai réalisé au cours de celui-ci.

1. Télédossier : dossier médical informatisé, à ne pas confondre avec le dossier médical personnel (DMP).

3.2 Travail effectué

Dans cette section d'une taille assez conséquente, je présente le travail que j'ai effectué lors de mon stage dans un ordre plus ou moins chronologique. L'approximation est faite dans le souci de clarifier la présentation.

Je commence par l'état de l'art sur différents aspects de mon sujet. Ensuite je reviendrai sur l'étape de prise en main des outils et les difficultés rencontrées. Puis je présenterai l'architecture retenue pour effectuer le contrôle d'accès. Enfin, je détaillerai la conception des politiques de sécurité.

3.2.1 État de l'art

Commençons par l'état de l'art, étape indispensable pour débroussailler le sujet, comprendre l'existant et les bonnes pratiques du domaine. Je ne détaille pas ici le processus que j'ai suivi pour l'établir ; j'en présente directement les résultats.

Le sujet recouvre plusieurs aspects que j'ai dû étudier. Le contrôle d'accès n'est pas nouveau dans le monde médical, il est même en partie standardisée au États-Unis par l'organisation IHE [2]. Il commence à être imposé en France via la « Politique générale de sécurité des systèmes d'information de santé » (PGSSI-S) [1].

Ensuite, il existe différents moyens de concevoir et d'implémenter le contrôle d'accès. J'ai passé en revue ces méthodes, avant de choisir une manière standardisée et suffisante pour répondre à mes besoins.

Dans le domaine de l'imagerie médicale

Dans le monde de l'imagerie médicale, DICOM est un standard incontournable qui spécifie à la fois un format de fichier et un protocole de transfert de ces fichiers. Il est utilisé par tous les équipementiers. Cependant, il est tellement vaste qu'il n'est, la plupart du temps, qu'implémenté en partie.

J'ai étudié les mécanismes de sécurité de ce standard, pour en évaluer la qualité de manière informelle et rechercher de potentiels mécanismes de contrôle d'accès.

Bien que le format spécifie comment chiffrer et authentifier un document, ce n'est pas une fonctionnalité très répandue. En effet, les données doivent rester accessibles, les chiffrer ne présente d'intérêt qu'en cas de vol du support, ce qui est difficilement envisageable (un scan peut peser 2 Go, je vous laisse imaginer la taille d'un serveur d'images). De plus, le standard suppose que l'environnement local est sécurisé par d'autres méthodes, comme l'isolation physique du matériel et du réseau. Cette couche de sécurité est implémentée en suivant la spécification RFC 5652 « Cryptographic Message Syntax » [2].

Il est possible de protéger les transferts DICOM en utilisant TLS [3]. Le standard propose [4] un profil optionnel pour ce faire, en utilisant TLS 1.0 avec des paramètres prédéfinis [3] qui sont considérés comme faibles [5]. On peut noter que l'utilisation de cette couche de sécurisation est ni recommandée ni déconseillée.

IHE propose aussi deux profils optionnels pour DICOM répondant à des objectifs de sécurité : « Audit Trail and Node Authentication » (ATNA) [6] et « Cross-Enterprise User Assertion » (XUA) [7]. ATNA permet d'authentifier des utilisateurs ou nœuds via une PKI [4] et d'enregistrer un historique de leurs actions. XUA, quant à lui, permet d'utiliser le standard SAML [8] pour partager des informations sur les droits des utilisateurs.

Concernant le contrôle d'accès, l'initiative IHE a publié un livre blanc [9] en 2009. Il propose des recommandations et présente des principes généraux sur le contrôle d'accès

2. Integrating the Healthcare Enterprise
3. TLS_RSA_WITH_AES_128_CBC_SHA ou TLS_RSA_WITH_3DES_EDE_CBC_SHA
4. PKI : *public key infrastructure*, infrastructure à clés publiques, cf. `https://fr.wikipedia.org/wiki/PKI`

en milieu hospitalier. Bien que ce document prône la simplicité, je trouve les méthodes proposées inutilement complexes et généralement imprécises sur les points délicats.

En France, le décret n° 2007-960 du 15 mai 2007 stipule que « les référentiels déterminent les fonctions de sécurité nécessaires à la conservation ou à la transmission des informations médicales en cause et fixant le niveau de sécurité requis pour ces fonctions ». Ces référentiels sont regroupés sous la Politique générale de sécurité des systèmes d'information de santé (PGSSI-S) [1], éditée par l'Agence des Systèmes d'Information Partagés de Santé.

La PGSSI-S rend obligatoire pour les mécanismes de contrôles d'accès d'intégrer un dispositif dit de « bris de glace », en référence aux boîtiers d'alarme incendie protégées par une glace. Ce mécanisme permet en cas d'urgence d'accorder des droits à un professionnel non identifié à priori. L'historique d'utilisation de ce dispositif doit cependant être conservé. Il s'agit aussi d'un mécanisme recommandé par l'IHE. C'est par ce biais qu'est concilié le besoin d'un accès rapide aux informations médicales et le besoin de vie privée du patient.

Cependant, la PGSSI-S va plus loin que l'IHE en autorisant le patient à limiter à sa convenance l'accès à ses informations médicales à tout ou partie des professionnels de santé, y compris au risque d'en bloquer l'accès aux acteurs de santé même en cas d'urgence.

L'IHE recommande de garder un historique de toutes les actions afin de pouvoir trouver le responsable en cas d'erreur, d'abus ou d'outrepassement de ses droits. Avec cette approche très répandue aux États-Unis, l'intérêt du contrôle d'accès logiciel est diminué. Il reste cependant obligatoire en France.

Méthodes de contrôle d'accès

Plusieurs manières de réaliser le contrôle d'accès ont été élaborées au fil des ans, la plus connue étant RBAC. Ces mécanismes peuvent se retrouver à différents niveaux dans un système d'information : lors de l'accès au système, lors de l'accès aux applications, lors de l'exécution de commandes ou lors de l'accès aux ressources. Je ne m'intéresse dans le cadre de ce stage qu'au contrôle d'accès au sein d'applications.

RBAC [10], pour *role-based access control*, permet de gérer les droits en assignant des rôles à des utilisateurs. Ces rôles peuvent, par exemple, être implémentés par des groupes UNIX. Dans ce modèle, les permissions sont accordées à des rôles, plutôt qu'aux utilisateurs ou aux exécutables. C'est un modèle très utilisé car simple à mettre en œuvre et suffisant dans la plupart des cas. Il ne permet cependant pas d'implémenter le mécanisme de bris de glace indispensable dans les applications médicales.

OrBAC [11] et PrivOrBAC [12] sont des modèles de contrôle d'accès développés par Frédéric et Nora Cuppens de l'ENSTB. Ils étendent le modèle RBAC pour introduire des éléments de contexte et permettent d'associer des obligations aux autorisations. PrivOrBAC étend le modèle défini dans OrBAC pour pouvoir l'appliquer dans différentes organisations connectés. Ainsi, le mécanisme de bris de glace peut être introduit grâce à la prise en compte d'informations contextuelles. Cependant, l'apport d'OrBAC semble très limité par rapport au modèle ABAC (présenté plus loin) qui est plus étudié et utilisé. Le modèle théorique OrBAC semble mature, mais à l'opposé son implémentation n'est pas suffisamment stable pour pouvoir être considérée dans notre projet. À titre d'exemple, les langages implémentant le modèle OrBAC sont des sérialisations de la représentation interne des politiques dans le moteur de décision.

RAdAC [13], pour *risk adaptable access control*, est une proposition de méthode de contrôle d'accès qui provient du NIST[5]. Son principe général est de restreindre les droits en fonction d'un niveau d'alerte général. Cette méthode encore récente semble complexe à

5. NIST : National Institute of Standards and Technology, une agence gouvernementale américaine de normalisation.

mettre en œuvre. Par exemple, elle suppose une analyse de risque automatisée qui reposerait sur sur une analyse heuristique à base de logique floue avec des implications légales encore inconnues. Elle semble plus adaptée à un environnement militaire que médical.

CBAC, pour *context-based access control*, est un modèle de contrôle d'accès spécifique aux pare-feux. Il permet de d'ajouter des élément de contexte applicatif dans les règles de filtrage, afin notamment de pouvoir prendre en compte les sessions en cours. Je le mentionne ici uniquement pour référence.

ABAC [14], pour *attribute-based access control*, est un modèle de contrôle d'accès où les règles ne s'applique plus sur des objets, mais sur leurs attributs. Ce modèle considère quatre type d'objets : les sujets (utilisateurs), les actions, les ressources et l'environnement. Un attribut peut être un identifiant, une classification, etc. Cette approche offre une grande souplesse dans l'écriture des politiques de contrôle d'accès, tout en restant simple. Par exemple, un attribut peut représenter un rôle ; ce qui peut permettre d'implémenter des politiques RBAC avec ce modèle. Le contexte peut aussi porter un attribut pour permettre le bris de glace.

Ce dernier modèle est le plus mature qui réponde aux besoins des applications considérées dans mes cas d'utilisation. En effet, il est suffisamment flexible pour pouvoir implémenter des politiques de sécurité complexes et il en existe une implémentation standardisée, XACML. C'est pourquoi j'ai choisi d'utiliser ce modèle pour représenter mes politiques de contrôle d'accès. Un autre atout de XACML est qu'il existe plusieurs moteurs d'évaluation de politiques supportés et maintenus par des entreprises.

Je présente cette implémentation du modèle ABAC dans la section suivante.

Autour de XACML

XACML [15], pour *eXtensible Access Control Markup Language*, est une implémentation du modèle de contrôle d'accès basé sur les attributs. Ce standard consiste principalement en un langage XML qui permet d'écrire des politiques de contrôle d'accès et un protocole qui permet d'interroger un module de décision. Ce protocole est lui aussi basé sur des messages écrits dans un format XML.

Le contrôle d'accès utilisant XACML fonctionne comme suit : lorsqu'une action qui nécessite une autorisation est effectuée, l'application émet une requête (dans le langage XACML) au module de décision. Cette requête est généralement de la forme « l'utilisateur untel peut-il faire telle action sur telle ressource dans tel contexte ? ». Le module de décision analyse alors quelles règles de ses politiques de contrôle d'accès (écrites dans le langage XACML) sont applicables pour cette requête. Ces règles sont exécutées et produisent des résultats parmi *accordé, refusé, indéterminé* ou *non applicable*. Ces résultats sont combinés selon des règles de priorité définies dans les politiques pour fournir une réponse. Celle-ci est transmise en réponse à l'application émettrice de la requête. L'application peut alors continuer l'action en cas de réponse positive.

XACML prévoit aussi d'adjoindre des conseils et des obligations à une autorisation ou un refus. Ceci peut permettre, par exemple, d'autoriser une action sous réserve qu'une trace en soit conservée dans un historique.

Pour qu'un tel mécanisme fonctionne, il faut que les applications et les politiques de contrôle d'accès utilisent un ensemble d'attributs et d'obligations commun. XACML définit une liste réduite d'attributs qu'il est possible d'étendre selon les besoins.

À titre d'illustration, le listing 3.1 page suivante présente une politique écrite en XACML. Cet exemple, issu de la documentation pour la version 2 de XACML, spécifie qu'un patient peut lire son dossier médical. Il est difficile de rédiger une politique plus courte. On note que XACML est extrêmement verbeux, même pour un langage XML. Il vaut donc mieux utiliser un outil pour générer du code XACML plutôt que l'écrire à la main.

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=" urn:oasis:names:tc:xacml:2.0:policy:schema:os http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd" xmlns:md="http://www.med.example.com/schemas/record.xsd" PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:1" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <PolicyDefaults>
    <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-19991116</XPathVersion>
  </PolicyDefaults>
  <Target/>
  <VariableDefinition VariableId="17590034">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:patient-number" DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <AttributeSelector RequestContextPath="//xacml-context:Resource/xacml-context:ResourceContent/md:record/md:patient/md:patient-number/text()" DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Apply>
    </Apply>
  </VariableDefinition>
  <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:1" Effect="Permit">
    <Description>
  A person may read any medical record in the
  http://www.med.example.com/schemas/record.xsd namespace
  for which he or she is the designated patient
    </Description>
    <Target>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
  http://www.med.example.com/schemas/record.xsd
  </AttributeValue>
            <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace" DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ResourceMatch>
          <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
  /md:record
  </AttributeValue>
            <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath" DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
  read
  </AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Condition>
      <VariableReference VariableId="17590034"/>
    </Condition>
  </Rule>
</Policy>
```

Listing 3.1 – Exemple de règle en XACML

En revanche, les requêtes sont beaucoup plus simples et courtes, pour du XML, comme le montre le listing 3.2. Cet exemple montre une demande d'autorisation pour l'utilisateur *Dr Who* qui souhaite *lire* le dossier d'*Amy*.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" CombinedDecision="false"
    ReturnPolicyIdList="false">
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
        </Attribute>
    </Attributes>
    <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Dr Who</AttributeValue>
        </Attribute>
    </Attributes>
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" IncludeInResult="false">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">case.Amy</AttributeValue
    >
        </Attribute>
    </Attributes>
</Request>
```

Listing 3.2 – Exemple de requête en XACML

Une possible réponse à cette requête est présentée dans le listing 3.3. Ici, la requête a été acceptée à la condition de garder une trace de l'action. L'obligation `urn:x-b-com:chc:xacml:obl:log-proof-of-action` est définie dans la politique qui a été utilisée pour évaluer la requête.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    <Result>
        <Decision>Permit</Decision>
        <Status>
            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
        </Status>
        <Obligations>
            <Obligation ObligationId="urn:x-b-com:chc:xacml:obl:log-proof-of-action">
            </Obligation>
        </Obligations>
    </Result>
</Response>
```

Listing 3.3 – Exemple de réponse en XACML

XACML propose aussi une architecture logicielle pour le contrôle d'accès. La figure 3.1 page 15 représente un diagramme du flux de données d'une demande d'autorisation selon cette architecture. Celle-ci reprend des bonnes pratiques de conception d'un système de contrôle d'accès, avec les acronymes idoines.

L'architecture que je présente à la section suivante est fortement inspirée de ce modèle et en réutilise le vocabulaire. Aussi vais-je le détailler ici. À ce propos, je vous recommande de prendre un café avant le lire le paragraphe suivant.

Au préalable, le module *policy access point* (PAP) fournit les politiques de contrôle d'accès au module de décision *policy decision point* (PDP). Ensuite le demandeur d'autorisation fait sa demande au *policy enforcement point* (PEP), le point d'application de la politique, qui la transmet au *context handler*, le gestionnaire de contexte. Ce dernier ajoute optionnellement des informations contextuelles à la requête et la traduit en XACML (cf. listing 3.2)

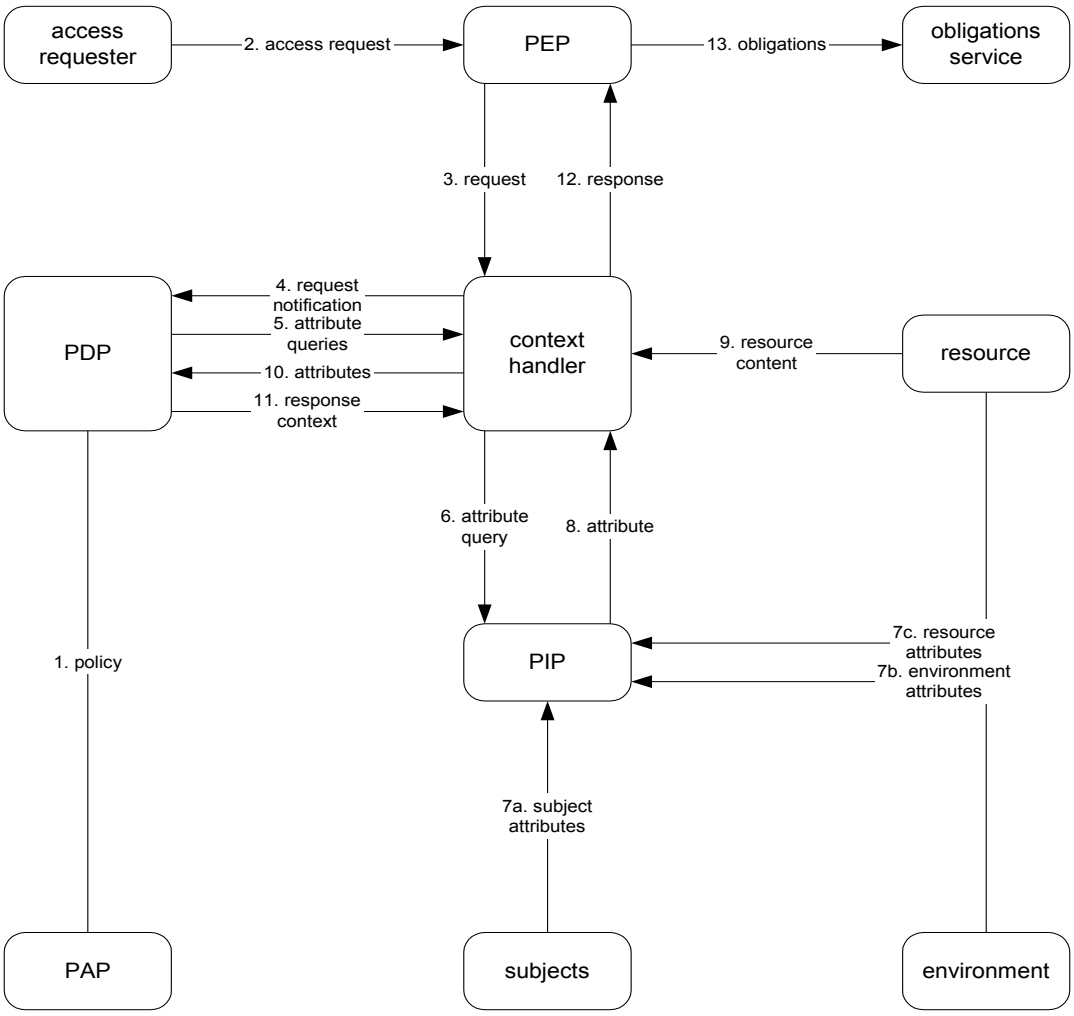

Figure 3.1 – Diagramme de flux de données lors d'une demande d'autorisation selon XACML

avant de l'envoyer au module de décision PDP. Ce module tente d'appliquer ses politiques XACML (cf. listing 3.1) à la requête. Dans le cas probable où celle-ci ne contient pas assez d'informations pour pouvoir être évaluée, comme par exemple la liste des médecins qui suivent le dossier du patient, le PDP demande ces informations complémentaires au *policy information point* (PIP) par l'intermédiaire du *context handler*. Le PIP interroge alors ses diverses bases de données ou sources d'informations pour répondre au PDP. Deux cas se présentent alors : soit le PDP a toutes les informations nécessaire et peut répondre à sa requête par une autorisation ou un refus, soit il lui manque trop d'informations et répond par la valeur *indeterminate* (ce qui signifie en français « je ne sais pas si tu en as le droit et je n'en ai cure, débrouille-toi »). Il se peut aussi qu'aucune politique ne puisse s'appliquer à une requête, auquel cas la réponse est *not applicable* (ce qui cette fois signifie « je ne sais pas si tu en as le droit, demande ailleurs »). Une réponse peut aussi être assortie de recommandations et d'obligations. Cette réponse est traduite par le *context handler* puis transmise au PEP. Fort de sa réponse, le PEP peut effectivement permettre au demandeur initial de continuer son action ou la bloquer. Il se charge aussi de remplir les obligations et généralement d'ignorer les recommandations (qui sont d'ailleurs ignorées sur ce diagramme officiel).

Pour éviter de déléguer au développeur du PEP de choisir que faire en cas de réponse indéterminée ou non-applicable, les différentes politiques peuvent être regroupées au sein d'une politique plus large qui prendra cette décision et ne renverra que des autorisations ou refus.

En environnement distribué

Le contrôle d'accès au sein d'applications distribuées peut se faire soit de manière centrale, par exemple en requérant un ticket authentifié listant les droits d'un utilisateur, soit de manière décentralisée où chaque nœud prend des décisions localement. Le principe est alors de rendre chaque nœud responsable des informations qu'il héberge.

Dans le cas qui nous intéresse, chaque application se trouve dans une communauté différente et est soumise à une politique de sécurité différente. Un centre hospitaliser ne déléguera pas la gestion de sa politique de sécurité à un organisme distant. Aussi, seule une solution décentralisée est envisageable dans notre cas.

Cuppens *et al.* proposent [16] de représenter des organisations extérieures par une unique organisation virtuelle, ce qui permet à chaque organisation de définir deux politiques, une locale et une pour les accès depuis l'organisation virtuelle. Cela évite aux organisations d'avoir une politique pour définir les droits de chaque autre organisation. Ainsi, pour en ensemble de n organisations, on passe d'un total de n^2 politiques à $2n$ politiques. Cependant, les organisations ne peuvent plus appliquer de traitements particuliers selon leurs partenaires.

Le point le plus critique n'est en réalité pas l'écriture des politiques, mais l'identification des utilisateurs extérieurs. Une fois ceux-ci identifiés localement, une politique de contrôle d'accès locale peut leur être appliquée.

Conclusion de l'État de l'art

Cet état de l'art a permis d'approfondir les problématiques spécifiques au domaine médical. J'ai ensuite passé en revue les principales méthodes de contrôle d'accès pour en sélectionner une, ABAC. Suite à quoi j'ai étudié le langage d'écriture de politique le plus utilisé pour réaliser la méthode ABAC : XACML. Ce standard défini aussi un langage et un processus d'interrogation du module de décision.

Une fois le sujet éclairci, il m'a fallu d'une part prendre en main les différents outils à utiliser pour réaliser le contrôle d'accès et d'autre part concevoir une architecture logicielle intégrable avec les produits existants. Chaque chose en son temps, je les présente ici dans les deux sections à venir, dans cet ordre.

3.2.2 Prise en main

Avant d'utiliser les différents outils pour développer une solution de contrôle d'accès, j'ai dû me familiariser avec ceux-ci. Parfois, plusieurs outils pouvaient remplir la même fonction, aussi j'ai dû choisir entre eux. Je présente cette étape de mon stage dans cette section.

La solution de contrôle d'accès que je développe doit pouvoir être intégrée dans un premier temps à l'application Nexus déployée sur les serveur de la société ETIAM. Elle doit cependant être suffisamment générique pour pouvoir être réimplémentée ou adaptée pour le démonstrateur de l'équipe dans un second temps.

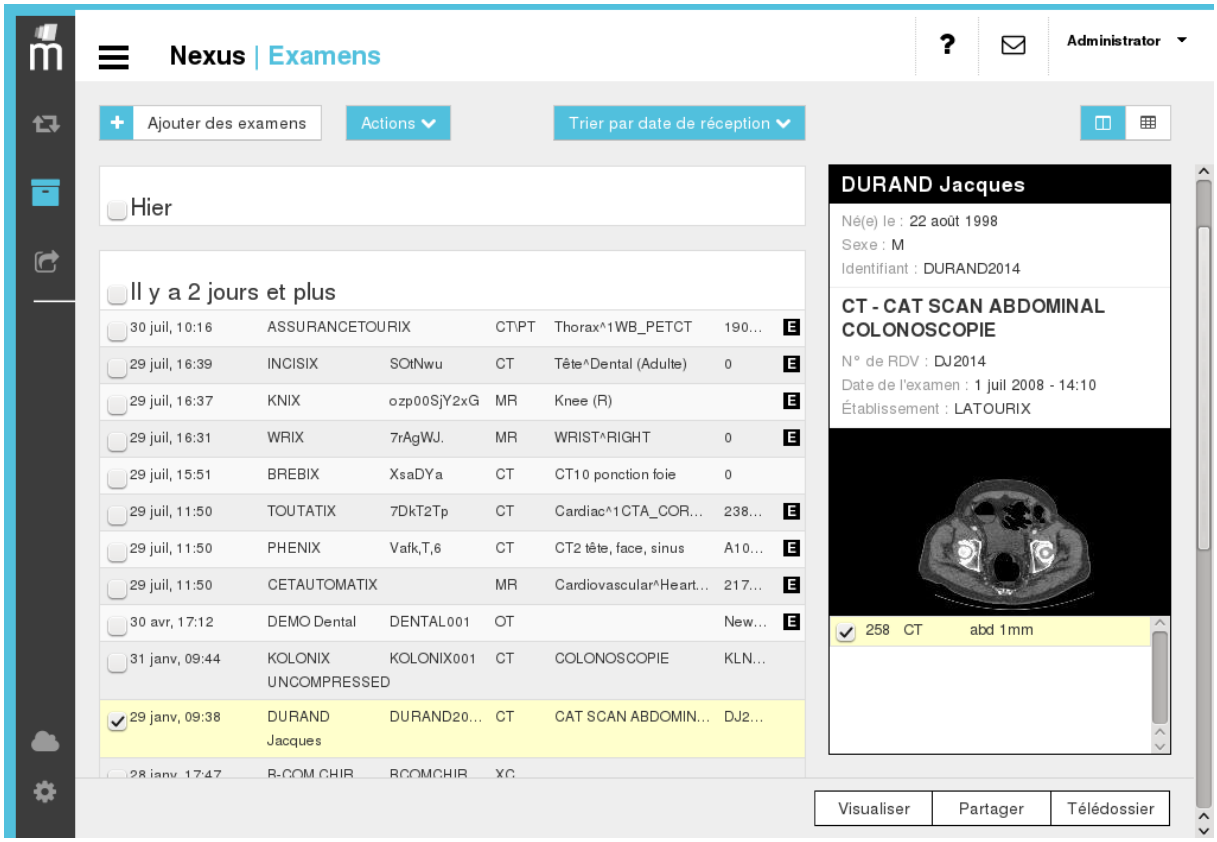

Figure 3.2 – Capture d'écran d'une version de test du logiciel web Nexus

La capture d'écran visible sur la figure 3.2 donne un aperçu de l'application dans laquelle le contrôle d'accès doit être repensé. En effet, à l'heure actuelle le contrôle d'accès est effectué de manière ad-hoc dans Nexus. Cette solution a le mérite d'être très efficace, cependant elle devient peu maintenable.

Sur cette capture, on peut voir au centre une liste d'examens médicaux (ce sont des examens test, ce qui explique certaines dénominations cavalières). L'examen de Jacques Durand y est sélectionné et sa prévisualisation est affichée à droite. En bas à droite, trois boutons permettent de visualiser l'examen avec les différentes coupes du scan, de partager l'examen avec un autre professionnel, ou de l'intégrer à un télédossier, qui est un dossier médical électronique.

Chaque centre hospitalier client d'ETIAM dispose de serveurs Nexus. Ces serveurs sont accessibles au personnel soignant via l'application web accessible uniquement depuis le réseau local. Ces serveurs sont connectés les uns aux autres afin de pouvoir partager des résultats d'examens avec d'autres centres hospitaliers. L'alternative à cette solution est

généralement de graver les résultats d'examens sur un CD dans un centre et les lire dans un autre, ce qui est une opération qui prend un temps non négligeable.

Cette application est développée en PHP selon une architecture à composants. Chaque composant communique avec les autres via des services REST [17] locaux. Toutes les données sont stockées dans une base de données SQL locale. Il est envisagé de transférer les données utilisateur vers un annuaire LDAP [18], ce qui permettrait de les interconnecter et ainsi fusionner ou associer les identités des utilisateurs des différents Nexus. Cela résoudrait aussi le problème de l'identification des utilisateurs distants et permettrait de prendre en compte les identités distantes dans les politiques de contrôle d'accès.

Afin de pouvoir développer et tester une solution de contrôle d'accès, j'ai entrepris de reproduire de manière simplifiée une architecture proche de celle du Nexus. Ainsi, j'ai déployé de manière locale et dans une VM des serveurs LDAP, et en local uniquement un serveur PHP avec sa base de données.

Le serveur PHP me permet de développer des pages de tests et de « simuler » une application Nexus. L'installation et la configuration d'un serveur PHP Apache et de la base de données MySQL associée s'est déroulée sans accrocs. Comme je connais déjà le langage PHP, j'ai pu éviter d'avoir à l'apprendre.

En revanche, prendre en main LDAP a été beaucoup plus compliqué. N'ayant pas d'expérience avec cette technologie, j'ai commencé par suivre des guides d'installation et des tutoriaux. J'ai testé les implémentations Apache Directory Service (en local) et OpenLDAP (dans une VM), qui ont chacune leurs avantages et inconvénients. J'ai aussi tenté de mettre en œuvre l'interconnexion entre ces serveurs LDAP.

OpenLDAP, l'implémentation de loin la plus utilisée du service LDAP, a subi une évolution majeure avec son passage à la version 2.4. Son format de configuration a été entièrement changé pour permettre de la modifier sans avoir à redémarrer le serveur. Cependant, la majeure partie de la documentation pour OpenLDAP, y compris l'officielle, ne prend pas en compte cette évolution. Il devient donc assez ardu de configurer OpenLDAP.

Apache Directory Service a une meilleure documentation officielle, mais il est néanmoins plus difficile de trouver de l'aide sur internet car sa base d'utilisateurs est bien plus réduite.

Après avoir choisi le langage XACML pour exprimer les politiques de contrôle d'accès, il m'a fallu choisir un moteur d'évaluation de politiques. Ce moteur remplit le rôle du *policy decision point* (PDP) présenté précédemment avec la figure 3.1 page 15. La version 3 du standard XACML introduit une fonctionnalité qui serait utile pour le cas d'utilisation n°1 « lister les télédossiers » (présenté en page 9) : la capacité de fusionner plusieurs requêtes en une seule. Cela permettrait de demander en une requête les droits d'accès pour chaque dossier d'un ensemble. J'ai ainsi regardé les moteurs les plus utilisés et ceux qui supportent la troisième version du standard.

XEngine[6] et l'implémentation de Sun[7] de XACML semblent être les plus utilisés mais il n'évoluent plus et ne supportent pas la version 3. Les deux candidats les plus matures sont les implémentations des sociétés Axiomatics[8] et WSO_2[9]. Mon choix s'est porté vers cette dernière car d'une part sa documentation est bien plus fournie et d'autre part elle est disponible sous licence open source (Apache) ce qui permet de la tester immédiatement.

J'ai donc installé sur mon ordinateur WSO_2 Identity Server, qui est un serveur Apache Tomcat pré-configuré avec différents modules, dont un évaluateur de politiques XACML. La communication avec l'évaluateur de politiques se fait par des services web. Sa prise en main a été relativement aisée grâce à sa documentation. J'ai profité de cette période pour approfondir ma connaissance du langage XACML et pour effectuer des tests avec différentes politiques.

6. XEngine : `http://xacmlpdp.sourceforge.net/`
7. Sun's XACML : `http://sunxacml.sourceforge.net/`
8. Axiomatics : `http://www.axiomatics.com/pure-xacml.html`
9. WSO_2 : `http://wso2.com/products/identity-server/`

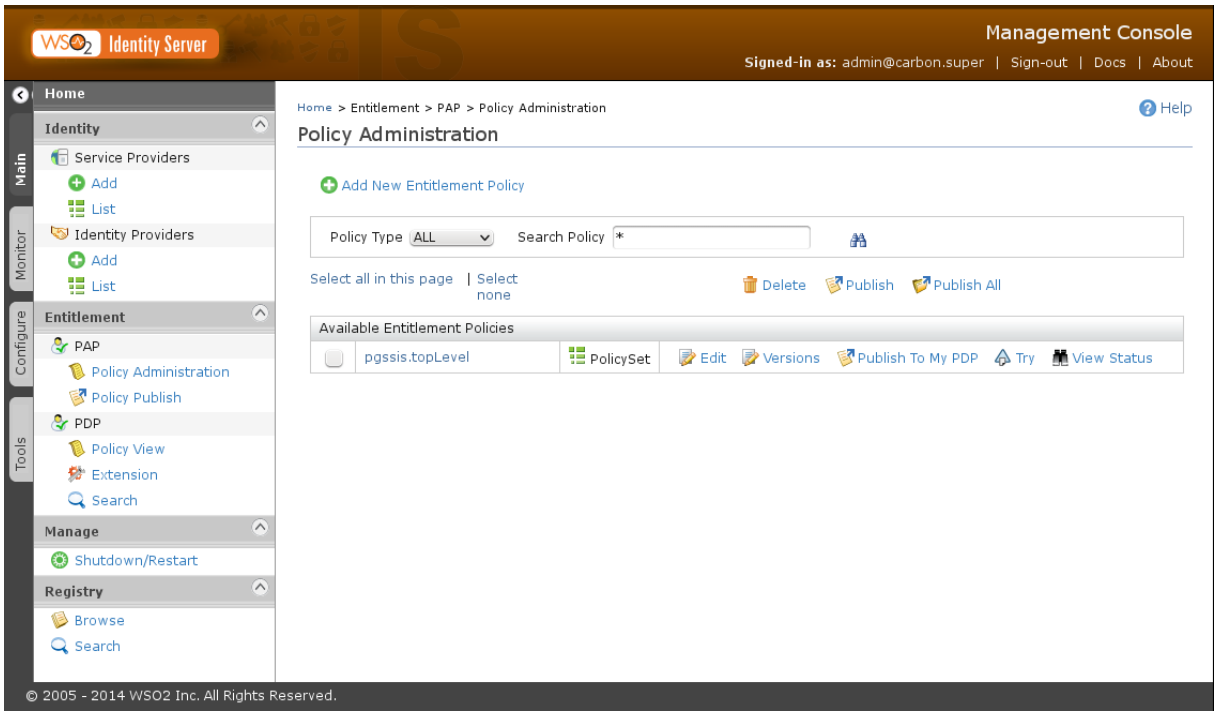

Figure 3.3 – Capture d'écran de l'interface web de WSO_2 Identity Server

La figure 3.3 montre une capture d'écran de l'interface de WSO_2 Identity Server. La page affichée est celle qui permet d'administrer les politiques écrites en XACML. Sur cette capture, une seule politique nommée « pgssis.topLevel » a été ajoutée.

Installer WSO_2 Identity Server sur le serveur d'ETIAM ne pose pas de soucis, un serveur Apache Tomcat étant déjà installé. Il faudra simplement configurer les règles du pare-feu pour que le service ne soit accessible qu'en local. Cette solution a été validée par le CTO d'ETIAM.

De plus, ce logiciel permet d'importer ou de mettre à jour des politiques de sécurité lors de l'exécution, sans avoir à redémarrer un service ou serveur. Cela permet d'éviter une interruption de service le temps d'une éventuelle mise à jour.

Écrire des politiques XACML à la main est une tâche assez pénible de par l'infernale verbosité du langage. WSO_2 Identity Server propose un outil pour écrire des politiques rudimentaires, ce qui est appréciable mais insuffisant pour mes besoins. Heureusement, Axiomatics met à disposition gratuitement un plugin pour l'IDE Eclipse qui permet d'écrire des politiques dans un langage beaucoup plus accessible, ALFA [19]. Ce plugin compile ces politiques ALFA vers du XACML, offre une complétion automatique lors de la rédaction et valide la politique lors de la compilation.

Pour illustrer l'intérêt de cet outil, la section 5.1 en annexe pages 27 à 29 compare une politique écrite en ALFA et sa version compilée en XACML.

La prise en main de ces différents outils — PHP, LDAP, Identity Server, XACML et ALFA — fut une étape indispensable et enrichissante de mon stage.

Tous ces outils m'ont permis d'évaluer la faisabilité en pratique de l'architecture de contrôle d'accès que j'ai proposée et que je présente à la section suivante.

3.2.3 Architecture

Maintenant que les outils nécessaires à l'accomplissement de mes objectifs sont présentés, il devrait être plus aisé de comprendre l'architecture logicielle qui les combine entre eux. Elle est présentée dans cette section.

Ma solution est relativement générique afin de pouvoir l'intégrer dans de multiples logiciels traitant des contenus médicaux. Cependant je la présente ici dans le contexte de son intégration dans le serveur Nexus d'ETIAM pour être plus concret et parce qu'il s'agit de notre objectif premier.

Puisqu'un un bon schéma vaut parfois mieux qu'un long discours, la figure 3.4 montre un diagramme de déploiement qui représente cette architecture logicielle. Il s'agit d'une représentation en « UML simplifié », où j'ai sacrifié un formalisme strict pour une meilleure lisibilité.

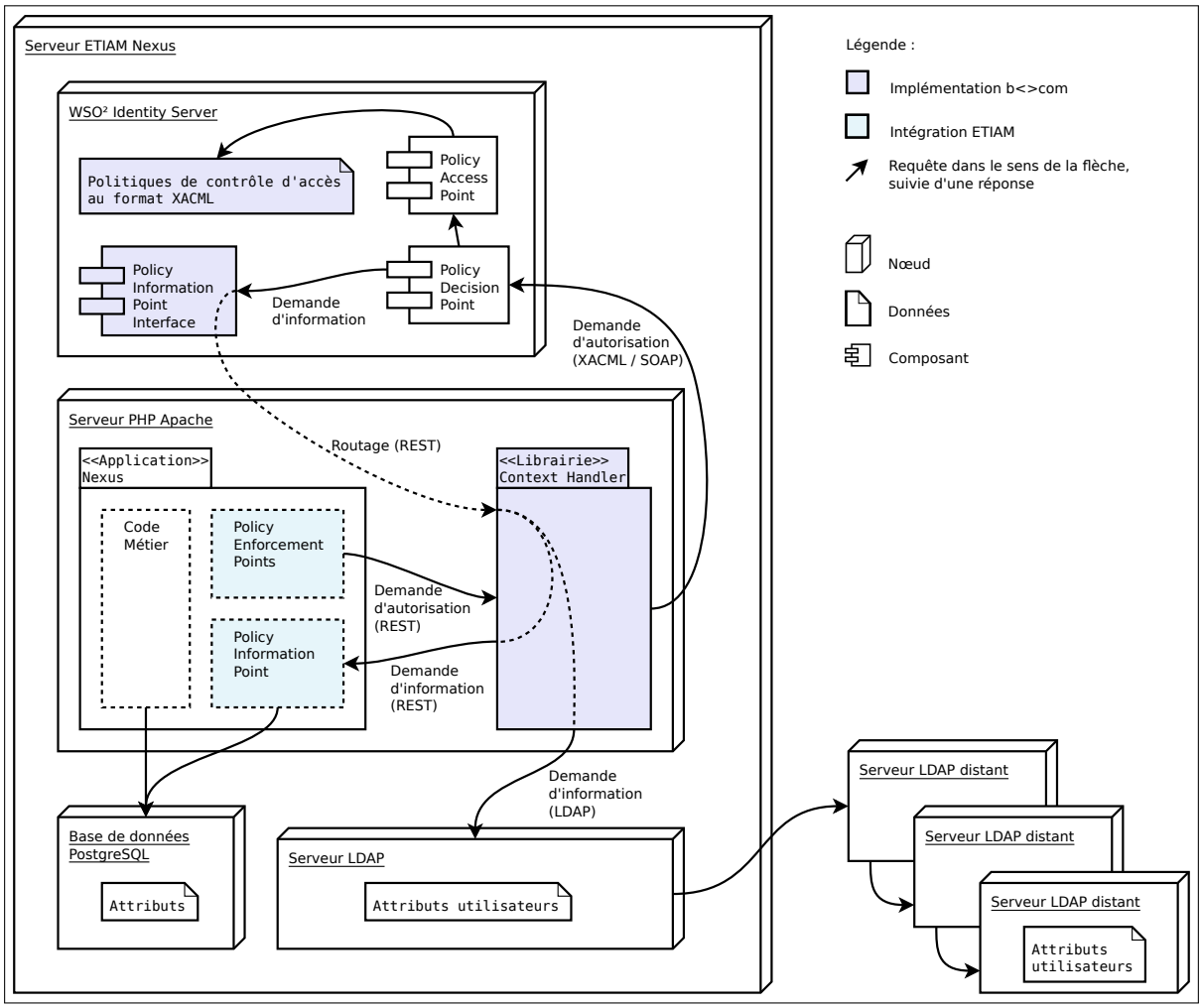

Figure 3.4 – Diagramme de déploiement du système de contrôle d'accès

Les parties en bleu pâle sont celles que je dois réaliser et en vert pâle celles qui sont du ressort de l'intégrateur, ETIAM pour ce schéma. Les flèches représentent les flux d'information relatifs au contrôle d'accès. Elles sont orientées dans le sens des requêtes, ce qui signifie que les demandes d'informations se font dans leur sens et que les réponses, qui contiennent l'information utile, circulent en sens inverse. Entre parenthèses est indiqué le protocole utilisé lorsque sa nature est importante. Les boîtes représentent des nœuds matériels ou logiciels, qui peuvent être imbriqués. Leur titre est souligné en haut à gauche de chaque boîte. Les données sont représentées par des rectangles au coin supérieur droit

replié. Les composants logiciels sont représentés par une boîte avec deux rectangles qui en débordent sur la gauche. Bien que l'application Nexus soit réalisée par des composants logiciels, je ne l'ai pas fait apparaître sur ce schéma.

Ce diagramme est à rapprocher de la figure 3.1, qui montre un flux de données selon XACML, dont je reprends la terminologie et les grandes idées. Lorsqu'un utilisateur effectue une action qui nécessite l'accès à des données sensibles via l'application Nexus, un bout de code (le *policy enforcement point*) envoie une demande d'autorisation au *context handler*. Cette requête est traduite en XACML et envoyée par web-services[10] au PDP de WSO_2 *Identity Server*. Elle est ensuite évaluée en fonction des politiques de contrôle d'accès. Le PDP va potentiellement lancer des demandes d'informations complémentaires aux modules d'information. Je fournis un de ces modules, le *policy information point interface* (PIP interface), implémenté en Java comme le reste d'*Identity Server*, qui ne fait que retransmettre la requête au *context handler*. Cette demande d'information est traitée pour pouvoir être reformulée en LDAP ou retransmise à l'interface de demande d'information du Nexus, qui fera en interne une requête en base de données pour récupérer les informations. Une fois les informations complémentaires transmises au PDP, il peut terminer d'évaluer sa requête d'autorisation.

Bien évidemment, les politiques XACML sont préalablement chargées en mémoire vive par le PDP et reformatées dans une structure plus adaptée au processus d'évaluation des requêtes.

En théorie, le PDP devrait demander les informations complémentaires directement au PIP de l'application Nexus ou au serveur LDAP, mais ce processus n'est pas normalisé par XACML. Identity Server est construit pour faire ces requêtes à un module Java, aussi j'en fournis un qui route ces requêtes vers le PIP effectif.

L'application Nexus étant construite en modules PHP communicant via le protocole REST, il est plus simple que les communications vers le *context handler* soient aussi en REST. Cependant, rien n'empêche d'inclure le *context handler* dans une application PHP comme une librairie classique et de l'utiliser par appel de méthode.

Le serveur LDAP local contenant les données sur l'identité des utilisateurs de l'application Nexus peut être connecté à des serveurs LDAP d'autres serveurs Nexus. En effet, ces serveurs sont déjà interconnectés et LDAP supporte nativement cette fonctionnalité.

La plus grosse partie de ma contribution technique est la librairie « context handler », selon la dénomination consacrée, qui fait principalement de la traduction et du routage de requêtes. L'autre importante partie est la rédaction de politiques de contrôle d'accès.

J'ai essayé de garder cette organisation logicielle simple en évitant le superflu. Le module *PIP Interface* n'est présent que parce que *Identity Server* que je réutilise se sert de ce mécanisme pour accéder aux informations complémentaires. En effet, *Identity Server* est conçu pour être utilisé en environnement Java où ce module d'interfaçage n'est plus nécessaire.

Ayant commencé mon stage relativement tard, le 22 avril, je n'ai pas encore beaucoup avancé sur l'implémentation, qui est pour l'instant très incomplète. En revanche, je compte avoir une implémentation complète d'ici la fin de mon stage, le 22 septembre. Celle-ci ne sera pas encore intégrée dans le serveur Nexus et encore moins portée sur le démonstrateur, il restera donc du travail pour plus tard. Mais j'y reviendrai dans la section 3.3. C'est donc à regret que je ne peux intégrer dans ce rapport d'exemples pertinents pour illustrer mon travail.

Je n'ai considéré dans le détail pour l'instant que l'architecture logicielle pour le serveur d'ETIAM, car il s'agit de l'objectif premier. Cependant elle devrait être aisément transposable pour le démonstrateur de l'équipe IsIMeD dans un second temps.

Les différents composants logiciels et leur organisation sont maintenant tous présentés. Vous devriez avoir une vision plus claire de la mécanique de contrôle d'accès que j'ai proposé durant mon stage. Il devient désormais possible d'aborder les politiques de sécurité elles-mêmes et leur interaction avec ce système.

10. Un protocole plus performant que les web-services est aussi disponible, mais il est plus complexe à mettre en œuvre. C'est une évolution future envisagée.

3.2.4 Politiques

Un système de contrôle d'accès serait bien inutile sans politiques décrivant sous quelles conditions des autorisations doivent être délivrées.

La contrainte principale pour la rédaction de politiques de contrôle d'accès dans le domaine médical est d'autoriser les médecins à accéder à des données confidentielles en urgence. Cela revient en pratique à définir deux jeux de politiques, une nominale qui se doit d'être précise, et une autre plus laxiste.

Puisque cette seconde politique risque de permettre l'accès à des personnes non autorisées, elle ne doit pas accorder de droits permettant des actions irréversibles. Ainsi, les droits d'écriture ne peuvent être accordés par cette politique, mais des droits d'ajout peuvent l'être.

Les autorisations délivrées selon cette politique doivent aussi être assortis de contraintes, comme la conservation d'un historique des actions effectuées.

Les règles du droit français en matière, entre autres, de contrôle d'accès se retrouvent dans la Politique générale de sécurité des systèmes d'information de santé (PGSSI-S) [1]. Le listing 5.3 en annexes page 30 présente la transposition dans le langage ALFA des règles majeures de ce référentiel qui peuvent être formulées dans une politique de contrôle d'accès.

La PGSSI-S n'est pas encore finalisé, aussi cette politique est amenée à évoluer. Cela demandera surtout un effort de vigilance plutôt que technique, ce n'est donc pas un problème majeur. On retrouve là un avantage à regrouper toutes les règles de sécurité en un point unique : leur maintenance s'en retrouve grandement facilitée.

Ces règles ne sont certainement pas suffisantes car elles ne prennent pas en compte tous les besoins spécifiques à l'application Nexus. Elle devra être complétée lors de la phase d'intégration par une politique écrite pour cette application.

Le droit français a aussi une spécificité qui pourrait être pénible à mettre en œuvre : « Sur décision du responsable de traitement, le patient titulaire des données peut avoir accès au contrôle du régime d'habilitation et des habilitations associées. (...) Un patient peut alors définir un régime d'habilitation restrictif au risque de bloquer l'accès à des acteurs de santé qui ne disposeront alors pas des données médicales, par exemple lors d'une prise en charge en urgence (...) » (§ 5.3.5 des principes fondateurs de la PGSSI-S). Il faudrait donc que le « responsable de traitement » ait un moyen d'ajouter une politique spécifique pour un patient. Je serais fort étonné qu'un administrateur prenne le temps d'apprendre le langage XACML pour cet hypothétique cas. D'un autre côté, je ne vois pas de solution plus simple.

Dans les règles de la politique montrée sur le listing 5.3, on peut voir deux types de conditions, celles précédées du mot clé *target* et celles qui suivent le mot *condition*. Elles ont un rôle proche mais leur différence est importante. Les première sont des filtres d'application de la règle. Ainsi, lorsqu'une requête est reçue par le PDP, les règles applicables sont filtrées selon les attributs de cette requête. Par exemple, dans la requête « *Alice* peut-elle *lire* le *dossier de Bob* ? », les attributs *Alice*, *lire* et *dossier de Bob* sont utilisés pour faire un premier tri rapide entre les règles applicables. On note qu'à cette étape, le PDP n'a effectué aucune demande d'information complémentaire et qu'il n'a pas la moindre idée de qui est Alice ou Bob. Une règle qui aurait un filtre portant sur, par exemple, le rôle d'un des acteurs ne sera jamais évaluée. Une telle contrainte doit être placée en tant que *condition*. Il faut donc faire très attention lors de la rédaction de règles filtrantes.

Pour palier cette difficulté, il est possible de n'utiliser que des *conditions* au lieu de *targets*. On perdrait alors l'intérêt de ces dernières, qui est d'accélérer le traitement des politiques grâce à une première évaluation rapide des règles applicables. En effet, les politiques XACML sont préchargées par le PDP qui peut utiliser une structure adaptée au filtrage. Ces

filtres ont d'ailleurs des possibilités d'expression bien plus réduites que les conditions pour permettre des optimisations.

Pour revenir sur l'exemple précédent, « *Alice* peut-elle *lire* le *dossier de Bob* ? », seules les règles *allowParticipant* et *allowCommunity* seront activées car leur filtre « actionId == "read" » s'applique sur l'attribut *lire* de la requête. Pour la première règle, une demande d'information complémentaire sera alors envoyée par le PDP pour savoir si Alice est un médecin ou une infirmière, si le dossier de Bob est suivi par Alice et si ce dossier est ouvert. Pour la seconde règle une demande sera aussi effectuée pour savoir si Alice fait partie de la même communauté (ou service) qu'un des participants au dossier de Bob.

Une fois que le PDP aura ces informations, il pourra évaluer les règles et rendre sa décision.

Les politiques appliquent leurs règles sur des attributs qui doivent avoir un sens pour l'application. En pratique, le PIP de l'application doit pouvoir répondre aux demandes d'informations sur l'ensemble des attributs utilisés dans les politiques. Par exemple, pour qu'un PIP puisse répondre à la question « Quel est le *rôle* de l'*utilisateur* Alice ? », il doit connaître l'attribut *rôle*.

Le modèle ABAC utilisé pour décrire les politiques considère quatre type d'objets : les sujets (utilisateurs), les actions, les ressources et l'environnement (le contexte). En me basant sur un document technique d'ETIAM, j'ai défini les attributs suivants qui peuvent être utilisés par les politiques de contrôle d'accès :

- Concernant les sujets :
  - `profile` : le ou les rôles de l'utilisateur ;
  - `community` : le ou les services de l'utilisateur.
- Concernant les ressources :
  - `resourceId` : un identifiant de ressource (déjà existant dans le standard XACML) ;
  - `case.patientId` : l'identifiant unique du patient dont c'est le télédossier ;
  - `case.step` : l'étape du télédossier ;
  - `case.participantId` : les identifiants des personnes impliquées ;
  - `case.workflow` : le type du télédossier ;
  - `case.state` : l'état du télédossier ;
  - `case.studyId` : les identifiants des études contenues ;
  - `case.priority` : la priorité ;
  - `case.closed` : indique si le télédossier est clos ;
  - `study.patientId` : le patient sujet de l'étude ;
  - `study.serieId` : la série dot fait partie l'étude ;
  - `study.instanceId` : l'identifiant d'instance ;
  - `study.caseId` : l'identifiant du télédossier qui contient l'étude ;
  - `patient.patientId` : l'identifiant unique du patient ;
  - `patient.name` : le nom d'usage du patient ;
  - `patient.altName` : les noms alternatifs du patient ;
  - `patient.birthdate` : la date de naissance du patient ;
  - `patient.gender` : le sexe du patient ;
  - `patient.ethnicity` : le groupe ethnique du patient
- Concernant l'environnement :
  - `lifeInPeril` : indique si une vie est en danger ;
  - `urgency` : indique un état d'urgence.

L'écriture de politiques de sécurité n'est pas la tâche la plus compliquée, mais elle suppose de connaître le fonctionnement de l'application, notamment ses attributs disponibles et de ses besoins.

Ceci conclut la présentation du travail que j'ai effectué lors de mon stage. Ce dernier n'est pas encore fini, il me restera trois semaines de travail après ma soutenance que je compte employer pour développer une démonstration du système de contrôle d'accès. Je reviendrai plus en détail sur ce point à la section suivante.

3.3 Travail restant

Dans cette section, je présente le travail qu'il me reste à faire d'ici la fin de mon stage, le 22 septembre, ainsi que le travail qu'il reste à réaliser pour remplir tous les objectifs du sujet de stage. Je devrais d'ailleurs être repris en CDD pour pouvoir remplir cette tâche. Cependant, pour l'instant aucun contrat n'est signé.

Deux tâches principales restent à compléter. En premier lieu, il me faut terminer le développement du contrôle d'accès. Ensuite il faudra l'intégrer sur le serveur d'ETIAM et dans le démonstrateur de l'équipe.

Développement

En ce qui concerne le développement, il me faut tout d'abord terminer la démonstration en local de ma solution, que j'ai présentée dans ce rapport. Je le ferai je pense sans difficultés d'ici la fin de mon stage. Il s'agit principalement d'un travail de développement en PHP relativement classique, qui prendra juste un peu de temps. Le plus dur sera sans doute d'obtenir des jolies pages web, mes talents de graphistes étant plutôt limités.

Cela devrait donner lieu à la rédaction d'un rapport interne sur le contrôle d'accès pour des applications médicales. Il est fort probable que je récupère une bonne partie du rapport de stage pour ce faire, après les reformulations d'usage pour adopter un style plus *business compliant*.

Ensuite il faudra ajouter des fonctionnalités pour répondre à toutes les exigences des cas d'utilisation [11], les tester, les affiner et en ajouter d'autres.

Il sera temps aussi de rédiger une documentation du processus d'installation et d'intégration du système de contrôle d'accès.

Il nous faudra aussi réfléchir plus en détail à la manière dont on l'inclura dans le démonstrateur de l'équipe. Peut-être faudra-il réimplémenter le *context handler* dans un autre langage que le PHP. L'architecture générale ne devrait cependant pas changer.

Intégration

L'intégration du système de contrôle d'accès dans le serveur Nexus devrait se faire dans un processus itératif où je modifierai ma solution en fonction des retours de l'ingénieur chargé de l'intégration. Cela pourra par exemple passer par l'ajout de fonctionnalités à l'API du *context handler*.

Nous prévoyons, avec le chef d'équipe, de rencontrer prochainement le ou les ingénieurs d'ETIAM qui seront en charge de l'intégration afin d'en discuter.

L'intégration au sein du démonstrateur de l'équipe devrait se faire à suivre. Cela devrait permettre de valider en milieu opérationnel le moteur de contrôle d'accès.

À ce point, le processus d'intégration devrait être suffisamment bien formalisé pour qu'un autre ingénieur que moi puisse la réaliser, si besoin.

11. Les cas d'utilisation ont été présentés page 9.

3.4 Acquis personnels

Revenons un instant sur le travail sur j'ai accompli jusqu'ici et plus particulièrement sur l'expérience que je pense en avoir retiré.

Ce stage m'a permis de me former sur les méthodes de contrôle d'accès au sein d'applications et au langage XACML. Il m'a permis de voir en pratique comment ma modeste expertise en sécurité informatique peut se marier avec des travaux de développement logiciel, pour lesquels j'ai un peu plus d'expérience.

Mon stage ne s'est bien sûr pas cantonné aux tâches spécifiées dans le sujet. J'ai ainsi eu l'opportunité de suivre une courte formation au dépôt de brevets et j'ai assisté à un exposé sur le chiffrement complètement homomorphe [12].

J'ai pu aussi assister à la conférence SSTIC [13]. Je remercie d'ailleurs sur ce point les enseignants qui nous ont incités à y aller.

J'ai aussi pu améliorer mes compétences en LaTeX, d'abord en transposant le modèle de **b com** pour les rapports depuis son format Microsoft Word, ensuite en l'utilisant pour ce rapport. À ce sujet, le logo que vous voyez dans ce paragraphe a été recréé en LaTeX.

12. « Chiffrement (complètement) homomorphe : de la théorie à la pratique » : `http://webmath.univ-rennes1.fr/crypto/2013/Tancrede_Lepoint_fr.html`
13. SSTIC : Symposium sur la sécurité des technologies de l'information et des communications. `https://www.sstic.org`

4 Conclusion

Ce stage a conclu ma formation en sécurité des systèmes d'information de l'ISTIC. Je l'ai effectué à **b com**, un récent institut de recherche technologique situé près de Rennes. J'ai pu y appliquer une partie de mes connaissances acquises lors de ma formation et je les ai approfondies dans le domaine du contrôle d'accès. Il fut un complément pratique à une formation initiale plus théorique.

Au cours de ce stage, j'ai étudié comment concilier des besoins de sécurité paradoxaux. En effet, j'ai été chargé de mettre en œuvre une solution de contrôle d'accès dans une application médicale. Elle doit permettre un accès rapide aux données relatives aux patients tout en préservant la confidentialité de ces mêmes données. Ce conflit est résolu par un mécanisme dit de « bris de glace », qui permet en cas d'urgence d'augmenter les privilèges d'un utilisateur en lui ajoutant des contraintes par ailleurs. Ce compromis permet à un médecin d'accéder à un dossier qu'il ne suit pas en notifiant un supérieur de l'accès potentiellement illégal.

Ce procédé devant être mis en œuvre dans une application existante, j'ai proposé une architecture logicielle qui permet de centraliser les règles de contrôle d'accès au sein d'une politique. Cela permet d'avoir un point unique à modifier lors de la maintenance des politiques de contrôle d'accès ou lors de l'ajout de fonctionnalités au logiciel. Afin de concevoir cette architecture, j'ai au préalable réalisé un état de l'art des méthodes et technologies existantes. Pour l'écriture des politiques, j'ai au final choisi d'utiliser le standard éprouvé XACML, qui est une implémentation du modèle théorique de contrôle d'accès basé sur les attributs.

Une implémentation initiale de ma solution est toujours en développement, mais j'ai bon espoir de la finir d'ici la fin de mon stage. Je devrai ensuite être embauché pour collaborer à son intégration dans le logiciel cible et dans un démonstrateur technique.

5 Annexes

5.1 Comparaison entre ALFA et XACML

```
namespace exemple {
    policy testPolicy {
        condition Attributes.profile == "physician"
        apply denyUnlessPermit

        rule allowParticipant {
            target clause Attributes.actionId == "read"
                       or Attributes.actionId == "write"
                       or Attributes.actionId == "append"
                       or Attributes.actionId == "list"
            condition Attributes.subjectId == Attributes.case.participantId
            permit
        }

        rule allowUrgency {
            target clause Attributes.actionId == "read"
                       or Attributes.actionId == "append"
                       or Attributes.actionId == "list"
            condition Attributes.lifeInPeril == true
            permit
        }
    }
}
```

Listing 5.1 – Exemple de politique en ALFA

Le listing 5.1 présente un exemple d'une courte politique de contrôle d'accès écrite dans le langage ALFA. Le listing suivant montre la même politique compilée en XACML :

```
<?xml version="1.0" encoding="UTF-8"?>
<xacml3:PolicySet xmlns:xacml3="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
    PolicySetId="exemple.testPolicy.conditionpolicyset"
    PolicyCombiningAlgId="urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:on-permit-apply-second"
    Version="1.0">
    <xacml3:Description/>
    <xacml3:PolicySetDefaults>
        <xacml3:XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</xacml3:XPathVersion>
    </xacml3:PolicySetDefaults>
    <xacml3:Target/>
    <xacml3:Policy xmlns:xacml3="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
    PolicyId="exemple.testPolicy.conditionpolicy"
        RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides"
        Version="1.0">
        <xacml3:Description/>
        <xacml3:PolicyDefaults>
            <xacml3:XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</xacml3:XPathVersion>
        </xacml3:PolicyDefaults>
        <xacml3:Target/>
        <xacml3:Rule Effect="Permit" RuleId="exemple.testPolicy.conditionrule">
            <xacml3:Description/>
```

```
            <xacml3:Target/>
            <xacml3:Condition>
                <xacml3:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
                    <xacml3:Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
                    <xacml3:AttributeValue
                        DataType="http://www.w3.org/2001/XMLSchema#string">physician</xacml3:AttributeValue>
                    <xacml3:AttributeDesignator
                        AttributeId="urn:x-b-com:chc:xacml:attr:subject:profile"
                        DataType="http://www.w3.org/2001/XMLSchema#string"
                        Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
                        MustBePresent="false"
                    />
                </xacml3:Apply>
            </xacml3:Condition>
        </xacml3:Rule>
    </xacml3:Policy>
    <xacml3:Policy xmlns:xacml3="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
        PolicyId="exemple.testPolicy"
        RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit"
        Version="1.0">
        <xacml3:Description/>
        <xacml3:PolicyDefaults>
            <xacml3:XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</xacml3:XPathVersion>
        </xacml3:PolicyDefaults>
        <xacml3:Target/>
        <xacml3:Rule
                Effect="Permit"
                RuleId="exemple.testPolicy.allowParticipant">
            <xacml3:Description/>
            <xacml3:Target>
                <xacml3:AnyOf>
                    <xacml3:AllOf>
                        <xacml3:Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                            <xacml3:AttributeValue
                                DataType="http://www.w3.org/2001/XMLSchema#string">read</xacml3:AttributeValue>
                            <xacml3:AttributeDesignator
                                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                                DataType="http://www.w3.org/2001/XMLSchema#string"
                                Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                                MustBePresent="false"
                            />
                        </xacml3:Match>
                    </xacml3:AllOf>
                    <xacml3:AllOf>
                        <xacml3:Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                            <xacml3:AttributeValue
                                DataType="http://www.w3.org/2001/XMLSchema#string">write</xacml3:AttributeValue>
                            <xacml3:AttributeDesignator
                                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                                DataType="http://www.w3.org/2001/XMLSchema#string"
                                Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                                MustBePresent="false"
                            />
                        </xacml3:Match>
                    </xacml3:AllOf>
                    <xacml3:AllOf>
                        <xacml3:Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                            <xacml3:AttributeValue
                                DataType="http://www.w3.org/2001/XMLSchema#string">append</xacml3:AttributeValue>
                            <xacml3:AttributeDesignator
                                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                                DataType="http://www.w3.org/2001/XMLSchema#string"
                                Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                                MustBePresent="false"
                            />
                        </xacml3:Match>
                    </xacml3:AllOf>
                    <xacml3:AllOf>
                        <xacml3:Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                            <xacml3:AttributeValue
                                DataType="http://www.w3.org/2001/XMLSchema#string">list</xacml3:AttributeValue>
                            <xacml3:AttributeDesignator
                                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                                DataType="http://www.w3.org/2001/XMLSchema#string"
                                Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                                MustBePresent="false"
                            />
                        </xacml3:Match>
                    </xacml3:AllOf>
                </xacml3:AnyOf>
            </xacml3:Target>
            <xacml3:Condition>
                <xacml3:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-any">
                    <xacml3:Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
                    <xacml3:AttributeDesignator
```

```
                    AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                    DataType="http://www.w3.org/2001/XMLSchema#string"
                    Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
                    MustBePresent="false"
                />
                <xacml3:AttributeDesignator
                    AttributeId="urn:x-b-com:chc:xacml:attr:case:participant-id"
                    DataType="http://www.w3.org/2001/XMLSchema#string"
                    Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                    MustBePresent="false"
                />
            </xacml3:Apply>
        </xacml3:Condition>
    </xacml3:Rule>
    <xacml3:Rule
            Effect="Permit"
            RuleId="exemple.testPolicy.allowUrgency">
        <xacml3:Description/>
        <xacml3:Target>
            <xacml3:AnyOf>
                <xacml3:AllOf>
                    <xacml3:Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <xacml3:AttributeValue
                            DataType="http://www.w3.org/2001/XMLSchema#string">read</xacml3:AttributeValue>
                        <xacml3:AttributeDesignator
                            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                            DataType="http://www.w3.org/2001/XMLSchema#string"
                            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                            MustBePresent="false"
                        />
                    </xacml3:Match>
                </xacml3:AllOf>
                <xacml3:AllOf>
                    <xacml3:Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <xacml3:AttributeValue
                            DataType="http://www.w3.org/2001/XMLSchema#string">append</xacml3:AttributeValue>
                        <xacml3:AttributeDesignator
                            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                            DataType="http://www.w3.org/2001/XMLSchema#string"
                            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                            MustBePresent="false"
                        />
                    </xacml3:Match>
                </xacml3:AllOf>
                <xacml3:AllOf>
                    <xacml3:Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <xacml3:AttributeValue
                            DataType="http://www.w3.org/2001/XMLSchema#string">list</xacml3:AttributeValue>
                        <xacml3:AttributeDesignator
                            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                            DataType="http://www.w3.org/2001/XMLSchema#string"
                            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                            MustBePresent="false"
                        />
                    </xacml3:Match>
                </xacml3:AllOf>
            </xacml3:AnyOf>
        </xacml3:Target>
        <xacml3:Condition>
            <xacml3:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
                <xacml3:Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-equal"/>
                <xacml3:AttributeValue
                    DataType="http://www.w3.org/2001/XMLSchema#boolean">true</xacml3:AttributeValue>
                <xacml3:AttributeDesignator
                    AttributeId="urn:x-b-com:chc:xacml:attr:ctx:life-in-peril"
                    DataType="http://www.w3.org/2001/XMLSchema#boolean"
                    Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
                    MustBePresent="false"
                />
            </xacml3:Apply>
        </xacml3:Condition>
    </xacml3:Rule>
  </xacml3:Policy>
</xacml3:PolicySet>
```

Listing 5.2 – Exemple de politique en XACML

En comparant les deux versions, on comprend aisément l'intérêt majeur d'utiliser ALFA pour rédiger des politiques. À cette concision s'ajoute la complétion automatique et la correction d'erreurs offerte par le plugin ALFA pour Eclipse.

5.2 Politique PGSSI-S transposée en ALFA

Le listing suivant présente des règles de la Politique Générale de Sécurité des Systèmes d'Information de Santé (PGSSI-S) transposées dans le langage ALFA. La version compilée en XACML de cette politique ne présente pas grand intérêt et prendrait 6 pleines pages dans ce document, je ne l'ai donc pas incluse.

```
namespace pgssis {

    namespace obligations {
        obligation identifyAPosteriori = "urn:x-b-com:chc:xacml:obl:identify-a-posteriori"
        obligation notifySuperior = "urn:x-b-com:chc:xacml:obl:notify-superior"
        obligation logProofOfAction = "urn:x-b-com:chc:xacml:obl:log-proof-of-action"
    }

    policyset topLevel {
        apply denyUnlessPermit


        /*
         * Règle PGSSI-S 5.2.3-1 :
         *
         * « Seuls les personnels de santé impliqués dans la prise en charge
         * du patient doivent pouvoir accéder à ses données de santé »
         */
        policy onlyInCharge {

            apply permitOverrides

            rule allowParticipantOpen {
                target clause Attributes.actionId == "open"

                condition Attributes.subjectId == Attributes.case.participantId
                    && Attributes.case.closed == true
                    && (Attributes.profile == "physician"||
                        Attributes.profile == "nurse")

                permit
            }

            rule allowParticipant {
                target clause Attributes.actionId == "read"
                           or Attributes.actionId == "write"
                           or Attributes.actionId == "append"
                           or Attributes.actionId == "list"
                           or Attributes.actionId == "archive"
                           or Attributes.actionId == "open"
                           or Attributes.actionId == "close"

                condition Attributes.subjectId == Attributes.case.participantId
                    && Attributes.case.closed == false
                    && (Attributes.profile == "physician"||
                        Attributes.profile == "nurse")

                permit
            }

            rule allowCommunity {
                target clause Attributes.actionId == "read"
                           or Attributes.actionId == "append"
                           or Attributes.actionId == "list"

                condition Attributes.community == Attributes.case.participantId
                    && Attributes.case.closed == false
                    && (Attributes.profile == "physician"||
                        Attributes.profile == "nurse")

```

```
                permit
            }

            rule default {
                deny
            }
        }

        /*
         * Règle PGSSI-S 5.2.3-3 :
         *
         * « Les dispositifs de sécurité doivent intégrer un mode d'accès
         * de type « bris de glace » qui peut être accordé en cas d'urgence
         * à des professionnels non identifiés a priori. Le recours à cet
         * accès en urgence doit être surveillé. »
         */
        policy glassBreak {

            apply denyUnlessPermit

            rule allowParticipant {
                target clause Attributes.actionId == "read"
                           or Attributes.actionId == "append"
                           or Attributes.actionId == "list"
                           or Attributes.actionId == "open"
                       clause Attributes.lifeInPeril == true
                           or Attributes.urgency == true

                condition Attributes.profile == "physician"
                      || Attributes.profile == "nurse"
                      || Attributes.profile == "computer"

                permit
            }

            on permit {
                obligation obligations.identifyAPosteriori
                obligation obligations.notifySuperior
            }
        }

        /*
         * Règle PGSSI-S 5.2.4-1 :
         *
         * « Tout accès aux données de santé personnelles doit être tracé
         * et transmis au patient lorsqu'il en fait la demande. »
         * « les traces des actions sur le SIS doivent (1) être recueillies
         * dans les systèmes, (2) être imputables à leur auteur et (3) avoir
         * valeur de preuve »
         */
        on permit {
            obligation obligations.logProofOfAction
        }
    }
}
```

Listing 5.3 – Politique PGSSI-S transposée en ALFA

BIBLIOGRAPHIE

[1] ASIP Santé et Ministère des affaires sociales et de la santé: *Politique Générale de Sécurité des Systèmes d'Information de Santé (PGSSI-S) – Principes Fondateurs*, 2013.

[2] National Electrical Manufacturers Association: *Digital Imaging and Communications in Medicine (DICOM) Part 10 : Media Storage and File Format for Media Interchange*, 2011.

[3] Dierks, T. et C. Allen: *RFC 2246 : The Transport Layer Security (TLS) Protocol*. IETF, 1999. https://tools.ietf.org/html/rfc2246.

[4] National Electrical Manufacturers Association: *Digital Imaging and Communications in Medicine (DICOM) Part 15 : Security and System Management Profiles*, 2011.

[5] Dierks, T. et E. Rescorla: *RFC 4346 : The Transport Layer Security (TLS) Protocol*. IETF, 2006. https://tools.ietf.org/html/rfc4346.

[6] Integrating the Healthcare Enterprise: *IHE IT Infrastructure Technical Framework Version 2 or later, Vol. 1 – Section 9, Vol. 2 – Sections 3.19 and 3.20*, 2013.

[7] Integrating the Healthcare Enterprise: *IHE IT Infrastructure Technical Framework Version 5 or later, Vol. 1 – Section 13, Vol. 2(b) – Section 3.40*, 2013.

[8] OASIS: *SAML V2.0 Specifications*, 2005.

[9] Jörg Caumanns, Raik Kuhlisch, Oliver Pfaff et Olaf Rode: *IHE IT-Infrastructure – White Paper – Access Control*, 2009.

[10] Ferraiolo, D.F. et D.R. Kuhn: *Role-Based Access Control*. Dans *15th National Computer Security Conference*, pages 554–563, 1992. http://csrc.nist.gov/groups/SNS/rbac/documents/ferraiolo-kuhn-92.pdf.

[11] Kalam, A. Abou El, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel et G. Trouessin: *Organization Based Access Control*. Dans IEEE (rédacteur) : *4th International Workshop on Policies for Distributed Systems and Networks*, 2003.

[12] Ajam, N., N. Cuppens-Boulahia et F. Cuppens: *Privacy Administration in Distributed Service Infrastructure*. Dans *6th International ICST Conference on Security and Privacy in Communication Networks (SecureComm)*, 2010.

[13] McGraw, R. W.: *Risk-Adaptable Access Control (RAdAC)*, 2009. http://csrc.nist.gov/news_events/privilege-management-workshop/radac-Paper0001.pdf.

[14] Yuan, E. et J. Tong: *Attributed based access control (ABAC) for Web services*. Dans IEEE (rédacteur) : *International Conference on Web Services (ICWS)*, 2005.

[15] OASIS: *XACML 3.0 Specifications*, 2013. http://docs.oasis-open.org/xacml/3.0/.

[16] Cuppens, F., N. Cuppens-Boulahia et C. Coma: *O2O : Managing Security Policy Interoperability with Virtual Private Organizations*. Dans *HPOVUA 2006*, 2006.

[17] Fielding, Roy Thomas: *Representational State Transfer (REST)*, 2000. https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

[18] J. Sermersheim, Ed.: *RFC 4511 : Lightweight Directory Access Protocol (LDAP) : The Protocol*. IETF, 2006. https://tools.ietf.org/html/rfc4511.

[19] Axiomatics: *Axiomatics Language for Authorization (ALFA)*, 2012. http://www.axiomatics.com/solutions/products/authorization-for-applications/developer-tools-and-apis/192-axiomatics-language-for-authorization-alfa.html.