Modélisation et vérification de contrôle d'accès selon une démarche critères communs

Amal Haddad - Thierry Moutet
Laboratoire d'Informatique de Grenoble (LIG)
UMR 5217
681, rue de la Passerelle
BP. 72
38402 Saint-Martin d'Hères cedex, France
{Amal.Haddad,Thierry.Moutet}@imag.fr

Résumé

Cet article présente une démarche de vérification du contrôle d'accès visant à certifier les programmes dont la sécurité est importante. Nous adoptons une approche Critères Communs, nous basant à la fois sur un modèle fonctionnel du système et une description formelle d'une politique de sécurité, tous deux présentés sous la forme de machines abstraites B. Nous présentons l'outil Meca, qui, à partir de ces deux modèles, génère un noyau de sécurité, sous la forme d'un modèle B, visant à sécuriser les accès réalisés dans le modèle fonctionnel d'origine. Ce noyau de sécurité est ensuite utilisé à des fins de preuve, dans l'objectif de vérifier formellement que le modèle fonctionnel ne viole pas la politique de sécurité ainsi décrite, par l'intermédiaire des obligations de preuve associées à la méthode B. Nous illustrons notre approche, de la spécification de la politique à la preuve du modèle, à travers une étude de cas menée sur une carte multi-applicative.

Mots-clés

Critères communs, contrôle d'accès, Meca, politique de sécurité, méthode B, modélisation, vérification, preuve

1. Introduction

La sécurité informatique est le souci majeur des développeurs de logiciels critiques et des utilisateurs. Traditionnellement, la sécurité se définit par les trois propriétés suivantes : la confidentialité, l'intégrité et la disponibilité. La confidentialité concerne la prévention d'une divulgation non autorisée des informations. L'intégrité concerne la prévention d'un altération interdite des informations. La disponibilité concerne la prévention des dénis de service. Pour garantir la sécurité, plusieurs techniques existent dont le chiffrement, l'authentification et le contrôle d'accès. Le contrôle d'accès vise à intercepter toutes les tentatives d'accès aux informations critiques. Il se définit à différents niveaux de conception par les composants suivants. Les politiques de sécurité définissent les règles de haut niveau qui régissent les accès. Les modèles de sécurité dressent une représentation formelle des politiques de sécurité. Les mécanismes de sécurité définissent les fonctions bas niveau (logicielles et matérielles) permettant d'implanter le contrôles imposés par la politique.

Par ailleurs, pour assister les développeurs dans la conception des systèmes critiques, des exigences ont été définies dans le cadre de la norme des *critères communs* [CC006c]. Cette norme permet entre autres aux développeurs d'évaluer la sécurité de leurs systèmes et d'accréditer cette évaluation auprès des utilisateurs potentiels.

Différents niveaux d'évaluation existent. Les niveaux d'évaluation élevés sont plus exigeants en terme de sécurité. Ils requièrent un modèle formel de la politique de sécurité et un modèle formel de la spécification fonctionnelle de la TOE. De plus, la correspondance entre ces deux modèles doit être prouver formellement [CC006b]. Sachant que le modèle de sécurité et la spécification fonctionnelle de la TOE sont développés séparément, cette correspondance est loin d'être une tâche intuitive. Nous proposons ici une approche qui permet d'établir systèmatiquement cette correspondance. Notre approche est aussi mise en oeuvre par l'outil Meca [Had07].

L'article s'organise de la manière suivante. Nous présentons dans la partie 2 un aperçu sur les politiques de contrôle d'accès. La partie 3 est consacrée à la démarche de développement et d'évaluation définie dans les critères communs. Nous mettons l'accent sur les composants que nous avons

utilisés au sein de notre démarche. Dans la partie 4 nous présentons notre outil Meca qui met en oeuvre notre démarche ainsi que la contribution de notre démarche au sein d'un développement conforme aux critères communs. Cette partie comporte aussi un survol sur les principales fonctionnalités de la méthode B. Dans la partie 5 nous présentons une expérimentation concernant une carte à puce multiapplicative. Finalement, nous présentons les conclusions et les perspectives de ces travaux dans la partie 6.

2. Les politiques de contrôle d'accès

Dans cette partie, nous définissons d'abord les principales notions relatives aux contrôle d'accès. Nous présentons ensuite un aperçu de quelques politiques de sécurité. La mise en œuvre du contrôle d'accès fait intervenir plusieurs notions essentielles. Il s'agit des notions d'objets, de sujets, de permission, d'autorisation et d'interdiction.Les *objets* sont des entités passives qui peuvent être de plusieurs natures. Ces entités peuvent contenir des informations élémentaires. Mais il peut également s'agir d'objets aux sens plus larges, comme des fichiers ou encore des opérations à contrôler. Dans tous les cas, les objets représentent des éléments dont les accès doivent êtres protégés. Inversément, les *sujets* sont des entités actives qui manipulent des objets. Parmi les accès que les sujets peuvent effectuer sur des objets, on peut entre autres trouver des accès en lecture ou en écriture à des données (ou variable). On peut encore trouver des accès correspondant à des appels d'opérations. Les *autorisations* (respectivement *interdictions*) décrivent quels accès les sujets ont le droit de faire (resp. sont interdits) sur les objets. Enfin, les permissions définissent l'ensemble des autorisations ou des interdictions d'un système. Dans le cadre du contrôle d'accès, trois principaux types de politiques de sécurité sont généralement utilisés. Les politiques discrétionnaires (DAC) [Lam71] accordent au propriétaire de l'information, généralement le créateur, tous les droits d'accès ainsi que la possibilité de les propager à d'autres selon sa discrétion. Dans le cadre d'une politique discrétionnaire, les permissions sont vérifiées en fonction des droits possédés par le sujet effectuant l'accès à l'objet. Les politiques obligatoires (MAC) prposent l'attribution des niveaux hiérarchiques aux sujets et aux objets. En fonction de ces niveaux, les permissions d'accès sont vérifiées. Les modèles développés dans le cadre de ces politiques visent à préserver la confidentialité (le modèle Bell et Lapadula [BL73]) ou l'intégrité (le modèle Biba [Bib97]) des données. Les politiques basées sur les rôles (RBAC) [FK92] introduisent la notion de rôle. Un rôle présente une tâche au sein de l'organisation. Il est lié à des droits d'accès sur des objets. Les sujets possèdent les droits d'accès accordés aux rôles qu'ils détiennent. Après ce bref survol des politiques de sécurité, nous présentons dans la partie suivante la démarche définie dans le cadre des critères communs.

3. La démarche critères communs

Dans le cadre des critères communs [CC006c], deux types d'exigences sont définies, les exigences fonctionnelles et les exigences d'assurance. Les exigences fonctionnelles offrent un catalogue de composants à utiliser pour respecter certaines objectifs de sécurité. Les exigences d'assurance définissent les actions à entreprendre pour garantir le respect de ces objectifs. Ces exigences sont ensuite passés à l'évaluation pour la vérification de leur véracité. Suite à cette évaluation, une note est attribuée à la partie critique du système désignée par TOE (Target of Evaluation).

3.1. Exigences fonctionnelles

Les exigences fonctionnelles [CC006a] définissent les règles par lesquelles la TOE gère les accès à ses ressources et contrôle les services offerts. Dans notre démarche, nous nous intéressons à la classe concernant la protection des données de l'utilisateur (FDP : user data protection) et particulièrement à la famille politique de contrôle d'accès (access control policy (FDP_ACC)). Les composants de cette famille se rapportent à la politique de sécurité qui doit être mise en oeuvre. Toutefois la description de la politique n'est pas clairement définie, le développeur peut en définir les règles en utilisant les composants fournis par ces exigences. On distingue deux types de politiques de sécurité : les politiques de contrôle d'accès et celles du contrôle de flot d'information. Dans le cadre des politiques de contrôle d'accès, des règles régissent les opérations que les sujets peuvent exécuter sur les objets. Les politiques de contrôle de flot, quant à elles, définissent des règles qui régissent les opérations que les sujets exercent sur les informations. Par exemple, supposons qu'il s'agit de protéger des fichiers sensibles, dans le cadre des politiques de contrôle d'accès, on s'intéresse à protéger les fichiers tandis que dans les politiques de contrôle de flot on s'intéresse aux informations contenues dans ces fichiers. Finalement, ajoutons que les composants de la famille de politique de contrôle de flot d'informations peuvent être utilisés pour définir une politique obligatoire (MAC). Dans la partie suivante, nous présentons les exigences d'assurance et les niveaux d'évaluation de sécurité.

3.2. Exigences d'assurance de sécurité et niveaux dévaluation

Les exigences d'assurance [CC006b] présentent les actions à entreprendre pour s'assurer que la TOE est correctement développée, implémentée et qu'elle ne comporte pas des erreurs pouvant introduire des failles de sécurité. En

évaluant les exigences d'assurance, la TOE peut recevoir une note certifiant le niveau d'évaluation d'assurance (evaluation assurance level (EAL)) atteint. Il existe sept niveaux d'évaluation. Pour atteindre un niveau d'évaluation plus élevé, il faut engager plus d'effort et de coût. Les niveaux d'évaluation élevés requièrent le respect des exigences définies dans certains composants d'assurances où la formalisation occupe une place importante.Dans cet article, nous nous sommes intéressés à certaines exigences d'assurance requises pour des niveaux d'évaluation 6 ou 7 concernant la validation de la classe développement (ADV). Cette classe concerne la représentation des fonctionnalités de la TOE depuis les niveaux abstraits jusqu'à l'implémentation. Elle concerne aussi la politique de sécurité de la TOE et la correspondance entre celle ci et la spécification fonctionnelle du système. Les niveaux d'évaluation élevés requièrent la présence d'un modèle formel de la politique de sécurité de la TOE (composant ADV_SPM1). Aussi, les développeurs sont censés établir, par des preuves mathématiques, la correspondance entre la spécification fonctionnelle et le modèle formel de la politique de sécurité. La consistence interne du modèle formel de la politique de sécurité doit être aussi vérifiée. Ajoutons à cela que les développeurs sont censés fournir un modèle formel de la spécification fonctionnelle de la TOE (composant ADV_FSP6) dans le cadre d'une évaluation à un niveau 7. Notre démarche s'inscrit dans le cadre d'une évaluation à des hauts niveaux des critères communs. Nous la présentons avec notre outil dans la partie suivante.

4. L'approche Meca

Dans cette partie, nous nous intéressons à Meca qui a déjà été présenté dans un travail amont [Had07]. Cet outil a pour objectif de valider du contrôle d'accès au moyen de modèles formels écrit en B. A partir d'un modèle fonctionnel du système et d'une politique de sécurité, Meca génère un noyau de sécurité visant à garantir uniquement des accès licites aux données sensibles. Ce modèle est une aide à la validation de contrôle d'accès. En effet, les critères de formats de sortie proposés permettent de choisir parmi deux approches de validation. Le premier format est orienté pour la génération de test fonctionnel sur une cible de sécurité. Cette approche, vise à garantir qu'une implémentation respecte la politique de sécurité telle qu'elle est spécifiée formellement en n'effectuant que des appels licites à des opérations manipulant des données sensibles [DHM07]

La seconde approche, que nous présentons ici est destinée à offrir une aide pour concevoir dès la spécification un système correct vis-à-vis d'une politique de sécurité. Pour garantir cela, nous réalisons un raffinement du modèle fonctionnel en remplaçant tout accès direct aux données par les accès sécurisés générés par Meca. Ensuite, nous nous basons sur les mécanismes de génération de preuves de modèle formel qui sont associés à la méthode B. Ainsi, nous présenterons par la suite comment les preuves permettent de détecter et de corriger des violations de la politique. Notre démarche est présentée sur la Fig. 1. Avant de la détailler, nous présentons rapidement la méthode B qui nous a offert à la fois le langage de modélisation et le mécanisme de vérification.

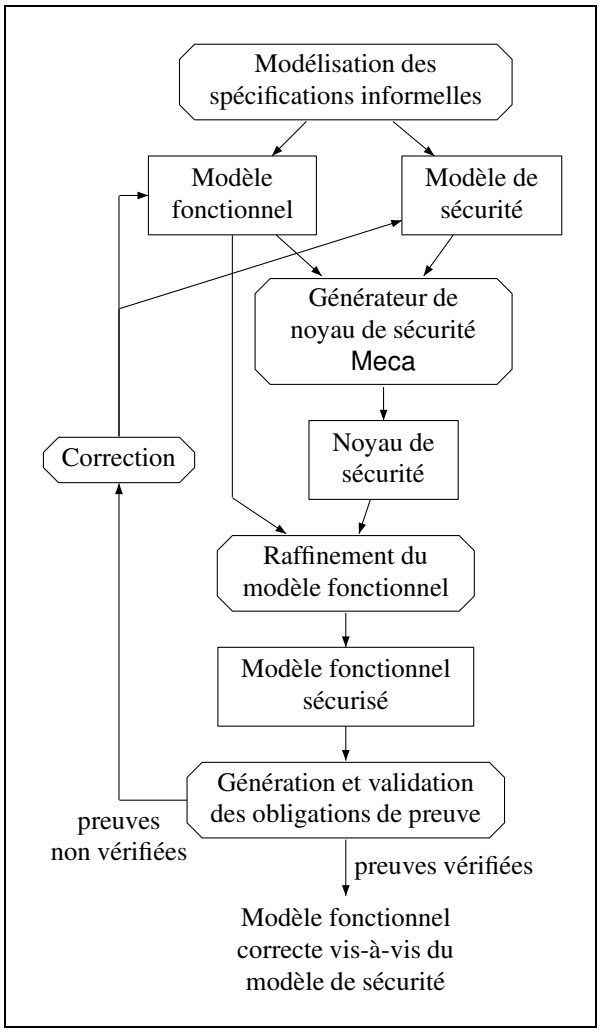

FIG. 1. Principe de l'approche proposée

4.1. La méthode B

La méthode B [Abr96] est une méthode formelle de spécification, de conception et d'implémentation de systèmes logiciels. Ses principaux atouts sont, d'une part, le processus de raffinement qui permet d'écrire des spécifications plus précises. Ce processus peut même aboutir à une implémentation. D'autre part, elle intègre des mécanismes

qui permettent de vérifier au moyen d'obligations de preuve qu'un logiciel est correct. Une obligation de preuve est une formule logique constituée d'un but et d'un ensemble d'hypothèses. Vérifier une obligation de preuve consiste à démontrer que la partie but est vraie sous les hypothèses considérées. Ainsi, cette méthode s'avère particulièrement adaptée au développement de logiciels dont la sécurité est critique. Les outils autour de la méthode B, tel que l'atelier B par exemple, permettent d'automatiser la génération des obligations de preuve. L'atelier B dispose aussi d'un prouveur automatique performant qui permet de décharger automatiquement les obligations de preuve. Dans le cas où la preuve n'a pas pu être vérifié automatiquement, le prouveur interactif permet de décharger ces obligations par l'intervention de l'utilisateur. Les modèles B intervenant dans l'outil Meca sont présentés dans la partie suivante.

4.2. Formats du modèle de sécurité

Le modèle de sécurité utilisé par Meca décrit déclarativement des modèles de politiques de contrôle d'accès avec les divers entités qui en font partie. Ainsi, pour définir une politique de sécurité pour Meca, il faut d'abord donner la liste des sujets et des objets. Le premier traitement effectué par Meca est un test de consistance de ces deux ensembles. Il vérifie l'existance de ces sujets et/ou de ces objets en fonction du type de politique définie. Ensuite, il faut décrire la liste des permissions du systèmes. La modélisation adoptée dans Meca est la suivante. On ne peut définir que des permissions correspondant à des autorisations. Par défaut, tout ce qui n'est pas décrit dans des permissions est interdit. La Figure 2 présente le format générique du modèle de sécurité de Meca. Meca propose plusieurs formats

MACHINE
generic_security_policy
SETS
SUBJECT={$s1$,$s2$,...} ; /* Entités actives */
OBJECT={$o1$,$o2$,...} /* Entités passives */
CONSTANTS
permission
PROPERTIES
permission ∈ SUBJECT ↔ OBJECT ∧
$condition \Rightarrow (s1 \mapsto o1) \in$ permission ∧
...
END

FIG. 2. Schéma générique du modèle de sécurité

de modèle de sécurité selon la nature du contrôle d'accès que l'on souhaite adopter. Parmi ceux-ci, on retrouve les formats classiques tels que DAC, MAC et RBAC. Le format discrétionnaire (DAC) est proposé avec deux variantes. Toutes deux suivent le schéma de la Figure 2. Dans la première, les sujets correspondent à des variables du modèle fonctionnel et les objets à des opérations. On cherche alors à contrôler les accès par les opérations en lecture et en écriture aux variables. La deuxième variante vise à sécuriser des appels d'opérations uniquement par des utilisateurs autorisés. Ces utilisateurs sont donc définis comme sujets tandis que les objets sont des opérations du modèle fonctionnel. Le format obligatoire (MAC), développé dans l'étude

MACHINE
MAC_security_policy
SETS
SUBJECT={$s1$,$s2$,...} ; /* Entités actives */
OBJECT={$o1$,$o2$,...} /* Entités passives */
LEVEL={$l1$,$l2$,...} /* Niveaux d'accès */
CONSTANTS
order, clearance, classification
PROPERTIES
order ∈ LEVEL → NAT_1 ∧
clearance ∈ SUBJECT → LEVEL ∧
classification ∈ OBJECT → LEVEL ∧
order = {($l1 \mapsto 1$), ($l2 \mapsto 2$), ...} ∧
clearance = {($s1 \mapsto l1$), ($s2 \mapsto l2$), ...} ∧
classification = {($o1 \mapsto l1$), ($o2 \mapsto l2$), ...}
END

FIG. 3. Schéma du modèle de sécurité du format MAC

de cas de cet article ressemble à la première variante du format DAC. On retrouve également un ensemble d'opérations comme sujets et de variables comme objets. Cependant, les propriétés de sécurité qui doivent être préservées sont un peu différentes. Il en résulte une formalisation des permissions un peu plus évoluée. La constante *permission* est remplacée par les constantes *level*, *clearance* et *classification*. Par l'intermédiaire de ces constantes, on définit un ensemble de niveaux d'accès grâce à *level* et on associe un niveau à chaque sujet et à chaque objet grâce respectivement à *clearance* et à *classification*. Ce format est montré sur la Figure 3. L'affectation des niveaux aux sujets et aux objets doit être judicieusement choisie en fonction des propriétés de sécurité à garantir. En effet, pour préserver la *confidentialité*, on attribue des niveaux de manière à ce que chaque sujet n'a accès en lecture à un objet que s'il a un niveau supérieur à ce dernier. Inversément, seul un sujet de niveau inférieur à celui d'un objet peut l'écrire. Dans le cas où c'est l'*intégrité* qui doit être préservée, l'affectation des niveaux est effectuée de manière inverse.

Nous pouvons constater alors que dans notre approche, nous avons suivi la démarche critères communs qui consistent à modéliser la politique de contrôle d'accès. Nous avons offert des formats génériques pour représen-

ter les différents modèles classiques tels que DAC, MAC et RBAC. Nous ne présentons pas ici le format du modèle RBAC faute d'espace.

4.3. Formats du noyau de sécurité

A partir du modèle fonctionnel et du modèle de sécurité, Meca génère un noyau de sécurité qui traduit l'aspect opérationnel de la politique. Il associe à chaque objet et pour chaque type d'accès (lecture, écriture, ...) une opération dont le rôle est de vérifier que l'on est dans un contexte de sécurité autorisé. Pour cela, le sujet qui accède à l'objet est passé en paramètre de chaque opération. L'opération s'assure alors que le sujet est défini dans une permission mais également que la condition de validité de cette permission est présente, comformément au modèle de sécurité.

```
MACHINE
    generic_security_kernel
...
OPERATIONS
    val ⟵ read_v1 (su) ≙ /* accès en lecture */
    PRE     su ∈ SUBJECT ∧
            PRE_SECU   /* issu du modèle de sécurité */
    THEN    val := v1
    END ;
    write_v1 (su, val) ≙     /* accès en écriture */
    PRE     su ∈ SUBJECT ∧
            PRE_SECU   /* issu du modèle de sécurité */
    THEN    v1 := val
    END ;
    execute_op (su) ≙        /* accès en exécution */
    PRE     su ∈ SUBJECT ∧
            PRE_SECU   /* issu du modèle de sécurité */
    THEN    op
    END ;
    ...
END
```

FIG. 4. Noyau de sécurite générique

Actuellement, deux types de contrôle d'accès sont disponibles dans Meca. L'utilisation d'un modèle de sécurité dans lequel on définit des variables comme objets se traduit par la génération d'un noyau de sécurité qui contrôle uniquement les accès en lecture et en écriture à ces variables. Ainsi, pour chaque variable *vn*, Meca génère une opération *read_vn(su)* et *write_vn(su)*. Dans le cas où ce sont des opérations qui sont définies comme objets dans le modèle de sécurité, Meca génère dans le noyau de sécurité une opération *execute_op* pour chaque opération *op*. La Fig. 4 présente la forme des opérations générées par Meca uniquement sous la forme qui est utilisée pour la preuve. Intéressons nous plus précisément au noyau de sécurité qui correspond au modèle de sécurité de type MAC abordé à la Fig. 3. Dans ce modèle, seuls sont contrôlées des accès en lecture et en écriture. On obtient donc uniquement des opérations *read_vn(su)* et *write_vn(su)*. Pour ce type de modèle, la précondition notée PRE_SECU est synthétisée de la manière suivante. Pour les accès en lecture et dans le cas d'une préservation de la confidentialité, la liste de tous les sujets dont le niveau est supérieur à celui de l'objet vn est récupérée. Ensuite, le paramètre su doit correspondre à un de ces sujets et la relation d'ordre doit être vérifiée entre ce sujet et cet objet. La Fig. 5 montre quelle est la précondition obtenue pour des niveaux définis comme sur la Fig. 3, avec l2 $\geq$ l1.

```
val ⟵ read_o1 (su) ≙
PRE     su ∈ SUBJECT ∧
        habilitation(su) ≥ classification(o1)
THEN    val := v1
END ;
write_o1 (su, val) ≙
PRE     su ∈ SUBJECT ∧
        /* val ∈ TYPE(v1) ∧ */
        habilitation(su) ≤ classification(o1)
THEN    v1 := val
END ;
```

FIG. 5. Schéma du modèle de sécurité du format MAC

4.4. Format du modèle fonctionnel sécurisé

Le noyau de sécurité comporte uniquement les opérations sécurisées afférentes au contrôle d'accès. Cependant, l'objectif initial de notre démarche est de vérifier que ce contrôle d'accès est correctement modélisé dans le modèle fonctionnel. Ainsi, nous avons inséré une étape supplémentaire de raffinement avec inclusion du noyau de sécurité pour permettre cette vérification. Le principe de cette étape consiste à remplacer chaque accès direct aux objets du modèle de sécurité par un accès sécurisé à ces objets. En pratique, cela se traduit pour une politique MAC par le remplacement de chaque accès en lecture à une variable var par l'opération read_var généré par Meca dans le noyau de sécurité. Les accès en écriture quant à eux remplacés par l'opération write_var. Ce nouveau modèle ainsi obtenu est appelé modèle fonctionnel sécurisé. Nous allons illustrer cela sur l'étude de cas présentée dans la section 5.

A travers ce modèle sécurisé sont générés les obligations de preuve pour valider le modèle fonctionnel initial vis-à-vis du modèle de sécurité.

4.5. Génération des obligations de preuve

Au sein du modèle fonctionnel sécurisé, les obligations de preuve sont générées par l'atelier B. Nous nous intéressons aux obligations de preuve pouvant attester du respect de la politique de sécurité par le modèle fonctionnel. Ces obligations de preuve se rapportent à l'appel des opérations du noyau de sécurité. Elles consistent à vérifier que la précondition de l'opération appelée est une conséquence logique de la précondition de l'opération appelante et des invariants des deux composants renfermant ces opérations. Ceci a pour but de vérifier que l'appel de l'opération avec les paramètres instanciés en cours ne viole pas la précondition de l'opération du noyau de sécurité. Nous illustrons cela au sein de l'étude de cas présentée dans la section suivante. Le prouveur de l''atelier B permet de prouver d'une façon automatique les obligations de preuve. Celles non prouvées automatiquement doivent être examinées car elles peuvent révéler une incohérence entre le modèle fonctionnel et la politique de sécurité et dans ce cas il faut corriger l'un des modèles. Le prouveur interactif est utilisé pour prouver le reste des obligations de preuve jugées correctes. Ainsi, à la fin de ce processus, le développeur a pu corriger les incohérences en terme de de sécurité et il possède en plus les preuves qui attestent de cela. Dans cette partie, nous avons présenté l'utilisation du noyau de sécurité généré par Meca pour prouver formellement la cohérence entre le modèle formelle de sécurité et le modèle fonctionnel de la TOE. Cette démarche nous permet d'être conforme aux recommandations de certification haut niveau des critères communs EAL6, EAL 7. Nous illustrons notre démarche sur une étude de cas qui sera présentée dans la partie suivante.

5. Etude de cas

Dans cette partie, nous présentons une étude de cas dans le cadre de la mise en œuvre de l'outil Meca. Il s'agit d'une carte de santé multi-applicative qui contient trois applets, chacune correspondant à une application. La première est une application de santé (notée *appletHC*), qui a été chargée dans la carte par son fournisseur. Cet applet contient un certain nombre de données administratives. Parmi celles-ci, le numéro de sécurité social du porteur de la carte. La seconde est un programme de fidélité à une chaîne de pharmacies (notée *appletDS*). Cette application peut lire *ssn* afin de permettre au pharmacien d'identifier le patient. Elle contient également une donnée (notée *allergies*) qui comporte la liste des médicaments auquel le patient est allergique. Elle gère aussi un compteur de points de fidélité (noté *pts*). Ce compteur est mis à jour à chaque fois que des médicaments sont délivrés. Enfin, la troisième application est un programme de fidélité à une chaîne de centres de sport (notée *appletSC*). Grâce à des accords avec la chaîne de pharmacies, chaque centre de sport peut accéder à *pts* et offrir des réductions en fonction du nombre de points. La Fig. 6 présente un extrait du modèle fonctionnel. Sur ce modèle apparaissent les variables *ssn*, *allergies* et *pts*. Seuls deux opérations du programme de fidélité de la chaîne pharmaceutique ont été décrites. La première $identifyPatientDrug$ est utilisée par le pharmacien pour récupérer le numéro de sécurité sociale du client. La seconde, $updateLoyalityPtsDrug$ met à jour les points de fidélité du patient suite à l'achat d'un médicament.

```
MACHINE
    ex_functional_model
VARIABLES
    ssn, pts, allergies ...
OPERATIONS
    secu_nb ⟵ identifyPatientDrug ≙
    BEGIN ;    secu_nb := ssn
    END ;
    updateLoyaltyPtsDrug (pt) ≙
    PRE        pt ∈ 0 .. 100000
    THEN
        IF          pts + pt <= 100000
        THEN        pts := pts + pt
        END
    END ;
END
```

FIG. 6. Schéma du modèle fonctionnel de l'application

5.1. Modèle de sécurité

Pour modéliser le contrôle d'accès de notre étude de cas nous avons choisi la politique obligatoire en privilégiant la confidentialité dans un format de type Bell et Lapadula. En effet, au sein de carte multiapplicative, nous voulons garantir qu'une applet ne transmet pas illégallement une donnée à une autre applet. Par exemple, l'applet pharmacie a le droit de lire le numéro de sécurité social mais n'a pas le droit de le donner aux autres applets par exemple à l'applet de centre de sport. La menace de sécurité peut être causée si l'applet de pharmacie copie le numéro de sécurité social dans un objet partagé par d'autres applets par exemple les points de fidélité auxquels l'applet de centre de sport a accès en lecture. Les politiques obligatoires peuvent contrer de tels menaces car elles proposent une classification des sujets et des objets qui régit les accès autorisés. Ainsi par rapport à notre étude de cas, le numéro de sécurité social qui est un objet partagé entre les deux applets de santé et de pharmacie sera classifié à un niveau particulier inférieur au niveau de l'applet de pharmacie et de celui de l'applet du centre de sport. Dans

le cadre de la préservation de la confidentialité, l'applet de pharmacie peut lire le numéro de sécurité social et elle peut le recopier dans un objet qui lui appartient, néanmoins elle ne pourra pas le transférer dans un autre objet partagé par d'autres applets et par la suite ayant un niveau inférieur car cela causera un flot d'information illégal d'un niveau plus haut vers un niveau plus bas.

Nous avons défini les niveaux suivants : DS (Drug Store) ce niveau est lié à l'applet pharamcie, SSN (Social Security number) est lié au numéro de sécurité social, il est inférieur au niveau DS puisqu'il est lié à un objet partagé entre les deux applets pharmacie et de santé. Le niveau PTS (Loyality points) est lié aux points de fidélité.

MACHINE
$ex\_security\_policy$
SETS
SUBJECT={$op\_IdentifyPatientDrug$, $op\_UpdateLoyaltyPtsDrug$};
OBJECT={var_ssn,var_pts,$var\_allergies$};
LEVEL={DS, SSN, PTS};
PROPERTIES
order = {($PTS \mapsto 1$), ($SSN \mapsto 1$), ($DS \mapsto 2$)} $\wedge$
clearance = {($op\_identifyPatientDrug \mapsto DS$), ($op\_updateLoyaltyPtsDrug \mapsto PTS$)} $\wedge$
classification = {($var\_ssn \mapsto SSN$), ($var\_pts \mapsto PTS$), ($var\_allergies \mapsto DS$)}
END

FIG. 7. Schéma du modèle de sécurité

5.2. Le noyau de sécurité et son utilisation pour la preuve

Le noyau de sécurité générée contient les opérations de lecture et d'écriture sécurisées pour chaque variable à protéger. Nous présentons dans la Fig.8 les opérations de lecture et d'écriture générées au sein du noyau pour la variable pts.

La figure 9 présente le raffinement de l'opération $updateLoyalityPtsDrug$ où les accès directs en lecture et écriture sur la variable pts sont remplacées par les opérations sécurisées read_pts et write_pts instanciées avec les valeurs courants.

Parmi les obligations de preuve générées par l'atelier B, nous retrouvons celles relatives aux appels d'opérations du noyau de sécurité. Par exemple concernant l'appel read _pts, l'obligation de preuve générée comporte dans sa partie but, le prédicat suivant : order(classification(var_pts)) <= order (clearance(op_updateLoyalityPtsDrug). Ce prédicat étant la précondition de l'opération read_var réflétant l'aspect opérationnel d'une politique à la Bell et Lapadula. En prouvant cette obligation de preuve, nous vérifions que l'opération $updateLoyalityPtsDrug$ a le droit

MACHINE
$ex\_security\_kernel$
VARIABLES
ssn, pts, allergies, ...
OPERATIONS
val $\longleftarrow$ $read\_pts$ (su) $\hat{=}$
PRE
su $\in$ SUBJECT $\wedge$
order(clearance(su)) $\geq$ order(classification(var_pts))
THEN
val := pts
END ;
$write\_pts$ (su, val) $\hat{=}$
PRE
su $\in$ SUBJECT $\wedge$
val $\in$ 0..100000 $\wedge$
order(clearance(su)) $\leq$ order(classification(var_pts))
THEN
pts := val
END
END

FIG. 8. Noyau de sécurite de l'exemple

de lire la variable pts. Finalement, nous signalons que pour cet exemple, l'atelier B a généré 61 obligations de preuve. Toutes ont été automatiquement prouvées.

6. Conclusions et Perspectives

Dans cet article, nous avons proposé une démarche pour la modélisation et la vérification du contrôle d'accès. S'inscrivant dans le cadre des critères communs, elle permet une évaluation de haut niveau pour des systèmes critiques en terme de sécurité.

Une démarche proche de la nôtre a été appliquée dans le cadre du développement d'une passerelle FOX [Bie96] qui contrôle la communication entre deux réseaux de niveaux de sécurité différents. Le but est de vérifier que les spécifications fonctionnelles de FOX sont conformes au modèle de politique de sécurité dans le cadre d'une évaluation selon les critères ITSEC [ITS90]. Le modèle de Bell et Lapadula a été choisi par l'auteur pour modéliser la politique de sécurité. Des services sécurisées similaires aux opérations de notre noyau de sécurité sont aussi proposées. Ces services sont utilisés ensuite par les opérations critiques de FOX extraites à partir des exigences fonctionnelles. L'auteur a aussi utilisé la méthode B pour la formalisation et la vérification. Cependant, notre démarche présente l'intérêt d'être plus générique et de couvrir plusieurs formats de politiques de sécurité. De plus, comme elle est outillée, elle permet d'automatiser la démarche. L'article [BCM07] présente aussi une démarche pour prouver le respect de la politique de sécurité

```
MACHINE
    ex_functional_secur
REFINES
    ex_functional_model
OPERATIONS
    updateLoyaltyPtsDrug (pt) ≙
    PRE        pt ∈ 0 .. 100000
    THEN
        VAR po IN
        po ⟵ read_pts(op_updateLoyaltyPtsDrug) ;
        IF          po + pt <= 100000
        THEN
            write_pts(op_updateLoyaltyPtsDrug,po+pt)
        END
    END ;
END
```

FIG. 9. Modèle fonctionnel sécurisé de l'exemple

par un système. Le modèle de politique de sécurité traité est ORBAC [AEKAG03]. Il est formalisé dans le cadre de la méthode B. A travers des raffinements successifs les auteurs construisent le système à partir de la politique de sécurité. La démarche n'est pas outillée. Notre démarche se distingue par le fait que nous partons de deux modèles séparés, l'un concernant la politique et l'autre concernant le modèle fonctionnel puis nous tissons la sécurité au sein du modèle fonctionnel sécurisé par les appels d'opérations du noyau de sécurité généré par notre outil Meca.

Nous proposons d'étendre notre approche pour tenir compte d'autres familles définies dans le cadre des exigences fonctionnelles des critères communs. Par exemple, la famille FDP_ACF préconise de traiter des politiques de sécurité dont les règles dépendent des attributs de sécurité, qui sont des entités utilisées par l'aspect opérationnel de la TOE. Ils sont liés aux utilisateurs, sujets et objets, comme la date de création, une valeur associée, la liste des accès autorisés. Dans l'article [DHM07] nous avons proposé un modèle UAC (User Access Control) qui correspond à ce type de politique de sécurité. Le modèle est en cours d'extention et il nous semble intéressant d'adapter notre démarche à ce format. Nous envisageons aussi de traiter la famille FMT_SMF qui concerne l'administration de la politique de sécurité et des attibuts de sécurité, plus la vérification que celle ci se fait conformément au modèle de sécurité.

Références

[Abr96] J.-R. Abrial. *The B-book : assigning programs to meanings*. Cambridge University Press, 1996.

[AEKAG03] Balbiani P. Benferhat S. Cuppens F. Deswarte Y. Miège A. Saurel C. Abou El Kalam A., EL Baida R. and Trouessin G. Organization based access control. In *Policy'03, the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, June 2003.

[BCM07] N. Benaïssa, D Cansell, and D Méry. Integration of security policy into system modeling. In *B'2007, the 7th Int. B Conference*, volume 4355 of *LNCS*, pages 232–247, Besançon, France, January 2007. Springer.

[Bib97] K. J. Biba. Integrity considerations for secure computer systems. Technical report tr-3153, The Mitre Corporation, Bedford, 1997.

[Bie96] P. Bieber. Interprétation d'un modèle de sécurité. *Techniques et Sciences Informatiques, TSI*, Avril 1996.

[BL73] D. E. Bell and L. J. Lapadula. Secure computer systems : A mathematical model. Technical report esd-tr-278, vol. 2, The Mitre Corporation, Bedford, 1973.

[CC006a] Common Criteria for Information Technology Security Evaluation, Part 2 : Security functional components. Technical Report CCMB-2006-09-002, sept 2006.

[CC006b] Common Criteria for Information Technology Security Evaluation, Part 3 : Security assurance components. Technical Report CCMB-2006-09-003, sept 2006.

[CC006c] Common Criteria for Information Technology Security Evaluation, version 3.1. Technical Report CCMB-2006-09-001, sept 2006.

[DHM07] F. Dadeau, A. Haddad, and T. Moutet. Test fonctionnel de conformité vis à vis d'une politique de contrôle d'accès. In *AFADL 2007, en cours de publication, Namur, Belgique*, June 2007.

[FK92] D. F. Ferraiolo and D. R. Kuhn. Role based access control. In *15th National Computer Security Conference*, 1992.

[Had07] Amal Haddad. Meca : A tool for access control models. In *B'2007, the 7th Int. B Conference*, volume 4355 of *LNCS*, pages 281–284, Besançon, France, January 2007. Springer.

[ITS90] EUROPEAN ECONOMIC COMMUNITY, Information Tecnnology Security Evaluation Criteria (ITSEC). Technical report, 1990.

[Lam71] Butler W. Lampson. Protection. In *5th Princeton Symposium on Information Science and Systems*, pages 437–443, 1971.