

Un cadre formel pour le contrôle d'accès

Mathieu Jaume

SPI - LIP6 - Université Paris 6
104 av. du Président Kennedy
F-75016 Paris, France

Abstract. Un des aspects de la sécurité en informatique concerne le contrôle des accès aux données d'un système pour lequel différentes politiques de sécurité peuvent être mises en application. Toutefois, rien ne sert de mettre en place une politique de sécurité pour gérer un système si les programmes chargés de garantir le bon fonctionnement de cette politique ne sont pas fiables. Cet article rend compte de manière informelle de différentes expériences permettant d'obtenir des développements formels de politiques de contrôle d'accès. Ces développements nous conduisent à introduire un "cadre sémantique" dans lequel il est possible de spécifier et d'implanter des politiques de contrôle d'accès. Ce cadre permet de définir des mécanismes de comparaison de modèles et d'analyser ces modèles en termes de flots d'information qu'ils autorisent.

Mots clés : Contrôle d'accès, Méthodes formelles

1 Introduction – Motivations

La protection des informations d'un système informatique est une préoccupation majeure. L'apparition de systèmes informatiques de plus en plus grands, la dissémination de l'information et le développement des réseaux, permettent dorénavant des attaques depuis l'extérieur et rendent la protection des informations de plus en plus complexe. Comme dans toutes les autres disciplines scientifiques, le besoin de recourir à des modèles et formalismes mathématiques se fait ressentir pour mieux comprendre et analyser les problèmes liés à l'informatique. C'est de ce besoin que viennent les méthodes formelles. Dans [1], P. Amey définit la "chose formelle" comme une "chose soutenue par une rigueur mathématique". Ainsi, les méthodes formelles peuvent être vues comme des "méthodes soutenues par une rigueur mathématique" dont l'absence d'ambiguïté permet de spécifier et d'implanter un système en garantissant que certaines propriétés sont respectées. Lorsqu'il s'agit de systèmes logiciels critiques, ces propriétés peuvent être vitales. Dans cet article, nous utilisons les méthodes formelles pour étudier certaines des propriétés classiques de sécurité des systèmes informatiques. Nous nous intéressons plus particulièrement au contrôle d'accès. Il s'agit de régir et de gérer les accès effectués selon certains modes (lecture, écriture, ...) par des sujets, les entités actives (processus, programmes, utilisateurs, ...) sur des objets, les entités passives (données, fichiers, programmes, ...). A plus long

terme, notre objectif est d'obtenir une bibliothèque certifiée de moniteurs de référence mettant en application différentes politiques de sécurité. En effet, le développement logiciel d'un moniteur de référence n'a de sens que s'il permet de garantir les propriétés de sécurité pour lesquelles il a été conçu. Pour atteindre de hauts niveaux de certification, il est nécessaire de fournir un modèle formel du système permettant d'obtenir des preuves formelles mécanisées. Nous présentons ici plusieurs expériences menées pour atteindre cet objectif. Comme nous allons le voir, trois difficultés sont à prendre en compte dans ce travail. La première est classique et provient de l'activité même de formalisation : le passage de l'informel au formel nécessite d'identifier les hypothèses implicites et d'explicitier totalement le système à modéliser. Afin de valider la formalisation obtenue, il s'agit alors de la "mécaniser" (i.e. de l'implanter). C'est la deuxième difficulté : certaines preuves "triviales" à obtenir sur le papier le sont beaucoup moins avec un assistant à la preuve. Quoi qu'il en soit, conduire un développement formel de cette nature est une activité chronophage. Il faut, dans la mesure du possible, factoriser les spécifications et les preuves formelles afin de faciliter la réutilisation de ces développements. Bien sûr, l'utilisation d'un atelier de développement formel muni de mécanismes facilitant l'écriture modulaire de spécifications, de définitions et de preuves permet d'atteindre une certaine "réutilisabilité" des développements conduits. Ce n'est toutefois pas suffisant. Il est en effet souhaitable de concevoir un cadre formel uniforme dans lequel puissent s'exprimer les modèles de contrôle d'accès que nous envisageons. C'est la troisième difficulté : il s'agit à la fois d'identifier les "ingrédients" communs aux politiques de contrôle d'accès, d'exprimer les propriétés génériques qu'ils vérifient, d'en prouver certaines et de formaliser les politiques envisagées comme des instances du cadre générique. Un tel cadre procure un formalisme commun pour décrire des modèles de contrôle d'accès et permet de dégager des techniques d'analyse de ces modèles (comparaison, flots d'informations, ...). Cet article a pour objectif de fournir une présentation synthétique des divers travaux que nous avons réalisés sur la formalisation et l'implantation des politiques de contrôle d'accès dans un cadre formel [12, 18–22, 14, 17].

2 Descriptions formelles de modèles de contrôle d'accès

Cette section rend compte de manière informelle de différentes expériences de développements formels de modèles de contrôle d'accès dont l'objectif est de certifier des programmes en charge de la sécurité dans un système d'information. Plus précisément, il s'agit de garantir qu'un moniteur de référence chargé du contrôle des accès dans un système maintient une politique de contrôle d'accès donnée. Cette propriété est bien sûr cruciale pour la plupart des systèmes. Comme nous l'avons suggéré dans l'introduction, la première difficulté provient du passage de l'informel au formel. En effet, bon nombre de politiques sont exprimées de manière informelle dans la littérature. Pour illustrer cela, considérons l'exemple classique de la politique de la Muraille de Chine, introduite par Brewer et Nash [6], pour résoudre les problèmes de conflit d'intérêt dans le monde des

consultants. Chaque objet du système appartient à un ensemble de données d’une compagnie, et chaque compagnie appartient à une unique classe de conflit d’intérêt. Ces classes de conflit correspondent à des milieux professionnels distincts, comme par exemple les banques ou les compagnies pétrolières. Selon cette politique, un consultant peut travailler en même temps pour une banque et une compagnie pétrolière, mais ne peut pas le faire pour deux banques ou deux compagnies pétrolières. Il existe de plus une classe de conflit spéciale qui ne contient qu’une seule compagnie et qui contient les informations “sanitisées”, c’est-à-dire celles qui peuvent être lues par tout le monde sans provoquer de conflit d’intérêt. La politique est énoncée comme la combinaison de deux propriétés de sécurité. Nous répétons volontairement en anglais l’une d’entre elles, afin de permettre au lecteur de mieux percevoir le niveau de formalisation de la description originale de cette politique.

[6] *Write access is only permitted if (a) access is permitted by the simple security rule, and (b) no object can be read which is in a different company dataset to the one for which write access is requested and contains unsanitized information.*

D’une part, cette propriété est difficilement compréhensible à la première lecture, du fait de la structure de la phrase employée. D’autre part, certaines notions restent indéfinies. Quel est le statut exact d’un objet qualifié de “*object can be read*” ? Est-ce un objet que le sujet est en train de lire ? ou bien un objet qu’il lira dans l’avenir ? cela signifie-t-il qu’il a les droits pour le faire ? Pour formaliser cette propriété, il faut faire des choix quant à l’interprétation de ces concepts. Le danger provient de ces choix qui permettent plusieurs formalisations non équivalentes de cette politique. Enfin, l’étape de formalisation permet de distinguer clairement d’une part les propriétés de sécurité souhaitées et d’autre part le moniteur de référence chargé de les faire respecter. En effet, la présentation informelle de la politique de la Muraille de Chine peut s’apparenter à une description algorithmique de la fonction d’autorisation des accès. Cela peut amener à considérer immédiatement le “comment”, donc l’implantation, au lieu de réfléchir d’abord au “quoi”, c’est-à-dire à la spécification de la politique de sécurité.

2.1 Formalisation du modèle de Bell & LaPadula avec Coq

Nous présentons ici brièvement une formalisation du modèle de Bell et LaPadula [23, 3] dans le système Coq à partir de laquelle un programme certifié implantant un moniteur de référence pour cette politique a été extrait. Ce travail est décrit dans [12].

Politique de Bell & LaPadula La politique de Bell et LaPadula est habituellement décrite par une machine à états [23, 3]. Elle dépend d’un ensemble \mathcal{S} de sujets, d’un ensemble \mathcal{O} d’objets, d’un ensemble \mathcal{A} de modes d’accès et d’un treillis fini $(\mathcal{L}, \preceq, \gamma, \lambda)$ de niveaux de sécurité. Un état du système est décrit par un

quadruplet (m, D, f_s, f_o) où m est l'ensemble des accès courants effectués dans le système, D est l'ensemble des droits d'accès et $f_s : \mathcal{S} \rightarrow \mathcal{L}$ (resp. $f_o : \mathcal{O} \rightarrow \mathcal{L}$) est une fonction associant un niveau de sécurité aux sujets (resp. aux objets). Les éléments de m et de D sont des accès représentés par des triplets de la forme $(s, o, a) : (s, o, a) \in m$ signifie qu'un sujet s accède à un objet o selon le mode d'accès a tandis que $(s, o, a) \in D$ signifie qu'un sujet s dispose des droits (discrétionnaires) pour accéder à un objet o selon le mode d'accès a . Les trois propriétés de sécurité définies dans la politique de Bell et LaPadula sont les suivantes.

- Propriété DAC (*Discretionary Access Control*) : La propriété DAC exprime que tout accès courant est conforme aux droits d'accès : $m \subseteq D$
- Propriétés MAC et MAC* (*Mandatory Access Control*) :
 - La propriété MAC (*“no read-up property”*) exprime qu'un sujet ne peut accéder en lecture à un objet que si son niveau de sécurité est supérieur à celui de l'objet accédé : $(s, o, \text{read}) \in m \Rightarrow f_s(s) \succeq f_o(o)$
 - La propriété MAC* (*“no write-down property”*) permet d'éviter qu'un sujet recopie de l'information sensible à un niveau de sécurité inférieur :

$$((s, o_1, \text{read}) \in m \wedge (s, o_2, \text{write}) \in m) \Rightarrow f_o(o_1) \preceq f_o(o_2) \quad (1)$$

Certification d'un moniteur de référence Nous esquissons ici les grandes lignes de la formalisation du modèle de Bell et LaPadula dans le système Coq. Ce développement est paramétré par $\mathcal{S}, \mathcal{O}, \mathcal{A}$ et un treillis $(\mathcal{L}, \preceq, \gamma, \wedge)$. Ce treillis est en fait obtenu à partir de deux paramètres : un ensemble \mathcal{K} de “domaines” (connu sous le nom de *“needs-to-know”*), comme par exemple {nucléaire, médical, ... }, et un ensemble \mathcal{C} de classifications, comme par exemple {Top-secret, secret, public, ... }, muni d'une relation d'ordre total. \mathcal{L} est alors défini comme le treillis produit $\mathcal{C} \times T_k$ où $T_k = (\wp(\mathcal{K}), \subseteq, \cup, \cap)$ est le treillis des parties de \mathcal{K} . A partir de la définition de l'ensemble Σ des états, les fonctions de transition sont définies comme des fonctions de $\mathcal{R} \times \Sigma$ dans $\mathcal{D} \times \Sigma$ où \mathcal{R} est un ensemble de requêtes et $\mathcal{D} = \{\text{yes}, \text{no}\}$ contient les réponses possibles. Une fonction de transition τ , correspondant à la mise en application d'une politique de sécurité, sera qualifiée de fonction “sûre” ssi elle transforme chaque état vérifiant les propriétés DAC, MAC et MAC* en états vérifiant encore ces propriétés. Le développement réalisé consiste en une implantation dans Coq de la fonction de transition τ_{BLP} introduite par Bell et LaPadula ainsi que de la preuve mécanisée du célèbre *“Basic Security Theorem”* [3] affirmant que τ_{BLP} est “sûre”. Le mécanisme d'extraction de Coq a permis, à partir de cette preuve, d'obtenir un programme certifié qui implante τ_{BLP} .

Ce travail de mécanisation de la preuve a non seulement fourni une implantation avec un niveau de confiance élevé mais a aussi permis de reconnaître, parmi les hypothèses sur la fonction de transition, celles qui sont nécessaires au maintien de la politique de sécurité. Cela permet donc d'envisager des variantes de la fonction τ_{BLP} qui préservent les propriétés de sécurité. Toutefois, si elle remplit ses objectifs en termes de garantie de correction, une telle approche peut conduire à des développements difficilement réutilisables. En effet,

la moindre modification tant sur la spécification de la politique que sur son implantation conduit à modifier une preuve de plus d'un millier de lignes. Il s'agit là d'un exercice particulièrement chronophage. Pour faciliter la réutilisation, il est nécessaire d'élaborer un cadre formel caractérisant les éléments communs aux politiques de contrôle d'accès et permettant d'en dériver progressivement la spécification complète de la politique choisie, puis différentes implantations. Il est également préférable d'utiliser un environnement de développement qui facilite l'implantation de ces différents traits. Même si le système Coq dispose à présent de mécanismes permettant de conduire des développements formels de manière modulaire, nous avons choisi d'utiliser l'atelier FoCaL, mieux adapté car offrant des traits objets.

2.2 Implantation de l'algèbre des modèles de sécurité de McLean

Afin de faciliter la réutilisation des développements formels de modèles de contrôle d'accès, l' "algèbre des modèles de sécurité" introduite par McLean [25] a été implantée avec l'environnement de développement FoCaL [28, 9]. Cette algèbre définit un cadre générique pour le contrôle d'accès qui peut ensuite être instancié afin d'en dériver une implantation de la politique de Bell et LaPadula, qui peut à son tour être utilisée pour gérer les accès à une base de données relationnelle. Nous décrivons ici l'architecture de l'ensemble de ces développements qui sont détaillés dans [26, 18, 19, 4].

Algèbre des modèles de sécurité L' "algèbre des modèles de sécurité" permet la description d'un modèle de contrôle d'accès à trois niveaux de spécification différents : les frameworks, les modèles et les systèmes. Le framework décrit l'environnement : il est paramétré par \mathcal{S} , \mathcal{O} , \mathcal{A} et un treillis \mathcal{L} et spécifie quels sont les ensembles de sujets qui pourront conjointement demander à accéder à un objet ou demander à changer le niveau de sécurité d'un sujet ou d'un objet (sans toutefois spécifier comment sera traitée cette demande). McLean introduit ici la notion d'accès conjoints : certaines opérations ne peuvent être autorisées que si elles sont demandées par un certain groupe de sujets. Dans le cas le plus général, les ensembles de sujets qui pourront conjointement soumettre une requête sont des parties non vides quelconques de \mathcal{S} (toute opération est initiée par au moins un sujet) mais il peut exister des contraintes sur ces ensembles. Par exemple, certaines politiques imposent que si un groupe de sujets peut conjointement soumettre une requête, alors tout sous-ensemble de ce groupe peut également soumettre cette requête. Dans le cas le plus courant, sans accès conjoints, les seuls ensembles autorisés à soumettre une requête sont les singletons construits à partir de \mathcal{S} . Les différentes possibilités pour les ensembles de sujets permettent de construire une hiérarchie de frameworks permettant de factoriser les spécifications. La notion de modèle, paramétrée par celle de framework, a été introduite afin de pallier aux problèmes posés par des systèmes de contrôle d'accès ne fixant aucune règle régissant le changement de niveau de sécurité d'un sujet ou d'un objet. Un modèle spécifie quels sont les ensembles de sujets qui seront

effectivement autorisés à modifier les niveaux de sécurité. Comme pour les frameworks, les différentes propriétés vérifiées par ces ensembles de sujets permettent de construire une hiérarchie de modèles. Certaines relations et opérations sur les modèles sont définies et permettent d’implanter l’ensemble des modèles comme une instance de treillis distributif. Les états décrivent les informations relatives aux différents niveaux de sécurité associés aux sujets et aux objets ainsi que les accès courants. Une fois la notion d’état définie, on spécifie par un prédicat quels sont les états sûrs, c’est-à-dire quelle est la politique de sécurité appliquée. C’est à partir de cette notion d’état et de modèle qu’est définie la notion de système. Celle-ci décrit la fonction de transition et les propriétés qu’elle doit vérifier. Cette fonction de transition entre états décrit les changements d’états produits lors des requêtes d’accès émises par les sujets. Ces trois notions ont été mécanisées dans l’atelier FoCaL puis utilisées pour traiter le cas particulier de la politique de Bell et LaPadula.

Détection de flots d’information non autorisés Ici, l’étape de formalisation et de mécanisation de la formalisation avec FoCaL a permis de corriger la spécification initiale. En effet, dans [25], McLean illustre l’utilisation du cadre qu’il introduit en spécifiant en son sein une version enrichie de la politique de Bell et LaPadula, essentiellement en considérant la notion d’accès conjoints. La propriété de sécurité MAC* définie en (1) est alors (re)définie comme suit :

[25] *a state is \star -secure if for any subjects S_1, S_2 and objects o_1, o_2 , if $(S_1, o_1, \text{read}) \in m$ and $(S_2, o_2, \text{write}) \in m$ and the classification of o_1 dominates that of o_2 , then $S_1 \cap S_2 = \emptyset$*

Ici, m est l’ensemble des accès courants, S_1 et S_2 sont des ensembles de sujets et un accès est un triplet (S, o, a) exprimant que les sujets présents dans l’ensemble S accèdent conjointement à l’objet o selon le mode a . La formalisation de cet énoncé permet d’obtenir, par contraposition, la spécification suivante :

$$((S_1, o_1, \text{read}) \in m \wedge (S_2, o_2, \text{write}) \in m \wedge S_1 \cap S_2 \neq \emptyset) \Rightarrow \neg(f_o(o_2) \prec f_o(o_1)) \quad (2)$$

Or, si on se limite au cas où S_1 et S_2 sont réduits à des singletons, c’est-à-dire si on ne considère pas les accès conjoints, la propriété (1) implique la propriété (2), mais la réciproque est en général fautive : (2) n’implique (1) que si l’ordre sur les niveaux de sécurité est total. En général cet ordre est partiel (puisqu’il définit une structure de treillis) et la propriété MAC* n’est alors pas vérifiée (bien que (2) soit satisfaite) comme le montre l’exemple suivant. En respectant la propriété (2), un sujet s_1 peut accéder simultanément en lecture à un objet de niveau de sécurité l_1 et en écriture à un objet de niveau de sécurité l_2 tels que l_1 et l_2 ne soient pas comparables. Un sujet s_2 peut alors lire ce dernier objet de niveau l_2 et écrire simultanément dans un objet de niveau l_3 tel que l_3 et l_2 ne soient pas comparables mais tel que l_3 soit inférieur à l_1 . Il y a alors une “fuite d’information vers le bas” puisqu’il devient ainsi possible de recopier les informations d’un objet de niveau l_1 dans un objet de niveau $l_3 \preceq l_1$.

L’activité de formalisation permet de mettre en lumière de tels problèmes. Ils peuvent être découverts en essayant de prouver des “énoncés faux” (par exemple

en essayant de prouver l'équivalence entre (1) et (2) lorsque S_1 et S_2 sont des singletons), mais ils peuvent parfois aussi être découverts en utilisant des méthodes permettant de tester une propriété, par exemple une propriété assurant qu'il n'existe pas de séquence d'accès permettant de copier de l'information de niveau élevé dans des objets de niveaux inférieurs. Pour ce faire, il est possible d'utiliser, dans certaines conditions, un outil de test intégré à l'atelier FoCaL développé par M. Carlier et C. Dubois [7]. Citons aussi les travaux de C. Morisset et A. Santana de Oliveira [15, 27, 26, 8] qui décrivent un algorithme permettant de détecter la "fuite d'information" correspondant à l'exemple présenté dans cette section à partir d'une spécification de politique par un système de réécriture. La description de politiques de sécurité *via* un système de réécriture a été étudiée par A. Santana de Oliveira dans [8].

Instanciation du modèle et application En définissant, c'est-à-dire en instanciant, chacune des entités spécifiées dans le cadre générique, nous avons défini avec FoCaL la politique de Bell et LaPadula et avons fourni une implantation de la fonction de transition τ_{BLP} . Cette implantation reste encore relativement générique : elle ne spécifie ni les sujets, ni les objets. En effet, un programme qui plante une certaine politique de contrôle d'accès est *a priori* indépendant du système sur lequel il doit s'appliquer (bases de données, systèmes de gestion de fichiers, ...). Il possède des paramètres destinés à prendre en compte l'environnement concret dans lequel il va s'exécuter. Nous allons maintenant illustrer succinctement l'intégration de tels programmes dans un système "concret", une base de données relationnelle. Il s'agit ici d'une mise à l'épreuve du terrain de la méthode toute entière.

La base de données (MySQL) contient à la fois les données des utilisateurs, et les données relatives à la sécurité (droits d'accès, accès courants, niveaux de classification, "needs-to-know"). Les paramètres du système sont instanciés par des objets stockés dans la partie "sécurité" de la base de données et les états de ce système sont construits à partir de ces données. Les objets du système sont les tables de la base de données. Une granularité plus fine pour la notion d'objet pourrait être envisagée mais soulèverait des problèmes connus dans le domaine des bases de données pour lesquels des solutions existent mais sortent du cadre de notre prototype. La figure 1 illustre le traitement des requêtes effectué par l'application obtenue. Etant donnée une requête SQL soumise par un sujet (authentifié), le programme d'analyse de requêtes SQL que nous avons développé fait appel au système de Bell et LaPadula et, selon la réponse obtenue, traite ou refuse l'exécution de la requête soumise. Pour cela, nous avons défini pour chaque requête SQL, un ensemble de requêtes pour le système de Bell et LaPadula, donnant en quelque sorte une sémantique en termes d'accès aux requêtes SQL. On remarquera que les requêtes SQL soumises par l'utilisateur utilisent la syntaxe standard de SQL, rendant ainsi transparente pour l'utilisateur la mise en œuvre de notre application, qui peut être vue comme un filtre entre l'utilisateur et le système de gestion de la base de données. Ce développement est décrit en détail dans [26, 4].

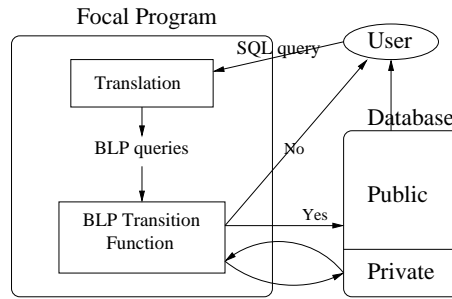


Fig. 1. Contrôle d'accès pour un SGBD

2.3 Nécessité de définir un cadre plus générique

S'il permet de caractériser certains aspects communs à différentes politiques de contrôle d'accès, le cadre introduit par McLean n'est pas encore suffisamment riche. Il est certes bien adapté à la définition de politiques basées sur un ensemble de niveaux de sécurité associés à certaines entités du système mais ne permet pas de définir des politiques comme la Muraille de Chine (dans le contexte de la Muraille de Chine nous ne disposons pas d'un treillis de niveaux de sécurité mais d'un ensemble de classes de conflits) ou encore des politiques à base de rôles (dans le contexte d'une politique à base de rôles, nous ne disposons pas d'un treillis de niveaux de sécurité mais d'un ensemble de rôles muni d'un ordre partiel). Le cadre de McLean se révèle donc à la fois trop contraignant et trop peu expressif (en termes de propriétés de comparaison de modèles, de simulation d'implantations permettant une plus grande réutilisabilité, d'analyse de flots d'informations). Nous avons alors été amenés à concevoir un cadre sémantique pour le contrôle d'accès permettant de répondre à nos objectifs. Il ne s'agit pas de la définition d'un langage mais plutôt d'une spécification de style mathématique de ce qui constitue une politique de contrôle d'accès. Ce cadre et son utilisation sont présentés dans la section suivante.

3 Un cadre formel pour le contrôle d'accès

3.1 Modèles de contrôle d'accès

Politiques de contrôle d'accès Une politique de contrôle d'accès permet de caractériser les états d'un système et de spécifier ce qu'est un état sûr en fonction d'informations de sécurité associées aux entités du système. Les entités du système peuvent être réparties dans deux ensembles : l'ensemble \mathcal{S} des sujets, qui correspondent aux entités qui effectuent les actions, et l'ensemble \mathcal{O} des objets, qui subissent les actions. Nous désignons par \mathcal{A} l'ensemble des modes d'accès qui caractérisent les différents types d'accès effectués par les sujets sur les objets. Cet ensemble contient généralement `read`, `write`, `append`, etc. Une approche

classique consiste à représenter un accès par un triplet (s, o, a) , signifiant que le sujet s accède à l'objet o selon le mode d'accès a . Néanmoins, d'autres approches existent (on peut par exemple regrouper les modes d'accès, ou encore considérer les accès conjoints). Afin de pouvoir prendre en compte ces différentes situations, nous nous limitons à dénoter par \mathbb{A} l'ensemble de tous les accès, sans introduire une véritable définition qui serait trop précise à ce stade.

Les systèmes de contrôle d'accès sont ici modélisés sous la forme de machines à états. Un état représente le système à un instant donné et contient au moins une description de l'ensemble des *accès courants*, c'est-à-dire de tous les accès qui ont été acceptés et qui n'ont pas encore été relâchés. Ces accès sont donc supposés être effectués simultanément dans le système. L'ensemble des états est noté Σ et l'ensemble des accès courants d'un état σ est noté $\Lambda(\sigma)$. Une politique de contrôle d'accès permet de spécifier un sous-ensemble de Σ , contenant les états sûrs, c'est-à-dire les états qui vérifient la politique. Afin de déterminer si un état est sûr, les entités du système sont associées à des informations de sécurité :

- le paramètre de sécurité, noté ρ , décrit les informations de sécurité *statiques* de la politique, c'est-à-dire les informations de sécurité qui ne changeront pas durant la vie du système,
- les fonctions de sécurité décrivent les informations de sécurité *dynamiques* de la politique, c'est-à-dire les informations de sécurité susceptibles d'évoluer durant la vie du système (étant donné un état σ , on note $\Upsilon(\sigma)$ les fonctions de sécurité associées à l'état σ).

A partir de ces informations, les états sûrs sont caractérisés par un prédicat Ω sur les états. On note Σ_{Ω} l'ensemble $\{\sigma \in \Sigma \mid \Omega(\sigma)\}$ des états sûrs. Toutes ces notions permettent de définir une politique de contrôle d'accès $\mathbb{P}[\rho]$, basée sur un paramètre de sécurité ρ , par un quintuplet :

$$\mathbb{P}[\rho] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma, \Omega)$$

A ce niveau de spécification, il est possible de caractériser certaines propriétés que peuvent vérifier les politiques de contrôle d'accès.

- Une politique est dite *compacte* si lorsque l'on supprime des accès de l'ensemble des accès courants d'un état sûr, l'état reste sûr :

$$\forall \sigma_1 \in \Sigma \quad \Omega(\sigma_1) \Rightarrow (\forall \sigma_2 \in \Sigma \quad (\Lambda(\sigma_2) \subseteq \Lambda(\sigma_1) \wedge \Upsilon(\sigma_1) = \Upsilon(\sigma_2)) \Rightarrow \Omega(\sigma_2))$$

La plupart des politiques classiques (HRU, Bell et LaPadula, Muraille de Chine, RBAC, ...) sont compactes. Toutefois, il peut être utile de définir des politiques non compactes dans certains contextes. Considérons par exemple une politique de contrôle des accès aux ressources d'un système qui stipule qu'un utilisateur est autorisé à utiliser des ressources d'une équipe à laquelle il n'appartient pas, mais seulement dans le cas où toutes les ressources de son équipe sont déjà accédées. Une telle politique n'est pas compacte puisque lorsqu'un utilisateur accède à une ressource d'une équipe à laquelle

il n'appartient pas, retirer un accès sur une ressource de son équipe conduit à un état non sûr.

- Une politique est dite *libre* si lorsqu'un accès est autorisé pour un état sûr du système, il est autorisé pour tous les états sûrs¹ :

$$\forall \sigma \in \Sigma \forall s \in \mathcal{S} \forall o \in \mathcal{O} \forall a \in \mathcal{A} \\ ((\exists \sigma' \in \Sigma (\Omega(\sigma') \wedge (s, o, a) \in \Lambda(\sigma'))) \wedge \Omega(\sigma)) \Rightarrow \Omega(\sigma \oplus (s, o, a))$$

Par exemple, HRU (uniquement basée sur un ensemble D d'accès autorisés) définit une politique libre puisque le prédicat caractérisant les états sûrs est défini en examinant chacun des accès courants indépendamment des autres accès en cours. En revanche, la politique de Bell et LaPadula n'est pas libre puisqu'il est possible qu'un sujet soit autorisé à écrire dans un objet lorsque ses accès en lecture vérifient certaines conditions et ne soit plus autorisé à écrire dans cet objet lorsque ses accès en lecture ne vérifient plus ces conditions.

Nous introduisons à présent la notion de modèle de contrôle d'accès, qui permet de spécifier comment passer d'un état du système à un autre. Pour cela, nous introduisons tout d'abord la notion de langage de requêtes.

Requêtes Une requête est soumise par un sujet afin de faire évoluer le système, soit en ajoutant ou en enlevant un accès, soit en changeant les informations de sécurité dynamiques du système. Dans ce dernier cas, on parle de *requêtes administratives*. Etant donné un ensemble de requêtes \mathcal{R} , il est nécessaire d'introduire une sémantique pour le langage de requêtes afin de pouvoir caractériser les modifications sur les états engendrées par l'application de ces requêtes lors d'une transition effectuée par le moniteur de référence. Le procédé le plus classique pour définir une sémantique du langage des requêtes consiste à introduire une "sémantique de transitions" *via* une relation $\llbracket \mathcal{R} \rrbracket_{\Sigma} \subseteq \Sigma \times \mathcal{R} \times \Sigma$. Avec une telle approche, $(\sigma_1, R, \sigma_2) \in \llbracket \mathcal{R} \rrbracket_{\Sigma}$ permet de spécifier les propriétés d'un état σ_2 lorsqu'il a été obtenu par application d'une requête R sur un état σ_1 . Par exemple, la sémantique de la requête $\langle +, s, o, a \rangle$ de demande d'ajout d'un accès du sujet s sur l'objet o selon le mode a est spécifiée par :

$$(\sigma_1, \langle +, s, o, a \rangle, \sigma_2) \in \llbracket \mathcal{R} \rrbracket_{\Sigma} \Leftrightarrow (\Lambda(\sigma_2) = \{(s, o, a)\} \cup \Lambda(\sigma_1) \wedge \Upsilon(\sigma_1) = \Upsilon(\sigma_2))$$

Modèles et Implantations La donnée d'une politique $\mathbb{P}[\rho]$ et d'un langage de requêtes \mathcal{R} muni d'une sémantique $\llbracket \mathcal{R} \rrbracket_{\Sigma}$ constitue un modèle :

$$\mathbb{M}[\rho] = (\mathbb{P}[\rho], \mathcal{R})$$

¹ $\sigma \oplus (s, o, a)$ (resp. $\sigma \ominus (s, o, a)$) dénote un état tel que :

$$\Lambda(\sigma \oplus (s, o, a)) = \Lambda(\sigma) \cup \{(s, o, a)\} \text{ et } \Upsilon(\sigma \oplus (s, o, a)) = \Upsilon(\sigma) \\ (\text{resp. } \Lambda(\sigma \ominus (s, o, a)) = \Lambda(\sigma) \setminus \{(s, o, a)\} \text{ et } \Upsilon(\sigma \ominus (s, o, a)) = \Upsilon(\sigma))$$

Implanter un modèle consiste à définir une paire (τ, Σ_I) où $\tau : \mathcal{R} \times \Sigma \rightarrow \mathcal{D} \times \Sigma$ est une fonction de transition entre états (\mathcal{D} est l'ensemble des réponses possibles) et où Σ_I est l'ensemble des états initiaux possibles (il s'agit bien sûr d'un sous-ensemble des états sûrs). A ce niveau, il est possible de spécifier les propriétés attendues sur (τ, Σ_I) . Les deux propriétés principales sont les propriétés de correction vis-à-vis de la politique (τ transforme tout état sûr en état sûr) et de la sémantique des requêtes (les états obtenus en appliquant τ à une requête R satisfont la relation $\llbracket \mathcal{R} \rrbracket_\Sigma$). Nous notons $\mathbb{M}[\rho] \vdash (\tau, \Sigma_I)$ lorsque toutes les propriétés de correction d'une implantation (τ, Σ_I) d'un modèle $\mathbb{M}[\rho]$ sont satisfaites.

Applications Le cadre formel décrit ci-dessus a été implanté avec l'atelier FoCaL et instancié pour obtenir une implantation des principales politiques que l'on peut rencontrer dans la littérature : [5] décrit une implantation de politique discrétionnaire à base de matrice de droits d'accès [16], [2] décrit les implantations d'une politique à la Unix intégrant la notion de groupes d'utilisateurs, d'une politique à base de "tickets" [24], et d'une politique discrétionnaire intégrant un mécanisme de délégation, [26] décrit une implantation de la politique de Bell et LaPadula [23, 3] et [13] décrit une implantation d'une politique à base de rôles (RBAC96) [10, 29].

3.2 Comparaison de modèles

Le cadre sémantique que nous venons d'introduire nous permet de formaliser les concepts nécessaires pour envisager la comparaison de modèles de contrôle d'accès. Intuitivement, un modèle de contrôle d'accès $\mathbb{M}_1[\rho_1]$ est plus restrictif qu'un modèle $\mathbb{M}_2[\rho_2]$ ssi toute implantation correcte de $\mathbb{M}_1[\rho_1]$ peut être simulée par une implantation correcte de $\mathbb{M}_2[\rho_2]$.

Préordre sur les modèles Afin de donner une définition du préordre sur les modèles, il faut préciser la notion de simulation d'implantations. Il s'agit du concept de simulation faible : la fonction de transition $\tau_2 : \mathcal{R}_2 \times \Sigma_2 \rightarrow \mathcal{D} \times \Sigma_2$ simule faiblement la fonction $\tau_1 : \mathcal{R}_1 \times \Sigma_1 \rightarrow \mathcal{D} \times \Sigma_1$, ce que nous notons $\tau_1 \stackrel{\kappa_\Sigma, \kappa_\mathcal{R}}{\sim} \tau_2$, ssi il existe deux relations $\kappa_\Sigma \subseteq \Sigma_1 \times \Sigma_2$ (caractérisant les états "sémantiquement" équivalents) et $\kappa_\mathcal{R} \subseteq \Sigma_1 \times \mathcal{R}_1 \times \mathcal{R}_2^*$ (mettant en correspondance les séquences de requêtes "sémantiquement" équivalentes) telles que :

$$\forall \sigma_1, \sigma'_1 \in \Sigma_1 \quad \forall \sigma_2^0 \in \Sigma_2 \quad \forall R_1 \in \mathcal{R}_1 \quad \forall d_1 \in \mathcal{D} \quad \left(\begin{array}{l} \tau_1(R_1, \sigma_1) = (d_1, \sigma'_1) \\ \wedge (\sigma_1, \sigma_2^0) \in \kappa_\Sigma \end{array} \right) \Rightarrow \left(\begin{array}{l} \exists \sigma_2^1, \sigma_2^2, \dots, \sigma_2^{n-1}, \sigma_2^n \in \Sigma_2 \\ \exists (R_2^1, \dots, R_2^n) \in \mathcal{R}_2^* \exists d_2^1, \dots, d_2^n \in \mathcal{D} \\ \tau_2(R_2^1, \sigma_2^0) = (d_2^1, \sigma_2^1) \\ \wedge \dots \wedge \tau_2(R_2^n, \sigma_2^{n-1}) = (d_2^n, \sigma_2^n) \\ \wedge d_1 = d_2^1 = d_2^2 = \dots = d_2^n \\ \wedge (\sigma_1, R_1, (R_2^1, \dots, R_2^n)) \in \kappa_\mathcal{R} \\ \wedge (\sigma'_1, \sigma_2^n) \in \kappa_\Sigma \end{array} \right)$$

Chaque transition de τ_1 doit donc pouvoir être simulée par une séquence de

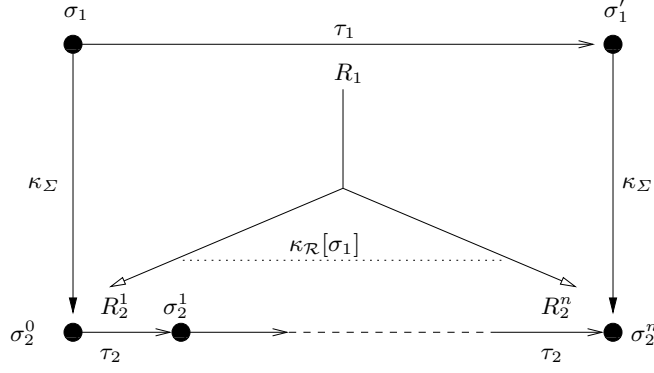


Fig. 2. Simulation faible

transitions de τ_2 . En effet, selon le pouvoir d'expression des deux langages de requêtes considérés \mathcal{R}_1 et \mathcal{R}_2 , il se peut que pour simuler l'effet d'une requête appartenant à \mathcal{R}_1 , il soit nécessaire d'appliquer plusieurs requêtes de \mathcal{R}_2 . C'est par exemple le cas pour la requête d'ajout d'un rôle autorisé pour un utilisateur dans le modèle RBAC, dont la simulation dans le modèle HRU nécessite d'ajouter plusieurs accès autorisés et donc d'appliquer plusieurs fois la requête du modèle HRU permettant d'autoriser un nouvel accès. Cette définition, illustrée sur la figure 2, est étendue aux implantations : l'implantation (τ_2, Σ_2^I) simule faiblement (τ_1, Σ_1^I) , ce que nous notons $(\tau_1, \Sigma_1^I) \stackrel{\kappa_{\Sigma}, \kappa_{\mathcal{R}}}{\sim} (\tau_2, \Sigma_2^I)$, ssi il existe deux relations $\kappa_{\Sigma} \subseteq \Sigma_1 \times \Sigma_2$ et $\kappa_{\mathcal{R}} \subseteq \Sigma_1 \times \mathcal{R}_1 \times \mathcal{R}_2^*$ telles que :

$$\tau_1 \stackrel{\kappa_{\Sigma}, \kappa_{\mathcal{R}}}{\sim} \tau_2 \wedge \forall \sigma_1 \in \Sigma_1^I \exists \sigma_2 \in \Sigma_2^I \quad (\sigma_1, \sigma_2) \in \kappa_{\Sigma}$$

La relation $\kappa_{\mathcal{R}}$ est une relation ternaire : le jugement $(\sigma, R_1, (R_2^1, \dots, R_2^n)) \in \kappa_{\mathcal{R}}$ exprime que la séquence de requêtes (R_2^1, \dots, R_2^n) simule la requête R_1 lorsque le système se trouve dans un état σ . En effet, la séquence de requêtes peut dépendre de l'état dans lequel se trouve le système. C'est par exemple le cas pour la requête d'ajout d'une permission (o, a) à un rôle r dans le modèle RBAC, dont la simulation dans le modèle HRU nécessite d'ajouter les accès (s, o, a) aux accès autorisés pour chaque sujet s ayant activé un rôle r' supérieur à r . Ici, l'ensemble des rôles activés par un sujet est défini par une fonction de sécurité, qui dépend donc de l'état du système.

Lors de la comparaison de modèles, les relations κ_{Σ} et $\kappa_{\mathcal{R}}$ utilisées doivent satisfaire certaines propriétés supplémentaires : d'une part κ_{Σ} et $\kappa_{\mathcal{R}}$ doivent être totales à gauche, et d'autre part, deux états en relation par κ_{Σ} doivent autoriser les mêmes accès (i.e. les ensembles d'accès que l'on peut ajouter aux accès courants de ces deux états sans violer la politique de sécurité sont identiques), on dira dans ce cas que κ_{Σ} est \mathcal{W} -préservante. Le préordre \preceq sur les modèles

peut à présent être défini comme suit :

$$\mathbb{M}_1[\rho_1] \trianglelefteq \mathbb{M}_2[\rho_2]$$

$$\Leftrightarrow \left(\begin{array}{l} \forall \tau_1 : \mathcal{R}_1 \times \Sigma_1 \rightarrow \mathcal{D} \times \Sigma_1 \quad \forall \Sigma_1^I \subseteq \Sigma_1 \\ \mathbb{M}_1[\rho_1] \vdash (\tau_1, \Sigma_1^I) \Rightarrow \left(\begin{array}{l} \exists \tau_2 : \mathcal{R}_2 \times \Sigma_2 \rightarrow \mathcal{D} \times \Sigma_2 \quad \exists \Sigma_2^I \subseteq \Sigma_2 \\ \exists \kappa_{\Sigma} \subseteq \Sigma_1 \times \Sigma_2 \quad \exists \kappa_{\mathcal{R}} \subseteq \Sigma_1 \times \mathcal{R}_1 \times \mathcal{R}_2^* \\ \kappa_{\Sigma} \text{ et } \kappa_{\mathcal{R}} \text{ sont totales à gauche} \\ \wedge \kappa_{\Sigma} \text{ est } \mathcal{W}\text{-préservante} \\ \wedge \mathbb{M}_2[\rho_2] \vdash (\tau_2, \Sigma_2^I) \wedge (\tau_1, \Sigma_1^I) \xrightarrow{\kappa_{\Sigma}, \kappa_{\mathcal{R}}} (\tau_2, \Sigma_2^I) \end{array} \right) \end{array} \right)$$

Remarquons que la comparaison de deux modèles $\mathbb{M}_1[\rho_1]$ et $\mathbb{M}_2[\rho_2]$ nécessite la construction d’une relation de simulation pour chaque implantation correcte de $\mathbb{M}_1[\rho_1]$. Toutefois, en pratique, les relations κ_{Σ} et $\kappa_{\mathcal{R}}$ sont définies indépendamment de l’implantation considérée et il suffit alors de restreindre $\kappa_{\mathcal{R}}$ en fonction de la fonction de transition à simuler.

Application Un modèle pouvant admettre un nombre important d’implantations différentes (plus ou moins restrictives), la définition du préordre \trianglelefteq sur les modèles est difficilement utilisable directement. Toutefois, en pratique, comparer deux modèles $\mathbb{M}_1[\rho_1]$ et $\mathbb{M}_2[\rho_2]$ conduit souvent à définir un “plongement” de $\mathbb{M}_1[\rho_1]$ vers $\mathbb{M}_2[\rho_2]$ satisfaisant de “bonnes propriétés” et à partir duquel on peut définir les relations de simulation κ_{Σ} et $\kappa_{\mathcal{R}}$. Sur tous les exemples concrets que nous avons envisagés, les relations obtenues satisfont de “bonnes propriétés” : elles préservent à la fois les propriétés de sécurité, définies par les prédicats de sécurité, et la sémantique des requêtes. Dans ce cas, il est possible de montrer directement que $\mathbb{M}_1[\rho_1] \trianglelefteq \mathbb{M}_2[\rho_2]$. Une fois établi, ce résultat générique permet donc de comparer deux modèles sans avoir à considérer les implantations de ces modèles, il suffit d’étudier les propriétés des relations de simulation.

En adoptant cette approche, nous avons prouvé que le modèle de la Muraille de Chine est strictement plus restrictif que le modèle de Bell et LaPadula, qui est lui même strictement plus restrictif que le modèle RBAC à base de rôles qui est lui même équivalent au modèle HRU à base de matrices de droits d’accès.

3.3 Analyse de flots d’informations

Dans cette sous-section, nous allons voir que le cadre formel introduit pour les modèles de contrôle d’accès permet de considérer ces modèles en termes de flots d’information qu’ils permettent. En effet, une politique de contrôle d’accès permet de spécifier quels sont les accès autorisés lorsque le système se trouve dans un certain état mais ne permet pas, tout du moins de manière explicite, de spécifier les flots d’information qui sont autorisés durant la vie du système. Ainsi, une fois qu’un sujet a pu accéder à un objet, il n’y a généralement aucun contrôle sur la propagation de l’information lue par le sujet. Par exemple, avec un modèle discrétionnaire, il est possible qu’un sujet s_1 lise un objet o_1 et recopie l’information lue dans un objet o_2 accessible en lecture par un sujet s_2 non autorisé à lire o_1 . La politique de contrôle d’accès est ici respectée mais ne coïncide

pas avec son interprétation en termes de flots d'information. La cohérence entre ces deux lectures sémantiques des modèles de contrôle d'accès permet d'exprimer la correspondance entre les flots engendrés par les exécutions des implantations d'un modèle de contrôle d'accès et les politiques de confidentialité et d'intégrité induites par la politique de contrôle d'accès. Lorsque cette propriété de cohérence n'est pas vérifiée, il peut être utile de compléter le mécanisme de contrôle d'accès à l'aide d'un mécanisme de détection de flots.

Flots et Politiques de flots Nous étudions ici les flots d'information qui se produisent entre les entités du système. Nous distinguons plusieurs sortes de flots.

- Nous notons \hookrightarrow^{OO} le produit cartésien $\mathcal{O} \times \mathcal{O}$. Un élément $o_1 \hookrightarrow^{OO} o_2$ de $\mathcal{O} \times \mathcal{O}$ permet d'exprimer que le contenu d'un objet o_1 est propagé dans un objet o_2 . Nous caractériserons par la suite plusieurs sous-ensembles de \hookrightarrow^{OO} permettant de décrire des flots d'information entre objets dans différents contextes. Une *politique de confinement* est un sous-ensemble \rightsquigarrow^{OO} de \hookrightarrow^{OO} .
- Nous notons \hookrightarrow^{OS} le produit cartésien $\mathcal{O} \times \mathcal{S}$. Un élément $o \hookrightarrow^{OS} s$ de $\mathcal{O} \times \mathcal{S}$ permet d'exprimer qu'un sujet s prend connaissance des informations contenues dans un objet o . Nous caractériserons par la suite plusieurs sous-ensembles de \hookrightarrow^{OS} permettant de décrire des flots d'information des objets vers les sujets dans différents contextes. Une *politique de confidentialité* est un sous-ensemble \rightsquigarrow^{OS} de \hookrightarrow^{OS} .
- Nous notons \hookrightarrow^{SO} le produit cartésien $\mathcal{S} \times \mathcal{O}$. Un élément $s \hookrightarrow^{SO} o$ de $\mathcal{S} \times \mathcal{O}$ permet d'exprimer qu'un sujet s propage les informations dont il dispose dans un objet o . Nous caractériserons par la suite plusieurs sous-ensembles de \hookrightarrow^{SO} permettant de décrire des flots d'information des sujets vers les objets dans différents contextes. Une *politique d'intégrité* est un sous-ensemble \rightsquigarrow^{SO} de \hookrightarrow^{SO} .

Les flots d'information créés durant la vie d'un système sont caractérisés à partir d'une séquence d'états décrivant les états successifs du système. A partir de l'ensemble des accès courants qui ont lieu lorsque le système se trouve dans un état σ , nous pouvons définir les flots d'information entre objets comme suit. Le contenu d'un objet o_1 est diffusé dans un objet o_2 s'il existe un ensemble de sujets dont les accès sur les objets du système permettent de recopier l'information de o_1 dans o_2 , cette information pouvant transiter par des objets intermédiaires :

$$\hookrightarrow_{\sigma}^{OO} = \left\{ o_1 \hookrightarrow^{OO} o_2 \mid \left\{ \begin{array}{l} \exists s_1, \dots, s_k, s_{k+1} \in \mathcal{S} \exists o^1, \dots, o^k \in \mathcal{O} \\ \left(\begin{array}{l} (s_1, o_1, \text{read}), (s_1, o^1, \text{write}), \\ (s_2, o^1, \text{read}), (s_2, o^2, \text{write}), \\ \dots, \\ (s_i, o^{i-1}, \text{read}), (s_i, o^i, \text{write}), \\ \dots, \\ (s_{k+1}, o^k, \text{read}), (s_{k+1}, o_2, \text{write}) \end{array} \right) \subseteq \Lambda(\sigma) \end{array} \right\} \right\}$$

Cette définition est illustrée sur la partie gauche de la figure 3. Nous pouvons

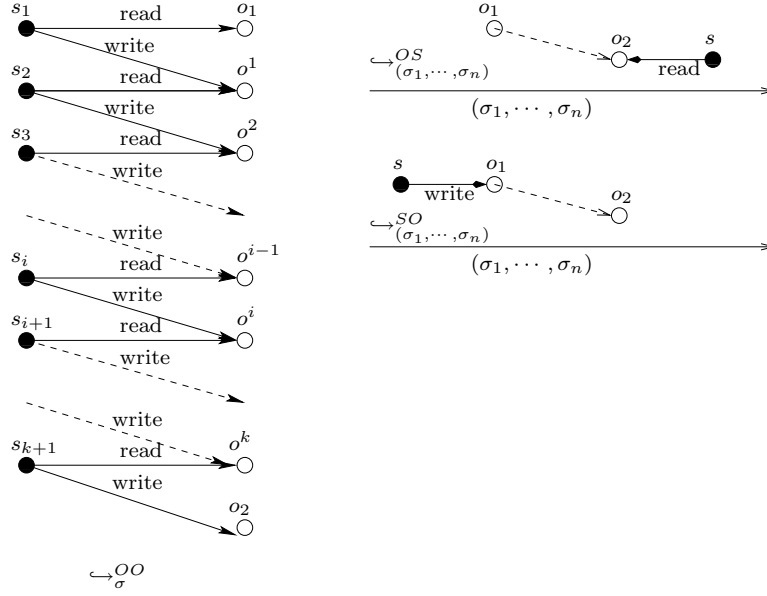


Fig. 3. Flots d'information

à présent définir les flots d'information qui ont lieu durant une séquence d'états $(\sigma_1, \dots, \sigma_n)$. Les flots entre objets sont obtenus par composition des flots engendrés par chacun des états apparaissant dans la séquence :

$$\hookrightarrow_{(\sigma_1, \dots, \sigma_n)}^{OO} = \begin{cases} \hookrightarrow_{\sigma_1}^{OO} & \text{si } n = 1 \\ \hookrightarrow_{\sigma_{k+1}}^{OO} \circ \hookrightarrow_{(\sigma_1, \dots, \sigma_k)}^{OO} & \text{si } n = k + 1 \end{cases}$$

Les flots d'un objet o vers un sujet s sont identifiés lorsqu'un objet o' a reçu l'information contenue dans o et que s accède ensuite en lecture à o' :

$$\hookrightarrow_{(\sigma_1, \dots, \sigma_n)}^{OS} = \bigcup_{i=1}^n \left\{ o_2 \hookrightarrow^{OS} s \mid o_2 \hookrightarrow_{(\sigma_1, \dots, \sigma_i)}^{OO} o_1 \wedge (s, o_1, \text{read}) \in \Lambda(\sigma_i) \right\}$$

Enfin, les flots d'un sujet s vers un objet o sont identifiés lorsque s accède en écriture à un objet o' dont le contenu est ensuite diffusé dans o :

$$\hookrightarrow_{(\sigma_1, \dots, \sigma_n)}^{SO} = \bigcup_{i=1}^n \left\{ s \hookrightarrow^{SO} o_2 \mid (s, o_1, \text{write}) \in \Lambda(\sigma_i) \wedge o_1 \hookrightarrow_{(\sigma_i, \sigma_{i+1}, \dots, \sigma_n)}^{OO} o_2 \right\}$$

Ces définitions, illustrées sur la partie droite de la figure 3, sont étendues aux ensembles de séquences d'états. Si $E \subseteq \Sigma$ est un ensemble d'états et $F \subseteq E^*$ est un ensemble de séquences d'états, on définit :

$$\hookrightarrow_F^X = \bigcup_{s \in F} \hookrightarrow_s^X \quad \text{où } X \in \{OO, OS, SO\}$$

Une politique de contrôle d'accès $\mathbb{P}[\rho] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma, \Omega)$ peut être interprétée par une politique de confidentialité et/ou une politique d'intégrité. Exprimées en termes de flots d'information, ces politiques sont définies comme suit :

$$\begin{aligned}\rightsquigarrow_{\mathbb{P}[\rho]}^{OS} &= \{o \hookrightarrow^{OS} s \mid \exists \sigma \in \Sigma_{|\Omega} (s, o, \text{read}) \in \Lambda(\sigma)\} \\ \rightsquigarrow_{\mathbb{P}[\rho]}^{SO} &= \{s \hookrightarrow^{SO} o \mid \exists \sigma \in \Sigma_{|\Omega} (s, o, \text{write}) \in \Lambda(\sigma)\}\end{aligned}$$

Ainsi, $o \rightsquigarrow_{\mathbb{P}[\rho]}^{OS} s$ signifie que pour un état sûr du système, le sujet s accède à l'information contenue dans l'objet o et $s \rightsquigarrow_{\mathbb{P}[\rho]}^{SO} o$ signifie que pour un état sûr du système, le sujet s propage l'information dont il dispose dans l'objet o .

En notant $Exec(I)$ l'ensemble des séquences d'états engendrées par l'implantation I d'un modèle $\mathbb{M}[\rho] = (\mathbb{P}[\rho], \mathcal{R})$, il est à présent possible d'exprimer formellement la propriété de cohérence introduite plus haut comme suit :

$$\hookrightarrow_{Exec(I)}^{OS} \subseteq \rightsquigarrow_{\mathbb{P}[\rho]}^{OS} \wedge \hookrightarrow_{Exec(I)}^{SO} \subseteq \rightsquigarrow_{\mathbb{P}[\rho]}^{SO}$$

Il a été prouvé que cette propriété de cohérence est vérifiée pour les politiques de la Muraille de Chine et de Bell et LaPadula (qui peuvent donc être vues comme de véritables politiques de flots). En revanche, cette propriété n'est pas vérifiée pour les modèles HRU et RBAC (qui sont des politiques libres). Toutefois, pour ces deux modèles, les propriétés permettant de garantir la cohérence ont été caractérisées : elles portent sur les informations de sécurité propres à ces modèles (matrice de droits d'accès, hiérarchie de rôles, permissions).

Mécanismes de détection de flots Lorsque la propriété de cohérence n'est pas satisfaite, on peut adjoindre un mécanisme de détection de flots au mécanisme de contrôle d'accès. Etant donné un ensemble de flots $\rightsquigarrow_{\mathbb{F}}$, détecter les flots appartenant à $\rightsquigarrow_{\mathbb{F}}$ lors de la vie d'un système consiste à observer les séquences d'états du système et à collecter des informations permettant d'identifier celles d'entre elles qui engendrent des flots recherchés. Chaque état σ est donc associé à une information $\psi(\sigma)$ qui rend compte d'une partie du passé de cet état. C'est à partir de cette information que l'on peut définir un prédicat \mathcal{U} sur Σ permettant de caractériser les états issus de séquences d'états engendrant un flot appartenant à $\rightsquigarrow_{\mathbb{F}}$. Ces états sont appelés des états d'alerte et on note $\Sigma_{|\mathcal{U}}$ l'ensemble $\{\sigma \in \Sigma \mid \mathcal{U}(\sigma)\}$. Etant donné un ensemble E d'états observables et un ensemble $F \subseteq E^*$ de séquences d'états qui peuvent se produire, un mécanisme de détection de flots est défini par :

$$\mathbb{F}[\rho, E, F, \rightsquigarrow_{\mathbb{F}}] = (\Sigma, \mathcal{U})$$

Bien sûr, il faut s'assurer que le prédicat \mathcal{U} caractérise bien les états issus d'une séquence produisant un flot dans $\rightsquigarrow_{\mathbb{F}}$. Pour cela, nous introduisons les deux propriétés classiques suivantes. $\mathbb{F}[\rho, E, F, \rightsquigarrow_{\mathbb{F}}] = (\Sigma, \mathcal{U})$ est :

- *pertinent* ssi tout état d'alerte est issu d'une séquence engendrant un flot dans $\rightsquigarrow_{\mathbb{F}}$:

$$\forall (\sigma_1, \dots, \sigma_n) \in F \quad \mathcal{U}(\sigma_n) \Rightarrow \hookrightarrow_{(\sigma_1, \dots, \sigma_n)}^X \cap \rightsquigarrow_{\mathbb{F}} \neq \emptyset$$

- *fiable* ssi tout état issu d'une séquence engendrant un flot dans $\rightsquigarrow_{\mathbb{F}}$ est un état d'alerte :

$$\forall(\sigma_1, \dots, \sigma_n) \in F \quad \hookrightarrow_{(\sigma_1, \dots, \sigma_n)}^X \cap \rightsquigarrow_{\mathbb{F}} \neq \emptyset \Rightarrow \mathcal{U}(\sigma_n)$$

où $X \in \{OO, OS, SO\}$.

Ce formalisme a été utilisé avec succès pour formaliser le mécanisme de détection d'intrusions pour le modèle HRU défini dans [11] ainsi que les preuves de fiabilité et de pertinence de ce mécanisme.

4 Conclusion – Perspectives

La sécurité, et plus particulièrement le contrôle d'accès, sont des problématiques actuelles en informatique. En effet, il devient aujourd'hui important de pouvoir contrôler les flots d'information dans les réseaux et dans les systèmes d'information. Il convient donc de développer au sein des systèmes informatiques des mécanismes permettant de filtrer les accès afin de ne laisser passer que ceux autorisés. La conception et le développement de ces mécanismes doivent être menés de manière à garantir leur fiabilité et leur sûreté. L'emploi des méthodes formelles dans le développement d'un moniteur de référence permet de garantir que certaines propriétés de sécurité sont toujours respectées. Dans cet article nous avons rendu compte de manière informelle de quelques expériences formelles que nous avons faites dans cette direction. L'ensemble de ces travaux a permis de définir un cadre formel pour les systèmes de contrôle d'accès. Il fournit un guide méthodologique lors de la conception d'un modèle et peut aider un utilisateur à adopter une démarche rigoureuse lors de son développement. De plus, exprimer des modèles au sein d'un même cadre permet de les comparer et d'obtenir une compréhension plus fine des propriétés souhaitées pour le modèle en cours d'élaboration.

Les perspectives de ces travaux sont multiples. Les modèles étudiés jusqu'à présent expriment des propriétés de sécurité sur les accès autorisés au sein d'un système. Toutefois, ils fournissent un langage de requêtes qui permet l'ajout ou le retrait d'un accès mais qui permet également de modifier les configurations de sécurité du système (requêtes administratives). Un axe de recherche à développer concerne donc l'étude des politiques administratives permettant de régir les modifications des informations de sécurité dynamiques. Cette étude pourrait être le point de départ pour envisager la notion de méta-politiques, permettant de régir les changements d'une politique de sécurité dynamique. De telles politiques pourraient être utilisées avec profit pour maintenir des propriétés de sécurité dans un contexte distribué (chaque site étant régi par une politique dynamique dont l'évolution est régie par une méta-politique).

L'étude de la comparaison de modèles de contrôle d'accès est un premier pas vers l'étude de la composition de ces modèles. Cette problématique mérite aussi d'être développée puisqu'elle correspond à un problème concret très répandu dans les systèmes d'information. En effet, dans la plupart de ces systèmes, un sujet accède généralement à un objet en passant au travers de plusieurs filtres. Par exemple, dans certains systèmes, les séquences d'accès peuvent être

régies par deux politiques : chaque accès est régi par une politique à base de rôles (RBAC), et le séquençement des accès est régi par une politique à base de tâches (TBAC). Il existe évidemment plusieurs procédés pour composer des politiques de sécurité mais il y a peu d'études sur les propriétés de ces compositions. Cette problématique mérite aussi d'être considérée afin de formaliser les concepts liés à la composition pour pouvoir non seulement identifier certains opérateurs génériques de composition, mais aussi pour exprimer les propriétés de sécurité que les mécanismes de composition envisagés peuvent garantir. Les mécanismes de comparaison déjà introduits pourraient s'appliquer pour définir certaines de ces propriétés.

Remerciements. Je remercie vivement Lionel Habib, Thérèse Hardin, Ludovic Mé, Charles Morisset et Valérie Viet Triem Tong avec qui j'ai la chance de travailler sur le domaine du contrôle d'accès et qui ont contribué à ce travail.

References

1. P. Amey. Dear sir, yours faithfully: an everyday story of formality. In *Practical Elements of Safety, Proc. of the Twelfth Safety-critical Systems Symposium*. Springer-Verlag, 2004.
2. F. Anseaume, J. Baron, P. Berthelin, M. Jacquelin, and D. Pison. Formalisation, spécification et implantation de politiques de contrôle d'accès avec l'atelier Focal. Master's thesis, UPMC, Paris, France, 2008.
3. D. Bell and L. LaPadula. Secure Computer Systems: a Mathematical Model. Technical Report MTR-2547 (Vol. II), MITRE Corp., Bedford, MA, May 1973.
4. J. Blond and C. Morisset. Un moniteur de référence sûr d'une base de données. *Technique et Science Informatiques*, 26(9):1091–1110, 2007.
5. F. Brecht and A.D. Kadja. Implantation d'une politique de contrôle d'accès discrétionnaire avec Focal. Master's thesis, UPMC, Paris, France, 2007.
6. D. F. C. Brewer and M. J. Nash. The chinese wall security policy. In *Proc. IEEE Symposium on Security and Privacy*, pages 206–214, 1989.
7. M. Carlier and C. Dubois. Functional testing in the Focal environment. In B.Beckert and R.Hähnle, editors, *Tests and Proofs, Second International Conference, TAP 2008, Prato, Italy, April 9-11, 2008. Proceedings*, volume 4966 of *Lecture Notes in Computer Science*, pages 84–98. Springer, 2008.
8. A. Santana de Oliveira. *Réécriture et Modularité pour les Politiques de Sécurité*. Phd thesis, Université Henri Poincaré, 2008.
9. C. Dubois, T. Hardin, and V. Vigié Donzeau Gouge. Building certified components within Focal. In *Symposium on Trends in Functional Programming*, 2004.
10. D. F. Ferraiolo and D. R. Kuhn. Role-based access control. In *Proceedings of the 15th National Computer Security Conference*, 1992.
11. G.Hiet, L.Mé, J.Zimmermann, C.Bidan, B.Morin, and V. Viet Triem Tong. Détection fiable et pertinente de flux d'information illégaux. In *6th Conference on Security and Network Architectures (SARSSI)*, 2007.
12. E. Gureghian, Th. Hardin, and M. Jaume. A full formalisation of the Bell and Lapadula security model. Technical Report 2003-007, Univ. Paris 6, LIP6, 2003.
13. L. Habib. Formalisation, comparaison et implantation d'un modèle de contrôle d'accès à base de rôles. Master's thesis, UPMC, Paris, France, 2007.

14. L. Habib, M. Jaume, and C. Morisset. A formal comparison of the Bell & LaPadula and RBAC models. In *Fourth International Symposium on Information Assurance and Security IAS'08*, pages 3–8. IEEE CS Press, 2008.
15. T. Hardin, M. Jaume, and C. Morisset. Access control and rewrite systems. In *1st International Workshop on Security and Rewriting Techniques, SecRet'06 (Satellite Workshop to ICALP'2006)*, 2006.
16. M. Harrison, W. Ruzzo, and J. Ullman. Protection in operating systems. *Communications of the ACM*, 19:461–471, 1976.
17. M. Jaume. *Descriptions formelles - Application au contrôle d'accès*. Habilitation à diriger des recherches, Université Paris 6, 2008.
18. M. Jaume and C. Morisset. Formalisation and implementation of access control models. In *Information Assurance and Security (IAS'05) International Conference on Information Technology, ITCC*, pages 703–708. IEEE CS Press, 2005.
19. M. Jaume and C. Morisset. A formal approach to implement access control. *Journal of Information Assurance and Security*, 2:137–148, 2006.
20. M. Jaume and C. Morisset. Towards a formal specification of access control. In P. Degano, R. Kusters, L. Vigano, and S. Zdancewic, editors, *Proceedings of the Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis, FCS-ARSPA'06*, pages 213–232, 2006.
21. M. Jaume and C. Morisset. Contrôler le contrôle d'accès : Approches formelles. In *Approches Formelles dans l'Assistance au Développement de Logiciels, AFADL'07*, 2007.
22. M. Jaume and C. Morisset. Un cadre sémantique pour le contrôle d'accès. *Technique et Science Informatiques*, 27(8):951–976, 2008.
23. L.J. LaPadula and D.E. Bell. Secure Computer Systems: A Mathematical Model. *Journal of Computer Security*, 4:239–263, 1996.
24. H.M. Levy. *Capability-Based Computer Systems*. Digital Press, Bedford, MA, 1984.
25. McLean. The algebra of security. In *Proc. IEEE Symposium on Security and Privacy*, pages 2–7. IEEE Computer Society Press, 1988.
26. C. Morisset. *Sémantique des systèmes de contrôle d'accès*. PhD thesis, Université Pierre et Marie Curie, 2007.
27. C. Morisset and A. Santana de Oliveira. Automated detection of information leakage in access control. In *Proceedings of the 2nd International Workshop on Security and Rewriting Techniques (SecReT'07)*, Paris, France, 2007.
28. Focal project. *Focal, version 0.3.1 Tutorial and reference manual*. LIP6 – INRIA – CNAM, sept 2006. Distribution available at: <http://focal.inria.fr>.
29. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.