Organisation des droits d'accès au système

Comment modéliser qui a droit à quoi ?

Romuald Thion

romuald.thion@liris.cnrs.fr
Laboratoire LIRIS, équipe BD
Lyon, France

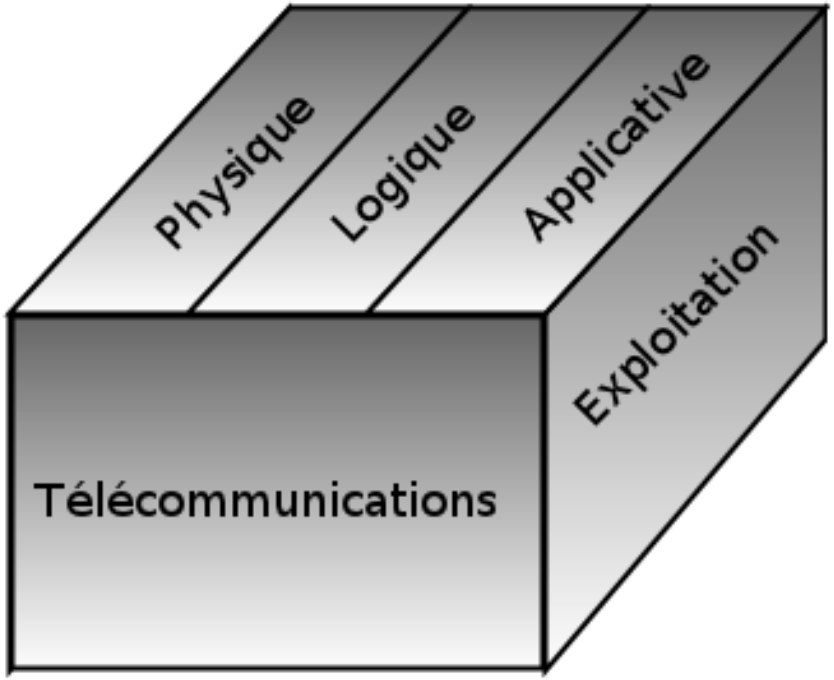
Lundi 24 octobre 2011
Tutoriel BDA'11 – Rabat

La sécurité en actualité

- CIA, Mossad, Also Targeted in Massive DigiNotar Cert Breach
- Researchers Release Attack Tool That Cripples Secure Websites
- RSA Blames Breach on Two Hacker Clans Working for Unnamed Government
- Who Else Was Hit by the RSA Attackers ?
- Hack Areva : analyse . . . à chaud
- Les sous-traitants des armées US leurs disent Bercy

La sécurité est une question de rapport coût/bénéfice

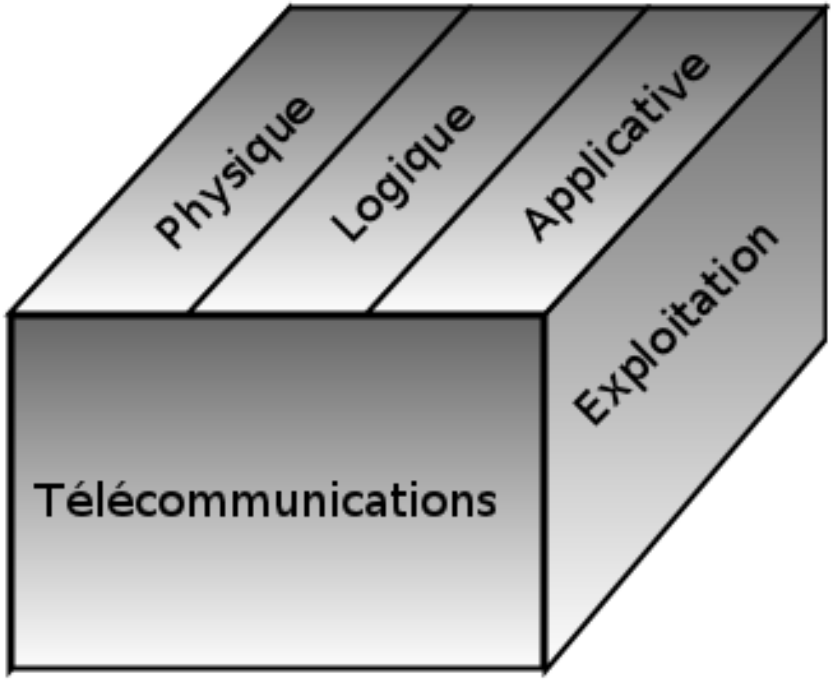
Domaines d'application de la sécurité



Dans ce tutoriel :

- sécurité logique ;
- confidentialité ;
- autorisations.

Domaines d'application de la sécurité



Dans ce tutoriel :

- sécurité logique ;
- confidentialité ;
- autorisations.

Ce tutoriel

Autorisation (ou contrôle d'accès)

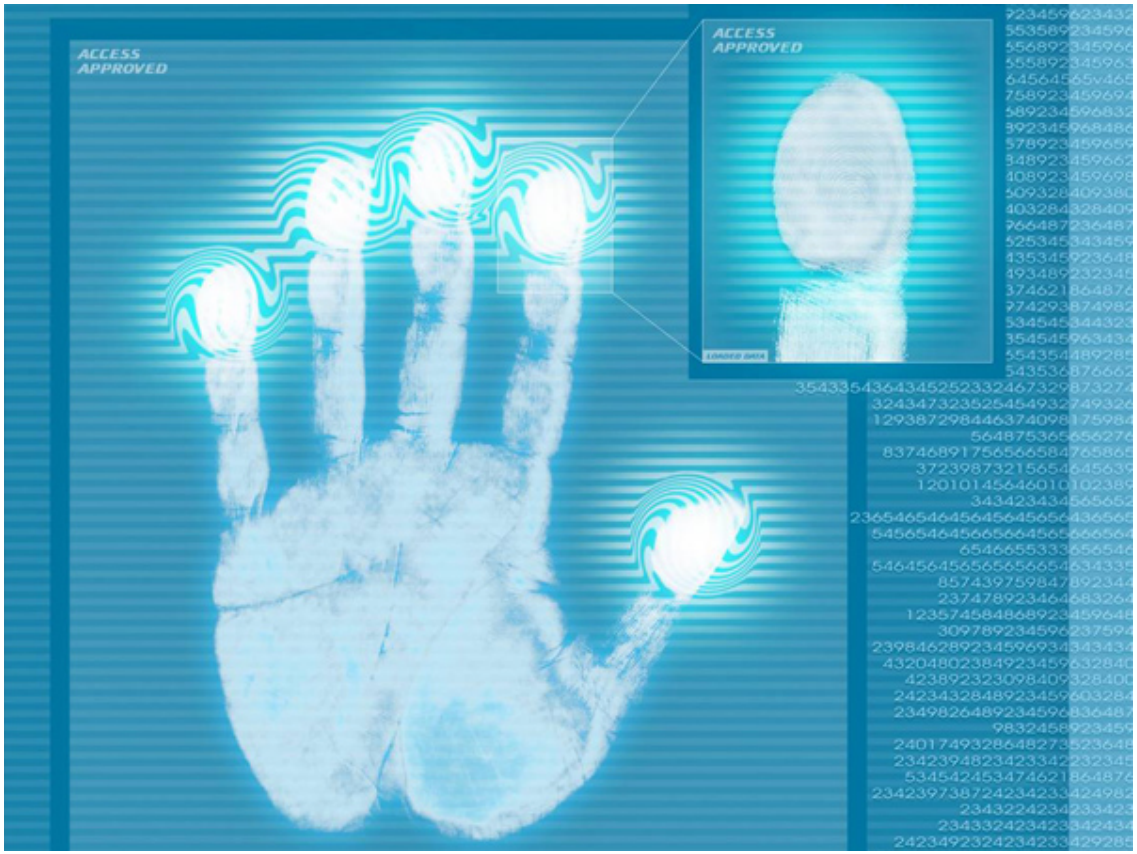
- préalable : authentification (c.f. protocole cryptographique)
- ensuite : autorisation

Le contrôle d'accès est *un* des mécanismes pour garantir la *confidentialité* de l'information

Objectifs

- Connaître les principes de l'organisation des droits ;
- Organiser les droits avec les rôles ;
- Organiser des droits dans les systèmes distribués ;

Authentification



Authentification

Un protocole cryptographique repose sur :

- chiffrement symétrique
- chiffrement asymétrique
- hachage
- génération de nombres aléatoires

Afin de garantir des propriétés de sécurité des échanges :

Confidentialité Seuls les légitimes ont accès à l'information

Intégrité Le message n'a pas été altéré

Authenticité On communique bien à la bonne personne

Non-répudiation On ne peut pas nier sa participation

Exemple d'authentification

Un protocole d'authentification non-cryptographique

`http://www.pokerstars.fr/nouvelle-procedure-inscription/`

1. enregistrement des données personnelles et login/pass
2. vérification de l'email
3. envoi informations bancaire et carte d'identité par courrier sous 30j
4. réception d'un code d'activation par courrier
5. saisie du code

Autorisation : une fois sur le site, qui peut écrire dans la chat-room, s'inscrire à une partie, prendre un siège, kick/ban un joueur...

Exemple d'authentification

Un protocole d'authentification non-cryptographique

`http://www.pokerstars.fr/nouvelle-procedure-inscription/`

1. enregistrement des données personnelles et login/pass
2. vérification de l'email
3. envoi informations bancaire et carte d'identité par courrier sous 30j
4. réception d'un code d'activation par courrier
5. saisie du code

Autorisation : une fois sur le site, qui peut écrire dans la chat-room, s'inscrire à une partie, prendre un siège, kick/ban un joueur...

Autorisation

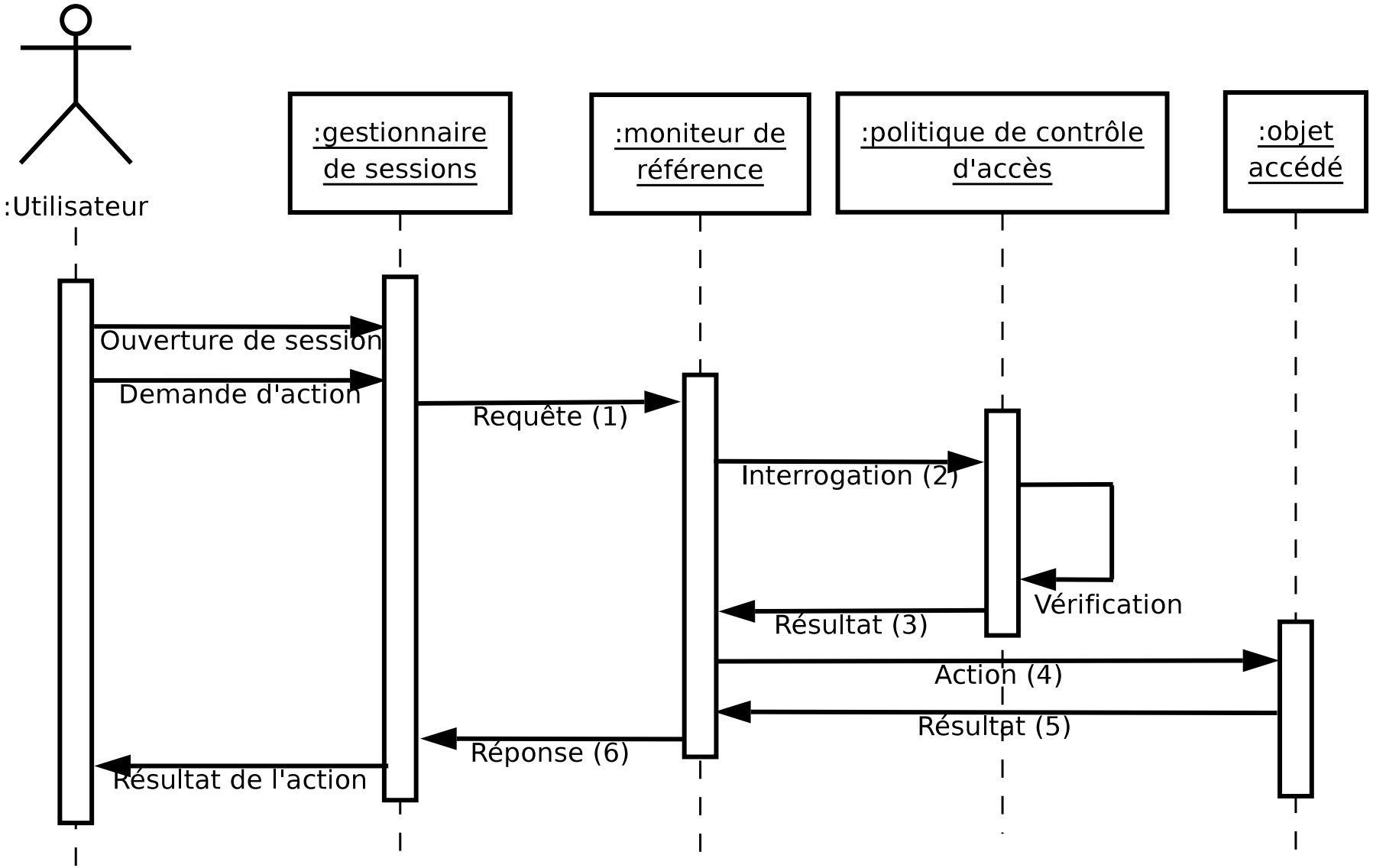
Contrôle d'accès : définition

Le contrôle d'accès est un mécanisme grâce auquel un système autorise ou interdit les actions demandées par des sujets sur des objets

Contrôle d'accès

- un moyen mis en œuvre *a priori*,
- via un moniteur,
- pour s'assurer de la légitimité des accès,
- selon une politique déterminée ($\cong$ *la loi*).

Le moniteur


:Utilisateur
:gestionnaire de sessions
:moniteur de référence
:politique de contrôle d'accès
:objet accédé
Ouverture de session
Demande d'action
Requête (1)
Interrogation (2)
Vérification
Résultat (3)
Action (4)
Résultat (5)
Réponse (6)
Résultat de l'action

Caractéristiques du moniteur

Un moniteur logiciel doit respecter les propriétés suivantes (NEAT) :

Non-bypassable the security functions cannot be circumvented

Evaluatable the security functions small enough and simple enough to be mathematically verified and evaluated

Always invoked the security functions are invoked each and every time

Tamperproof subversive code cannot alter the function of the security functions by exhausting resources, overrunning buffers, or other forms of making the security software fail

Gestion des autorisations

Principes généraux

- classification selon sensibilité,
- moindre privilège,
- séparation des tâches,
- conflits d'intérêts.

- modèles discrétionnaires,
- modèles mandataires (contrôle de flux),
- modèles à rôles,
- modèles *ad hoc*,
- modèles issus de la recherche.

Policitique de contrôle d'accès

Exemple académique

Un professeur gère l'accès à ses cours :

- donne des accès nominativement (identifiants),
- donne des accès à des agrégats de sujets (groupes).

Le professeur désire certaines propriétés :

- tous ses étudiants doivent avoir accès en lecture aux cours (disponibilité),
- aucun étudiant ne peut modifier les ressources (intégrité),
- les étudiants qui ne suivent pas son cours ne peuvent pas accéder aux ressources (confidentialité)

Policitique de contrôle d'accès

Exemple académique

Un professeur gère l'accès à ses cours :

- donne des accès nominativement (identifiants),
- donne des accès à des agrégats de sujets (groupes).

Le professeur désire certaines propriétés :

- tous ses étudiants doivent avoir accès en lecture aux cours (disponibilité),
- aucun étudiant ne peut modifier les ressources (intégrité),
- les étudiants qui ne suivent pas son cours ne peuvent pas accéder aux ressources (confidentialité)

Modèles discrétionnaires

Trusted Computer System Evaluation Criteria, DoD*, 1985* :

. . . a means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control).

Particularités

- modèles historiques,
- décentralisés,
- basés sur l'identité de l'accédant,
- difficiles à contrôler.

Modèles discrétionnaires

Trusted Computer System Evaluation Criteria, DoD*, 1985* :

> *. . . a means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control).*

Particularités

- modèles historiques,
- décentralisés,
- basés sur l'identité de l'accédant,
- difficiles à contrôler.

Matrice de contrôle d'accès

Exemple (systèmes de fichiers UNIX/LINUX)

	Fichier1 (A)	Fichier2 (B)	Fichier3 (C)	Fichier4 (D)
Alice	owns	r	r	
Bob	r	owns	r	owns
Charly	r	r	owns	rwx
Denise			r	r

Gestion des droits

Deux lectures

- permissions en lignes : *Capabilities List* (CL),
- permissions en colonnes : *Access Control List* (ACL).

Matrice sous forme de listes

```
GRANT owns
  ON : Fichier1
  TO : Alice.
[...]
GRANT r
  ON : Fichier3
  TO : Alice, Bob, Denise
  ON : Fichier2
  TO : Alice, Charly.
GRANT w
  ON : Fichier4
  TO : Charly.
```

```
SUBJECT Alice
  IS GRANTED owns
  ON : Fichier1
  IS GRANTED r
  ON : Fichier2, Fichier3.
[...]
SUBJECT Bob
  IS GRANTED r
  ON : Fichier1, Fichier3
  IS GRANTED owns
  ON : Fichier2, Fichier4.
```

Gestion des droits

Deux lectures

- permissions en lignes : *Capabilities List* (CL),
- permissions en colonnes : *Access Control List* (ACL).

Matrice sous forme de listes

```
GRANT owns
  ON : Fichier1
  TO : Alice.
[...]
GRANT r
  ON : Fichier3
  TO : Alice, Bob, Denise
  ON : Fichier2
  TO : Alice, Charly.
GRANT w
  ON : Fichier4
  TO : Charly.
```

```
SUBJECT Alice
  IS GRANTED owns
  ON : Fichier1
  IS GRANTED r
  ON : Fichier2, Fichier3.
[...]
SUBJECT Bob
  IS GRANTED r
  ON : Fichier1, Fichier3
  IS GRANTED owns
  ON : Fichier2, Fichier4.
```

Language concret (1/2)

Autorisations statiques LDAP

```
<access directive> ::= access to <what>
    [by <who> [<access>] [<control>] ]+
<what> ::= * |
    [dn[.<basic-style>]=<regex> | dn.<scope-style>=<DN>]
    [filter=<ldapfilter>] [attrs=<attrlist>]
<basic-style> ::= regex | exact
<scope-style> ::= base | one | subtree | children
<attrlist> ::= <attr> [val[.<basic-style>]=<regex>] | <attr> , <attrlist>
<attr> ::= <attrname> | entry | children
<who> ::= * | [anonymous | users | self
        | dn[.<basic-style>]=<regex> | dn.<scope-style>=<DN>]
    [dnattr=<attrname>]
    [group[/<objectclass>[/<attrname>][.<basic-style>]]=<regex>]
    [peername[.<basic-style>]=<regex>]
    [sockname[.<basic-style>]=<regex>]
    [domain[.<basic-style>]=<regex>]
    [sockurl[.<basic-style>]=<regex>]
    [set=<setspec>]
    [aci=<attrname>]
<access> ::= [self]{<level>|<priv>}
<level> ::= none | disclose | auth | compare | search | read | write | manage
<priv> ::= {=|+|-}{m|w|r|s|c|x|d|0}+
<control> ::= [stop | continue | break]
```

Language concret (2/2)

Types de privilèges SGBDR

- privilèges niveau système (opérations sur la base)
- privilèges niveau objet :
  - {SELECT, INSERT, UPDATE, ...} sur les *tables*.
  - {SELECT, INSERT, UPDATE, ...} sur les *vues*.

```
GRANT <action> ON <objet> TO <sujet> [WITH GRANT OPTION]

    <action> ::= ALL |  SELECT | DELETE |
        INSERT [<attribut>]| UPDATE [<attribut>]|
        REFERENCE [<attribut>]
    <objet> ::= TABLE <relation> * | VIEW <vue> *
    <sujet> ::= PUBLIC| <sujet> *
```

Modèles mandataires

Contrôle mandataire – MAC

Le contrôle d'accès mandataire est exprimé en termes de niveaux de sécurité associés aux sujets et aux objets et à partir desquels sont dérivés les actions autorisées. Il vise à contrôler le flux de l'information entre les classes de confidentialité.

Particularités

- Niveau d'indirection intermédiaire,
- Fortement centralisés,
- Rigides mais évaluables,
- Mise en œuvre dans les langages (e.g., typage).

Modèles mandataires

Contrôle mandataire – MAC

Le contrôle d'accès mandataire est exprimé en termes de niveaux de sécurité associés aux sujets et aux objets et à partir desquels sont dérivés les actions autorisées. Il vise à contrôler le flux de l'information entre les classes de confidentialité.

Particularités

- Niveau d'indirection intermédiaire,
- Fortement centralisés,
- Rigides mais évaluables,
- Mise en œuvre dans les langages (e.g., typage).

Dérivations des autorisations

Classes de sécurité (simplifié)

- Organisation de niveaux en ordre total ou treillis :
  - attribution de niveaux aux *sujets* : accréditation,
  - attribution de niveaux aux *objets* : classification,
- Règles de dérivations de autorisations.

Règle	Description
No read up [a]	Un sujet accrédité d'un niveau donné ne peut pas accéder en lecture à des objets d'un niveau plus élevé
No write down [b]	Un sujet accrédité d'un niveau donné ne peut pas accéder en écriture à des objets d'un niveau moins élevé

a. Simple security rule
b. $\star$-property

Classification française

Classification	Description
Très Secret-Défense	Le niveau *Très Secret-Défense* est réservé aux informations ou supports protégés dont la divulgation est de nature à nuire très gravement à la défense nationale et qui concernent les priorités gouvernementales en matière de défense.
Secret-Défense	Le niveau *Secret-Défense* est réservé aux informations ou supports protégés dont la divulgation est de nature à nuire gravement à la défense nationale.
Confidentiel-Défense	Le niveau *Confidentiel-Défense* est réservé aux informations ou supports protégés dont la divulgation est de nature à nuire à la défense nationale ou pourrait conduire à la découverte d'un secret de la défense nationale classifié au niveau *Très Secret-Défense* ou *Secret-Défense*.

Vers la flexibilité

Insuffisances des modèles classiques

- nouvelles dimensions,
- nouveaux usages,
- nouvelles organisations.

Vers de nouveaux modèles

- coût et flexibilité de l'administration,
- composition sur mesure,
- nouveaux objectifs

Vers la flexibilité

Insuffisances des modèles classiques

- nouvelles dimensions,
- nouveaux usages,
- nouvelles organisations.

Vers de nouveaux modèles

- coût et flexibilité de l'administration,
- composition sur mesure,
- nouveaux objectifs

Contrôle d'accès à rôles

Limites des modèles existants

- systèmes avec de nombreux utilisateurs,
- systèmes avec de nombreux objets,
- organisation/administration flexibles.

Approche

Politique proche de l'organisation qui l'utilise :

- RBAC est basé sur l'identification de rôles,
- les permissions sont associées aux rôles,
- les utilisateurs endossent des rôles aux travers de sessions.

Contrôle d'accès à rôles

Limites des modèles existants

- systèmes avec de nombreux utilisateurs,
- systèmes avec de nombreux objets,
- organisation/administration flexibles.

Approche

Politique proche de l'organisation qui l'utilise :

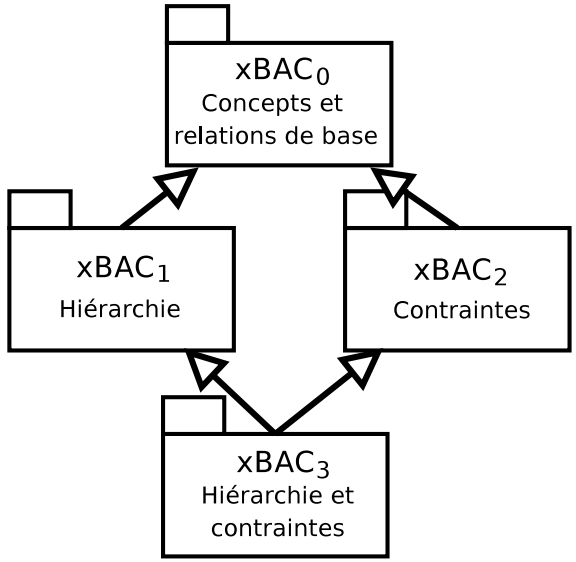
- RBAC est basé sur l'identification de rôles,
- les permissions sont associées aux rôles,
- les utilisateurs endossent des rôles aux travers de sessions.

Famille standard

Modèles standards

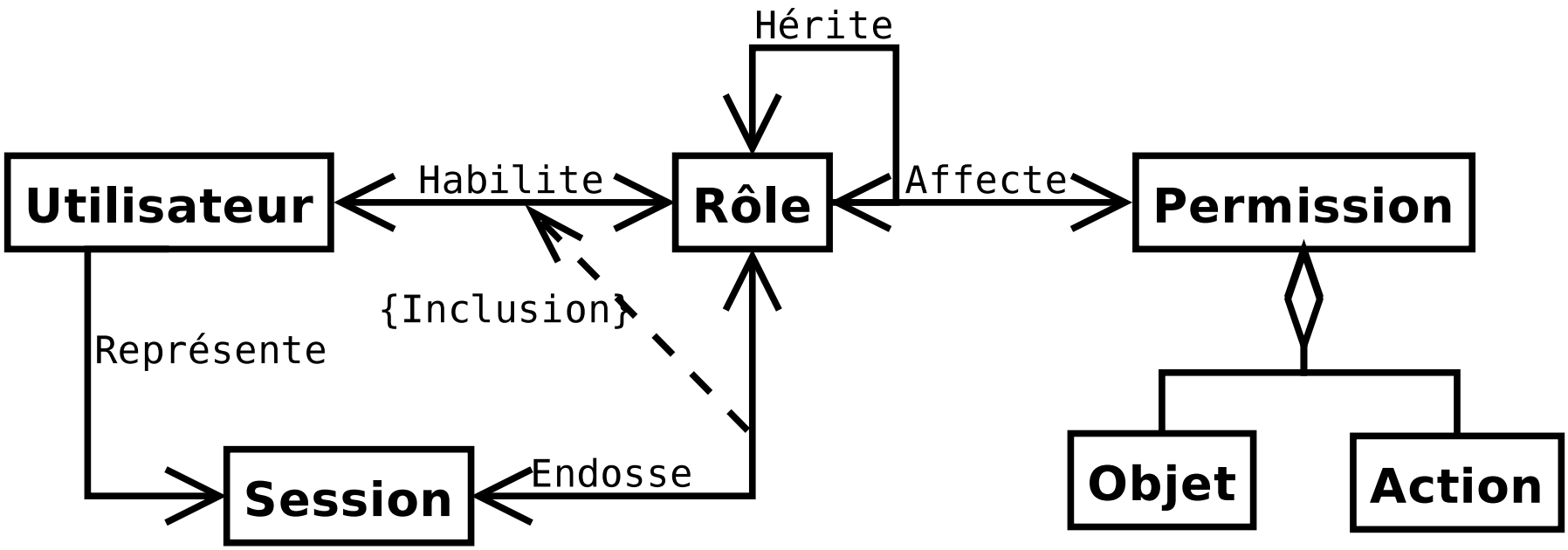
Plusieurs modèles RBAC définis dans le standard ANSI :

- RBAC_0 : le noyau,
- RBAC_1 : la hiérarchie,
- RBAC_2 : les contraintes,
- RBAC_3 : hiérarchies et contraintes.



RBAC$_0$ et RBAC$_1$

. . . a job function or job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role.



Formalisation ensembliste

	Notation	Description
Concepts	$\mathcal{U}$	ensemble fini d'utilisateurs
	$\mathcal{R}$	ensemble fini de rôles
	$\mathcal{A}$	ensemble fini d'actions
	$\mathcal{O}$	ensemble fini d'objets
	$\mathcal{S}$	ensemble fini de sujets (sessions)
Relations	$\mathcal{P} \subseteq \mathcal{O} \times \mathcal{A}$	une action sur un objet
	$\mathcal{URA} \subseteq \mathcal{U} \times \mathcal{R}$	affectation de rôles aux utilisateurs
	$\mathcal{PRA} \subseteq \mathcal{R} \times \mathcal{P}$	affectation de permissions aux rôles
	$\mathcal{SU} \subseteq \mathcal{S} \times \mathcal{U}$	relation entre sessions et utilisateurs
	$\mathcal{SR} \subseteq \mathcal{S} \times \mathcal{R}$	relation entre sessions et rôles
	$\mathcal{RH} \subseteq \mathcal{R} \times \mathcal{R}$	relation d'héritage entre rôles

C'est le schéma d'une BD (relationnelle)

Formalisation ensembliste

	Notation	Description
Concepts	$\mathcal{U}$	ensemble fini d'utilisateurs
	$\mathcal{R}$	ensemble fini de rôles
	$\mathcal{A}$	ensemble fini d'actions
	$\mathcal{O}$	ensemble fini d'objets
	$\mathcal{S}$	ensemble fini de sujets (sessions)
Relations	$\mathcal{P} \subseteq \mathcal{O} \times \mathcal{A}$	une action sur un objet
	$\mathcal{URA} \subseteq \mathcal{U} \times \mathcal{R}$	affectation de rôles aux utilisateurs
	$\mathcal{PRA} \subseteq \mathcal{R} \times \mathcal{P}$	affectation de permissions aux rôles
	$\mathcal{SU} \subseteq \mathcal{S} \times \mathcal{U}$	relation entre sessions et utilisateurs
	$\mathcal{SR} \subseteq \mathcal{S} \times \mathcal{R}$	relation entre sessions et rôles
	$\mathcal{RH} \subseteq \mathcal{R} \times \mathcal{R}$	relation d'héritage entre rôles

C'est le schéma d'une BD (relationnelle)

Exemple de politique

Politique RBAC de la matrice jouet ▸ voir la matrice

URA

	I	M	G	P	S
Alice	×	×			
Bob	×		×		
Charly	×			×	
Denise					×

PRA

	r1	w1	**r2**	w2	**r3**	w3	r4	w4	x4
Infirmier	×		×		×				
Médecin		×							
Gastrologue				×			×	×	×
Pédiatre						×	×	×	×
Secrétaire					×		×		

Exemple de politique

Politique RBAC de la matrice jouet ▸ voir la matrice

URA

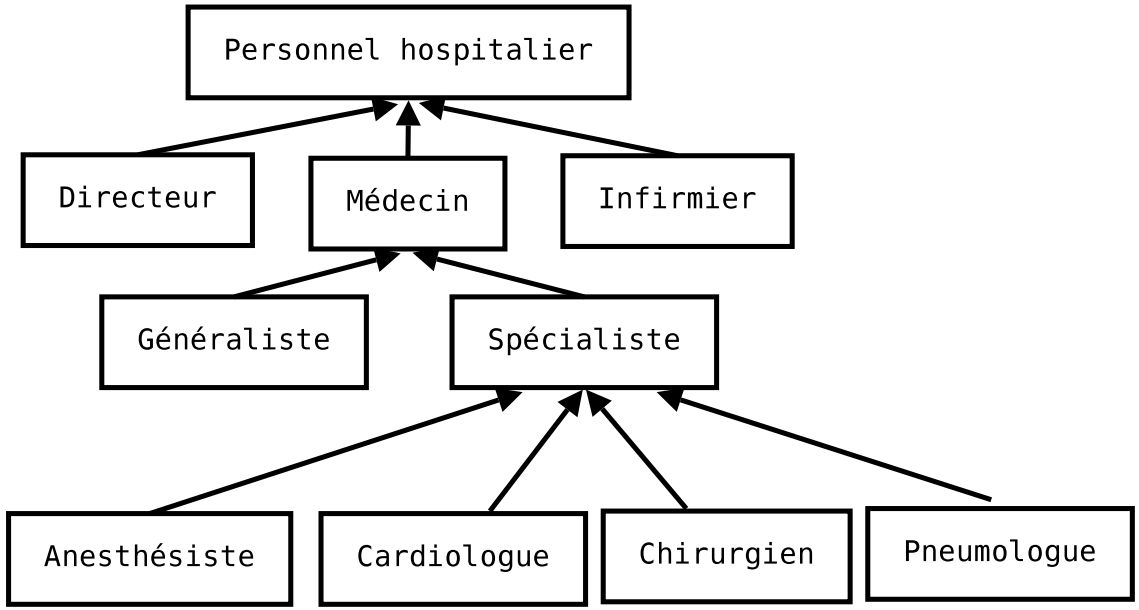
	I	M	G	P	S
Alice	×	×			
Bob	×		×		
Charly	×			×	
Denise					×

PRA

	r1	w1	**r2**	w2	**r3**	w3	r4	w4	x4
Infirmier	×		×		×				
Médecin		×							
Gastrologue				×			×	×	×
Pédiatre						×	×	×	×
Secrétaire					×		×		

Hiérarchisation des rôles

Une hiérarchie en arbre



Plus expressif, plus complexe

- Nécessite de parcourir la hiérarchie ;
- Problème avec SGBDR ;
- Possible de restreindre l'expressivité.

Déterminer les autorisations en présence de hiérarchie

Utilisateurs affectés et autorisés

$assigned\_users : \mathcal{R} \rightarrow 2^{\mathcal{U}}$ et $auth\_users : \mathcal{R} \rightarrow 2^{\mathcal{U}}$:

$$assigned\_users\ (\mathsf{r}) = \{u \in \mathcal{U} \mid (u, r) \in \mathcal{URA}\}$$

$$auth\_users\ (\mathsf{r}) = \{u \in \mathcal{U} \mid r' \succeq r \wedge (u, r') \in \mathcal{URA}\}$$

$$\forall r \in \mathcal{R}\ assigned\_users(r) \subseteq auth\_users(r)$$

Permissions affectées et autorisées

$assigned\_perms : \mathcal{R} \rightarrow 2^{\mathcal{P}}$ et $auth\_perms : \mathcal{R} \rightarrow 2^{\mathcal{P}}$:

$$assigned\_perms\ (\mathsf{r}) = \{p \in \mathcal{P} \mid (r, p) \in \mathcal{PRA}\}$$

$$auth\_perms\ (\mathsf{r}) = \{p \in \mathcal{P} \mid r' \preceq r \wedge (r', p) \in \mathcal{PRA}\}$$

$$\forall r \in \mathcal{R}\ assigned\_perms(r) \subseteq auth\_perms(r)$$

Déterminer les autorisations en présence de hiérarchie

Utilisateurs affectés et autorisés

$assigned\_users : \mathcal{R} \rightarrow 2^{\mathcal{U}}$ et $auth\_users : \mathcal{R} \rightarrow 2^{\mathcal{U}}$:

$$assigned\_users\ (\mathsf{r}) = \{u \in \mathcal{U} \mid (u, r) \in \mathcal{URA}\}$$

$$auth\_users\ (\mathsf{r}) = \{u \in \mathcal{U} \mid r' \succeq r \wedge (u, r') \in \mathcal{URA}\}$$

$$\forall r \in \mathcal{R}\ assigned\_users(r) \subseteq auth\_users(r)$$

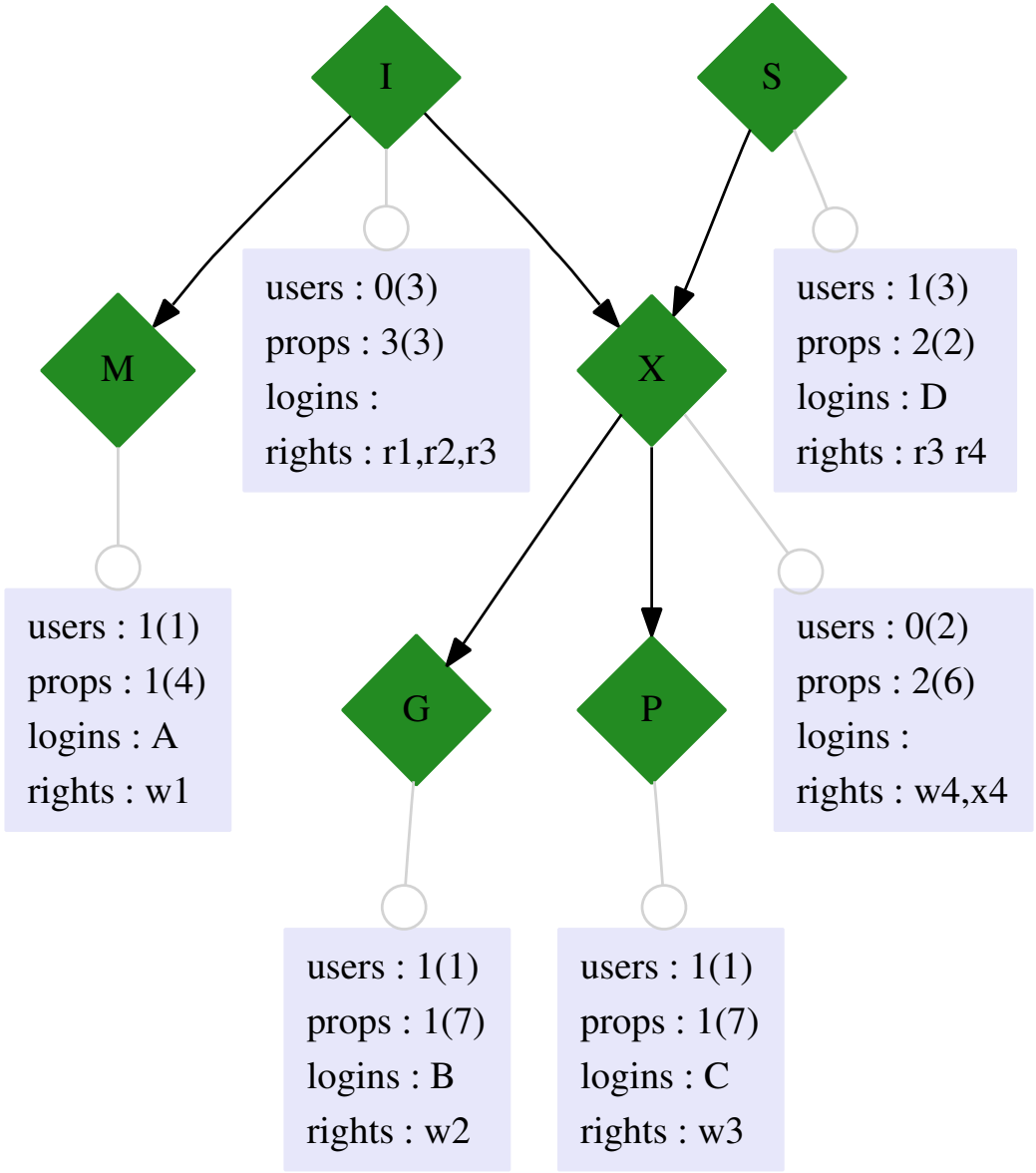
Permissions affectées et autorisées

$assigned\_perms : \mathcal{R} \rightarrow 2^{\mathcal{P}}$ et $auth\_perms : \mathcal{R} \rightarrow 2^{\mathcal{P}}$:

$$assigned\_perms\ (\mathsf{r}) = \{p \in \mathcal{P} \mid (r, p) \in \mathcal{PRA}\}$$

$$auth\_perms\ (\mathsf{r}) = \{p \in \mathcal{P} \mid r' \preceq r \wedge (r', p) \in \mathcal{PRA}\}$$

$$\forall r \in \mathcal{R}\ assigned\_perms(r) \subseteq auth\_perms(r)$$



Une hiérarchisation de la politique jouet ▸ voir la politique

Séparation des tâches

Principe de séparation des tâches

La *séparation des tâches* est un principe de sécurité qui impose que les acteurs qui interviennent dans la réalisation d'une tâche soient différents.

Mécanisme de contrôle

- Il faut qu'au moins n utilisateurs différents interviennent dans ce processus métier.
- Il ne peut pas y avoir $(n-1)$ utilisateurs qui disposent à eux seuls de l'ensemble des permissions pour effectuer l'ensemble du processus.

Séparation des tâches

Principe de séparation des tâches

La *séparation des tâches* est un principe de sécurité qui impose que les acteurs qui interviennent dans la réalisation d'une tâche soient différents.

Mécanisme de contrôle

- Il faut qu'au moins n utilisateurs différents interviennent dans ce processus métier.
- Il ne peut pas y avoir $(n-1)$ utilisateurs qui disposent à eux seuls de l'ensemble des permissions pour effectuer l'ensemble du processus.

Dans RBAC$_2$

Contrainte d'exclusion

- *séparation des tâches* : un principe, *objectif* à garantir,
- *exclusion mutuelle* : une contrainte, un *moyen*.

Exclusion mutuelle entre rôles

- relation binaire interne $\mathcal{MER}$,
- symétrique et irréflexive,
- sémantiques multiples :
  - *User-based separation of duty*.
  - *Permission-based separation of duty*.
  - *Object-based separation of duty*.

Dans RBAC$_2$

Contrainte d'exclusion

- *séparation des tâches* : un principe, *objectif* à garantir,
- *exclusion mutuelle* : une contrainte, un *moyen*.

Exclusion mutuelle entre rôles

- relation binaire interne $\mathcal{MER}$,
- symétrique et irréflexive,
- sémantiques multiples :
  - *User-based separation of duty*.
  - *Permission-based separation of duty*.
  - *Object-based separation of duty*.

Dans RBAC$_3$

Contraintes avec/sur la hiérarchie

- limites du standard,
- prérequis,
- spécialisation de l'héritage,

$$\forall r_1, r_2\ (r_1, r_2) \in \mathcal{MER} \Rightarrow auth\_users(r_1) \cap auth\_users(r_2) = \emptyset$$

Interactions

- $\forall r_1, r_2\ (r_1, r_2) \in \mathcal{MER} \Rightarrow \neg(r_1 \succeq r_2)$,
- $\forall r_1, r_2, r\ r \succeq r_1 \wedge r \succeq r_2 \Rightarrow (r_1, r_2) \notin \mathcal{MER}$,
- $\forall r_1, r_2, r\ (r_1, r_2) \in \mathcal{MER} \wedge r \succeq r_1 \Rightarrow (r, r_2) \in \mathcal{MER}$.

Dans RBAC$_3$

Contraintes avec/sur la hiérarchie

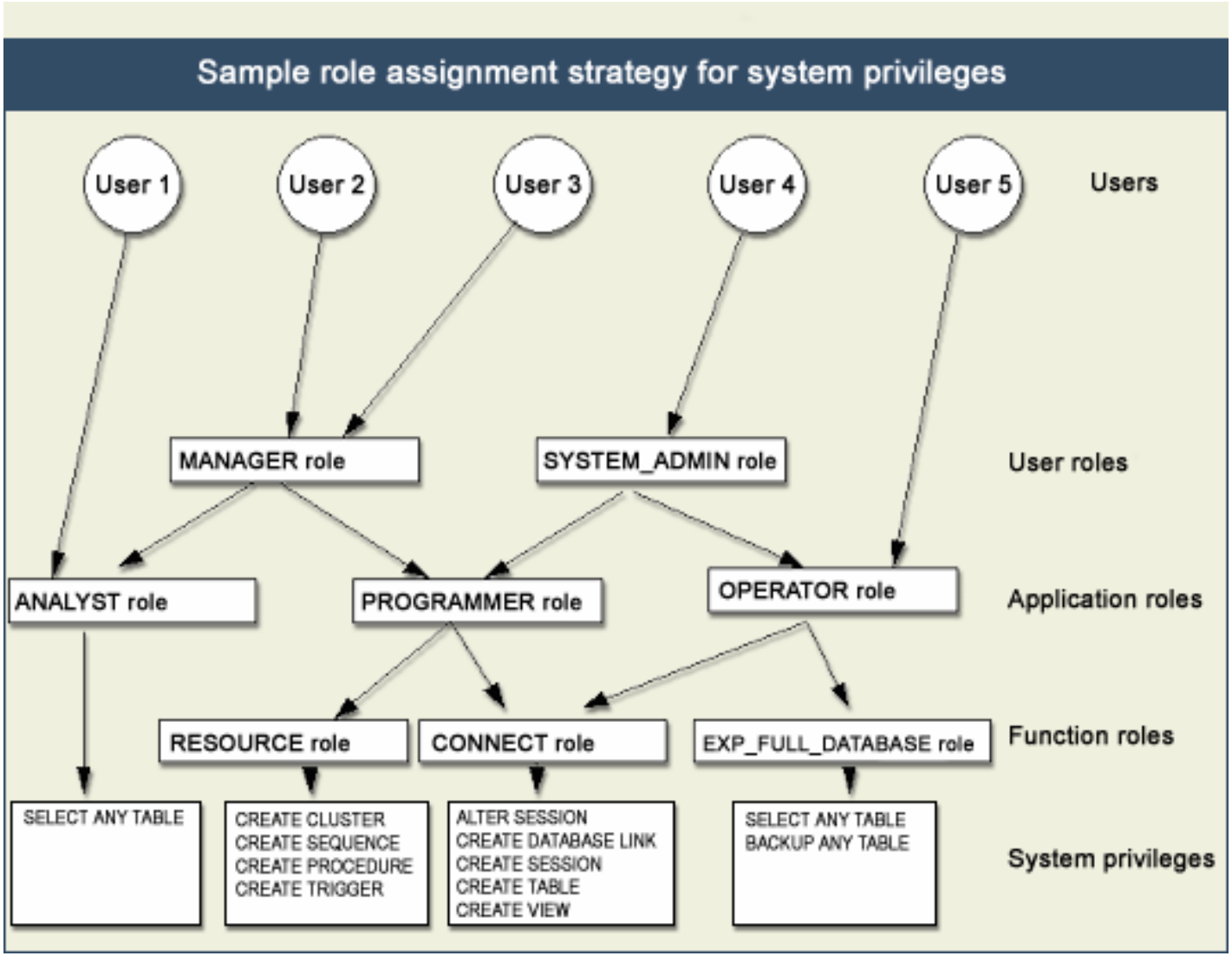
- limites du standard,
- prérequis,
- spécialisation de l'héritage,

$$\forall r_1, r_2 \ (r_1, r_2) \in \mathcal{MER} \Rightarrow auth\_users(r_1) \cap auth\_users(r_2) = \emptyset$$

Interactions

- $\forall r_1, r_2 \ (r_1, r_2) \in \mathcal{MER} \Rightarrow \neg(r_1 \succeq r_2)$,
- $\forall r_1, r_2, r \ r \succeq r_1 \wedge r \succeq r_2 \Rightarrow (r_1, r_2) \notin \mathcal{MER}$,
- $\forall r_1, r_2, r \ (r_1, r_2) \in \mathcal{MER} \wedge r \succeq r_1 \Rightarrow (r, r_2) \in \mathcal{MER}$.

RBAC dans Oracle (1/3)



RBAC dans Oracle (2/3)

Gestion des rôles

- création de rôles,
- affectation/révocation $\mathcal{PRA}$,
- affectation/révocation $\mathcal{URA}$,
- activation de rôles $\mathcal{SR}$.

```
CREATE ROLE <role> [NOT IDENTIFIED | IDENTIFIED
  [BY <password> | EXTERNALLY]];

GRANT/REVOKE <role> TO <sujet>;
GRANT/REVOKE <role> TO <role>;

ALTER USER <user> DEFAULT ROLE [EXCEPT] [<role> * | NONE];

SET ROLE [EXCEPT] [<role> * | NONE];
```

RBAC dans Oracle (3/3)

DBA_COL_PRIVS	Column object grants in the database.
DBA_ROLE_PRIVS	Roles granted to all users and roles
DBA_SYS_PRIVS	System priv. granted to users and roles.
DBA_TAB_PRIVS	Describes all object grants in the database.
DBA_TS_QUOTAS	Tablespace quotas for all users.
PRODUCT_PRIVS	Roles that are currently enabled.
ROLE_ROLE_PRIVS	Roles granted to other roles.
ROLE_SYS_PRIVS	System privileges granted to roles.
ROLE_TAB_PRIVS	Table privileges granted to roles.
SESSION_PRIVS	System privileges currently enabled.
COLUMN_PRIVILEGES	Depreciated use DBA_COL_PRIVS.
TABLE_PRIVILEGES	Depreciated use DBA_TAB_PRIVS.

Limites

- pas de contraintes de cardinalité
- pas de contraintes d'exclusion, ni statiques, ni dynamiques
- passage à l'échelle de la gestion des rôles

Au-dela d'RBAC

Autres concepts que le rôle

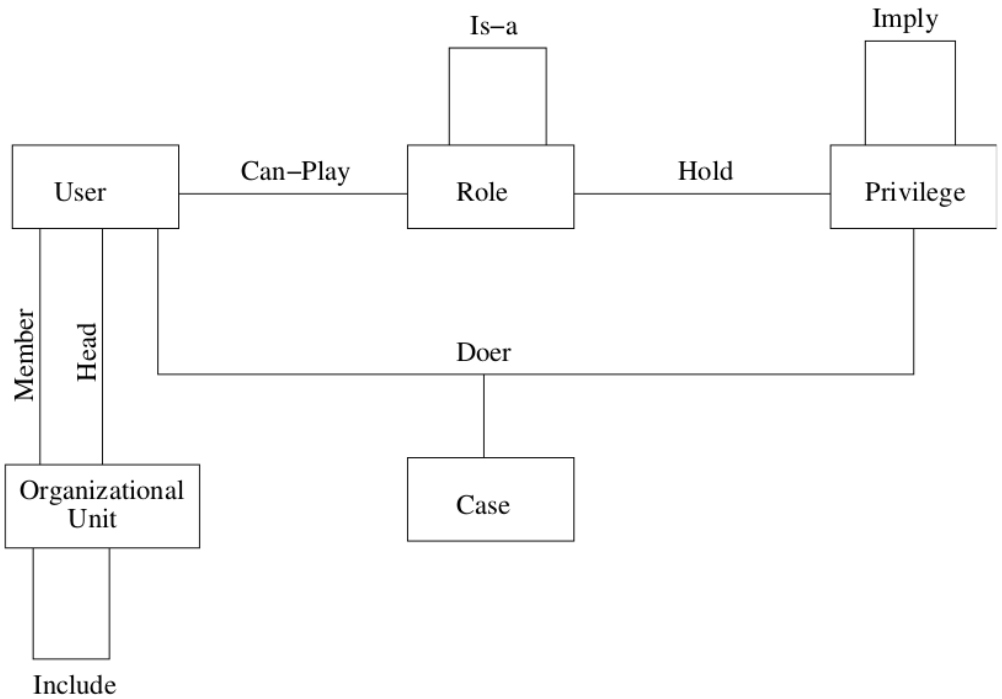
- tâches : Task-BAC,
- équipes : Team-BAC,
- espaces géographiques : Geographical-RBAC,
- dates & temps : Temporal-RBAC ...

Grands principes unificateurs... versus

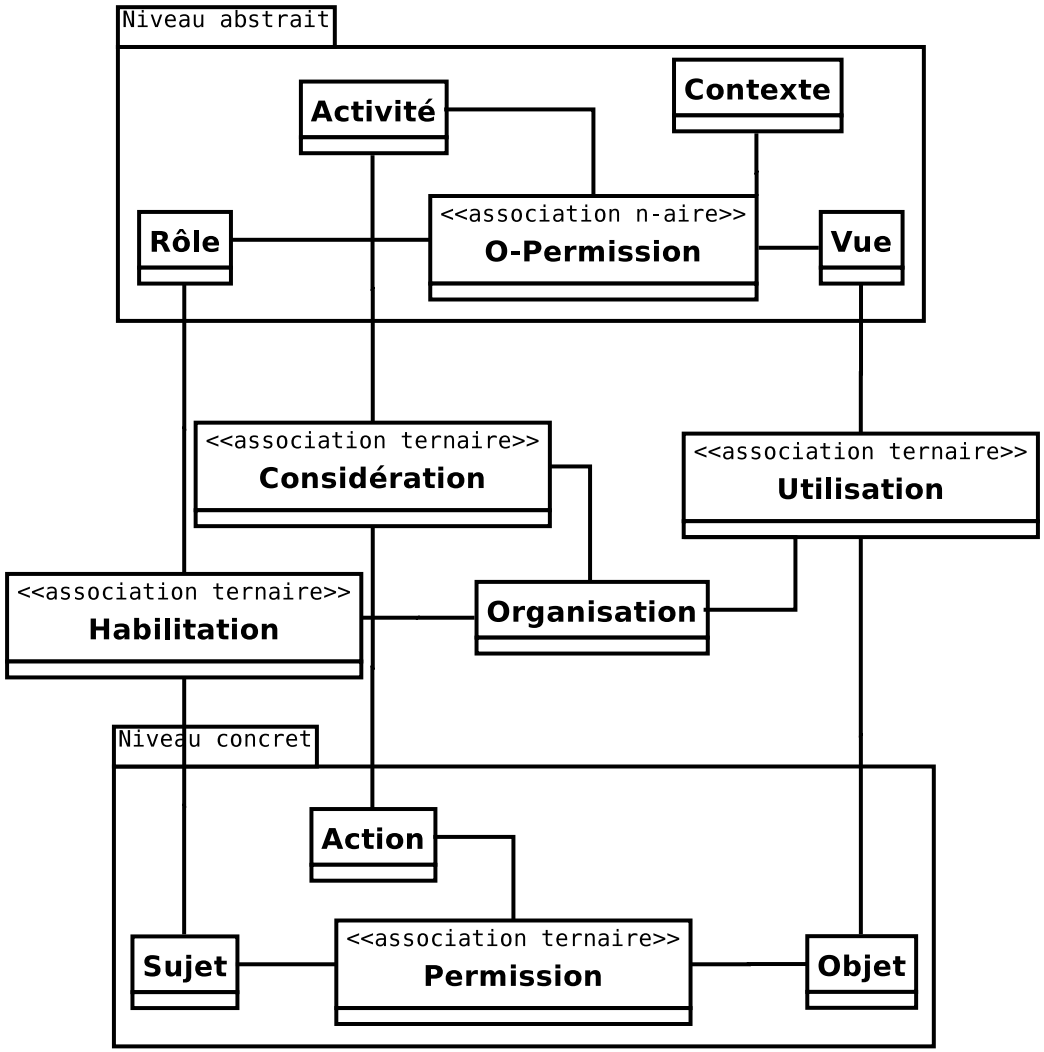
- poid du *legacy system*
- besoin de contrôle *fin*
- besoin de contrôle *souple*
- besoin de solutions *simples*

En pratique, des solutions *ad hoc*

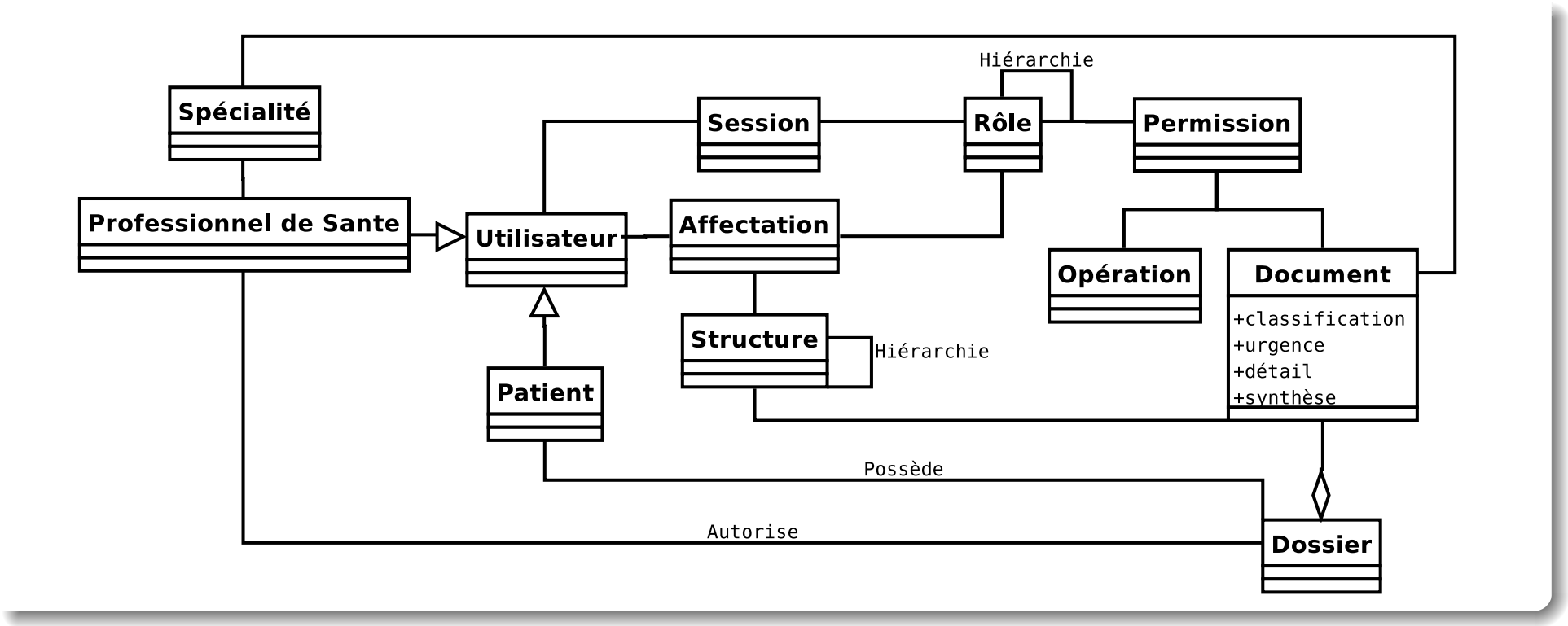
Modèle W-RBAC



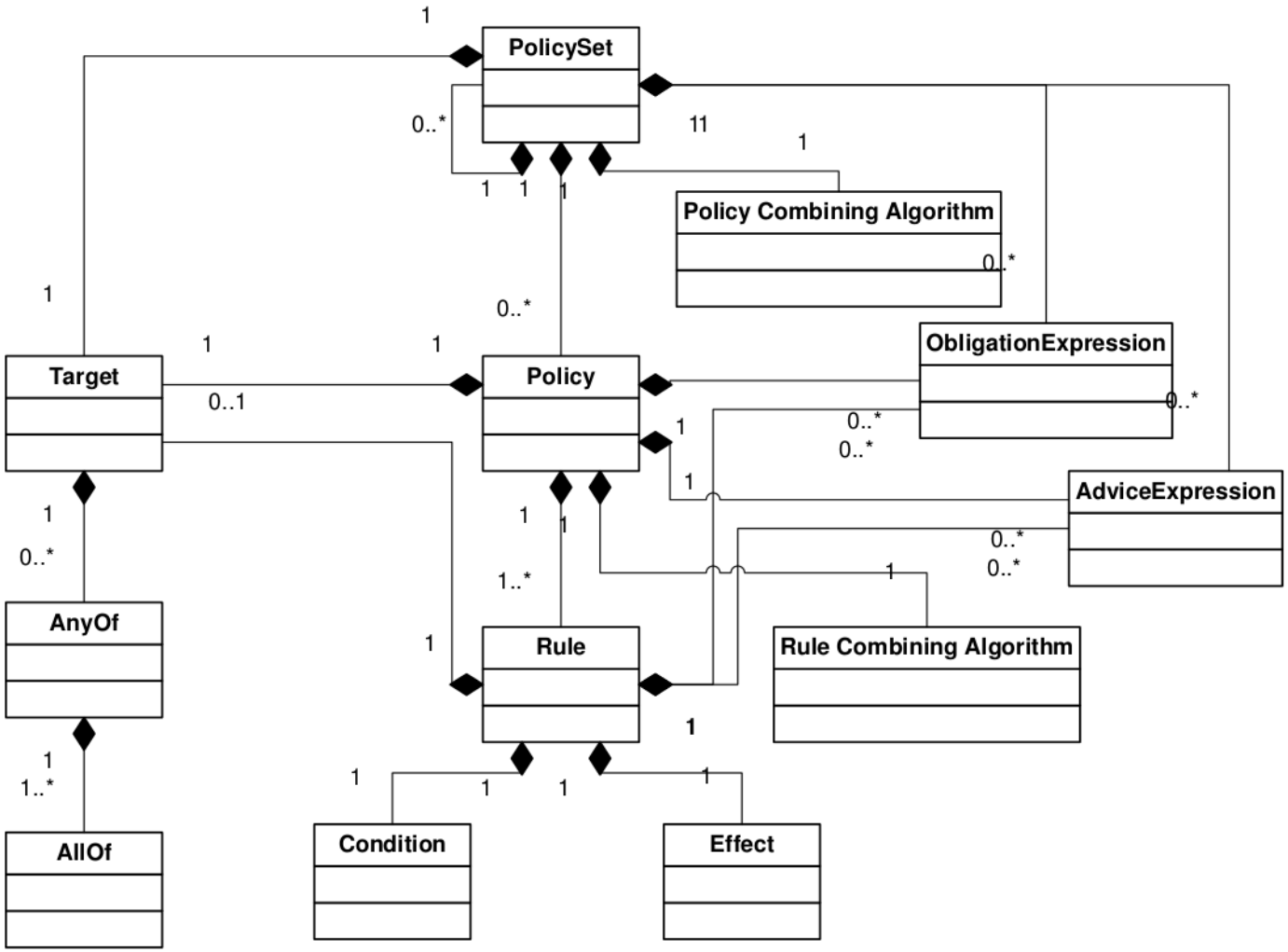
Modèle OrBac



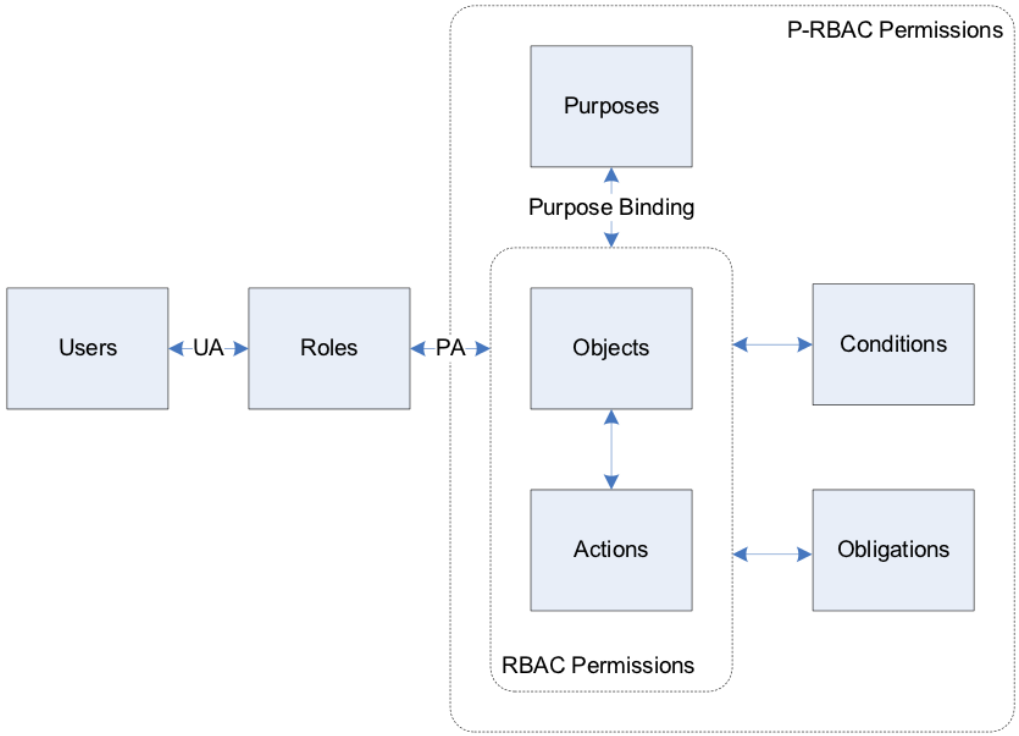
Modèle du GMSIH



Modèle XACML



Modèle P-RBAC



Ingénierie des rôles

Dans RBAC, le rôle est central

- rôle : collection d'utilisateurs et de permissions
- rôles : reflètent les fonctions et reponsabilités dans l'organisation
- hiérarchie : reflète la structure de l'organisation
- toutes les opérations administratives impliquent un ou plusieurs rôles

Bonne politique $\cong$ bons rôles

Ingénierie des rôles

Problème : difficile d'avoir de "bons" rôles :
l'ingénierie des rôles

- compatibles avec l'existant
- fidèles à l'organisation
- corrects et complets
- peu nombreux (efficacité)

Un processus

- coûteux (un rôle $\cong$ 5.000 $)
- long
- difficile (expertise, conception)
- . . . mais incontournable

Ingénierie des rôles

The concept of role engineering (RE) is an approach to defining roles and assigning permissions to the roles. RE must capture the organization's business rules, as these relate to access control. . .

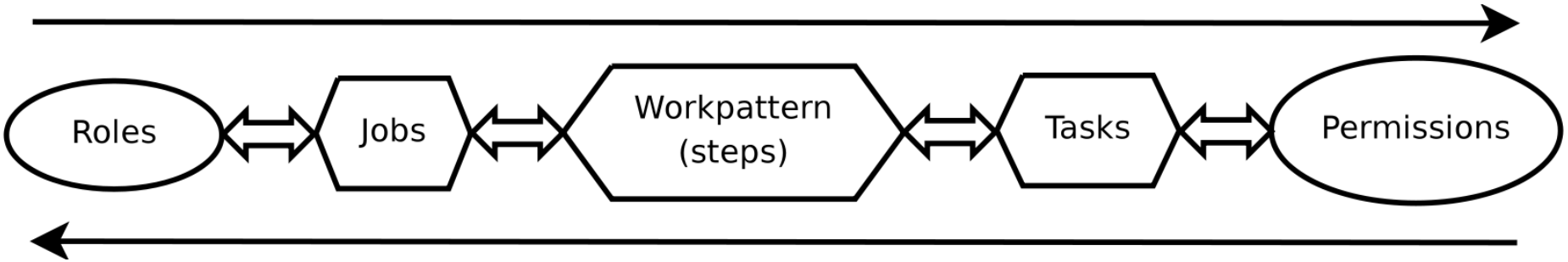


FIGURE: Démarche d'ingénierie des rôles

Top-down

L'approche *top-down*

- décomposer l'organisation : rafinements successifs
- identification des processus métiers et fonctions
- ne considère pas ou peu les permissions existantes

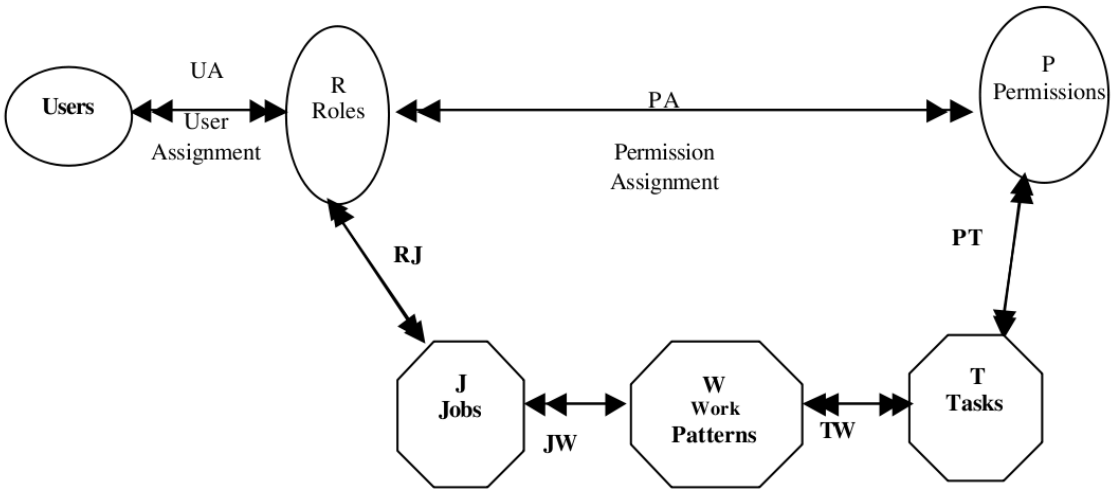


FIGURE: Modèle RBAC étendu

Bottom-up

Bottom-up

- palier au problème du *top-down* : très difficile à automatiser
- s'appuyer sur les permissions existantes

Techniques existantes

- techniques de fouilles de données
- analyse de relations binaires (contextes formels)
- critères de correction et d'optimalité (e.g., redondance, couverture)

- $\forall X \in 2^U, \alpha(X) = \{y \in P \mid \forall\, x \in X, (x, y) \in A\}$
- $\forall Y \in 2^P, \omega(Y) = \{x \in U \mid \forall\, y \in Y, (x, y) \in A\}$

Example

- $\alpha(23) = abgh$
- $\alpha(236) = ah$
- Utilisateurs :
- Permissions :

	a	**b**	c	d	e	f	**g**	**h**
1		×	×	×	×			
2	×	×	×				×	×
3	×	×				×	×	×
4				×	×			
5			×	×				
6	×							×

- $\forall X \in 2^U, \alpha(X) = \{y \in P \mid \forall x \in X, (x, y) \in A\}$
- $\forall Y \in 2^P, \omega(Y) = \{x \in U \mid \forall y \in Y, (x, y) \in A\}$

Example

- $\alpha(23) = abgh$
- $\alpha(236) = ah$
- Utilisateurs :
  bob
  charles
- Permissions :
  rRes1,wRes1
  rRes3,wRes3

	a	**b**	c	d	e	f	**g**	**h**
1		×	×	×	×			
2	×	×	×				×	×
3	×	×				×	×	×
4				×	×			
5			×	×				
6	×							×

- $\forall X \in 2^U, \alpha(X) = \{y \in P \mid \forall x \in X, (x, y) \in A\}$
- $\forall Y \in 2^P, \omega(Y) = \{x \in U \mid \forall y \in Y, (x, y) \in A\}$

Example

- $\alpha(23) = abgh$
- $\alpha(236) = ah$
- Utilisateurs :
  bob,charles
  fred
- Permissions :
  rRes1
  wRes3

	a	b	c	d	e	f	g	**h**
1		×	×	×	×			
2	×	×	×				×	×
3	×	×				×	×	×
4				×	×			
5			×	×				
6	×							×

Bottom-up

Concept formel

$c = (X, Y)$ tels que $Y = \alpha(X)$ (ou $X = \omega(Y)$)

- X : extension : les utilisateurs affectés (URA)
- Y : intention : les permissions affectées (PRA)

Ordre partiel :
$\leq_C$: $(X_1, Y_1) \leq_C (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2 \ (ou\ Y_2 \subseteq Y_1)$

Example

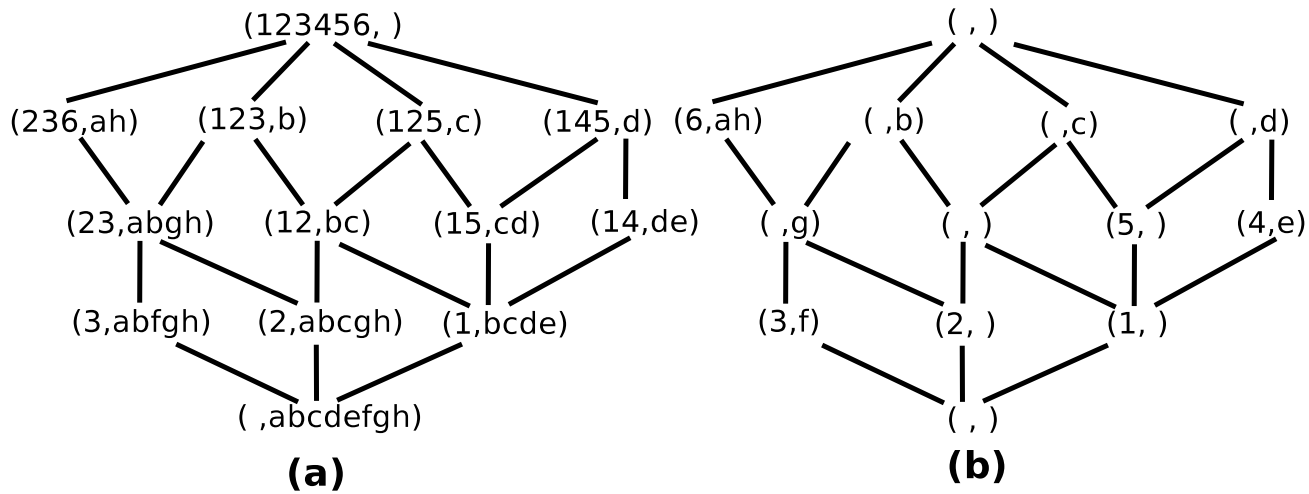
- $(12, bc)$ est un concept formel
- $(2, bc)$ n'est pas un concept car $2 \neq \omega(bc)$

Bottom-up

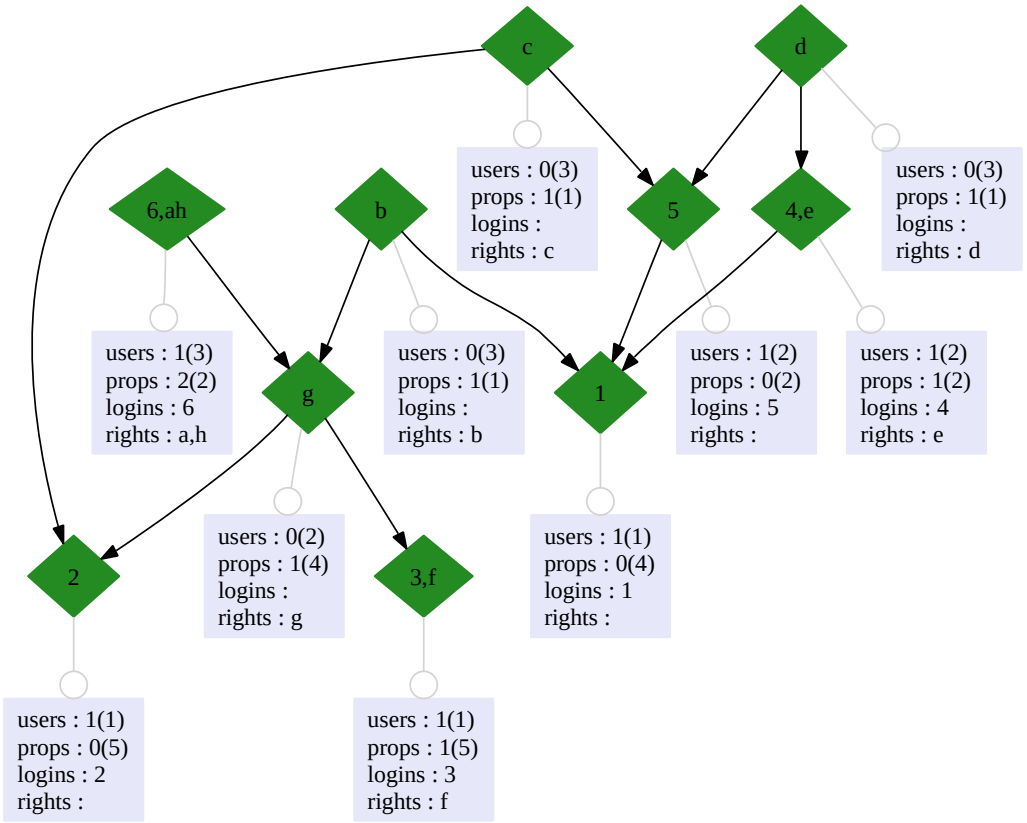
Théorème fondamental de l'analyse de concepts formels

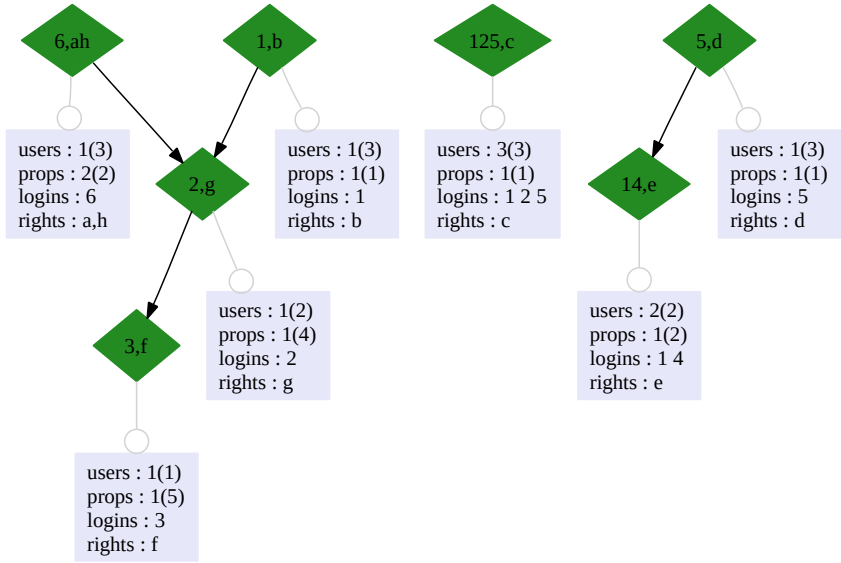
L'ensemble des concepts formels forme un treillis complet

Treillis des concepts



Simplification du treillis





Bottom-up

Problème de décomposition de matrices booléennes

- voir *Optimal Boolean Matrix Decomposition : Application to Role Engineering* ;
- problèmes de pavages ;
- problèmes de découverte de bases.

Variations sur le thème

Connaissant X, trouver C et R tels que $X = C \otimes R$:

- exactement (pas d'erreur) ;
- autorisant au plus δ erreurs ;
- avec nombre d'erreurs minimales pour largeur de C fixée à k ;
- avec un nombre de 1 minimal.

Problématique

Jusqu'à présent :

- confiance implicite dans le système
- système homogène
- vision globale
- contrôle absolu & centralisé

La réalité : systèmes distribués

- très grande taille
- hétérogènes
- autonomes
- défaillants

Problématique

Nécessite un effort substantiel

- paradigmes expressifs (comparé à matric binaire)
- langage de modélisation
- notion de politique de plus haut niveau

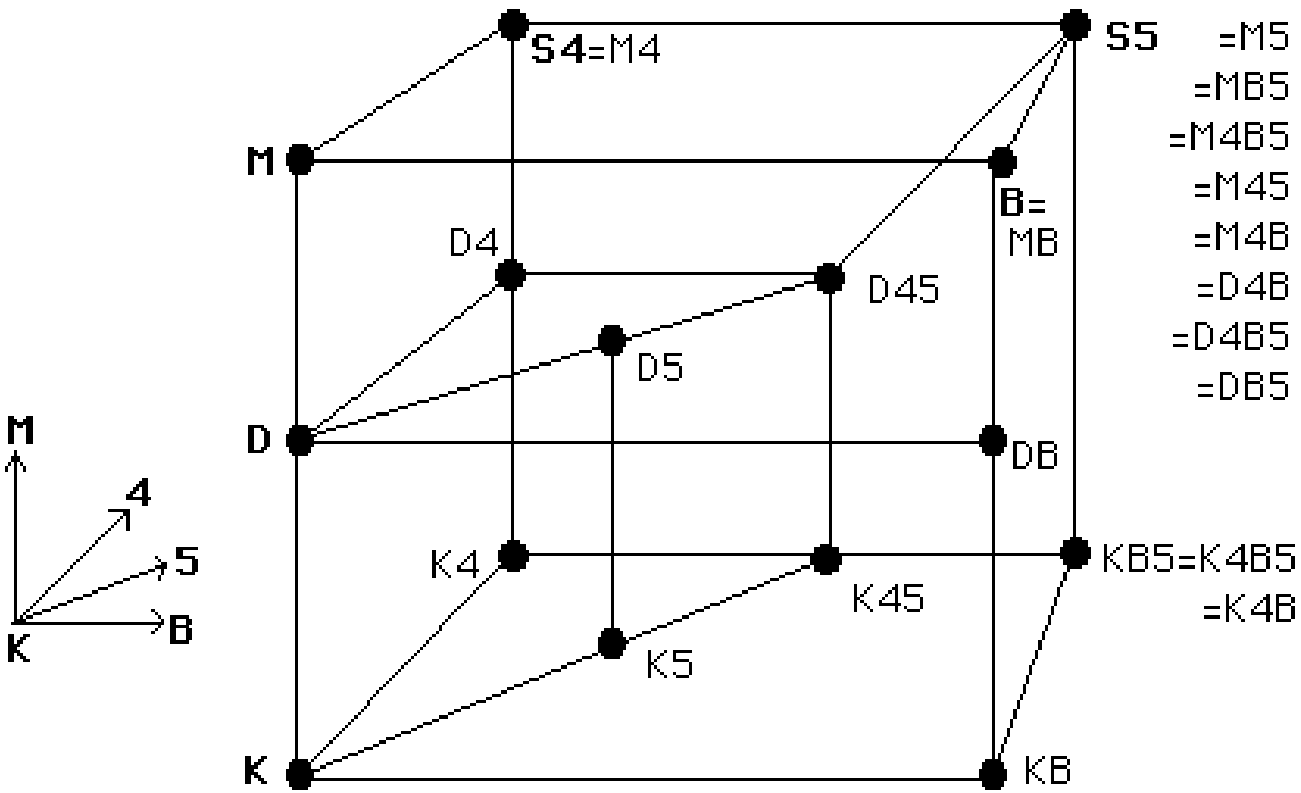
Propositions pour le contrôle d'accès distribué

- PolicyMaker & KeyNote
- SDSI
- SPKI
- D1LP & RT
- SD3
- Binder

Logiques modales

Syntaxe

$$\mathcal{L} ::= p \in \mathcal{P} \mid \bot \mid \top \mid \phi \vee \psi \mid \phi \wedge \psi \mid \phi \rightarrow \psi \mid \Diamond \phi \mid \Box \phi$$



Logiques modales

Généralités

- Modalités : quantificateurs contrôllés
- "interpretation philosophique" des modalités
- Structure $< W, R \subseteq W \times W, l : W \times P \rightarrow \{\mathbf{tt}, \mathbf{ff}\} >$

Example

- *Alethic* : nécessité ($\Box$) et possibilité ($\Diamond$) ;
- *Epistemic* : connaissance (K_A) ;
- *Doxastic* : croyance (B_A) ;
- *Deontic* : obligation ($\Box$) et permissions ($\Diamond$) ;
- *Temporal* : toujours ($\Box$) et *eventually* ($\Diamond$) ;

Logiques modales

Généralités

- Modalités : quantificateurs contrôllés
- "interpretation philosophique" des modalités
- Structure $< W, R \subseteq W \times W, l : W \times P \rightarrow \{\mathbf{tt}, \mathbf{ff}\} >$

Example

- *Alethic* : nécessité ($\Box$) et possibilité ($\Diamond$) ;
- *Epistemic* : connaissance (K_A) ;
- *Doxastic* : croyance (B_A) ;
- *Deontic* : obligation ($\Box$) et permissions ($\Diamond$) ;
- *Temporal* : toujours ($\Box$) et *eventually* ($\Diamond$) ;

Logiques pour le contrôle distribué

Syntaxe

$$\mathcal{L} ::= \top \mid \phi \vee \psi \mid \phi \wedge \psi \mid \phi \rightarrow \psi \mid A \texttt{ says } \phi \mid X \mid \forall X.\phi$$

- X variable
- A dans $\mathcal{P}$ (acteurs)
- ϕ et ψ dans $\mathcal{L}$

Sucre syntaxique

- A *parle pour* B : $\forall X.(A \texttt{ says } X \rightarrow B \texttt{ says } X)$
  permet d'exprimer la délégation de l'autorité de B *à* A
- A *contrôle* s : $A \texttt{ says } s \rightarrow s$
  A *a une totale autorité sur* s

Logiques pour le contrôle distribué

Abstraction sur *comment* A `says` ϕ

- dans un canal de confiance local d'un OS
  e.g., communication inter-processus, boucle locale
- sur un canal sûr entre deux machines
  e.g., liaison spécialisé, accès physique
- dans un canal sécurisé à clef partagée
  e.g., une session ssh, un échange PGP
- dans un certificat avec signature numérique à clef publique
  e.g., architecture de gestion des clefs

Système déductif

Décider des accès

- syntaxe : formules autorisées
- sémantique : assertions sur les accès
- preuve : *algorithme pour la décision d'accès*

Axiomes pour `says`

- tautologies de la logique des propositions + *Modus Ponens*
- $((A \texttt{ says } (\phi \rightarrow \psi)) \rightarrow (A \texttt{ says } \phi) \rightarrow (A \texttt{ says } \psi))$
- $\phi \rightarrow (A \texttt{ says } \phi)$
- $A \texttt{ says } (A \texttt{ says } \phi) \rightarrow (A \texttt{ says } \phi)$

Système déductif

Politique en langue naturelle

1. If *admin* says that *file* should be deleted, then it's the case
2. *admin* trusts *Bob* to decide whether *file* should be deleted
3. *Bob* wants to delete *file*

Traduction formelle

1. $(admin\ \texttt{says}\ delfile) \rightarrow delfile$
2. $admin\ \texttt{says}\ ((Bob\ \texttt{says}\ delfile) \rightarrow delfile)$
3. $Bob\ \texttt{says}\ delfile$

Prouver formellement que $delfile$

Système déductif

`says` un opérateur versatile

- modélise les requêtes d'accès
- permet de diviser la politique selon l'autorité
- traduit une forme de délégation

Logique modale

- A `says` peut être vu comme une modalité (indexée par $\mathcal{P}$)
- choix des axiomes pour A `says` : à débattre
- théorie de la preuve satisfaisante, mais sémantique complexe

Évolution des problématiques

Contrôle d'usage

- *Enterprise*-Digital Right Management,
- obligations dans le futur,
- modélisation des organisations et de leurs relations.

Protection de la vie privée

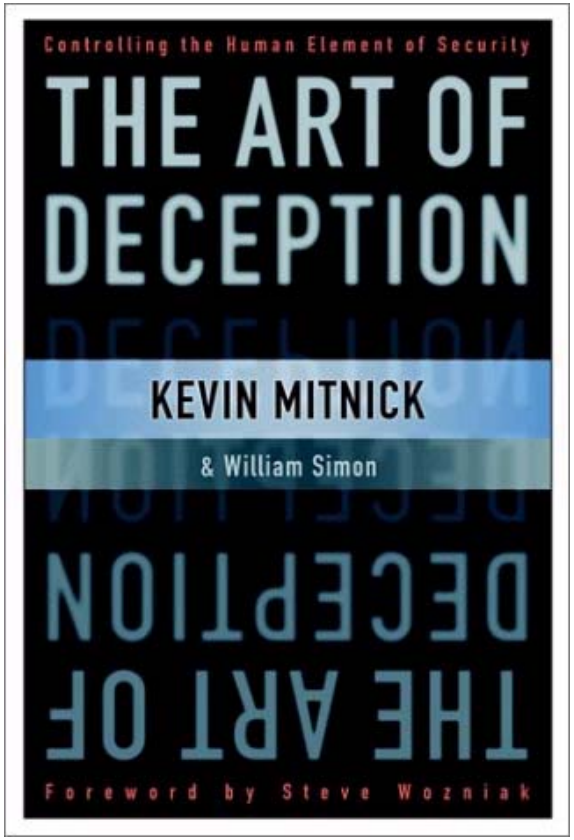
Contrôle d'accès *nécessaire* . . . mais pas *suffisant* !

- accès *réciproque* de la personne identifiée,
- retention limitée des données,
- proportionnalité et finalité des données,
- contrôle du transfert.

Le mot de la fin

"Don't rely on network safeguards and firewalls to protect your information. Look to your most vulnerable spot. You'll usually find that vulnerability lies in your people."

Kevin D. Mitnick – "The Art of Deception", p.79



Merci de votre attention

Protocole Kerberos

- Protocole d'authentification réseau client/serveur
- Basé sur les tickets
- Authentification mutuelle (client et serveur)
- Version 5 utilisée, après la 4 qui était faible
- Basé sur Needham-Shroeder symétrique
- Extension asymétrique possible (rfc4556)

Rôles dans le protocole

S	Serveur
C	Client
AS	Authentication Server
TGT	Ticket-Granting Server
KDC	Key Distrution Center = AS + TGT

Kerberos

1. $C \rightarrow AS$: un ticket à presenter à TGS ?
   demande de passe 3j de ski sur 4 domaines
2. $AS \rightarrow C$: voici un ticket pour C à présenter à TGS.
   voici le passe (TGT) à présenter à l'arrivée dans un domaine
3. $C \rightarrow TGS$: voici mon ticket de AS, un ticket pour S ?
   voici mon passe (TGT), je souhaite skier dans votre domaine
4. $TGS \rightarrow C$: voici un ticket pour C à présenter à C
   voici votre passe (ST) pour notre domaine
5. $C \rightarrow S$: voici mon ticket de TGS.
   voici mon passe (ST) pour votre domaine, puis-je utiliser cette remontée ?

Kerberos

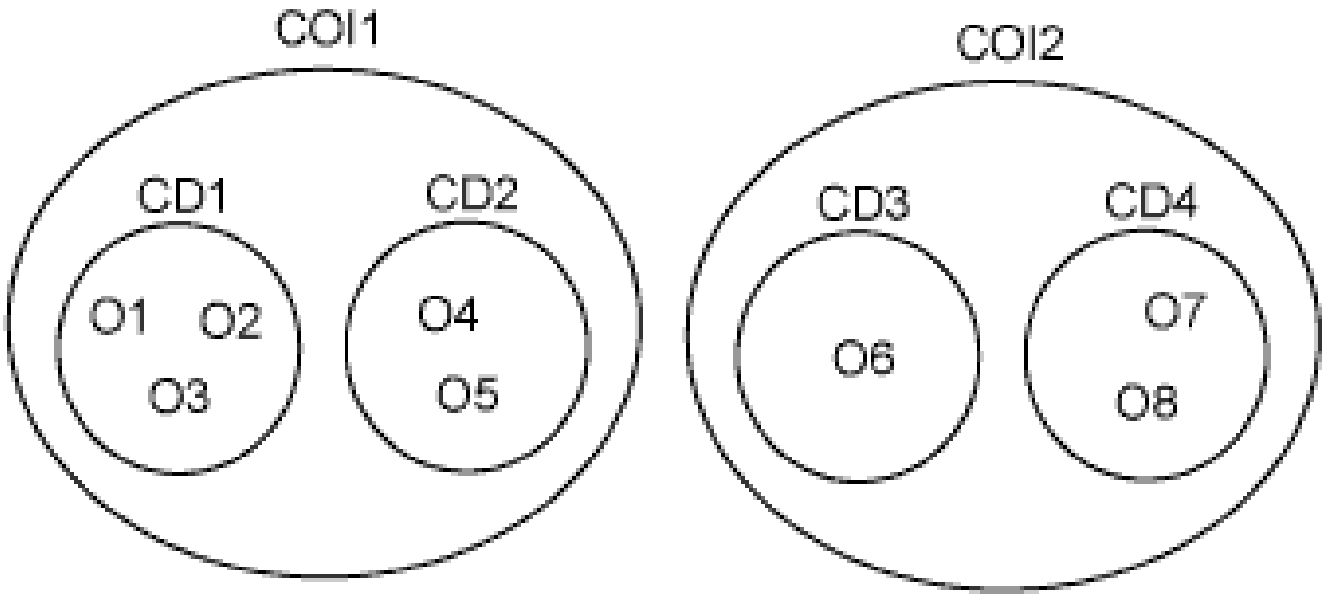
$$
\begin{aligned}
C \to AS &: [C, TGS] \\
AS \to C &: [\{K_{C,TGS}\}_{K_C}, \{T_{C,TGS}\}_{K_{TGS}}] \\
&\quad \text{où } T_{C,TGS} = [C, validity, K_{C,TGS}] \\
C \to TGS &: [S, \{T_{C,TGS}\}_{K_{TGS}}, \{A_C\}_{K_{C,TGS}}] \\
&\quad \text{où } A_C = [C, timestamp] \\
TGS \to C &: [\{K_{C,S}\}_{K_{C,TGS}}, \{T_{C,S}\}_{K_S}] \\
&\quad \text{où } T_{C,S} = [C, validity'] \\
C \to S &: [\{A_C\}_{K_{C,S}}, \{T_{C,S}\}_{K_S}]
\end{aligned}
$$

- $K_{C,TGS}$ est la clef symétrique de session ;
- K_C est la clef (symétrique) de C ;
- K_{TGS} est la clef (symétrique) du TGS ;
- K_S est la clef (symétrique) de S ;
- $T_{C,TGS}$ est le *TGT* fourni par l'AS à présenter au TGS ;
- $T_{C,S}$ est le *ST* fourni par le TGS à présenter à S ;

Muraille de chine

Modèle de Brewer & Nash

- basé sur l'historique des accès : dynamique
- combinaison de mandataire et de libre arbitre
- basé sur la notion de conflits d'intérêts



Autorisations

- S peut lire O
  - si O est dans le même dataset qu'un objet auquel S a déjà accédé
  - *ou*, si O appartient à une classe de conflit à laquelle S n'a jamais accédé
- S peut écrire O
  - si S peut lire O
  - *et*, si S n'a jamais lu dans un autre dataset que celui où il essaie d'écrire

Limitations

- gestion de l'historique,
- trop restrictif (sur l'écriture).