République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira de Béjaïa

Faculté des Sciences et des Sciences de l'Ingénieur
Département d'Informatique
Ecole Doctorale Réseaux et Systèmes Distribués

Mémoire de Magistère
En Informatique

Option
Réseaux et Systèmes Distribués

Thème

Un modèle hybride pour la sécurité des Services Web (MHS-SW)

Présenté par

KEDJAR Saâdia

Devant le jury composé de :

Président :	DAHMANI Abdenasser	Professeur	Université A/Mira de Béjaia, Algérie
Rapporteur :	TARI Zahir	Professeur	Université RMIT de Melbourne, Australie
Examinateurs :	BADACHE Nadjib	Professeur	Université USTHB, Algérie
	AHMED-NACER Mohamed	Professeur	Université USTHB, Algérie
Invité:	TARI Abdelkamel	Chargé de cours	Université A/Mira de Béjaia, Algérie

Promotion 2006-2007

Dédicaces

A ma mère

A mon père

A mes grands-parents

A toute ma famille

A ma belle-famille

Remerciements

Je tiens à exprimer mes reconnaissances aux personnes qui ont contribué de près ou de loin pour accomplir ce travail. Cet espace ne suffira sûrement pas pour citer toutes ces personnes. Qu'elles trouvent ici mes sincères remerciements.

Mes vifs remerciements accompagnés de gratitude sont adressés à mon encadreur TARI Zahir, professeur à l'université RMIT d'Australie, pour avoir dirigé ce mémoire et pour ses conseils précieux.

Je tiens à remercier vivement Mr TARI Abdelkamel, chargé de cours et chef de département d'informatique à l'université de Béjaia de m'avoir co-encadré, orienté et encouragé. Je remercie également BERREHOUMA Nabil, étudiant à l'école doctorale ReSyD de Béjaia, pour son aide et sa disponibilité

Mes remerciements vont aussi aux membres de jury pour avoir voulu juger mon travail et l'intérêt qu'ils y portent. J'adresse mes remerciements à Monsieur DAHMANI Abdenasser d'avoir présidé le jury. Je tiens à remercier Messieurs BADACHE Nadjib et AHMED Nacer pour avoir accepté de faire partie de jury et d'avoir examiné ce travail.

Je ne pourrai oublier d'adresser ma reconnaissance, mes remerciements et ma plus profonde gratitude à mes parents qui sont toujours derrière ma réussite et que je mets en tête de la liste de tous ceux que je dois remercier. Je remercie mes sœurs et mes frères pour leurs encouragements et aides.

Je remercie particulièrement mon fiancé qui n'a pas cessé de me soutenir durant cette année.

Mes remerciements vont à toutes mes amies qui m'ont accompagné le long de tout mon cursus et tous mes collègues de l'école doctorale ReSyD de Béjaia.

Table des matières

Liste des figures

Liste des tableaux

Liste des algorithmes

Introduction générale

De plus en plus les entreprises adoptent les services Web comme technologie nécessaire à l'intégration de leurs applications. Les progrès rapides dans ce domaine sont dus à l'intérêt permanent des industriels et à la disponibilité immédiate d'outils et de standards (WSDL, SOAP, UDDI, etc.) permettant l'invocation automatisée des fonctions métiers via un échange de messages.

Les services Web offrent un moyen de coopération et d'interopérabilité entre des applications écrites dans des langages de programmation différents et s'exécutant sur des plateformes différentes. De plus, l'utilisation des standards et des protocoles très connus facilite aux développeurs la compréhension et l'adoption de cette technologie.

Cependant, l'ouverture et l'accessibilité des services Web sur Internet les rendent vulnérables aux différentes attaques et le protocole HTTP est une source non négligeable de cette vulnérabilité.

Des standards de sécurité sont mis en place (tels que X.509, SSL, SAML, etc.) pour protéger les services Web des invocations malicieuses et assurer des échanges de messages sécurisés. Cependant, la plupart de ces mécanismes sont concentrés sur la définition de la sécurité au niveau transport et messages et ignorent la sécurité au niveau application.

Pour assurer la sécurité de bout en bout, le niveau application du modèle OSI offre deux mécanismes de sécurité à savoir le contrôle d'accès et le contrôle de flux d'information.

Le contrôle d'accès est nécessaire pour toute entreprise pour gérer les accès à ses services et aux ressources partagées. Dans ce contexte, des techniques de spécification des politiques d'accès et de leur application doivent être disponibles.

En outre, un mécanisme qui vérifie les flux d'information lors des échanges de données entre les objets du système manipulés par les services Web est indispensable pour accomplir la tâche du contrôle d'accès. Le contrôle de flux d'information assure la confidentialité et l'intégrité des informations même après leurs disséminations.

Notre contribution s'inscrit dans le contexte du contrôle d'accès et du contrôle de flux d'information. Notre objectif est de proposer un modèle hybride pour sécuriser les services Web en se basant sur les points suivants : a) étude et vérification du modèle de contrôle de flux d'information pour les services Web proposé par Z. Tari et al [TAR06] et qui est basé sur les labels décentralisés associés aux objets d'un système ; b) amélioration de ce modèle en définissant une nouvelle structure des labels pour assurer non seulement la confidentialité des informations mais aussi leur intégrité et en proposant de nouvelles règles et de nouvelles définitions des jointures ; c) extension du modèle pour permettre le contrôle d'accès aux méthodes des services Web et aux objets qu'elles manipulent vu que le modèle de contrôle de

flux d'information proposé dans [TAR06] ne contrôle que les flux provoqués par les transmissions d'information entre les objets mais ne gère pas les accès des demandeurs de services Web.

Notre approche est globale puisqu'elle est définie d'une manière formelle et permet de spécifier des politiques d'accès et des règles de flux d'information pour n'importe quel système de services Web. Le modèle proposé est aussi flexible parce qu'il est indépendant des services Web et ne perturbe pas l'exécution normale de la logique métier de ces derniers. De plus, la définition du modèle prend en compte l'aspect ouvert et distribué des services Web en utilisant un mécanisme de gestion de confiance basé sur les attributs des utilisateurs.

Notre mémoire est organisé en cinq chapitres. Le premier chapitre traite d'une manière générale les services Web et leur sécurité où nous citons les standards les plus connus pour la mise en œuvre des services Web et les mécanismes de sécurité souvent utilisés dans cette technologie. Dans le deuxième chapitre, nous introduisons le contrôle d'accès d'une manière générale et nous présentons les modèles de contrôle d'accès traditionnels les plus connus puis nous présentons quelques modèles de contrôle d'accès conçus pour les systèmes de services Web. Le troisième chapitre, quant à lui, introduit le mécanisme de contrôle de flux d'information et les travaux importants qui s'inscrivent dans ce contexte.

Nous présentons le modèle hybride proposé dans le quatrième chapitre où nous définissons clairement notre contribution pour l'amélioration du modèle de Z. TARI et al. et nous donnons une architecture logique pour le système de contrôle d'accès et de contrôle de flux d'information. Le cinquième chapitre est consacré au développement formel du modèle de sécurité proposé. Ce chapitre contient les définitions nécessaires des concepts de bases du modèle, la définition des nouvelles règles de contrôle d'accès et de contrôle de flux d'information et les démonstrations concernant la sûreté des flux d'informations qui utilisent les jointures définies dans le modèle.

CHAPITRE 1

Les services Web et la sécurité des Services Web

1.1 Introduction

Les services Web constituent une nouvelle technologie qui est le pont de convergence des acteurs du marché de l'informatique (IBM, SUN, Microsoft, etc.)
L'objectif des services Web est de faciliter l'accès aux applications et ainsi de simplifier les échanges de données entre les entreprises. Ils poursuivent donc le rêve de l'informatique distribuée où les applications pourraient interopérer à travers le réseau, indépendamment de leur plateforme et de leur langage d'implémentation. Dans ce sens, ils s'inscrivent dans la continuité des initiatives telles que CORBA (Common Object Request Broker Architecture, de l'OMG) en apportant toutefois une réponse plus simple, s'appuyant sur des technologies et standards reconnus et maintenant acceptés de tous (XML, HTTP, etc.).

Dans ce chapitre, nous introduisons les concepts de base, les technologies des services Web et nous expliquons comment publier, rechercher et invoquer des services Web dans un environnement distribué. De plus, nous présentons différents standards de sécurité organisés en trois niveaux à savoir le niveau transport, le niveau messages et le niveau application.

1.2 Définition des services Web

La technologie des services Web est initié par IBM et Microsoft et normalisée par le W3C (World Wide Web Consortium). Ce dernier définit un service Web comme étant une application ou un composant logiciel

(i) identifié par un URI (Uniform Resource Identifier),
(ii) dont ses interfaces et ses liens (binding) peuvent être décrits en XML,
(iii) sa définition peut être découverte par d'autres services Web et
(iv) il peut interagir directement avec d'autres services Web à travers le langage XML et en utilisant des protocoles Internet [KEL03].

Dans [SOF07], un service Web est considéré comme un composant implémenté dans n'importe quel langage, déployé sur n'importe quelle plateforme et enveloppé dans une couche de standards dérivés du XML. Il doit pouvoir être découvert et invoqué dynamiquement par d'autres services.

Un service Web est souvent mal compris et quand il est mentionné il se rapporte parfois à un service général fourni sur le Web, comme les prévisions météorologiques sur une page Web par exemple. Les prévisions météorologiques sont un service qui fournit sa fonctionnalité pour une variété d'utilisateurs mais s'il ne comporte pas une interface pour communiquer avec d'autres applications par l'intermédiaire du SOAP il n'est pas un service Web par définition.

1.3 Architecture des services Web

Les principaux concepts [KAC06] intervenant dans l'architecture de base des services Web sont:

1. *Le fournisseur du service* : d'un point de vue conceptuel, il désigne la personne ou l'organisation responsable juridiquement de l'agent logiciel (i.e. le service). D'un point de vue opérationnel, il désigne le serveur qui héberge les services déployés.
2. *Le client du service* : une personne ou une organisation, client potentiel des services de l'agent logiciel. D'un point de vue opérationnel, l'application cliente qui invoque le service.
3. *Le registre des services* : il représente la personne ou l'organisation responsable de l'hébergement du dépôt de publication des services. D'un point de vue fonctionnel, il représente l'entité logicielle qui joue le rôle de l'intermédiaire entre les clients et les fournisseurs de services. Le concept registre ou dépôt de service est essentiel dans l'architecture services Web. Il joue un rôle central dans le processus de localisation des besoins et dans l'interopérabilité, car il est supposé fournir aux clients les informations techniques et sémantiques sur le fonctionnement du service et ceci dans des langages formels (interprétables par les machines).
4. *Le service* : c'est une notion abstraite pour désigner les fonctionnalités d'un agent logiciel qui implémente une fonctionnalité du service. Quand on parle de service, il est essentiel de faire la distinction entre le service vu d'un angle sémantique et sa réalisation qui est l'entité logicielle. Un seul service Web, par exemple, peut avoir plusieurs agents qui l'implémentent (utilisant des langages différents).
5. *La description du service* : c'est la spécification du service exprimée dans un langage de description interprétable par les machines. Il existe deux descriptions de services : une

description technique (le service est vu en terme de messages, de formats, de types, de protocoles de transport et d'une adresse physique) et une description sémantique (description formelle ou un accord entre le fournisseur et le client). L'architecture des services Web impose à chaque entité logicielle accessible sur le Web d'éditer sa description afin d'assurer l'interopérabilité entre le client et le service.

6. *Le message* : il joue un rôle important dans l'architecture de référence, c'est la plus petite unité d'échange de données entre les clients et les services. Le concept message ne reflète aucunement la sémantique de son contenu, seuls la structure et les mécanismes de transport sont considérés dans l'architecture. La structure des messages qui permettent l'invocation des différentes unités fonctionnelles qui composent le service doit figurer dans la description du service.
7. *La ressource* : le concept ressource désigne l'identifiant du service. Il constitue un point important dans l'architecture des services Web. Chaque service doit avoir une identification en terme d'adresse, généralement une URI. Les services Web, comme leur nom l'indique, doivent être accessibles à partir d'une adresse Web.

La figure1.1 représente les différents concepts et relations de l'architecture de référence des services Web.

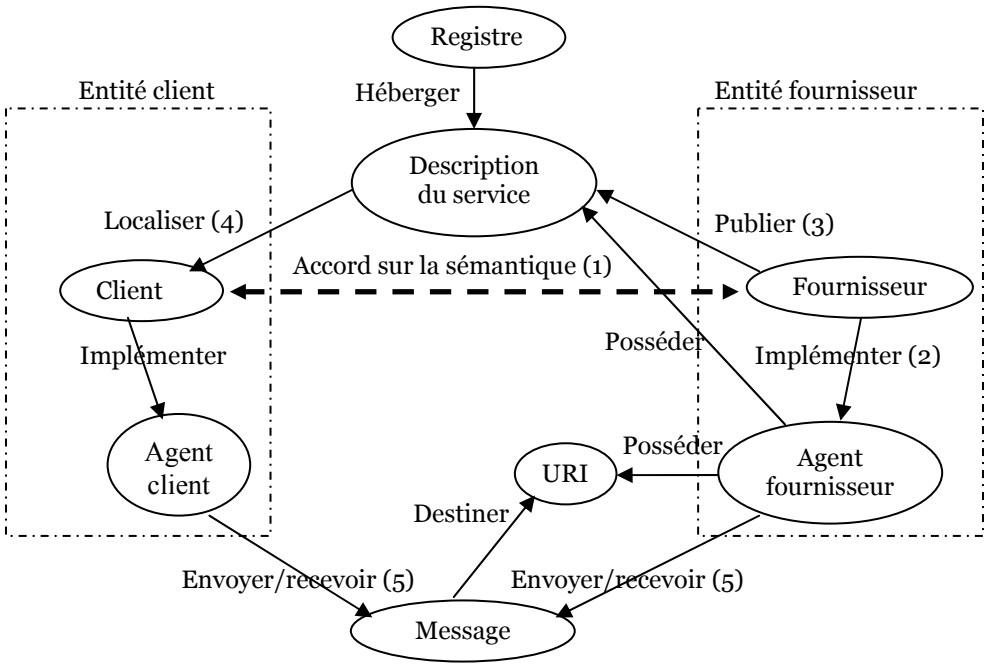


Figure 1.1 : Le modèle conceptuel de l'architecture service Web [MEL04]

Le modèle de la figure ci-dessus représente une vue globale sur les concepts ainsi que les relations qui existent entre eux afin de montrer le fonctionnement global de l'architecture service Web. Chaque concept dans le modèle est défini par l'ensemble des relations qu'il

mène avec les autres concepts. Le modèle de l'architecture donne également des indications sur le cycle de développement des services Web ainsi que leur cycle de vie. La figure 1.1 maintient également une numérotation des relations qui indique les différentes étapes de la mise en place et de l'invocation d'un service Web.

- Chaque service Web est défini par un fournisseur. Le fournisseur de services déploie et publie la description de son service dans des registres en vue d'être localisé par des clients.
- Les clients localisent leurs besoins en terme de services en effectuant des recherches sur les registres de services Web.
- Une fois le service localisé, le client extrait sa description du registre.
- Sur la base des informations définies dans la description du service, le client entreprend une interaction.

1.4 L'infrastructure des services Web

L'infrastructure offre un certain nombre de services qui permettent de gérer l'interaction entre les composants de l'architecture des services Web. Les services offerts par l'infrastructure des services Web concerne essentiellement deux aspects fondamentaux : Un service de communication qui permet l'échange de données entre les services Web et un ensemble de services techniques destinés à automatiser le processus de localisation et d'invocation des composants [CUR01]. L'originalité de l'infrastructure des services Web consiste à mettre en place ces services en se basant exclusivement sur les protocoles les plus répandus d'Internet.

Aujourd'hui, l'infrastructure services Web s'est concrétisée autour de trois spécifications considérées comme des standards [CHA02] :

- Simple Object Access Protocol (SOAP), assure la communication avec et inter services Web ;
- Web Services Description Langage (WSDL), offre un schéma formel de description des services Web.
- Universal Description, Discovery and Integration (UDDI), offre une manière uniforme de définir des registres des services Web et aussi un schéma uniformément extensible de descriptions des services Web.

La figure 1.2 représente la relation entre les composants de l'architecture de base et les technologies standard des services Web.

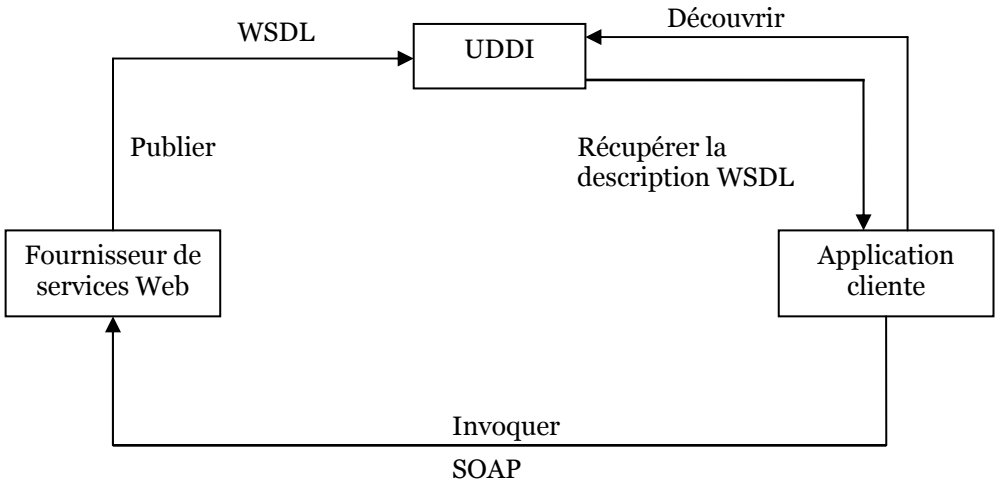


Figure 1.2 : Déploiement, recherche et invocation des services Web

1.4.1 SOAP

Comme chaque autre plateforme de l'informatique distribué, la plateforme des services Web est également arrangée sur un protocole standard. Pour CORBA, le protocole était IIOP; pour RMI, le premier JRMP ensuite IIOP ; dans les environnements de Microsoft, DCOM.
Pour les architectures de services Web, le protocole de base pour la communication entre deux parties est SOAP. SOAP a présenté une variation fondamentale dans les systèmes distribués qui consiste à faire communiquer deux composants distants basés sur des technologies hétérogènes (matérielle et/ou logicielle) et il est basé sur la technologie XML.

Avant SOAP, Les organismes ont eu deux options principales pour communiquer entre deux points géographiquement ou physiquement distribués. La première approche était EDI (Electronic Data Interchange). La deuxième approche a commencé par normaliser le protocole de communication et les infrastructures impliquées : CORBA, DECOM, et RMI. Le problème avec ces technologies est qu'une application CORBA ne pourrait communiquer qu'avec CORBA, RMI avec RMI, DECOM avec DECOM. En plus de l'incapacité à fonctionner en présence de firewalls.

Le protocole SOAP [W3C07] permet de pallier ces limites et il permet de traverser les firewalls. En effet, SOAP ne passe pas par des ports précis mais utilise le protocole HTTP qui lui permet de traverser les *Proxy* et *firewalls*.

SOAP fournit une structure normalisée de message basée sur XML, un modèle de traitement qui décrit comment un service devrait traiter les messages, un mécanisme pour lier les messages SOAP aux différents protocoles de transport du réseau et une manière d'attacher l'information encodée non-XML (contenu du message n'est pas au format XML, par exemple un message contenant un fichier image) aux messages SOAP.

La structure d'un message SOAP

Un message SOAP est composé de deux parties indépendantes :

- une structure XML qui constitue le message SOAP : le message SOAP correspond à un document XML défini par une application cliente ou une application service suivant une structure particulière.
- un entête du protocole de transport : le message SOAP est encapsulé dans un protocole de transport (HTTP par exemple) en vue d'être livré à l'application destination.

Un message SOAP se compose d'un élément obligatoire appelé "SOAP envelope" qui contient zéro ou plusieurs entêtes (extensions) et un corps (charge utile) comme illustré sur la figure 1.3.

L'entête du message (optionnelle), est utile quand le message doit être traité par plusieurs intermédiaires, et obéit à un schéma spécifique et il est destiné à contenir des informations nécessaires aux applications pour interpréter la charge utile. L'entête peut avoir plusieurs fils (Header Blocks). Ces fils sont utilisés pour ajouter une fonctionnalité au message comme l'authentification et la gestion des transactions.

Le corps (obligatoire) contient le message réel destiné au récepteur final. Un corps, par exemple, renferme les méthodes et paramètres qui seront exécutés par le destinataire final s'il s'agit d'une requête, une valeur retournée s'il s'agit d'une réponse ou tout autre contenu XML. Le corps SOAP peut contenir un "SOAP fault". Ce bloque est utilisé pour transmettre l'information des erreurs durant le traitement du message.

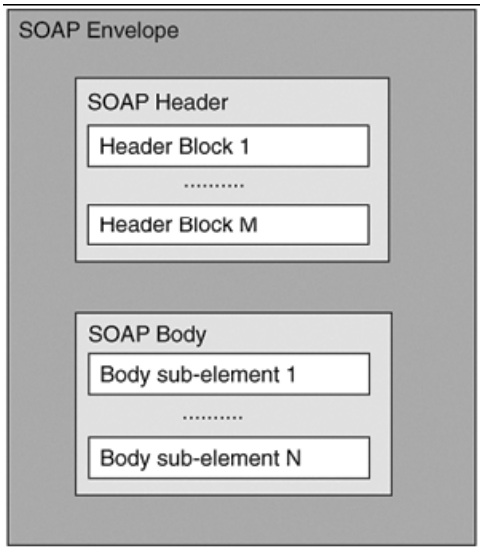


Figure 1.3 : Structure d'un message SOAP [WEE05].

Le transport

SOAP fait une séparation totale entre le message (i.e. le document XML) et son moyen de transport. Il laisse aux développeurs le choix du moyen de transport. Ceci donne un certain nombre d'avantages :

1. La neutralité et l'extensibilité du protocole facilite la mise en place des spécifications pour plusieurs protocoles de transport existants ce qui assure une large portée pour le message SOAP.
2. SOAP est capable de s'adapter aux différents modèles d'échange de messages proposés par les protocoles de transport.
3. SOAP profite des services offerts par les protocoles pour son transport tels que l'authentification, le chiffrement, le contrôle d'accès, la fiabilité, etc.

Le transport sur HTTP

C'est le protocole le plus utilisé pour le transport des messages SOAP. Le protocole HTTP constitue un excellent transport en raison de sa popularité sur le Web. Il existe deux règles à respecter pour pouvoir utiliser SOAP via HTTP [MEL04] :

1. Le mécanisme d'envoi doit utiliser la méthode HTTP POST standard.
2. La destination de la requête HTTP doit être adressée à une URI donnée sur le serveur Web.

1.4.2 WSDL

Avec SOAP, la communication entre les services Web est possible et chaque participant sait comment envoyer et recevoir les messages SOAP correspondants. Pour une exploitation robuste des services Web, ils sont décrits par des métadonnées lisibles par les machines et qui autorisent l'interopérabilité. Ces métadonnées servent à décrire les formats de messages que le service prend en charge et les modèles d'échange de messages valides pour un service. Elles décrivent aussi les capacités et les exigences d'un service. Cette dernière forme de métadonnée se nomme « la stratégie » d'un service. Les formats et les modèles d'échange de message sont exprimés en WSDL. Les stratégies sont formulées en utilisant WS-Policy.

La norme WSDL (Web Service Description Language) [W3C01] est fondée sur XML. Elle permet de spécifier le format de messages, les protocoles qui doivent être utilisés et la localisation des différentes machines qui mettent en œuvre un Service Web.

WSDL décrit un service Web en deux étapes fondamentales (Figure 1.4) :

1. ***abstraite*** : WSDL décrit un service Web en terme des messages qu'il envoie et qu'il reçoit sans aucune référence à une technologie.

2. ***concrète*** : propose une ou plusieurs réalisations de la partie abstraite (par exemple, un type de port peut être réalisé par SOAP+RPC+HTTP et/ou par SOAP+RPC+SMTP).

La structure d'un document WSDL

De point de vue XML, la structure d'un document WSDL se compose de cinq éléments principaux : le service, les ports, les bindings, les portTypes, les messages et les types :

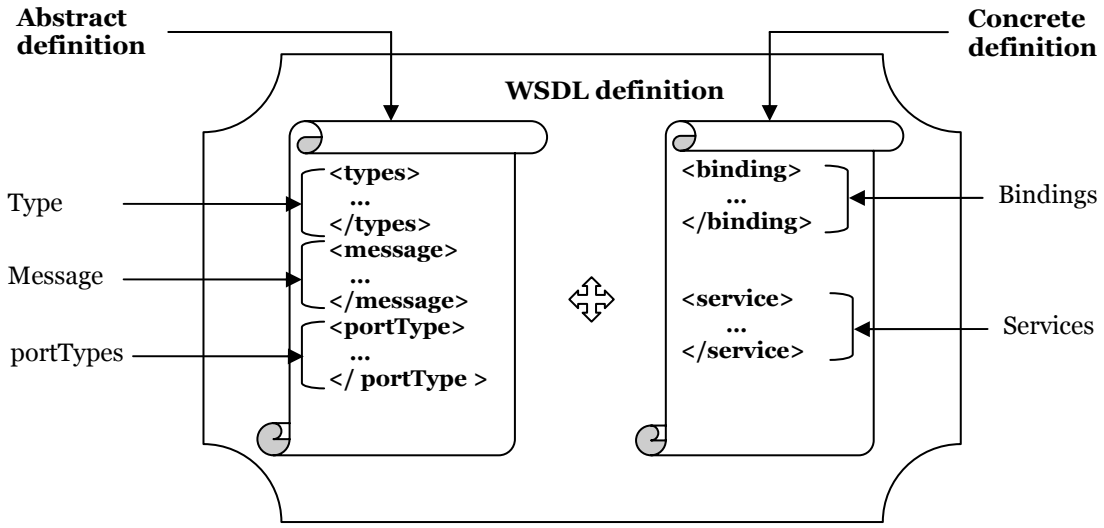


Figure 1.4 : La structure de « WSDL definition », WSDL 1.1.

1. l'élément *types* : contient les définitions des types de données appliqués aux messages échangés. WSDL utilise XSD (XML Schema Definition) en tant que système de type.
2. l'élément *message* : cet élément associe un nom au message à l'aide de l'attribut *name*. Il contient une ou plusieurs parties (élément *part*). Chaque partie utilise les attributs *type* ou *element* pour identifier les types de données. A l'attribut *type* on peut affecter un type simple ou complexe. Il est généralement utilisé dans le modèle de programmation RPC par contre le modèle de programmation document/littéral se fonde typiquement sur l'attribut *element*. L'attribut *name* dans l'élément *part* fournit un nom unique à chacune des parties d'un message.
3. l'élément *portType (interface)* : décrit un ensemble d'opérations (méthodes). Une opération est définie en utilisant l'élément *operation*. Chaque élément *operation* contient des éléments enfants *input* et/ou *output* qui représentent les messages de requêtes et de réponses.

   Les opérations peuvent être de natures : unidirectionnelle, requête/réponse, sollicitation/ réponse, notification.

4. les *bindings* : spécifie une liaison d'un portType à un protocole concret (SOAP, HTTP, MIME, …). Un portType peut avoir plusieurs liaisons.
5. l'élément *port* : fournit une adresse physique à laquelle le service peut être accédé.
6. l'élément *service* : Un service est un ensemble d'éléments port.

Le problème qui reste est comment un utilisateur peut trouver la description correspondante d'un service Web. La section suivante traite la découverte des services Web.

1.4.3 UDDI

UDDI [OAS07] est un standard basé sur XML pour la localisation et la publication des services Web. Il permet d'enregistrer l'information concernant les types de services disponibles et les personnes pouvant utiliser ces services. Il fournit donc un mécanisme de publication et de recherche de services.
Il offre ainsi aux développeurs une grande visibilité des services existants. Il est possible d'obtenir des informations pendant l'exécution sur la disponibilité du service pour des applications dynamiques au travers des services UDDI.

Ainsi lorsque l'on veut mettre à disposition un nouveau service, on crée un fichier appelé *Business Registry* qui décrit le service en utilisant un langage dérivé d'XML. Un registre UDDI est composé de pages blanches, jaunes et vertes. Les pages blanches incluent des informations telles que le nom et l'adresse d'une entreprise. Elles comprennent aussi une liste des identifiants grâce auxquels une entreprise peut être repérée. Les pages jaunes comprennent la description, au format WSDL, des services Web déployés par les entreprises. Elles catégorisent les entreprises en fonction du type d'affaires qu'elles effectuent. Finalement, les pages vertes décrivent le genre de services offerts ainsi que les techniques de communication à employer pour joindre ces services.

Modèle d'informations UDDI

Le modèle d'informations UDDI comporte cinq structures de données principales (figure 1.5), définies dans la spécification sous forme de schéma XML [KAC06] :

- *BusinessEntity* : contient les informations de type pages blanches. Cette structure représente n'importe quel fournisseur d'un service, pas nécessairement une compagnie. Elle contient zéro ou plusieurs éléments businessServices.
- *BusinessService* : contient les informations de type pages jaunes. Cette structure contient des informations sur les services métiers. Elle appartient à exactement un élément businessEntity. Elle contient zéro ou plusieurs éléments bindingTemplates.

- *BindingTemplate* : structure contenant les informations techniques nécessaires pour utiliser un service Web. Elle sert également de pages vertes. Elle appartient à exactement un élément businessService.
- *tModel (technical model)* : La fonction principale de cet élément est de mettre à disposition les pointeurs vers les spécifications techniques du service Web. C'est un concept très important car tModel permet d'identifier les implémentations du service.

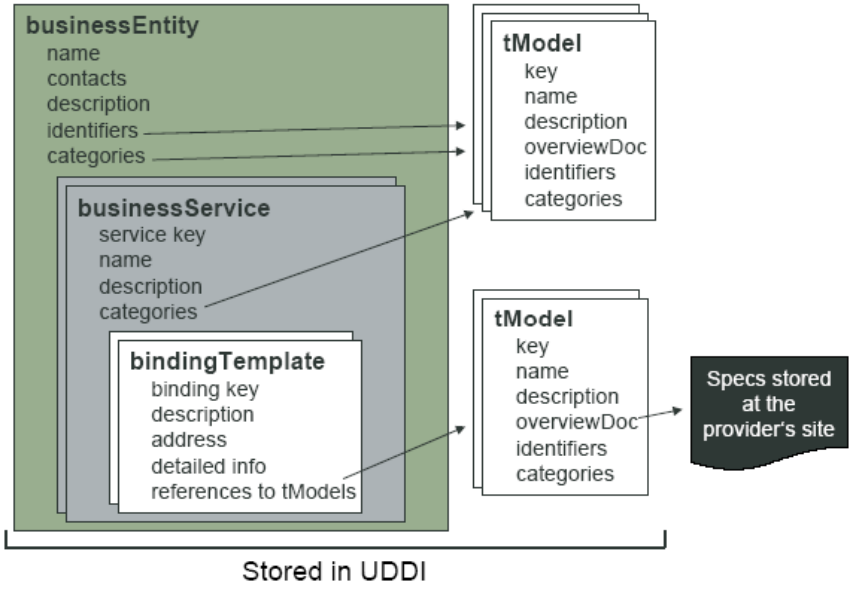


Figure 1.5 : Structure de données simplifiée d'UDDI.

Si deux businessEntity référencent le même tModel, cela garantit que les deux fournisseurs implémentent la même spécification du service Web. Les tModel peuvent pointer vers un fichier WSDL. Cela paraîtrait logique dans le sens d'un service Web défini autour du triplet UDDI/WSDL/SOAP. Mais, un service Web peut être mis en œuvre sur des modes de transport différents. Par exemple, les spécifications RosettaNet et ebXML (electronic business XML) ne s'appuient pas sur la spécification WSDL et sont néanmoins référencées dans l'annuaire UDDI. Si WSDL devient le standard universel de spécification d'un service, on peut penser que l'entité tModel se limitera, à terme, à un document WSDL.

- *publisherAssertion* : Permet à de grandes entreprises, ou à des groupements d'entreprises, ayant plusieurs domaines d'activités de montrer les relations qui existent entre leurs différentes businessEntity. Cette structure comporte trois champs : le premier champ fromKey qui référence la première businessEntity, le deuxième qui référence la seconde businessEntity et le dernier champ qui référence un tModel qualifiant la relation entre les deux businessEntity.

1.5 Utilisation d'un service Web

Pour utiliser un service Web, plusieurs étapes doivent être effectuées [PLA04]. Nous résumons ces étapes dans ce qui suit :

1. Localisation du service Web : cela peut être réalisé soit en explorant un registre UDDI publique ou au moyen d'un document WSDL existant. Il est possible aussi bien de créer un registre UDDI privé. Les registres privés sont faciles à maintenir en raison de leur taille mais il peut être difficile de découvrir la position du registre UDDI. Parfois, la page Web principale d'une entreprise est liée à des documents WSDL.
2. Création du message SOAP : elle est effectuée dans la plupart des cas par des outils de développement. Des outils comme *Weblogic Workshop* de BEA ou *Web service Development Kit* de Microsoft créent des messages SOAP valides pour les méthodes décrites dans le document WSDL ou le registre UDDI.
3. Transmission : transmission du message SOAP via HTTP.
4. Analyse du message SOAP : elle est effectuée par le serveur d'application du fournisseur. L'analyseur vérifie si la requête est valide et décide quelle procédure il doit appeler.
5. Traitement : le fournisseur de service appelle les procédures nécessaires ou même d'autres services Web pour compléter la tâche demandée.
6. retourner le résultat : le résultat de la requête est enveloppé dans une réponse SOAP et retourné au demandeur où l'application cliente peut analyser le message et évaluer les données incluses.

1.6 La sécurité des services Web

De nos jours les entreprises ont adopté définitivement les Services Web comme instrument nécessaire à l'intégration de leur métier dans la sphère de l'e-commerce. Les progrès rapides dans ce domaine sont dus à l'intérêt permanent des industriels mais aussi à la disponibilité immédiate d'outils et de standards permettant l'invocation automatisée des fonctions métiers via un échange de messages informatisés (SOAP, UDDI, WSDL et BPEL4WS).

Cependant, plusieurs enjeux doivent être pris en considération pour que les services Web deviennent une solution viable pour la construction des architectures orientées services. L'un de ces enjeux importants est la sécurité.

Dans les services Web, l'échange des informations est effectué via la technologie fondamentale non sécurisée et qui est la source de leur vulnérabilité. En outre, les clients doivent avoir une garantie sur l'intégrité des données envoyées par les fournisseurs de

services et ces derniers doivent avoir des informations de sécurité concernant les clients en raison de l'aspect dynamique des services Web et l'absence de connaissance à priori entre les différentes entités du système des services Web.

La sécurité des services Web nécessite la définition des différents aspects à assurer. Dans notre contexte, nous nous intéressons à la confidentialité et l'intégrité des informations échangées (contrôle de flux d'information) ainsi que l'authentification et l'autorisation (contrôle d'accès) abordés dans les prochains chapitres.

Plusieurs modèles et standards de sécurité ont été mis en œuvre pour répondre à la problématique de la sécurité. Les solutions de la sécurité des services Web concernent trois niveaux [MIC04]:

- Sécurité au niveau de la plateforme et du transport (point à point)
- Sécurité au niveau des messages (bout en bout)
- Sécurité au niveau de l'application (personnalisée)

Chaque méthode présente divers points forts et points faibles. Le choix de la méthode dépend en grande partie des caractéristiques de l'architecture et des plateformes impliquées dans l'échange de messages et des aspects de sécurité qu'on veut assurer.

1.6.1 Sécurité au niveau de la plateforme et du transport

Le modèle de sécurité au niveau du transport est un modèle simple, maîtrisé et adapté à un grand nombre de scénarios (principalement des scénarios impliquant des intranets) où les mécanismes de transport et la configuration des points de terminaison peuvent faire l'objet d'un contrôle étroit. Vu que les services Web utilisent le protocole HTTP pour le transport des messages, on peut appeler ce niveau "niveau de sécurité HTTP".

Un canal de transport placé entre deux points de terminaison (client de service Web et service Web) peut être utilisé pour fournir une sécurité point à point. La figure 1.6 ci-après illustre cette situation.

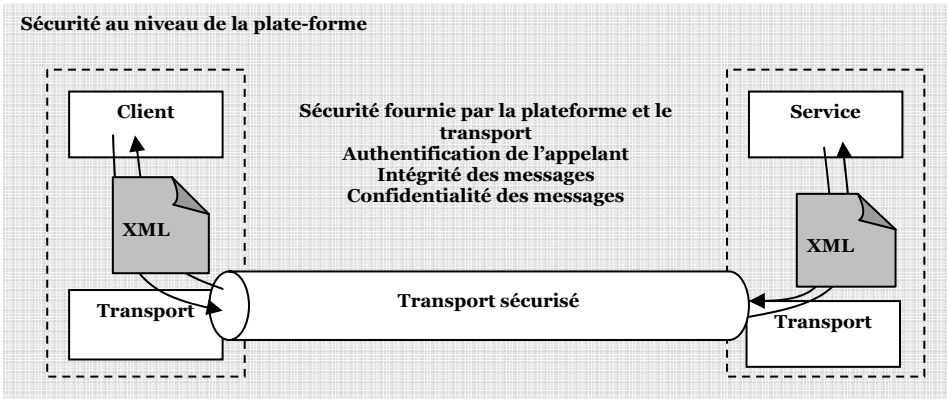


Figure 1.6 : Sécurité au niveau de la plateforme et du transport [MIC04]

La sécurité au niveau de transport devient étroitement intégrée (créant alors une dépendance) à la plateforme sous-jacente, au mécanisme de transport et au fournisseur de services de sécurité (NTLM, Kerberos, etc.). De plus, la sécurité s'applique en mode point à point et ne permet pas l'utilisation de plusieurs tronçons ni le routage de données via des nœuds d'applications intermédiaires.

Dans ce qui suit, nous présentons quelques solutions qui répondent aux exigences de la sécurité de ce niveau.

1.6.1.1 X.509

X.509 [ITT88] [HOU99] est un format standard pour les certificats à clés publiques. Ces derniers sont un élément principal dans la distribution et le transfert de confiance dans les clés publiques qui est lui-même la base d'autres transferts de confiance en utilisant la cryptographie à clé publique. L'objectif du certificat à clé publique est de distribuer les informations de clés publiques d'une manière sécurisée et bien contrôlée.
X.509 est basé sur le concept d'autorité de certification (Certifying Authority ou CA) qui vérifie l'identité d'une personne ou d'une entité qui a également prouvé sa possession de la clé privée. Le CA délivre alors un certificat, qui est une structure de données contenant (entre autres) l'identité et une copie de la clé publique, tout signé en utilisant la clé privée du CA.
X.509 est une recommandation et n'est pas encore défini pour une utilisation standard. Par conséquent, des entreprises l'ont implémenté de différentes manières. Ainsi, un certificat X.509 généré par une entreprise peut ne pas être lu par une autre.

1.6.1.2 Kerberos

Kerberos [KOH93] est un protocole d'authentification de réseau. Il est conçu pour fournir un mécanisme d'authentification aux applications client/serveur en utilisant la cryptographie à clé secrète.
Les composants du modèle Kerberos sont un serveur Kerberos (A Kerberos server), les serveurs d'attribution de ticket (Ticket-granting servers ou TGS), les serveurs d'application (Application servers) et les clients.

Pour utiliser un service, le client doit obtenir un ticket pour ce service auprès de serveurs d'attribution de tickets TGS. Mais, pour obtenir ce ticket, le client doit obtenir un ticket d'attribution de ticket (ticket–granting-ticket ou TGT) auprès du serveur d'authentification Kerberos AS. Chaque client partage une clé secrète avec le serveur d'authentification (AS)

qui est utilisée pour encrypter les clés de session générées par AS. AS partage une clé secrète avec TGS qui est utilisée pour encrypter le ticket fourni. TGS partage aussi des secrets avec les services pour lesquels il distribue des clés. Ceci permet au TGS de créer un ticket de service pour le client à présenter pour gagner l'accès au service correspondant.

1.6.1.3 Secure Socket Layer (SSL)

SSL est un protocole développé par Netscape [W3C02] pour transmettre des documents privés via Internet. Il fonctionne en utilisant une clé privée pour encrypter les données transférées sur des connections SSL. Plusieurs sites Web utilisent ce protocole pour obtenir les informations confidentielles de l'utilisateur tel que le numéro de carte de crédit.
L'objectif principal du protocole SSL est de garantir la confidentialité et la fiabilité entre deux applications communicantes.
Le protocole est composé de deux couches :

1. le *SSL Record Protocol* : au niveau le plus bas situé sur quelques protocoles de transport fiable d'Internet tel que TCP (Transmission Control Protocol). Il est utilisé pour encapsuler différents protocoles de haut niveau tel que le protocole suivant.
2. *le SSL Handshake Protocol* : permet au serveur et au client de s'authentifier et négocier l'algorithme et les clés de cryptographie avant la transmission et la réception des données.

Le protocole SSL assure la sécurité de la connexion qui a trois propriétés de base :

- ✓ la connexion est privée par l'utilisation de la cryptographie symétrique tel que DES (Data Encryption Standard) pour crypter les données.
- ✓ L'identité peut être authentifiée en utilisant la cryptographie asymétrique (à clé publique) tel que RSA (Rivest-Shamir-Adleman).
- ✓ La connexion est fiable. Le transport de message inclut le contrôle d'intégrité du message en utilisant MAC (Media Access Control) verrouillé.

1.6.1.4 Secure HTTP (S-HTTP)

S-HTTP est une extension du HTTP qui permet l'échange sécurisé de fichier sur Internet. Chaque fichier S-HTTP est soit encrypté soit il contient un certificat numérique ou les deux. La différence majeure avec SSL est que S-HTTP permet au client d'envoyer un certificat pour l'authentification de l'utilisateur alors qu'en utilisant SSL le but est d'encrypter la couche transport et d'authentifier le serveur. S-HTTP est utilisé dans les situations où le serveur représente une banque et exige l'authentification de l'utilisateur qui est plus sécurisée que l'identifiant de l'utilisateur et le mot de passe. SSL fonctionne au niveau du programme légèrement supérieur à la couche TCP. S-HTTP fonctionne aussi au niveau haut de

l'application HTTP. Les deux protocoles de sécurités peuvent être utilisés par un navigateur d'un utilisateur, mais un seul protocole peut être utilisé pour un document donné.

1.6.2 Sécurité au niveau des messages

Ce modèle de sécurité développe une structure d'échange de messages sécurisés dans un environnement de services Web hétérogène. Il convient parfaitement à des environnements hétérogènes et des scénarios où on ne contrôle pas directement à la fois la configuration des points de terminaison et celle des nœuds d'applications intermédiaires.
La sécurité au niveau des messages présente les caractéristiques suivantes :

- Elle peut être indépendante du transport sous-jacent.
- Elle autorise une architecture de sécurité hétérogène.
- Elle prend en charge la non répudiation.
- Elle prend en charge plusieurs technologies de cryptage.
- Elle offre une sécurité de bout en bout et facilite le routage des messages via des nœuds d'applications intermédiaires.

Cette méthode, sans doute est souple et puissante. La figure 1.7 ci-après illustre son fonctionnement

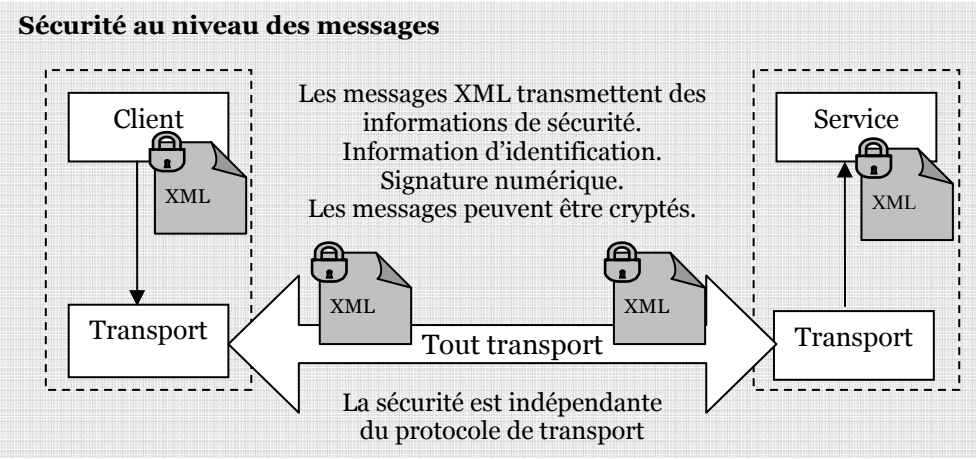


Figure 1.7 : Sécurité au niveau des messages [MIC04]

Dans ce qui suit, nous présentons quelques solutions qui se rapportent à la sécurité des messages.

1.6.2.1 XML Encryption

XML Encryption [W3C02] est une spécification de W3C qui décrit comment encrypter (chiffrer) des portions d'un document XML et comment chiffrer des données et les représenter sous format XML. Il garantit la confidentialité pour les services Web.

La plus petite unité d'un document XML qui peut être sélectivement chiffrée est l'élément de ce document.

La possibilité de chiffrer une partie d'un document XML est utile pour diverses raisons. Elle préserve la sécurité durant le routage entre différents services Web. Elle permet d'encrypter uniquement les informations sensibles telles que la charge utile et laisse quelques éléments tels que les informations de routage en texte clair. Le chiffrement sélectif a aussi l'avantage de productivité car le chiffrement est une opération coûteuse en temps de calcul et donc la limitation du processus de chiffrement uniquement aux données qui doivent être protégées augmente la performance. Ce standard permet la confidentialité de message non seulement pendant le transfert mais aussi au repos. Il utilise les algorithmes de cryptographie tels que RSA et Triple-DES.

1.6.2.2 XML Signature

XML Signature [EAS00] est un standard de W3C et IETF. Il est semblable à XML Encryption parce qu'il définit comment des parties d'un document XML peuvent être signées numériquement et comment signer n'importe quelles données et les représenter sous format XML. Ainsi, il a les mêmes avantages dus à la possibilité de sélectionner les éléments à signer. XML Signature offre aux services Web l'intégrité (i.e. la capacité de détecter si un message est altéré) et l'authentification. Il utilise les algorithmes de hachage tels que SHA-1

1.6.2.3 SAML

Le Securite Assertion Markup Language (SAML) [HAL02] est une spécification développée par OASIS (Organization for the Advancement of Structured Information Standards). Il définit une manière standard pour le transfert des assertions de sécurité entre les services exprimées en XML pour établir le lien et assurer l'interopérabilité entre les différents modèles de sécurité. Selon [BUR01], une assertion de sécurité est "une déclaration de vérité" qui peut être testée. Les assertions de sécurité contiennent des informations sur des entités (une personne, un agent, etc.), par exemple les identités, les attributs et les droits et peuvent être insérées dans les messages SOAP pour être utilisées dans l'authentification et les décisions d'autorisation. Quand un système A autorise une entité en se basant sur des informations d'identification passées par un système B veut dire que le système A fait confiance à B et suppose que B a effectué l'autorisation d'une manière correcte. Ceci est parfois désigné sous le nom de *"la confiance portative" (portable trust)*.

1.6.2.4 WS-Security

Le WS-Security [IBM02] [HON02] est une spécification développée sous l'OASIS à laquelle il a été soumis par Microsoft et IBM en juin 2002. Il peut être décrit comme un modèle définissant comment insérer dans l'entête du message SOAP les informations de sécurité et comment XML Encryption et XML Signature peuvent être utilisés pour protéger ces informations. Il étend donc SOAP pour fournir l'intégrité, la confidentialité et une authentification simple d'un message. Il fournit un niveau d'abstraction au dessus des technologies de sécurité différentes utilisées dans des organisations différentes.

Cette extension du standard SOAP vise à assurer une sécurité de bout en bout plutôt que la sécurité point à point comme pour SSL par exemple. Il utilise les standards de sécurité du niveau XML particulièrement XML Encryption et XML Signature.

SAML est donc utilisé pour exprimer les assertions de sécurité et WS-Security définit comment les insérer dans un message SOAP.

1.6.3 Sécurité au niveau de l'application

Les solutions de sécurité au niveau application visent d'une manière générale à contrôler les accès des clients aux services Web (authentification et autorisation ou contrôle d'accès) et assurer la confidentialité et l'intégrité des données manipulées par ces derniers (contrôle de flux d'information).

Avec ces méthodes, l'application se charge de la sécurité en faisant appel à des fonctions de sécurité personnalisées. Par exemple une application peut utiliser un entête SOAP pour transférer les informations d'identification de l'utilisateur afin d'authentifier ce dernier à chaque demande de service Web et de prendre des décisions d'autorisation.

Dans ce qui suit, nous définissons brièvement les deux techniques de sécurité de niveau application à savoir le contrôle d'accès et le contrôle de flux d'information.

1.6.3.1 Le contrôle d'accès et le contrôle de flux d'information

L'objectif du contrôle d'accès est de limiter les actions qu'un utilisateur légitime d'un système informatique ou un programme s'exécutant au profit des utilisateurs peuvent effectuer. Le contrôle d'accès [SAN94] définit qu'est ce qu'un sujet peut effectuer directement sur les objets du système. Le sujet peut être un être humain, une application cliente, un agent logiciel, un service Web, etc.

En général, le contrôle d'accès tire profit des services de sécurité des couches inférieures pour accomplir ses tâches. Par exemple, il a besoin d'établir correctement les identités des

utilisateurs qui est la tâche du service d'authentification. Pour les services Web, le contrôle d'accès utilise les solutions de sécurité du niveau transport et du niveau messages.

Le standard le plus connu dans le domaine de contrôle d'accès pour les services Web est **XACML** (**X**ML **A**ccess **C**ontrol **M**arkup **L**anguage) [GOD02]. C'est une spécification d'OASIS pour représenter d'une manière standard les politiques (règles) de contrôle d'accès dans un format XML. Il permet aussi d'inclure ces politiques dans les messages envoyés et elle permet de délivrer les assertions des droits avec les assertions des identités.

Le contrôle de flux d'information vise à assurer que les transferts d'information à travers les systèmes informatiques sont effectués d'une manière sécurisée i.e. le transfert se fait toujours d'une source de niveau de sécurité bas vers une destination de niveau de sécurité haut. Ainsi, il empêche la divulgation d'information. Le contrôle de flux d'information permet l'application des politiques de sécurité de bout en bout et assure la confidentialité et l'intégrité des données transmises.

Le contrôle d'accès permet de contrôler les tentatives d'accès direct aux informations mais le contrôle de flux d'informations est nécessaire pour contrôler la propagation de ces informations (après leur dissémination).

1.7 Conclusion

Les services Web sont une technologie relativement nouvelle qui suscite un grand intérêt dans l'industrie du traitement de l'information. Cette technologie est encore en développement et de nombreuses nouveautés sont attendues dans le domaine des outils de développement et de déploiement comme dans celui des spécifications. Le développement d'applications utilisant les services Web deviendra ainsi de plus en plus aisé, jusqu'à ce que les couches sous-jacentes mettant en oeuvre ces services finissent par être totalement invisibles pour le développeur.

Nous avons introduit dans ce chapitre les services Web en définissant la notion d'un service Web, l'architecture et l'infrastructure de cette technologie et les standards les plus utilisés pour la mise en œuvre des services Web. Nous avons également présenté quelques standards et recommandations de sécurité pour les services Web classés en trois niveaux : le niveau de la plateforme, le niveau des messages et le niveau application.
Nous avons défini brièvement les deux techniques de sécurité du niveau application. Notre solution de sécurité décrite dans ce mémoire est un modèle de contrôle d'accès et de contrôle de flux d'information. Pour cela, ces deux techniques feront l'objet des prochains chapitres pour en parler d'une manière générale et particulièrement pour les services Web.

CHAPITRE 2

Contrôle d'accès et gestion de confiance pour les services Web

2.1 Introduction

Dans les systèmes informatiques, quand une requête pour un certain service est reçue par un principal (agent) de la part d'un autre principal, le récepteur se pose deux questions. Premièrement, est ce que le principal demandeur est celui qu'il prétend être ? Deuxièmement, est ce que ce principal a le privilège approprié pour ce service ? Ces deux questions sont relatives aux enjeux de l'authentification et de l'autorisation (contrôle d'accès).
L'authentification est le processus qui consiste à vérifier l'identité d'une entité. Elle nécessite des preuves appelées *informations d'identification* (par exemple un mot de passe fourni par une application cliente). Une fois que l'identité d'une entité est authentifiée, des autorisations d'accès peuvent être accordées en comparant les informations concernant l'entité avec des informations de contrôle d'accès (par exemple ACL : Access Control List).

Le contrôle d'accès est une solution de sécurité du niveau application. Il constitue une composante essentielle dans n'importe quel système informatique Multi-Utilisateurs pour le partage et la protection des informations et les ressources physiques.
Ce mécanisme de sécurité est implémenté par les fournisseurs de services pour gérer les accès des utilisateurs aux informations du système en se basant sur les identités des utilisateurs et un ensemble de règles (ou autorisations) qui déterminent, pour chaque utilisateur et chaque objet dans le système, les types des accès (exemple : lire, écrire ou exécuter) que l'utilisateur peut effectuer sur l'objet.

Dans ce chapitre, nous abordons d'une manière générale les modèles de contrôle d'accès traditionnels, puis leurs améliorations et adaptations aux services Web en tenant compte de la notion de gestion de confiance (*trust management*).

2.2 Définitions

Dans chaque système informatique, se retrouvent deux classes d'entités :

- *Les entités actives ou les sujets* : tels que les utilisateurs et les programmes qui s'exécutent au nom des utilisateurs.
- *Les entités passives ou les objets* : tels que les dossiers, les fichiers, les imprimantes, etc.

La manière dont les sujets accèdent aux objets est appelée *permission* (aussi appelée *mode d'accès* ou *privilège*). La permission est une autorisation d'effectuer une action particulière sur le système. Il s'agit en général de la combinaison d'un sujet, objet et d'une opération. Le privilège d'accès permet aux sujets :

- De manipuler les objets par les opérations de lecture, écriture, exécution, etc.
- De modifier les informations de contrôle d'accès ou les privilèges d'accès.

Les mécanismes de contrôle d'accès doivent définir des politiques (règles) différentes. Le choix de ces politiques de sécurité influe sur la flexibilité et les performances d'un système de contrôle d'accès. Leur conception doit donc prendre en compte quelques principes [BAR96] tels que le principe de privilège minimum et maximum, le principe de système ouvert ou fermé, le principe d'administration centralisée ou décentralisée, le principe de granularité et le principe de privilège d'accès.

2.3 Les formes de contrôle d'accès

Généralement, les mécanismes ou les modèles de contrôle d'accès peuvent avoir une ou une combinaison de deux formes :
Discrétionnaire (discretionary) ou mandataire (mandatory) [YUA05].

2.3.1 Le contrôle d'accès discrétionnaire

Le contrôle d'accès discrétionnaire (**DAC** pour **D**iscretionary **A**ccess **C**ontrol) est émergé avec les systèmes en temps partagé dans les années 1970. Les idées de bases sont introduites par Lampson [LAM73].
DAC est une manière de restreindre l'accès aux objets en se basant sur les identités et la connaissance préalable de l'utilisateur, du processus et/ou le groupe auquel il appartient. Les contrôles sont discrétionnaires dans le sens où un sujet avec une certaine permission d'accès est capable de passer cette permission à un autre sujet. En d'autres termes, le sujet qui est propriétaire d'un objet peut déterminer quels sujets peuvent accéder à cet objet et avec quels privilèges d'accès. Dans cette forme de contrôle d'accès, la manière dont les objets sont

accédés par des sujets est spécifiée par des règles d'accès explicites. DAC est une forme de contrôle d'accès largement utilisée dans les systèmes informatiques et réseaux. Par exemple, dans le système d'exploitation Windows, le propriétaire d'un fichier ou d'un dossier peut accorder ou refuser l'accès d'un utilisateur ou un groupe d'utilisateurs.

DAC est basé sur la matrice de contrôle d'accès (Access Control Matrix : ACM). Le modèle de matrice a été développé par Lampson [LAM74] et étendu par Graham et Denning [GRA72]. Par la suite, Harrisson, Ruzzo et Ullman [HAR76] ont développé une version plus générale du modèle.

Cette matrice est définie comme suit : les lignes correspondent aux identités des sujets et les colonnes correspondent aux objets. Chaque entrée de cette matrice (intersection d'une ligne s et d'une colonne o) contient les privilèges de sujet s pour l'objet o.
Dans de grands systèmes, cette matrice devient énorme en taille et plusieurs de ces entrées sont vides. Par conséquent, cette matrice est rarement implémentée comme une matrice mais plusieurs autres formes sont utilisées telles que ACL (Access Control List), CL (Capability List) et AR (Authorization Relations) [SAN94] (cf. § 3.4.1).

Les politiques de contrôle d'accès discrétionnaire n'offrent pas une vraie assurance sur le flux d'information dans le système. Par exemple, un sujet qui est capable de lire un objet peut le passer à un autre sujet non autorisé à le lire sans la connaissance du propriétaire de cet objet.

2.3.2 Le contrôle d'accès mandataire

MAC (Mandatory Access Control) est originellement formalisé par Bell et LaPadula [BEL75]. MAC est une manière de restreindre l'accès aux objets en se basant sur des attributs de sécurité ou "labels" associés aux sujets et aux objets. Un label d'un objet est souvent appelé une "classification ou niveau de sécurité" et le label d'un sujet est appelé "habilitation". Les contrôles sont mandataires dans le sens où elles sont appliquées par le système et ne peuvent pas être modifiés par les utilisateurs ou leurs programmes. MAC est utilisé quand la politique de sécurité dicte qu'un propriétaire d'une ressource ne doit pas prendre les décisions d'autorisation, cette situation est souvent retrouvée dans les systèmes gouvernementaux et les systèmes de défense.

Un sujet peut accéder à un objet donné si une relation d'ordre (selon le mode d'accès) est satisfaite entre leurs deux niveaux de sécurité. Particulièrement, il y a deux principes à respecter :
Lire vers le bas (read down) : l'habilité de sujet doit dominer (≥) le niveau de sécurité de l'objet à lire.

Ecrire vers le haut (write up) : l'habilité de sujet doit être dominée (≤) par le niveau de sécurité de l'objet à écrire.
La satisfaction de ces deux principes empêche les informations des objets de haut niveau de sécurité (i.e. plus sensible) d'être transférées aux objets de niveaux de sécurité inférieurs. Dans ce type de systèmes de contrôle d'accès, l'information peut uniquement être transférée vers le haut ou au même niveau de sécurité.

Les formes DAC et MAC ne sont pas mutuellement exclusives. En effet, plusieurs systèmes utilisent MAC et DAC en conjonction [YUA05]. Dans ce cas, les contrôles mandataires sont vérifiés suivis des contrôles discrétionnaires ou les contrôles discrétionnaires sont appliqués puis les contrôles mandataires sont vérifiés.

2.4 Modèles de contrôle d'accès

Il y a plusieurs modèles de contrôle d'accès qui implémentent DAC et/ou MAC. Dans ce qui suit, nous allons décrire brièvement trois modèles traditionnels qui sont souvent utilisés.

2.4.1 Contrôle d'accès basé sur l'identité

Le contrôle d'accès basé sur l'identité (Identity Based Access Control : IBAC) utilise l'identité du sujet (exemple le nom de l'utilisateur) pour attribuer les permissions d'accès aux ressources. Les exemples de ce type de contrôle d'accès sont ceux dérivés de la matrice d'accès et qui utilisent la forme DAC.

2.4.1.1 Access Control List (ACL)

La liste de contrôle d'accès est l'une des formes utilisées pour implémenter la matrice d'accès [SAN94]. Elle est généralement utilisée dans les systèmes d'exploitation et les services de sécurité réseaux.

Chaque objet est associé à une ACL qui indique pour chaque sujet du système quels sont les accès autorisés à cet objet. Cette approche correspond à la sauvegarde de la matrice d'accès en colonnes. La figure 2.1 montre un exemple d'ACL pour deux objet (Fichier1 et Fichier2) et des sujets X, Y et Z représentant trois utilisateurs. Les modes d'accès *R*, *W* et *Own* correspondent respectivement à *lire*, *écrire* et *propriétaire*.

Avec ACL, il est facile de déterminer les privilèges d'accès associés à chaque sujet pour un objet donné. Cependant, pour déterminer tous les accès permis à un sujet dans le système, toutes les ACLs doivent être parcourues.

Certains systèmes (tels que UNIX) utilisent les noms de groupes au lieu des identités des utilisateurs.

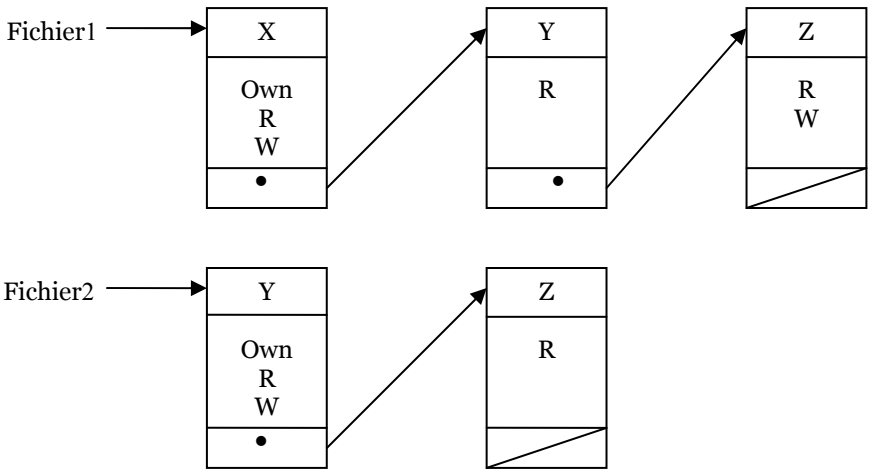


Figure 2.1 : Exemple de liste de contrôle d'accès (ACL)

2.4.1.2 Capability List (CL)

La liste de capacités ou d'aptitudes (CL) est une approche duale à ACL. Chaque sujet est associé à une CL qui indique quels sont les accès autorisés pour chaque objet du système. Cette approche correspond à la sauvegarde de la matrice d'accès par lignes.
La figure 2.2 montre les CLs correspondantes aux ACLs de la figure 2.1.

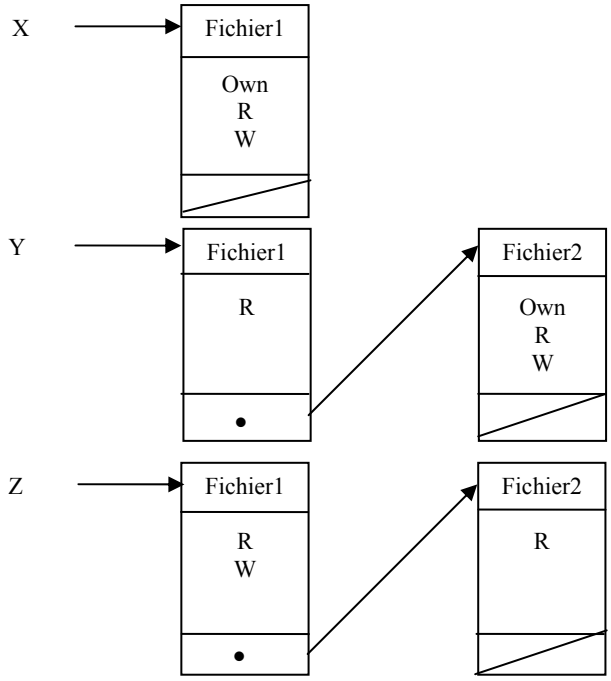


Figure 2.2 : Exemple de liste de capacités (CL)

Avec CL, il est facile de déterminer tous les privilèges d'accès associés à un sujet. Cependant, la détermination de tous les sujets dans le système qui peuvent accéder à un objet donné exige de parcourir toutes les CLs.

En utilisant cette approche dans les systèmes distribués, l'authentification répétée du sujet n'est pas nécessaire. Un sujet est authentifié une seul fois, obtient sa CL puis l'utilise pour obtenir les services auprès de différents serveurs du système.

2.4.1.3 Authorization Relations (AR)

Il est mentionné précédemment que ACL et CL ont des avantages et des inconvénients duaux par rapport à l'accès des listes. La représentation de la matrice d'accès par une table de relations d'autorisation ne favorise aucun type d'accès (par rapport aux sujets ou objets). Chaque ligne de cette table spécifie un seul mode d'accès d'un sujet à un objet.

Par conséquent, AR a les effets de ACL et CL.

Cette représentation est typiquement utilisée dans les systèmes de gestions de bases de données relationnelles.

La table 2.1 montre la table de relations d'autorisation correspondante à l'ACL et la CL de l'exemple précédent.

Sujet	Mode d'accès	Objet
X	Own	Fichier1
X	R	Fichier1
X	W	Fichier1
Y	R	Fichier1
Y	Own	Fichier2
Y	R	Fichier2
Y	W	Fichier2
Z	R	Fichier1
Z	W	Fichier1
Z	R	Fichier2

Table 2.1 : Exemple de table de relations d'autorisation (AR)

Le problème majeur des modèles IBAC est le nombre des identités qui augmente avec l'augmentation des utilisateurs dans un grand système et qui devient difficile à maintenir. De plus, les décisions de contrôle d'accès ne prennent pas en considération les caractéristiques et les fonctions des utilisateurs mais uniquement leurs identités. Ce type de modèles s'avère très limité pour être appliqué dans les systèmes distribués et ouverts tels que les services Web où les identités des utilisateurs ne sont pas connues à l'avance.

2.4.2 Contrôle d'accès basé sur un treillis

Le modèle de contrôle d'accès basé sur un treillis ou Lattice Based Access Control (LBAC) a été proposé dans par Bell et LaPadula [BEL75] dans les années 70. Puis, implémenté par le secteur de défense des USA. LBAC impose un flux d'information unidirectionnel dans un treillis de labels de sécurité. LBAC adresse le problème MAC et il est aussi appelé modèle de sécurité à multiniveaux. Selon la nature du treillis, le flux unidirectionnel d'information peut assurer la confidentialité et/ou l'intégrité.

Comme il a été mentionné pour la forme MAC, la politique de sécurité est exprimée en terme de niveaux de sécurité attachés aux sujets (habilité) et aux objets (classification de sécurité).

Sous LBAC, l'ensemble de niveaux de sécurité totalement ordonné (exemple TS > S > C > U) peut être combiné avec un ensemble de catégories (e.g. X, Y, Z) pour former un treillis. Dans ce cas, un label de classification associé à chaque sujet et à chaque objet consiste en une paire composée d'un niveau de sécurité et d'un ensemble de catégories. L'ensemble de catégories associé à un sujet spécifie les secteurs dans lesquels le sujet travaille, et celui associé à un objet spécifie les secteurs auxquels l'information contenue dans cet objet est référée.

Il est à noter qu'un utilisateur secret (S) peut exécuter un même programme (tel que l'éditeur de texte) qu'un sujet secret (S) ou un sujet non classifié (U). Malgré que ces deux sujets exécutent le même programme au profil du même utilisateur, ils obtiennent des privilèges différents dus à leurs labels de sécurité.

Les labels de sécurité forment un treillis comme il est défini ci-dessous [SAN96] :

- **Treillis de sécurité :** il y a un treillis fini SC de labels de sécurité avec une relation de dominance partiellement ordonné (≥) et l'opérateur de borne supérieure.

Un exemple de treillis de sécurité est représenté dans la figure 2.3. L'information peut uniquement se déplacer vers le haut du treillis.

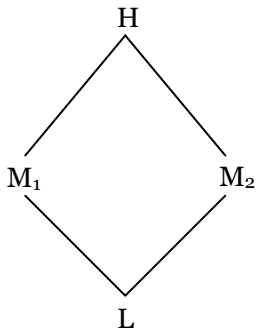


Figure 2.3 : un treillis partiellement ordonné

Dans cet exemple, H (High) et L (Low) correspondent respectivement à haut et bas. M_1 et M_2 sont deux labels incomparables et intermédiaires à H et L. C'est un treillis de confidentialité où l'information circule de bas en haut.

Les règles d'accès mandataire souvent spécifiées pour un treillis sont décrites dans ce qui suit où **λ** signifie le label de sécurité du sujet ou de l'objet indiqué.

- *Sécurité simple :* le sujet s peut lire l'objet o seulement si λ(s) ≥ λ(o).
- *La propriété * libérale (large) :* le sujet s peut écrire l'objet o seulement si λ(s) ≤ λ(o).

Cette propriété permet au sujet de bas niveau d'écrire un objet de haut niveau. Ainsi, les données de haut niveau peuvent être malicieusement détruites par un sujet de bas niveau.
Pour éviter ce genre de situations, la *propriété * stricte* est définie.

- *La propriété * stricte :* le sujet s peut écrire l'objet o seulement si λ(s) =λ(o).

La propriété * large et la propriété * stricte sont aussi appelées respectivement Write-up et Write-equal

Généralement, LBAC est utilisé quand il y a un petit nombre de niveaux de sécurité (n) et de catégories (m) statiques tel que le nombre total de combinaisons de niveaux de sécurité et de catégories est n * m.

2.4.3 Contrôle d'accès basé sur les rôles

Le modèle de contrôle d'accès basé sur les rôles ou Role Based Access Control (RBAC) est introduit par Sandhu et al. [SAN+96]. Il est défini pour pallier les limitations des modèles IBAC et LBAC. En effet, d'une part, RBAC ne se base pas uniquement sur les identités des utilisateurs mais ajoute un niveau d'indirection qui est le *rôle* entre les identités et les ressources, ce qui permet l'extensibilité du modèle et réduit significativement la complexité de l'administration. D'autre part, il n'impose aucun type de politiques d'accès mais supporte n'importe quelles politiques pour gérer l'accès aux ressources.

Les politiques du RBAC permettent de spécifier les autorisations accordées aux utilisateurs (ou groupes d'utilisateurs) sur les objets comme dans l'approche discrétionnaire avec la possibilité de spécifier des restrictions sur l'assignation et l'utilisation de telles autorisations comme dans l'approche mandataire [SAN94].

Le modèle RBAC [SAN96] est basé sur trois ensembles d'entités :

- ***Les utilisateurs (U) :*** qui peuvent être des êtres humains ou des agents autonomes.
- ***Les rôles (R) :*** fonctions ou travaux dans une entreprise avec quelques sémantiques concernant l'autorité et les responsabilités conférés à un membre d'un rôle.
- ***Les permissions (P) :*** approbations des modes d'accès particuliers aux objets du système.

Le modèle RBAC est basé sur deux relations d'assignations :

- ***L'assignation des utilisateurs (UA : User Assignment) :*** c'est une relation N-M (doubles flèches dans la figure 2.4) qui permet de spécifier le rôle d'un utilisateur. Un utilisateur peut être membre de plusieurs rôles et un rôle peut avoir plusieurs utilisateurs.
- ***L'assignation des permissions (UA : Permission Assignment) :*** c'est aussi une relation N-M qui permet de spécifier les permissions accordées à un rôle. Un rôle peut avoir plusieurs permissions et une permission peut être assignée à plusieurs rôles.

Il y a aussi une hiérarchie des rôles partiellement ordonnée :

- ***RH (Role Hierarchy) :*** aussi écrite avec l'opérateur ≥ où x ≥ y signifie que le rôle x hérite les permissions assignées au rôle y. L'héritage le long de la hiérarchie est transitif et l'héritage multiple est permis dans les ordres partiels.

Le modèle RBAC présente deux autres concepts qui sont la session et les contraintes :

- ***La session :*** une session relie un utilisateur à un ou plusieurs rôles. Un utilisateur établit une session durant laquelle il peut activer les rôles dont il est membre (directement ou indirectement à travers la hiérarchie). Plusieurs rôles peuvent être activés simultanément. Une session est associée à un seul utilisateur. Les permissions disponibles pour un utilisateur sont l'union des permissions de tous les rôles activés pendant sa session. Un utilisateur peut ouvrir en même temps plusieurs sessions.
- ***Les contraintes :*** elles peuvent s'appliquer à chacun des composants précédents. Un exemple de contraintes est celui des rôles mutuellement disjoints tels que le gestionnaire des achats et le gestionnaire des comptes à payer où un même utilisateur ne peut pas être membre de ces deux rôles.

Les concepts du modèle RBAC sont résumés dans la figure 2.4 et il peut être formalisé comme suit :

- U, R, P et S, sont les ensembles d'utilisateurs, de rôles, de permissions et de sessions respectivement.
- PA $\subseteq$ P $\times$ R, est une relation N-M d'assignation de permissions aux rôles.
- UA $\subseteq$ U $\times$ R, est une relation N-M d'assignation d'utilisateurs aux rôles.
- RH $\subseteq$ R $\times$ R, une hiérarchie des rôles partiellement ordonnée (notée ≥).
- *user* : S $\rightarrow$ U, une fonction qui associe à chaque session s_i un seul utilisateur, *user* (s_i) (constant pendant la durée de vie de la session).
- *roles* : S $\rightarrow 2^R$, une fonction qui associe à chaque session s_i un ensemble de rôles, *roles* (s_i) *roles* (s_i) $\subseteq$ {r / ($\exists$ r' ≥ r) [(*user*(s_i), r'$\in$ UA]} (qui peut changer avec le temps).
  Tel que les permissions de la session s_i sont données par :

  $$\bigcup_{r \in roles(si)} \{p / (\exists\ r'' \leq r)\ [(p, r'') \in PA]\}$$

- une collection de contraintes qui déterminent si les valeurs des différentes composantes du modèle RBAC sont acceptables ou non.

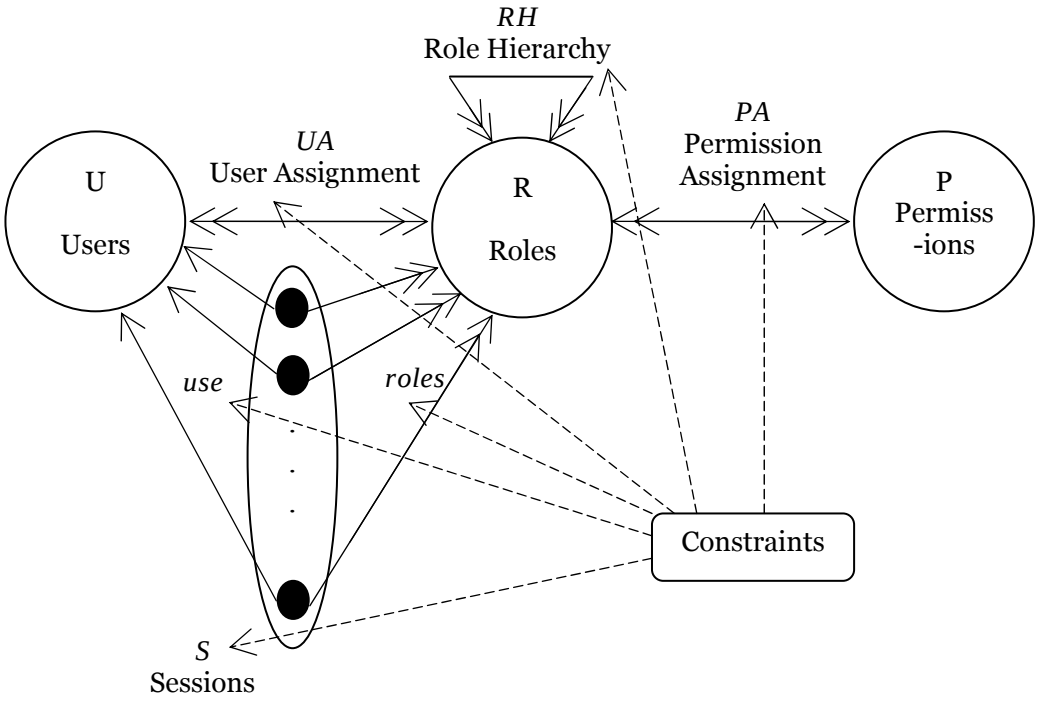


Figure 2.4 : Le modèle RBAC [SAN96]

2.5 Gestion de confiance et contrôle d'accès pour les services Web

Dans les services Web, les systèmes collaboratifs exigent d'une part que les informations soient partagées et accessibles et d'une autre part, les organisations doivent continuer à protéger les informations sensibles et confidentielles en ne permettant l'accès à ces dernières qu'au personnel autorisé. Par conséquent, des modèles d'accès fiables et flexibles sont nécessaires.

Les modèles d'accès traditionnels (LBAC, RBAC, etc.) sont conçus pour les applications basées sur les réseaux LAN tels que les applications bancaires qui sont des systèmes fermés. Ces modèles ne sont pas convenables pour les systèmes ouverts comme les services Web vu leur aspect distribué et décentralisé sauf s'ils sont étendus pour traiter les exigences des services Web [TAR04]. En effet, l'application de ces contrôles se base principalement sur les identités et les attributs des utilisateurs connus en utilisant un moniteur de références et des

règles d'autorisation spécifiques. Cependant, les utilisateurs dans les environnement ouverts, distribués et dynamiques ne sont pas connus au préalable. Les systèmes e-book et les distributions de fichiers de musiques sont des exemples de systèmes où les utilisateurs sont étrangers.

Avec l'arrivée de l'infrastructure à clé publique, la recherche récente dans les autorisations pour les utilisateurs étrangers ont été poursuivies sous le nom de "***la gestion de confiance***" [BLA96] [HER00] [WEE01]. T. GRANDISON et al. ont fait dans [GRA00] une étude et établit un état de l'art sur la confiance et les systèmes de gestions de confiance dans les applications Internet.

Dans la plupart des cas, la gestion de confiance utilise les capacités (aptitudes) ou les propriétés de l'utilisateur sous forme de qualifications numériques ou certificats [PAR04].

Plusieurs modèles ont été proposés dans la littérature pour définir des modèles de contrôle accès efficaces pour les environnements distribués des services Web. Certains de ces modèles adaptent et étendent les modèles traditionnels comme RBAC en introduisant des mécanismes de gestion de confiance tels que les attributs des demandeurs de services Web et la réputation de ces derniers.

Dans les sections suivantes, nous citons quelques travaux de recherches relatifs à notre problématique et nous présentons quelques exemples de modèles de contrôle d'accès conçus pour les services Web.

2.5.1 Les travaux antérieurs

Les modèles de contrôle d'accès proposés pour les services Web se distinguent par les concepts qu'ils utilisent pour l'authentification et la définition des politiques d'autorisation. Nous pouvons classifier ces modèles en trois classes ; les modèles basés sur les rôles, les modèles de contrôle d'accès basés sur les attributs et les modèles hybrides.

2.5.1.1 Les modèles basés sur les rôles

Ces modèles adaptent la notion de rôle introduite par le modèle de contrôle d'accès traditionnels RBAC. Ces modèles regroupent les fonctionnalités disponibles dans un système de services Web sous forme de rôles et assignent les utilisateurs à ces rôles selon des politiques prédéfinies. Les autorisations d'accès aux services Web sont définies et assignées aux rôles d'un système. Ainsi, un utilisateur peut accéder à un service Web si le rôle auquel il est assigné est autorisé à invoquer ce service Web.

Tari et al. ont proposé dans [TAR04] un modèle de contrôle d'accès pour les services Web qui supporte des politiques de contrôle d'accès pour les services Web globaux ou les services Web composés. Le modèle consiste en une adaptation du modèle RBAC (Role-based Access

Control) où les utilisateurs sont assignés aux rôles pour utiliser les services Web. Le contrôle d'accès s'effectue à deux niveaux : au niveau des services Web et au niveau de leurs attributs. Les autorisations sont définies sur des services simples et leurs attributs et sur les services composés (services conçus en utilisant d'autres services).

Un modèle général pour contrôler le comportement des utilisateurs en se basant sur leurs rôles est proposé par C. XU et al. [XU 01]. Dans le modèle, la notion de Role-Playing est introduite pour désigner le rôle activé dans un contexte particulier. Le role-playing est modélisé comme une classe et ses objets (instances) interagissent avec l'utilisateur et contrôlent ses comportements d'une manière active. Pour déployer le modèle RBAC dans le Web, les auteurs ont proposé d'utiliser les cookies sécurisées pour protéger les informations concernant l'utilisateur.

R. Bhatti et al. ont proposé X-RBAC (XML-based Role Based Access Control) [BHA+03]. Le modèle spécifie les politiques du RBAC en se basant sur XML pour appliquer le contrôle d'accès dans les services Web dynamiques. Les auteurs ont par la suite étendu le modèle pour supporter la notion de confiance pour répondre aux exigences de la sécurité des services Web. Dans [BHA03] le modèle X-GTRBAC (**X**ML-based **G**eneralized **T**emporal **R**ole **B**ased **A**ccess **C**ontrol) est proposé comme extension du modèle X-RBAC pour la prise en considération des contraintes temporelles dans l'assignation des utilisateurs aux rôles et l'assignation des permissions aux rôles. R. Bhatti et al. ont montré dans |BHA04] comment configurer X-GTRBAC pour l'intégration des contraintes contextuelles dans le contrôle d'accès.

Les modèles basés sur la notion de rôle sont extensibles et faciles à administrer car les utilisateurs n'accèdent pas directement aux services et aux objets du système mais passent par un niveau d'abstraction qui est le rôle. Cependant, ils ne proposent aucun mécanisme de gestion de confiance pour pallier les limites du modèle RBAC traditionnel. RBAC utilise les identités des utilisateurs qui ne sont pas connues au préalable dans les systèmes des services Web.

2.5.1.2 Les modèles basés sur les attributs

Dans ce type de modèles, la notion de gestion de confiance est supportée en utilisant différents attributs de l'utilisateur au lieu de se baser uniquement sur leurs identités et/ou mots de passe qui ne sont pas souvent connus à l'avance dans les systèmes distribués comme les services Web.

S. Hai-bo et H. Fan [HAI06] ont proposé un modèle pour l'authentification et le contrôle d'accès aux services Web en se basant sur les attributs des entités (WS-ABAC). Le modèle fournit aussi un mécanisme de négociation de confiance pour protéger les attributs. Ce

modèle permet aux fournisseurs et aux demandeurs de services l'établissement de confiance en se basant sur les attributs en plus des identités.

Un modèle semblable à WS-ABAC a été proposé par E. Bertino et al. [BER04]. Il utilise la notion des attributs des individus et les paramètres des services Web. Les politiques de contrôle d'accès contiennent les conditions sur les attributs des utilisateurs et les paramètres des services invoqués. Si une requête correspond à une politique d'accès, le service Web correspondant est invoqué. Le modèle fournit un mécanisme de négociation d'accès qui permet aux utilisateurs de changer dynamiquement leurs requêtes d'accès si ces dernières sont mal formulées.

Eric Yuan et Jin Tong ont présenté dans [YUA05] un modèle de contrôle d'accès en se basant sur les identités des sujets, des ressources et de l'environnement. Leur modèle utilise des fonctions booléennes pour la spécification des politiques de contrôle d'accès. La fonction booléenne est une combinaison de contraintes sur les attributs des utilisateurs, les attributs des ressources et les attributs de l'environnement. La notion de rôle dans ce modèle est vue comme un attribut de l'utilisateur.

Les modèles de contrôle d'accès qui utilisent les attributs des utilisateurs et d'autres attributs comme les attributs des ressources et les attributs du contexte permettent la gestion de confiance et fournissent un mécanisme puissant pour l'authentification dans les systèmes distribués et ouverts. Cependant, les mécanismes d'authentification et les mécanismes d'autorisation et d'assignation de permissions sont confondus, ce qui complique l'administration et l'extensibilité des modèles.

2.5.1.3 Les modèles hybrides

Nous avons appelé ces modèles "modèles hybride" car ils utilisent différents concepts pour définir des modèles de contrôle d'accès fiables qui regroupent un ensemble de caractéristiques nécessaires pour le contexte des services Web.

Un exemple de ces modèles est le modèle TrustBAC qui a été proposé par S. Chakraborty et I. Ray [CHA06]. TrustBAC adapte le modèle RBAC aux systèmes ouverts et décentralisés en intégrant la notion de confiance (Trust). Dans TrustBAC, les utilisateurs sont assignés aux *niveaux de confiance* au lieu des rôles en se basant sur un nombre de facteurs comme les informations d'identification des utilisateurs, l'historique du comportement des utilisateurs et la recommandation sur les utilisateurs (réputation). Les niveaux de confiance sont assignés aux rôles qui sont assignés aux permissions comme dans RBAC. Le changement dans le niveau de confiance de l'utilisateur change les rôles de l'utilisateur dans le système et par conséquent ses privilèges.

Ainsi, TrustBAC intègre les avantages du modèle RBAC et ceux du modèle de contrôle d'accès basé sur les informations d'identification (CBAC : Credential Based Access Control). Toutefois, le modèle n'exprime pas les règles d'autorisation ou d'assignations des permissions aux rôles.

M. LIU, H. GUO et J. SU [LIU05] ont définit le modèle ARBAC (an Attribute and Role based Access Control) pour le contrôle d'accès aux services Web en intégrant la notion d'attributs avec la notion de rôle du modèle RBAC. Ainsi, le modèle permet la gestion de confiance et facilite l'administration du système de contrôle d'accès. Le modèle utilise les attributs des utilisateurs pour l'authentification et l'assignation des utilisateurs aux rôles prédéfinis dans le système. Les ressources du système pour lesquels les accès sont contrôlés incluent les méthodes des services Web et les données qu'elles manipulent. ARBAC prend en compte les conditions sur le contexte d'exécution d'une requête et la notion de séparation des tâches.
Le modèle ARBAC permet un contrôle d'accès à granularité fine en séparant l'ensemble des méthodes des services Web et l'ensemble des données du système. Il permet un contrôle plus précis en utilisant des conditions concernant le contexte et il empêche l'activation des rôles contradictoire en introduisant la séparation statique et la séparation dynamique des tâches. Le modèle s'avère très adéquat pour le contexte des services Web, mais il n'effectue pas un contrôle sur les flux d'informations entre les objets manipulés par les services Web.

L'absence de contrôle de flux d'information entre les objets d'un système de services Web constitue la limite de la plupart des modèles de contrôle d'accès trouvés dans la littérature. En effet, ces modèles contrôlent les accès directs aux ressources d'un système mais ils n'effectuent aucun contrôle sur les informations après leurs dissémination ou distribution.

2.5.2 Exemples de modèles de CA pour les services Web

2.5.2.1 WS-ABAC: An Attribute Based Access Control for WS

WS-ABAC a été proposé par S. Hai-bo et H. Fan [HAI06]. Ce modèle utilise les attributs des entités pour l'authentification et le contrôle d'accès aux services Web. Il fournit aussi un mécanisme de négociation de confiance pour protéger les attributs. Ce modèle permet l'établissement de confiance entre les fournisseurs et les demandeurs de services en se basant sur les différents attributs.

Une entité peut être un sujet, une ressource ou un environnement. Par conséquent, les attributs peuvent être les attributs de sujet (l'identité de sujet, rôle, âge, code, adresse IP, nationalité, etc.), les attributs de ressource (l'identité de ressource, endroit, taille, etc.) ou les

attributs de l'environnement (l'heure, la date, l'état du système, etc.). Les assertions numériques sont souvent utilisées par les entités pour prouver leurs attributs.

Les éléments du modèle WS-ABAC sont :

- ***Le service :*** un service Web noté ***srv***. C'est une abstraction d'une opération (méthode) fournie par un système et il est décrit par WSDL. L'ensemble des services est noté par SRVS.
- ***L'ensemble d'utilisateurs :*** les entités qui vont accéder aux services Web.
- ***Type d'attribut :*** c'est une propriété d'une entité tel que l'identité de sujet (exemple : une clé publique), l'identité de ressource (exemple : URL du service Web), les paramètres du service, la date, etc. Il est noté par AT.
- ***L'ensemble d'attributs :*** c'est l'ensemble des types d'attributs et il est noté par AS. Exemple : AS = {Identité, Heure, Endroit, Charge_systeme}.
- ***Condition sur un attribut :*** c'est une expression logique sous la forme " <AT> <OP> <VALUE>", où AT ∈ AS, OP est un opérateur logique dans l'ensemble {=, >, <, ≥, ≤,≠} et VALUE est une valeur spécifique de AT. Exemple : Condition = Age > 65.
- ***Contrainte d'attribut :*** c'est une combinaison de conditions sur un ensemble d'attributs et elle est notée par C. Exemple : C = Heure ≥ 10 :00 ∩ Heure ≤ 15 :00 ∩ Charge_systeme = bas ∩ Endroit = "Office" ∩ Identité = "Manager".
- ***Requête d'accès au service :*** elle est notée par *acc* est définie par un triplet acc = <U, srv, RA> où

  (1) U est un utilisateur dans l'ensemble des utilisateurs qui demande le service

  (2) srv est l'identificateur du service demandé;

  (3) RA (runtime attribute) est l'ensemble de valeurs pour chaque type d'attribut de l'ensemble AS. RA = {v_1 de AT_1, v_2 de AT_2, …, v_n de AT_n}.

  A partir de cette requête, le système détermine quelle est la politique d'accès à appliquer pour le service.
- ***La politique d'accès :*** elle est notée par *pol*. Elle consiste en une collection de conditions d'accès. Formellement, elle est définie par un triplet pol = <S, srv, C>, où

  (1) S est le sujet dans cette politique, qui peut être un utilisateur;

  (2) srv est l'identificateur du service demandé;

  (3) C est la contrainte d'attributs qui doit être vérifiée pour accéder au service dans cette politique.
- ***La règle de contrôle d'accès :*** une requête d'accès au service acc = <U, srv, RA> est accordée seulement s'il existe une politique d'accès pol = <S, srv, C>, tel que les conditions suivantes sont vérifiées :

  (1) **acc**.srv = **pol**.srv;

(2) **U** ∈ S;

(3) **C** est vrai pour les valeurs RA, (c.-à-d., quand tous les AT_k dans la contrainte C sont remplacés par leurs valeurs de RA le résultat de l'expression booléenne est *vrai*).

En se basant sur le schéma de contrôle d'accès décrit ci-dessus, un algorithme basique d'évaluation d'une requête pour prendre la décision d'autorisation est conçu et présenté dans [HAI06]. Pour la spécification et l'application des politiques de contrôle d'accès, les auteurs ont choisi le standard XACML. Concernant la négociation de confiance, l'architecture de l'implémentation inclut le composant TrustBuilder [WIN02].

2.5.2.2 A Trust-based Context-Aware Access Control Model for Web Services

R. Bhatti, E. Bertino et A. Ghafoor ont proposé dans |BHA04] un modèle de contrôle d'accès aux services Web dont la motivation majeure est la prise en considération du contexte En effet, les services Web sont caractérisés par leur nature distribuée et dynamique, et donc un système de contrôle d'accès doit pouvoir capturer les informations contextuelle pertinentes pour la sécurité telles que la date, l'endroit, l'heure, l'état de l'environnement, etc. Les paramètres du contexte (appelés aussi profil d'utilisateur) capturent les conditions d'accès qui changent dynamiquement.

Les composantes du modèle formel sont résumées dans ce qui suit :

- *L'ensemble des noms des paramètres PN :* pour dénoter les noms possibles des paramètres du contexte.
- *L'ensemble des types des paramètres TP :* dénote les types possibles des paramètres du contexte.
- *Un paramètre du contexte p :* une structure de données contenant trois champs : name ∈ PN, type ∈ PT, et une fonction getValue().
- *L'ensemble des rôles RR :* contient les rôles réguliers (les rôles non administratifs).
- *L'ensemble des opérations RO :* contient les opérations régulières.
- *Un service srv :* abstraction d'une opération. Il est décrit avec le langage WSDL.
- *L'ensemble des services SRVS :* SRVS = {srv_1, ..., srv_k}, où **srv**$_i$, 1≤ i ≤k est un service.
- *L'ensemble du contexte C :* consiste en un ensemble de n paramètres du contexte {p_1, p_2, ..., p_n} tel que les noms de ces paramètres doivent être distincts.

Les ensembles PN et PT constituent des ensembles prédéfinis de noms et types de paramètres. Exemple : PN = {time_of_day, location, duration, system_load} et l'ensemble PT correspondant est PT = {Time, String, Long, Integer}. La fonction p. getValue() calcule dynamiquement la valeur du paramètre p.

- *La requête d'accès au service :* c'est un triplet <role, srv, context> où *role* ∈ RR, *srv* ∈ SRVS, et *context* est défini ci-dessus et capturé dynamiquement au moment de la requête. Selon la requête, le système détermine la politique d'accès applicable. La politique se basera sur l'ensemble de contraintes sur le rôle et le nom du service, et elle est évaluée avec les informations du contexte disponibles pour appliquer le contrôle d'accès à granularité fine.

  L'ensemble d'attributs A ($A \subseteq C$) d'un rôle contient une collection d'attributs contextuels (tels que l'heure) qui doivent être utilisés pour définir des conditions du contexte sur les rôles. La fonction getAttributesSet (role) retourne l'ensemble A pour un rôle donné.
- *La politique d'accès PA:* soit r ∈ RR un rôle et srv ∈ SRVS un nom d'un service. La politique d'accès AP pour une paire (r, srv) est un ensemble de clauses où chaque clause est une combinaison d'expressions booléennes. Une expression est sous la forme <attr □ val> où attr est un attribut d'un rôle r (attr ∈ getAttributesSet(*r*)), val est la valeur de l'attribut attr qui permet au rôle r d'accéder au service srv et □ est un opérateur de comparaison quelconque (exemple >, =, etc.).

Les algorithmes qui évaluent la requête de contrôle d'accès sont donnés dans algorithme 2.1.

Algorithme 2.1 : Les algorithmes de Context-Aware X-GTRBAC

Algorithm : ComputeAccess
Input : role, srv, C //C is context array
Output : decision d, d ∈ {YES, NO, PENDING, N/A}

```
1. CL [ ] = getClauses(role,srv)
2. A [ ] = getAttributesSet(role)
3. FOR i = 1 to length (CL) DO
   clause = CL [i]
   access = getDecision (clause,A,C)
   IF access = false
   return NO
4. IF access = true
return YES
```

Algorithm : getDecision
Input : clause,A,C
Output : result // boolean

```
1. initialize(result [])
2. FOR i = 1 to size(clause) DO
      expr = clause.getExpr (i)
      result [i]=evaluateExpr (expr,A,C)
3. return computeResult (result [ ])
```

Algorithm : EvaluateExpression
Input : expr, A, C
Output : result // boolean

```
1. name = expr.getAttrName()
2. attr = getAttribute(A,name)
3. result = checkCondition(attr, C)
4. return result
```

Algorithm : checkCondition
Input : attr, C
Output : result // boolean

```
1. p = match (C,attr)
   IF p.getValue() □ val
   result = true
   ELSE
   result = false
2.return result
```

Un domaine de confiance doit être incorporé à X-GTRBAC pour permettre un contrôle d'accès effectif dans un environnement distribué où les identités des utilisateurs ne sont pas connues à priori [BHA04]. Puisque X-GTRBAC prend les décisions d'accès en se basant sur les rôles, les informations d'identification (*credentials*) de la gestion de confiance (Trust Management) peuvent être utilisées pour assigner les rôles aux utilisateurs. L'indirection à travers les rôles permet l'extensibilité et la flexibilité dans le cas de larges systèmes distribués spécialement les services Web.

2.5.2.3 TrustBAC

TrustBAC [CHA06] a été proposé par S. Chakraborty et I. Ray pour adapter le modèle RBAC aux systèmes ouverts et décentralisés en intégrant la notion de confiance (Trust). Dans TrustBAC, les utilisateurs sont assignés aux *niveaux de confiance* en se basant sur les informations d'identification des utilisateurs, l'historique du comportement des utilisateurs et la recommandation sur les utilisateurs (réputation). Les niveaux de confiance sont assignés aux rôles qui sont à leur tour assignés aux permissions.
Le changement dans le niveau de confiance de l'utilisateur entraîne un changement dans ses rôles et par conséquent dans ses privilèges.

Le modèle est défini en terme d'ensemble d'éléments et de relations entre ces éléments. Les définitions formelles de ces notions sont résumées dans ce qui suit [CHA06].

- $User \in$ USERS : il représente un être humain dans ce modèle.
- $User\_properties \subseteq$ USER_PROPERTIES : chaque utilisateur u a un ensemble de propriétés P_u.
- $Session\_instance \in$ SESSION_ INSTANCES : c'est une instance « login » d'un utilisateur. Un utilisateur peut instancier plusieurs sessions en même temps.
- $Session\_type \in$ SESSION_TYPES : pour une session_instance s invoquée par un utilisateur u avec les propriétés p ($p \subseteq P_u$) a une session_type (type de session) p. formellement SESSION_TYPES = $2^{\text{USER\_PROPERTIES}}$
- $Session \in$ SESSIONS : une session avec session_instance s et session_type p est dénotée par le symbole s^p
- $Session\_history \in$ SESSION_HISTORY : ensemble d'informations concernant le comportement et le niveau de confiance de l'utilisateur dans la session précédente de même type.
- $Trust\_level \subseteq$ TRUST_LEVELS : un ensemble de nombres réels entre -1 et +1. trust_level est associé à un utilisateur avec une session particulière.
- $Role \in$ ROLES : c'est le même concept que celui du modèle RBAC.

- *Object* ∈ OBJECTS : c'est une ressource de données ou ressource du système.
- *Action* ∈ ACTIONS : exemple 'read', 'write', 'execute'.
- *Permission* ∈ PERMISSIONS : c'est une autorisation d'effectuer une certaine tâche dans le système. Formellement, l'ensemble de toutes les permissions est donné par PERMISSIONS = $2^{(OBJECTS \times ACTIONS)}$.
- *Constraint* ∈ CONSTRAINTS : semblable au concept de contrainte du modèle RBAC. Il est défini comme un prédicat appliqué à une relation entre deux éléments de TrustBAC et retourne la valeur ″acceptable″ ou ″non acceptable″.

Les relations entre ces éléments sont spécifiées par des relations mathématiques. TrustBAC a les relations suivantes :

- *sua* : USERS × SESSION_INSTANCES × SESSION_TYPES → SESSIONS : Relation d'assignation d'un utilisateur à une session. Sua (u, s, P) = s^P
- *UTA* ⊆ USERS × TRUST_LEVELS : relation d'assignation d'un utilisateur à un niveau de confiance. (u, L) ∈ UTA, cela veut dire que l'utilisateur u a le niveau de confiance L. Où L = {l}, l ∈ [-1, 1].
- *STA* ⊆ SESSIONS × TRUST_LEVELS : définit l'assignation session - trust_level

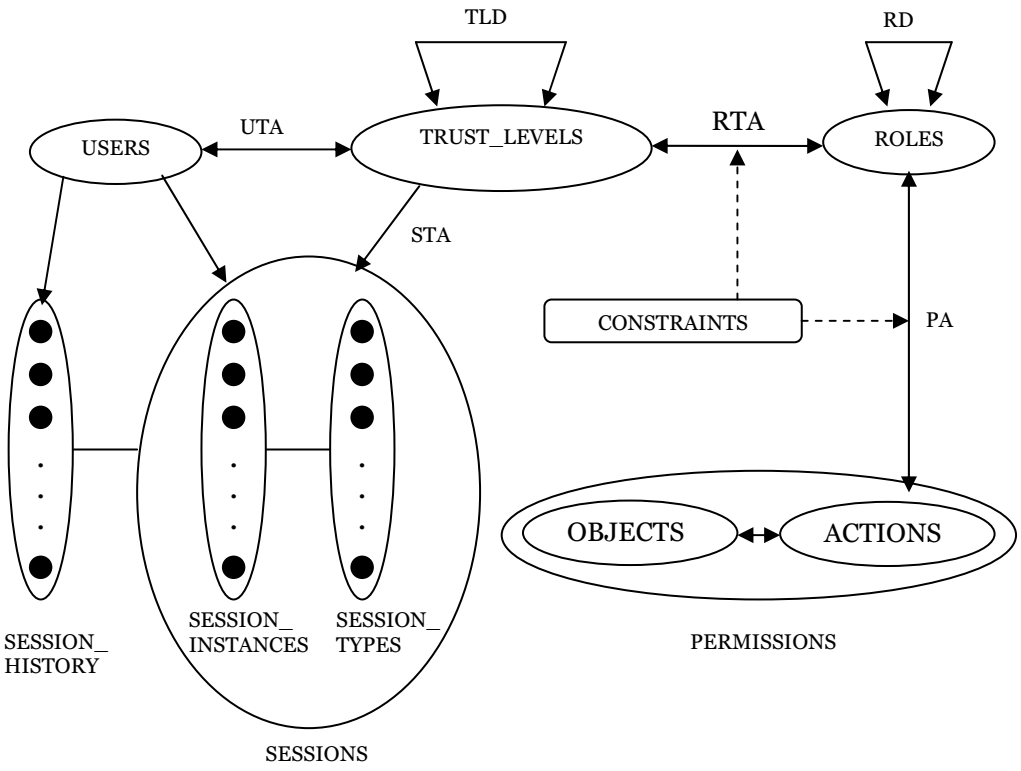


Figure 2.5 : Le modèle TrustBAC [CHA06]

- $RTA \subseteq$ ROLES × TRUST_LEVELS : définit la relation d'assignation role – trust_level
- La fonction *ush*: USERS × SESSIONS TYPES → SESSION_HISTORY définit une relation entre un utilisateur, un type de session et un historique de confiance de l'utilisateur pour une session de même type. ush (u, P) = ${}_u h^P$ où ${}_u h^P$ est associé à un utilisateur u et une session s^p de type p.
- $PA \subseteq$ PERMISSIONS × ROLES : relation d'assignation de permissions aux rôles.
- La fonction *Assigned_Roles* : TRUST_LEVELS → 2^{ROLES} retourne les rôles correspondants à un niveau de confiance $L \subseteq [-1, 1]$.

  Formellement, *Assigned_Roles* (L) = {r ∈ ROLES | (r,L) ∈ RTA}.
- La fonction *Assigned_Permission* : ROLES → $2^{PERMISSIONS}$ retourne les permissions accordées à un rôle donné.

  Formellement, *Assigned_Permission* (r) = {p ∈ PERMISSIONS | (p, r) ∈ PA}.
- Role dominance $RD \subseteq$ ROLES × ROLES : c'est un ordre partiel sur les rôles noté par ≤. C'est l'équivalent de la relation d'hiérarchie des rôles (RH) du RBAC. Formellement,

  $(r_1, r_2) \in RD \Rightarrow r_1 \leq r_2$ et $r_1 \leq r_2 \Rightarrow$ *Assigned_Permissions* $(r_1) \subseteq$ *Assigned_Permission* (r_2).
- Trust level dominance, $TLD \subseteq$ TRUST_LEVELS × TRUST_LEVELS: c'est un ordre partiel sur les niveaux de confiance note par ≤'. On dit que L_2 'domine' L_1 ou $(L_1, L_2) \in TLD$, seulement si $L_1 \subseteq L_2$.

  Si L_1 et L_2 sont des singletons et $L_1 = \{l_1\}$ et $L_2 = \{l_2\}$ alors $L_1 \leq' L_2 \Rightarrow l_1 \leq l_2$. La relation TLD est induite par RD, c.-à-d. pour chaque $(r_1, r_2) \in RD$, $\exists (L_1,L_2) \in TLD$ tel que $r_1 \in$ Assigned Roles (L_1) et $r_2 \in$ Assigned Roles (L_2).

Le modèle TrustBAC est illustré dans la figure 2.5 où les lignes sans flèches représentent les relations 1-1, les flèches simples représentent les relations 1-N et les flèches doubles dénotent les relations N-M.

Le modèle d'évaluation de relations de confiance

Dans TrustBAC, le modèle de vecteur de confiance introduit dans [RAY04] est adopté pour calculer les valeurs de confiance des utilisateurs.

La relation de confiance

La relation de confiance simple entre deux entités A et B notée par $(A \overset{c}{\rightarrow} B)_t$ est représentée par un vecteur de trois composantes :

$(A \overset{c}{\rightarrow} B)_t = [{}_A E^c_B \ , {}_A K^c_B , {}_\varphi R^c_B]$ où

${}_{A}E^{c}_{B}$: représente l'expérience de A sur B dans le contexte c.

${}_{A}K^{c}_{B}$: représente la connaissance de A sur B.

${}_{\varphi}R^{c}_{B}$: représente l'effet cumulatif de toutes les recommandations sur B envoyées à A de différentes sources.

Les trois facteurs sont exprimés par des valeurs numériques dans l'ensemble [-1, 1] ∪ {⊥}. Le symbole ⊥ est utilisé pour indiquer un manque de valeurs dû à l'insuffisance d'informations pour l'une des composantes.

1. ***calcul de la composante d'expérience :*** l'expérience est modélisée en termes de nombre d'événements capturés par l'entité A concernant l'entité B dans le contexte c durant une période de temps spécifiée $[t_o, t_n]$.

La politique d'expérience : spécifie un ensemble totalement ordonné des intervalles de temps non chevauchés avec un ensemble de poids non négatifs correspondants à chacun de ces intervalles de temps.

Les incidents I_j du $j^{ième}$ intervalle de temps est une somme normalisée de tous les événements. La normalisation est faite d'une manière à avoir $I_j \in [-1, 1]$.

$$I_j = \begin{cases} \perp & \text{si } \neg\exists\, e_k \in [t_{j-1}, t_j] \text{ pour tout k} \\ \dfrac{\sum_{k=1}^{n_j} v_k^j}{\sum_{k=1}^{n_j} |v_k^j|} & \text{Sinon} \end{cases}$$

Où e_k est le $k^{ième}$ événement dans l'intervalle $[t_{j-1}, t_j]$.

n_j est le nombre d'événements dans le $j^{ième}$ intervalle de temps.

v_k^j dénote la valeur associée à e_k

Ainsi, l'expérience de A sur B dans le contexte c est donnée par :

$${}_{A}E^{c}_{B} = \sum_{i=1}^{n} w_i I_i$$

Où n est le nombre d'intervalles

I_i les incidents du $i^{ième}$ intervalle

w_i est un poids non négatif associé au $i^{ième}$ intervalle.

2. ***calcul de la composante de connaissance :*** la composante de connaissance contient deux parties : la connaissance directe *d* et la connaissance indirecte (réputation) *r*. La connaissance de A sur B pour un contexte particulier c est donné par :

$$ {}_{A}K_{B}^{c} = \begin{cases} d & \text{si } r = \perp \\ r & \text{si } d = \perp \\ w_d \,.\, d + w_r \,.\, r & \text{si } d \neq \perp, r \neq \perp \\ \perp & \text{si } d = r = \perp \end{cases} $$

Où $d, r \in [-1, 1] \cup \{\perp\}$,

$w_d + w_r = 1$.

w_d et w_r les poids de d et r respectivement.

3. ***calcul de la composante de recommandation :*** la recommandation de A concernant B pour un contexte c est donnée comme suit :

$$ {}_{\varphi}R_{B}^{c} = \frac{\sum_{i=1}^{n} (v\,(A \xrightarrow{rec} j)_{t}^{N}) \,.\, V_j}{\sum_{i=1}^{n} (v\,(A \xrightarrow{rec} j)_{t}^{N})} $$

Où φ est le groupe de recommandeurs, $v\,(A \xrightarrow{rec} j)_{t}^{N}$ est la valeur de confiance du j[ième] recommandeur et V_j est la valeur de recommandation du j[ième] recommandeur sur B.

La relation de confiance normalisée

Pour normaliser le vecteur de confiance, il est combiné avec un vecteur de trois poids W en utilisant l'opérateur ⊙.

$$ (A \xrightarrow{c} B)_{t}^{N} = W \odot (A \xrightarrow{c} B)_t = [W_E, W_K, W_R] \odot [{}_{A}E_{B}^{c}, {}_{A}K_{B}^{c}, {}_{\varphi}R_{B}^{c}] $$

$$ = [W_E \cdot {}_{A}E_{B}^{c}, W_K \cdot {}_{A}K_{B}^{c}, W_R \cdot {}_{\varphi}R_{B}^{c}] = [{}_{A}E'_{B}^{c}, {}_{A}K'_{B}^{c}, {}_{\varphi}R'_{B}^{c}] $$

La valeur de la relation de confiance

La valeur de la relation de confiance est donnée par la somme des composantes du vecteur normalisé :

$$ v\,(A \xrightarrow{c} B)_{t}^{N} = {}_{A}E'_{B}^{c} + {}_{A}K'_{B}^{c} + {}_{\varphi}R'_{B}^{c} $$

Cette valeur aide à déterminer la confiance de la manière suivante :

$$ v\,(A \xrightarrow{c} B)_{t}^{N} = \begin{cases} [-1, 0) & \Rightarrow \text{méfiance} \\ 0 & \Rightarrow \text{neutre} \\ (0, 1] & \Rightarrow \text{confiance} \\ \perp & \Rightarrow \text{indéfini} \end{cases} $$

L'effet du temps

TrustBAC supporte la nature dynamique de confiance qui change avec le temps. Premièrement, la relation de confiance établie à un moment donné dans le passé influence le calcul de la relation à l'instant courant. Deuxièmement, la confiance se délabre (se dégrade) avec le temps. Cette notion est détaillée formellement dans [CHA06].

2.6 Conclusion

Le contrôle d'accès est un mécanisme de sécurité au niveau application. Cette technique est utilisée pour protéger les applications, les ressources informatiques et les données des systèmes informatiques fermés et elle s'étend et s'adapte pour répondre aux exigences de la sécurité des systèmes ouverts tels que les services Web en utilisant la notion de gestion de confiance.

Cependant, le contrôle d'accès seul ne répond pas aux exigences de la sécurité dans les systèmes distribués et décentralisés tels que les services Web. En effet, ce mécanisme protège les informations et empêchent les accès non autorisés mais il n'effectue aucun contrôle sur les informations après leur distribution.

Les modèles de contrôle d'accès présentés dans la section précédentes seront plus complets et assurent une sécurité fiable pour les services Web s'ils sont étendus pour effectuer le contrôle de flux d'information entre les objets manipulés par les services Web.

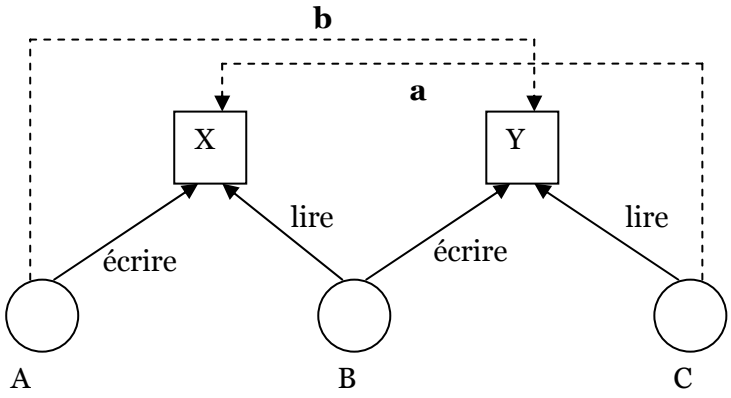
Dans le prochain chapitre, nous allons présenter la notion de contrôle de flux d'information et sa relation avec le contrôle d'accès.

CHAPITRE 3

Contrôle de Flux d'information

3.1 Introduction

Les systèmes informatiques utilisent typiquement le mécanisme de contrôle d'accès pour assurer la confidentialité et l'intégrité des informations. Le principe basique des modèles de contrôle d'accès est d'associer une clé d'accès à chaque opération sur un objet ; uniquement le processus possédant une clé peut exécuter l'opération associée sur un objet [BRY95]. L'exemple de la figure 3.1 montre que la confidentialité d'information (a) et l'intégrité d'information (b) qui est un problème dual à la confidentialité ne peuvent pas être assurés par le contrôle d'accès seul.



a : lecture indirecte
b : écriture indirecte

Figure 3.1 : Un flux d'information non sécurisé

Dans l'exemple, l'utilisateur B peut lire l'information de l'objet X puis l'écrire dans l'objet Y. L'utilisateur C a le droit de lire l'objet Y et ainsi il accède indirectement à l'information contenue dans l'objet X même s'il n'est pas autorisé à lire X, ce qui induit à une divulgation d'information. De la même manière, l'utilisateur A écrit une information dans l'objet X qui sera écrite dans Y par l'utilisateur B. Par conséquent, l'utilisateur A peut écrire indirectement

dans Y même s'il n'est pas autorisé à écrire Y, ce qui induit à une violation d'intégrité de l'objet Y.
Un mécanisme de contrôle des informations après leur dissémination est donc nécessaire pour compléter la tâche du contrôle d'accès.

Dans ce chapitre, nous introduisons la notion de contrôle de flux d'information, nous résumons la taxonomie des modèles de contrôle de flux d'information, nous citons des travaux significatifs dans différents domaines et nous présentons un exemple de modèle de contrôle de flux d'information qui servira d'introduction à notre approche de contrôle de flux d'information pour les services Web.

3.2 Le contrôle de flux d'information

Le contrôle de flux d'information (IFC : Information Flow Control) est un aspect de la sécurité informatique qui consiste à régulariser le flux d'information entre les objets d'un système informatique pour assurer la confidentialité et l'intégrité des informations.

3.2.1 Définitions

La transmission d'information entre les exécutions se distingue de flux d'information conséquent qui peut être imposé entre les objets [BER98].

❖ **La transmission d'information**

Chaque message entre les objets a comme conséquence une transmission *d'*information. La transmission *vers l'avant* diffuse l'information de l'expéditeur au récepteur par la liste de paramètres de message. La transmission *en arrière* diffuse l'information du récepteur à l'expéditeur par la réponse. La transmission d'information peut être directe par l'envoi de messages entre deux exécutions ou indirecte sans échange de messages. Par exemple, si une exécution e_i invoque une exécution e_h qui à son tour invoque une exécution e_k alors il y a une transmission directe à travers les paramètres du message, de e_i à e_h et de e_h à e_k, et par conséquent, une transmission indirecte de e_i à e_k.

On appelle un *canal* une transmission directe ou indirecte d'informations. Selon la direction de l'information transmise, on distingue entre un canal *vers l'avant* (de l'émetteur au récepteur) et un canal *en arrière* (du récepteur à l'émetteur).

❖ **Le flux d'information (FI)**

Il existe un flux d'information entre un objet o_h et un objet o_k quand l'information sur l'état de o_h est rangée dans o_k. Un flux d'information intuitivement exige une transmission

d'information entre quelques exécutions fonctionnant sur les deux objets. Cependant, la transmission d'information entre des exécutions fonctionnant sur différents objets ne provoque pas nécessairement un flux. Nous considérons ces deux suppositions [BER98]:

1) Un flux d'information à partir d'un objet peut exister seulement si l'information est *lue* à partir de cet objet.
2) Un flux d'information vers un objet peut exister seulement si l'information est *écrite* dans cet objet.

Un flux d'information d'un objet o_h vers un objet o_k qui est dénoté par $o_h \Rightarrow o_k$ nécessite :

1) Une opération de *lecture* à partir de l'objet "source" o_h,
2) Une opération *d'écriture* ou de *création* sur l'objet "destinataire " o_k et
3) Un *canal de transmission* de l'exécution de lecture vers l'exécution d'écriture.

❖ **Un flux d'information sain**

Un flux d'information sain (sûr) est un flux d'information qui ne provoque pas une fuite d'information vers les utilisateurs non autorisés à y accéder. Un flux d'information d'un objet obéit aux spécifications de sécurité (c.-à-d. les accès qui sont ou qui ne sont pas permis) si et seulement si les utilisateurs qui sont autorisés à lire un objet :

1) Sont permis de lire l'information coulée vers eux soit par une autorisation explicite dans la liste de contrôle d'accès (ACL) soit par des exceptions laxistes durant la transmission d'information,
2) Ne sont pas empêchés de lire cette information par des exceptions restrictives (négatives) appliquées durant la transmission d'information.

En d'autres termes, le flux d'information est sain si tous les utilisateurs autorisés à lire un objet sont autorisés de voir l'information transmise vers cet objet (selon l'ACL de l'objet duquel l'information a été lue et les exceptions produites le long de la transmission)

L'exemple de la figure 3.2 résume le problème de la sécurité durant les flux d'informations. Un observateur possède une copie d'un code d'une application. L'observateur peut voir ce qui se passe pendant l'exécution car la politique de sécurité du système lui permet de lire les valeurs de certaines variables. Dans la figure, les variables que l'observateur est autorisé de voir sont désignées par les disques entourés par le cercle, les disques en dehors du cercle représentent les variables contenant les informations que l'observateur est interdit de voir. La condition de sécurité est la suivante : étant donné que l'observateur connaît le texte du programme du système, il ne doit pas être capable de déduire les valeurs des variables en dehors du cercle via les valeurs des variables à l'intérieur

de ce cercle. En d'autres termes, il ne doit pas y avoir un flux d'information des variables en dehors du cercle vers celles de l'intérieur.

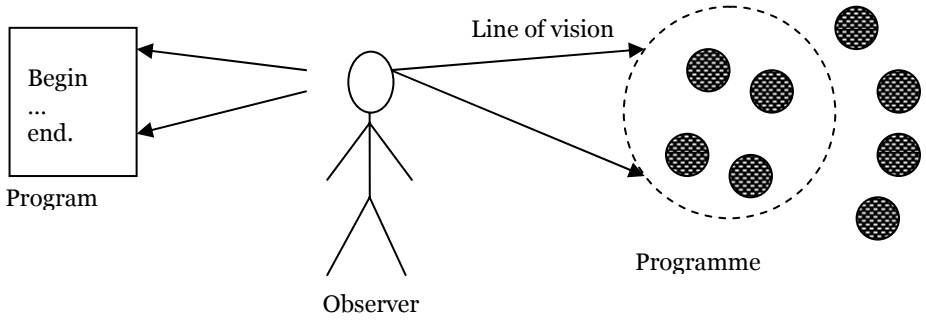


Figure 3.2 : Exemple de contrôle de flux d'information [BRY95]

3.2.2 Les concepts de contrôle de flux d'information

Le contrôle de flux d'information (CFI) introduit de nouveaux concepts que nous essayons de définir dans cette section.

❖ **Un flux d'information explicite et implicite**

Un flux d'information explicite est causé par une influence directe d'un objet sur un autre. Un flux d'information d'un objet *a* à un objet *b* se produit en assignant la valeur de *a* à *b* ($a \leftarrow b$). Le flux d'information implicite est causé par une influence logique d'un objet sur un autre. Par exemple, la figure 3.3 illustre un flux d'information causé par une structure de contrôle d'un programme, où la valeur secrète de *b* peut être déterminée en examinant la valeur de la variable publique *x* après l'exécution du programme. Ainsi, il y a un flux implicite de b vers x.

```
int x; // public
boolean b; // secret
x = 0;
  if (b)
    {
    x = 1;
    }
```

Figure 3.3 : Exemple de flux d'information implicite

❖ **Le contrôle de flux d'information statique et dynamique**

Le contrôle de flux d'information statique consiste à définir (à travers des règles de flux d'information) et éliminer les flux d'information illégaux avant l'exécution du système sécurisé (à la compilation par exemple). Dans le contrôle de flux d'information dynamique, les flux illégaux sont découverts avant et durant l'exécution. Mais, ils sont interdits au temps d'exécution.

Le contrôle de flux d'information statique ne consomme pas du temps d'exécution (ou minimise l'*overhead*) et permet de discerner les flux implicites. En revanche, il a une vue partielle du système réel. Le contrôle de flux d'information dynamique a une vue plus grande sur le système, mais il peut échouer. Les contrôles de flux d'information statique et dynamique peuvent être combinés.

❖ **La flexibilité d'un contrôle de flux d'information**

Le contrôle de flux d'information impose une restriction sur les systèmes contrôlés puisque parfois il interdit quelques flux légaux. La flexibilité veut dire que tous les flux d'information légaux sont autorisés. Ainsi, elle peut être réalisée en définissant des exceptions aux règles de contrôle de flux d'information d'une manière à préserver l'exécution normale du système. Par exemple, on peut autoriser le flux du mot de passe d'un objet à un autre afin d'obtenir l'accès à quelques ressources bien que ce n'est pas permis.

3.3 La taxonomie des modèles de CFI

Généralement, on peut catégoriser les travaux de contrôle de flux d'information en trois classes : le contrôle de flux d'information basé sur les langages de programmation, le contrôle de flux d'information basé sur les messages et le contrôle de flux d'information basé sur la théorie de l'information [BER06].

3.3.1 Le CFI basé sur les langages de programmation

Le contrôle de flux d'information basé sur les langages de programmation consiste à utiliser un système de type de sécurité. Des annotations sont associées aux types de variables et expressions d'un programme pour spécifier des politiques sur l'utilisation des données de ces types. Ces politiques de sécurité sont ensuite appliquées par la vérification de type à la compilation. Ainsi, ce contrôle ne consomme pas du temps d'exécution.

Les mécanismes basés sur les langages sont utilisés pour différents objectifs de la sécurité. Le mécanisme de sécurité de l'environnement Java est un exemple bien connu y compris le

vérificateur de *byte code* [LIN99], le modèle sandbox [SUN07] et l'inspection de pile (*the stack inspection*) [WAL00]. Tous ces mécanismes sont basés sur le langage, en d'autres termes, ils sont appliqués via le langage Java. Mais, aucun d'entre eux n'est prévu pour le contrôle de flux d'information.

Le travail de Denning pour l'analyse de flux sécurisé et le modèle de treillis [GRA72] a été le premier qui a initié la certification de programme, qui est une forme efficace d'analyse statique et qui a pu être facilement incorporée dans un compilateur pour vérifier les flux d'informations sécurisés dans les programmes.

D'autres travaux et modèles de contrôle de flux d'informations pour les langages de programmations sont cités dans la section (3.4).

Le contrôle de flux d'information basé sur les langages de programmation présente différents inconvénients. La stratégie de contrôle de flux d'information doit être fixée à la compilation par le développeur de l'application et ne peut pas être modifiée pendant l'exécution. En outre, l'intégration de contrôle de flux d'information dans les langages existants nécessite une compréhension profonde des compilateurs des langages de programmation qui n'est pas toujours une solution pratique. De plus, ce type de contrôle n'est pas convenable aux systèmes distribués et hétérogènes où les applications sont écrites dans différents langages de programmation et s'exécutent sur des plateformes incompatibles. Pour ces raisons, le contrôle de flux d'information basé sur les messages est apparu.

3.3.2 Le CFI basé sur les messages

Dans ces systèmes, les composantes peuvent communiquer uniquement par l'envoi de messages. Le mécanisme de contrôle de flux d'information intercepte les messages passés entre les composantes du système et applique les politiques de sécurité pour permettre ou interdire ces flux.

McCollumn et al [MCC90] ont proposé une nouvelle forme de contrôle d'accès qui impose des restrictions sur le flux d'information dans le système. Leur modèle permet à un propriétaire d'un fichier d'associer à chaque objet *o* contenu dans ce fichier une liste de contrôle d'accès (ACL). Cette liste se propage via les labels des sujets et des objets à tous les objets que *o* peut affecter.

Une approche similaire a été proposée par Stoughton [STO81]. Dans cette proposition, chaque objet a deux attributs de protection : l'attribut d'accès *courant* décrit ce qui doit être fait par le destinataire de l'information de cet objet ; et l'attribut d'accès *potentiel* décrit ce qui doit être fait avec l'information même si elle est copiée dans un autre objet.

Boebert et Furguson [BOE85] ont proposé de contrôler les *Trojan Horse*[1] en interposant un sous système protégé entre le programme suspecté et le système de fichiers. Une autre approche proposée par Karger [KAR87] contrôle les effets des *Trojan Horse* dans les systèmes discrétionnaires en limitant les fichiers accessibles par les programmes d'application à base de quelques connaissances sur les programmes eux-mêmes. Tous les accès demandés par les applications sont arbitrés par un contrôleur de messages. Le contrôleur de messages compare le nom de l'objet à accéder au modèle spécifié pour l'application. Si le nom de l'objet satisfait le modèle alors l'accès est accordé.

Elisa Bertino, Sushil Jajodia et al. [BER97] ont décrit un modèle de contrôle de flux d'information d'une grande assurance pour les systèmes orienté objet. Dans ce modèle, les flux d'information bidirectionnels sont produits par l'envoi de messages où un filtre de messages intercepte chaque message échangé entre les objets pendant l'exécution d'une transaction pour garantir qu'aucun flux illégal ne se produit.

Les travaux dans ce domaine ne sont pas uniquement motivés par les limites de contrôle de flux d'information basé sur les langages de programmation mais aussi par la popularisation des systèmes d'appel de procédures distantes (RPC : Remote Procedure Call), les systèmes distribués orientés objet (tel que CORBA) et les services Web. Néanmoins, la supposition que les informations circulent uniquement par l'échange de messages les rend inappropriés pour certains systèmes tels que les systèmes orientés objet où l'information peut circuler par les relations d'hiérarchie et d'agrégation entre les objets.

3.3.3 Le CFI basé sur la théorie de l'information

La plupart des travaux dans le contrôle de flux d'information sont intéressés par les flux causés par les transmissions au sein des réseaux et la sauvegarde d'information. Cependant, d'autres types de flux d'information peuvent mener à des fuites d'information. Les flux d'information peuvent être interprétés par les modèles de la théorie de l'information. Ceci est principalement dû au fait que les systèmes non déterministes peuvent montrer des canaux secrets probabilistes (probabilistic covert channels)[2] qui ne sont pas régnés par les modèles standard de la sécurité informatique.

McLean [MCL90] a donné un traitement très général des modèles de sécurité de flux d'information. Sa définition de sécurité (appelée FM) est donnée sous forme d'équation

[1] Le Trojan Horse est un programme dans lequel une partie nocive (par exemple, le sous-programme qui cause la fuite de l'information aux lecteurs non autorisés) est contenue à l'intérieur et semble un code inoffensif

[2] Les mécanismes de transmission d'informations à travers les systèmes informatiques sont appelés canaux. Les canaux qui exploitent un mécanisme dont le but primaire n'est pas le transfert de l'information s'appellent les canaux secrets.

entraînant des probabilités conditionnelles. J.Gray [GRA91] a proposé un modèle général et probabiliste de machine à état qui peut être utilisé pour modéliser une grande classe des systèmes informatiques non déterministes et déterministes. Il a développé la théorie des probabilités nécessaire pour énoncer et montrer des propriétés de flux d'information du système modélisé.

Tandis que le contrôle de flux d'information basé sur la théorie de l'information offre une assurance élevée concernant la non divulgation d'information à travers les canaux secrets, son manque à une implémentation concrète demeure une question cruciale à résoudre.

3.4 Exemples de techniques de CFI

Dans ce qui suit, nous résumons quelques techniques de contrôle de flux d'information dans trois domaines, les langages de programmation, les bases de données et les services Web.

Un système de type puissant pour une analyse de flux sécurisé a été proposé par Volpano [VOL96]. Le modèle contribue au contrôle de flux d'information dans des programmes dans le contexte des niveaux de sensibilité multiples. En relation avec les sémantiques des langages de programmation, la vérification de la sécurité est confirmée en montrant que les programmes bien typés sont caractérisés par la propriété de la non interférence[3].

Le travail le plus récent et le plus significatif dans le domaine de contrôle de flux d'information pour les langages de programmation est celui de Myers et Liskov [MYE97]. Ils ont proposé un modèle de labels avec une administration décentralisée. Le modèle permet aux utilisateurs de contrôler le flux de leurs informations d'une manière décentralisée et sans imposer des contraintes rigides d'un système de sécurité à multi niveaux traditionnel. En outre, ils ont montré que la vérification statique des labels peut être utilisée pour certifier la légalité de flux d'information tout en réduisant la vérification dynamique des labels. Un autre travail de Myers et Liskov a étendu le modèle des labels décentralisés avec plus de flexibilité et d'expressivité. La solution a pallié les limites de leur travail tels que le remplacement des propriétaires et l'extension de lecteurs. Par la suite, Myers et Liskov ont montré dans [MYE00] comment le modèle décentralisé de labels peut être utilisé pour protéger l'intimité (*privacy*). De plus, ils ont introduit une extension au langage Java (appelée Jif : Java Information Flow) qui fournit un mécanisme de vérification statique en utilisant le modèle de labels proposé.

[3] Les donnés confidentielles ne doivent pas interférées avec (ou affecter) les données publiques visibles par d'autres utilisateurs. Une politique de ce genre est appelée *la politique de non interférence*.

Un autre travail de Myers [MYE99] a introduit un nouveau langage, appelé Jflow, pour la vérification statique de flux d'information. C'est une extension au langage Java en ajoutant des annotations aux informations statiquement vérifiées. Le langage proposé inclut le modèle de labels décentralisé, le polymorphisme de labels, la vérification dynamique de labels et l'inférence automatique de labels.
Un travail récent par Li [LI 02] a analysé le flux d'information dans les programmes Java et a proposé une technique de découpage en tranches. L'utilisation de cette technique permet le calcul de la quantité, de la largeur et de la corrélation de flux d'information.

Une approche flexible pour le contrôle de flux d'information dans les systèmes orientés objet a été proposée par Ferrari dans [FER97]. Des exceptions sur les restrictions dans le flux d'information sont spécifiées au moyen des *waivers* associés aux méthodes. Un ensemble de conditions formelles est défini pour assurer la consistance dans le flux d'information sûr.
Une approche d'analyse de flux d'information pour un système RBAC est proposée dans [OSB02]. Dans cette approche, à partir d'un graphe arbitraire de rôles est produit un graphe de flux d'information. D'une manière semblable, Belokosztolszki et al. [BEL03] ont présenté une approche qui examine les fuites d'information involontaires pendant l'application des politiques du modèle RBAC.

E. Bertino et al. ont décrit dans [BER02] un nombre de solutions pour le problème de distribution sélective et sécurisée de documents XML. L'une de ces solutions utilise le chiffrement des portions de documents en utilisant des clés différentes et la distribution de clés sélective.
Z. Tari et al. [TAR06] ont proposé un modèle de contrôle de flux d'information pour les services Web basé sur la vérification dynamique de labels des objets manipulés par les services Web. Ce modèle consiste en une adaptation du modèle de Myers et Liskov [MYE97] proposé pour les langages de programmation. Le but du modèle est de protéger la confidentialité des données manipulées par un système de services Web en empêchant la divulgation d'information.
Le modèle a été étendu par N. Berrehouma [BER06] pour supporter les aspects de propagation et de dé-classification.

3.5 Le modèle de labels décentralisés de Myers et Liskov

Dans cette section, nous décrivons le modèle de contrôle de flux d'information introduit par Myers et Liskov [MYE00] [MYE98]. Le modèle améliore les modèles de sécurité à multi

niveaux existants en permettant aux utilisateurs de déclassifier les informations d'une manière décentralisée et en développant un support de partage de données à granularité fine.
Les concepts de ce modèle sont :

1. un contrôle de flux d'information décentralisé : chaque sujet (entité active) définit une politique de flux pour une information. L'ensemble des politiques de tous les sujets est représenté dans une structure de données appelée ″label″ et associée à cette information, et le système garantit que toutes les politiques sont respectées simultanément.
2. le modèle fonctionne même si les sujets ne se font pas confiance (pas de relations de confiance entre eux).
3. le modèle permet aux sujets de déclassifier les labels en modifiant leurs propres politiques de flux. La dé-classification permet aux programmeurs d'enlever quelques restrictions quand c'est nécessaire pour augmenter la flexibilité.

3.5.1 Les principaux

Les sujets dans ce modèle sont appelés ″principaux″ qui sont des utilisateurs, des groupes ou des rôles. Ils sont responsables des informations qu'ils possèdent ; ils les modifient et les révèlent à d'autres principaux.
Certains principaux peuvent agir pour d'autres principaux. Un principal p peut agir pour un autre principal q (formellement $p \geq q$) veut dire que p possède tous les privilèges de q. La relation ″agir pour″ est réflexive et transitive, et elle définit une hiérarchie ou un ordre partiel des principaux. La relation d'hiérarchie des principaux est similaire à la relation d'hiérarchie des rôles pour le modèle RBAC.
La notation $P \vdash x \geq y$ veut dire que dans la hiérarchie des principaux P, le principal x agit pour le principal y. Si une hiérarchie des principaux P' contient plus de relations ″agir pour″ qu'une autre hiérarchie P alors on dit que P' étend P ou $P' \supseteq P$.
La figure suivante montre un exemple d'hiérarchie des principaux.

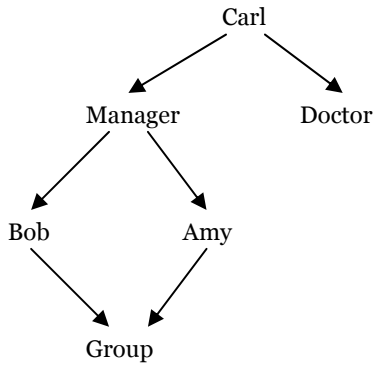


Figure 3.4 : une hiérarchie des principaux [MYE00]

Dans cet exemple, un groupe est modélisé en introduisant les relations "agir pour" entre chaque membre du groupe (*Bob* et *Amy*) et le principal *group*. Ces relations permettent à *Bob* et *Amy* de lire les données lisibles par le groupe et de contrôler les données contrôlables par le groupe. Le principal *manager* est capable d'agir pour les principaux *Bob* et *Amy*. Ce principal est l'un des rôles que le principal *Carl* peut accomplir. *Carl* a aussi un autre rôle qui est *Doctor*. Carl peut utiliser ses rôles pour empêcher les fuites d'information accidentelles entre les objets associés à ses différentes fonctions.

3.5.2 Les labels

Un label est une structure de données contenant un ensemble de composantes qui expriment les politiques de flux des différents principaux.
Chaque composante dans un label *L* représente la politique d'un propriétaire sur l'utilisation de la donnée à laquelle *L* est associé. Elle contient deux parties, un propriétaire et un ensemble de lecteurs, et elle est écrite sous la forme *propriétaire:lecteurs* :

- Le propriétaire *O* d'une composante est la source de l'information, et
- Les lecteurs sont les principaux autorisés par le propriétaire *O* pour lire l'information qu'il possède. L'ensemble de ces lecteurs est noté par *lecteurs* (*L*, *O*).

Les propriétés d'un label sont les suivantes :

- Un même principal peut être un propriétaire dans plusieurs composantes,
- Chaque politique (composante) doit être respectée lorsque les données circulent dans le système. Ainsi, un utilisateur peut lire une information si et seulement si le principal de l'utilisateur agit au moins pour un lecteur dans chaque politique du label de cette information. Ce principal est alors appelé un *lecteur effectif* (effective reader).
- Pour chaque composante K d'un label L, on dénote le propriétaire par O_K et l'ensemble des lecteurs par R_K.
- Un label peut être vu comme étant un ensemble de flux où chaque flux est une pair (propriétaire, lecteurs). Ainsi, le propriétaire est la source de l'information et les lecteurs sont ses destinations possibles.

Un exemple de label est L = {o_1 : r_1, r_2 ; o_2 : r_2, r_3} où o_1, o_2, r_1, r_2, r_3 dénotent les principaux. Les points virgules séparent les politiques (composantes) dans le label L. Les propriétaires de ces politiques sont o_1, o_2 et les ensembles de lecteurs de ces politiques sont {r_1, r_2}, {r_2, r_3} respectivement.

Les labels sont utilisés pour contrôler le flux d'information dans les programmes en annotant leurs variables par des labels. Le label d'une variable contrôle comment l'information rangée dans cette variable peut être disséminée. La clé du contrôle de flux

d'information est d'assurer que même si l'information circule à travers le système, son label devient plus restrictif : les labels ont plus de propriétaires ou certains propriétaires permettent moins de lecteurs. Si le contenu d'une variable affecte une autre variable, il y a un flux d'information de la première variable à la deuxième, et par conséquent, le label de la seconde variable doit être plus restrictif que celui de la première variable.

3.5.3 Relabeling

Le *relabeling* (ré–étiquetage) est l'opération qui permet la dérivation des labels après chaque opération sur les objets correspondants. La dérivation de labels doit garantir la sécurité de flux d'information et elle est dite saine si elle ne cause pas une fuite d'information. En d'autres termes, l'ensemble des lecteurs effectifs du nouveau label est un sous ensemble de celui de label originel.

Quatre manières de dérivation saines sont définies :

1. enlever un lecteur : la dérivation de labels en enlevant un lecteur d'une politique est saine.
2. ajouter une politique : la dérivation en ajoutant une politique est saine puisque les politiques déjà existantes dans le label sont toujours appliqués.
3. ajouter un lecteur : l'ajout d'un lecteur r' à une politique est sain si la politique contient un lecteur r pour lequel r' agit (r' agit pour r) car r' a tous les privilèges de r.
4. remplacer un propriétaire : la dérivation en remplaçant un propriétaire o par un autre principal o' qui agit pour o est saine (semblable à l'ajout de lecteurs).

La dérivation par restriction

La dérivation par restriction est causée par les opérations de calcul sur l'information. Le nouveau label contient tous les propriétaires du label originel. L'ensemble des lecteurs du nouveau label est un sous ensemble des lecteurs autorisés par le label originel. La dérivation par restriction d'un label L_1 à un label L_2 est notée par $L_1 \subseteq L_2$ et elle est définie comme suit :

$$L_1 \subseteq L_2 \Leftrightarrow \begin{cases} propriétaires\ (L_1) \subseteq propriétaires\ (L_2) \\ \forall\ o \in propriétaires\ (L_1),\ lecteurs\ (L_1, o)\ \supseteq lecteurs\ ((L_2, o) \end{cases}$$

Exemple : la dérivation de $\{o : r_1, r_2\}$ à $\{o : r_1\}$ est une dérivation par restriction.

Un nouveau label est obtenu à partir de deux labels L_1 et L_2 de deux opérandes avec une opération de jointure notée par $L_1 \sqcup L_2$ et elle est définie comme suit : $L_1 \sqcup L_2 = L_1 \cup L_2$

Exemple : la jointure de deux labels {A : B} et {C : A} est {A : B ; C : A}
Cette règle assure que les politiques dans un label d'une valeur se propagent vers les labels des valeurs affectées. Ainsi, la confidentialité de la valeur (la donnés) est protégée même après son utilisation dans les opérations de calcul.

La dérivation par dé-classification

La dé-classification d'information est parfois nécessaire pendant l'exécution d'un système. Les propriétaires dans un label peuvent relaxer leurs politiques restrictives quand c'est nécessaire et ce type de dérivation est une forme de dé-classification.
Chaque propriétaire effectue la dé-classification de ses politiques indépendamment des autres propriétaires, donc il n'affecte pas leurs politiques. Ainsi, la dé-classification est saine car les lectures sont autorisées par le consensus de tous les propriétaires de l'information.
La dé-classification est réalisée par un propriétaire en ajoutant des lecteurs à son ensemble de lecteurs autorisés à lire l'information ou en enlevant entièrement la politique (qui est l'équivalent de mettre tous les principaux comme lecteurs dans une politique).
Par conséquent, un label L_1 peut être dérivé à un label L_2 par la dé-classification tel que $L_1 \sqsubseteq (L_2 \sqcup L_A)$, où L_A est un label contenant exactement les politiques de la forme {p :} pour chaque principal p dans l'autorité courante d'un processus[4].
Formellement, la dérivation par dé-classification est exprimée comme suit :

$$L_1 \sqsubseteq (L_2 \sqcup L_A) \Rightarrow L_1 \text{ peut être déclassifié à } L_2$$

Où,

$$L_A = \bigsqcup_{\text{(p dans l'autorité courante)}} \{p :\}$$

3.5.4 Les labels d'intégrité

Les labels décrits ci-dessus contiennent les politiques de confidentialité. Le modèle a été généralisé dans [MYE00] pour supporter les politiques d'intégrité qui sont duales aux politiques de confidentialité. Les politiques de confidentialité protègent les données contre des lectures impropres même après leurs utilisations et les politiques d'intégrité protègent les données des écritures impropres.
Un label d'intégrité garde les traces de toutes les sources qui ont affecté (directement ou indirectement) la valeur correspondante.

[4] L'autorité d'un processus est un ensemble de principaux pour lesquels ce processus agit ou s'exécute.

La structure d'un label d'intégrité est identique à celle d'un label de confidentialité. Chaque politique d'intégrité a un propriétaire et un ensemble de rédacteurs autorisés à modifier la valeur. Une politique d'intégrité est considérée comme une garantie de qualité. Par exemple la politique $\{o_1 : w_1, w_2\}$ est une garantie par le principal o que seuls w_1, w_2 peuvent affecter la valeur de la donnée.

Un label d'intégrité permet de protéger les données des modifications impropres.

Exemple : soit une variable avec une seule politique d'intégrité suivante : $\{o_1 : w_1, w_2\}$. Une valeur dénotée par le label $\{o_1 : w_1\}$ peut être écrite dans cette variable car cette valeur a été affectée par w_1 et la politique de la variable permet à w_1 d'affecter cette dernière. Une valeur avec le label $\{o_1 : w_1, w_3\}$ ne peut pas être affectée à la variable car la valeur a été écrite par w_3 qui n'est pas mentionné dans le label de la variable.

Relabeling

Quatre manières de dérivation saines sont aussi définies pour un label d'intégrité [MYE00]:

1. ajouter un rédacteur.
2. enlever une politique.
3. remplacer un rédacteur.
4. ajouter une politique.

Les règles de dérivation des labels d'intégrités sont duales aux règles de dérivation des labels de confidentialité. Pour les labels de confidentialité L_1, L_2 et les labels d'intégrité correspondants L_1', L_2', l'équivalence suivante est définie : $L_1 \sqsubseteq L_2 \leftrightarrow L_2' \sqsubseteq L_1'$.

La dé-classification

La dé-classification est aussi définie pour les labels d'intégrité. Pour le cas des labels de confidentialité, le mécanisme de dé-classification permet de supprimer des politiques qui sont jugées trop restrictives. L'opération duale concernant la dé-classification des labels d'intégrité est d'ajouter de nouvelles politiques d'intégrité dans le cas où la données a une intégrité plus élevée que l'analyse de dépendance stricte peut suggérer.

L'ajout d'une politique d'intégrité à un label ou la suppression des rédacteurs d'une politique existante représente un rapport de confiance sur l'intégrité de la donnée qui lui permet ultérieurement d'être utilisée librement.

La dé-classification peut être décrite formellement de la manière suivante :

La dé-classification d'un label d'intégrité L_1 à un label L_2 est permise quand :

$$L_2 \sqcap L_A' \sqsubseteq L_1$$

Où L_A' est le label d'intégrité contenant une politique pour chaque principale dans l'autorité du processus. Une telle politique liste tous les principaux du système comme des

rédacteurs et le symbole ⊓ dénote le rassemblement des deux labels qui est calculé en prenant l'union des politiques d'intégrité de ces labels.

Le modèle de labels décentralisé permet un contrôle de flux d'information décentralisé et assure la confidentialité et l'intégrité des informations. Cependant, sa dépendance aux langages de programmation rend son intégration dans les systèmes existants très difficile.

3.6 Conclusion

Le contrôle de flux d'information est largement adopté, amélioré et déployé dans différents domaines. C'est un mécanisme nécessaire pour assurer la confidentialité et l'intégrité de bout en bout dans les systèmes ouverts tels que les services Web où très peu de travaux s'inscrivent dans ce contexte.

Dans ce chapitre, nous avons introduit les concepts du contrôle de flux d'information, nous avons cité quelques travaux concernant ce mécanisme de sécurité et s'inscrivant dans différents domaines et nous avons présenté le modèle de contrôle de flux d'information adapté pour notre modèle détaillé dans le dernier chapitre.

CHAPITRE 4

Un modèle hybride pour la sécurité des Services Web (MHS-SW)

4.1 Introduction

La confidentialité et l'intégrité des informations échangées dans les systèmes de services Web restent toujours un enjeu fondamental de la technologie des services Web. La plupart des mécanismes et des standards de sécurité des services Web sont axés sur la définition de la sécurité au niveau transport et messages et ignorent la sécurité au niveau application.
Dans ce rapport, nous allons proposer une approche hybride pour la sécurité de bout en bout des services Web. Cette approche consiste en un modèle de contrôle d'accès (CA) et de contrôle de flux d'information (CFI) flexible. Le mot "*flexible*" veut dire le modèle n'interfère pas avec la logique métier des services Web.
Nous allons définir les propriétés et les exigences particulières pour le contrôle d'accès et le contrôle de flux d'information, nous allons montrer à travers un scénario leur nécessité et nous allons présenter notre modèle hybride pour la sécurité des services Web.

4.2 CA et CFI pour les services Web

Le contrôle d'accès et le contrôle de flux d'information dans les services Web sont différents de ceux définis pour les systèmes fermés (systèmes basés sur des réseaux privés). En effet, les services Web sont des applications de nature distribuée et décentralisée intéropérant indépendamment de leurs plateformes. De plus, les interfaces des services Web sont exposées à des consommateurs de services inconnus préalablement. Par conséquent, les systèmes centralisés de CA et de CFI doivent être adaptés aux services Web.
Le contrôle de flux d'information peut être considéré comme une généralisation et complément au contrôle d'accès.

Le CA ou encore l'authentification et l'autorisation assure une sécurité locale permettant de contrôler l'accès aux ressources du système (services et objets). L'aspect important du modèle de CA est la définition des politiques pour les décisions d'autorisation. Un modèle de CA doit être aussi flexible pour supporter les différentes exigences du contrôle d'accès pour les environnements des services Web.

Le CFI assure la confidentialité des informations même après leur dissémination. Ainsi, il est fortement lié aux mécanismes de sécurité du niveau transport pour l'échange des informations et la relation de confiance entre les modules coopératifs.

Le CA et le CFI pour les services Web doivent vérifier certaines conditions :

- ✓ Indépendance de la logique métier des services Web;
- ✓ Support des politiques multiples des utilisateurs;
- ✓ Services et informations contrôlés avec différents niveaux de granularité;
- ✓ Gestion des relations de confiance entre les différents modules de CA et de CFI;
- ✓ Une approche décentralisée pour leur gestion et application;
- ✓ Flexibilité et possibilité de dé-classification.

4.3 Scénario

Afin de montrer la nécessité de sécuriser les informations d'un système de services Web, nous avons considéré le scénario suivant [BER06].

Supposons qu'une clinique médicale utilise des services Web pour interagir avec le monde externe. Les services que cette clinique offre sont : la récupération des informations médicales, des informations administratives, et des informations financières concernant ses patients. Ces informations sont sous forme de données XML, elles sont confidentielles et doivent être protégées par un mécanisme de sécurité très efficace. Ce mécanisme doit prendre en considération l'intimité, l'intégrité et la non divulgation de l'information et il doit être assez flexible pour ne pas perturber la logique métier[5] des services Web.

Le patient est le propriétaire des informations médicales. Donc, il peut désigner ceux qui peuvent les lire et/ou les modifier. Les propriétaires des informations administratives sont l'administrateur de la clinique et le patient, chacun d'eux peuvent sélectionner d'autres personnes pour lire et/ou modifier ces informations. L'administrateur de la clinique et l'agent financier sont les propriétaires des informations financières. Enfin, l'administrateur de la clinique est le propriétaire du dossier contenant toutes ces informations.

[5] On qualifie de code métier le code qui implémente les fonctionnalités de l'application visibles des utilisateurs (par exemple "ordre de virement" pour une application bancaire). Le code métier s'oppose au code technique qui permet au code métier de s'exécuter. La gestion de la persistance, est par exemple, une fonctionnalité technique.

Les informations offertes par cette clinique sont importantes pour d'autres organisations. Par exemple, le centre national des statistiques a besoin des informations sur les maladies chroniques. D'autres cliniques ont besoin des historiques médicaux concernant des patients. Les institutions du ministère de la santé ont besoin des informations administratives dans un but d'organisation et les informations financières peuvent être échangées avec les entreprises financières comme les banques.

Les services Web de la clinique permettent aux autres d'obtenir ces informations si et seulement s'ils garantissent qu'elles ne seront pas lues ou modifiées par des utilisateurs non autorisés, elles ne seront pas divulguées après leur dissémination et elles ne seront utilisées par le service demandeur qu'à des fins connues et précises. Cela peut être atteint en fournissant un cadre de confiance entre les services Web, par exemple uniquement ceux certifiés sont autorisés à interagir avec les services de la clinique. Un contrôle d'accès flexible gère et contrôle les accès aux services et aux objets du système et un contrôle de flux d'information assure la non divulgation de l'information.

Ce scénario peut être illustré par la figure 4.1 suivante

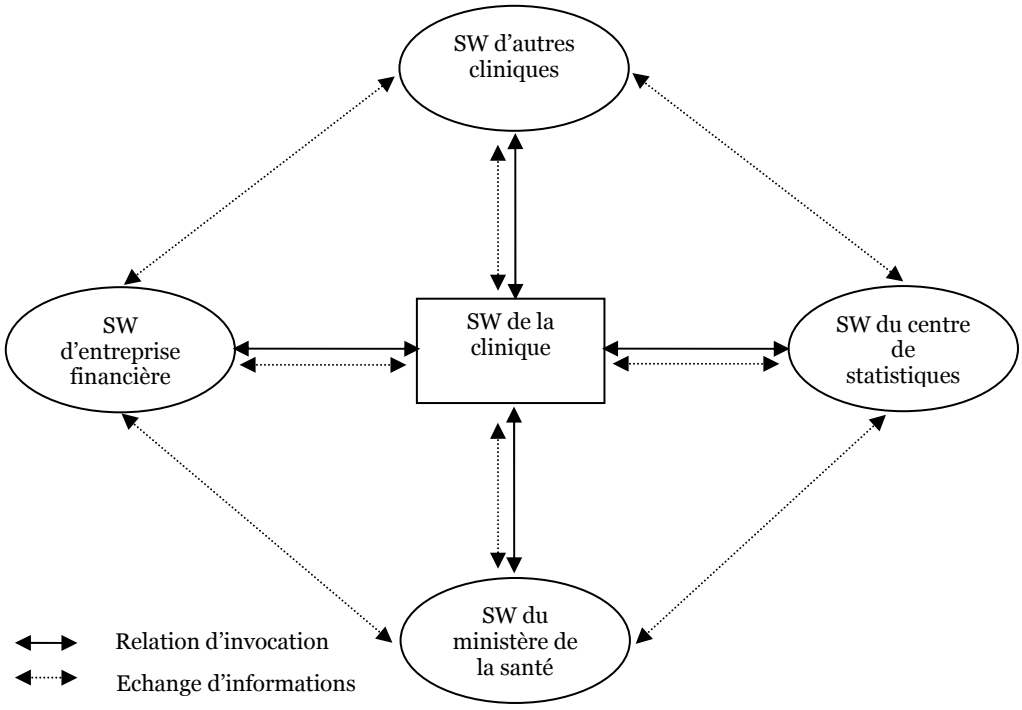


Figure 4.1 : Exemple de motivation

4.4 Présentation de la solution

Un travail récent pour le contrôle de flux d'information pour les services Web a été proposé par Z. Tari et al. [TAR06]. Il consiste en une adaptation du modèle de labels décentralisé de Myers [MYE98] pour le contrôle de flux d'information dans les langages de programmation.

N. Berrehouma dans [BER06] a apporté des améliorations à ce modèle en ajoutant deux jointures (propagation et dé-classification), des preuves de sûreté, des algorithmes pour permettre leur mise en œuvre et de nouvelles règles. Cependant, il reste encore des enjeux relatifs à ce modèle et qui sont posées dans [BER06] à savoir la granularité fine des services Web et la mise en relation du modèle de contrôle de flux d'information proposé avec un modèle de contrôle d'accès local. Après notre étude détaillée et vérification de la sûreté du modèle proposé dans [BER06], nous avons noté des lacunes au niveau des définitions des jointures et par conséquent dans les algorithmes correspondants.

Dans ce travail, nous présentons une nouvelle approche hybride qui intègre un modèle de contrôle de flux d'information et un modèle de contrôle d'accès aux services Web qui permet la gestion de la confiance et l'autorisation.

L'amélioration du modèle de contrôle de flux d'information concerne essentiellement :

a. La modification des définitions des jointures de dérivation des labels pour réaliser des flux d'information sains pour assurer la non divulgation d'informations
b. Extension du modèle de CFI pour permettre non seulement la confidentialité des informations mais aussi leur intégrité en considérant l'opération d'écriture dans les politiques des propriétaires des objets.

Le modèle de contrôle d'accès intégré quant à lui permet à la fois l'authentification des demandeurs de services et l'autorisation d'accès aux ressources du système (services et données). Il est mis en relation avec le modèle de contrôle de flux d'information en utilisant le concept de "rôle" considéré dans le modèle de contrôle d'accès RBAC (**R**ole **B**ased **A**ccess **C**ontrol). Pour cela, nous avons adapté le modèle ARBAC (**A**ttribute and **R**ole **B**ased **A**ccess **C**ontrol) proposé par M. LIU et al. [LIU05].

Le choix du ARBAC est justifié par :

- ✓ La gestion de confiance et l'authentification en utilisant la notion d'attributs des utilisateurs;
- ✓ Utilisation de la notion de rôle pour regrouper les fonctionnalités d'une position de travail et faciliter l'extensibilité et l'administration du modèle;

- ✓ La granularité fine des services Web en considérant les méthodes de ces derniers et les données manipulées ;
- ✓ La prise en considération des paramètres du contexte dans les politiques de contrôle d'accès.

Le modèle de sécurité proposé permet à un système de services Web à la fois le contrôle d'accès à ces services et le contrôle de flux d'information entre ses objets.
La solution consiste en un ensemble de définitions, de règles, de preuves de sûreté pour la confidentialité et l'intégrité des informations et des algorithmes permettant leur implémentation.

4.5 Le modèle hybride pour la sécurité des SW

Le principe du modèle RBAC traditionnel est : l'assignation des utilisateurs aux rôles en se basant sur leurs identités et l'assignation des permissions et privilèges d'accès aux rôles.
Notre modèle illustré dans la figure 4.2 garde les mêmes concepts de base du modèle RBAC (cf. § 2.4.3). Les définitions formelles des éléments du modèle proposé seront énoncées dans le chapitre suivant.

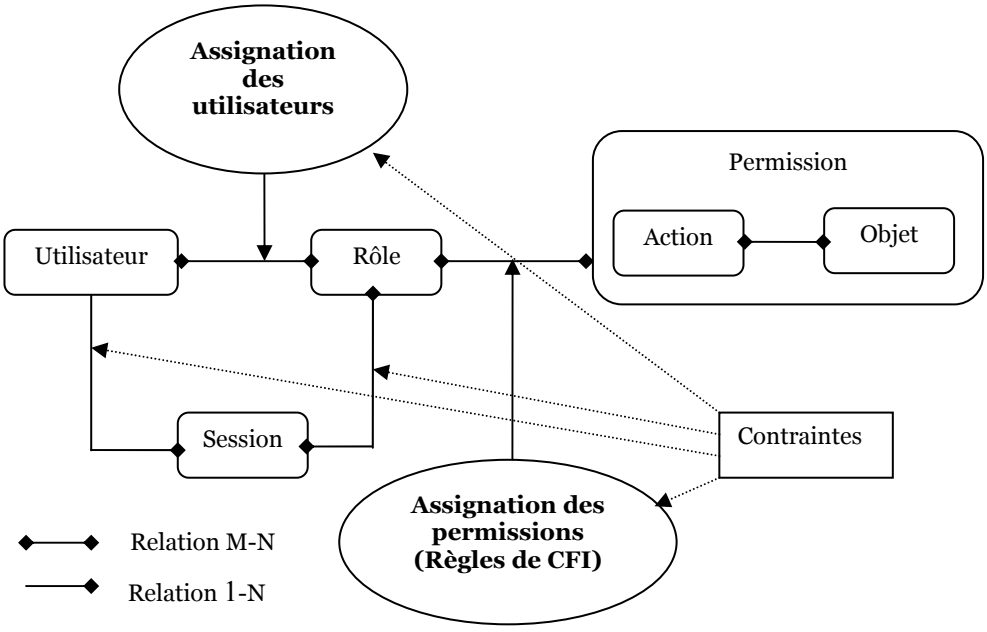


Figure 4.2 : le modèle de CA et de CFI

Dans notre approche hybride, nous nous basons sur le modèle ARBAC pour le contrôle d'accès au système de services Web et le modèle de CFI pour l'assignation des permissions

aux rôles. Ainsi, notre modèle étend le modèle ARBAC en utilisant les règles de contrôle de flux d'information pour permettre la confidentialité et l'intégrité des informations du système même après leur dissémination et il étend le modèle de CFI en permettant le contrôle d'accès des demandeurs de services Web en utilisant la gestion de confiance.

4.5.1 Le modèle de contrôle d'accès

Pour pouvoir intégrer le modèle de contrôle d'accès avec le modèle de contrôle de flux d'information avec la notion du *rôle*, nous adaptons le modèle ARBAC en introduisant les modifications suivantes :

1. Dans le modèle ARBAC, les ressources du système que les permissions d'accès considèrent sont les méthodes des services Web et les données manipulées par ces dernières. Dans notre modèle, les permissions d'accès considèrent uniquement les objets du système qui sont des données XML. Les opérations (ou méthodes) des services Web sont prises en compte dans les politiques d'assignation des utilisateurs aux rôles où un rôle regroupe un ensemble de méthodes offertes par des services Web.
2. Les politiques d'assignation des utilisateurs aux rôles prennent en considération non seulement les attributs des utilisateurs comme dans ARBAC mais aussi les paramètres relatifs aux méthodes des services Web et du contexte.
3. La fonction d'assignation des permissions aux rôles utilise les règles du modèle de CFI pour permettre, en plus de contrôle d'accès aux objets du système, le contrôle de flux d'informations entres ces objets. Cette fonction contrôle l'information au moment de l'accès et après sa dissémination.

Notre modèle de contrôle d'accès utilise les attributs des demandeurs de services (par exemple leurs identités et leurs clés) pour la gestion de confiance et l'authentification.
Pour permettre un contrôle d'accès à granularité fine, nous avons considéré dans notre modèle les méthodes fournies par les services Web au lieu des services eux-mêmes. Ces méthodes sont regroupées sous formes de rôles où chaque rôle désigne les fonctionnalités de même niveau d'autorité, par exemple: toutes les méthodes que l'administrateur de la clinique du scénario cité ci-dessus peut invoquer sont regroupées dans le rôle *administrateur*.
Le modèle prend en compte le contexte de la requête pour offrir plus de précision dans la définition des politiques d'autorisation.
Une politique de contrôle d'accès dans notre modèle utilise les attributs du demandeur de services, les paramètres de la méthode demandée et les paramètres du contexte. Si une requête correspond à l'une des politiques définissant un rôle, la requête est acceptée et le rôle correspondant est activé. La fonction d'assignation des permissions aux rôles utilise les règles

définies par le modèle de contrôle de flux d'information où un principal dans notre cas est un rôle[6]. Dans ce cas, les politiques de contrôle d'accès sont définies par les rôles et non pas par les services Web, ce qui permet l'indépendance entre le système de sécurité et la logique métier des services Web.

4.5.2 Le modèle de contrôle de flux d'information

Le modèle de contrôle de flux d'information proposé par Z. Tari et al. utilise la notion de labels de sécurité associés aux objets manipulés par les services Web. Ce modèle assure un contrôle de flux d'information décentralisé en permettant à l'ensemble des propriétaires (services Web) des objets du système de spécifier leurs politiques sur leurs objets. Un label associé à un objet contient toutes les politiques des différents propriétaires de cet objet où une politique permet à un propriétaire de désigner l'ensemble de principaux qui peuvent lire l'objet et un ensemble de principaux auxquels il fait confiance. Le modèle définit des règles d'accès aux objets et à leurs labels et des règles permettant un flux d'information sain (un flux qui n'induit pas à une divulgation d'information). Il introduit aussi la notion de jointure qui permet la dérivation de labels pour les objets obtenus suites à des opérations d'affectation et des opération de calculs.

Une nouvelle version du modèle est proposée par N. Berrehouma. Cette version étend le modèle par la notion de propagation imposée par la structure hiérarchique des données XML et la notion de dé-classification pour augmenter la flexibilité du modèle. Cependant, cette amélioration ne traite pas certaines lacunes posées dans [BER06] à savoir la granularité fine des services Web et la mise en relation du modèle de contrôle de flux d'information proposé avec un modèle de contrôle d'accès.

Après vérification du modèle proposé dans [BER06], nous avons noté des lacunes aux niveaux des jointures de dérivation des labels à l'aide des contre exemples pour les démonstrations de sûreté. Pour cela, nous avons proposé une amélioration pour ce modèle en présentant de nouvelles définitions de jointure avec des démonstrations détaillées de sûreté et de nouvelles règles. En outre, nous avons ajouté un nouvel champ pour la politique d'un label pour contenir l'ensemble de principaux autorisés à écrire dans l'objet correspondant. L'ensemble des rédacteurs permet le contrôle d'intégrité des informations en plus de la confidentialité.

Les règles de contrôle de flux que nous avons proposées sont utilisées pour l'assignation des permissions aux rôles du système.

[6] Un principal dans le modèle originel du contrôle de flux d'information est un "service Web".

4.6 L'architecture distribuée du MHS-SW

L'architecture distribuée des systèmes de services Web exige une architecture distribuée du système de contrôle d'accès et de flux d'information.

Le système de contrôle d'accès et de flux d'information est un ensemble de modules distribués dans les différents fournisseurs de services. Nous supposons que ces modules partagent une relation de confiance (par exemple : certifiés par une autorité de certification) et sont indépendants des services Web du même fournisseur de services.

Les flux d'information entre les variables internes d'une même méthode d'un service Web sont considérés sains.

Le contrôle est effectué :

- Au moment d'accès des méthodes des services Web aux objets du système et leurs labels, par exemple aux moments de lecture, d'écriture, de création, etc.
- Avant et après le flux d'information entre les objets du système, par exemple : au moment d'assignation d'objets, pendant l'exécution des opérations de calculs sur les objets, etc.

La figure 4.3 montre l'architecture logique distribuée du modèle MHS-SW.

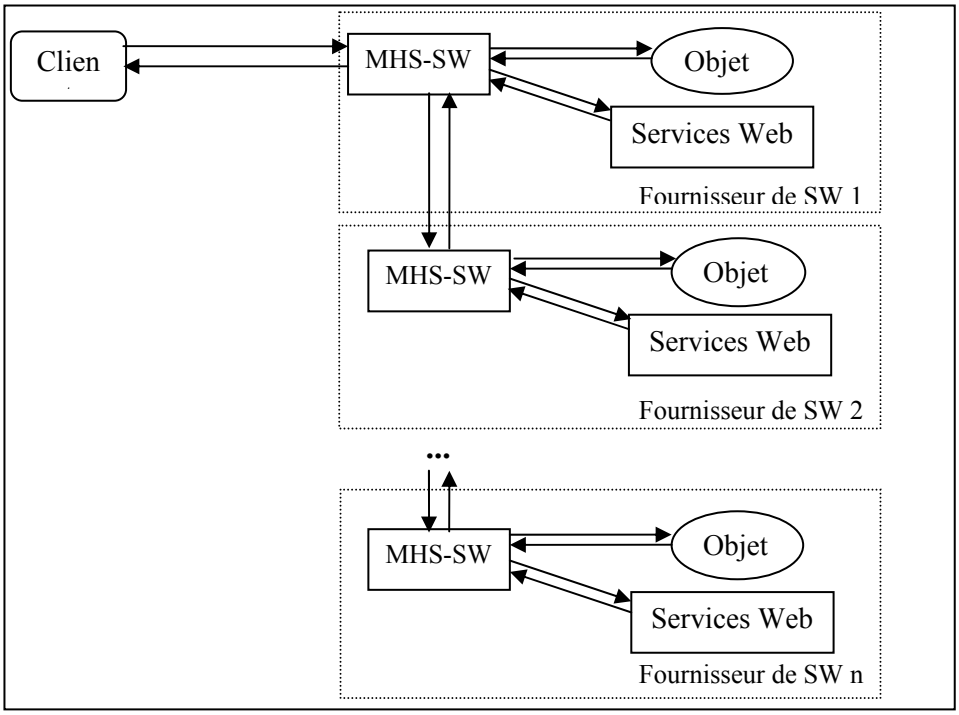


Figure 4.3 : Architecture distribuée du MHS-SW

Chaque requête d'accès à une méthode du service Web doit passer par le module de contrôle d'accès et de contrôle de flux d'information (MHS-SW).

Chaque modification des services Web et/ou des objets du système entraîne la mise à jour de la base des politiques de contrôle d'accès et/ou de contrôle de flux d'information.

4.7 L'architecture de la sécurité locale avec MHS-SW

Le module local de MHS-SW d'un fournisseur de services est composé de deux fonctions fondamentales : la fonction de création des politiques de contrôle d'accès et de flux d'information (service d'administration de politiques) et la fonction d'application de ces politiques lors de la réception d'une requête. La figure 4.4 montre l'architecture de la sécurité pour les services Web et les objets du système.

1) La création des politiques du modèle MHS-SW

1. Spécification des politiques définissant les différents rôles dans le fournisseur de services.
2. Collection des politiques des propriétaires (rôles) sur les objets de leur autorité.
3. Création d'un label pour chaque objet.
4. Application des règles de propagation des labels.
5. Construire la base des politiques du modèle de MHS-SW.

2) Application des politiques du modèle MHS-SW

6. L'utilisateur distant commence une session (login) dans le domaine de demandeur de service (DS). Le demandeur de service effectue une authentification.
7. Si l'utilisateur est accepté, le DS encapsule les informations d'attributs de l'utilisateur, son certificat d'identité et les labels des objets transmis comme paramètres sous forme d'assertions SAML (constructions XML), le met dans l'entête d'un message SOAP et le transmet au fournisseur de service (FS) via HTTP.
8. A la réception du message SOAP, le gestionnaire de messages SOAP valide le certificat d'identité de demandeur de service. S'il est de confiance (trusted), il récupère les attributs de l'utilisateur (AU), les paramètres de la méthode (PM) demandée et les politiques de CFI des objets passés en paramètre à partir du message SOAP et les passe au module de MHS-SW.
9. Le module de MHS-SW utilise les informations extraites de l'entête du message SOAP et les informations du contexte (AC) pour effectuer l'assignation de l'utilisateur et activer le rôle correspondant, et il utilise les attributs du service demandé pour valider la requête.

10. Le module de MHS-SW effectue le contrôle de séparation des fonctions et si l'activation du rôle est validée, une permission d'invocation de la méthode de service Web est envoyée au gestionnaire de messages SOAP et le module de MHS-SW applique les règles d'accès et de flux d'information pour chaque action sur les objets.
11. Si une requête d'un utilisateur est acceptée et exécutée, une réponse est retournée à l'utilisateur dans un message SOAP, sinon un message d'erreur est envoyé.
12. Après chaque action d'accès ou de flux d'information, les labels des objets correspondants sont mis à jour.

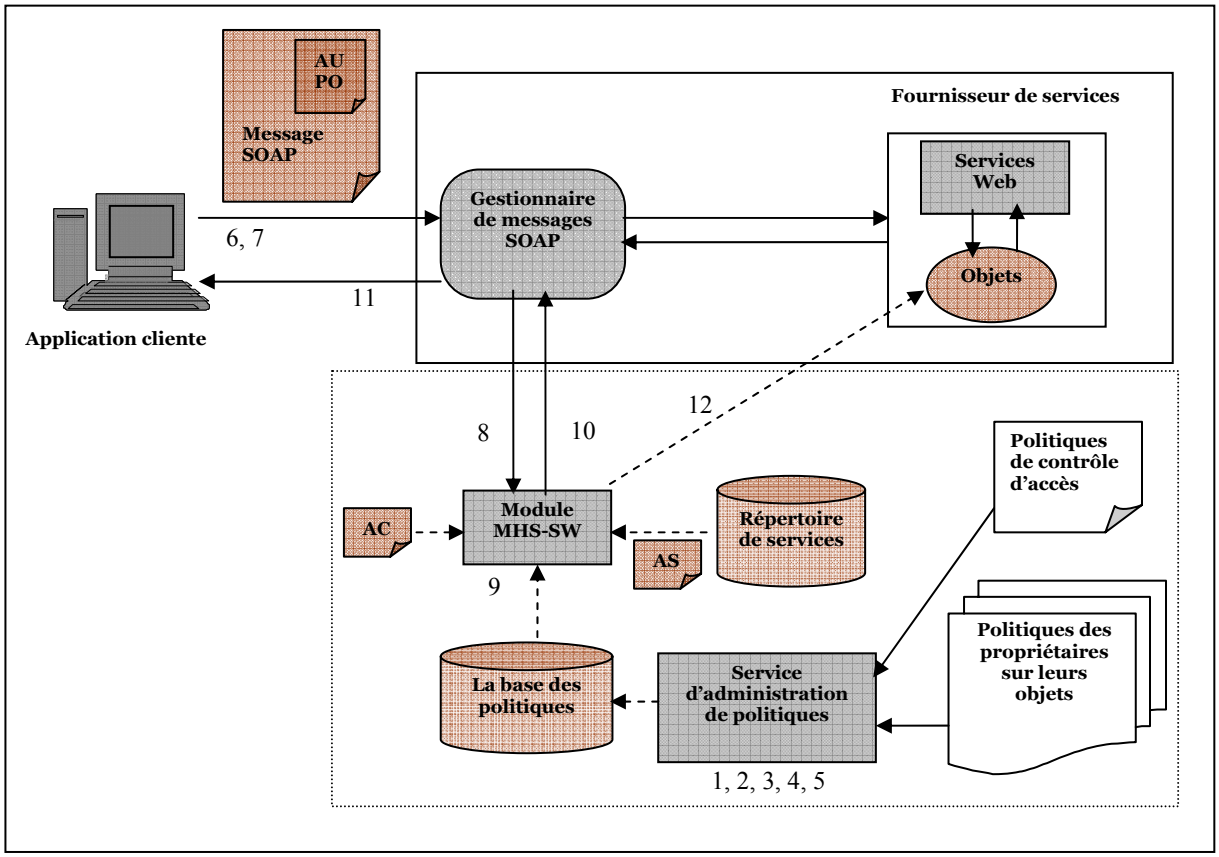


Figure 4.4 : Architecture de sécurité

4.8 Conclusion

Nous avons donné dans ce chapitre une présentation globale de notre modèle hybride pour la sécurité des services Web. Nous avons expliqué les spécificités du contrôle d'accès et du contrôle de flux d'information du modèle MHS-SW et nous avons montré comment ces deux mécanismes de sécurité sont liés pour obtenir un modèle de sécurité de bout en bout

fiable pour les services Web. Nous avons également fourni une architecture logique pour le système de sécurité.

Le développement et la modélisation du modèle MHS-SW que nous avons proposé feront l'objet du prochain chapitre.

CHAPITRE 5

Développement du modèle MHS-SW

5.1 Introduction

Le modèle hybride pour la sécurité des services Web (MHS-SW) proposé dans le quatrième chapitre consiste en une intégration de deux mécanismes de sécurité de niveau application à savoir le contrôle d'accès et le contrôle de flux d'information.

Dans ce chapitre, nous développons le modèle en définissant formellement les différents concepts du MHS-SW, ses politiques de contrôle d'accès, ses règles de contrôle de flux d'information exploitées pour l'assignation des permissions aux rôles, les jointures de dérivation de labels et les algorithmes correspondants. Les preuves formelles de la sûreté des flux d'information en utilisant les jointures de dérivation de labels sont aussi données en détail dans ce chapitre.

5.2 Le modèle MHS-SW

5.2.1 Le modèle de contrôle d'accès

Dans ce qui suit, nous énonçons les concepts et les définitions nécessaires pour la modélisation du modèle de contrôle d'accès et l'algorithme permettant son implémentation.

5.2.1.1 Les concepts de base

1. Les utilisateurs

Les utilisateurs (sujets) peuvent être des êtres humains, des agents autonomes ou des services Web. Ce sont des entités qui effectuent des actions sur les objets du système.

Exemples : l'administrateur dans le scénario de la section 4.3, une application cliente, le service Web d'une autre organisation, etc.

Un attribut : un attribut d'un sujet (utilisateur) est une description d'une propriété de ce dernier. Il est représenté par une paire **(nom_attr, type).**

Exemple : parmi les attributs qui doivent figurer dans une politique de contrôle d'accès :

- l'attribut d'identité d'un sujet **(ID, string)**. Chaque utilisateur dans le domaine de fournisseur de service a une valeur d'ID différente.
- L'attribut **(clé, string)** qui représente le domaine auquel le sujet appartient. La valeur de la clé est la clé publique du certificat d'identité du demandeur de service.

Le fournisseur de service identifie chaque sujet à travers l'ID et la clé.

Un utilisateur : un utilisateur est représenté par un ensemble d'attributs.

$$\text{Utilisateur}_i = \{(\text{ID}, \text{valeur\_ID}^i), (\text{clé}, \text{valeur\_clé}^i), (\text{nom\_attr}_1, \text{valeur}_1^i),\ldots, (\text{nom\_attr}_m, \text{valeur}_m^i)\}$$

L'ensemble des utilisateurs : $\text{Utilisateur} = \{\text{utilisateur}_1, \text{utilisateur}_2, \ldots, \text{utilisateur}_u\}$

2. Les méthodes des services Web

Les services Web ont un ensemble de méthodes qui manipulent les objets du système, invoquent et répondent aux requêtes d'autres services.

Les paramètres d'une méthode : Chaque méthode est caractérisée par un ensemble de paramètres d'entrée et de sortie. Un paramètre est de la forme (nom_par, type).

$$\text{method}_i = \{(\text{nom\_par}_1, \text{valeur}_1^i), (\text{nom\_par}_2, \text{valeur}_2^i), \ldots, (\text{nom\_par}_l, \text{valeur}_l^i)\}$$

Exemple : nom_par_1 et nom_par_2 donnent respectivement le nom du service contenant la méthode et le nom de la méthode.

L'ensemble des méthodes : $\text{Méthode} = \{\text{method}_1, \text{method}_2, \ldots, \text{method}_o\}$

3. Le contexte

Le contexte est l'environnement opérationnel, technique ou la situation où l'accès à l'information apparaît, par exemple la date et l'heure courante.

Les paramètres du contexte : Un paramètre du contexte est une structure de données sous la forme (nom_c, type_c, getvalue()), où la valeur de nom_c est le nom du paramètre de contexte, type_c est le nom du type de ce paramètre et getvalue() retourne la valeur de ce paramètre à un instant donné.

L'ensemble des paramètres du contexte : $\text{C} = \{\text{Cp}_1, \text{Cp}_2, \ldots, \text{Cp}_t\}$

Où $Cp_i = (nom\_c_i, type\_c_i, getvalue_i())$ et $1 \le i \le t$

Exemple : Cp_i = (heure_accès, time, get_time()).

4. Les conditions

Une politique de contrôle d'accès est un ensemble de conditions sur les attributs du sujet, les paramètres des méthodes et les paramètres du contexte.
L'ensemble de toutes ces conditions est :

$$Condition = \{cond_1, cond_2, ..., cond_r\}$$

Tel que $cond_i = \{(nom\_attr_1, VA_1), ..., (nom\_attr_{m'}, VA_{m'}), (nom\_par_1, VP_1), ..., (nom\_par_{l'}, VP_{l'}), (nom\_c_1, VC_1), ..., (nom\_c_{t'}, VC_{t'})\}$

Où $1 \le m' \le m$; $1 \le l' \le l$; $1 \le t' \le t$;
Et VA_i $(1 \le i \le m')$; Vp_j $(1 \le j \le l')$; VC_h $(1 \le h \le t')$ sont les plages de valeurs permises pour un attribut de sujet (nom_attr_i), un paramètre d'une méthode (nom_par_j) et un paramètre du contexte (nom_c_h) respectivement. Elles sont sous la forme $\{valeur_1, valeur_2, ..., valeur_n\}$ ou $[valeur_{min}, valeur_{max}]$.

5. Les rôles

En général, un rôle dans le modèle RBAC est un titre de fonction ou travail dans une organisation plus quelques sémantiques concernant l'autorité et la responsabilité accordée à un membre du rôle.
Exemple : l'administrateur, l'agent financier et le patient peuvent constituer des rôles dans le scénario de la section 4.3.
Il y a une hiérarchie des rôles (HR) partiellement ordonnée, où $x \ge y$ signifie que x hérite toutes les permissions assignées au rôle y. L'héritage le long des rôles est transitif et de multiples héritages sont permis dans l'ordre partiel.
Nous représentons l'ensemble de tous les rôles du système par :

$$Rôle = \{r_1, r_2, ..., r_n\}$$

Proposition : la relation d'hiérarchie peut être définie comme suit

$$r_i \le r_j \Leftrightarrow \text{les permissions de } r_i \subseteq \text{permissions de } r_j$$

Exemple : *agent financier* ≤ *administrateur*, veut dire que *administrateur* a toutes les permissions assignées à *agent financier*.

6. Les permissions

L'ensemble de toutes les actions ou opérations pouvant être effectuées sur les objets du système est représenté par :

$$\text{Permission} = \{P_1, P_2, ..., P_p\}$$

Où $P_i = (\text{action}_i, \text{ressource}_i)$

Et action_i : lire, écrire, ajouter, créer ou supprimer

ressource_i : un objet du système qui est une donnée XML (exemple : les informations médicales d'un patient)

7. La fonction d'assignation rôle-condition

La fonction d'assignation entre l'ensemble des rôles (Rôle) et l'ensemble des conditions (Condition) donne un ensemble de conditions définissant un rôle du système. En d'autres termes, cette fonction définit les conditions qu'une requête doit vérifier pour invoquer un rôle dans le système.

$$\text{MapRC} : \text{Rôle} \rightarrow \text{Condition}$$

$\text{MapRC}(r_i) = \text{Cond}$ tel que : i est un entier et $\text{Cond} \subseteq \text{Condition}$

8. Une requête d'accès

Une requête d'accès à une méthode d'un service Web au niveau d'un fournisseur de service doit être sous la forme :

$$\text{Requet\_acces} = (\text{attr\_sujet}, \text{par\_méthode}, \text{par\_contexte})$$

Où attr_sujet sont les attributs de l'utilisateur qui invoque la méthode, par_méthode sont les paramètres de la méthode invoquée et par_contexte sont les paramètres du contexte d'exécution de la requête.

attr_sujet et par_méthode sont envoyés avec la requête de demandeur de service et par_contexte sont capturés par le système de contrôle d'accès du fournisseur de service au moment de la réception de la requête.

Exemple : Dans le scénario de la section 4.3, un service Web du centre des statistiques envoie à chaque clinique d'un pays une requête d'invocation de la méthode qui calcule le nombre de patients atteints des maladies chroniques dans chaque clinique pour pouvoir calculer le nombre total de ces patients dans tout le pays.

9. la fonction d'assignation utilisateur-rôle

Quand le système de contrôle d'accès du fournisseur de service reçoit une requête du demandeur de service, il détermine le rôle correspondant au sujet à l'origine de la requête.
Cette assignation se base sur les attributs et les paramètres inclus dans la requête, ceux récupérés par le fournisseur de service concernant le contexte et la fonction d'assignation rôle-condition. La fonction est définie comme suit :

$$\text{MapRR} : \text{Requet} \rightarrow \text{Rôle}$$

$$\begin{aligned}\text{MapRR (Requet\_acces)} = \{r_i \;/\; \exists h\, (cond_h \in \text{MapRC}(r_i)) \\ \wedge\, [\forall k\, ((\text{paramètre}_k, \text{valeur}_k) \in cond_h) \\ \exists l\, ((\text{paramètre}_l, \text{valeur}_l) \in \text{Requet\_acces} \\ \wedge\ \text{paramètre}_l = \text{paramètre}_k \\ \wedge\ \text{valeur}_l \in \text{Vparamètre}_k)\,]\}\end{aligned}$$

Où paramètre_i peut être un attribut de sujet, un paramètre d'une méthode ou un paramètre du contexte.

Et Vparamètre_k est soit VA_k, VP_k ou VC_k

10. la fonction d'assignation permission-rôle

La fonction d'assignation entre l'ensemble des permissions et l'ensemble des rôles permet de déterminer l'ensemble de tous les rôles qui ont le privilège d'utiliser une permission donnée. Cette fonction est donnée par :

$$\text{MapPR} : \text{Permission} \rightarrow \text{Rôle}$$

$$\text{MapPR}(P_i) = \text{Rôle'} \qquad \text{tel que Rôle'} \subseteq \text{Rôle}$$

La fonction **MapPR** utilise les règles qui seront définies dans la section concernant le modèle de contrôle de flux d'information.

11. Séparation des tâches (SDT)

La séparation des tâches (en anglais separation of duties) a pour objectif la réduction du risque de fraude. Elle est représentée généralement par un ensemble de rôles mutuellement exclusifs. La SDT empêche un utilisateur de jouer (ou d'activer) deux rôles contradictoires.

L'ensemble des rôles mutuellement exclusifs : il est définit comme suit :

$$RE = \{r_1, r_2, ..., r_k\}$$

Tel que : $r_i \in$ Rôle et $0 \leq k < |\text{Rôle}|$

Une session : une session est une instance particulière d'une connexion d'un utilisateur au système, elle inclut des rôles actifs, le temps de sa création, des valeurs ID et Clé des sujets, etc. Elle est représentée par une structure de données qui contient les champs pour ID, clé et Srôle (ensemble des rôles actifs).

L'administrateur de sécurité spécifie les opérations qui sont mutuellement exclusives pour chaque ressource. Ainsi, quand l'ensemble des permissions est produit, le système aussi produit l'ensemble des permissions mutuellement exclusives (PME). En utilisant la fonction d'assignation des permissions aux rôles, on obtient l'ensemble des rôles RE tel que :

$$|\text{Session.Srôle} \cap \text{RE}| \leq 1$$

5.2.1.2 L'algorithme de contrôle d'accès

L'algorithme (algorithme 5.1) résume toutes les étapes qu'un système de contrôle d'accès au niveau d'un fournisseur de service effectue suite à une réception d'une requête d'invocation d'une méthode d'un service Web.

Dans notre modèle, nous avons considéré uniquement la séparation statique des tâches contrairement au modèle ARBAC adapté [LIU05] qui utilise la séparation statique et la séparation dynamique des tâches. En effet, si les rôles sont clairement définis au moment de la conception du système des services Web et si toutes les opérations mutuellement exclusives sont spécifiées par l'administrateur alors en empêchant l'activation de deux rôles exclusifs avant l'exécution de la requête ils ne pourront être activés ultérieurement.

De plus, dans notre algorithme, la fonction d'assignation de requêtes retourne un seul rôle (étape 1) contrairement au modèle proposé dans [LIU05] qui retourne un ensemble de rôle pour un utilisateur donné. En effet, une requête dans notre modèle spécifie les attributs de demandeur de services et les paramètres de la méthode qu'on veut invoquer ce qui permet

d'activer un seul rôle pour une requête donnée. Cependant notre modèle permet aussi à un utilisateur d'activer plusieurs rôles en demandant différentes méthodes des services Web.

Algorithme 5.1 Algorithme du contrôle d'accès

```
Entrées : Requet_acces, action, ressource    // la requête d'accès et l'action qu'on veut effectuer sur le ressource
Sorties : {autoriser, non autoriser}
Début
   1. Role ← MapRR (Requet_acces)                    // assignation d'une requête à un rôle
   2. PerRoles ← MapPR (action, ressource)           // assignation de la permission aux rôles
   3. Si Role ∈ PerRoles alors
         Si ¬ ∃ Session (Requet_acces. attr_sujet) alors   // s'il n'existe pas une session pour le sujet
            Session ← Créer_Session (Requet_acces. attr_sujet)   // créer une nouvelle session
         Sinon
            Session ←Récupérer_session (Requet_acces. attr_sujet) // récupération de la session déjà
                                                                   // activée
         FinSi
      FinSi
   4. Si |Session. Srôle ∩ RE| >1                    // le contrôle de la séparation des fonctions
      alors
         Retourner (non autoriser)                   // la requête est refusée
      Sinon
        Session.AjouterRôle (Role)                   // ajouter le rôle activé à la session de l'utilisateur
        Retourner (autoriser)                        // la requête est autorisée
      FinSi
Fin
```

5.2.2 Le modèle de contrôle de flux d'information

Comme nous l'avons mentionné précédemment, la fonction d'assignation des permissions aux rôles utilise les règles définies par le modèle de contrôle de flux d'information pour permettre non seulement le contrôle d'accès aux objets, comme dans le cas du RBAC traditionnel et d'autres modèles dérivés de ce dernier, mais aussi le contrôle de flux d'informations entre ces objets.

Dans ce qui suit, nous allons présenter une nouvelle version pour le modèle de CFI présenté dans [BER06].

5.2.2.1 Description du modèle de CFI

Ce modèle considère deux ensembles disjoints : l'ensemble des rôles et l'ensemble des objets (figure 5.1). Les rôles représentent les positions de travail ou les fonctions de même niveau d'autorité et de responsabilité dans le système. Un rôle regroupe un ensemble de méthodes des services Web et une méthode peut appartenir à plusieurs rôles selon les valeurs de ses paramètres. Il y a une relation d'hiérarchie entre ces rôles. Les objets sont des données XML et ils sont aussi organisés hiérarchiquement selon la structure hiérarchique des documents XML, et possédés par plusieurs rôles.

A chaque objet est associé un label contenant un nombre de composantes, où chaque composante correspond à la politique d'un propriétaire de cet objet.

Un rôle est activé au sein d'un fournisseur de service pour effectuer un certain nombre de fonctions qui manipulent les objets du système.

Un propriétaire d'un objet partage une relation de confiance avec quelques autres propriétaires d'un même objet et définit l'ensemble de lecteurs et l'ensemble de rédacteurs qui peuvent manipuler cet objet.

Les services Web peuvent invoquer d'autres services Web. On dit qu'un service s_1 invoque s_2 si s_1 contient au moins une méthode ($method_1$) qui invoque une autre méthode de s_2 ($method_2$). Par conséquent, le rôle activé pour la méthode $method_1$ induit à l'activation du rôle correspondant à la méthode $method_2$. Ceci peut se produire si les systèmes de contrôle d'accès des deux services le permettent.

Le flux d'information est causé soit par une opération d'invocation ou une opération de calcul.

Les composants de base cités ci-dessus sont illustrés par le modèle de la figure 5.1. Ils seront détaillés dans les prochaines sections.

5.2.2.2 Notations et définitions

- **L'ensemble d'objets**

Les objets représentent des données sous forme de document XML. Il peut être un élément de ce document, un document ou une collection de documents.

L'ensemble d'objets est défini comme suit

$$Q = \{q_1, q_2, \ldots, q_m\} \; ; \quad \text{Où m est le nombre d'objets dans le système}$$

Nous supposons que tous les objets sont disjoints entre eux. Ainsi, un objet a au plus un père. Ils sont spécifiés avec le langage XPath [W3C99].

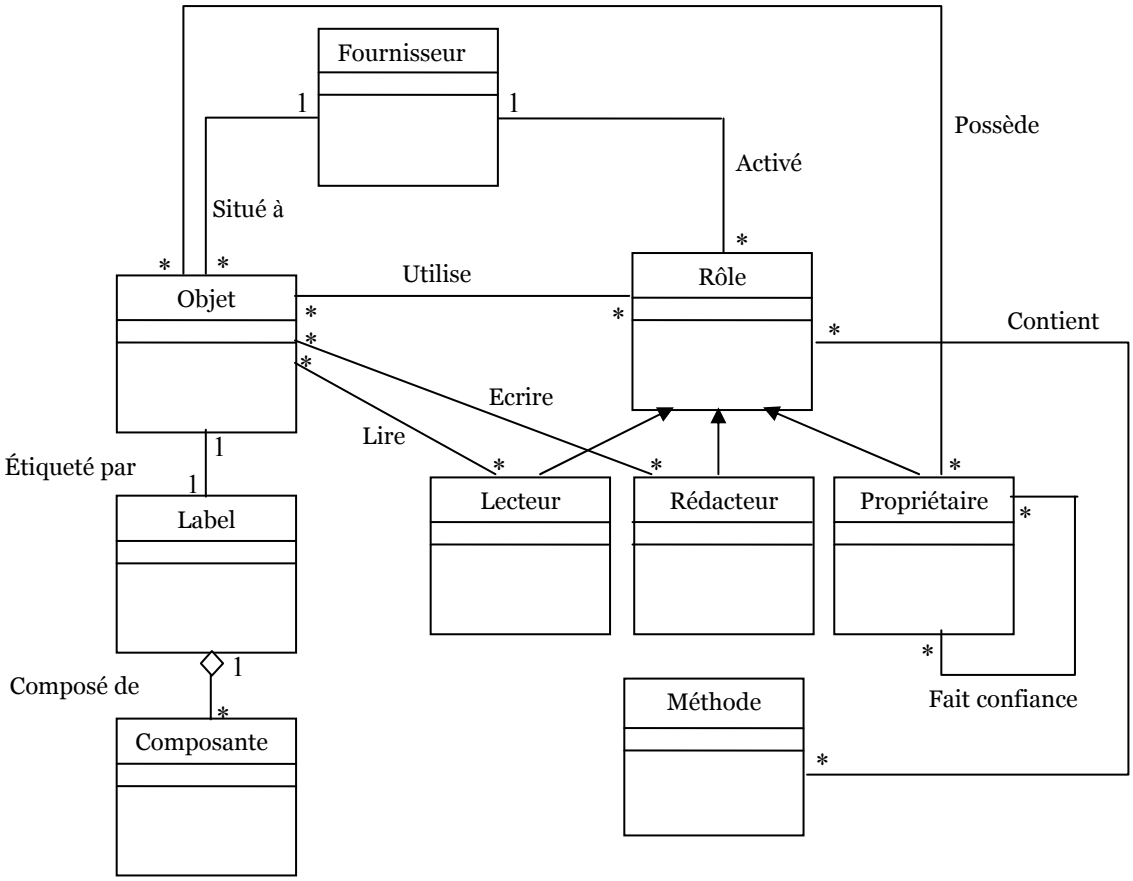


Figure 5.1 : Les composants du modèle de CFI

La relation d'hiérarchie :

Cette relation reflète la structure hiérarchique des données XML. Elle est définie comme suit

$$HO = \{(q_1, q_2) / q_1, q_2 \in Q \wedge q_2 \text{ est un sous élément de } q_1\}$$

Où Q est l'ensemble des objets,

q_1 est l'élément père,

Et q_2 est l'élément fils

q_1 et q_2 sont des documents XML, des éléments XML ou des attributs XML.

On dit que q_1 est le père de q_2 si q_2 est un sous élément de q_1 ou si q_1 est un document et q_2 est la racine de ce document.

Cette relation est notée par $q_1 \rightarrow_{ho} q_2$. Elle est réflexive, transitive et antisymétrique.

La figure 5.2 montre cette relation pour le scénario énoncé précédemment.

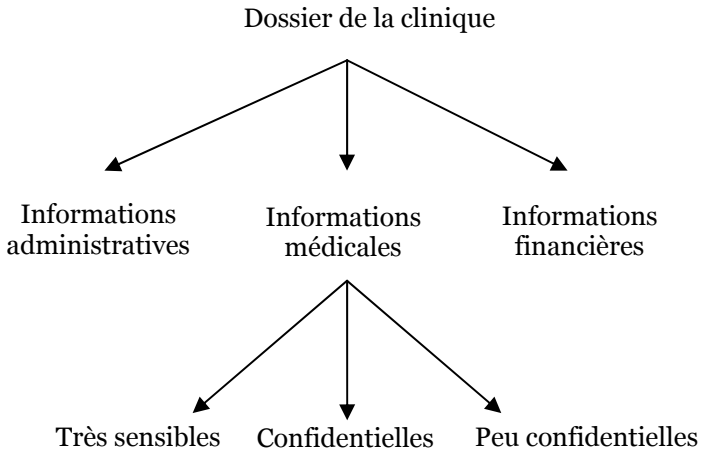


Figure 5.2 : Exemple d'hiérarchie d'objets

Le dossier contenant toutes les information de la clinique est le père des informations administratives, des informations médicales et des informations financières. Les informations médicales à leur tour contiennent des informations de différents niveaux de sensibilité ; très sensibles, confidentielles et peu confidentielles.

- **L'ensemble de rôles**

Cet ensemble contient tous les rôles identifiés dans le système. Il est défini comme suit :

$$R = \{r_1, r_2, ..., r_n\} = \text{Rôle}$$

Où Rôle est défini dans la section du modèle de contrôle d'accès.

n est le nombre de tous les rôles

et r_i est un rôle défini sur un fournisseur de service.

Un fournisseur F de service a un ensemble de rôles locaux ($R_p \subseteq R$) et un ensemble d'objet locaux ($Q_p \subseteq Q$).

Chaque rôle représente un ensemble de méthodes et possède un ensemble d'objets. Formellement on a:

$$\forall r \in R, \text{Method}_r = \{ \text{method}_1^r, \text{method}_2^r, ..., \text{method}_k^r \}$$

$$Q_r = \{q_1^r, q_2^r, ..., q_l^r\}$$

Où method_i^r est la i^{ième} méthode du rôle r ($1 \le i \le k$)

Et q_j^r est le j^{ième} objet dont r est le propriétaire ($1 \le j \le l$). Il se trouve ou non sur le même fournisseur de service.

La relation d'hiérarchie :

Cette relation représente la structure hiérarchique des rôles d'un système. Elle est définie comme suit

$$HR = \{(r_1, r_2) / r_1, r_2 \in R \wedge (r_1 \geq r_2)\}$$

Cette relation est notée par $r_1 \rightarrow_{hr} r_2$. Elle est réflexive, transitive et antisymétrique.

La figure suivante montre un exemple d'hiérarchie des rôles.

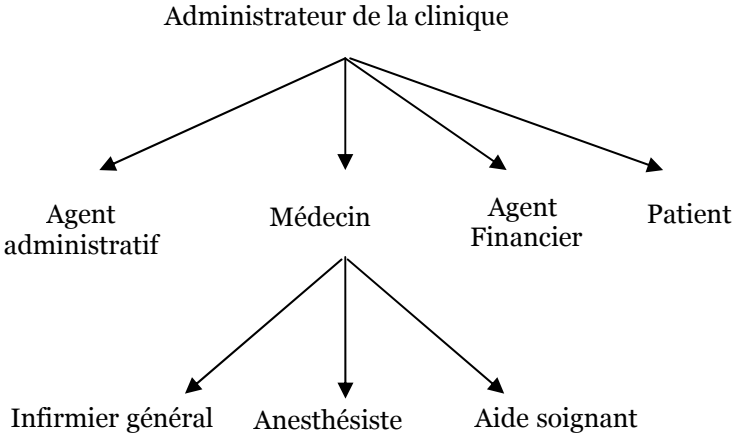


Figure 5.3 : Exemple d'hiérarchie des rôles

Le rôle administrateur de la clinique est supérieur aux rôles : agent administratifs, le médecin, l'agent financier et le patient où les quatre derniers rôles sont incomparables. Le médecin est aussi supérieur aux rôles infirmier général, anesthésiste et aide soignant. Ainsi le médecin dans cette hiérarchie a toutes les permissions assignées à l'infirmier général, l'anesthésiste et l'aide soignant

- **L'ensemble de propriétaires d'un objet**

Pour chaque objet $q \in Q$ est défini un ensemble de rôles qui le possèdent ou qui possèdent son père selon la relation d'hiérarchie. Cet ensemble est défini comme suit :

$$O_q = \{r \in R / q \in Q_r \vee [\exists q' \in Q / q' \rightarrow_h q \wedge r \in O_{q'}]\}$$

- **Les propriétaires de confiance**

A. La relation de confiance

La relation de confiance (***trust*** en anglais) reflète l'intention des membres d'un rôle envers les membres des autres rôles. On peut définir formellement cette relation par un ensemble ordonné de paires de rôles :

$$T = \{(r_1, r_2) / r_1, r_2 \in R \wedge r_1 \text{ fait confiance à } r_2\}$$

Nous nous intéressons à la relation de confiance entre les propriétaires d'un même objet.

B. L'ensemble de propriétaires de confiance

Chaque propriétaire d'un objet a un sous ensemble des propriétaires de ce même objet auxquels il fait confiance. Il est défini par :

$$TO_{q,o} = \{r \in O_q / o \in O_q \wedge (o, r) \in T\}$$

C. Les propriétaires de confiance effectifs

Quand tous les propriétaires d'un même objet font confiance à un propriétaire, il devient un *propriétaire effectif*. L'ensemble de tous les propriétaires effectifs (Effective Owners) d'un objet q est donné par :

$$EO_q = \bigcap_{oi \in Oq} TO_{q,oi}$$

- **La confidentialité des objets**

Pour assurer la confidentialité de l'information d'un objet, chaque propriétaire de cet objet peut choisir un sous ensemble de rôles qui peuvent lire cet objet.

A. L'ensemble de lecteurs

L'ensemble de lecteurs (Reader Set) d'un objet défini par un propriétaire de ce dernier est un sous ensemble de rôles. Il est donné par :

$$R_{q,o} = \{r \in R / \text{les conditions 1 et 2 sont vérifiées}\}$$

- **Condition 1 :** $o \in O_q$
- **Condition 2 :** $(r' \in R_{q,o} \wedge r \rightarrow_{hr} r') \vee$ (o permet à r de lire q)

B. L'ensemble de lecteurs effectifs

Un objet peut effectivement lire un objet (lecteur effectif) s'il est membre de tous les ensembles de lecteurs définis par les propriétaires de cet objet. L'ensemble de tous les lecteurs effectifs d'un objet q est :

$$ER_q = \bigcap_{oi \in Oq} R_{q,oi}$$

C. L'ensemble de lecteurs communs

Cet ensemble représente tous les rôles qui peuvent lire un objet q. Il contient tous les lecteurs effectifs de l'objet et ses propriétaires.

$$JR_q = O_q \cup ER_q$$

- **L'intégrité des objets**

Pour assurer l'intégrité de l'information d'un objet, chaque propriétaire de cet objet peut choisir un sous ensemble de rôles qui peuvent écrire (ou modifier) cet objet.

A. L'ensemble de rédacteurs

L'ensemble de rédacteurs (Writer Set) d'un objet défini par un propriétaire de ce dernier est un sous ensemble de rôles. Il est donné par :

$$W_{q,o} = \{r \in R \text{ / les conditions 1 et 2 sont vérifiées}\}$$

- **Condition 1 :** $o \in O_q$
- **Condition 2 :** $(r' \in W_{q,o} \wedge r \rightarrow_{hr} r') \vee$ (o permet à r d'écrire et modifier q)

B. L'ensemble de rédacteurs effectifs

Un objet peut effectivement écrire sur un objet (rédacteur effectif) s'il est membre de tous les ensembles de rédacteurs définis par les propriétaires de cet objet. L'ensemble de tous les rédacteurs effectifs d'un objet q est :

$$EW_q = \bigcap_{oi \in Oq} W_{q,oi}$$

C. L'ensemble de rédacteurs communs

Cet ensemble représente tous les rôles qui peuvent écrire une information dans un objet q. Il contient tous les rédacteurs effectifs de l'objet et ses propriétaires.

$$JW_q = O_q \cup EW_q$$

- **La propagation des politiques**

En raison de la structure hiérarchique des objets, une relation de propagation des politiques d'un objet père à un objet fils est définie. Chaque propriétaire o définit sa politique de propagation sur son objet q. Cette politique est notée par $P_{q,o}$ et elle est définie par une pair :

(règles d'acceptation, règles de propagation)

Où les règles d'acceptation concernent les objets fils et définissent quand accepter la propagation des politiques des labels des objets pères. Les règles de propagation concernent les objets pères et définissent quand propager les politiques de ces derniers vers les objets fils. On note une règle de propagation d'un propriétaire o pour son objet q par $P_{q,o}$ tel que :

$$P_{q,o} = (p_{accepter}, p_{propager})$$

Les valeurs spéciales pour ces règles sont :

- ***Une règle faible :*** qui veut dire *toujours accepter* pour la règle d'acceptation ou *non propager* pour la règle de propagation.
- ***Une règle forte :*** qui veut dire *jamais accepter* pour la règle d'acceptation et *toujours propager* pour la règle de propagation.

- **La relation d'invocation**

Les services Web peuvent échanger de l'information au moyen d'invocation des méthodes.

On dit qu'il y a une relation d'invocation entre deux services s_1 et s_2 s'il y a une méthode de s_1 capable d'appeler une autre méthode de s_2 et que la dernière envoie une réponse à la première.

On note cette relation par :

$$s_1 \rightarrow_i s_2$$

Cette relation est réflexive, symétrique et transitive.

Dans l'exemple de motivation, le centre national des statistiques peut calculer le nombre total des patients d'une maladie chronique en Algérie pour une période en invoquant la méthode qui retourne le nombre de ces patients dans tous les services Web des cliniques du pays.

- **Une transaction**

C'est une séquence d'invocations de services Web initié par un utilisateur. Les méthodes des services Web appèlent d'autres méthodes du même service ou des autres services. Les invocations des méthodes forment un arbre d'invocation où chaque branche de la racine à une feuille est appelée une ***transaction***.

Une transaction T peut être représentée comme suit :

$$\text{Utilisateur} \rightarrow_i^T s_1 \rightarrow_i^T s_2 \rightarrow_i^T \ldots \rightarrow_i^T s_n$$

On note par :

init(T) le premier service invoqué par l'initiateur de la transaction (ici s_1)

last(T) l'avant dernier service qui demande l'accès à un objet (ici s_{n-1})

- **La dé-classification des objets**

1. Cas de lecture (confidentialité)

La dé-classification en cas de lecture permet à un rôle de lire un objet même s'il n'est pas parmi les lecteurs effectifs. Cela permet donc à une transaction initiée par un service d'un rôle autorisé à lire un objet de continuer son exécution quoique l'un des services invoqués ne soit pas autorisé à lire cet objet. L'ensemble de dé-classification d'un propriétaire o sur son objet q peut être défini comme suit :

$$DR_{q,\,o} = \{\delta_1, \delta_2, \ldots, \delta_p\}$$

p : est le nombre de composantes de dé-classification pour le propriétaire o.

Chaque composante δ contient deux parties :

1. $\delta_{init} \in R_{q,o}$ le rôle au profit duquel la dé-classification va apparaître.
2. $\delta_{inter} \subseteq R - R_{q,o}$ le sous ensemble de rôles intermédiaires pour lesquels la dé-classification va être effectuée tel que $\{\delta_{init}\} \cap \delta_{inter} = \varnothing$

L'ensemble de dé-classification effectif en cas de lecture : c'est l'intersection de tous les ensembles de dé-classification des propriétaires effectifs:

$$EDR_q = \bigcap_{oi \in EOq} DR_{q,oi}$$

Où l'intersection entre deux ensembles de dé-classification est définit par cet ensemble :

$$DR_{q,o1} \cap DR_{q,o2} = \{(\delta_{init}, \delta_{inter}) \,/\, \exists\ \delta_i \in DR_{q,o1} \wedge \delta_j \in DR_{q,o2} \quad (\delta_{iinit} = \delta_{jinit} = \delta_{init} \wedge \delta_{inter} = \delta_{iinter} \cap \delta_{jinter})\}$$

2. Cas d'écriture (intégrité)

La dé-classification en cas d'écriture permet à un rôle d'écrire un objet même s'il n'est pas parmi les rédacteurs effectifs. Cela permet donc à une transaction initiée par un service d'un rôle autorisé à écrire un objet de continuer son exécution quoique l'un des services invoqués ne soit pas autorisé à écrire cet objet. L'ensemble de dé-classification d'un propriétaire o sur son objet q peut être défini comme suit :

$$DW_{q,\,o} = \{\delta_1, \delta_2, \ldots, \delta_p\}$$

p : est le nombre de composantes de dé-classification pour le propriétaire o.

Chaque composante δ contient deux parties :

3. $\delta_{init} \in W_{q,o}$ le rôle au profit duquel la dé-classification va apparaître.
4. $\delta_{inter} \subseteq R - W_{q,o}$ le sous ensemble de rôles intermédiaires pour lesquels la dé-classification va être effectuée tel que $\{\delta_{init}\} \cap \delta_{inter} = \varnothing$

L'ensemble de dé-classification effectif en cas d'écriture: c'est l'intersection de tous les ensembles de dé-classification des propriétaires effectifs:

$$EDW_q = \bigcap_{oi \in EOq} DW_{q,oi}$$

Où l'intersection entre deux ensembles de dé-classification est définit par cet ensemble :

$$DW_{q,o1} \cap DW_{q,o2} = \{(\delta_{init}, \delta_{inter}) / \exists\ \delta_i \in DW_{q,o1} \wedge \delta_j \in DW_{q,o2} \quad (\delta_{iinit} = \delta_{jinit} = \delta_{init} \wedge \delta_{inter} = \delta_{iinter} \cap \delta_{jinter})\}$$

- **Le label**

Un label est une structure de donnée attachée à un objet. Il contient toutes les politiques des propriétaires concernant le contrôle d'accès et de flux d'information pour l'objet correspondant.

Un label est un ensemble de composantes où chaque composante représente la politique d'un propriétaire sur son objet. Une composante à son tour contient le propriétaire, l'ensemble de propriétaires de confiances spécifié par ce propriétaire, l'ensemble de lecteurs, l'ensemble de rédacteurs, les règles de propagation et les règles de dé-classification. Le label d'un objet q est noté par L(q). La structure d'un label est définie comme suit :

$$L(q) = \{K_1, K_2, \ldots, K_c\}$$

Où c est le nombre de composantes.

Une composante K_i est un 7-uplet.

Posons $K_i = K$ alors K est donnée par :

$$K = \{K_o, TO_{q,Ko}, R_{q,Ko}, W_{q,Ko}, P_{q,Ko}, DR_{q,Ko}, DW_{q,Ko}\}$$

Où :

- $\mathbf{K_o} \in R$ est le propriétaire de q dans la composante K.
- $\mathbf{TO_{q,Ko}}$ est l'ensemble de propriétaires de confiance de K_o.
- $\mathbf{R_{q,Ko}}$ est l'ensemble de lecteurs de q défini par K_o.

- $\mathbf{W_{q,Ko}}$ est l'ensemble de rédacteurs de q défini par K_o.
- $\mathbf{P_{q,Ko}}$ est l'ensemble de règles de propagation
- $\mathbf{DR_{q,Ko}}$ est l' ensemble de politiques de dé-classification pour l'opération de lecture.
- $\mathbf{DW_{q,Ko}}$ est l' ensemble de politiques de dé-classification pour l'opération d'écriture.

- **Un flux d'information**

Un flux d'information concerne toute transmission d'information entraînant une affectation entre les objets impliqués.

Un flux d'information peut être explicite (par l'assignation directe de valeurs ou opération de calcul) ou implicite (par l'utilisation des instructions conditionnelles ou de boucles).

Les règles de flux d'information définies dans ce modèle concernent le flux explicite.

L'opérateur → est introduit pour exprimer la direction de flux d'information. Dans chaque forme de flux d'information, les propriétés de label de l'objet source sont portées à l'objet destination [TAR06].

Il peut y avoir un flux d'information entre deux objets q_1 et q_2 (i.e. $q_1 \rightarrow q_2$) si et seulement s'il y a:

- ✓ Une opération de lecture de l'objet q_1
- ✓ Transmission d'information vers q_2
- ✓ Une opération d'écriture sur l'objet q_2

5.2.2.3 Les règles d'accès

Dans ce qui suit, nous donnons l'ensemble des règles qui gèrent l'assignation des permissions aux rôles pour accéder aux objets du système et à leurs labels.

Reprenons la fonction d'assignation des permissions aux rôles définie dans la partie contrôle d'accès (cf. § 5.2.1.1)

$$\text{MapPR}(P_i) = \text{Rôle' tel que Rôle'} \subseteq \text{Rôle}$$

Où Rôle = R

On aura donc $\text{MapPR}(P_i) = R' \text{ tel que } R' \subseteq R$

A. Les règles d'assignation des permissions d'accès aux données

Les règles suivantes sont responsables d'assigner des permissions aux rôles pour accéder aux données ou en d'autres termes aux objets du système.

Règle 1 :

Cette règle définit l'ensemble des rôles qui peuvent lire un objet q

$$MapPR(lire, q) = \{r / r \in JR_q\}$$

La règle peut être interprétée comme suit : un rôle peut lire l'objet q si et seulement si ce rôle appartient à l'ensemble des lecteurs communs

Règle 2 :

Cette règle définit l'ensemble des rôles qui peuvent écrire (modifier) un objet q

$$MapPR(écrire, q) = \{r / r \in JW_q\}$$

Règle 3 :

Cette règle définit l'ensemble des rôles qui peuvent créer un objet q_2 comme fils de l'objet q_1

$$MapPR(créer, q_2) = \{r / r \in O_{q1} \wedge q_1 \rightarrow_{ho} q_2\}$$

Règle 4 :

Cette règle définit l'ensemble des rôles qui peuvent supprimer un objet q

$$MapPR(supprimer, q) = \{r / r \in O_q \wedge |O_q| = 1\}$$

B. Les règles d'assignation des permissions d'accès aux labels des objets

Ces règles font l'assignation permissions-rôles pour la modification des contenus des labels des objets qui revient à la modification des droits d'accès aux objets.

Pour cela, on note cette fonction comme suit

$$MapPR(actions, E)$$

Où : **actions** est un ensemble d'actions possibles

E est l'un des sous ensembles de rôles définis dans la partie *notations et définitions* (exemple : $R_{q,Ko}$).

Règle 5 :

Cette règle contrôle la modification de l'ensemble de lecteurs d'un objet q.

$$MapPR(\{lire, écrire, ajouter, supprimer\}, K.R_{q,Ko}) = \{K_o\}$$

K est une composante de L(q).

La règle peut être interprétée comme suit : seulement le propriétaire d'une composante dans un label peut lire, écrire, ajouter et supprimer un élément de l'ensemble de lecteurs de la même composante

Règle 6 :

Cette règle contrôle la modification de l'ensemble de rédacteurs d'un objet q

$$\text{MapPR}(\{\text{lire, écrire, ajouter, supprimer}\}, K.\ W_{q,K_o}) = \{K_o\}$$

K est une composante de L(q).

Règle 7 :

Cette règle contrôle la modification de l'ensemble des propriétaires de confiance d'un objet q

$$\text{MapPR}(\{\text{lire, écrire, ajouter, supprimer}\}, K.\ TO_{q,K_o}) = \{K_o\}$$

K est une composante de L(q).

Règle 8 :

Cette règle contrôle le changement d'un propriétaire d'un objet q (i.e. Ajouter ou supprimer un propriétaire de q)

$$\text{MapPR}(\text{ajouter}, K.\ O_q) = \{r \mid r \in EO_q\}$$

$$\text{MapPR}(\text{supprimer}, K.\ O_q.o) = \{r \mid r \in EO_q \wedge o \notin EO_q\}$$

K est une composante de L(q).

Un rôle qui appartient à l'ensemble des propriétaires effectifs peut ajouter des propriétaires au label de l'objet mais il ne peut supprimer que les propriétaires qui ne sont pas des propriétaires effectifs.

Règle 9 :

Cette règle contrôle le changement des lecteurs effectifs d'un objet q. Les propriétaires effectifs peuvent déclassifier leurs objets en étendant l'ensemble des lecteurs effectifs.

$$\text{MapPR}(\text{ajouter}, ER_q) = \{r \mid r \in EO_q\}$$

Règle 10 :

Cette règle contrôle le changement des rédacteurs effectifs d'un objet q. Les propriétaires effectifs peuvent déclassifier leurs objets en étendant l'ensemble des rédacteurs effectifs.

$$\text{MapPR}(\text{ajouter}, EW_q) = \{r \mid r \in EO_q\}$$

Règle 11 : <propagation>

Les propriétaires des objets pères sont aussi des propriétaires de leurs fils.

$q_1 \rightarrow_{ho} q_2 \wedge r \in O_{q1} \Rightarrow$ ajouter (r) à O_{q2}

Règle 12 : <dé-classification de politiques de confidentialité>

Soit T une transaction.

Si pour chaque composante $K \in L(q)$ et $K_o \in EO_q$ il existe δ_i tel que :

$\delta_{iinit} = init(T) \wedge last(T) \in \delta_{iinter}$

Alors

$last(T) \in ER_q$

On peut écrire cette règle comme suit :

$(r_1, \varepsilon) \in EDR_q \wedge r_1 \in ER_q \Rightarrow$ ajouter (ε) à ER_q

Règle 13 : <dé-classification de politiques d'intégrité>

Soit T une transaction.

Si pour chaque composante $K \in L(q)$ et $K_o \in EO_q$ il existe δ_i tel que :

$\delta_{iinit} = init(T) \wedge last(T) \in \delta_{iinter}$

Alors

$last(T) \in EW_q$

On peut écrire cette règle comme suit :

$(r_1, \varepsilon) \in EDW_q \wedge r_1 \in EW_q \Rightarrow$ ajouter (ε) à EW_q

Règle 14 : <Un flux d'information sain>

Un flux d'information est dit sain (sûr) si :

1. il ne mène pas à une divulgation d'information transmise ou encore garantit la confidentialité de l'information issue de l'objet source. Cela veut dire qu'avant et après le flux d'information, il n'y aura pas plus de lecteurs autorisés pour cette information.
2. il ne mène pas à une écriture non autorisée dans objet ou encore garantit l'intégrité de l'objet destination avant et après le flux d'information. Cela veut dire que
   a. avant que le flux d'information soit effectué, on doit s'assurer que l'information de l'objet source est susceptible d'être écrite par des rédacteurs autorisés par la politique de l'objet destination. et
   b. après le flux d'information, on doit s'assurer que l'objet destination n'aura pas plus de rédacteurs que ceux autorisés avant le flux d'information.

Formellement, on peut écrire :

$$q_1 \rightarrow q_2 \text{ est sain } \Leftrightarrow \begin{cases} JR_{q2} \subseteq JR_{q1} \\ EDR_{q2} \subseteq EDR_{q1} \end{cases}$$

De plus,

$$\begin{cases} JW_{q1} \subseteq JW_{q2} \\ EDW_{q1} \subseteq EDW_{q2} \end{cases} \text{ Et } \begin{cases} JW'_{q2} \subseteq JW_{q2} \\ EDW'_{q2} \subseteq EDW_{q2} \end{cases}$$

Où JW'_{q2} et EDW'_{q2} sont les nouveaux ensembles calculés pour q_2 après le flux d'information.

5.2.2.4 Les règles de mise à jour des labels

Les règles de mise à jour des labels (en anglais *relabeling rules*) sont imposées par l'aspect dynamique des services Web où de nouveaux objets sont crées ou modifiés pendant un processus de calcul.

Dans ce qui suit, nous allons montrer comment les labels existants sont mis à jour et comment dériver de nouveaux labels pour les nouveaux objets crées.

Pour ce faire, un opérateur noté "_" est utilisé pour indiquer la jointure des labels.

La jointure des labels est effectuée pour calculer le label de l'objet obtenu avec l'opération d'assignation, de calcul, de propagation ou de dé-classification. Elle est notée comme suit :

$$L(\text{nouvel objet}) = L(\text{objet}_1) \_ L(\text{objet}_2)$$

Nous avons introduit des modifications au niveau des définitions des jointures présentées dans [BER06] ainsi qu'aux preuves de sûreté et aux algorithmes correspondants. Et ce pour préserver les deux propriétés suivantes même après le flux d'information :

1. garantir la confidentialité des informations transmises.
2. garantir l'intégrité des objets.

1) Les jointures de calcul

Nous définissons trois formes différentes de jointures de calcul ; la jointure d'assignation $\_^a$, la jointure restrictive $\_^r$ et la jointure de fusion $\_^f$.

A. La jointure d'assignation

Cette jointure est idéale dans le cas d'assignation des objets. Au lieu de maintenir exactement le label source comme un nouveau label de l'objet destination, il est combiné avec le label de l'objet destination.

La jointure d'assignation donnant un nouveau label pour l'objet destination est définie comme suit :

Soient $q_1, q_2 \in Q$, $L(q_1)$, $L(q_2)$ les labels de q_1, q_2 respectivement, $L(x)$ un nouveau label de q_2

Si $L(x) = L(q_1) \_^a L(q_2)$ alors :

- $O_x = O_{q1} \cup (O_{q2} \cap JR_{q1})$
- $\forall\ o_i \in O_x \setminus O_{q1}$ $(o_i \in O_x$ et $o_i \notin O_{q1})$ et $\forall\ o_j \in O_{q1}$
  - $R_{x,oi} = ER_{q1} \cup R_{q2,oi}$; $R_{x,oj} = R_{q1,oj}$
  - $W_{x,oi} = W_{q2,oi}$; $W_{x,oj} = W_{q1,oj} \cap JW_{q2}$
  - $TO_{x,oi} = EO_{q1} \cup TO_{q2,oi}$; $TO_{x,oj} = TO_{q1,oj}$
  - $DR_{x,oi} = EDR_{q1} \cup DR_{q2,oi}$; $DR_{x,oj} = DR_{q1,oj}$
  - $DW_{x,oi} = DW_{q2,oi}$; $DW_{x,oj} = DW_{q1,oj} \cap EDW_{q2}$
  - $P_{x,oi} = P_{q2,oi}$; $P_{x,oj} = P_{q1,oj}$
- $L(q_2) \leftarrow L(x)$

L'algorithme suivant permet l'implémentation de la jointure d'assignation.

Algorithme 5.2 : jointure d'assignation

```
Entrées : L(q1), L(q2)        Sorties : L(q2) = L(q1) _a L(q2)
Début
    L(x) ← ∅
    Pour chaque oj ∈ Oq1 tel que 1≤ j ≤ |Oq1|
        % |Oq1| : nombre de composantes de L(q1)
        K ← { oj, TOq1,oj, Rq1,oj, Wq1,oj ∩ JWq2, Pq1,oj, DRq1,oj, DWq1,oj ∩ EDWq2}
        L(x).ajouterComposante(K)
    FinPour
    Pour chaque oi ∈ Oq2 tel que 1≤ i ≤ |Oq2|        % |Oq2| : nombre de composantes de L(q2)
            Si (oi ∈ JRq1 ∧ oi ∉ Oq1) alors
              K ← { oi, EOq1 ∪ TOq2,oi, ERq1 ∪ Rq2,oi, Wq2,oi, Pq2,oi, EDRq1 ∪ DRq2,oi, DWq2,oi }
              L(x).ajouterComposante(K)
            FinSi
    FinPour
L(q2) ← L(x)
Fin
```

Lemme 1 : Le flux d'information utilisant la jointure d'assignation est sain.

Preuve:

Supposons qu'il y a un flux d'information $q_1 \rightarrow q_2$ causé par l'opération d'assignation.

Soient $L(q_1)$ et $L(q_2)$ les labels des objets q_1 et q_2 respectivement.

Comme il est déjà mentionné ci-dessus :

$$q_1 \to q_2 \text{ est sain} \Leftrightarrow \begin{cases} JR_x \subseteq JR_{q1} \quad \text{et} \quad EDR_x \subseteq EDR_{q1} & \textbf{(1)} \\ JW_x \subseteq JW_{q2} \text{ et} \quad EDW_x \subseteq EDW_{q2} & \textbf{(2)} \end{cases}$$

Nous voulons démontrer cette assertion: $L(q_2) \leftarrow L(q_1) \_^a L(q_2) \Rightarrow q_1 \to q_2$ est sain

Cela revient à montrer les expressions (1) et (2)

(1) $JR_x \subseteq JR_{q1}$ et $EDR_x \subseteq EDR_{q1}$

o D'après la définition de jointure d'assignation, nous avons: $O_x = O_{q1} \cup (O_{q2} \cap JR_{q1})$

Nous avons $O_{q2} \cap JR_{q1} \subseteq JR_{q1}$ et $O_{q1} \subseteq JR_{q1}$. Donc,

$$O_{q1} \cup (O_{q2} \cap JR_{q1}) \subseteq JR_{q1} \quad \text{Alors,}$$

$$O_x \subseteq JR_{q1} \qquad \text{(A.a)}$$

De plus, $\forall\, o_i \in (O_x \setminus O_{q1})$, $R_{x,oi} = ER_{q1} \cup R_{q2,oi}$

Nous pouvons noter que $(O_x \setminus O_{q1}) \subseteq O_{q2}$

Nous avons $ER_x = \bigcap_{oi \in Ox} R_{x,oi} = [\bigcap_{oi \in Oq1} R_{q1,oi}] \bigcap [\bigcap_{oi \in Ox \setminus Oq1} (ER_{q1} \cup R_{q2,oi})]$

$$ER_x = ER_{q1} \bigcap [\bigcap_{oi \in Ox \setminus Oq1} (ER_{q1} \cup R_{q2,oi})]$$

$$ER_{q1} \bigcap [\bigcap_{oi \in Ox \setminus Oq1} (ER_{q1} \cup R_{q2,oi})] \subseteq ER_{q1}$$

Donc, $ER_x \subseteq ER_{q1}$

Or $ER_{q1} \subseteq JR_{q1}$. Alors

$$ER_x \subseteq JR_{q1} \qquad \text{(A.b)}$$

D'après (A.a) et (A.b), nous déduisons que

$$JR_x \subseteq JR_{q1} \qquad \text{(A.c)}$$

o Nous avons d'autre part : $\forall\, o_i \in (O_x \setminus O_{q1})$, $DR_{x,oi} = EDR_{q1} \cup DR_{q2,oi}$

Et $EDR_x = \bigcap_{oi \in Ox} DR_{x,oi} = [\bigcap_{oi \in Oq1} DR_{q1,oi}] \bigcap [\bigcap_{oi \in Ox \setminus Oq1} (EDR_{q1} \cup DR_{q2,oi})]$

$$EDR_x = EDR_{q1} \bigcap [\bigcap_{oi \in Ox \setminus Oq1} (EDR_{q1} \cup DR_{q2,oi})]$$

Or $EDR_{q1} \bigcap [\bigcap_{oi \in Ox \setminus Oq1} (EDR_{q1} \cup DR_{q2,oi})] \subseteq EDR_{q1}$

Donc, nous avons, $EDR_x \subseteq EDR_{q1}$ (A.d)

(2) $JW_x \subseteq JW_{q2}$ et $EDW_x \subseteq EDW_{q2}$

o Nous avons avant le flux d'information : $JW_{q1} \subseteq JW_{q2}$

Donc $O_{q1} \cup EW_{q1} \subseteq JW_{q2}$ Alors

$$O_{q1} \subseteq JW_{q2} \qquad \text{(A.a')}$$

Nous avons aussi $O_{q2} \cap JR_{q1} \subseteq O_{q2}$ et $O_{q2} \subseteq JW_{q2}$ Donc

$$O_{q2} \cap JR_{q1} \subseteq JW_{q2} \qquad \text{(A.b')}$$

De (A.a') et (A.b') nous obtenons $O_{q1} \cup (O_{q2} \cap JR_{q1}) \subseteq JW_{q2}$

Nous déduisons que $O_x \subseteq JW_{q2}$ (A.c')

Nous avons $EW_x = \bigcap_{oi \in Ox} W_{x,oi} = [\bigcap_{oi \in Oq1} (W_{q1,oj} \cap JW_{q2})] \bigcap [\bigcap_{oi \in Ox \setminus Oq1} W_{q2,oi}]$

Nous avons aussi $W_{q1,oj} \cap JW_{q2} \subseteq JW_{q2}$

Donc $\bigcap_{\mathbf{oi \in Oq1}} (W_{q1,oj} \cap JW_{q2}) \subseteq JW_{q2}$

Ainsi $[\bigcap_{\mathbf{oi \in Oq1}} (W_{q1,oj} \cap JW_{q2})] \bigcap [\bigcap_{oi \in Ox \setminus Oq1} W_{q2,oi}] \subseteq JW_{q2}$

Par conséquent $EW_x \subseteq JW_{q2}$ (A.d')

De (A.c') et (A.d') nous obtenons $JW_x \subseteq JW_{q2}$ (A.e')

- Nous avons d'autre part :

$$EDW_x = \bigcap_{oi \in Ox} DW_{x,oi} = [\bigcap_{oi \in Oq1} (DW_{q1,oj} \cap EDW_{q2})] \bigcap [\bigcap_{oi \in Ox \setminus Oq1} DW_{q2,oi}]$$

Nous avons $DW_{q1,oj} \cap EDW_{q2} \subseteq EDW_{q2}$

Donc $\bigcap_{\mathbf{oi \in Oq1}} (DW_{q1,oj} \cap EDW_{q2}) \subseteq EDW_{q2}$

Ainsi $[\bigcap_{\mathbf{oi \in Oq1}} (DW_{q1,oj} \cap EDW_{q2})] \bigcap [\bigcap_{oi \in Ox \setminus Oq1} DW_{q2,oi}] \subseteq EDW_{q2}$

Par conséquent $EDW_x \subseteq EDW_{q2}$ (A.f')

Ainsi, d'après (A.c), (A.d), (A.e'), (A.f') et ($L(q_2) \leftarrow L(x)$) nous concluons que $q_1 \rightarrow q_2$ est sain.

B. La jointure restrictive $\_^r$

Cette jointure est idéale pour les procédures de calcul délicates. Elle donne à l'objet résultant les propriétés issues des propriétés des objets entraînés dans l'opération de calcul mais avec plus de restrictions.

La jointure restrictive qui calcule le label de l'objet contenant le résultat de l'opération en utilisant les labels des objets sources est définie comme suit :

Soient q_1, q_2, $q_3 \in Q$, $L(q_1)$, $L(q_2)$, $L(q_3)$ les labels de q_1, q_2, q_3 respectivement et L(x) un nouveau label.

Supposons qu'il y a une opération de calcul sur q_1 et q_2 : $q_3 = q_1 \; op \; q_2$

Si $L(x) = L(q_1) \_^r L(q_2)$ alors :

- $O_x = O_{q1} \cap O_{q2}$
- Si $O_x \neq \varnothing$ alors

  $\forall \; o_i \in O_x$

  - $R_{x,oi} = (R_{q1,oi} \cup R_{q2,oi}) \cap (JR_{q1} \cap JR_{q2})$
  - $W_{x,oi} = (W_{q1,oi} \cup W_{q2,oi}) \cap JW_{q3}$
  - $TO_{x,oi} = (TO_{q1,oi} \cup TO_{q2,oi}) \cap (EO_{q1} \cap EO_{q2})$
  - $DR_{x,oi} = (DR_{q1,oi} \cup DR_{q2,oi}) \cap (EDR_{q1} \cap EDR_{q2})$
  - $DW_{x,oi} = (DW_{q1,oi} \cup DW_{q2,oi}) \cap EDW_{q3}$

- $P_{x,oi}$ = more restrictive ($P_{q1,oi}$, $P_{q2,oi}$)

- Sinon

  $O_x = O_{q3} \cap (JR_{q1} \cap JR_{q2})$

  $\forall\ o_i \in O_x$

  - $R_{x,oi} = R_{q3,oi} \cap (JR_{q1} \cap JR_{q2})$
  - $W_{x,oi} = W_{q3,oi} \cap JW_{q3}$
  - $TO_{x,oi} = TO_{q3,oi}$
  - $DR_{x,oi} = DR_{q3,oi} \cap (EDR_{q1} \cap EDR_{q2})$
  - $DW_{x,oi} = DW_{q3,oi} \cap EDW_{q3}$
  - $P_{x,oi} = P_{q3,oi}$

- $L(q_3) \leftarrow L(x)$

Nous définissons dans le tableau 5.1 les opérations moins/plus restrictives pour les règles de propagation.

	Règle d'acceptation	Règle de propagation
Opération moins Restrictive	Faible et faible = faible Faible et forte = forte Forte et faible= forte Forte et forte= forte	Faible et faible = faible Faible et forte = faible Forte et faible = faible Forte et forte = forte
Opération plus Restrictive	Faible et faible = faible Faible et forte = faible Forte et faible = faible Forte et forte = forte	Faible et faible = faible Faible et forte = forte Forte et faible = forte Forte et forte = forte

Table 5.1 : Les opérations plus/moins restrictives pour les règles de propagation

L'algorithme 5.3 permet l'implémentation de la jointure restrictive.

Lemme 2 : Le flux d'information utilisant la jointure restrictive est sain.

Preuve:

Supposons qu'il y a un flux d'information de deux objets q_1, q_2 entraînés dans une opération (concaténation par exemple) vers un objet destinataire q_3.

Soient $L(q_1)$, $L(q_2)$ et $L(q_3)$ les labels des objets q_1, q_2 et q_3 respectivement, L(x) un nouveau label de q_3

Nous avons l'assertion : $L(q_3) = L(q_1) \_^r L(q_2) \Leftrightarrow (q_1 \rightarrow q_3$ est sain et $q_2 \rightarrow q_3$ est sain)

Nous voulons montrer que :

Algorithme 5.3 : jointure restrictive

Entrées : $L(q_1)$, $L(q_2)$, $L(q_3)$ Sorties : $L(q_3) = L(q_1) \_^r L(q_2)$

Début

$L(x) \leftarrow \varnothing$

Pour chaque $o_i \in O_{q1}$ tel que $1 \leq i \leq |O_{q1}|$ % $|O_{q1}|$: nombre de composantes de $L(q_1)$

Si $o_i \in O_{q2}$ alors

$K \leftarrow \{ o_i, (TO_{q1,oi} \cup TO_{q2,oi}) \cap (EO_{q1} \cap EO_{q2}), (R_{q1,oi} \cup R_{q2,oi}) \cap (JR_{q1} \cap JR_{q2}), (W_{q1,oi} \cup W_{q2,oi}) \cap JW_x,$ more restrictive $(P_{q1,oi}, P_{q2,oi}), (DR_{q1,oi} \cup DR_{q2,oi}) \cap (EDR_{q1} \cap EDR_{q2}), (DW_{q1,oi} \cup DW_{q2,oi}) \cap EDW_x \}$

$L(x)$.ajouterComposante(K)

FinSi

FinPour

Si $L(x) = \varnothing$ alors

Pour chaque $o_i \in O_{q3} \cap (JR_{q1} \cap JR_{q2})$

$K \leftarrow \{ o_i, TO_{q3,oi}, R_{q3,oi} \cap (JR_{q1} \cap JR_{q2}), W_{q3,oi} \cap JW_{q3}, P_{q3,oi}, DR_{q3,oi} \cap (EDR_{q1} \cap EDR_{q2}), DW_{q3,oi} \cap EDW_{q3}\}$

$L(x)$.ajouterComposante(K)

FinPour

FinSi

$L(q_3) \leftarrow L(x)$

Fin

$$q_1 \rightarrow q_3 \text{ est sain} \Leftrightarrow \begin{cases} JR_x \subseteq JR_{q1} \text{ et } EDR_x \subseteq EDR_{q1} \\ JW_x \subseteq JW_{q3} \text{ et } EDW_x \subseteq EDW_{q3} \end{cases}$$

Et

$$q_2 \rightarrow q_3 \text{ est sain} \Leftrightarrow \begin{cases} JR_x \subseteq JR_{q2} \text{ et } EDR_x \subseteq EDR_{q2} \\ JW_x \subseteq JW_{q3} \text{ et } EDW_x \subseteq EDW_{q3} \end{cases}$$

Cela revient donc à montrer que : $\begin{cases} JR_x \subseteq (JR_{q1} \cap JR_{q2}) \text{ et } EDR_x \subseteq (EDR_{q1} \cap EDR_{q2}) & \textbf{(3)} \\ JW_x \subseteq JW_{q3} \text{ et } EDW_x \subseteq EDW_{q3} & \textbf{(4)} \end{cases}$

(3) $JR_x \subseteq (JR_{q1} \cap JR_{q2})$ et $EDR_x \subseteq (EDR_{q1} \cap EDR_{q2})$

i. Cas : $O_{q1} \cap O_{q2} \neq \varnothing$

o D'après la définition de la jointure $O_x = O_{q1} \cap O_{q2}$

Cela veut dire que, $O_x \subseteq O_{q1}$ et $O_x \subseteq O_{q2}$

Or, $O_{q1} \subseteq JR_{q1}$ et $O_{q2} \subseteq JR_{q2}$ donc $O_x \subseteq JR_{q1}$ et $O_x \subseteq JR_{q2}$

Ainsi, $O_x \subseteq (JR_{q1} \cap JR_{q2})$ (B.a)

Nous avons $ER_x = \bigcap_{oi \in Ox} R_{x,oi} = \bigcap_{oi \in Ox} [(R_{q1,oi} \cup R_{q2,oi}) \cap (JR_{q1} \cap JR_{q2})]$

$ER_x = (JR_{q1} \cap JR_{q2}) \bigcap [\bigcap_{oi \in Ox} (R_{q1,oi} \cup R_{q2,oi})]$

Or $(JR_{q1} \cap JR_{q2}) \bigcap [\bigcap_{oi \in Ox} (R_{q1,oi} \cup R_{q2,oi})] \subseteq (JR_{q1} \cap JR_{q2})$

Ainsi, nous avons $ER_x \subseteq (JR_{q1} \cap JR_{q2})$ (B.b)

De (B.a) et (B.b) on obtient $(O_x \cup ER_x) \subseteq (JR_{q1} \cap JR_{q2})$

Donc $JR_x \subseteq (JR_{q1} \cap JR_{q2})$ (B.c)

- D'autre part, $\forall o_i \in O_x$, $DR_{x,oi} = (DR_{q1,oi} \cup DR_{q2,oi}) \cap (EDR_{q1} \cap EDR_{q2})$

Nous avons $EDR_x = \bigcap_{oi \in Ox} DR_{x,oi} = \bigcap_{oi \in Ox} [(DR_{q1,oi} \cup DR_{q2,oi}) \cap (EDR_{q1} \cap EDR_{q2})]$

$EDR_x = (EDR_{q1} \cap EDR_{q2}) \bigcap [\bigcap_{oi \in Ox} (DR_{q1,oi} \cup DR_{q2,oi})]$

Or $(EDR_{q1} \cap EDR_{q2}) \bigcap [\bigcap_{oi \in Ox} (DR_{q1,oi} \cup DR_{q2,oi})] \subseteq (EDR_{q1} \cap EDR_{q2})$

Ainsi, nous avons, $EDR_x \subseteq (EDR_{q1} \cap EDR_{q2})$ (B.d)

ii. Cas : $O_{q1} \cap O_{q2} = \varnothing$

Nous pouvons démontrer de la même façon que le cas précédent en remplaçant :

- ✓ $(R_{q1,oi} \cup R_{q2,oi})$ par $R_{q3,oi}$
- ✓ $(DR_{q1,oi} \cup DR_{q2,oi})$ par $DR_{q3,oi}$
- ✓ Et $O_x \subseteq (JR_{q1} \cap JR_{q2})$ car $O_x = O_{q3} \cap (JR_{q1} \cap JR_{q2})$

(4) $JW_x \subseteq JW_{q3}$ et $EDW_x \subseteq EDW_{q3}$

i. Cas : $O_{q1} \cap O_{q2} \neq \varnothing$

- Nous avons $O_x \subseteq O_{q1}$ donc $O_x \subseteq JW_{q1}$

  Et comme $JW_{q1} \subseteq JW_{q3}$ (vérifié avant le flux d'information de q_1 vers q_3)

  Alors $O_x \subseteq JW_{q3}$ (B.a')

Nous avons $EW_x = \bigcap_{oi \in Ox} W_{x,oi} = \bigcap_{oi \in Ox} [(W_{q1,oi} \cup W_{q2,oi}) \cap JW_{q3}]$

$EW_x = JW_{q3} \cap [\bigcap_{oi \in Ox} (W_{q1,oi} \cup W_{q2,oi})]$

Par conséquent $EW_x \subseteq JW_{q3}$ (B.b')

De (B.a') et (B.b') nous obtenons $JW_x \subseteq JW_{q3}$ (B.c')

- Nous avons d'autre part :

$EDW_x = \bigcap_{oi \in Ox} DW_{x,oi} = \bigcap_{oi \in Ox} [(DW_{q1,oi} \cup DW_{q2,oi}) \cap EDW_{q3}]$

$EDW_x = EDW_{q3} \cap [\bigcap_{oi \in Ox} (DW_{q1,oi} \cup DW_{q2,oi})]$

Par conséquent $EDW_x \subseteq EDW_{q3}$ (B.d')

ii. Cas : $O_{q1} \cap O_{q2} = \varnothing$

Nous pouvons démontrer de la même façon que le cas précédent en remplaçant :

- ✓ $(W_{q1,oi} \cup W_{q2,oi})$ par $W_{q3,oi}$
- ✓ et $(DW_{q1,oi} \cup DW_{q2,oi})$ par $DW_{q3,oi}$
- ✓ $O_x \subseteq JW_{q3}$ car :

  $O_x = O_{q3} \cap (JR_{q1} \cap JR_{q2})$ donc $O_x \subseteq O_{q3}$

  et $O_{q3} \subseteq JW_{q3}$

Ainsi, d'après (B.c), (B.d), (B.c'), (B.d') et $(L(q_3) \leftarrow L(x))$ nous concluons que :

$q_1 \rightarrow q_3$ et $q_2 \rightarrow q_3$ sont sains.

C. La jointure de fusion $\_^f$

Cette jointure est plus flexible que la jointure restrictive car il garde plus de propriétaires dans le nouveau label.

La jointure de fusion qui calcule le label de l'objet contenant le résultat de l'opération est définie comme suit :

Soient q_1, q_2, $q_3 \in Q$, $L(q_1)$, $L(q_2)$, $L(q_3)$ les labels de q_1, q_2, q_3 respectivement et L(x) un nouveau label.

Supposons qu'il y a une opération de calcul sur q_1 et q_2 : $q_3 = q_1 \; op \; q_2$

Si $L(x) = L(q_1) \; \_^f \; L(q_2)$ alors :

- $O_x = (O_{q1} \cup O_{q2}) \cap (JR_{q1} \cap JR_{q2})$
- $\forall \; o_i \in O_x$ et $o_i \in O_{q1} \cap O_{q2}$
  - $R_{x,oi} = (R_{q1,oi} \cup R_{q2,oi}) \cap (JR_{q1} \cap JR_{q2})$
  - $W_{x,oi} = (W_{q1,oi} \cup W_{q2,oi}) \cap JW_{q3}$
  - $TO_{x,oi} = (TO_{q1,oi} \cup TO_{q2,oi}) \cap (EO_{q1} \cap EO_{q2})$
  - $DR_{x,oi} = (DR_{q1,oi} \cup DR_{q2,oi}) \cap (EDR_{q1} \cap EDR_{q2})$
  - $DW_{x,oi} = (DW_{q1,oi} \cup DW_{q2,oi}) \cap EDW_{q3}$
  - $P_{x,oi}$ = less restrictive $(P_{q1,oi}, P_{q2,oi})$
- $\forall \; K_i$ tel que $K_{io} \in (O_x \; / \; (O_{q1} \cap O_{q2}))$

  $L(x) \leftarrow L(x) \cup \{K_i\}$
- $L(q_3) \leftarrow L(x)$

L'algorithme 5.4 permet l'implémentation de la jointure de fusion.

Lemme 3 : Le flux d'information utilisant la jointure de fusion est sain.

Algorithme 5.4 : jointure de fusion

Entrées : $L(q_1)$, $L(q_2)$, $L(q_3)$ Sorties : $L(q_3) = L(q_1) \_^r L(q_2)$

Début

$L(x) \leftarrow \varnothing$

Pour chaque $K_j \in L(q_1)$

Si $K_{jo} \in (JR_{q1} \cap JR_{q2})$ alors

L(x).ajouterComposante(K_j)

FinSi

FinPour

Pour chaque $K_j \in L(q_2)$

Si $(K_{jo} \in (JR_{q1} \cap JR_{q2})) \wedge (K_{jo} \notin O_x)$ alors

L(x).ajouterComposante(K_j)

FinSi

FinPour

Pour chaque $K_j \in L(x)$

Si $K_{jo} \in (O_{q1} \cap O_{q2})$ alors

$K_j \leftarrow \{ K_{jo}, (TO_{q1,Kjo} \cup TO_{q2,Kjo}) \cap (EO_{q1} \cap EO_{q2}), (R_{q1,Kjo} \cup R_{q2,Kjo}) \cap (JR_{q1} \cap JR_{q2}),$

$(W_{q1,Kjo} \cup W_{q2,Kjo}) \cap JW_{q3}$, less restrictive $(P_{q1,Kjo}, P_{q2,Kjo}), (DR_{q1,Kjo} \cup DR_{q2,Kjo}) \cap (EDR_{q1} \cap EDR_{q2}),$

$(DW_{q1,Kjo} \cup DW_{q2,Kjo}) \cap EDW_{q3}\}$

FinSi

FinPour

$L(q_3) \leftarrow L(x)$

Fin

Preuve:

Comme pour la jointure restrictive, nous voulons montrer ce qui suit

$$\begin{cases} JR_x \subseteq (JR_{q1} \cap JR_{q2}) \text{ et } EDR_x \subseteq (EDR_{q1} \cap EDR_{q2}) & \textbf{(5)} \\ JW_x \subseteq JW_{q3} \text{ et } EDW_x \subseteq EDW_{q3} & \textbf{(6)} \end{cases}$$

(5) $JR_x \subseteq (JR_{q1} \cap JR_{q2})$ et $EDR_x \subseteq (EDR_{q1} \cap EDR_{q2})$

- Nous avons $(O_{q1} \cup O_{q2}) \cap (JR_{q1} \cap JR_{q2}) \subseteq (JR_{q1} \cap JR_{q2})$

Donc, $O_x \subseteq (JR_{q1} \cap JR_{q2})$ (C.a)

Nous avons $ER_x = \bigcap_{oi \in Ox} R_{x,oi} = [\bigcap_{oi \in (O1 \cap O2)} [(R_{q1,oi} \cup R_{q2,oi}) \cap (JR_{q1} \cap JR_{q2})]] \bigcap$

$$[\bigcap_{oi \in O1 \wedge oi \notin (Oq1 \cap Oq2)} R_{q1,oi}] \bigcap [\bigcap_{oi \in O2 \wedge oi \notin (Oq1 \cap Oq2)} R_{q2,oi}]$$

$ER_x = (JR_{q1} \cap JR_{q2}) \bigcap [[\bigcap_{oi \in (O1 \cap O2)} (R_{q1,oi} \cup R_{q2,oi})] \bigcap [\bigcap_{oi \in O1 \wedge oi \notin (Oq1 \cap Oq2)} R_{q1,oi}] \bigcap [\bigcap_{oi \in O2 \wedge oi \notin (Oq1 \cap Oq2)} R_{q2,oi}]]$

Donc, nous avons $ER_x \subseteq (JR_{q1} \cap JR_{q2})$ (C.b)

De (C.a) et (C.b), nous obtenons $JR_x \subseteq (JR_{q1} \cap JR_{q2})$ (C.c)

- D'autre part, Nous avons

$EDR_x = \bigcap_{oi \in Ox} DR_{x,oi} = [\bigcap_{oi \in (O1 \cap O2)} [(DR_{q1,oi} \cup DR_{q2,oi}) \cap (EDR_{q1} \cap EDR_{q2})]] \bigcap [\bigcap_{oi \in O1 \wedge oi \notin (Oq1 \cap Oq2)} DR_{q1,oi}] \bigcap [\bigcap_{oi \in O2 \wedge oi \notin (Oq1 \cap Oq2)} DR_{q2,oi}]$

$EDR_x = (EDR_{q1} \cap EDR_{q2}) \bigcap [[\bigcap_{oi \in (O1 \cap O2)} (DR_{q1,oi} \cup DR_{q2,oi})] \bigcap [\bigcap_{oi \in O1 \wedge oi \notin (Oq1 \cap Oq2)} DR_{q1,oi}] \bigcap [\bigcap_{oi \in O2 \wedge oi \notin (Oq1 \cap Oq2)} DR_{q2,oi}]]$

Ainsi, nous avons $EDR_x \subseteq (EDR_{q1} \cap EDR_{q2})$ (C.d)

(6) $JW_x \subseteq JW_{q3}$ et $EDW_x \subseteq EDW_{q3}$

- Nous avons $JW_{q1} \subseteq JW_{q3}$ et $JW_{q2} \subseteq JW_{q3}$ (vérifiés avant le flux d'information)

  Donc, $O_{q1} \subseteq JW_{q3}$ et $O_{q2} \subseteq JW_{q3}$

  Ainsi, $O_{q1} \cup O_{q2} \subseteq JW_{q3}$

  Or, $(O_{q1} \cup O_{q2}) \cap (JR_{q1} \cap JR_{q2}) \subseteq JW_{q3}$

  D'où $O_x \subseteq JW_{q3}$ (C.a')

  Nous avons aussi :

  $EW_x = \bigcap_{oi \in Ox} W_{x,oi} = [\bigcap_{oi \in (O1 \cap O2)} [(W_{q1,oi} \cup W_{q2,oi}) \cap JW_{q3}] \bigcap [\bigcap_{oi \in O1 \wedge oi \notin (Oq1 \cap Oq2)} W_{q1,oi}] \bigcap [\bigcap_{oi \in O2 \wedge oi \notin (Oq1 \cap Oq2)} W_{q2,oi}]$

  $EW_x = JW_{q3} \bigcap [[\bigcap_{oi \in (O1 \cap O2)} (W_{q1,oi} \cup W_{q2,oi})] \bigcap [\bigcap_{oi \in O1 \wedge oi \notin (Oq1 \cap Oq2)} W_{q1,oi}] \bigcap [\bigcap_{oi \in O2 \wedge oi \notin (Oq1 \cap Oq2)} W_{q2,oi}]]$

Donc, nous avons $EW_x \subseteq JW_{q3}$ (C.b')

De (C.a') et (C.b'), nous obtenons $JW_x \subseteq JW_{q3}$ (C.c')

- Nous avons :

  $EDW_x = \bigcap_{oi \in Ox} DW_{x,oi} = [\bigcap_{oi \in (O1 \cap O2)} [(DW_{q1,oi} \cup DW_{q2,oi}) \cap EDW_{q3}] \bigcap [\bigcap_{oi \in O1 \wedge oi \notin (Oq1 \cap Oq2)} DW_{q1,oi}] \bigcap [\bigcap_{oi \in O2 \wedge oi \notin (Oq1 \cap Oq2)} DW_{q2,oi}]$

  $EDW_x = EDW_{q3} \bigcap [[\bigcap_{oi \in (O1 \cap O2)} (DW_{q1,oi} \cup DW_{q2,oi})] \bigcap [\bigcap_{oi \in O1 \wedge oi \notin (Oq1 \cap Oq2)} DW_{q1,oi}] \bigcap [\bigcap_{oi \in O2 \wedge oi \notin (Oq1 \cap Oq2)} DW_{q2,oi}]]$

Donc, $EDW_x \subseteq EDW_{q3}$ (C.d')

Ainsi, d'après (C.c), (C.d), (C.c'), (C.d') et $(L(q_3) \leftarrow L(x))$ nous concluons que :

$q_1 \rightarrow q_3$ et $q_2 \rightarrow q_3$ sont sains.

D. La jointure de propagation $\_^p$

Les propriétaires des objets pères sont aussi des propriétaires des objets fils. Par conséquent, les labels des objets pères peuvent être propagés vers les labels des objets fils si les politiques de propagation le permettent. Chaque propriétaire de l'objet père décide s'il propage son label ou non et chaque propriétaire de l'objet fils décide s'il accepte cette propagation.

La propagation des politiques consiste à appliquer les politiques du propriétaire de l'objet père s'il est aussi le propriétaire du fils, ou simplement insérer ce propriétaire et sa politique dans le label du fils.

Nous définissons la jointure de propagation comme suit :

Soient : une jointure de propagation $L(x) \leftarrow L(q_1) \_^p L(q_2)$.

$q_1, q_2 \in Q$, tels que $q_1 \rightarrow_{ho} q_2$ (q_1 objet père et q_2 objet fils).

$L(q_1)$, $L(q_2)$ les labels de q_1, q_2 respectivement et $L(x)$ un nouveau label de q_2 après la propagation.

- $O_x = O_{q1} \cup O_{q2}$ et $L(x) = L(q_1) \cup L(q_2)$
- $\forall\ o_i \in O_{q1} \cap O_{q2}$
  1. Si $P_{q2,oi}.p_{accept}$ = faible $\wedge$ $P_{q1,oi}.p_{propagate}$ = forte alors
     - $R_{x,oi} = (R_{q1,oi} \cup R_{q2,oi}) \cap (ER_{q1} \cup ER_{q2})$
     - $W_{x,oi} = (W_{q1,oi} \cup W_{q2,oi}) \cap (EW_{q1} \cup EW_{q2})$
     - $TO_{x,oi} = (TO_{q1,oi} \cup TO_{q2,oi}) \cap (EO_{q1} \cup EO_{q2})$
     - $DR_{x,oi} = (DR_{q1,oi} \cup DR_{q2,oi}) \cap (EDR_{q1} \cup EDR_{q2})$
     - $DW_{x,oi} = (DW_{q1,oi} \cup DW_{q2,oi}) \cap (EDW_{q1} \cup EDW_{q2})$
     - $P_{x,oi} = P_{q2,oi}$
  2. Sinon
     - $R_{x,oi} = R_{q2,oi}$
     - $W_{x,oi} = W_{q2,oi}$
     - $TO_{x,oi} = TO_{q2,oi}$
     - $DR_{x,oi} = DR_{q2,oi}$
     - $DW_{x,oi} = DW_{q2,oi}$
     - $P_{x,oi} = P_{q2,oi}$
- $L(q_2) \leftarrow L(x)$

L'algorithme 5.5 donne les étapes qui permettent l'implémentation de la jointure de propagation

Algorithme 5.5 : jointure de propagation

Entrées : $L(q_1)$, $L(q_2)$ Sorties : $L(q_2) = L(q_1) \_^p L(q_2)$

Début

$L(x) \leftarrow \varnothing$

Pour chaque $K_j \in L(q_1)$

L(x).ajouterComposante(K_j)

FinPour

Pour chaque $K_j \in L(q_2)$

Si $K_j \notin L(x)$ alors % i.e. $K_{jo} \notin O_x$

L(x).ajouterComposante(K_j)

FinSi

FinPour

Pour chaque $K_j \in L(x)$

Si $K_{jo} \in (O_{q1} \cap O_{q2})$ alors

Si $P_{q2,oi}.p_{accept}$ = faible $\wedge$ $P_{q1,oi}.p_{propagate}$ = forte alors

$K_j \leftarrow \{ K_{jo}, (TO_{q1,oi} \cup TO_{q2,oi}) \cap (EO_{q1} \cup EO_{q2}), (R_{q1,oi} \cup R_{q2,oi}) \cap (ER_{q1} \cup ER_{q2}),$

$(W_{q1,oi} \cup W_{q2,oi}) \cap (EW_{q1} \cup EW_{q2}), P_{q2,oi}, (DR_{q1,oi} \cup DR_{q2,oi}) \cap (EDR_{q1} \cup EDR_{q2}),$

$(DW_{q1,oi} \cup DW_{q2,oi}) \cap (EDW_{q1} \cup EDW_{q2})\}$

Sinon

$K_j \leftarrow \{ K_{jo}, TO_{q2,oi}, R_{q2,oi}, W_{q2,oi}, P_{q2,oi}, DR_{q2,oi}, DW_{q2,oi} \}$

FinSi

FinSi

FinPour

$L(q_2) \leftarrow L(x)$

Fin

Lemme 4 : La propagation des labels est saine.

Preuve:

Nous voulons montrer ce qui suit :

$$\begin{cases} ER_x \subseteq (ER_{q1} \cup ER_{q2}) \text{ et } EDR_x \subseteq (EDR_{q1} \cup EDR_{q2}) & \textbf{(7)} \\ EW_x \subseteq (EW_{q1} \cup EW_{q2}) \text{ et } EDW_x \subseteq (EDW_{q1} \cup EDW_{q2}) & \textbf{(8)} \end{cases}$$

(7) $ER_x \subseteq (ER_{q1} \cup ER_{q2})$ et $EDR_x \subseteq (EDR_{q1} \cup EDR_{q2})$

Cas 1: Si $P_{q2,oi}.p_{accept}$ = faible $\wedge$ $P_{q1,oi}.p_{propagate}$ = forte

- Nous avons

$ER_x = \bigcap_{oi \in Ox} R_{x,oi} = [\bigcap_{oi \in (O1 \cap O2)} [(R_{q1,oi} \cup R_{q2,oi}) \cap (ER_{q1} \cup ER_{q2})]] \bigcap$

$[\bigcap_{oi \in O1/(O1 \cap O2)} R_{q1,oi}] \bigcap [\bigcap_{oi \in O2/(O1 \cap O2)} R_{q2,oi}]$

$ER_x = (ER_{q1} \cup ER_{q2}) \bigcap [[\bigcap_{oi \in (O1 \cap O2)} (R_{q1,oi} \cup R_{q2,oi})] \bigcap [\bigcap_{oi \in O1/(O1 \cap O2)} R_{q1,oi}] \bigcap$

$[\bigcap_{oi \in O2/(O1 \cap O2)} R_{q2,oi}]]$

Ainsi, nous avons $ER_x \subseteq (ER_{q1} \cup ER_{q2})$ (D.a)

- D'autre part,

$EDR_x = \bigcap_{oi \in Ox} DR_{x,oi} = [\bigcap_{oi \in (O1 \cap O2)} [(DR_{q1,oi} \cup DR_{q2,oi}) \cap (EDR_{q1} \cup EDR_{q2})]] \bigcap$

$[\bigcap_{oi \in O1/(O1 \cap O2)} DR_{q1,oi}] \bigcap [\bigcap_{oi \in O2/(O1 \cap O2)} DR_{q2,oi}]$

$EDR_x = (EDR_{q1} \cup EDR_{q2}) \bigcap [[\bigcap_{oi \in (O1 \cap O2)} (DR_{q1,oi} \cup DR_{q2,oi})] \bigcap [\bigcap_{oi \in (Ox \setminus O2)} DR_{q1,oi}]$

$\bigcap [\bigcap_{oi \in (Ox \setminus O1)} DR_{q2,oi}]]$

Ainsi, nous avons $EDR_x \subseteq (EDR_{q1} \cup EDR_{q2})$ (D.b)

Cas 2: $P_{q2,oi}.p_{accept} = \text{forte} \vee P_{q1,oi}.p_{propagate} = \text{faible}$

- Nous avons $ER_x = \bigcap_{oi \in Ox} R_{x,oi} = [\bigcap_{oi \in (Ox \setminus O2)} R_{q1,oi}] \bigcap [\bigcap_{oi \in O2} R_{q2,oi}]$

$ER_x = [\bigcap_{oi \in (Ox \setminus O2)} R_{q1,oi}] \bigcap ER_{q2}$

Or $ER_x \subseteq ER_{q2}$

Donc, nous avons $ER_x \subseteq (ER_{q1} \cup ER_{q2})$ (D.c)

- D'autre part,

Nous avons $EDR_x = \bigcap_{oi \in Ox} DR_{x,oi} = [\bigcap_{oi \in (Ox \setminus O2)} DR_{q1,oi}] \bigcap [\bigcap_{oi \in O2} DR_{q2,oi}]$

$EDR_x = [\bigcap_{oi \in (Ox \setminus O2)} DR_{q1,oi}] \bigcap EDR_{q2}$

Or $EDR_x \subseteq EDR_{q2}$ et $EDR_{q2} \subseteq (EDR_{q1} \cup EDR_{q2})$

Alors $EDR_x \subseteq (EDR_{q1} \cup EDR_{q2})$ (D.d)

D'après (D.a), (D.b), (D.c) et (D.d), (7) est vérifié.

(8) $EW_x \subseteq (EW_{q1} \cup EW_{q2})$ et $EDW_x \subseteq (EDW_{q1} \cup EDW_{q2})$

Nous pouvons démontrer de la même façon que (7) en remplaçant :

- ✓ $R_{q1,oi}$ par : $W_{q1,oi}$
- ✓ $R_{q2,oi}$ par : $W_{q2,oi}$
- ✓ $DR_{q1,oi}$ par : $DW_{q1,oi}$
- ✓ $DR_{q2,oi}$ par : $DW_{q2,oi}$
- ✓ ER_{q1} par : EW_{q1}
- ✓ ER_{q2} par : EW_{q2}
- ✓ EDR_{q1} par : EDW_{q1}
- ✓ EDR_{q2} par : EDW_{q2}

D'après (7) et (8), nous concluons que : $L(q_2) \leftarrow L(q_1) \_^p L(q_2)$ est saine.

E. La jointure de dé-classification

Pendant l'exécution d'une transaction, si une méthode ne peut pas être exécutée à cause du mécanisme de contrôle de flux d'information toute la transaction sera annulée.
Pour éviter ce genre de situation, des règles strictes de dé-classification sont apparues. L'idée de dé-classification est d'ajouter un principal (rôle) intermédiaire de la transaction à l'ensemble de lecteurs effectifs en cas de lecture (ou à l'ensemble de rédacteurs effectifs en cas d'écriture) de l'objet demandé si l'initiateur de la transaction est déjà un lecteur effectif (ou rédacteur effectif) de cet objet.
La dé-classification est un consensus des propriétaires effectifs. Et elle est effectuée en utilisant l'ensemble de dé-classification effectif.

Si une dé-classification en cas de lecture est apparue, une information doit être ajoutée à l'objet recevant l'information déclassifiée pour empêcher sa sauvegarde ou son utilisation ultérieure. Cela peut être réalisé en mettant le label de l'objet recevant l'information déclassifié à vide ($\varnothing$). Après l'opération de dé-classification, le principal intermédiaire ajouté à l'ensemble de lecteurs effectifs doit être supprimé.

Si une dé-classification en cas d'écriture est apparue, le label de l'objet écrit ne doit pas être altéré par celui de la valeur écrite. Après l'opération de dé-classification, le principal intermédiaire ajouté à l'ensemble de rédacteurs effectifs doit être supprimé.

Dans ce qui suit, nous donnons les algorithmes permettant l'opération de dé-classification en cas de lecture et en cas d'écriture.

Algorithme 5.6 : jointure de dé-classification en cas de lecture

```
Entrées : L(q) % le label de l'objet à déclassifier
          S1 = init(T) ∈ JRq  % le principal ou rôle au profit duquel la dé-classification sera effectuée
          S2 = last(T) ∉ JRq  % le principal ou rôle intermédiaire pour lequel l'objet demandé sera déclassifié
Sorties : Ld  % Ld label de l'objet recevant l'information déclassifiée
          {Oui, Non}
Début
   Pour chaque δ ∈ EDRq
      Si (S1 = δinit et S2 ∈ δiinter) alors
        Ld ← ∅
        Retourner (Ld, Oui)
      FinSi
   FinPour
   Retourner (Non)
Fin
```

Algorithme 5.7 : jointure de dé-classification en cas d'écriture

```
Entrées : L(q) % le label de l'objet à déclassifier
          S1 = init(T) ∈ JWq  % le principal ou rôle au profit duquel la dé-classification sera effectuée
          S2 = last(T) ∉ JWq  % le principal ou rôle intermédiaire pour lequel l'objet demandé sera déclassifié
Sorties : {Oui, Non}
Début
   Pour chaque δ ∈ EDWq
      Si (S1 = δinit et S2 ∈ δiinter) alors
        Retourner (Oui)
      FinSi
   FinPour
   Retourner (Non)
Fin
```

Lemme 5 : la dé-classification d'objets est saine.

Preuve :

Pour prouver que la dé-classification est saine, il faut montrer ce qui suit :

1. *La dé-classification est un résultat de consensus des propriétaires effectifs :*
   Selon la définition de EDR_q et EDW_q, seuls les propriétaires effectifs (les propriétaires auxquels tous les propriétaires font confiance) de l'objet q ont la possibilité de déclassifier cet objet aux rôles qui n'appartiennent pas à l'ensemble des lecteurs ou rédacteurs communs.
   Alors la dé-classification est bien contrôlée.
2. *l'information déclassifiée en cas de lecture ne sera pas déclassifiée ultérieurement :*
   Puisque le label de l'objet portant l'information déclassifiée est vide, alors l'ensemble de dé-classification effectif est aussi vide. Ainsi, l'objet déclassifié ne sera pas déclassifié une autre fois.
3. *les propriétés de l'objet déclassifié en cas d'écriture ne seront pas altérées :*
   Puisque le label de l'objet déclassifié reste toujours le même, alors ses propriétés restent inchangées.
4. *l'information déclassifiée est utilisée uniquement pour répondre à la requête de l'initiateur de la transaction et ne sera pas sauvegardée:*
   Cela est réalisé en supposons que les modules de MHS-SW dans les différents fournisseurs de services partagent une relation de confiance entre eux.

5.2.2.5 Les fonctions de CFI

Le CFI fournit des fonctions utiles pour les procédures de calcul. Nous citons ces fonctions telles qu'elles sont définies dans [TAR06].
La structure *Wrap* introduite contient un objet et son label. Le tout encrypté par une clé connue uniquement par le module MHS-SW.

Les fonctions de calcul sur les objets sont définies dans ce qui suit :

- Wrap Assign (Wrap to, Wrap from): un flux simple par l'assignation d'un objet à un autre qui respecte la règle 10 en utilisant la jointure d'assignation.
- Wrap StrictExpr (Wrap to, String expr, Wrap[]): assigne l'information en dérivant une expression avec les objets enveloppés associés en utilisant la jointure restrictive.
- Wrap FlexiExpr (Wrap to, String expr, Wrap[]): assigne l'information en dérivant une expression avec les objets enveloppés associés en utilisant la jointure de fusion.
- Wrap Wrapper (Object source): enveloppe une information qui n'a pas de label (associe un label à une information).

Le CFI fournit aussi des fonctions utilisées pour accéder à des objets ou pour modifier directement des labels.
L'objet Auth est passé via des méthodes et il contient les informations d'authentification du sujet (rôle dans notre cas). OPath utilise le langage XPath pour référencer les objets. Subject représente un sujet dans le système.

- Wrap Read (OPath object, Auth subject): requête de lecture d'un objet (suit la règle 1).
- boolean Write (OPath object, Auth subject, Wrap newobject): requête d'accès à un objet en écriture (suit la règle 2).
- boolean AddReader (OPath object, Auth subject, Subject reader): ajoute un lecteur à un label (suit la règle 5)
- boolean RemoveReader (OPath object, Auth subject, Subject reader): supprime un lecteur d'un label (suit la règle 5).
- boolean AddOwner (OPath object, Auth subject, Subject owner): ajoute un nouveau propriétaire à un label (suit la règle 8).
- boolean RemoveOwner(OPath object, Auth subject, Subject owner): supprime un propriétaire d'un label (suit la règle 8).
- boolean TrustOwner (OPath object, Auth subject, Subject owner): ajoute un propriétaire à l'ensemble des propriétaires de confiance (suit la règle 7).

D'autres fonctions utilisant d'autres règles de flux d'information définies par le modèle peuvent être rajoutées pour compléter l'ensemble des fonctions. Des exemples de ces fonctions sont :

- boolean Create (OPath object1, OPath object2, Auth subject, Wrap newobject): requête de création de l'objet object2 comme fils de objest1 (suit la règle 3).
- boolean Declassify (OPath object, Auth subject, Subject reader): ajoute un lecteur à l'ensemble des lecteurs effectifs (suit la règle de dé-classification des politiques de confidentialité - règle 11).
- boolean DeclassifyInt (OPath object, Auth subject, Subject writer): ajoute un rédacteur à l'ensemble des rédacteurs effectifs (suit la règle de dé-classification de politiques d'intégrité - règle 11).

Quand les règles appliquées par les fonctions définies ci-dessus ne sont pas respectées, les fonctions retournent les valeurs **Null** pour le type *Wrap* et **faux** pour le type *boolean*.

Les fonctions suivantes sont appelées par les fonctions de calcul pour mettre à jour les labels des objets obtenus.

- Label AssigningJoin (Label source, Label destination): jointure d'assignation de deux labels.
- Label RestrictiveJoin (Label l1, Label l2): jointure restrictive de deux labels.
- Label FusingJoin (Label l1, Label l2): jointure de fusion de deux labels.
- Label PropagationJoin (Label l1, Label l2): jointure de propagation de deux labels.
- Label DeclassReadJoin (Label l1, Label l2): jointure de dé-classification en cas de lecture de deux labels.
- Label DeclassWriteJoin (Label l1, Label l2): jointure de dé-classification en cas d'écriture de deux labels.

5.3 Conclusion

Le modèle MHS-SW permet l'authentification du demandeur de la requête, l'activation du rôle correspondant à la méthode demandée si la requête d'accès est acceptée et l'application des règles de flux d'information à chaque tentative d'accès aux objets du système ou à leurs labels.

A travers ce chapitre, nous avons défini formellement notre modèle MHS-SW pour la sécurité des services Web. Nous avons modélisé le contrôle d'accès et donné l'algorithme correspondant en précisant les modifications apportées par rapport au modèle adapté. Nous

avons également développé le modèle de contrôle de flux d'information en définissant ses concepts de base et ses règles que nous avons utilisées pour l'assignation de permissions aux rôles. Nous avons donné de nouvelles définitions pour les jointures de dérivation de labels avec les algorithmes correspondants et les démonstrations de sûreté de ces jointures. Des fonctions permettant l'application des règles et les jointures de flux d'information sont citées pour avoir une vision sur l'implémentation du modèle.

Conclusion et perspectives

Compte tenu des nombreux aspects liés à la sécurité, à l'authentification et aux autorisations, à la confidentialité et à l'intégrité des données, et compte tenu du fait que les solutions de sécurité ne sont pas intégrées au sein de l'architecture des services Web, la sécurisation des services Web reste un enjeu important qui motive beaucoup de chercheurs pour développer diverses solutions et approches de sécurité.

Notre travail se concentre sur deux aspects de sécurité à savoir le contrôle d'accès et le contrôle de flux d'information dans les systèmes de services Web. Notre modèle présenté dans ce mémoire est développé en se basant sur le modèle proposé par Z. Tari et al pour le contrôle de flux d'information dans les services Web en utilisant la vérification dynamique de labels. Le modèle de Z. Tari et al. a été amélioré par N. Berrehouma en introduisant les notions de propagation et de dé-classification pour augmenter la flexibilité du modèle.

Notre contribution à ce modèle consiste à vérifier et à compléter la version du modèle de contrôle de flux d'information présenté par N. Berrehouma, de l'étendre pour assurer l'intégrité des données en plus de la confidentialité, et d'intégrer un modèle de contrôle d'accès aux services Web et aux objets d'un système. Ainsi, notre solution fournit un modèle hybride pour la sécurité de bout en bout des services Web (MHS-SW) en intégrant un modèle de contrôle d'accès avec un modèle de contrôle de flux d'information.

Le modèle de contrôle d'accès est basé sur le modèle RBAC (Role Based Acces Control) et l'utilisation des attributs des demandeurs de services. En effet, RBAC introduit le concept du rôle qui simplifie la gestion et l'application des règles et la notion d'attributs facilite la gestion de confiance dans les environnements ouverts tels que les services Web. La nouvelle version du modèle de contrôle de flux d'information que nous avons proposée dans ce mémoire permet de contrôler les flux d'information en assurant, en plus de la confidentialité, l'intégrité des informations échangées entre les objets manipulés par les servies Web. Le contrôle d'intégrité est assuré en proposant une extension aux labels des objets pour contenir l'ensemble des rédacteurs autorisés à écrire ces objets et l'ensemble des rôles auxquels les règles d'intégrité peuvent être déclassifiées. En outre, nous avons donné de nouvelles définitions pour les jointures de dérivation de labels pour pallier les lacunes des jointures définies dans le modèle originel avec les démonstrations de leur sûreté.

Dans les travaux ultérieurs, nous comptons ajouter un mécanisme de négociation au modèle de contrôle d'accès. Ce mécanisme permettra de récupérer les attributs de l'utilisateur qui manquent pour l'application des règles d'accès.

Un autre mécanisme est nécessaire pour l'établissement des relations de confiance entre les fournisseurs et les demandeurs de services quand ils sont étrangers entre eux.

Nous voulons dans le futur exprimer les règles et les politiques d'accès et de flux d'information sous forme de prédicats logique pour augmenter la flexibilité et l'expressivité du modèle.

Les politiques de contrôle d'accès et les règles de contrôle de flux d'information peuvent être améliorées pour fournir une approche de contrôle d'accès et de contrôle de flux d'information plus flexible et plus complète.

Nous avons commencé l'implémentation d'une plateforme pour tester la fiabilité et l'efficacité de notre modèle en l'appliquant à un système des services Web. Une architecture distribuée de cette plateforme a été décrite dans ce mémoire.

Bibliographie

[APA07] Apache Axis Project Home Page.
2007. Disponible sur: http://ws.apache.org/axis

[BAR96] A. Baraani, J. Pieprzyk, and R. Safavi-Naini. Security in databases: A survey study, 1996.

[BEL03] A. Belokosztolszki, D.M. Eyers, K. Moody. Policy contexts: Controlling information flow in parameterized RBAC.
IEEE 4th International Workshop on Policies for Distributed Systems and Networks, June 2003.

[BEL75] D.E. Bell and L.J. LaPadula. Secure Computer Systems: Mathematical Foundations and Model. *Mitre Corp. Report No. M74-244*, Bedford, MA, 1975

[BER01] E. Bertino, M. Braun, S. Castano, E. Ferrari, and M. Mesiti. Author-X: A Java-Based System for XML Data Protection.
In *Proceedings of the IFIP TC11/ WG11.3 Fourteenth Annual Working Conference on Database Security*, pages 15-26, Deventer, The Netherlands, Kluwer, B.V. 2001.

[BER02] E. Bertino, E. Ferrari. Secure and selective dissemination of XML documents.
ACM Transactions on Information and System Security (TISSEC), 5(3), 290-331, August 2002.

[BER04] E. Bertino, A.C. Squicciarini, L. Martino, F. Paci. A flexible access control model for web services. *CERIAS and CS Department, Purdue University DICO, University of Milano.* 2004.
Available at :*dimacs.rutgers.edu/Workshops/Commerce/slides/bertino.ppt*

[BER06] N. Berrehouma. Flexible information flow control for web services.
Master thesis, Béjaià University, October 2006.

[BER97] E. Bertino, P. Samarati, A. Ciampichetti, and S. Jajodia. Information flow control in object-oriented systems. *IEEE Transactions on Knowledge and Data Engineering*, 09(4):524-538, 1997.

[BER98] E. Bertino, S. C.VIMERCATI, E. Ferrari and P. Samarati. Exception-Based Information Flow Control In Object-Oriented Systems.
ACM Transactions on Information and System Security, Vol.1, No.1, Pages 26–65. November 1998.

[BHA+03] R. Bhatti, J. B. D. Joshi, E. Bertino, A. Ghafoor, Access Control in Dynamic XML-based Web-Services with XRBAC,
In proceedings of The First International Conference on Web Services, Las Vegas, June 23-26, 2003.

[BHA03] R. Bhatti, X-GTRBAC: An XML-based Policy Specification Framework and Architecture for Enterprise-Wide Access Control. *Masters thesis, Purdue University,* May 2003. *Available as CERIAS technical report 2003-27.*

[BHA04] R. Bhatti, E. Bertino and A. Ghafoor. A Trust-based Context-Aware Access Control Model for Web-Services. *In Proceedings of the IEEE International Conference on Web Services (ICWS'04),* June 2004.

[BLA96] M. Blaze, J. Feigenbaum and J. Lacy. Decentralized trust management. *Proceedings of IEEE Symposium on Security and Privacy*. 1996.

[BOE85] W.E Boebert and C.T Ferguson. A partial solution to the discretionary Trojan horse problem. In *Proceedings of Eighth Nat'l Computer Security Conf.*, pages 141-144, Gaithersburg, Md, 1985.

[BRY95] C. Bryce, J-P. Baniitre, D. Le Mktayer. An Approach to Information Security in Distributed Systems. Rennes University, France, pages 384-394. 1995.

[BUR01] S. Burns. Web Services Security – An Overview.
2001. Available on the Internet (031202):
http://www.sans.org/rr/papers/index.php

[CHA02] JM. Chauvet. Services WEB avec SOAP, WSDL, UDDI, ebXML. Eyrolles, 2002.

[CHA06] S. Chakraborty, I. Ray. TrustBAC: integrating trust relationships into the RBAC model for access control in open systems.
Proceedings of the eleventh ACM symposium on Access control models and technologies. Lake Tahoe, California, USA. 2006.
www.cs.colostate.edu/~indrajit/Security/trust/trustbac-sacmat06.pdf

[CUR01] F. Curbera, W.A. Nagy, and S. Weerawarana. Web services : Why and how?
oopsla 2001 workshop on object-oriented web services. 2001.

[EAS00] D. Eastlake, J. Reagle, and D. Solo, XML-Signature Syntax and Processing,
W3C Recommendation xmldsig-core, October 2000.

[FER97] E. Ferrari, P. Samarati, E. Bertino, S. Jajodia. Providing flexibility in information flow control for object-oriented systems,
IEEE Symposium on Security and Privacy, May 1997.

[GOD02] S. Godik, and T. Moses, OASIS eXtensible Access Control Markup Language (XACML). OASIS *Committee Specification cs-xacml-specification-1.0*, November 2002

[GRA00] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys.* Fourth Quarter 2000. *Available at :http://www.comsoc.org/pubs/surveys*.

[GRA72] G. S. Graham and P. J. Denning. Protection - principles and practice. *AFIPS:Conference Proceedings Volume 40 Spring Joint Computer Conference*, 40:417-429, 1972.

[GRA91] J. W. Gray III. Toward a mathematical foundation for information flow security. *sp*, 00:21, 1991.

[HAI06] S. Hai-bo and H. Fan. An Attribute-Based Access Control Model for Web Services. *In Proceedings of the Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06).* December 2006.*http://csdl.computer.org/dl/proceedings/pdcat/2006/2736/00/27360074.pdf*

[HAL02] P. Hallam-Baker and E. Maler. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML). *OASIS Committee Specification sstc-core,* May 2002.

[HAR76] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Commun. ACM*, 19(8):461-471, 1976.

[HER00] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor and Y. Ravid, Access control meets public key infrastructure, or assigning roles to strangers. *Proceedings of IEEE Symposium on Security and Privacy*. 2000.

[HON02] M. Hondo D. Melgar and A. Nadalin. Web services security: Moving up the stack. December 2002. Available at : http://www-128.ibm.com/developerworks/webservices/library/ws-secroad/

[HOU99] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. *RFC 2459 (Proposed Standard), Obsoleted by RFC 3280*. January 1999.

[IBM02] IBM & Microsoft. Security in a Web Services World: A Proposed Architecture and Roadmap. Version 1.0. April 7, 2002. Available on the Internet (031207): *http://www-106.ibm.com/developerworks/webservices/library/ws-secmap/*

[ITT88] International Telephone and Telegraph Consultative Committee. Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, CCITT Recommendation X.509, November 1988.

[KAC06] L. Kachna. Infrastructures pour la création de dépôts de protocoles de services web. *Mémoire de magistère. Université de Béjaià*. 2006.

[KEL03] P. Kellert and F. Toumani. Les web services sémantiques.
in web sémantique, action spécifique 32 cnrs/stic. October 2003.

[KOH93] J. Kohl and B. Neuman. The Kerberos Network Authentication Service (V5).
RFC 1510, September 1993.

[LAM73] B. W. Lampson. A note on the confinement problem.
Communications of the ACM, 16(10):613-615, 1973.

[LAM74] B. W. Lampson. Protection. *SIGOPS Oper. Syst. Rev.*, 8(1):18-24, 1974.

[LI 02] B. Li. Analyzing information-flow in java program based on slicing technique.
ACM SIGSOFT Software Engineering Notes, 27(5), 98-103, September 2002.

[LIN99] T. Lindholm and F. Yellin. The java virtual machine specification.
Sun Microsys-tems Inc., 1999.

[LIU05] M. Liu, H. Guo and J. SU. An attribute and role based access control model for web services. *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21* August 2005.

[MCC90] C.J. McCollumn, J.R. Messing and L. Notargiacomo. Beyond the pale of mac and dac - defining new forms of access control. In *Proceedings of the 1990 IEEE Computer Society Symposium on Security and Privacy*, Oakland, CA, 1990.

[MCL90] J. McLean. Security models and information flow.
In IEEE Symposium on Security and Privacy, pages 180-189, 1990.

[MEL04] T. Melliti. Interopérabilité des services Web complexes. Application aux systèmes multi-agents. *PhD thesis, Paris IX Dauphine*, 2004.

[MIC04] Microsoft. Module 10 – Sécurité des services Web.
Dernière mise à jour août 2004. disponible sur :
http://www.microsoft.com/france/msdn/securite/secmod10.mspx

[MYE00] A.C. Myers, B. Liskov. Protecting privacy using the decentralized label model.
ACM Transactions on Software Engineering and Methodology, 9(4), 410-442, October 2000.

[MYE97] A.C. Myers, B. Liskov. A decentralized model for information flow control.
In *Proc. of the 16th ACM Symposium on Operating Systems Principles*, Saint-Malo France, October 1997.

[MYE98] A.C. Myers and B. Liskov. Complete, safe information flow with decentralized labels. *sp*, 00, 1998.

[MYE99] A.C. Myers. JFlow: Practical mostly-static information flow control. In *Proc. of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 1999.

[OAS07] OASIS. Online community for the Universal Description, Discovery, and Integration OASIS Standard. http://uddi.xml.org/. 2007.

[OSB02] S.L. Osborn. Information flow analysis of an RBAC system. In *Proc. of the 17th ACM Symposium on Access Control Models and Techniques*, June 2002

[PAR04] J. Park and R. Sandhu. The $UCON_{ABC}$ Usage Control Model. *ACM Transactions on Information and System Security, Vol. 7, No. 1, Pages 128–174*. February 2004.

[PLA04] C. Platzer. Trust-based Security in Web Services. *Master's thesis*. Technical University of *Vienna*, May 2nd 2004. Available at: www.infosys.tuwien.ac.at/Staff/sd/DA/ChristianPlatzer.pdf

[RAY04] I. Ray and S. Chakraborty. A Vector Model of Trust for Developing Trustworthy Systems. *In Proceedings of the 9th European Symposium of Research in Computer Security (ESORICS 2004), volume 3193 of Lecture Notes in Computer Science, pages 260–275, Sophia Antipolis, France,* September 2004.

[SAN+96] R. S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. IEEE Computer, 29(2):38-47, February 1996.

[SAN94] R. S. Sandhu and P. Samarati. Access control: Principles and practice. *IEEE Communications Magazine*, 32(9):40-48, Septembre 1994.

[SAN96] R. Sandhu, Role Hierarchies and Constraints for Lattice-Based Access Controls *Proc. Fourth European Symposium on Research in Computer Security, Rome, Italy*, September 25-27, 1996.

[SOF07] Softeam. Technologies-Web Services. *http://www.softeam.fr/technologies_web_services.php*. April 2007.

[STO81] A. Stoughton. Access flow: A protection model which integrates access control and information flow. In *Proceedings of the 1981 IEEE Computer Society Symposium on Security and Privacy*, pages 9-18, Oakland, CA, 1981.

[SUN07] Sun Microsystems Inc. Java security architecture. Avril 2007. Disponible sur *http://java.sun.com/j2se/1.3/docs/guide/security/spec/security-pec.doc12.html*.

[TAR04] Z. Tari and R. Wonohoesodo. A role based access control for web services. Proceedings of the 2004 IEEE International Conference on Services Computing (SCC'04).2004.

[TAR06] Z. Tari, P. Bertok, and D. Simic. A dynamic label checking approach for information flow control in web services. *International Journal of Web Services Research*, 3(1):1 - 28, 2006

[VOL96] D. Volpano, G. Smith, and C. Irvine. A sound type system for secure flow analysis. *Journal of Computer Security*, 4(3):167-187. December 1996.

[W3C01] World Wide Web Consortium (W3C). Web Services Description Language (WSDL) 1.1. March 2001, http://www.w3.org/TR/wsdl.

[W3C02] World Wide Web Consortium (W3C). XML encryption syntax and processing. December 2002. Available at: http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/

[W3C07] World Wide Web Consortium (W3C). SOAP Version 1.2 Specification Assertions and Test Collection (Second Edition). W3C Recommendation. 27 April 2007. http://www.w3.org/TR/2007/REC-soap12-testcollection-20070427/

[W3C99] World Wide Web Consortium (W3C). XML path language (XPath) version 1.0. November 1999. Available at: *http://www.w3.org/TR/xpath*

[WAL00] D. S. Wallach, A. W. Appel, and E. W. Felten. Safkasi: a security mechanism for language-based systems. *ACM Trans. Softw. Eng. Methodol.*, 9(4):341-378, 2000.

[WEE01] WEEKS, S. Understanding trust management systems. *Proceedings of IEEE Symposium on Security and Privacy*. 2001.

[WEE05] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D.F. Ferguson. Web Services Platform Architecture: SOAP, WSDL, WS-Policy,WS-addressing, WSPBEL, WS-Reliable Messaging, and More. Prentice Hall PTR, 2005.

[WIN02] M.Winslett, T.Yu, K.E.Seamons, et al. Negotiating Trust on the Web. IEEE Internet Computing, *Vol.6, No.6, pp. 30-37.* November/December 2002

[XU 01] C. Xu, H. Yan and F. Liu. The implementation of role-based access control on the Web. *0-7803-7010-4/01$10.00©2001 IEEE*, pp.251-255. April 2001.

[YUA05] E. Yuan, J. Tong. Attributed Based Access Control (ABAC) for Web Services. *Proceedings of the IEEE International Conference on Web Services (ICWS'05)*, 2005.

Résumé

Le contrôle d'accès (CA) et le contrôle de flux d'information (CFI) sont des mécanismes de sécurité très importants pour les environnements informatiques distribués et dynamiques tels que les services Web. Le CA permet de gérer les accès aux systèmes informatiques. Il est utilisé pour les bases de données et pour les applications distribuées sur des réseaux fermés tels que les réseaux LAN et il est adopté pour les services Web. Le CFI quant à lui, assure que les flux d'information au sein d'un système ne provoquent pas des divulgations ou des destructions d'information. Il a été introduit pour les langages de programmation et il est utilisé dans d'autre domaine comme les bases de données et les systèmes distribués.

Dans ce projet, nous avons amélioré un modèle de CFI pour les services Web proposé par Z. Tari et al et qui assure la confidentialité et l'intégrité des informations de bout en bout en utilisant les labels décentralisés. Nous avons amélioré les règles et les définitions des jointures des labels et nous avons étendu la structure du label pour faciliter la vérification d'intégrité. En outre, nous avons intégré un modèle de CA pour les services Web qui adapte le modèle RBAC (Role Based Access Control) en utilisant la notion d'attributs pour permettre une gestion de confiance flexible et en tenant compte des attributs concernant les demandeurs de services, les fonctions des services ainsi que les contextes des requêtes pour la spécification des politiques d'accès.

Mots clé : Services Web, sécurité dans les services Web, Contrôle d'accès, Contrôle de flux d'information, Gestion de confiance dans les services Web

Abstract

Access Control (AC) and Information flow control (IFC) are important security mechanisms for distributed and dynamic environment such as web services. AC allows access management to computer systems. It is used for data bases and LAN-based applications, and it is adopted for web services. IFC guarantees that information flows through computer systems do not lead to information leakage and destruction. IFC is introduced in programming languages and is used in other fields such as databases and distributed computing.

In this project, we have improved an IFC model for web services proposed by Z. Tari and al. which ensures an end to end confidentiality and integrity of information using decentralized labels. We have improved the rules and the labels join definitions and extended the label structure to facilitate integrity verification. Furthermore, we have integrated an AC model for web services which adapt the RBAC (Role Based Access Control) model using the attribute concept for trust management and taking into account service requestor's attributes, services' methods attributes and context attributes for access policy specification.

Key Words: Web services, Security in web services, Access Control, Information Flow Control, trust management in web services.