TIW4 : SÉCURITÉ DES SYSTÈMES D'INFORMATION

GESTION DES AUTORISATIONS

romuald.thion@univ-lyon1.fr

http://liris.cnrs.fr/~rthion/dokuwiki/enseignement:tiw4

Master « Technologies de l'Information et Web »

Objectifs

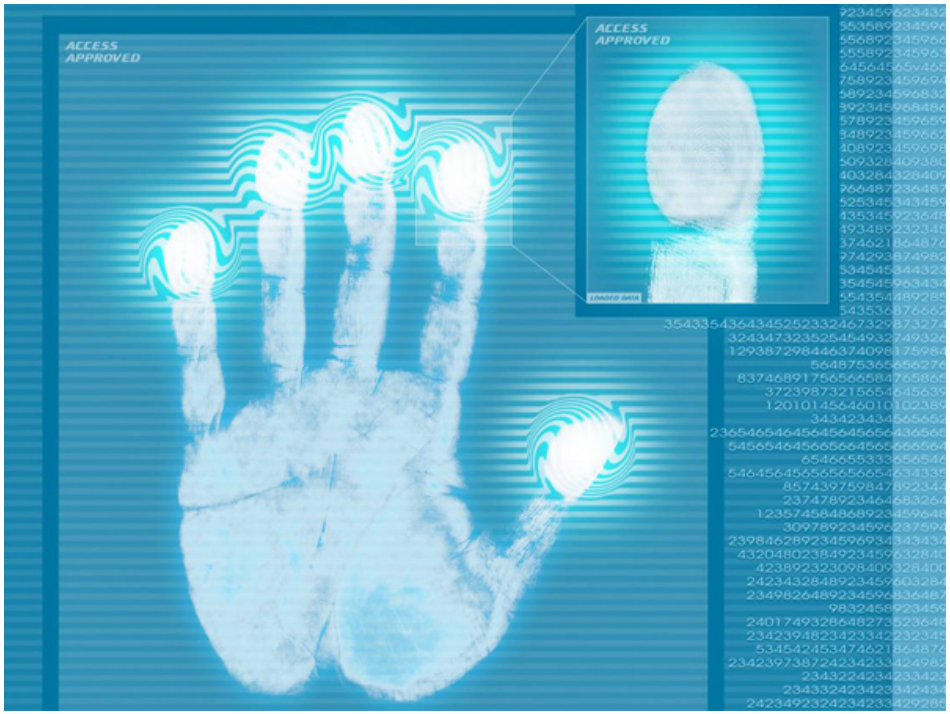
Objectifs

- Connaître les principes de l'organisation des droits
- Déterminer les droits dans un modèle donné
- Organiser les droits avec les rôles
- Choisir et concevoir un modèle de contrôle d'accès

Autorisation (ou contrôle d'accès)

- préalable : authentification (en : *authentication*)
- ensuite : autorisation (en : *authorization*)

Authentification et autorisation



Authentification

Autorisation

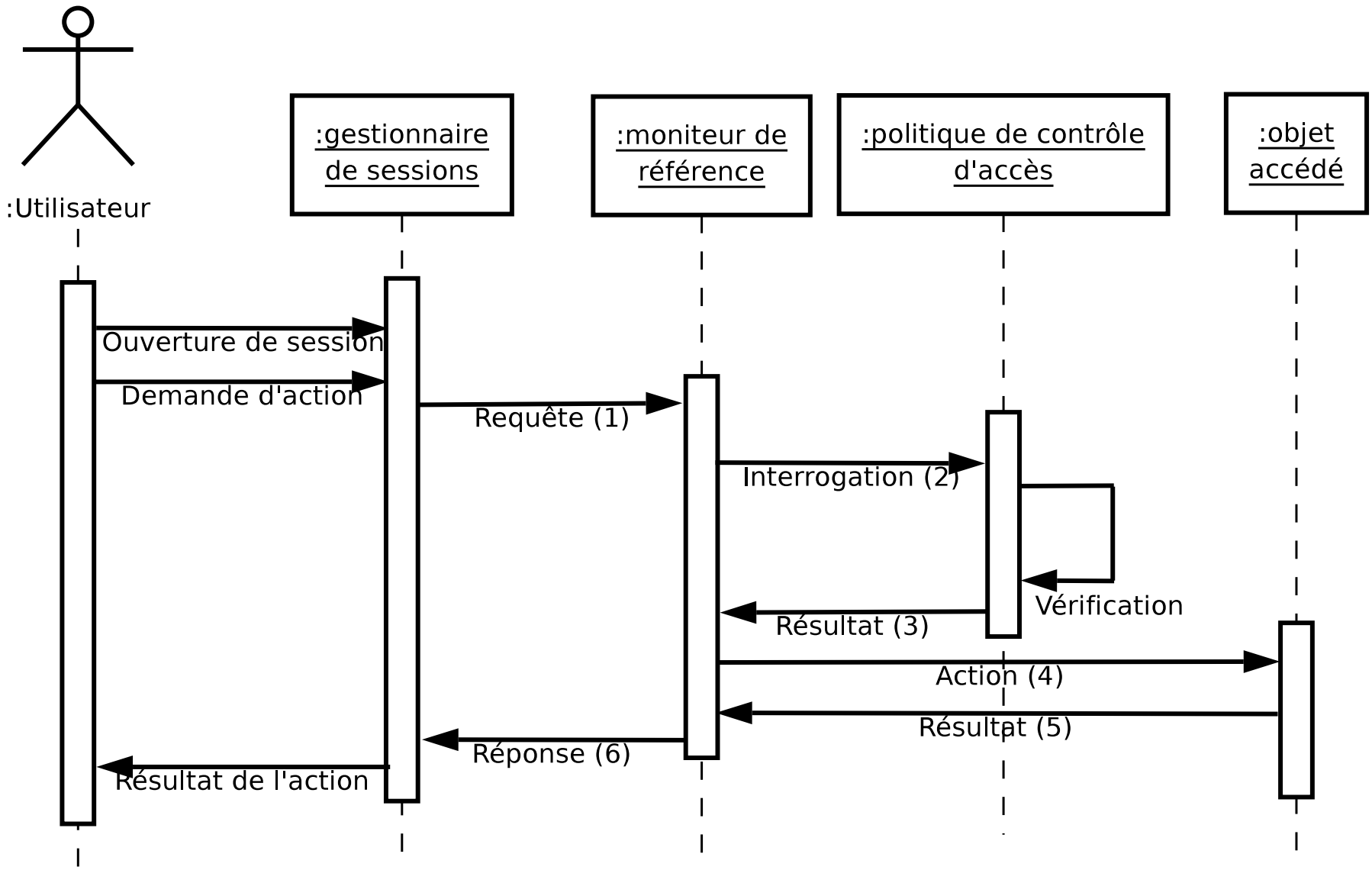
Contrôle d'accès : définition

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, quand il est « séparé »



Caractéristiques NEAT du moniteur

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

Toute la sécurité réside dans la confiance accordée au moniteur

Poliitique 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)

Poliitique 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)

Le moniteur, formellement (1/2)

Le moniteur réalise une fonction $\delta : Q \times \Sigma \to \{0, 1\}$

- $q \in Q$ est la demande d'accès, typiquement un triplet $\langle s, a, o \rangle \in (S \times A \times O)$;
- Σ la politique (l'état du système) sur lequel prendre la décision ;
- $D = \{0, 1\}$ l'espace binaire des décisions possibles.

Variations sur le thème

- *Systèmes dynamiques* $\delta : (S \times A \times O) \times \Sigma \to \{0, 1\} \times \Sigma$
- *Triplet en intention* $\delta : (2^S \times 2^A \times 2^O) \times \Sigma \to \{0, 1\}$
- *Décisions riches* $\delta : (S \times A \times O) \times \Sigma \to D$
  - $D = \{0, \bot, 1\}$ logique multivaluée avec état « indéfini » ;
  - $D = \{0, 1, \bot, \top\}$ logique de Belnap ;
  - 6 valeurs pour XACML !

Le moniteur, formellement (2/2)

Instances du cadre formel

Les modèles que nous verrons sont (presque) tous des instances de $\delta : (S \times A \times O) \times \Sigma \to \{0, 1\}$:

- c'est le cadre *classique* ;
- définir $S \times A \times O$, c'est choisir un domaine d'application ;
- définir la structure de Σ, c'est donner un langage de politiques ;
- définir δ à proprement parler, c'est donner la fonction de décision ;

Administrer une politique de contrôle d'accès, c'est modifier l'état courant $\sigma \in \Sigma$ utilisé par le moniteur

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

- décentralisé,
- basé sur l'identité de l'accédant,
- impossible [a] à contrôler.

a. Le problème de savoir si d'un état *sûr* arbitraire on ne peut atteindre que des états *sûrs* est indécidable dès que le modèle est trop riche https://dl.acm.org/doi/10.1145/360303.360333.

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

- décentralisé,
- basé sur l'identité de l'accédant,
- impossible [a] à contrôler.

---

a. Le problème de savoir si d'un état *sûr* arbitraire on ne peut atteindre que des états *sûrs* est indécidable dès que le modèle est trop riche https://dl.acm.org/doi/10.1145/360303.360333.

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

Modélisation

Politique et décision

$$\Sigma = \mathcal{P}(S \times A \times O)$$

$$\delta(\sigma)(< s, a, o >) = 1 \equiv < s, a, o > \in \sigma$$

Opérations administratives

- `enter` a `into` $M(s, o)$, e.g., `chown` et `chmod`
- `delete` a `into` $M(s, o)$, e.g., `chown` et `chmod`
- `create subject` s, e.g., `useradd`
- `delete subject` s, e.g., `userdel`
- `create object` o, e.g., `touch` et `tee`
- `delete object` o, e.g., `rm`

Représentations 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.
```

Représentations 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.
```

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 de bonnes propriétés formelles,
- Mise en œuvre dans les langages (e.g., typage),
- Mécanisme de contrôle de flux

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 de bonnes propriétés formelles,
- Mise en œuvre dans les langages (e.g., typage),
- Mécanisme de contrôle de flux

Dérivations des autorisations

Classes de sécurité (simplifié)

- Organisation de niveaux en treillis (ordre partiel avec glb et lub) :
  - 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

Modélisation

Politique et décision

$$\Sigma = L^O \times L^S \text{ avec } \langle L, \preceq_L \rangle \text{ treillis}$$

$$\delta(\langle \alpha, \beta \rangle)(< s, r, o >) = 1 \equiv \ \alpha(o) \preceq_L \beta(s)$$
$$\delta(\langle \alpha, \beta \rangle)(< s, w, o >) = 1 \equiv \ \beta(s) \preceq_L \alpha(o)$$

Opérations administratives

- `classification` l `to` o
- `credential` l `to` s
- `create subject` s
- `delete subject` s
- `create object` o
- `delete object` o

Modèles mandataires

Donner un exemple de niveaux utilisés en France ?

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*.

Contrôle de flux

Intérêt du modèle

- simple et vérifiable en un sens
- il garantit que *le niveau d'un objet ne décroit jamais*
- il n'y a pas de *fuite d'information* dans un *système d'échanges*

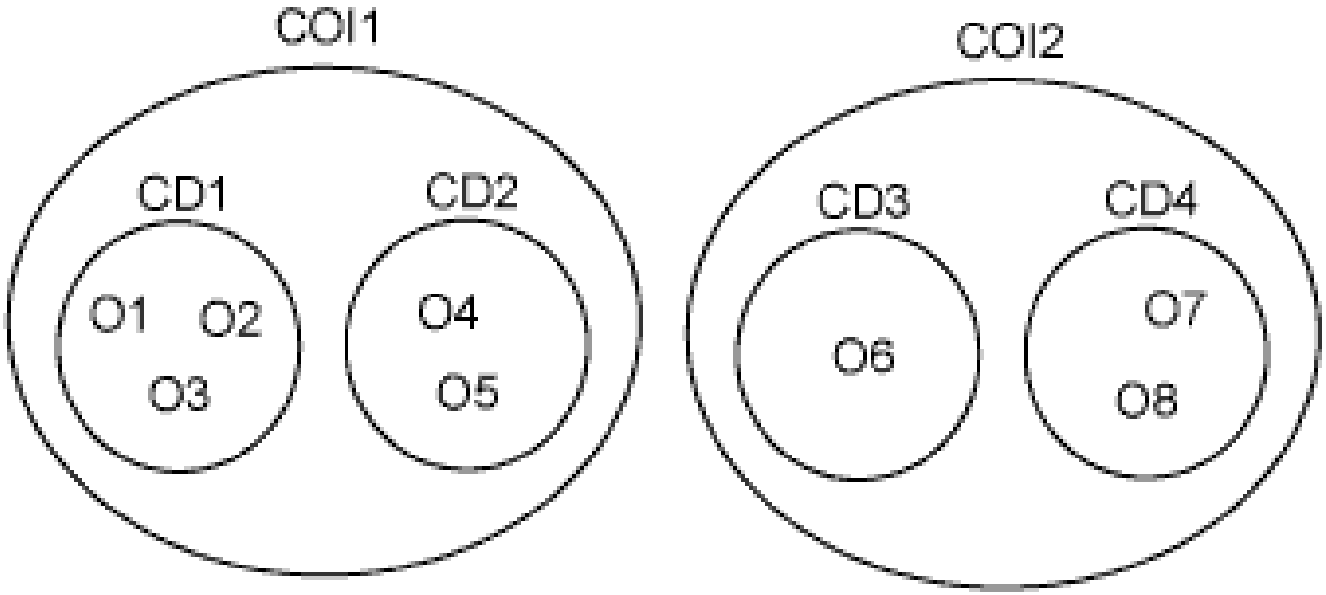
Problème : on peut vouloir *fuiter* volontairement

- C'est la déclassification
  e.g., les archives militaires libérées 20 ans plus tard
- Un système avec déclassification contrôlée doit avoir une meilleure sécurité que sans. . .

Muraille de chine

Modèle de Brewer & Nash

- basé sur l'historique des accès : dynamique
- basé sur la notion de conflits d'intérêts :
  - Les objets sont regroupés en *datasets*
  - Les *datasets* sont regroupés en *classes de conflits d'intérêts*



Autorisations

- s peut lire o
  - si o est dans le même dataset qu'un objet auquel s a déjà accédé (« derrière le mur »)
  - ou, si o appartient à une autre classe de conflits d'intérêt
- s peut écrire o
  - si s peut lire o
  - et, si s ne peut lire aucun objet qui appartienne à un dataset de la même classe de conflits d'intérêt

Limitations

- complexité fonction de la taille de l'historique,
- difficile à mettre en œuvre.

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.

Ici, c'est une présentation formelle, théorique, des concepts. En pratique, ces concepts sont certes très utilisés, mais souvent d'une façon *ad hoc* avec des concepts supplémentaires.

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.

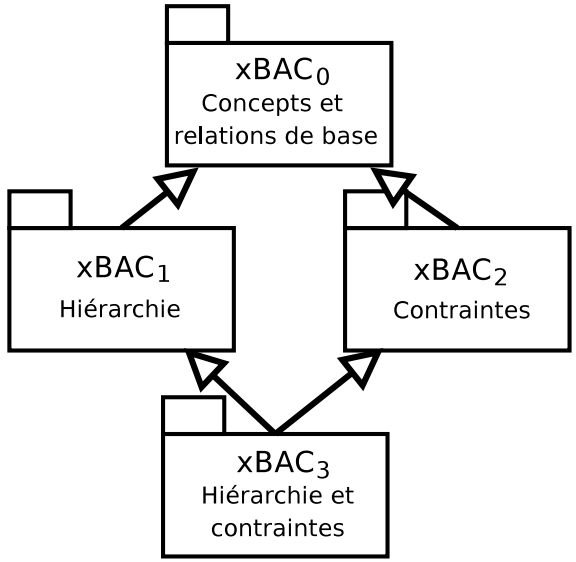
Ici, c'est une présentation formelle, théorique, des concepts. En pratique, ces concepts sont certes très utilisés, mais souvent d'une façon *ad hoc* avec des concepts supplémentaires.

Famille standard

Modèles standards

Plusieurs modèles RBAC définis dans le standard ANSI :

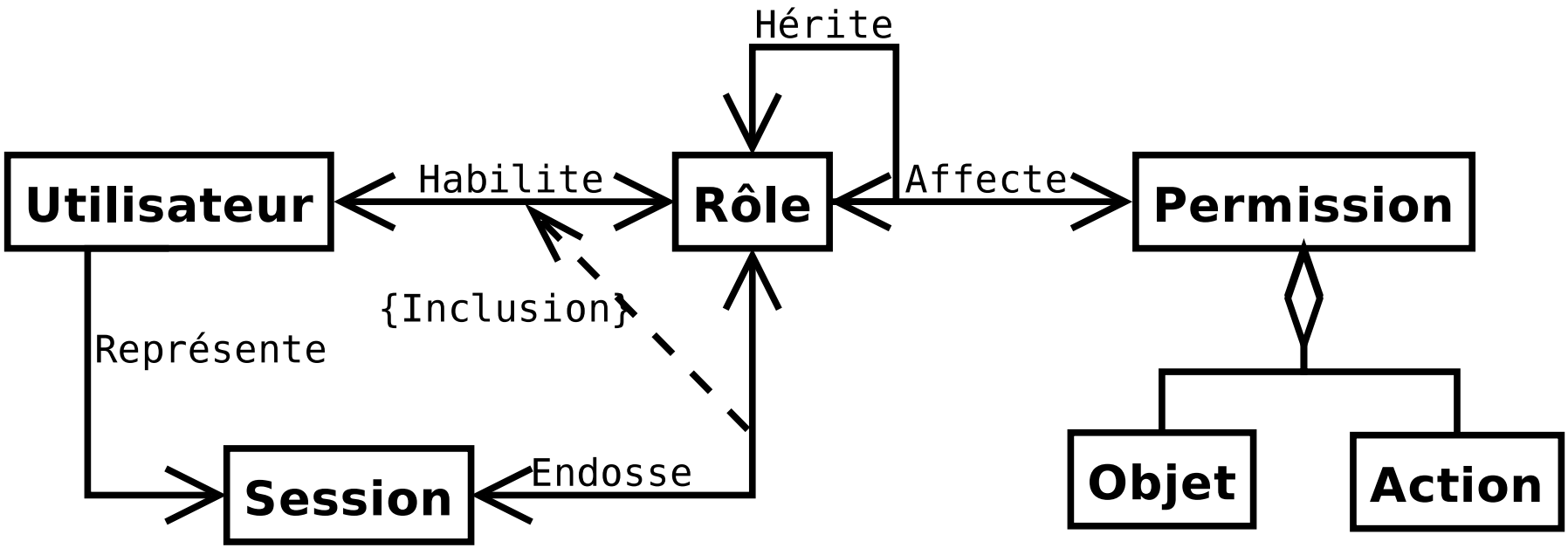
- RBAC_0 : le noyau, les concepts
- RBAC_1 : la hiérarchie de rôles,
- RBAC_2 : les contraintes,
- RBAC_3 : hiérarchies *et* contraintes.



RBAC_0 les concepts qui sont quasiment omniprésents, RBAC_1 très commun aussi, RBAC_2 et RBAC_3, très peu d'intérêt concrètement.

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),
dont une instance est une politique

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),
dont une instance est une politique

Exemple de politique

Politique RBAC de la matrice jouet

▸ voir la matrice

$\mathcal{URA}$

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

$\mathcal{PRA}$

	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

$\mathcal{URA}$

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

$\mathcal{PRA}$

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

Modélisation

Politique et décision

$$\Sigma = \mathcal{P}(S \times R) \times \mathcal{P}(R \times A \times O)$$

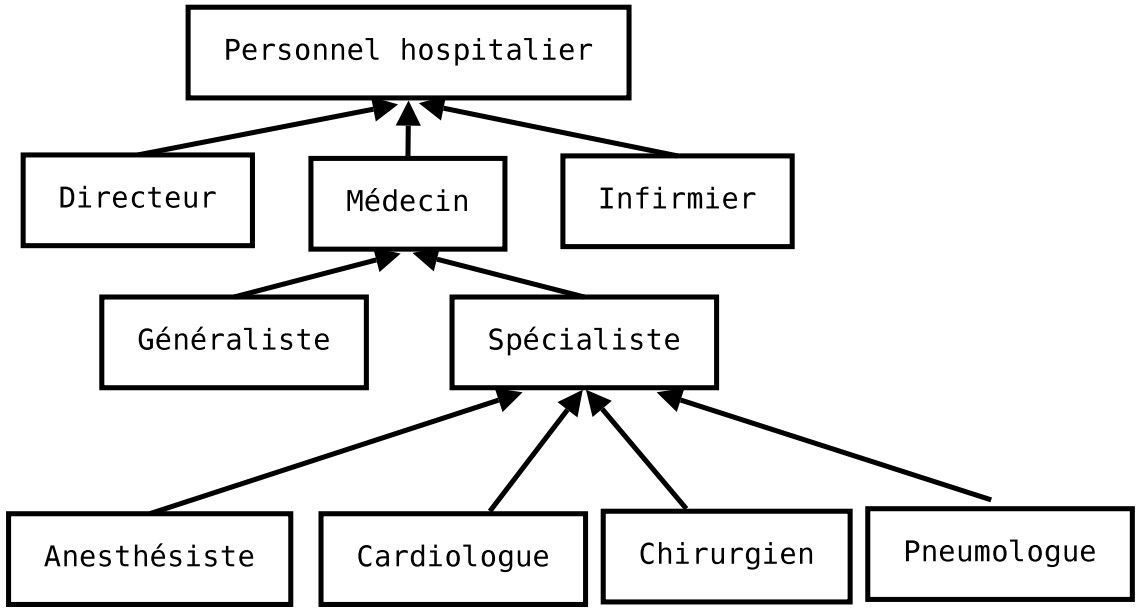
$$\delta(\langle ura, pra \rangle)(< s, a, o >) = 1 \equiv \exists r \in R. \langle s, r \rangle \in ura \wedge \langle r, a, o \rangle \in pra$$

Opérations administratives (ARBAC97)

- On organise les privilèges sur Σ avec un modèle à rôles (dit « administratifs »)
- URA97
  - `canAssign` $\subseteq AR \times C \times 2^R$
  - `canRevocke` $\subseteq AR \times 2^R$
- PRA97
  - `canAssignP` $\subseteq AR \times C \times 2^R$
  - `canRevockeP` $\subseteq AR \times 2^R$

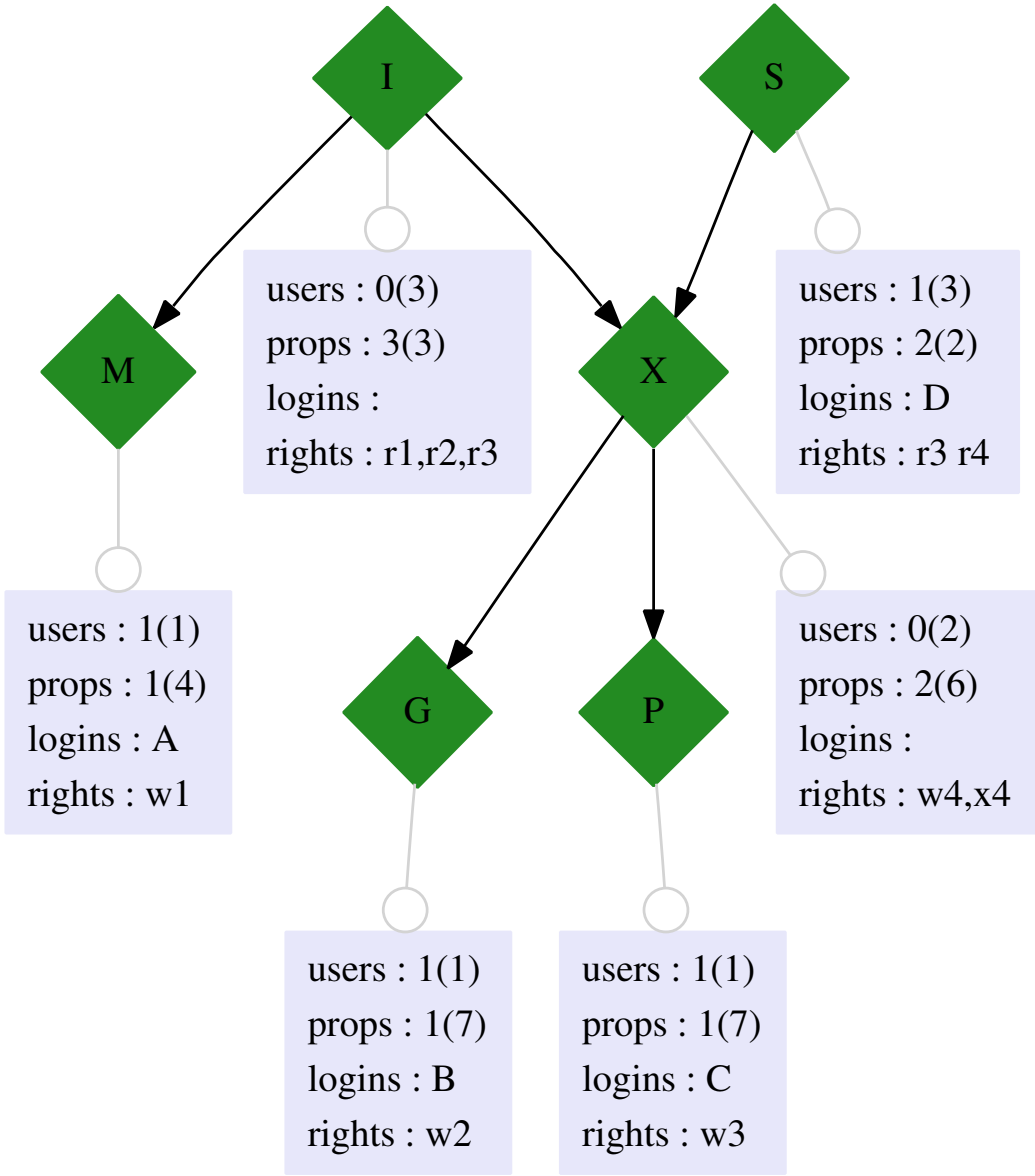
Hiérarchisation des rôles

Une hiérarchie en arbre



Plus expressif, plus complexe

- Nécessite de parcourir la hiérarchie,
- Problème avec SGBDR (lequel ?),
- Possible de restreindre l'expressivité



Une hiérarchisation de la politique jouet ▸ voir la politique

Modélisation

Hiérarchie des rôles

Soit $\preceq \subseteq R \times R$ une relation d'ordre partiel entre rôles :

- $r \preceq r'$ se lit r' *hérite de* r et dispose de tous ses privilèges
- réflexive : $\forall r.r \preceq r$
- transitive : $\forall r, r', r''.r \preceq r' \wedge r' \preceq r'' \Rightarrow r \preceq r''$
- antisymétrique : $\forall r, r'.r \preceq r' \wedge r' \preceq r \Rightarrow r = r'$

Politique et décision

$$\Sigma = \mathcal{P}(S \times R) \times \mathcal{P}(R \times R) \times \mathcal{P}(R \times A \times O)$$

$$\delta(\langle ura, \preceq, pra \rangle)(< s, a, o >) = 1 \equiv$$

$$\exists r, r' \in R.(r \preceq r') \wedge \langle s, r' \rangle \in ura \wedge \langle r, a, o \rangle \in pra$$

Affectés *versus* autorisés

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

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

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

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

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.

Le concept est très intéressant, en pratique c'est assez complexe et assez peu implanté.

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.

Le concept est très intéressant, en pratique c'est assez complexe et assez peu implanté.

Filtrage de pare-feux

$\delta : Q \times \Sigma \to \{0, 1\}$ pour les pare-feux

- sujet : l'IP source, le port source, le protocole ;
- objet : l'IP destination, le port destination ;
- action : autoriser, refuser, rejeter (effet de bord) ;
- Σ : liste de règles ;
- définition de δ est la stratégie de décision : généralement, lire dans l'ordre et appliquer la première règle applicable.

Exemple minimaliste

	IPsrc	*IPdst*	*Prot*	*Psrc*	*Pdst*	***Action***
1	192.168.10.20	194.154.192.3	tcp	*	25	Accept
2	*	194.168.10.3	tcp	*	80	Accept
3	192.168.10.0/24	*	tcp	*	80	Accept
4	*	*	*	*	*	Deny

PostgreSQL : `CREATE ROLE` et `GRANT role TO`

Gestion des rôles

- création de rôles, affectation/révocation $\mathcal{PRA}$, $\mathcal{URA}$ et $\mathcal{RH}$.
- Attention : en PostgreSQL, les utilisateurs sont des rôles comme les autres avec le droit se logguer.

https://www.postgresql.org/docs/current/user-manag.html
https://www.postgresql.org/docs/current/sql-createrole.html

```
CREATE ROLE name [ [ WITH ] option [ ... ] ]
where option can be:
    | LOGIN | NOLOGIN
    | [ ENCRYPTED ] PASSWORD 'password' | PASSWORD NULL
    | IN ROLE role_name [, ...]
    | ROLE role_name [, ...]
    | ...

-- /!\ pour la hiérarchie de rôles /!\
GRANT role_name [, ...] TO role_specification [, ...]
    [ WITH ADMIN OPTION ]
```

PostgreSQL : GRANT et REVOKE

Syntaxe générale : on donne via GRANT

- soit un privilège sur un objet (système ou utilisateur)
- soit un rôle à un autre rôle (cf. slide précédent)

Attention : PostgreSQL a une notion de *schéma* intermédiaire entre base de données et tables !

https://www.postgresql.org/docs/current/sql-grant.html

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIG
    [, ...] | ALL [ PRIVILEGES ] }
    ON { [ TABLE ] table_name [, ...]
         | ALL TABLES IN SCHEMA schema_name [, ...] }
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | CONNECT | TEMPORARY | TEMP } [, ...] | ALL [ PRIVILEGES
    ON DATABASE database_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]
```

PostgreSQL : tables système ou *catalogue*

https://www.postgresql.org/docs/current/catalogs.html

Tables de la gestion des droits (hors *policies*)

- `pg_authid` : *The catalog `pg_authid` contains information about database authorization identifiers (roles).*
- `pg_roles` *This is simply a publicly readable view of `pg_authid` that blanks out the password field.*
- `pg_auth_members` *The catalog `pg_auth_members` shows the membership relations between roles. Any non-circular set of relationships is allowed.*
- `pg_user` et `pg_shadow` : les alternatives obsolètes à `pg_roles` et `pg_authid`. Idem pour `pg_group`

Attention `pg_authid`, `pg_roles` et `pg_auth_members` sont contrôlées niveau *cluster*, pas *database*.

PostgreSQL : exemple

Exemple : création et catalogue

```
CREATE ROLE test ROLE romulus IN ROLE pedago;

SELECT r1.rolname,
       r2.rolname,
       r3.rolname,
       admin_option
FROM pg_auth_members a
     JOIN pg_roles r1 ON a.roleid = r1.oid
     JOIN pg_roles r2 ON a.member = r2.oid
     JOIN pg_roles r3 ON a.grantor = r3.oid;
```

Modèle XACML : motivation

[Axiomatics] "Attribute Based Access Control (ABAC) is a "next generation" authorization model that provides dynamic, context-aware and risk-intelligent access control. It helps achieve efficient regulatory compliance, effective cloud services, reduced time-to-market for new applications, and a top-down approach to governance through transparency in policy enforcement."

[Wikipedia] "XACML stands for "eXtensible Access Control Markup Language". The standard defines a declarative access control policy language implemented in XML and a processing model describing how to evaluate access requests according to the rules defined in policies."

XACML est une DTD XML pour un modèle ABAC, dont la sémantique est spécifiée par un standard.

Modèle XACML : choix architecturaux

Policy-based access control

Spécification déclarative des droits d'accès, au plus proche de la Politique de Sécurité du Système d'Information (PSSI).

Attribute-based access control

Sujet, action et objets sont spécifiés en intention, par des propriétés qui leurs sont associées (clef-valeurs). Le cadre classique $\langle s, a, o \rangle \in (S \times A \times O)$ est le *cas particulier* où l'unique clef est l'identifiant.

Tree-structured policies

Les politiques sont organisées hiérarchiquement. Chaque feuille est une règle de contrôle d'accès avec une décision `Permit` ou `Deny`. Chaque noeud de l'arbre spécifie comment combiner les décisions prises par ses sous-arbres.

Policy-based access control : exemple

Example 1 (Patient Policy). The general policy in the hospital in particular:

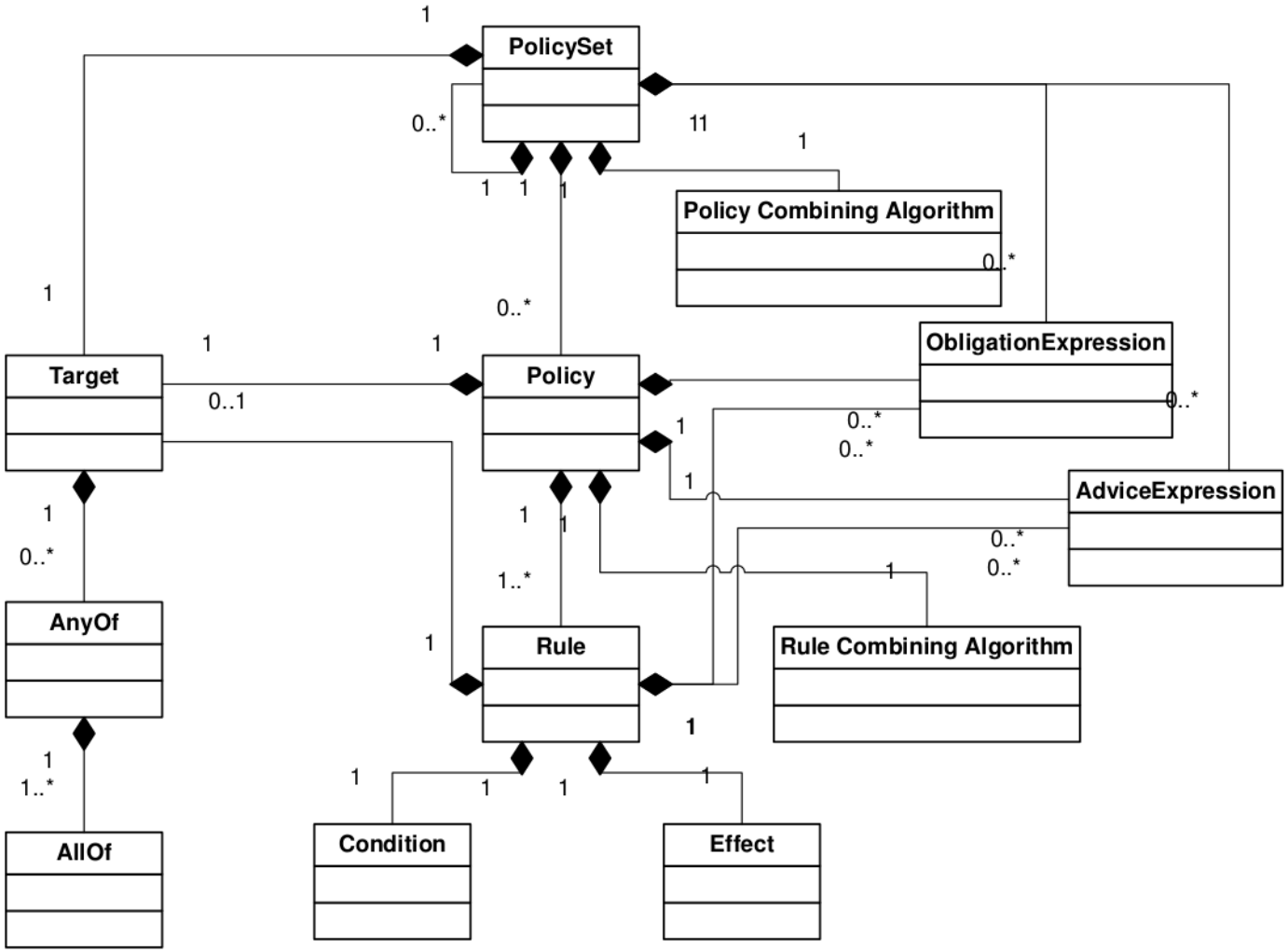
1. Patient Record Policy
   - RP1: only designated patient **can** read his or her patient record except that if the patient is less than 18 years old, the patient's guardian is **permitted** also read the patient's record,
   - RP2: patients **may** only write patient surveys into their own records
   - RP3: both doctors and nurses are **permitted** to read any patient records,
2. Medical Record Policy
   - RM1: doctors **may** only write medical records for their own patients and
   - RM2: **may not** write any other patient records,

Exemple

Un médecin qui tente de lire le dossier d'un patient : sa requête est représentée par des attributs

```
{subject(doctor), action(read), resource(record),
doctor(id,d), patient(id,p), patientRecord(id,p)}
```

Modèle XACML : UML



Modèle XACML : grammaire

Table 1. Abstraction of XACML 3.0 Components

XACML Policy Components		
PolicySet	::= $\langle$Target, $\langle$PolicySet$_1$, ..., PolicySet$_m\rangle, \theta\rangle$	
	| $\langle$Target, $\langle$Policy$_1$, ..., Policy$_m\rangle, \theta\rangle$	where $m \geq 0$
Policy	::= $\langle$Target, $\langle$Rule$_1$, ..., Rule$_m\rangle, \theta\rangle$	where $m \geq 1$
Rule	::= $\langle$*Effect*, Target, Condition$\rangle$	
Condition	::= *propositional formulae*	
Target	::= **Null**	
	| AnyOf$_1 \wedge \ldots \wedge$ AnyOf$_m$	where $m \geq 1$
AnyOf	::= AllOf$_1 \vee \ldots \wedge$ AllOf$_m$	where $m \geq 1$
AllOf	::= Match$_1 \wedge \ldots \wedge$ Match$_m$	where $m \geq 1$
Match	::= $\Phi(\alpha)$	
Φ	::= **subject** | **action** | **resource** | **enviroment**	
α	::= *attribute value*	
θ	::= **p** $-$ **o** | **d** $-$ **o** | **f** $-$ **a** | **o** $-$ **1** $-$ **a**	
Effect	::= **d** | **p**	
XACML Request Component		
Request	::= $\{ A_1, \ldots, A_m \}$	where $m \geq 1$
A	::= $\Phi(\alpha)$ | *external state*	

http://arxiv.org/abs/1110.3706v1

Modèle XACML : *matching* de *target*

$$\mathcal{L}_3 = \{\bot, I, \top\} \text{ avec } \bot \leq I \leq \top$$

$$[\![\mathcal{M}]\!](\mathcal{Q}) = \begin{cases} \top & \mathcal{M} \in \mathcal{Q} \\ \bot & \mathcal{M} \notin \mathcal{Q} \\ I & \text{there is an error during the evaluation} \end{cases}$$

$$[\![\mathcal{A}]\!](\mathcal{Q}) = \bigsqcap_{i=1}^{m} [\![\mathcal{M}_i]\!](\mathcal{Q})$$

$$[\![\mathcal{E}]\!](\mathcal{Q}) = \bigsqcup_{i=1}^{n} [\![\mathcal{A}_i]\!](\mathcal{Q})$$

$$[\![\mathcal{T}]\!](\mathcal{Q}) = \bigsqcap_{i=1}^{o} [\![\mathcal{E}_i]\!](\mathcal{Q})$$

In summary, we can simplify the `Target` evaluation as follows:

$$[\![\mathcal{T}]\!](\mathcal{Q}) = \bigsqcap \bigsqcup \bigsqcap [\![\mathcal{M}]\!](\mathcal{Q})$$

Modèle XACML : évaluation de règles et de politique

De l'applicabilité aux décisions possibles

- Applicable Permit ($\top_p$) : applicable policy to permit an access.
- Applicable Deny ($\top_d$) : applicable policy to deny an access.
- Indeterminate Deny (I_d) : an indeterminate from a policy which *could* have evaluated to deny but not permit, e.g., a `Rule` which evaluates to indeterminate and its effect is deny.
- Indeterminate Permit (I_p) : an indeterminate from a policy which *could* have evaluated to permit but not deny, e.g., a `Rule` which evaluates to indeterminate and its effect is permit.
- Indeterminate Permit (I_{dp}) : an indeterminate from a policy which could have effect either deny or permit.
- Not Applicable ($\bot$).

Modèle XACML : évaluation de règles et de politique

Règle $\langle \star, \mathcal{T}, \mathcal{C} \rangle$: décision, cible, contrainte

$$[\![\mathcal{R}]\!](\mathcal{Q}) = \begin{cases} \top_* & [\![\mathcal{T}]\!](\mathcal{Q}) = \top \text{ and } [\![\mathcal{C}]\!](\mathcal{Q}) = \top \\ \bot & \big([\![\mathcal{T}]\!](\mathcal{Q}) = \top \text{ and } [\![\mathcal{C}]\!](\mathcal{Q}) = \bot\big) \text{ or } [\![\mathcal{T}]\!](\mathcal{Q}) = \bot \\ I_* & \text{otherwise} \end{cases}$$

Policy (resp. *policy sets*) $\langle \mathcal{T}, \mathbb{R}, \theta \rangle$

$$[\![\mathcal{P}]\!](\mathcal{Q}) = \begin{cases} I_* & [\![\mathcal{T}]\!](\mathcal{Q}) = I \text{ and } \bigoplus_\theta(\mathbb{R}') \in \{ \top_*, I_* \} \\ \bot & [\![\mathcal{T}]\!](\mathcal{Q}) = \bot \text{ or} \\ & [\![\mathcal{T}]\!](\mathcal{Q}) = \top \text{ and } \forall \mathcal{R}_i : [\![\mathcal{R}_i]\!](\mathcal{Q}) = \bot \\ \bigoplus_\theta(\mathbb{R}') & \text{otherwise} \end{cases}$$

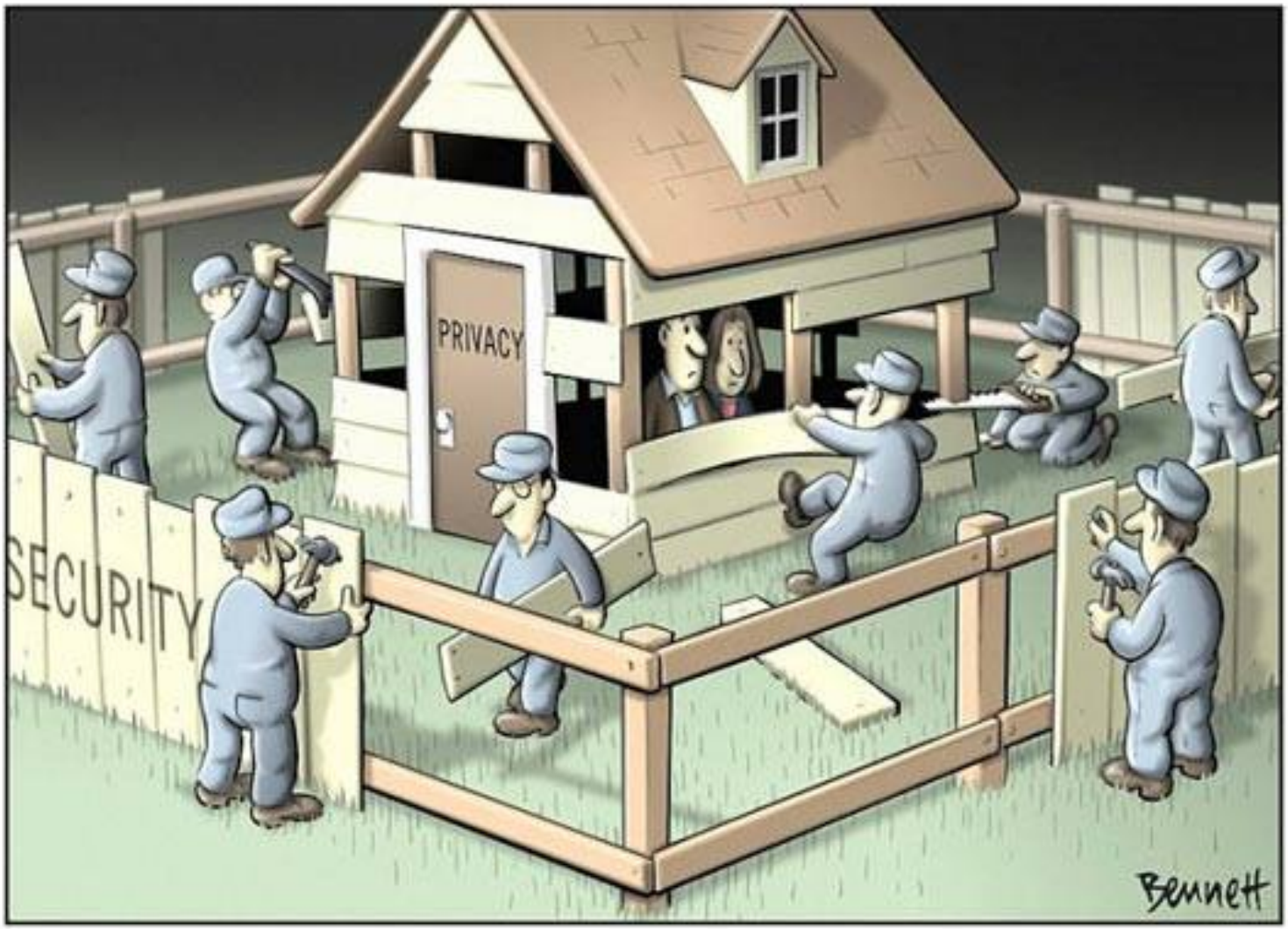
Modèle XACML : combinateurs

Comment combiner plusieurs décisions

- Permit-Overrides / Deny-Overrides
- First-Applicable
- Only-One-Applicable

Exemple : Permit-Overrides

1. If any decision is $\top_p$ then the result is $\top_p$
2. otherwise, if any decision is I_{dp} then the result is I_{dp}
3. otherwise, if any decision is I_p and another decision is I_d or $\top_d$ then the result is I_{dp}
4. otherwise, if any decision is I_p then the result is I_p.
5. otherwise, if any decision is $\top_d$ then the result is $\top_d$
6. otherwise, if any decision is I_d then the result is I_d.
7. otherwise, the result is $\bot$



Security versus privacy