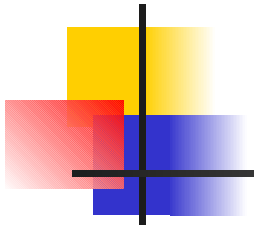
MODELISATION
ET
VERIFICATION DE POLITIQUES DE SECURITE A L'AIDE DE LA METHODE B

Amal HADDAD
1ère année de thèse INPG

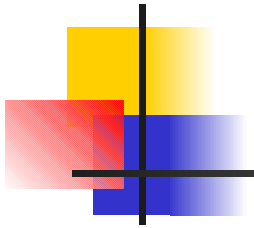
LSR-VASCO

Directeurs :
Mme. Marie-Laure POTET
M. Yves LEDRU



Cadre du travail

- **Problématique :**
  - Attaques contre la sécurité
  - Conséquences : Dommages matériels et humains
    - Codes malveillants ont causé + de 12 millions d'euros de dommages uniquement en 2001 (Carlsbad)
- **Objectifs :**
  - Introduire les concepts de sécurité dans le cycle de développement des applications
  - Décrire des politiques de sécurité
  - Produire des conditions de vérification
  - Annoter une application par des assertions
- **Moyens :**
  - Méthodes formelles pour la spécification et la vérification par preuves → Critères Commun (Norme ISO 15408-Version 3.0-2005)
- **Techniques :**
  - **Contrôle d'accès (préserver confidentialité et intégrité)**
  - Authentification, Chiffrement…



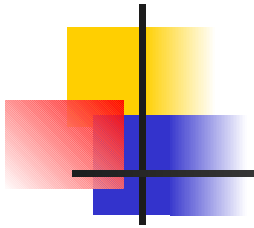
Contrôle accès

- Entités :
  - objets : entités passives contenant des informations
  - sujets : entités actives accédant aux objets et possèdant des droits d'autorisations.

  N.B : Un sujet peut être un objet s'il est manipulé par d'autres sujets

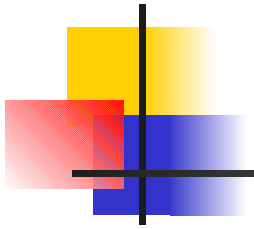
- Définition: Vérification des conditions d'accès des sujets aux objets

- Phases de construction :
  - Politiques : lois, règles et pratiques régissant la protection des informations sensibles
  - Modèles : représentation formelle des politiques de sécurité
  - Implantation : mécanismes de bas niveau



Politiques de sécurité

- Politiques discrétionnaires (DAC):
  - Les propriétaires de l'information possèdent les droits d'accès et la possibilité de les propager
- Politiques obligatoires (MAC):
  - Les privilèges des sujets sur les objets sont données par des règles générales
  - Origine: Systèmes militaires
  - Contrôle des flots d'information
- Politiques basées sur les rôles (RBAC):
  - + récente, adaptée aux organisations
  - Les droits sont accordés à des rôles, des rôles sont attribués aux sujets. Les sujets possèdent les droits selon les rôles qu'ils détiennent.



Vérification statique et vérification dynamique

- Vérification dynamique :
  - A l'exécution: un moniteur de référence contrôle les accès et décide lesquels sont autorisés
  - - ralentit l'exécution, coûteuse en mémoire
  - + permet de détecter en temps réel certaines aggressions
- Vérification statique :
  - Vérification sans activation réelle : preuves mathématiques (ex : vérification à l'aide des obligations de preuve) , analyse du comportement …
  - + valable pour toute exécution
  - + découvrir les erreurs à priori et avant l'exécution
  - - maîtriser le contexte d'exécution

Outil MECA

Objectifs :

3) Définir un langage d'entrée pour les différentes politiques de sécurité : DAC, MAC, RBAC
4) Vérifier la cohérence de la politique de sécurité spécifiée => vérification des fichiers modélisant les politiques
5) Générer automatiquement un noyau de sécurité :
   - empêche les accès directs des sujets sur les objets (médiateur)
   - fournir des services (opérations) préconditionnés par les critères qui doivent être satisfaits pour chaque accès
6) Produire des spécifications qui doivent être vérifiées par la preuve (vérification statique)
   - Appels des opérations sûres du noyau de sécurité

Méthode B

- Langage pour décrire les politiques de sécurité (les fichiers d'entrée pour MECA)
- Un mécanisme de génération de preuves (Atelier B)

  ⇒ Prise en charge de la production des obligations de preuve (appels d'opérations)

- Un prouveur automatique et interactif

  ⇒ La méthode B garantit, par des obligations de preuve, que les appels aux opérations vérifient toujours les préconditions de celles-ci :

  $$[\text{Pre } P \text{ then } S \text{ END }]\ R \Leftrightarrow P \wedge [S]\ R$$

- Certaines vérifications pour le fichier politique : Contrôleur de type (AtelierB)

Politiques obligatoires

- Classe d'accès associée à chaque sujet et à chaque objet .
- Classe d'accès :
  - Niveau de sécurité : élément d'un ensemble totalement ordonné :
    fabricant de la carte> émetteur de la carte (banque)> porteur de la carte
  - Ensemble de catégories (administratif, financier, transactionnaire pour les systèmes bancaires).

$$CA1 \geq CA2 \Leftrightarrow$$
$$CA1.\text{niveau-sécurité} \geq CA2.\text{niveau-sécurité} \wedge$$
$$CA1.\text{catégories} \supseteq CA2.\text{catégories}$$

Modèle Biba

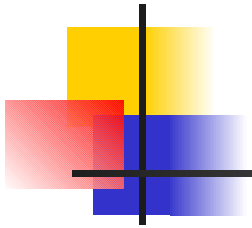
- Préserver l'intégrité
- Ken Biba en 1977
- Niveau d'intégrité d'un sujet (habilitation) : le niveau de confiance qui lui est accordé.
- Niveau d'intégrité d'un objet (classification) : l'importance des dommages causées par les modifications non autorisées.

- Les deux principes :

  - No-write-up :
    permettre(s,o,écrire) ssi s.CA ≥ o.CA

  - No-read-down :
    permettre(s,o,lire) ssi o.CA ≥ s.CA

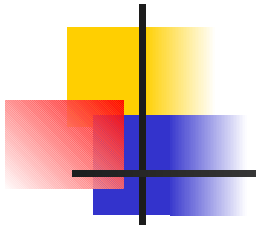
- Pour maintenir la confidentialité (modèle David Bell et Leonard Lapadula en 1973)
  - No-write-down et No-read-up



Modélisation Biba pour MECA

- Les sujets sont des opérations et les objets sont des variables.
- Les ensembles SUJET et OBJET sont constants (on ne tient pas compte des mises à jour)
- Pas de catégories, utilisation des niveaux de sécurité seulement.

Carte à puce bancaire



- Les acteurs : fabricant de la puce, fabricant de carte, émetteur de la carte (banque) et porteur de la carte.
- Le cycle de vie d'une carte à puce: fabrication de la puce, fabrication de la carte, initialisation de la carte (avec les données de l'application), utilisation.
- Le cycle de vie d'une application sur la carte à puce:
  - chargement du code de l'application sur la carte
  - installation d'une instance de l'application
  - sélection de l'instance de l'application
  - Personnalisation
  - Utilisation
- Dans la phase de personnalisation, le code pin du porteur de la carte (hpc) est enregistré par l'émetteur de la carte via l'opération (SetHPC). Le porteur de la carte n'a pas le droit de modifier son hpc.
- Dans la phase d'utilisation et si la carte n'est pas blockée (tryleft≠0), l'opération (checkHPC) vérifie si le porteur de carte entre le bon hpc.

Biba pour carte à puce

MACHINE POLITIQUE_OBLIGATOIRE

SETS

SUJET = {checkHPC, setHPC} ;

OBJET= {hpc, tryleft } ;

NIVEAU= {émetteur_carte, porteur_carte}

CONSTANTS

ordre, habilitation , classification

PROPERTIES

ordre ∈ NIVEAU >→ N

& habilitation ∈ SUJET→ NIVEAU

& classification ∈ OBJET→ NIVEAU

& ordre = { (émetteur_carte 2) , (porteur_carte 1)}

& habilitation = { (checkHPC porteur_carte) , (setHPC émetteur_carte) }

& classification = { (hpc émetteur_carte) , (tryleft porteur_carte)) }

END

Noyau de sécurité généré systématiquement

Pour toute variable de l'ensemble OBJET

- ecrire_tryleft (su,vn) =

  PRE

  su : SUJET

  ∧ vn opens tryleft_type /* opens=opérateur ensembliste */

  ∧ **(ordre (habilitation(su)) ≥ (ordre (classification(tryleft))**

  THEN

  tryleft := vn

  END

- val ← lire_ tryleft (su) =

  PRE

  su ∈ SUJET

  ∧ **(ordre(habilitation(su)) ≤ (ordre(classification(tryleft))**

  THEN

  val := tryleft

  END

Le raffinement sécurisé APPLICATION_REF

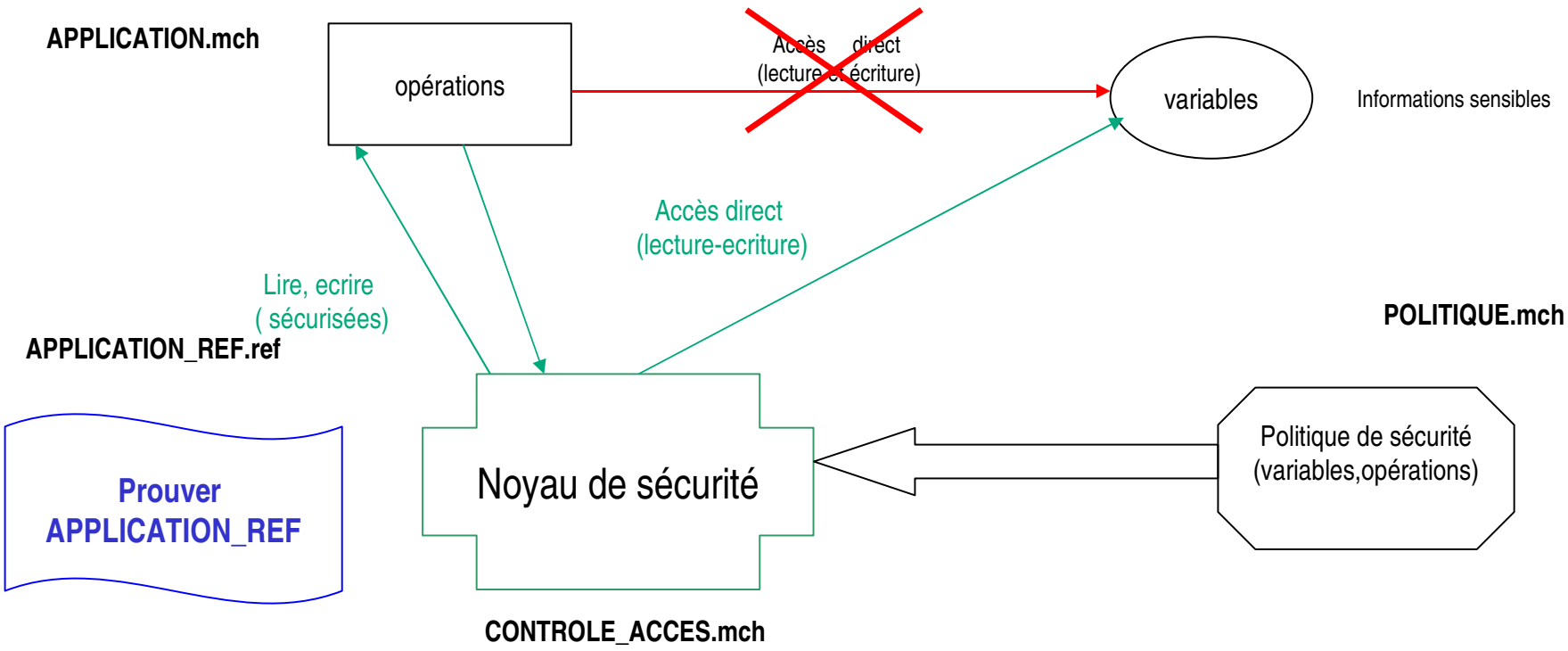
- Dans le corps d'une opération op (op $\in$ SUJET), toute lecture de la variable à protéger var (var € OBJET) dans un prédicat ou expression génère le bloc suivant:

nouvelle_var <-- lire_var(op)

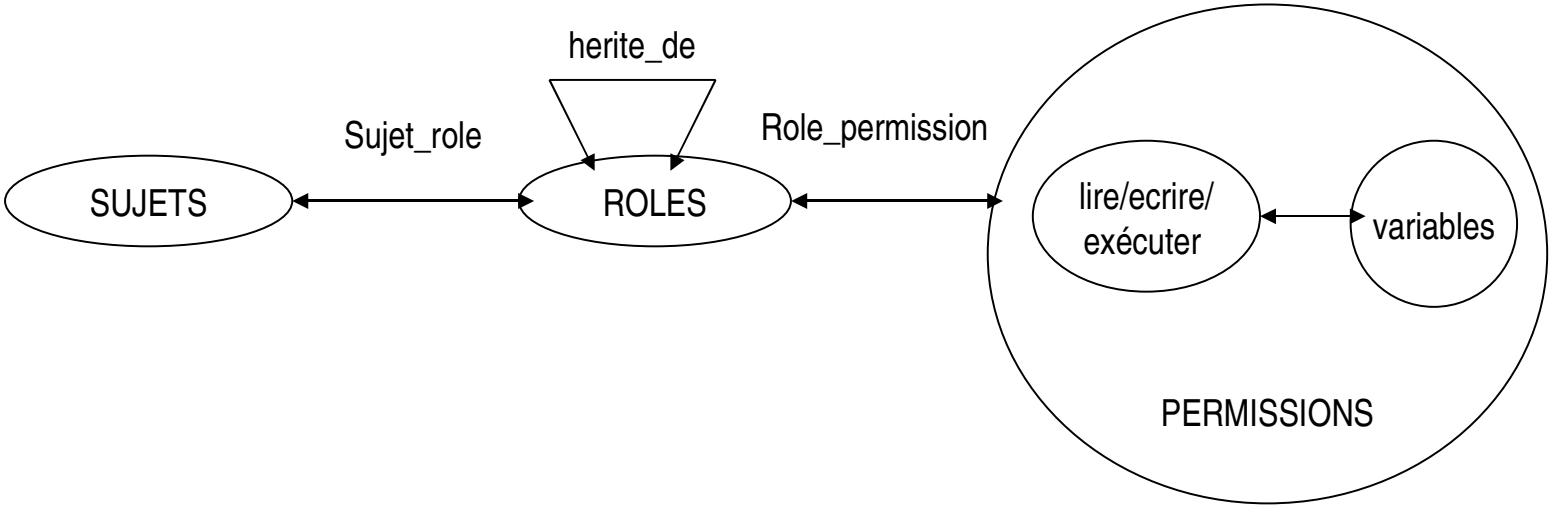
- Dans le corps d'une opération op (op $\in$ SUJET), toute assignation de la valeur vn à la variable var (var $\in$ OBJET) dans une substitution d'affectation génère la substitution suivante :

ecrire_var (su, vn)

Schéma de l'approche



Le modèle RBAC (aspect statique)



DEMONEY (étude de cas)

- Sécurité dans DEMONEY:
  - Les sujets (terminaux) : PDA, GSM, terminal dans magasin, terminal bancaire, terminal adminstratif
  - Les roles : admin, credit, debit, public
  - Les permissions (opérations) :
    - lire informations publiques : balance courant
    - débiter porte-monnaie
    - Créditer le porte-monnaie via cash ou transfert d'un compte bancaire
    - Effectuer des opérations administratives (mettre à jour des paramètres)
  - L'héritage : admin > credit > debit > public
  - Sujet_role :

    PDA, GSM → public, terminal magasin → debit, terminal bancaire → credit, terminal adminstratif → admin
  - Role_permission :
    - Public → lire informations publiques
    - Debit → débiter porte-monnaie
    - Credit → créditer le porte-monnaie
    - Admin → effectuer des opérations administratives

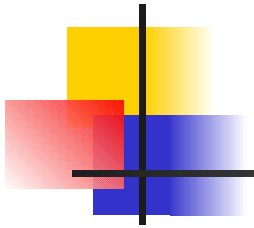
Le modèle RBAC pour MECA (notre choix)

```
MACHINE Politique_rbac_demoney

SETS

    SUJET={Liste_ident};
    ROLE={ Liste_ident };
    PERMISSION={ Liste_ident }

CONSTANTS

  sujet_role,role_permission,herite_de,conflit

PROPERTIES

sujet_role : SUJET --> ROLE  /* pas d'aspect dynamique, un sujet a un seule
role*/
&
role_permission : ROLE <-> PERMISSION
&
herite_de: ROLE <->ROLE  /* chef de projet hérite de employé */
&
conflit: ROLE <->ROLE
&
```

Le modèle RBAC pour MECA(suite)

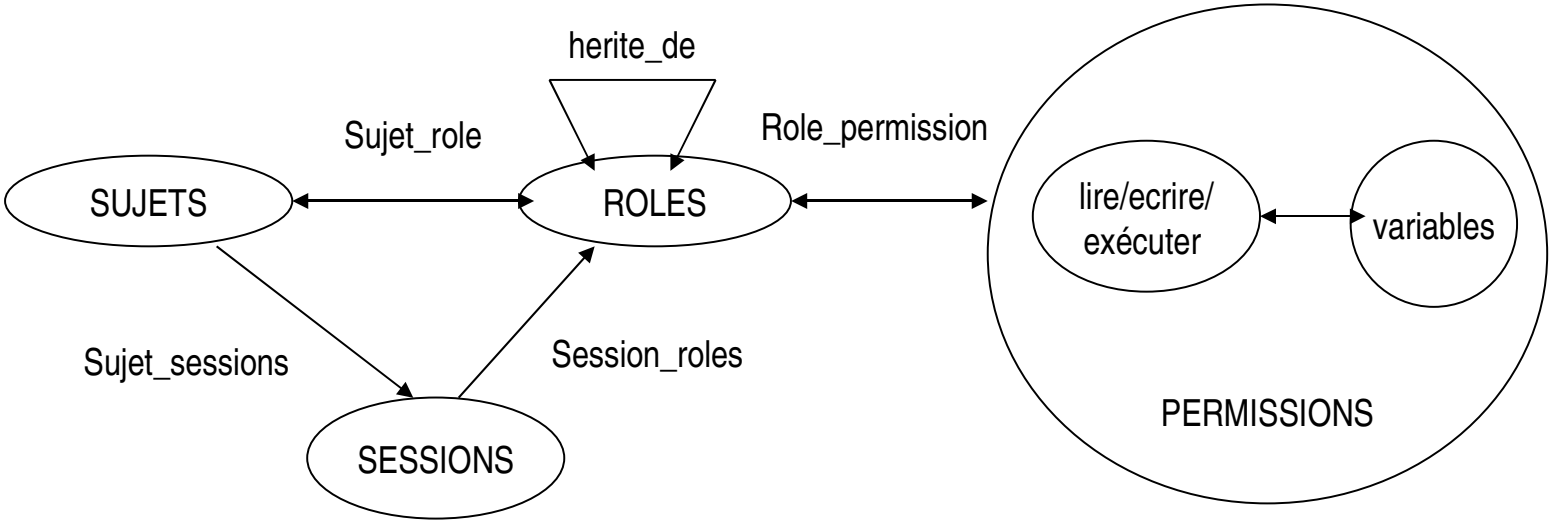
```
/* on ne veut pas avoir un cycle dans l'héritage*/
closure1(herite_de) /\ id(ROLE) = {}
&
/* conflit est symetrique */
conflit = conflit~
&
/*conflit est anti-réflexive*/
conflit /\ id(ROLE)={}
&
/* héritage et conflit sont des relations exclusives */
closure(herite_de) /\ conflit = {}
&
/* un sujet ne peut pas posséder 2 rôles en conflit */
((sujet_role ; closure(herite_de));conflit) /\ (sujet_role ; closure(herite_de))
= {}

&
sujet_role= {Liste_couple}
&
role_permission= {Liste_couple}
&
herite_de= {Liste_couple}
&
conflit={Liste_couple}

END
```

Le noyau de sécurité pour DEMONEY



```
MACHINE CONTROLE_ACCES_ProcessAPDU
 SEES Constantes,Politique_rbac_demoney
INCLUDES  ProcessAPDU
OPERATIONS
    INS_APDU_StoreData_c(su)=
    PRE
     TransEc=FALSE
     & su : SUJET
     & (sujet_role(su),INS_APDU_Select_p):(closure(herite_de); role_permission)
     THEN
     INS_APDU_StoreData
   END
END
```

L'aspect dynamique dans RBAC



L'aspect dynamique pour MECA la machine Connexion

```
MACHINE Connexion
SEES  Politique_rbac_demoney
VARIABLES
Connecte
INVARIANT
  Connecte : SUJET +-> ROLE & Connecte ⊆ (sujet_role ; closure(herite_de))
INITIALISATION
 Connecte := {}
OPERATIONS
   Connecter (su, ro) =
                PRE
                  su  : SUJET &
                  ro : ROLE &
                  (su,ro) : (sujet_role;closure(herite_de)) &
                   su /: dom(Connecte)
                THEN
                 Connecte:=Connecte\/{(su |-> ro)}
                END
 END
```

Le noyau de sécurité pour DEMONEY (aspect dynamique)

```
MACHINE CONTROLE_ACCES_ProcessAPDU
  SEES Constantes,Politique_rbac_demoney
  INCLUDES  ProcessAPDU, Connexion
  OPERATIONS

      INS_APDU_StoreData_c(su)=

      PRE
       TransEc=FALSE

        & su : SUJET

        & su : dom(Connecte)

        &  (Connecte(su), INS_APDU_Select_p) :(closure(herite_de);role_permission)
      THEN

        INS_APDU_StoreData

       END
    END
```

Outil MECA

- Composants donnés par l'utilisateur :
  - **APPLICATION : l'application fournie par l'utilisateur**
  - **POLITIQUE : machine modélisant la politique mise en œuvre (DAC, MAC, RBAC)**
- Composants résultats :
  - **CONTROLE_ACCES_APPLICATION :**
    - **les services sécurisés**
  - **APPLICATION_REF: raffinement de l'application en introduisant les appels aux services de lecture et d'écriture.**
  - **vérification du respect de la politique (obligations de preuve en rapport avec les appels d'opérations)**

    ---→ Granularité plus fine
- Tâches effectuées par MECA:
  - **3 formats d'entrée de la machine POLITIQUE : DAC, MAC et RBAC**
  - **Vérification de la cohérence de la politique de sécurité**
  - **Construction de la machine CONTROLE_ACCES**
  - **Construction du raffinement sécurisé**
  - **→ Utilisation de la BOB pour analyser et construire des composants B.**