

Contrôler le contrôle d'accès : Approches formelles

Mathieu Jaume and Charles Morisset

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. Ne pas apporter de garanties fortes sur la correction de tels programmes reviendrait à construire un château fort avec une porte en papier. 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, implanter et comparer des politiques de contrôle d'accès.

Keywords: 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 [3], 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 dans certains cas 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, ...). Dans ce cadre, on distingue essentiellement trois propriétés : la confidentialité, qui assure que les données ne sont lues que par les personnes autorisées, l'intégrité, qui assure que les données ne sont modifiées que par les personnes autorisées et la disponibilité, qui assure que les données sont accessibles aux personnes autorisées.

La disponibilité semble être la propriété la plus difficile à garantir, car elle ne dépend pas uniquement du système en question, mais également de son environnement et de ses ressources (par exemple, il suffit de couper l'alimentation électrique d'un serveur pour que celui-ci ne soit plus disponible).

Deux procédés sont habituellement utilisés pour assurer la confidentialité et l'intégrité : l'utilisation de protocoles cryptographiques pour les communications au sein du système, ainsi que depuis ou vers l'extérieur, et la mise en place d'un moniteur de référence qui va gérer au sein du système l'accès aux données. Les mécanismes de ces deux approches sont différents (chacun est plus ou moins sensible à certaines attaques) :

- la cryptographie assure la protection *a posteriori* : n'importe qui peut accéder aux données, mais seuls les utilisateurs possédant la clé peuvent les lire ou les modifier,
- un moniteur de référence assure la protection *a priori* : seuls les utilisateurs autorisés peuvent accéder à l'information.

Dans cet article nous nous intéressons à l'approche "moniteur de référence" pour gérer les accès effectués dans un système. Les Critères Communs (recueil de normes définies par des agences gouvernementales) fournissent une méthodologie permettant d'atteindre des hauts niveaux de sécurité. Ils définissent à la fois un cadre de travail pour la conception et la réalisation de logiciels et une référence pour les utilisateurs de ces logiciels. Les hauts niveaux de sûreté des Critères Communs [1] (EAL 5 à 7) requièrent l'utilisation de méthodes formelles dans les étapes de spécification et de conception du logiciel. Selon les Critères Communs, un système est vu comme une installation donnée de technologies de l'information, avec un objectif et un environnement opérationnel particuliers. Une politique de sécurité est un ensemble de règles qui précisent comment gérer, protéger ou distribuer les informations ou ressources du système. Un moniteur de référence est une machine abstraite qui applique les politiques de contrôle d'accès d'un système, ces politiques étant un sous-ensemble des politiques de sécurité. Toujours selon ces mêmes Critères Communs, un moniteur de référence doit posséder les trois caractéristiques suivantes :

- Des sujets non sûrs ne peuvent pas interférer avec son fonctionnement, i.e. il est à l'épreuve d'une intrusion physique.
- Des sujets non sûrs ne peuvent pas court-circuiter les contrôles qu'il effectue, i.e. il est systématiquement appelé.
- Il est suffisamment simple pour être analysé et pour comprendre son comportement, i.e. sa conception est simple.

Ces trois caractéristiques, introduites dans [4], sont connues sous l’acronyme *NEAT*, pour “*Non-bypassable*” (il n’est pas possible d’éviter les fonctions de sécurité), “*Evaluatable*” (les fonctions de sécurité sont suffisamment simples pour être mathématiquement vérifiées et évaluées), “*Always Invoked*” (les fonctions de sécurité sont tout le temps appelées) et “*Tamperproof*” (les fonctions de sécurité ne peuvent pas être altérées). Cet acronyme est défini dans le cadre de *MILS* (*Multiple Independent Levels of Security*), une approche de développement de systèmes sécurisés (<http://www.ois.com/MILS/>).

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’explicitement totalement le système à modéliser. Cette difficulté est illustrée dans la section 2.

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. La section 3 présente deux développements formels de politiques de contrôle d’accès.

Quoi qu’il en soit, conduire un développement formel de cette nature est une activité chronophage en temps. 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. La section 4 décrit les grandes lignes d’un tel cadre.

La suite de cet article illustre ces difficultés en considérant deux politiques classiques de contrôle d’accès :

- la politique de Bell et LaPadula [18, 5], initialement conçue pour les militaires, repose sur un ensemble partiellement ordonné de niveaux de sécurité associés aux objets et aux sujets (il s’agit plus précisément d’un treillis);
- la politique de la Muraille de Chine [8], permettant d’éviter les conflits d’intérêt et conçue pour le monde des consultants, repose sur un partitionnement des objets.

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 [14–17].

2 Formaliser pour mieux comprendre

Des détails qui n’en sont pas ...

Le passage de l’informel au formel permet souvent de détecter des erreurs. Nous présentons ici un exemple de manque de précision dans la formalisation d’une politique de contrôle d’accès qui à première vue peut sembler sans conséquence mais qui permet finalement de violer la politique de sécurité modélisée.

La politique de Bell et LaPadula est habituellement décrite par une machine à états. 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 un tuple (m, D, f_s, f_o) où m est l’ensemble des accès courants, 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 (resp. de D) sont des accès représentés par des triplets de la forme (s, o, a) exprimant qu’un sujet s accède (resp. 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 simplement que tout accès courant est conforme aux droits d’accès. Plus formellement, cette propriété s’écrit :

$$m \subseteq D$$

- Propriétés MAC et MAC* (*Mandatory Access Control*) :
 - La propriété MAC connue sous le nom de “*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. Plus formellement, cette propriété s’écrit :

$$(s, o, \text{read}) \in m \Rightarrow f_s(s) \succeq f_o(o)$$

- La propriété MAC* connue sous le nom de “*no write-down property*” permet d’éviter qu’un sujet “malicieux” recopie de l’information sensible à un niveau de sécurité inférieur. Plus formellement, cette propriété s’écrit :

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

Dans [19], McLean introduit une “algèbre de sécurité” à partir de laquelle il formalise une version enrichie de la politique de Bell et LaPadula, essentiellement

en considérant la notion d'accès conjoints. Cette notion vient du fait que dans certains systèmes, il existe des opérations qui doivent être effectuées par plusieurs sujets en même temps pour pouvoir être autorisées. L'exemple le plus classique vient du monde militaire, où pour lancer un missile, il faut que des personnes habilitées appuient sur un bouton en même temps. La propriété de sécurité MAC* est alors (re)définie comme suit :

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

Ici, 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 la spécification suivante :

$$((S_1, o_1, read) \in m \wedge (S_2, o_2, write) \in m \wedge f_o(o_2) \prec f_o(o_1)) \Rightarrow S_1 \cap S_2 = \emptyset$$

Par contraposition, on obtient finalement :

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

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, les deux propriétés (1) et (2) ne sont pas équivalentes. En effet, pour que ces deux propriétés soient équivalentes il faudrait que l'ordre sur les niveaux de sécurité soit total. Or, il ne s'agit que d'un ordre partiel puisque l'ensemble des niveaux de sécurité est muni d'une structure de treillis. Malheureusement, la propriété (2) n'exprime alors pas la propriété souhaitée comme l'illustre l'exemple suivant. En effet, comme le montre la figure 1, si l'on se contente de respecter la propriété (2), il est possible qu'un sujet s_1 accède simultanément en lecture à un objet dont le niveau de sécurité est l_1 et en écriture à un objet dont le niveau de sécurité est l_2 tel 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 aussi être découverts en utilisant des méthodes de test (par exemple en testant 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).

Objets informels et définitions formelles

La politique de la Muraille de Chine [8] a été créée pour résoudre les problèmes de conflit d'intérêt dans le monde des consultants. Chaque objet du système

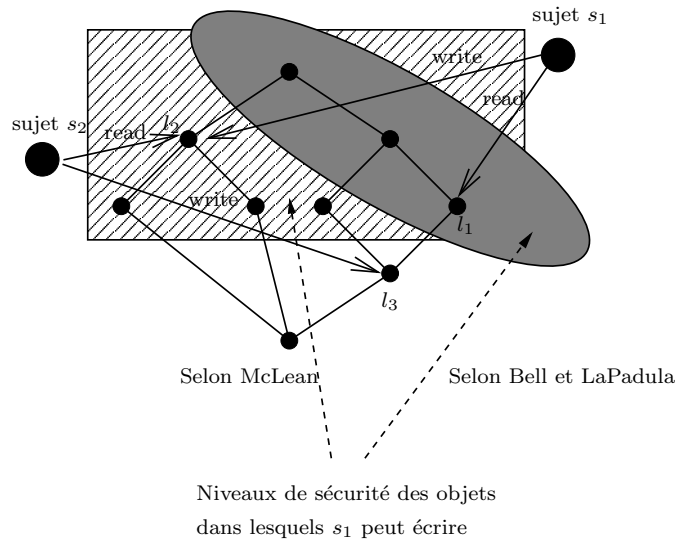


Fig. 1. Violation de la politique de sécurité

appartient à un ensemble de données d’une compagnie, et chaque compagnie appartient à une unique classe de conflit d’intérêt. Ces classes correspondent à des milieux professionnels distincts, comme par exemple les banques ou les compagnies pétrolières. La politique de sécurité consiste à dire qu’un consultant peut travailler en même temps pour une banque et une compagnie pétrolière, mais pas pour deux banques ou deux compagnies pétrolières. A cela il faut ajouter l’existence d’une classe de conflit spéciale qui ne contient qu’une seule compagnie et qui contient les informations “sanitisées”, c’est-à-dire qu’elles 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. La première est la règle de sécurité-simple :

[8] “Access is only granted if the object requested: (a) is in the same company dataset as an object already accessed by that subject, i.e. within the Wall, or (b) belongs to an entirely different conflict of interest class.”

La deuxième est celle de sécurité- \star :

[8] “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.”

Ces deux propriétés sont volontairement répétées en anglais, afin de permettre au lecteur de mieux percevoir le niveau de formalisation de la description originale de cette politique. Ces deux propriétés sont difficilement compréhensibles à

la première lecture et également difficiles à formaliser. En effet, quel est le statut exact d'un objet qui "peut être lu" ? Est ce un objet que le sujet est en train de lire ? ou bien un objet qu'il lira dans l'avenir ? ou encore est ce que cela signifie qu'il a les droits pour le faire ? Dans ce dernier cas, cela signifie que lorsqu'un sujet veut écrire dans un objet, il doit révoquer tous les droits qu'il a sur des objets appartenant à une autre classe de conflit. D'autre part, dans l'article original, il n'est mentionné nulle part que les accès peuvent être relâchés. Est ce que cela signifie qu'il n'est pas possible de le faire, et qu'un accès est éternel ? Dans ce cas, le modèle de la Muraille de Chine est très restrictif, puisqu'une fois qu'un sujet a écrit dans un objet, il ne peut plus avoir d'accès en lecture ou en écriture à un objet d'une classe de conflit différente. Pour formaliser ces deux propriétés de sécurité, il faut faire des choix quant à l'interprétation de ces propriétés. Le danger provient de ces choix qui permettent plusieurs formalisations non équivalentes de cette politique.

Enfin, l'étape de formalisation permet de distinguer clairement les propriétés de sécurité souhaitées du moniteur de référence chargé de les faire respecter. En effet, avec la présentation informelle de la politique de la Muraille de Chine, il est tentant de penser que la description des propriétés de sécurité constitue une description algorithmique de la fonction d'autorisation des accès ... ce qui n'est pas le cas.

3 Mécaniser la formalisation

3.1 Formalisation de la politique de Bell et LaPadula avec Coq

Nous considérons ici la formalisation complète d'une politique de contrôle d'accès, celle de Bell et LaPadula [18, 5], à l'aide du système Coq [23], une implantation d'un λ -calcul typé avec types dépendants et types inductifs permettant d'obtenir des preuves formelles de manière interactive.

Ce développement est décrit en détail dans [14] et nous n'en esquissons ici que les grandes lignes. Il est paramétré par un ensemble \mathcal{S} de sujets, un ensemble \mathcal{O} d'objets, un ensemble \mathcal{A} de modes d'accès et un treillis fini $(\mathcal{L}, \preceq, \gamma, \wedge)$ de niveaux de sécurité. 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 notion d'états du système, 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, $\mathcal{D} = \{\text{yes, no}\}$ contient les réponses possibles et Σ est l'ensemble des états. Une fonction de transition τ correspond à la mise en application d'une politique de sécurité et sera qualifiée de fonction "sûre" si et seulement si elle préserve les propriétés de sécurité (i.e. ssi elle transforme chaque état vérifiant les propriétés DAC, MAC et MAC* en états vérifiant encore ces propriétés). Le

développement obtenu 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*” [5] 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 développement nous a permis de spécifier, implanter et raisonner dans un cadre formel et d’atteindre un niveau de confiance élevé sur le programme obtenu. Toutefois, si elle remplit ses objectifs en termes de garantie de correction, une telle approche peut conduire en revanche 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 souhaitable de se placer dans un cadre formel caractérisant les éléments communs aux politiques de contrôle d’accès et permettant d’en dériver différentes implantations. Il est d’autre part préférable d’utiliser un environnement de développement qui facilite l’implantation de ces différents traits. Le système Coq dispose à présent de mécanismes permettant de conduire des développements formels de manière modulaire, mais nous avons choisi par la suite d’utiliser l’atelier Focal développé au sein de notre équipe. Le paragraphe suivant présente une expérience suivant cette approche.

3.2 Développement de la politique de Bell et LaPadula vue comme une instance de l’algèbre de McLean avec l’atelier Focal

Afin d’éviter les inconvénients du développement présenté dans la section 3.1, nous avons choisi de définir, dans l’environnement de développement Focal [22, 12], l’“algèbre de sécurité” introduite par McLean [19]. Ce cadre générique pour le contrôle d’accès 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 [15, 16, 7].

Implantation de l’“algèbre de sécurité” de McLean avec Focal

L’atelier Focal [22, 12, 20], et la méthodologie sous-jacente, ont initialement été appliqués au calcul formel. C’est en fait ce domaine qui a servi de modèle et donné les lignes directrices du développement de l’atelier. Grâce à l’atelier Focal, R. Rioboo a pu construire une bibliothèque de calcul formel assez conséquente [24] comprenant des algorithmes complexes et dont l’efficacité est comparable à celle des meilleurs systèmes de calcul formel. Aujourd’hui, Focal est aussi utilisé dans le domaine de la sécurité : outre les développements que nous effectuons avec Focal sur les politiques de contrôle d’accès, Focal a été utilisé avec succès pour formaliser la politique de sécurité au sol d’un aéroport [10].

Focal est un environnement de programmation basé sur un langage muni de traits objets (héritage multiple, liaison tardive, redéfinition, ...) permettant non seulement de structurer un développement de manière à le rendre facilement

réutilisable, mais aussi d’obtenir un logiciel par raffinements successifs en passant progressivement de la spécification à l’implantation. En effet, les constructions de ce langage autorisent l’écriture de spécifications, de programmes et de preuves. D’autre part, l’atelier Focal dispose à présent d’un démonstrateur automatique, Zenon, permettant d’obtenir de nombreuses preuves de manière automatique.

Toutes ces caractéristiques font de Focal un bon candidat pour implanter l’“algèbre de sécurité” introduite par McLean. Enfin, le choix de Focal est aussi motivé par notre expérience dans le domaine de la réutilisation et le “management” de preuves dans cet atelier [11, 21].

Pour profiter pleinement de la puissance des constructions de Focal en terme de réutilisabilité, nous avons choisi de ne pas développer directement telle ou telle politique de sécurité mais plutôt de considérer cette politique comme une instance d’un modèle plus général. Afin de factoriser le travail de formalisation et de développement, il est en effet souhaitable de se placer dans un cadre abstrait générique permettant de considérer chacune des politiques envisagées comme une instance. La littérature sur ce domaine est encore relativement restreinte et nous avons choisi d’implanter l’“algèbre des modèles de sécurité” introduite par J. McLean en 1988 [19]. Cette algèbre fournit un cadre dans lequel peuvent se définir certains des concepts que beaucoup de politiques de sécurité partagent. D’autre part, l’existence d’une importante bibliothèque de calcul formel [24] permet d’envisager aisément l’implantation dans Focal de cette algèbre qui fait appel à de nombreuses structures algébriques classiques (treillis, algèbres de Boole, ...).

Le cadre introduit par McLean permet la description d’une politique de contrôle d’accès à trois niveaux de spécification différents (figure 2) : les frameworks, les modèles et les systèmes.

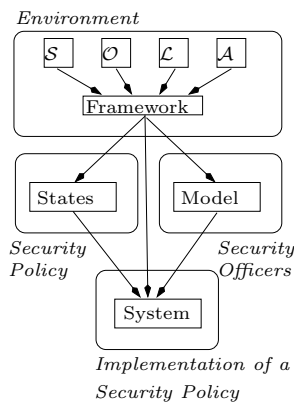


Fig. 2. Algèbre de Sécurité

Un framework décrit l'environnement : il est paramétré par un ensemble \mathcal{S} de sujets, un ensemble \mathcal{O} d'objets, un ensemble \mathcal{A} de modes d'accès, un treillis \mathcal{L} de niveaux de sécurité 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). Dans le cas le plus général, il s'agit des parties non vides de \mathcal{S} (toute opération est initiée par au moins un sujet) mais il peut exister des contraintes sur ces ensembles. Pour certaines politiques, il est par exemple spécifié que si un groupe de sujets peut conjointement soumettre une requête, alors tout sous-ensemble de ce groupe peut également soumettre une requête. Dans le cas le plus courant pour lequel les accès conjoints ne sont pas pris en compte, les seuls ensembles autorisés à soumettre une requête sont les singletons construits à partir de \mathcal{S} . Une hiérarchie de frameworks est alors définie selon les propriétés vérifiées par ces ensembles de sujets.

Introduite par J.McLean afin de pallier les problèmes posés par des systèmes de contrôle d'accès parfois trop rudimentaires, la notion de modèle permet de fixer les règles régissant le changement de niveau de sécurité d'un sujet ou d'un objet. La notion de modèle est paramétrée par celle de framework. Un modèle spécifie quels sont les ensembles de sujets qui seront effectivement autorisés à modifier les niveaux de sécurité. Ici, encore, une hiérarchie de modèles est définie selon les propriétés vérifiées par ces ensembles de sujets. Certaines relations et opérations sur les modèles sont définies et permettent d'implanter l'ensemble des modèles comme une instance de treillis distributif.

Enfin, la notion d'état d'un système est définie. Les états décrivent les informations relatives aux divers niveaux de sécurité associés aux sujets et aux objets ainsi que les accès courants. Une fois la notion d'état définie, il est possible de spécifier par un prédicat quels sont les états sûrs, c'est-à-dire quelle est la politique de sécurité appliquée.

C'est à partir de la notion d'état et de modèle qu'il est alors possible de définir la notion de système qui spécifie ce qu'est une fonction de transition et quelles sont les propriétés qu'elle doit vérifier. Une fonction de transition entre états permet de décrire les changements d'états produits lors des requêtes d'accès émises par les sujets.

En définissant, c'est-à-dire en instanciant, chacune des entités spécifiées dans le cadre obtenu, nous avons défini la politique de Bell et LaPadula et avons fourni une implantation de la fonction de transition τ_{BLP} évoquée dans la section 3.1.

Application au contrôle des accès d'une base de données

L'implantation de la politique de Bell et LaPadula obtenue dans le cadre décrit précédemment reste encore relativement générique. Par exemple, 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 des systèmes “concrets”. Il s’agit ici d’une mise à l’épreuve du terrain de la méthode toute entière.

Afin d’illustrer l’utilisation du programme Focal implantant le système de Bell et LaPadula, nous avons mis en application ce programme pour le contrôle des accès dans une base de données relationnelle (MySQL). Cette base de données 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 de Bell et LaPadula obtenu dans le paragraphe précédent 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. Deux tables particulières concernant les sujets et les objets sont présentes dans la zone dédiée aux données relatives à la sécurité de la base de données. La table `sujets` contient pour chaque sujet son identifiant unique, son login, son mot de passe, ainsi que son niveau de sécurité. De même, la table `objets` contient pour chaque objet son identifiant unique, le nom de la table à laquelle il correspond ainsi que son niveau de sécurité. Ainsi, les objets du système d’information sont les tables de la base de données. Une granularité plus fine 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. Par exemple, considérer les tuples d’une base de données comme des objets conduit à un phénomène de polyinstanciation [25] lorsque deux tuples ont la même clé primaire, mais des informations de niveaux de sécurité différents ([6] propose des solutions à ce problème).

L’architecture de l’implantation se décompose en trois parties : la politique de sécurité, le gestionnaire de base de données et le moniteur de référence. La partie concernant la politique de sécurité est entièrement définie en Focal et repose sur le modèle de Bell et LaPadula. L’accès au gestionnaire de base de données s’effectue grâce à une interface définie en Focal proposant des fonctions de connexion/déconnexion à la base de données, ainsi que des fonctions permettant l’exécution d’une requête et la récupération du résultat. Ces fonctions reposent sur la librairie `ocaml-mysql` [2], et sont donc spécifiques au gestionnaire de base de données MySQL. La figure 3 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 oeuvre de notre application, qui peut être vue comme un filtre entre l’utilisateur et le système de gestion de la base de données.

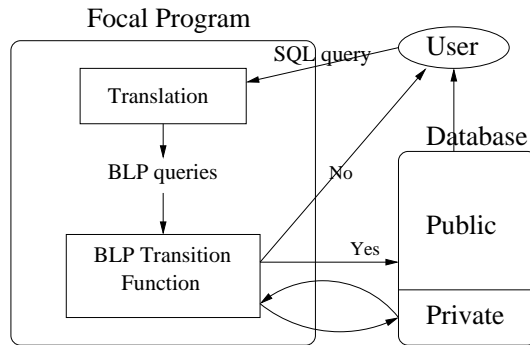


Fig. 3. Contrôle d'accès pour un SGBD

4 Vers un cadre sémantique pour le contrôle d'accès

A l'issue des travaux que nous venons de décrire, nous avons souhaité obtenir un développement certifié de la politique de la Muraille de Chine. Bien sûr, comme nous l'avons vu dans la section 2, les mêmes problèmes liés au passage de l'informel au formel sont apparus. D'autre part, lors de la formalisation de cette politique, le cadre de McLean s'est révélé à la fois trop contraignant (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) et trop peu expressif (en termes de propriétés de simulation permettant une plus grande réutilisabilité). 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 abstraite de ce qui constitue une politique de contrôle d'accès.

Afin de réutiliser le plus facilement possible le développement en Focal de la politique de Bell et LaPadula, nous souhaitons obtenir une implantation certifiée de la politique de la Muraille de Chine basée sur les travaux de R.S. Sandhu [27, 26] proposant une interprétation de la Muraille de Chine qui repose sur un treillis de niveaux de sécurité. Nous souhaitons également "comparer" cette implantation avec celle de la politique de Bell et LaPadula. Pour adopter cette approche, il est nécessaire de formaliser complètement, non seulement cette interprétation, mais aussi les notions de comparaison et d'interprétation d'une politique par une autre politique. Le cadre sémantique que nous proposons répond à ces objectifs :

- d'un point de vue pratique, il permet de comprendre certains mécanismes de réutilisation de politiques de sécurité
- d'un point de vue plus théorique, il permet d'introduire un préordre entre ces politiques.

Nous présentons ici les grandes lignes de ce cadre. Le détail d'une version préliminaire de ce cadre est décrit dans [17].

Une politique de contrôle d'accès repose sur des paramètres de sécurité (par exemple un treillis des niveaux de sécurité pour la politique de Bell et LaPadula ou des ensembles de classes de conflit et de compagnies pour la politique de la Muraille de Chine). Si l'on note ρ les paramètres de sécurité sur lesquels elle repose, une politique de contrôle d'accès $\mathbb{P}[\rho]$ est définie par un tuple :

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

où \mathcal{S} est un ensemble de sujets, \mathcal{O} est un ensemble d'objets, \mathcal{A} est un ensemble de modes d'accès, Σ est l'ensemble des états du système et Ω est un prédicat sur Σ caractérisant les états sûrs (Ω correspond à la spécification des propriétés de sécurité de la politique et on note $\Sigma_{|\Omega}$ le sous-ensemble de Σ contenant les états sûrs). Une politique de contrôle d'accès correspond essentiellement à la description de ce qu'est un état "sûr".

Dans un système, les utilisateurs (i.e. les sujets) soumettent des requêtes pour accéder et/ou modifier des informations. On note \mathcal{R} le langage de requêtes utilisé et on désigne par :

$$\|\mathcal{R}\|_{\Sigma} \subseteq \mathcal{R} \times \Sigma$$

la sémantique de ce langage. Il s'agit d'une relation permettant de décrire de manière plus ou moins fine certaines propriétés vérifiées par un état obtenu après application d'une requête. Etant donnée une requête R et un état σ , l'énoncé $(R, \sigma) \in \|\mathcal{R}\|_{\Sigma}$ permet de caractériser les propriétés qu'un état σ doit satisfaire quand il est obtenu à partir d'un état en appliquant avec succès la requête R .

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

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

Implanter un modèle de contrôle d'accès $\mathbb{M}[\rho] = (\mathbb{P}[\rho], \|\mathcal{R}\|_{\Sigma})$ 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 $\|\mathcal{R}\|_{\Sigma}$). Si l'on note $\Gamma_{\tau}(E)$ l'ensemble des états atteignables à partir des états contenus dans l'ensemble E par application de la fonction de transition τ , une implantation (τ, Σ_I) est dite :

- $\mathbb{P}[\rho]$ -correcte ssi chaque état atteignable est sûr : $\Gamma_{\tau}(\Sigma_I) \subseteq \Sigma_{|\Omega}$
- $\|\mathcal{R}\|_{\Sigma}$ -correcte ssi :

$$\forall \sigma_1, \sigma_2 \in \Sigma \quad \forall R \in \mathcal{R} \quad \tau(R, \sigma_1) = (\text{yes}, \sigma_2) \Rightarrow (R, \sigma_2) \in \|\mathcal{R}\|_{\Sigma}$$

D'autres propriétés permettent de caractériser plus finement les implantations d'une politique de contrôle d'accès en spécifiant les liens qui existent entre les

états accessibles selon les types de requêtes qui permettent de passer de l'un à l'autre.

Un modèle admet plusieurs implantations correctes. Par exemple, la fonction de transition qui retourne (no, σ) pour tout état σ et toute requête est une implantation trivialement correcte (dont l'intérêt est cependant assez limité). Aussi, il peut être intéressant de comparer les implantations d'une même politique afin, par exemple, de déterminer si une implantation est plus restrictive qu'une autre. On définit un préordre sur les implantations : intuitivement une implantation $I = (\tau, \Sigma_I)$ est plus restrictive qu'une implantation $I' = (\tau', \Sigma'_I)$, ce que l'on note $I \sqsubseteq I'$, ssi les états accessibles avec I le sont également avec I' et si I' permet de faire de "plus petits pas" que I et d'agir avec une granularité plus fine sur les états (la définition formelle de cette condition nécessite la définition d'un préordre sur les états que nous ne détaillons pas ici).

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. Ces concepts reposent sur la notion classique de relation de simulation. Etant donnés deux modèles de contrôle d'accès $\mathbb{M}_1[\rho_1] = (\mathbb{P}_1[\rho_1], \|\mathcal{R}\|_{\Sigma_1})$ et $\mathbb{M}_2[\rho_2] = (\mathbb{P}_2[\rho_2], \|\mathcal{R}\|_{\Sigma_2})$ (partageant les mêmes ensembles de sujets, d'objets et de modes d'accès), l'implantation $I_2 = (\tau_2, \Sigma_2^I)$ de $\mathbb{M}_2[\rho_2]$ simule l'implantation $I_1 = (\tau_1, \Sigma_1^I)$ de $\mathbb{M}_1[\rho_1]$, ce que l'on note $I_1 \stackrel{\kappa_{\Sigma}}{\sim} I_2$, ssi il existe une relation $\kappa_{\Sigma} \subseteq \Sigma_1 \times \Sigma_2$ qui relie tout état initial de I_1 à au moins un état initial de I_2 et qui soit une relation de simulation de τ_1 par τ_2 , c'est-à-dire :

$$\begin{aligned} & \forall \sigma_1, \sigma'_1 \in \Sigma_1 \quad \forall \sigma_2 \in \Sigma_2 \quad \forall R \in \mathcal{R} \quad \forall a \in \mathcal{D} \\ & ((\sigma_1, \sigma_2) \in \kappa_{\Sigma} \wedge \tau_1(R, \sigma_1) = (a, \sigma'_1)) \\ & \Rightarrow (\exists \sigma'_2 \in \Sigma_2 \quad (\sigma'_1, \sigma'_2) \in \kappa_{\Sigma} \wedge \tau_2(R, \sigma_2) = (a, \sigma'_2)) \end{aligned}$$

Interpréter un modèle de contrôle d'accès par un autre consiste alors à définir une relation de simulation satisfaisant de bonnes propriétés. Concrètement, si l'on note $\mathbb{M}_{BLP}[\rho_{BLP}] = (\mathbb{P}_{BLP}[\rho_{BLP}], \|\mathcal{R}\|_{\Sigma_{BLP}})$ le modèle de Bell et LaPadula et $\mathbb{M}_{CW}[\rho_{CW}] = (\mathbb{P}_{CW}[\rho_{CW}], \|\mathcal{R}\|_{\Sigma_{CW}})$ le modèle de la Muraille de Chine, donner une interprétation de $\mathbb{M}_{CW}[\rho_{CW}]$ basée sur un treillis de niveaux de sécurité consiste tout d'abord à définir un treillis de niveaux de sécurité ρ_{BLP} à partir de ρ_{CW} puis à définir une relation $\kappa_{\Sigma} \subseteq \Sigma_{CW} \times \Sigma_{BLP}$ telle que l'implantation classique $I_{CW} = (\tau_{CW}, \Sigma_I^{CW})$ de la Muraille de Chine soit simulable par une implantation correcte mais non standard $I'_{BLP} = (\tau'_{BLP}, \Sigma_I'^{BLP})$ du modèle de Bell et LaPadula. Bien sûr, I'_{BLP} s'obtient en réutilisant l'implantation standard de Bell et LaPadula.

D'un point de vue plus théorique, les notions introduites dans cette section permettent de définir un préordre sur les modèles de contrôle d'accès. On dira qu'un modèle $\mathbb{M}_1[\rho_1]$ est plus restrictif qu'un modèle $\mathbb{M}_2[\rho_2]$, noté $\mathbb{M}_1[\rho_1] \preceq \mathbb{M}_2[\rho_2]$, si et seulement si toute implantation correcte de $\mathbb{M}_1[\rho_1]$ peut être simulée par une implantation correcte de $\mathbb{M}_2[\rho_2]$.

Avec une telle définition, comparer deux modèles nécessite de considérer toutes les implantations d'un modèle ce qui dans la pratique pose un problème

évident. Toutefois, si la relation de simulation qui permet d'envisager la comparaison vérifie des propriétés supplémentaires, ce problème peut être évité.

En effet, c'est le cas lorsque la relation de simulation construite est injective et fonctionnelle puisque dans ce cas il suffit seulement de montrer que toutes les implantations maximales¹ peuvent être simulées. On montre dans ce cas que si une implantation I est simulable, alors toute implantation I' telle que $I' \sqsubseteq I$ est aussi simulable.

C'est aussi le cas lorsque l'on sait définir une relation de simulation qui préserve à la fois les propriétés de sécurité des politiques et la sémantique des requêtes. On montre en effet directement dans ce cas que $\mathbb{M}_1[\rho_1] \trianglelefteq \mathbb{M}_2[\rho_2]$.

Ce préordre nous a permis de montrer que le modèle de la Muraille de Chine est strictement plus restrictif que le modèle de Bell et LaPadula. En effet, on a bien $\mathbb{M}_{CW}[\rho_{CW}] \trianglelefteq \mathbb{M}_{BLP}[\rho_{BLP}]$, ce qui se prouve en montrant que l'implantation standard de la Muraille de Chine est l'unique implantation maximale de ce modèle. D'autre part, il est possible de trouver une implantation du modèle de Bell et LaPadula qui ne soit pas simulable par une implantation du modèle de la Muraille de Chine.

5 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'informations dans les réseaux et dans les systèmes d'information. Il convient de développer au sein des systèmes informatiques des mécanismes permettant de filtrer les accès afin de ne laisser passer que ceux autorisés. Il s'agit pour cela de définir une politique de sécurité, c'est-à-dire la caractérisation des accès permis. Le programme chargé de mettre en application cette politique, le moniteur de référence, est souvent considéré comme l'une des clés de voûte de la sécurité d'un système. Sa conception et son développement doivent être menés de manière à garantir sa fiabilité et sa sûreté. En effet, toute faille au sein de ce programme pourrait entraîner des violations de la politique de sécurité. L'emploi des méthodes formelles dans le développement d'un moniteur de référence permet de garantir que certaines propriétés sont toujours respectées.

Dans cet article nous avons rendu compte de manière informelle de quelques expériences formelles que nous avons faites dans cette direction. Comme nous l'avons vu, les problèmes posés par une approche formelle du contrôle d'accès ne sont pas aussi simples qu'ils paraissent.

L'ensemble des travaux décrits ci-dessus méritent à présent d'être poursuivis. Tout d'abord, l'indispensable travail de formalisation a mis en relief certaines confusions : plusieurs définitions non équivalentes d'une même propriété coexistent dans la littérature. Nous envisageons de continuer notre étude en formalisant d'autres politiques de sécurité (par exemple en considérant RBAC [13], déjà formalisé dans [28]). D'autre part, nous souhaitons enrichir le cadre sémantique

¹ L'implantation I d'un modèle $\mathbb{M}[\rho]$ est maximale, s'il n'existe pas d'implantation I' (avec $I' \neq I$) de ce même modèle telle que $I \sqsubseteq I'$.

proposé afin de permettre d’envisager la composition de politiques de contrôle d’accès.

Nous souhaitons aussi “comparer” les “comparaisons de modèles de contrôle d’accès”. En effet, quelques travaux ont déjà eu lieu sur ce sujet mais ils sont encore parcellaires: [29] envisage la comparaison de politiques de contrôle d’accès en terme de puissance d’expression, [9] utilise des techniques de simulation pour comparer des modèles de contrôle d’accès discrétionnaire, [30] envisage la comparaison de politiques sous l’angle de la combinaison de politiques. Toutes ces approches portent sur le même objet et méritent d’être reconsidérées et étendues dans un cadre uniforme afin d’en étudier les liens et d’en dégager des techniques de réutilisation. Une telle étude doit permettre de mieux identifier les “ingrédients” présents dans une politique de contrôle d’accès, et d’en comprendre le rôle dans une implantation.

A plus long terme, nous souhaitons implanter le cadre obtenu dans l’atelier Focal et en dériver une bibliothèque certifiée de moniteurs de référence. Parallèlement, nous envisageons d’étudier et de caractériser les fonctionnalités qu’un environnement intégré de développement (IDE), comme Focal, doit offrir non seulement pour produire des logiciels en conformité avec les exigences requises pour atteindre de hauts niveaux de confiance (EAL 5 6 7), mais aussi pour faciliter le processus d’évaluation de ces logiciels selon les standards (IEC61508, CC, ...). C’est d’ailleurs la thématique d’un récent projet de l’ANR, le projet SSURF (*Safety and Security under Focal*).

Remerciements. Nous remercions vivement Thérèse Hardin avec qui nous avons de nombreuses conversations fructueuses autour de ce travail.

References

1. Common Criteria for Information Technology Security Evaluation, <http://www.commoncriteriaportal.org/>.
2. Ocaml-mysql v. 1.0.4: <http://raevnos.pennmush.org/code/ocaml-mysql/>.
3. 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.
4. J. P. Anderson. Computer security technology planning study. ESD-TR-73-51, vol 1 AD-758 206, ESD/AFSC Hanscom, AFB Bedford, Mass, 1972.
5. D. Bell and L. LaPadula. Secure Computer Systems: a Mathematical Model. Technical Report MTR-2547 (Vol. II), MITRE Corp., Bedford, MA, May 1973.
6. E. Bertino and R.S. Sandhu. Database security-concepts, approaches, and challenges. *IEEE Trans. Dependable Sec. Comput*, 2(1), 2005.
7. J. Blond and C. Morisset. Formalisation et implantation d’une politique de sécurité d’une base de données. In INRIA, editor, *17ème Journées Francophones des Langues Applicatifs, JFLA’2006*, pages 71–86, 2006.
8. 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.
9. A. Chander, J.C. Mitchell, and D. Dean. A state-transition model of trust management and access control. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop CSFW*, pages 27–43. IEEE Computer Society Press, 2001.

10. D. Delahaye, J.F. Étienne, and V. Donzeau-Gouge. Certifying airport security regulations using the Focal environment. In J. Misra, T. Nipkow, and E. Sekerinski, editors, *FM 2006: Formal Methods, 14th International Symposium on Formal Methods, 2006, Proceedings*, volume 4085 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 2006.
11. C. Dubois, J. Grandguillot, and M. Jaume. Réutilisation de preuves formelles : Une étude pour le système FoC. In INRIA, editor, *14ème Journées Francophones des Langages Applicatifs, JFLA'2003*, pages 63–75, 2003.
12. C. Dubois, T. Hardin, and V. Vigié Donzeau Gouge. Building certified components within Focal. In *Symposium on Trends in Functional Programming*, 2004.
13. S. Gavrilu and J. Barkley. Formal specification for RBAC user/role and role/role relationship management. In *Proceedings of the 3rd ACM Workshop on Role-Based Access Control (RBAC-98)*, pages 81–90. ACM Press, 1998.
14. 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.
15. 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.
16. M. Jaume and C. Morisset. A formal approach to implement access control. *Journal of Information Assurance and Security*, 2:137–148, 2006.
17. M. Jaume and C. Morisset. Towards a formal specification of access control. In P. Degano, R. Kusters, L. Viganò, 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.
18. L.J. LaPadula and D.E. Bell. Secure Computer Systems: A Mathematical Model. *Journal of Computer Security*, 4:239–263, 1996.
19. McLean. The algebra of security. In *Proc. IEEE Symposium on Security and Privacy*, pages 2–7. IEEE Computer Society Press, 1988.
20. V. Prevosto and D. Doligez. Algorithms and proof inheritance in the Foc language. *Journal of Automated Reasoning*, 29(3-4):337–363, dec 2002.
21. V. Prevosto and M. Jaume. Making proofs in a hierarchy of mathematical structures. In *Proceedings of the 11th Calculemus Symposium*, Rome, sep 2003.
22. Focal project. *Focal, version 0.3.1 Tutorial and reference manual*. LIP6 – INRIA – CNAM, sept 2006. Distribution available at: <http://focal.inria.fr>.
23. Logical Project. *The Coq Proof Assistant Reference Manual*. INRIA-Rocquencourt, 2002.
24. R. Rioboo. *Programmer le Calcul Formel, des Algorithmes à la Sémantique, Mémoire d'Habilitation*. Mémoire d'habilitation, Université Pierre et Marie Curie, Paris, France, 2002.
25. Sandhu and Chen. The multilevel relational (MLR) data model. *ACMTISS: ACM Transactions on Information and System Security*, 1, 1998.
26. R. S. Sandhu. A lattice interpretation of the chinese wall policy. In *Proc. 15th NIST-NCSC National Computer Security Conference*, pages 329–339, 1992.
27. R. S. Sandhu. Lattice-Based Access Control Models. *IEEE Computer*, 26(11):9–19, November 1993.
28. H. Toussaint. Formalisation des politiques de contrôle d'accès. Master's thesis, FUNDP, Namur, Belgium, 2006.
29. M.V. Tripunitara and N. Li. Comparing the expressive power of access control models. In *SIGSAC: 11th ACM Conference on Computer and Communications Security*. ACM SIGSAC, 2004.

30. M.C. Tschantz and S. Krishnamurthi. Towards reasonability properties for access-control policy languages. In D.F. Ferraiolo and I. Ray, editors, *SACMAT 2006, 11th ACM Symposium on Access Control Models and Technologies, Proceedings*, pages 160–169. ACM, 2006.