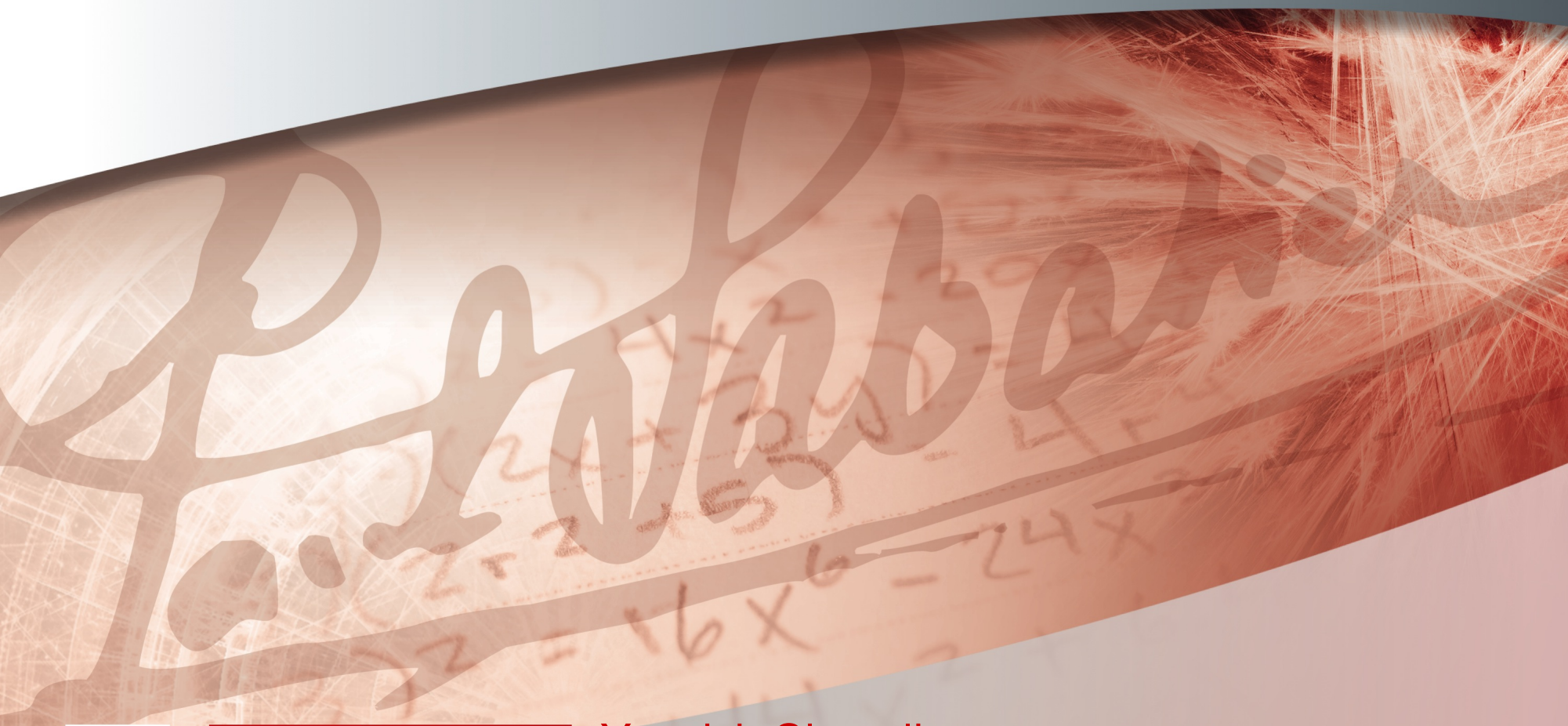
Modèles pour le Contrôle d'accès



Yannick Chevalier
Université de Toulouse
CSA M1, Security

Présentation

Nomenclature

Vocabulaire pour parler plus préciséments de systèmes de contrôle d'accès

Différents types de contrôle d'accès

1. Matrice de contrôle d'accès (modèle HRU)
2. Listes de contrôle d'accès (ACL)
3. RBAC

PLAN

Domaine

Objet

- élément du SI contenant ou recevant de l'information
- exemples : enregistrement et champ (BD), blocs, pages, segments (mémoire), fichiers, répertoires (système de fichier), programmes, périphériques (vidéo, son, réseau)

Sujet

- entité causant un flux d'information entre objets
- exemples : processus, personne, périphérique

Opération

- séquence d'instructions demandée par un sujet
- exemples : lecture, écriture, exécution, opérations ReST, création d'une table ou d'une base de données

Permission

Définition

Une permission est une autorisation d'effectuer une opération qui affecte un objet.

Note

- un clerc de banque n'a pas la permission de faire des retraits ou des ajouts (opérations)
- par contre, pour chaque compte de ses clients, il a la permission de faire ces opérations

MATRICE DE CONTRÔLE D'ACCÈS

MODÈLE HRU [76]

Matrice dans laquelle :

- chaque ligne représente un sujet
- chaque colonne représente un objet
- chaque case contient l'ensemble des opérations sur l'objet qui sont autorisées pour le sujet

Exemple :

	o_1	o_2	o_3
alice	r,w	r	r,x
bob	r	r	

Listes de contrôle d'accès (ACL)

Définition

- liste associée à un objet qui spécifie tous les sujets pouvant accéder à l'objet ainsi que leurs droits
- Une telle liste est attachée à un objet, et ses éléments sont des couples (sujet, {ens. de permissions})

Exemple :

	o_1	o_2	o_3
alice	r,w	r	r,x
bob	r	r	

LISTES DE CONTRÔLE D'ACCÈS (ACL)

DÉFINITION

- liste associée à un objet qui spécifie tous les sujets pouvant accéder à l'objet ainsi que leurs droits
- Une telle liste est attachée à un objet, et ses éléments sont des couples (sujet, {ens. de permissions})

Exemple :

	o_1	o_2	o_3
alice	r,w	r	r,x
bob	r	r	

ACL de o_1 : [(alice,{r,w}),(bob,{r})]

LISTES DE CONTRÔLE D'ACCÈS (ACL)

DÉFINITION

- liste associée à un objet qui spécifie tous les sujets pouvant accéder à l'objet ainsi que leurs droits
- Une telle liste est attachée à un objet, et ses éléments sont des couples (sujet, {ens. de permissions})

Exemple :

	o_1	o_2	o_3
alice	r,w	r	r,x
bob	r	r	

ACL de o_2 : [(alice,{r}),(bob,{r})]

Listes de contrôle d'accès (ACL)

Définition

- liste associée à un objet qui spécifie tous les sujets pouvant accéder à l'objet ainsi que leurs droits
- Une telle liste est attachée à un objet, et ses éléments sont des couples (sujet, {ens. de permissions})

Exemple :

	o_1	o_2	o_3
alice	r,w	r	r,x
bob	r	r	

ACL de o_3 : [(alice,{r,x}),(bob,$\emptyset$)]

ACL SOUS LINUX

SYNTAXE

```
setfacl mode <-d> type:nom:perm
```

- mode : -m (modifier) ou -x (supprimer)
- -d : pour les répertoires uniquement, indique les ACL des fichiers qui seront créés
- type : u our user, g pour groupe
- nom : nom de l'utilisateur ou du groupe
- perm : rwx

CAPACITÉS

DÉFINITION

les capacités d'un utilisateur sont toutes les permissions de cet utilisateur

EXEMPLES

- historiquement trop lourd à implémenter dans des systèmes d'exploitation pour des utilisations réelles
- Sous Linux, chaque capabilities est un ensembles de capacités
- Permissions sous Android (les applications sont des sujet)
- Fuchsia (OS Google, en cours de développement) semble utiliser les capacités

Exemple :

	o_1	o_2	o_3
alice	r,w	r	r,x
bob	r	r	

CAPACITÉS

DÉFINITION

les capacités d'un utilisateur sont toutes les permissions de cet utilisateur

EXEMPLES

- historiquement trop lourd à implémenter dans des systèmes d'exploitation pour des utilisations réelles
- Sous Linux, chaque capabilities est un ensembles de capacités
- Permissions sous Android (les applications sont des sujet)
- Fuchsia (OS Google, en cours de développement) semble utiliser les capacités

Exemple :

	o_1	o_2	o_3
alice	r,w	r	r,x
bob	r	r	

Capacités de alice : $[(o_1,\{r,w\}),(o_2,\{r\}),(o_3,\{r,x\})]$

CAPACITÉS

DÉFINITION

les capacités d'un utilisateur sont toutes les permissions de cet utilisateur

EXEMPLES

- historiquement trop lourd à implémenter dans des systèmes d'exploitation pour des utilisations réelles
- Sous Linux, chaque capabilities est un ensembles de capacités
- Permissions sous Android (les applications sont des sujet)
- Fuchsia (OS Google, en cours de développement) semble utiliser les capacités

Exemple :

	o_1	o_2	o_3
alice	r,w	r	r,x
bob	r	r	

Capacités de bob : $[(o_1,\{r\}),(o_2,\{r\}),(o_3,\emptyset)]$

Domaines et types

Domaine

regroupement de sujets (dans un groupe ou un rôle)

Type

regroupement d'objets

Utilité

- on peut utiliser des domaines et/ou des types pour obtenir une politique de contrôle d'accès plus simple
- exemple : domaine $\Rightarrow$ rôle dans une application Web, type $\Rightarrow$ classe d'un modèle

PLAN

CONCEPTS

MODALITÉS DU CONTRÔLE D'ACCÈS

RBAC

Contrôle d'accès discrétionnaire

Contrôle d'accès discrétionnaire (DAC—ACL, capacités)

Chaque objet est la propriété d'un sujet qui définit les droits d'accès sur cet objet

Contrôle d'accès mandataire (MAC—Bell-LaPadula, Biba)

Tous les objets sont la propriété virtuelle d'un sujet hors du système qui délègue les permissions aux différents sujets

Contrôle d'accès non-discrétionnaire (nouveau)

Le système est en principe discrétionnaire, mais des règles limitent les possibilités des utilisateurs :

1. en fonction de leurs actions passées (History-BAC)
2. en fonction de règles de séparation de tâche (SoD-BAC)
3. en fonction de rôles (Role-BAC, RBAC)

En général, on adopte en fonction du contexte un mélange de ces différents types de politiques.

Évaluation DAC

Limites des ACL

- il est facile de savoir qui a accès à une ressource donnée
- il est très difficile, voire impossible, de connaître les permissions d'un sujet

Limites des capacités

- il est facile de savoir les ressources auxquelles un sujet a accès
- il est très difficile, voire impossible, de connaître les permissions associées à un objet

Suite

Exploration de systèmes non-mandataires plus adaptés pour la sécurisation de grands systèmes d'information

PLAN

RBAC

Types de RBAC

- Core : rôle = groupe
- Hiérarchique : introduction d'une hiérarchie de rôle (et d'héritage des permissions)
- Contraint : introduction de contraintes de séparation de tâche (un sujet ne peut pas jouer dans 2 équipes pendant un même match)
- On ne s'intéresse qu'aux deux premiers cas

Core RBAC

Principe

- Classification des sujets : dès qu'il est authentifié, un sujet acquiert plusieurs rôles (relation `Role` entre sujets et rôles)
- Politique de contrôle d'accès : une relation `Perm` relie les rôles, les actions, et les objets
- Application : un sujet s a la permission de faire l'opération a sur l'objet o si s a un rôle r tel que $\texttt{Perm}(r, a, o)$ est vrai

Formalisation logique

$$\texttt{Autorise}(s, a, o) \leftarrow \texttt{Role}(s, r), \texttt{Perm}(r, a, o)$$

RBAC HIÉRARCHIQUE

EN PLUS DE CORE

PRINCIPE

- héritage : si $r \leq r'$ alors r' peut faire tout ce que peut faire r
- On peut de passer de la relation `Role` en disant que tout sujet s a un rôle s (c'est le cas pour les bases de données)
- Politique de contrôle d'accès : une relation `Perm` relie les rôles, les actions, et les objets
- Application : un sujet s a la permission de faire l'opération a sur l'objet o si s a un rôle r **et qu'il existe un rôle** $r' \leq r$ tel que $\texttt{Perm}(r', a, o)$ est vrai

FORMALISATION LOGIQUE

$$\texttt{Autorise}(s, a, o) \leftarrow \texttt{Role}(s, r), \texttt{Perm}(r', a, o), r' \leq r$$

Limitations de RBAC

Ingénierie des rôles

- l'écriture de politiques RBAC (hiérarchiques) suppose une certaine uniformité
- i.e., le passage par la relation `Role` doit réduire le nombre de permissions `Perm` à gérer
- ingénierie des rôles : trouver les rôles qui permettent la meilleure réduction possible

Politique des moindres privilèges

- but de la mise en place d'un système de contrôle accès : que chaque sujet ne puisse faire que ce dont il a besoin
- tentation (compréhensible) de simplifier le contrôle d'accès en ayant des rôles plus larges que nécessaire pour regrouper (uniformiser) différents cas
- d'où un conflit avec l'ingénierie des rôles